21

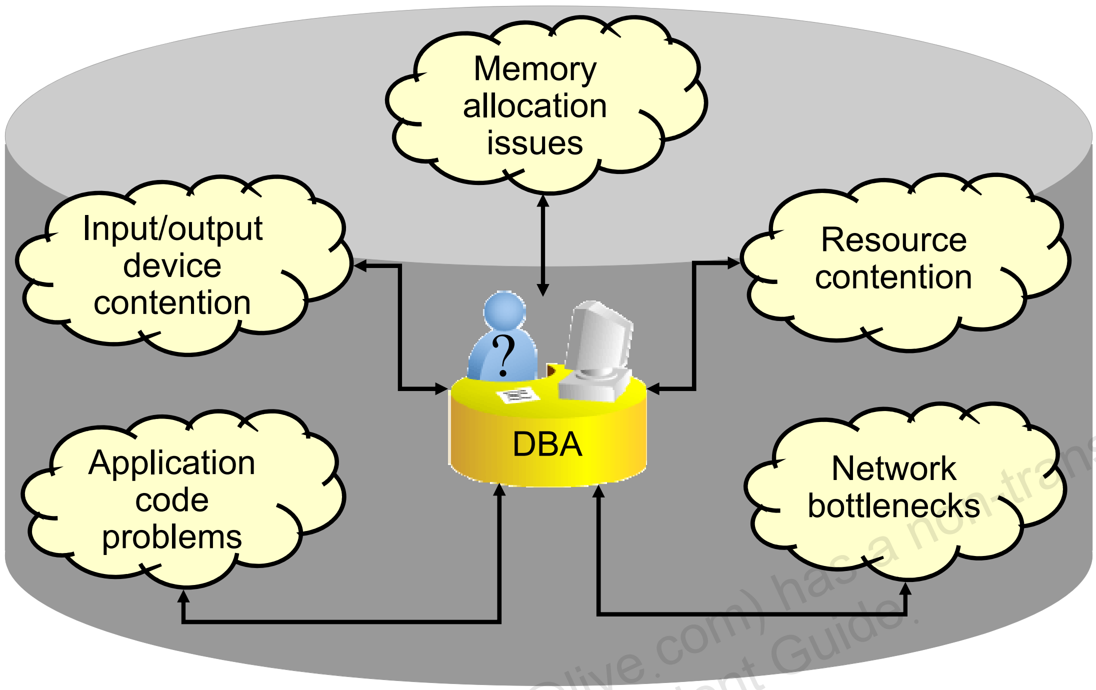
Managing Performance

Objectives

After completing this lesson, you should be able to use:

- Enterprise Manager to monitor performance
- Automatic Memory Management (AMM)
- The Memory Advisor to size memory buffers

Performance Monitoring



To administer Oracle Database and keep it running smoothly, the database administrator (DBA) must regularly monitor its performance to locate bottlenecks and to correct problem areas.

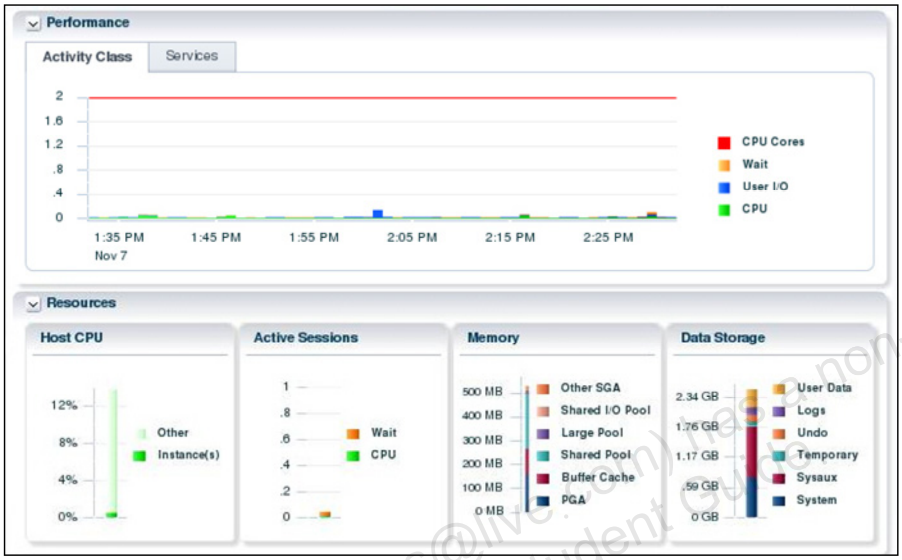
A DBA can look at hundreds of performance measurements, covering everything from network performance and disk input/output (I/O) speed to the time spent working on individual application operations. These performance measurements are commonly referred to as *database metrics*.

Note: For more information about Oracle database performance, see the *Oracle Database 12c: Performance Tuning* course.

Performance Monitoring

Use the Enterprise Manager Database Express home page for:

- Performance overview
- Graphs of metrics and details



You can respond to changes in performance only if you know the performance has changed. Oracle Database provides several ways to monitor the current performance of the database instance. The database home page of Enterprise Manager Database Express provides a quick check of the health of the instance and the server, with graphs showing CPU usage, active sessions, memory and data storage usage. The home page also shows any alerts that have been triggered.

Additional detail is available on the Performance Hub page. This page will be reviewed later in the lesson.

The information displayed in Enterprise Manager is based on performance views that exist in the database. You can access these views directly with SQL*Plus. Occasionally, you may need to access these views for some detail about the raw statistics.

Tuning Activities

The three activities in performance management are:

- Performance planning
- Instance tuning
- SQL tuning

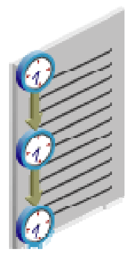
The three facets of tuning involve performance planning, instance tuning, and SQL tuning.

- Performance planning is the process of establishing the environment: the hardware, software, operating system, network infrastructure, and so on.
- Instance tuning is the actual adjustment of Oracle database parameters and operating system (OS) parameters to gain better performance of the Oracle database.
- SQL tuning involves making your application submit efficient SQL statements. SQL tuning is performed for the application as a whole, as well as for individual statements. At the application level, you want to be sure that different parts of the application are taking advantage of each other's work and are not competing for resources unnecessarily.

Note: For more information about performance tuning, refer to the *Oracle Database Performance Tuning Guide*.

Performance Planning

- Investment options
- System architecture
- Scalability
- Application design principles
- Workload testing, modeling, and implementation
- Deploying new applications



There are many facets to performance planning. Planning must include a balance between performance (speed), cost, and reliability. You must consider the investment in your system architecture: the hardware and software infrastructure needed to meet your requirements. This, of course, requires analysis to determine the value for your given environment, application, and performance requirements. For example, the number of hard drives and controllers has an impact on the speed of data access.

The ability of an application to scale is also important. This means that you are able to handle more and more users, clients, sessions, or transactions, without incurring a huge impact on overall system performance. The most obvious violator of scalability is serializing operations among users. If all users go through a single path one at a time, then, as more users are added, there are definitely adverse effects on performance. This is because more and more users line up to go through that path. Poorly written SQL also affects scalability. It requires many users to wait for inefficient SQL to complete; each user competing with the other on a large number of resources that they are not actually in need of.

The principles of application design can greatly affect performance. Simplicity of design, use of views and indexes, and data modeling are all very important.

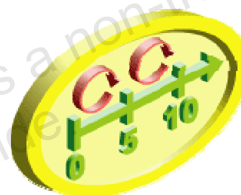
Any application must be tested under a representative production workload. This requires estimating database size and workload, and generating test data and system load.

Performance must be considered as new applications (or new versions of applications) are deployed. Sometimes, design decisions are made to maintain compatibility with old systems during the rollout. A new database should be configured (on the basis of the production environment) specifically for the applications that it hosts.

A difficult and necessary task is testing the existing applications when changing the infrastructure. For example, upgrading the database to a newer version, or changing the operating system or server hardware. Before the application is deployed for production in the new configuration, you want to know the impact. The application will almost certainly require additional tuning. You need to know that the critical functionality will perform, without errors.

Instance Tuning

- Have well-defined goals.
- Allocate memory to database structures.
- Consider I/O requirements in each part of the database.
- Tune the operating system for optimal performance of the database.



At the start of any tuning activity, it is necessary to have specific goals. A goal, such as "Process 500 sales transactions per minute" is easier to work toward than one that says, "Make it go as fast as you can, and we'll know when it's good enough."

You must allocate Oracle database memory suitably for your application to attain optimum performance. You have a finite amount of memory to work with. Too little memory allotted to certain parts of the Oracle database server can cause inefficient background activity, which you may not even be aware of without doing some analysis.

Disk I/O is often the bottleneck of a database and, therefore, requires a lot of attention at the outset of any database implementation.

The operating system configuration can also affect the performance of an Oracle database. For more information, see the *Oracle Database Installation Guide* for your particular platform.

Performance Tuning Methodology

The tuning steps:

- Tune from the top down. Tune the:
  1. Design
  2. Application code
  3. Instance
- Tune the area with the greatest potential benefit. Identify and tune:
  – SQL using the greatest resources
  – The longest waits
  – The largest service times
- Stop tuning when the goal is met.

Oracle has developed a tuning methodology based on years of experience. The basic steps are:

1. Check the OS statistics and general machine health before tuning the instance to be sure that the problem is in the database. Use the Enterprise Manager database home page.
2. Tune from the top down. Start with the design, then the application, and then the instance. For example, try to eliminate the full tables scans causing the I/O contention before tuning the tablespace layout on disk. This activity often requires access to the application code.
3. Tune the area with the greatest potential benefit. The tuning methodology presented in this course is simple. Identify the biggest bottleneck and tune it. Repeat this step. All the various tuning tools have some way to identify the SQL statements, resource contention, or services that are taking the most time. The Oracle database provides a time model and metrics to automate the process of identifying bottlenecks. The Advisors available in Oracle Database use this methodology.
4. Stop tuning when you meet your goal. This step implies that you set tuning goals.

This is a general approach to tuning the database instance and may require multiple passes.

Performance Tuning Data

Type of data gathered:

- Cumulative statistics:
  - Wait events with time information
  - Time model
- Metrics: Statistic rates
- Sampled statistics: Active session history
  - Statistics by session
  - Statistics by SQL
  - Statistics by service
  - Other dimensions

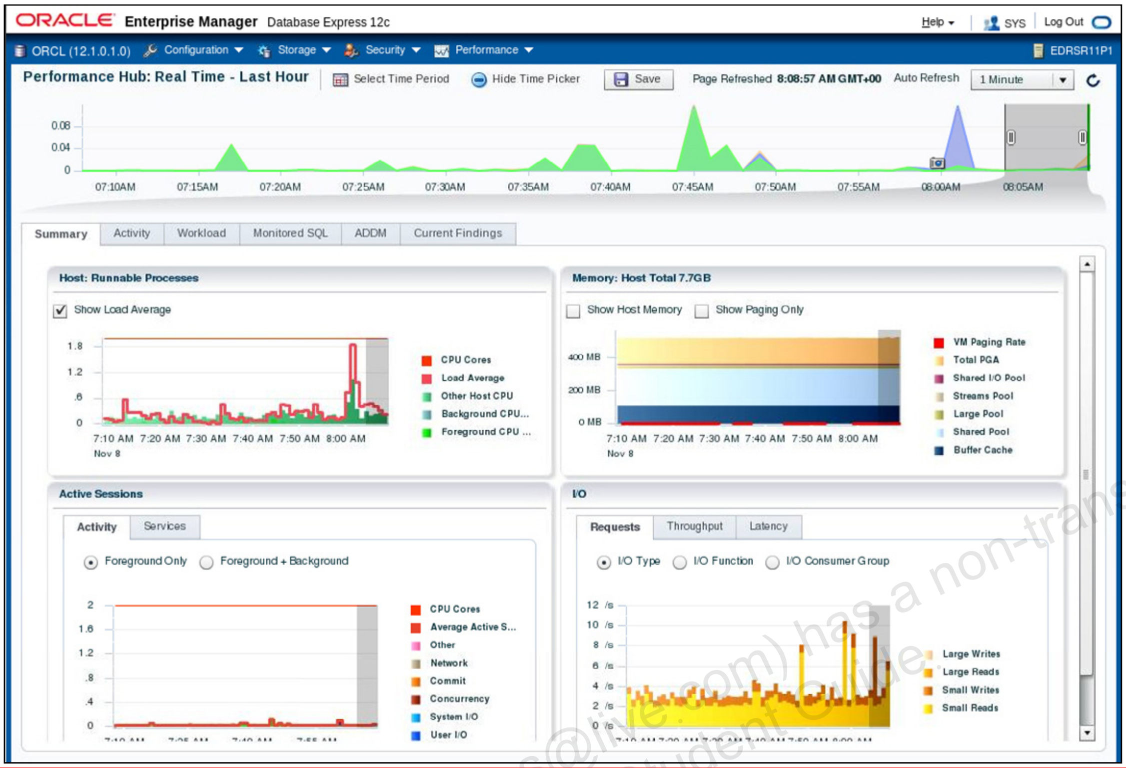
The Oracle database server software captures information about its own operation. Three major types of data are collected: cumulative statistics, metrics, and sampled statistics.

Cumulative statistics are counts and timing information of a variety of events that occur in the database server. Some are quite important, such as buffer busy waits. Others have little impact on tuning, such as index block split. The most important events for tuning are usually the ones showing the greatest cumulative time values. The statistics in Oracle Database are correlated by the use of a time model. The time model statistics are based on a percentage of DB time, giving them a common basis for comparison.

Metrics are statistic counts per unit. The unit could be time (such as seconds), transaction, or session. Metrics provide a base to proactively monitor performance. You can set thresholds on a metric, causing an alert to be generated. For example, you can set thresholds for when the reads per millisecond exceed a previously recorded peak value or when the archive log area is 95% full.

Sampled statistics are gathered automatically when `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`. Sampled statistics allow you to look back in time. You can view session and system statistics that were gathered in the past, in various dimensions, even if you had not thought of specifying data collection for these beforehand.

Using the Enterprise Manager Database Express Performance Hub Page



You can access the Performance Hub page from the Performance menu. On the Performance Hub page, you can view all the performance data available for a specified time period.

You can choose to view real-time data or historical data. More granular data is shown when you choose real-time data. When you view historical data, the data points are averaged out to the Automatic Workload Repository (AWR) interval (usually an hour).

Different tabs are available on the Performance Hub page based on whether you choose real-time or historical data.

The following tabs are available for both real-time and historical data:

- **Summary:** Provides an overall view of the performance of the system for the specified time period
- **RAC:** Appears only when Enterprise Manager Database Express is being used with an Oracle Real Application Clusters (RAC) database or cluster database
- **Activity:** Shows Active Session History (ASH) analytics
- **Workload:** Profile charts show the pattern of user calls, parse calls, redo size, and SQL*Net over the last 60 minutes in real-time mode. The Sessions chart shows the logon rate, current logons, and open cursors.

- **Monitored SQL:** Shows information about monitored SQL statements that were executing or that completed during the selected time period
- **ADDM:** Shows performance findings and recommendations from the Automatic Database Diagnostics Monitor (ADDM) for tasks performed in the database during the selected time period

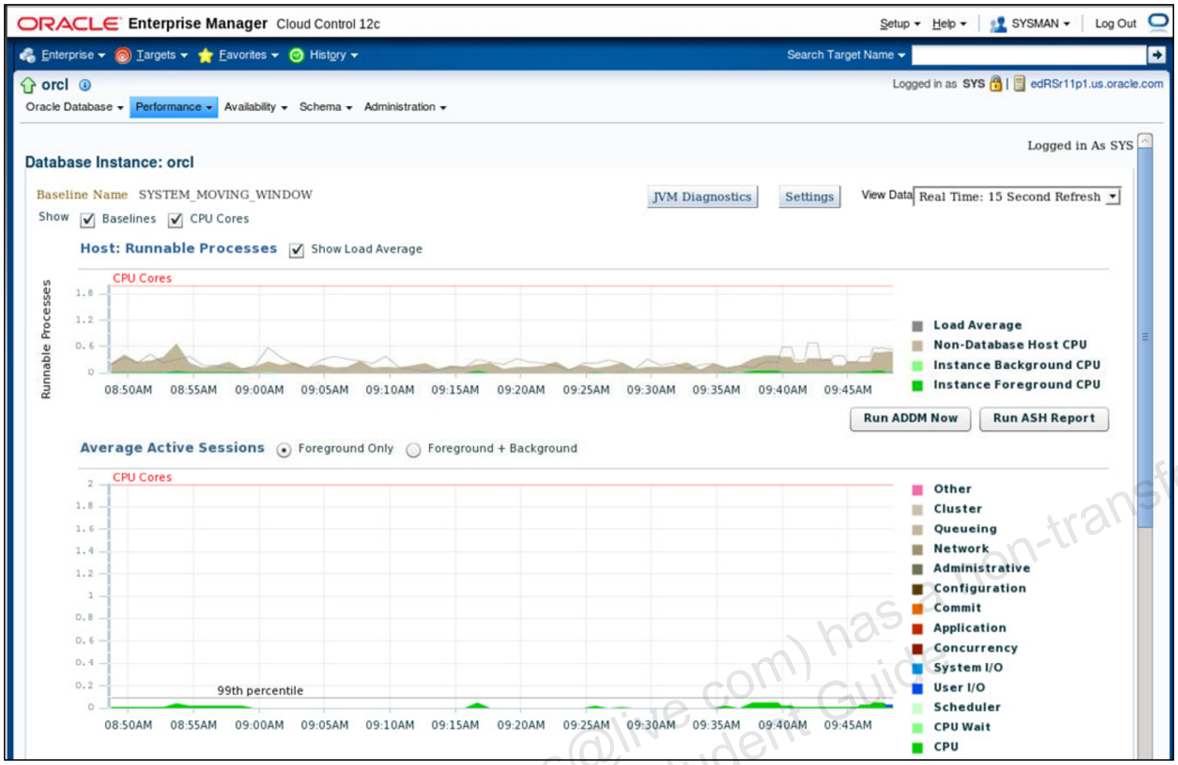
The following tab is only available if you select real-time data:

- **Current Findings:** Shows ADDM findings for the past five minutes

The following tabs are only available if you select historical data:

- **Database Time:** Shows wait events by category for various metrics, and to view time statistics for various metrics for the selected time period
- **Resources:** Shows operating system resource usage statistics, I/O resource usage statistics, and memory usage statistics for the selected time period
- **System Statistics:** Shows database statistics by value, per transaction, or per second for the selected time period

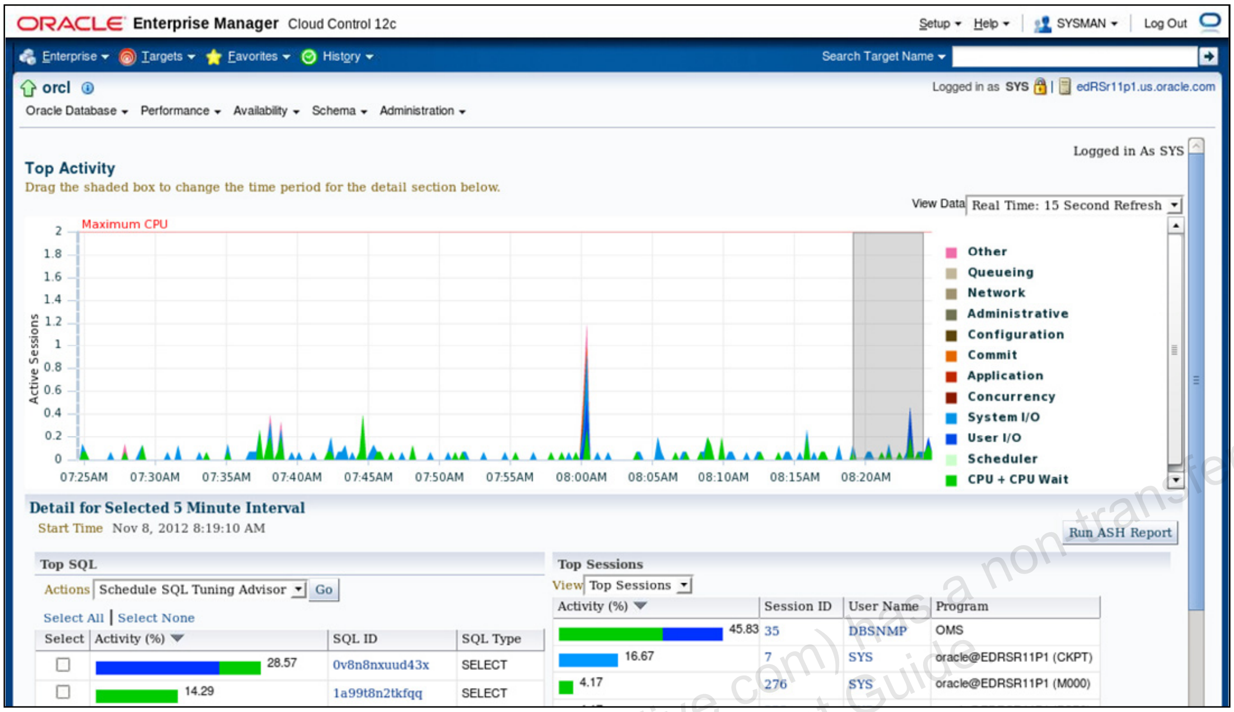
Using the Enterprise Manager Cloud Control Performance Home Page



You can also use Enterprise Manager Cloud Control to manage the performance of your database. Access the Performance Home page by selecting Performance Home in the Performance menu.

The Performance Home page provides an overall view of performance statistics for your database. You can use the information on this page to determine whether resources need to be added or redistributed.

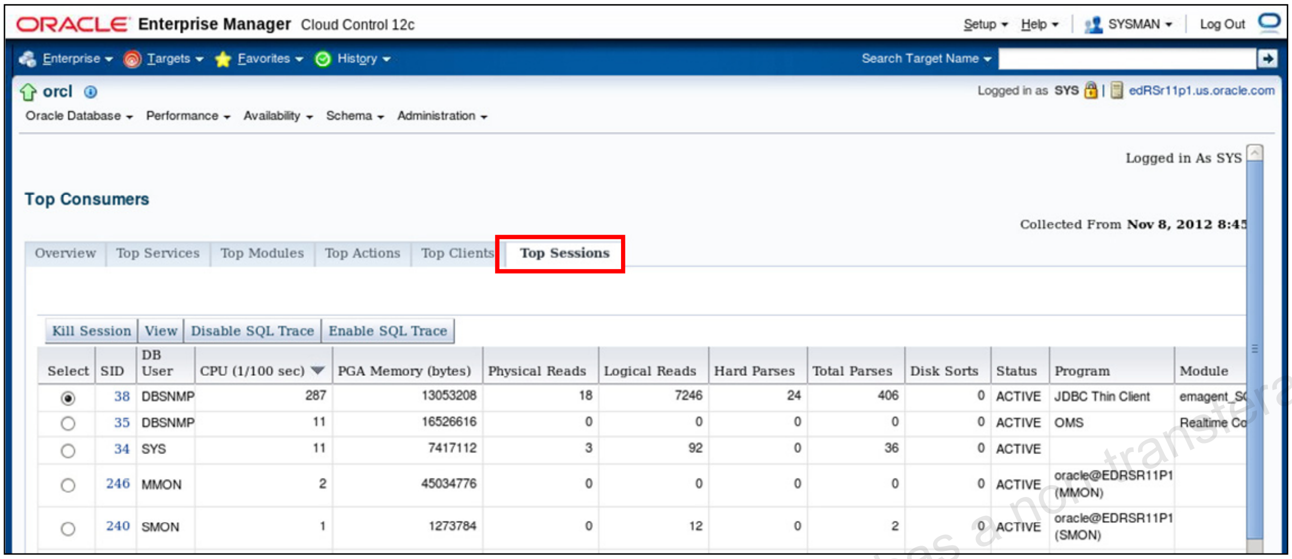
Monitoring Session Performance



Enterprise Manager Cloud Control provides session detail pages so that you can view the wait events occurring in individual sessions. Select Top Activity in the Performance menu to view the summary of all sessions. In the lower right corner of the Top Activity page is a list of the Top Sessions. Click the session identifier to view the Session Details page.

The Top Activity page and Session Details page are based on performance views in the database.

Performance Monitoring: Top Sessions



At the bottom of the Top Activity page, click Top Consumers in the Additional Monitoring Links section to access the Top Consumers page.

The Top Consumers Overview page shows in graphical format:

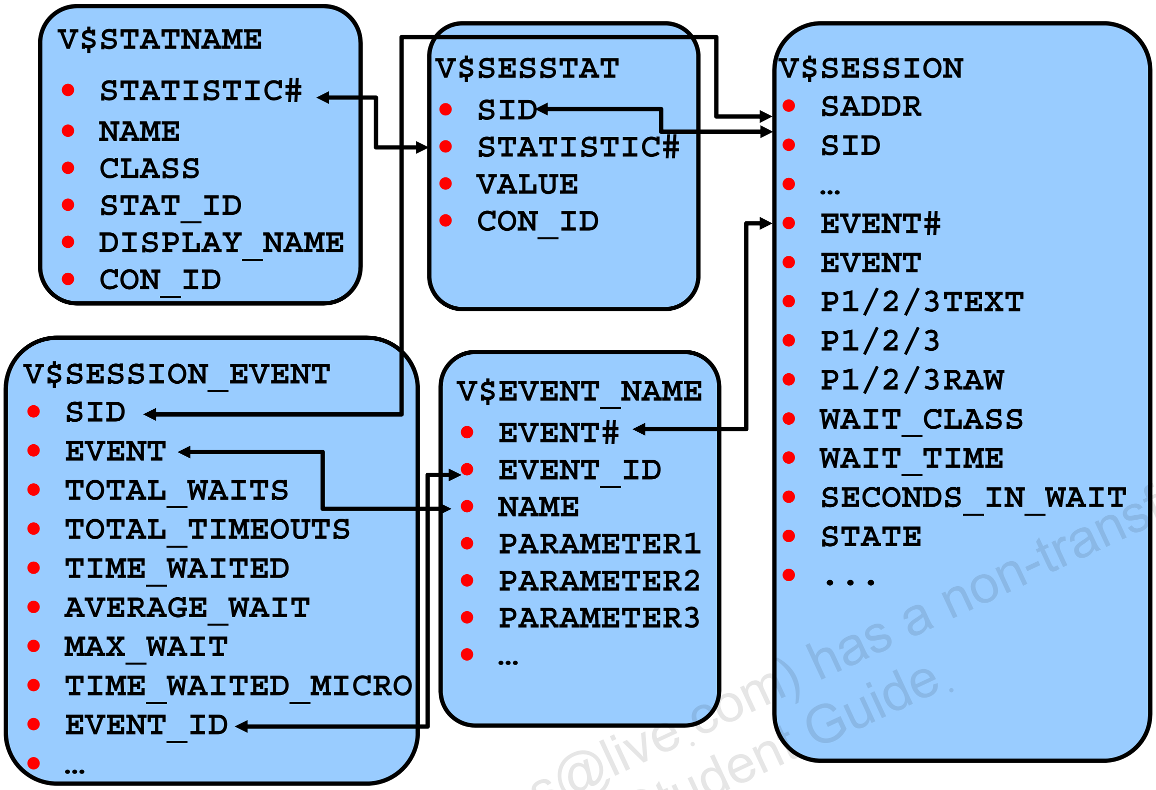
- Top services
- Top modules (by service)
- Top actions (by service and module)
- Top clients

On the Top Consumers page, click the Top Sessions tab to see critical statistics of the sessions using the most resources:

- CPU
- PGA Memory
- Logical Reads
- Physical Read
- Hard Parse count
- Sort count

Click a column name to have the results sorted by the value in that column.

Displaying Session-Related Statistics



You can display current session information for each user logged on by querying V$SESSION. For example, you can use V$SESSION to determine whether a session represents a user session, or was created by a database server process (BACKGROUND).

You can query either V$SESSION or V$SESSION_WAIT to determine the resources or events for which active sessions are waiting.

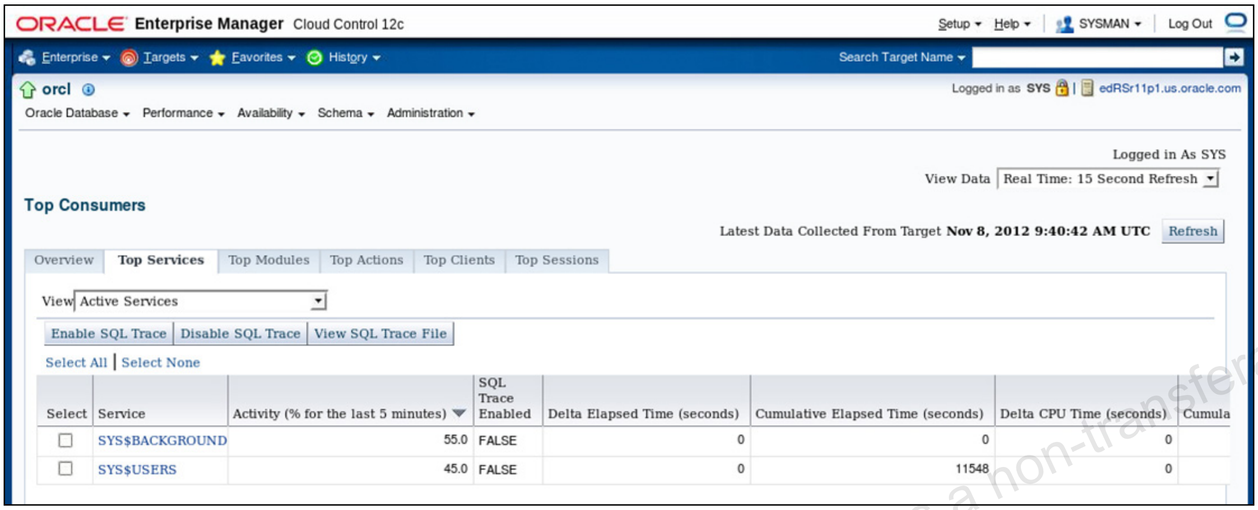
You can view user session statistics in V$SESSTAT. The V$SESSION_EVENT view lists information about waits for an event by a session.

Cumulative values for instance statistics are generally available through dynamic performance views, such as V$SESSTAT and V$SYSSTAT. Note that the cumulative values in dynamic views are reset when the database instance is shut down.

The V$MYSTAT view displays the statistics of the current session.

You can also query V$SESSMETRIC to display the performance metric values for all active sessions. This view lists performance metrics, such as CPU usage, number of physical reads, number of hard parses, and the logical read ratio.

Performance Monitoring: Top Services



In multitier systems where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately. Regardless of the session that was used for a particular request, if it connected via one of these services, the performance data of the session is captured under that service name.

Displaying Service-Related Statistics

For *n*-tier environments, because session statistics are not as helpful, you can see service-level statistics in these views:

- **`V$SERVICE_EVENT`:** Aggregated wait counts and wait times for each service, on a per-event basis
- **`V$SERVICE_WAIT_CLASS`:** Aggregated wait counts and wait times for each service on a wait-class basis

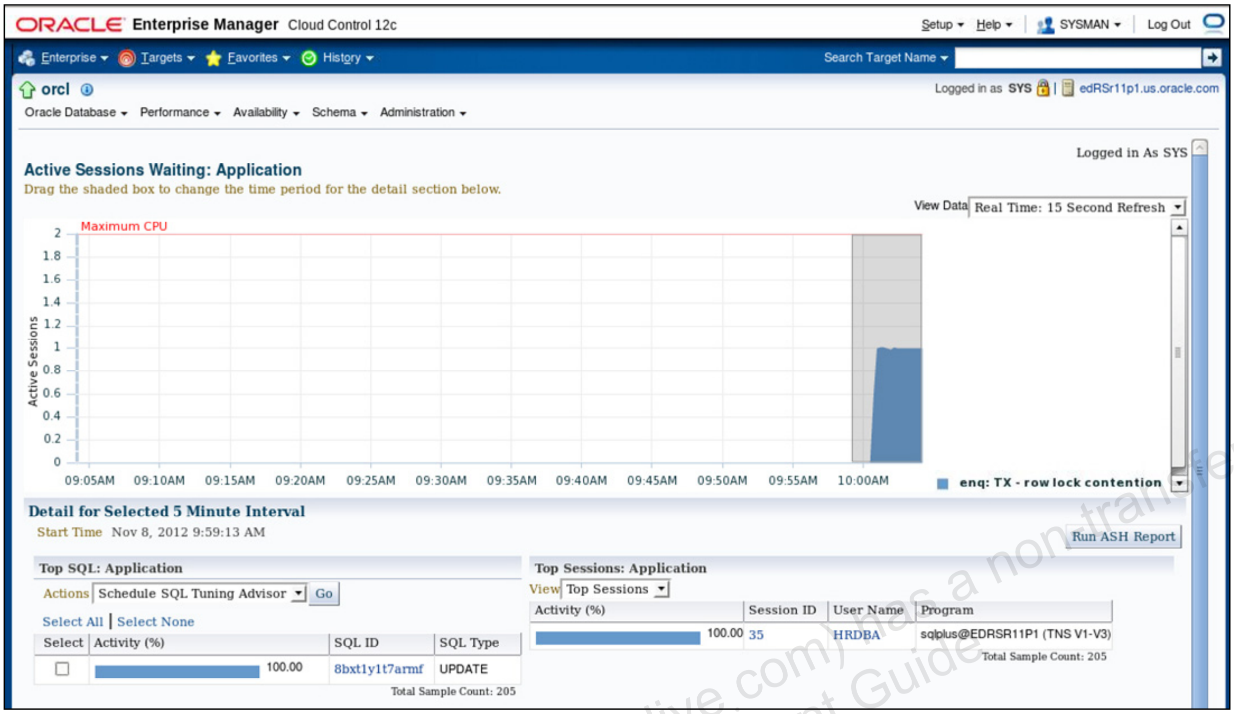
In an *n*-tier environment where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately. These two views provide the same information that their like-named session counterparts provide, except that the information is presented at the service level rather than at the session level.

`V$SERVICE_WAIT_CLASS` shows wait statistics for each service, broken down by wait class.

`V$SERVICE_EVENT` shows the same information as `V$SERVICE_WAIT_CLASS`, except that it is further broken down by event ID.

You can define a service in the database by using the `DBMS_SERVICE` package and use the net service name to assign applications to a service.

Viewing Wait Events



You can access detailed information on waits from the Average Active Sessions graph on the Performance Home page.

When you drill down to a particular wait category, you can view details of specific five-minute intervals and also see the Top Working SQL and the Top Working Sessions associated with that particular wait event during that time. This enables you to perform after-the-fact analysis of system slowdowns and determine potential causes.

Oracle Wait Events

- A collection of wait events provides information about the sessions or processes that had to wait or must wait for different reasons.
- These events are listed in the `V$EVENT_NAME` view.

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 being able to continue processing. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention. Remember that these are only symptoms of problems, not the actual causes.

Wait events are grouped into classes. The wait event classes include: Administrative, Application, Cluster, Commit, Concurrency, Configuration, Idle, Network, Other, Scheduler, System I/O, and User I/O.

There are more than 800 wait events in the Oracle database, including `free buffer wait`, `latch free`, `buffer busy waits`, `db file sequential read`, and `db file scattered read`.

Memory Management: Overview

DBAs must consider memory management to be a crucial part of their job because:

- There is a finite amount of memory available
- Allocating more memory to serve certain types of functions can improve overall performance
- Automatically tuned memory allocation is often the appropriate configuration, but specific environments or even short-term conditions may require further attention

Because there is a finite amount of memory available on a database server and thus, on an Oracle database instance, you must pay attention to how memory is allocated. If too much memory is allowed to be used by a particular area that does not need it, other areas may not function properly because of lack of memory. With the ability to have memory allocation automatically determined and maintained for you, the task is simplified greatly. But even automatically tuned memory needs to be monitored for optimization and may need to be manually configured to some extent.

Managing Memory Components

- Automatic Memory Management (AMM) enables you to specify total memory allocated to instance (including both SGA and PGA)
- Automatic Shared Memory Management (ASMM):
  – Enables you to specify total SGA memory through one initialization parameter
  – Enables the Oracle server to manage the amount of memory allocated to the shared pool, Java pool, buffer cache, streams pool, and large pool
- Manually setting shared memory management:
  – Sizes the components through multiple individual initialization parameters
  – Uses the appropriate Memory Advisor to make recommendations

Oracle Database enables you to specify the total memory allocated to the instance. Memory will be dynamically reallocated between the System Global Area (SGA) and Program Global Area (PGA) as needed. This method is called Automatic Memory Management (AMM) and is available on only those platforms that support dynamic release of memory. This simplifies your memory management tasks.

Memory advisors are available to help you set the initialization parameters on various levels. The advisor available depends on the level on which you are specifying the memory parameters. If you enable AMM, only the Memory Size Advisor is available.

Automatic Shared Memory Management (ASMM) enables you to manage the SGA as a whole. The SGA comprises several components. The sizes of many of these components are dynamically adjusted for best performance within the limits of the initialization parameters. When the AMM is enabled, the ASMM is automatically enabled. If the ASMM is enabled but not the AMM, the SGA Size Advisor is available.

You can manage the size of individual components manually by setting the initialization parameter for each component. If the Oracle server notifies you of a performance problem that is related to the size of an SGA or PGA component, you can use the Memory Advisor for the component to determine appropriate new settings. The Memory Advisor can model the effect of parameter changes.

Efficient Memory Usage: Guidelines

- Fit the SGA into physical memory.
- Use the Memory Advisors.
- Tune for the most efficient use of memory
  – Reduce overall physical I/O
  – Reduce the total memory needs

If possible, it is best to fit the SGA into physical memory, which provides the fastest access. Even though the OS may provide additional virtual memory, that memory, by its nature, can often be swapped out to disk. On some platforms, you can use the `LOCK_SGA` initialization parameter to lock the SGA into physical memory. This parameter cannot be used in conjunction with AMM or ASMM.

When a SQL statement executes, data blocks are requested for reading or writing, or both. This is considered a logical I/O. As the block is requested, the block is checked to see whether it already exists in memory. If it is not in memory, it is read from disk, which is called a physical I/O. When the block is found in memory, the cost is several orders of magnitude less than the cost of reading the block from disk. The size of the SGA components in combination with the workload has a large affect on the number of physical reads. A simple view of this implies that you should increase the memory for the SGA components as much as possible. A larger SGA is not always better. There is a point where adding more memory yields diminishing returns. This principle applies to the buffer cache, the shared pool, and other SGA components. In practice, you find that shifting memory from one SGA component to another may increase overall performance, without changing the total amount of memory given to the SGA depending on the characteristics of the workload.

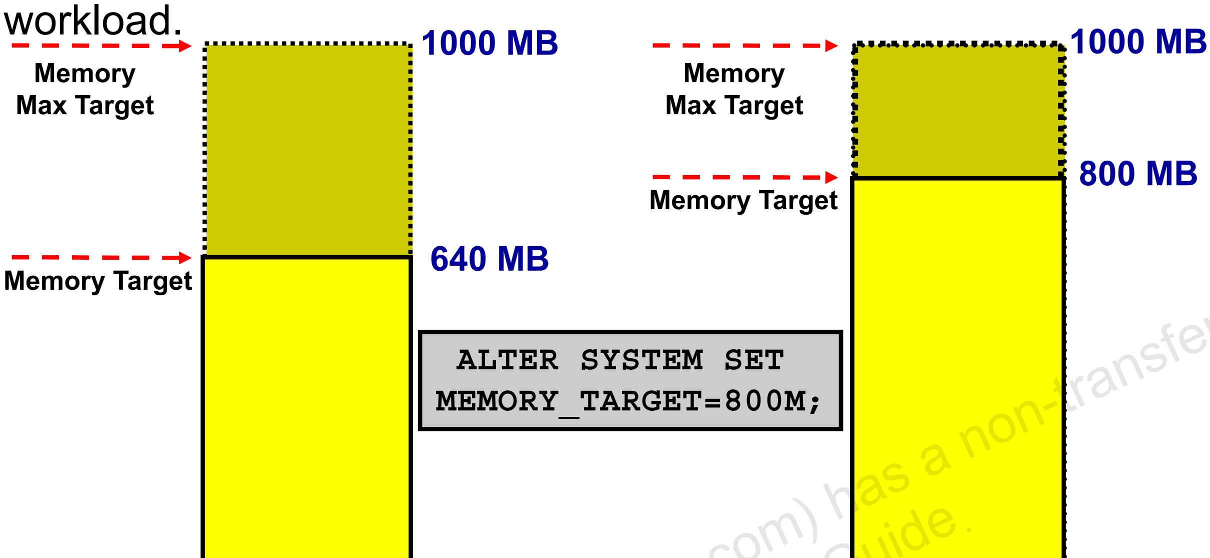
Memory has an upper limit in all current machines. That limit may be imposed by the hardware, operating system, or the cost of the memory. The goal of memory tuning is to produce the most efficient use of existing memory. When the workload changes often, the most efficient division of memory between the SGA components will change. There is also the amount of memory allocated to SGA and PGA. Online transaction processing (OLTP) systems typically use very little PGA memory compared to data warehouse (DW) or decision support systems (DSS).

Enterprise Manager Cloud Control and Enterprise Manager Database Express both provide Memory Advisors. These tools monitor the memory usage by the SGA components, and PGA, and project the differences in terms of efficiency for increased and decreased memory allocations. These projections use the current workload. They can help you size the SGA on the basis of the activity in your particular database. The advisors will make sizing recommendations for manual settings, when the automatic memory management is disabled. These same advisors provide input to the automatic memory management, to determine the most efficient component sizes.

Using the existing memory efficiently also includes tuning the applications. A poorly tuned application can use large quantities of memory. For example, an application that uses frequent full table scans because indexes do not exist or are unusable can cause a large amount of I/O, reducing performance. The first and most effective tuning technique is to tune high cost SQL statements. Tuning SQL statements is covered in more detail in the lesson titled “*Managing Performance: SQL Tuning.*”

Automatic Memory Management: Overview

With Automatic Memory Management, the database server can size the SGA and PGA automatically according to your workload.



Oracle recommends the use of AMM unless you have special requirements.

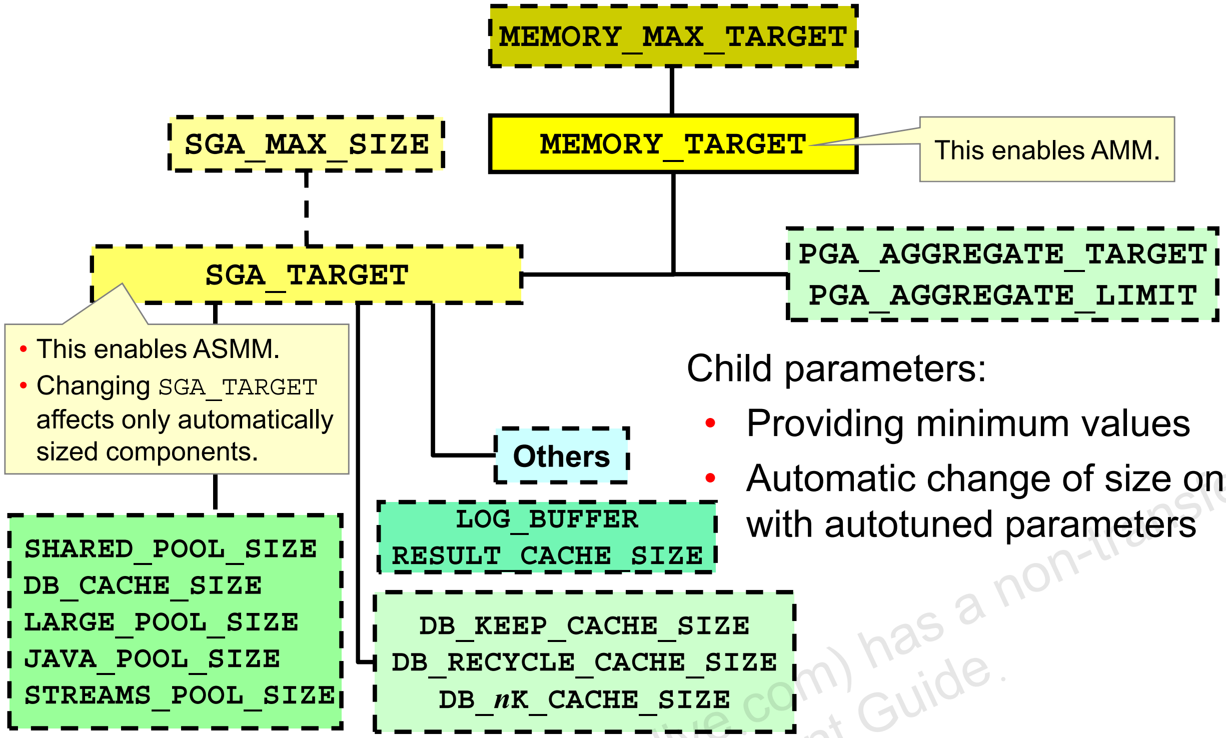
Automatic Memory Management (AMM) allows the Oracle Database server to manage SGA memory and instance PGA memory sizing automatically. To do so (on most platforms), you set only a target memory size initialization parameter (`MEMORY_TARGET`) and a maximum memory size initialization parameter (`MEMORY_MAX_TARGET`), and the database server dynamically exchanges memory between the SGA and the instance PGA as needed to meet processing demands.

With this memory management method, the database server also dynamically tunes the sizes of the individual SGA components and the sizes of the individual PGAs.

Because the target memory initialization parameter is dynamic, you can change the target memory size at any time without restarting the database instance. The maximum memory size serves as an upper limit so that you cannot accidentally set the target memory size too high. Because certain SGA components either cannot easily shrink or must remain at a minimum size, the database server also prevents you from setting the target memory size too low.

This indirect memory transfer relies on the operating system (OS) mechanism of freeing shared memory. After memory is released to the OS, the other components can allocate memory by requesting memory from the OS. Currently, Automatic Memory Management is implemented on Linux, Solaris, HPUX, AIX, and Windows.

Oracle Database Memory Parameters

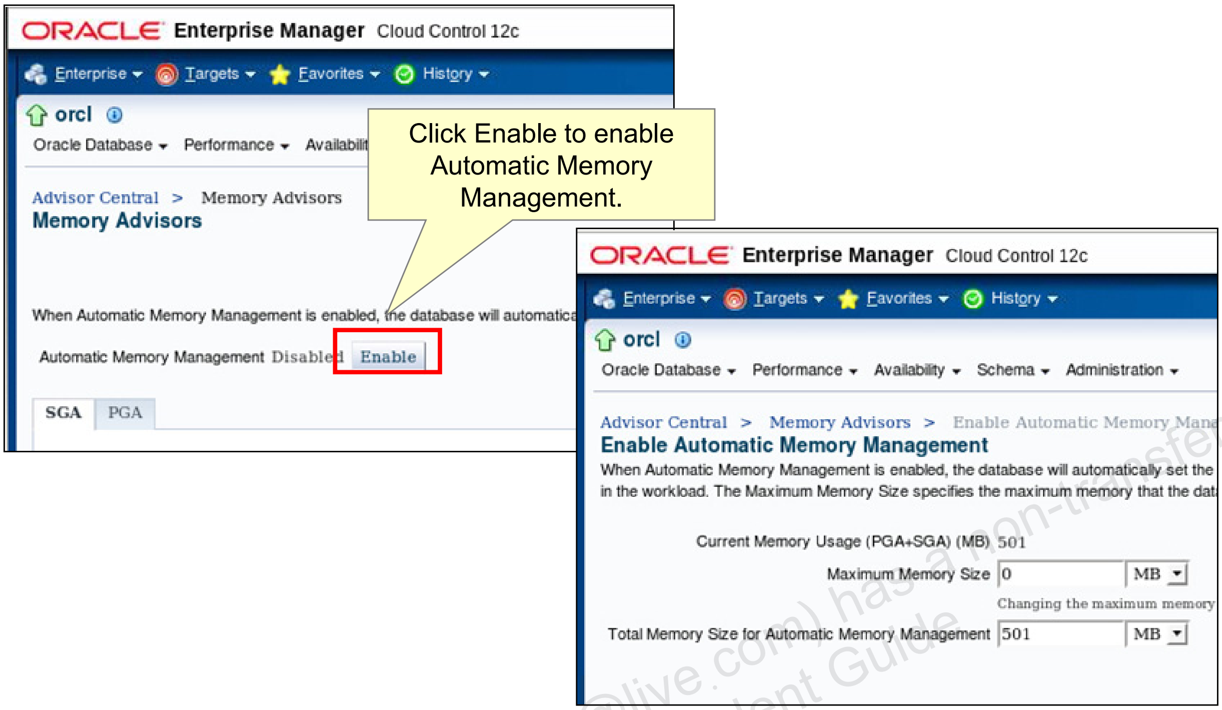


Child parameters:
- Providing minimum values
- Automatic change of size only with autotuned parameters

The graphic in the slide shows you the memory initialization parameters hierarchy. Although you have to set only MEMORY_TARGET to trigger Automatic Memory Management, you still have the ability to set lower bound values for various caches. Therefore, if the child parameters are user-set, they will be the minimum values below which the Oracle database server will not autotune that component.

- If SGA_TARGET and PGA_AGGREGATE_TARGET are set to a nonzero value, they are considered to be the minimum values for the sizes of the SGA and the PGA, respectively. MEMORY_TARGET can take values from SGA_TARGET + PGA_AGGREGATE_TARGET to MEMORY_MAX_SIZE.
- If SGA_TARGET is set, the database server autotunes only the sizes of the subcomponents of the SGA. PGA is autotuned independent of whether it is explicitly set or not. However, the whole SGA (SGA_TARGET) and the PGA (PGA_AGGREGATE_TARGET) are not autotuned—that is, do not grow or shrink automatically.

Enabling Automatic Memory Management (AMM) by Using Enterprise Manager Cloud Control



If you did not enable Automatic Memory Management (AMM) when you configured your database, you can enable it by performing the following steps:

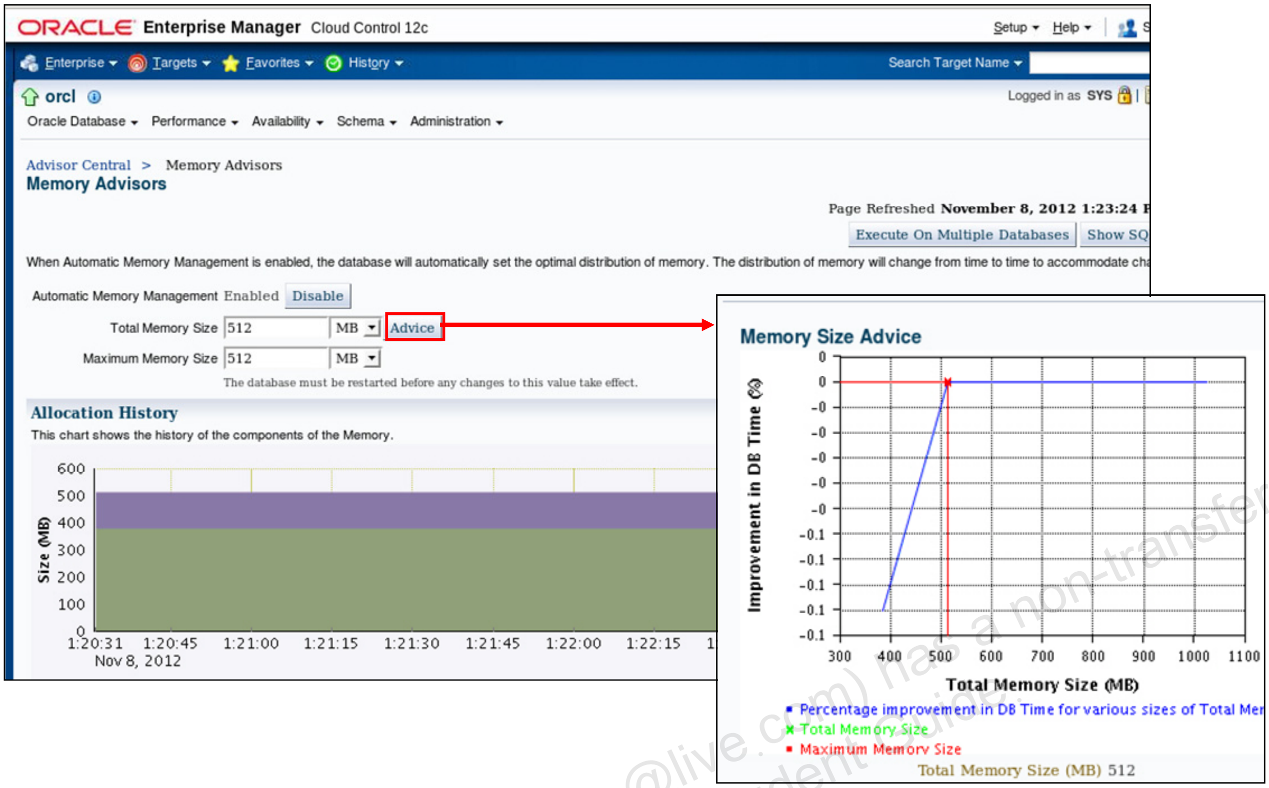
1. Select Memory Advisors in the Performance menu.
2. Click Enable for Automatic Memory Management.
3. The Enable Automatic Memory Management page appears. Set the values for Total Memory Size and Maximum Memory Size for Automatic Memory Management.
   Note: If you change the Maximum Memory Size, the database instance must be restarted.
4. Click OK.

You can increase the size at a later time by increasing the value of the Total Memory Size field or the `MEMORY_TARGET` initialization parameter. However, you cannot set it higher than the value specified by the Maximum Memory Size field or the `MEMORY_MAX_TARGET` parameter. For more information, see the *Oracle Database Administrator's Guide*.

After AMM is enabled, the Memory Size Advisor is available to help you adjust the maximum and target memory sizes.

Note: Oracle recommends that you use Automatic Memory Management to simplify memory management tasks.

Monitoring Automatic Memory Management



After Automatic Memory Management is enabled, you can see the graphical representation of the history of your memory size components in the Allocation History section of the Memory Advisors page. The top portion in the first histogram is tunable PGA only and the lower portion is all of SGA.

Additional data is available on the page showing the SGA components history and a graphical representation of the SGA allocation.

On this page, you can also access the memory target advisor by clicking the Advice button. This advisor gives you the possible DB time improvement for various total memory sizes.

Note: You can also look at the memory target advisor by using the `V$MEMORY_TARGET_ADVISOR` view.

Monitoring Automatic Memory Management

Use the following views to monitor Automatic Memory Management:

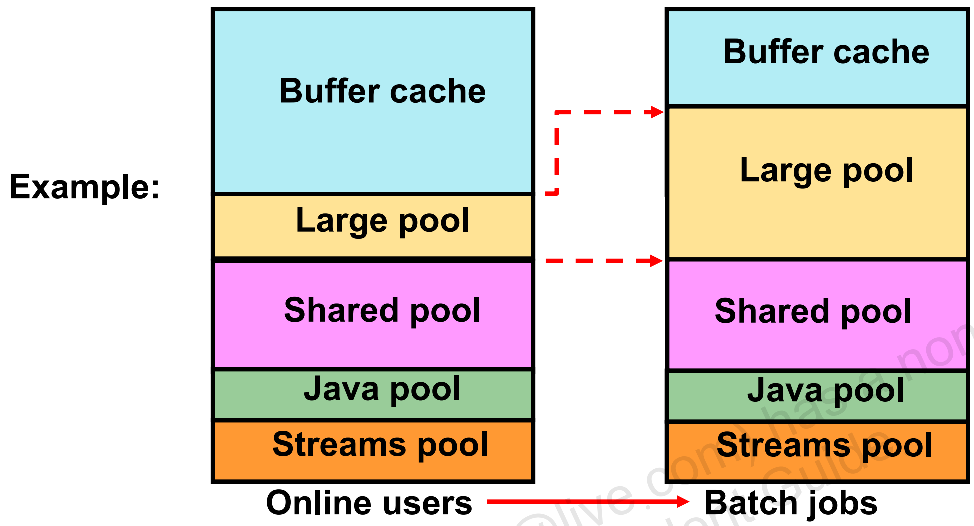
- V$MEMORY_DYNAMIC_COMPONENTS: Current status of all memory components
- V$MEMORY_RESIZE_OPS: Circular history buffer of the last 800 memory resize requests
- V$MEMORY_TARGET_ADVICE: Tuning advice for the MEMORY_TARGET initialization parameter

The V$MEMORY_DYNAMIC_COMPONENTS dynamic performance view shows the current sizes of all dynamically tuned memory components, including the total sizes of the SGA and instance PGA. The V$MEMORY_TARGET_ADVICE view provides tuning advice for the MEMORY_TARGET initialization parameter.

In the V$MEMORY_TARGET_ADVICE view, the row with the MEMORY_SIZE_FACTOR of 1 shows the current size of memory, as set by the MEMORY_TARGET initialization parameter, and the amount of DB time required to complete the current workload. In previous and subsequent rows, the results show several alternative MEMORY_TARGET sizes. For each alternative size, the database server shows the size factor (the multiple of the current size), and the estimated DB time to complete the current workload if the MEMORY_TARGET parameter were changed to the alternative size. Notice that for a total memory size smaller than the current MEMORY_TARGET size, the estimated DB time increases.

Automatic Shared Memory Management: Overview

- Automatically adapts to workload changes
- Maximizes memory utilization
- Helps eliminate out-of-memory errors



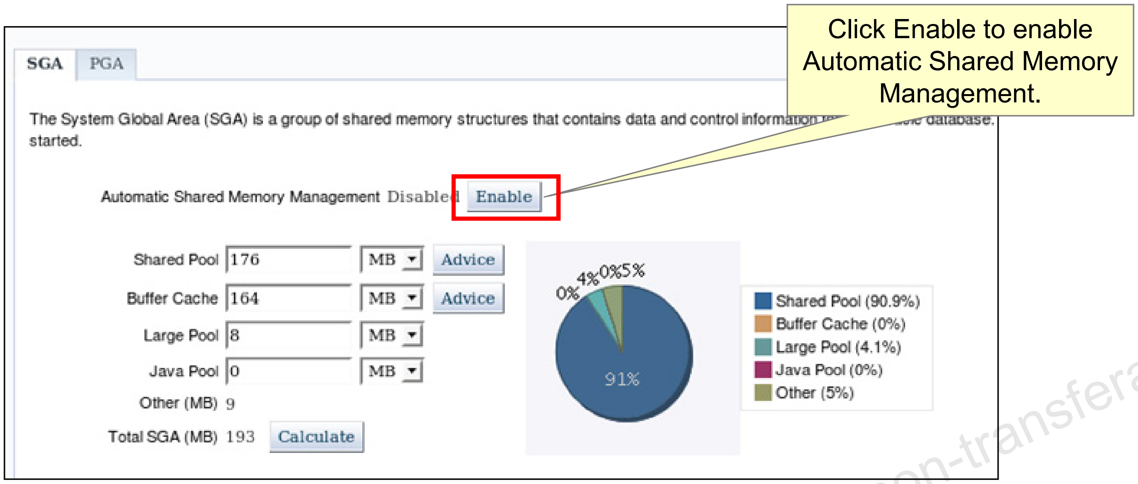
If AMM does not work for you, because you need a fixed PGA, consider the use of Automatic Shared Memory Management (ASMM) which simplifies SGA memory management. You specify the total amount of SGA memory available to an instance by using the `SGA_TARGET` initialization parameter and Oracle Database server automatically distributes this memory among the various SGA components to ensure the most effective memory utilization.

For example, in a system that runs large online transactional processing (OLTP) jobs during the day (requiring a large buffer cache) and runs parallel batch jobs at night (requiring a large value for the large pool), you would have to simultaneously configure both the buffer cache and the large pool to accommodate your peak requirements.

With ASMM, when the OLTP job runs, the buffer cache uses most of the memory to allow for good I/O performance. When the data analysis and reporting batch job starts up later, the memory is automatically migrated to the large pool so that it can be used by parallel query operations without producing memory overflow errors.

The Oracle Database server remembers the sizes of the automatically tuned components across instance shutdowns if you are using a server parameter file (SPFILE). As a result, the system does need to learn the characteristics of the workload again each time an instance is started. It can begin with information from the past instance and continue evaluating workload where it left off at the last shutdown.

Enabling Automatic Shared Memory Management (ASMM)



Automatic Shared Memory Management is automatically enabled if you have enabled AMM. If you have not enabled AMM or did not enable ASMM when you configured your database, you can enable Automatic Shared Memory Management by performing the following steps:

1. Select Memory Advisors in the Performance menu.
2. The Memory Advisors page appears. Scroll down to the SGA section. Click Enable for Automatic Shared Memory Management.
3. The Enable Automatic Shared Memory Management page appears. Specify the total SGA size. Click OK.

You can increase the total SGA size at a later time by increasing the value of the Total SGA Size field or the `SGA_TARGET` initialization parameter. However, you cannot set it higher than the value specified by the Maximum SGA Size field or the `SGA_MAX_SIZE` parameter. For more information, see the *Oracle Database Administrator's Guide*.

When AMM is disabled, the PGA advisor is accessible. The PGA advisor is recommended for setting the PGA memory value. Click the PGA tab to access the PGA property page. Click Advice to invoke the PGA Advisor.

Note: Oracle recommends that you use Automatic Shared Memory Management to simplify your memory management tasks.

Understanding Automatic Shared Memory Management

- ASMM is based on workload information that `MMON` captures in the background.
- `MMON` uses memory advisors.
- Memory is moved to where it is needed the most by `MMAN`.
- If an SPFILE is used (which is recommended):
  - Component sizes are saved across shutdowns
  - Saved values are used to bootstrap component sizes
  - There is no need to relearn optimal values

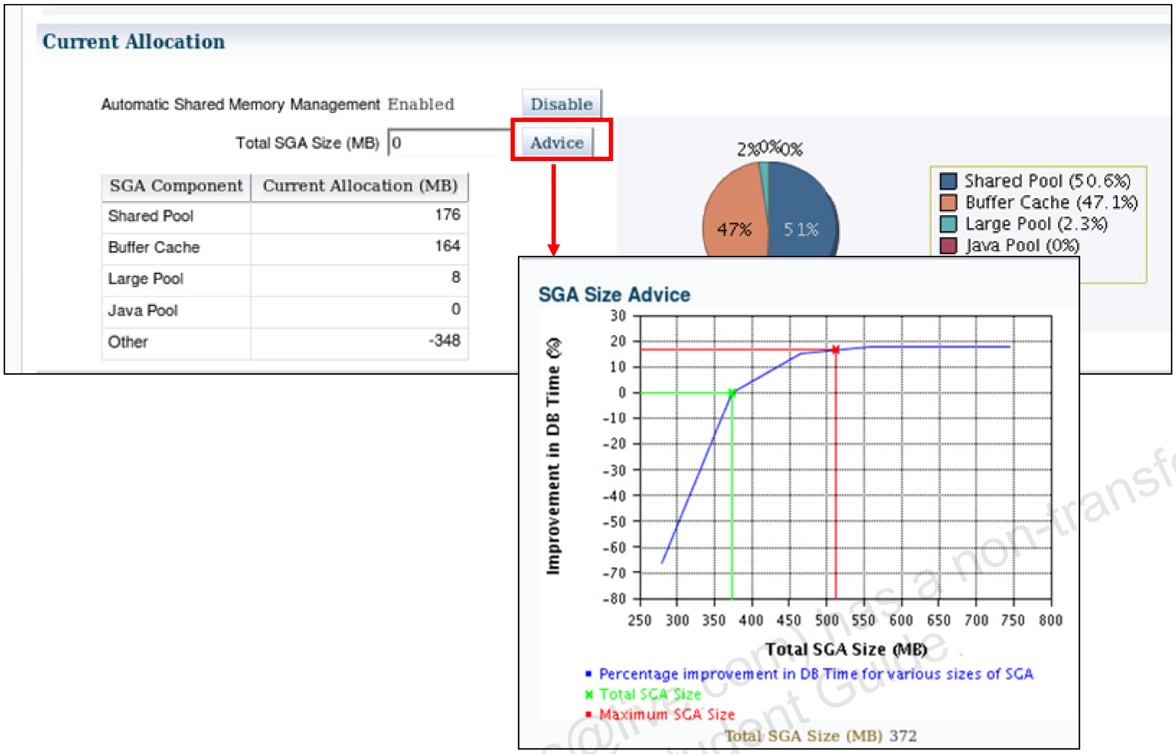
The Automatic Shared Memory Management feature uses the SGA memory broker that is implemented by two background processes: Manageability Monitor (`MMON`) and Memory Manager (`MMAN`). Statistics and memory advisory data are periodically captured in memory by `MMON`. `MMAN` coordinates the sizing of the memory components according to `MMON` decisions. The SGA memory broker keeps track of the sizes of the components and pending resize operations.

The SGA memory broker observes the system and workload in order to determine the ideal distribution of memory. It performs this check every few minutes so that memory can always be present where needed. In the absence of Automatic Shared Memory Management, components had to be sized to anticipate their individual worst-case memory requirements.

On the basis of workload information, Automatic Shared Memory Management:

- Captures statistics periodically in the background
- Uses memory advisors
- Performs what-if analysis to determine the best distribution of the memory
- Moves memory to where it is most needed
- Saves component sizes across shutdown if an SPFILE is used (the sizes can be resurrected from before the last shutdown)

Automatic Shared Memory Advisor



Before enabling ASMM, you should remove the individual memory area parameters from the SPFILE because having them set can impose restrictions on ASMM. After ASMM is enabled, the SGA Size Advisor is available to help you choose the best value for total SGA size.

If after seeing the effects of the ASMM allocations you decide that you want to adjust certain component allocations, you can specify values for those components. If the values you are specifying are lower than the current values, those values are treated as minimum memory sizes for their respective components. If the values you are specifying are larger than the current values, the sizes of the memory components are resized upward to the values you provided as long as free memory is available. Setting limits reduces the amount of memory available for automatic adjustment, but the capability is available and can help overall performance.

The initialization parameters of concern are the following:

- `SHARED_POOL_SIZE`
- `LARGE_POOL_SIZE`
- `JAVA_POOL_SIZE`
- `DB_CACHE_SIZE`
- `STREAMS_POOL_SIZE`

To adjust these parameters while ASMM is enabled, you must use the `ALTER SYSTEM` command.

Enabling Automatic Shared Memory Management

To enable ASMM from manual shared memory management:

1. Get a value for **SGA_TARGET**:

```
SELECT ((SELECT SUM(value) FROM V$SGA) - (SELECT CURRENT_SIZE
FROM V$SGA_DYNAMIC_FREE_MEMORY)) "SGA_TARGET" FROM DUAL;
```

2. Use that value to set **SGA_TARGET**.
3. Set the values of the automatically sized SGA components to 0.

To switch to ASMM from Automatic Memory Management:

1. Set the **MEMORY_TARGET** initialization parameter to 0.
2. Set the values of the automatically sized SGA components to 0.

The procedure for enabling ASMM differs depending on whether you are changing to ASMM from manual shared memory management or from automatic memory management. To change to ASMM from manual shared memory management:

1. Execute the following query to obtain a value for SGA_TARGET:

```
SELECT ((SELECT SUM(value) FROM V$SGA) - (SELECT CURRENT_SIZE
  FROM V$SGA_DYNAMIC_FREE_MEMORY)) "SGA_TARGET" FROM DUAL;
```

2. Set the value of SGA_TARGET:

```
ALTER SYSTEM SET SGA_TARGET=value [SCOPE={SPFILE|MEMORY|BOTH}]
```

   where *value* is the value computed in step 1 or is some value between the sum of all SGA component sizes and SGA_MAX_SIZE.
3. Set the values of the automatically sized SGA components to 0. Do this by editing the text initialization parameter file or by issuing ALTER SYSTEM statements. Restart the instance if required.

To change to ASMM from automatic memory management:

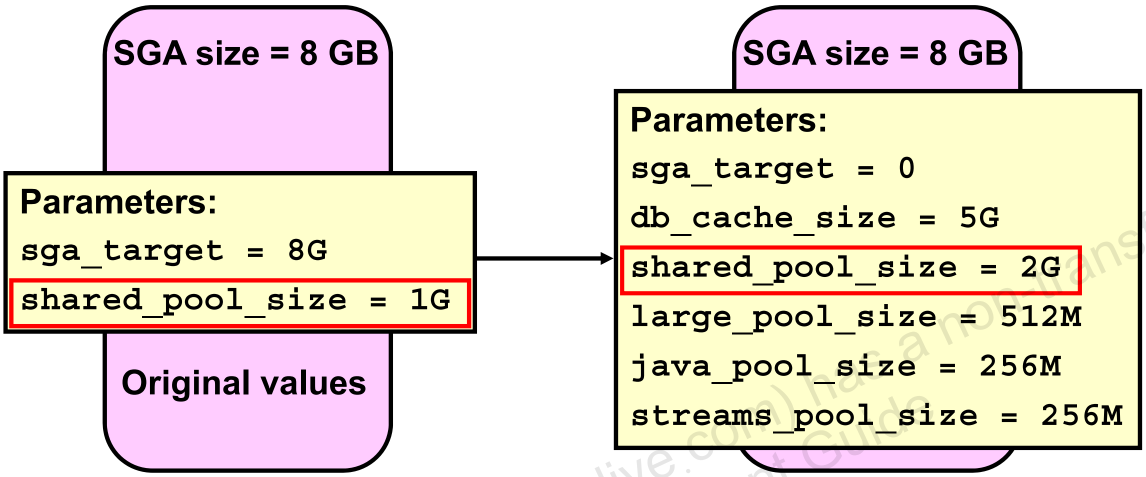
1. Set the MEMORY_TARGET initialization parameter to 0.

```
ALTER SYSTEM SET MEMORY_TARGET = 0;
```

   The database sets SGA_TARGET based on current SGA memory allocation.
2. Set the values of the automatically sized SGA components to 0. Restart the instance when finished.

Disabling Automatic Shared Memory Management

- Setting `SGA_TARGET` to 0 disables autotuning.
- Autotuned parameters are set to their current sizes.
- The SGA size as a whole is unaffected.



You can dynamically choose to disable Automatic Shared Memory Management by setting `SGA_TARGET` to 0. In this case, the values of all the autotuned parameters are set to the current sizes of the corresponding components, even if the user had earlier specified a different nonzero value for an autotuned parameter.

In the example in the slide, the value of `SGA_TARGET` is 8 GB and the value of `SHARED_POOL_SIZE` is 1 GB. If the system has internally adjusted the size of the shared pool component to 2 GB, setting `SGA_TARGET` to 0 results in `SHARED_POOL_SIZE` being set to 2 GB, thereby overriding the original user-specified value.

Using V$PARAMETER to View Memory Component Sizes

```
SGA_TARGET = 8G
```

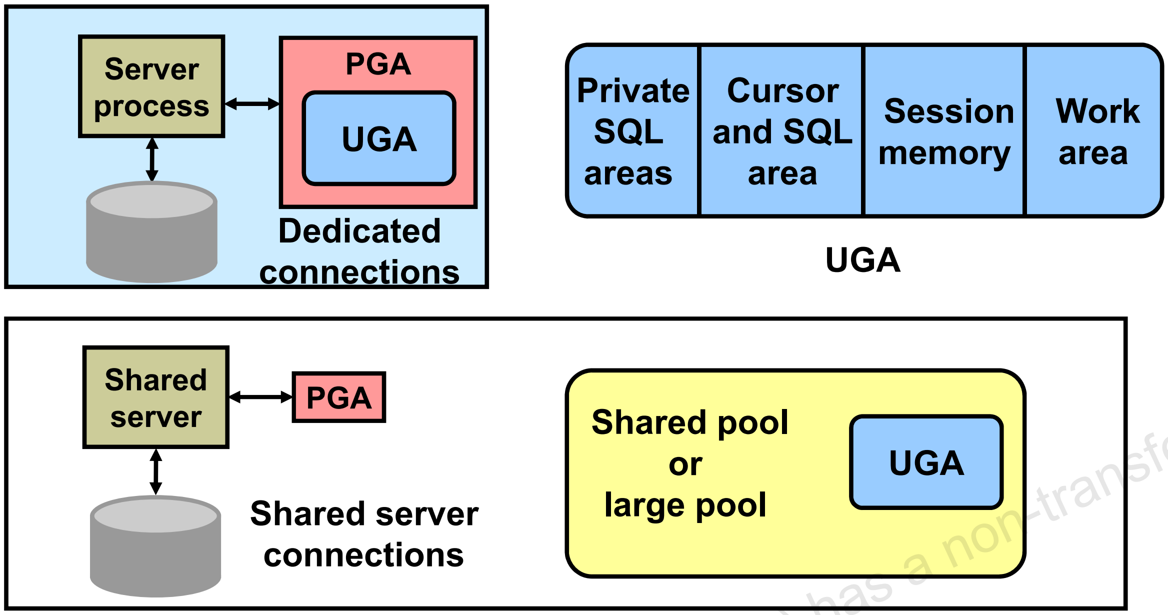
```
DB_CACHE_SIZE = 0
JAVA_POOL_SIZE = 0
LARGE_POOL_SIZE = 0
SHARED_POOL_SIZE = 0
STREAMS_POOL_SIZE = 0
```

```
SELECT name, value, isdefault
FROM   v$parameter
WHERE  name LIKE '%size';
```


When you specify a nonzero value for SGA_TARGET and do not specify a value for an autotuned SGA parameter, the value of the autotuned SGA parameters in the V$PARAMETER view is 0, and the value of the ISDEFAULT column is TRUE.

If you have specified a value for any of the autotuned SGA parameters, the value that is displayed when you query V$PARAMETER is the value that you specified for the parameter.

Managing the Program Global Area (PGA)



Automatic PGA memory management is enabled by default.

The Program Global Area (PGA) is a memory region that contains data and control information for a server process. It is nonshared memory created by the Oracle server when a server process is started. Access to it is exclusive to that server process. The total PGA memory allocated by all server processes attached to an Oracle instance is also referred to as the *aggregated PGA* memory allocated by the instance.

Part of the PGA can be located in the SGA when using shared servers.

PGA memory typically contains the following:

Private SQL Area

A private SQL area also called the user global area (UGA) contains data, such as bind information and runtime memory structures. This information is specific to each session's invocation of the SQL statement; bind variables hold different values, and the state of the cursor is different, among other things. Each session that issues a SQL statement has a private SQL area. Each user that submits the same SQL statement has his or her own private SQL area that uses a single shared SQL area. Thus, many private SQL areas can be associated with the same shared SQL area. The location of a private SQL area depends on the type of connection established for a session. If a session is connected through a dedicated server, private SQL areas are located in the server process's PGA. However, if a session is connected through a shared server, part of the private SQL area is kept in the SGA.

Cursor and SQL Areas

The application developer of an Oracle Pro*C program or Oracle Call Interface (OCI) program can explicitly open *cursors* or handles to specific private SQL areas, and use them as a named resource throughout the execution of the program. Recursive cursors that the database issues implicitly for some SQL statements also use shared SQL areas.

Work Area

For complex queries (for example, decision support queries), a big portion of the PGA is dedicated to work areas allocated by memory-intensive operators, such as:

- Sort-based operators, such as `ORDER BY`, `GROUP BY`, `ROLLUP`, and window functions
- Hash-join
- Bitmap merge
- Bitmap create
- Write buffers used by bulk load operations

A sort operator uses a work area (the sort area) to perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (the hash area) to build a hash table from its left input.

The size of a work area can be controlled and tuned. Generally, bigger work areas can significantly improve the performance of a particular operator at the cost of higher memory consumption.

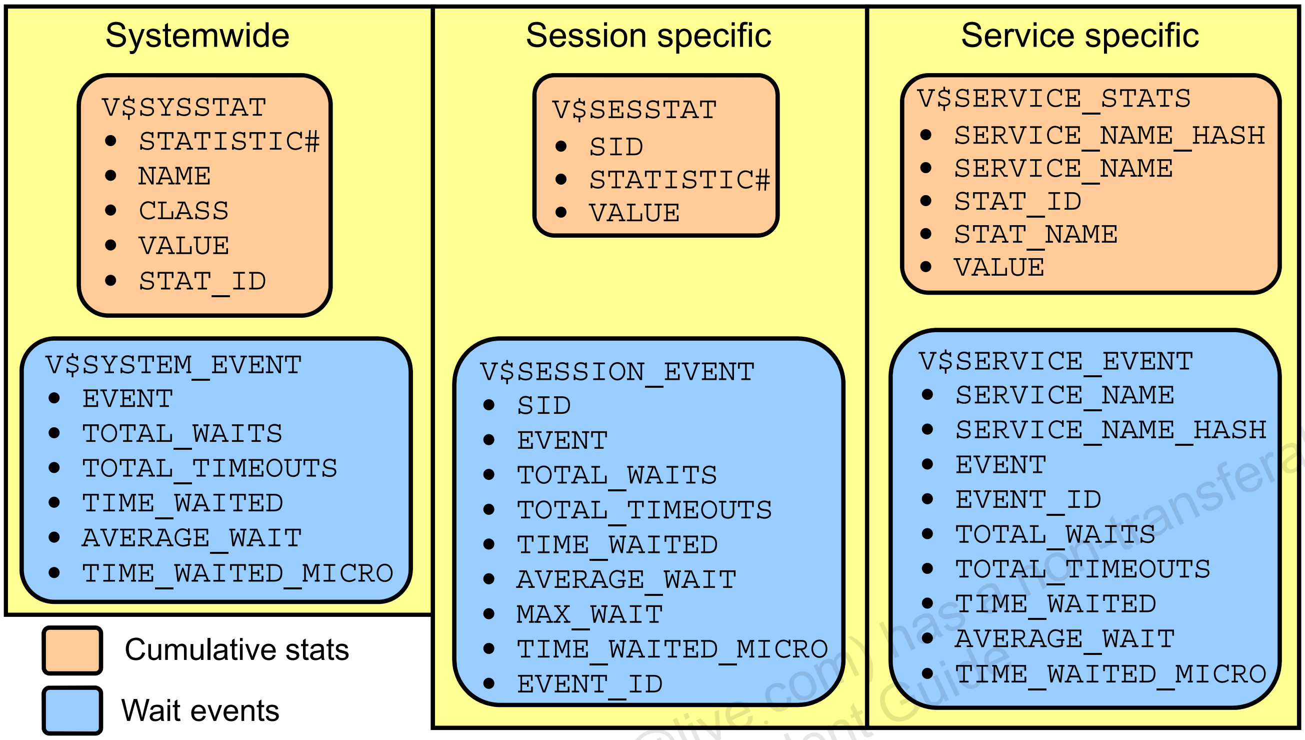
Session Memory

Session memory is the memory allocated to hold a session's variables (logon information) and other information related to the session. For a shared server, the session memory is shared and not private.

Automatic PGA Memory Management

By default, Oracle Database automatically and globally manages the total amount of memory dedicated to the instance PGA. You can control this amount by setting the initialization parameter `PGA_AGGREGATE_TARGET`. Oracle Database then tries to ensure that the total amount of PGA memory allocated across all database server processes and background processes never exceeds this target. But this is target value and not a hard limit. `PGA_AGGREGATE_LIMIT` sets a hard limit for the amount of PGA that can be used. The minimum value is `1024 MB` and the maximum is 120% of physical memory minus the total SGA, and it must be at least as large as `PGA_AGGREGATE TARGET`. If `PGA_AGGREGATE_LIMIT` is not set, it defaults to 200% of `PGA_AGGREGATE_TARGET` within the same minimum and maximum as stated. When `PGA_AGGREGATE_LIMIT` is exceeded, the sessions using the most memory will have their calls aborted. Parallel queries will be treated as a unit. If the total PGA memory usage is still over the limit, sessions using the most memory will be terminated. `SYS` processes and fatal background processes are exempt from this limit.

Dynamic Performance Statistics



Statistics must be available for the effective diagnosis of performance problems. The Oracle server generates many types of statistics for different levels of granularity.

At the systemwide, session, and service levels, both wait events and accumulated statistics are computed. In the slide, the top row of views shows the cumulative statistics. The bottom row shows the wait event views.

When analyzing a performance problem in any of these scopes, you typically look at the change in statistics (delta value) over the period of time you are interested in. All the possible wait events are cataloged in the `V$EVENT_NAME` view. All statistics are cataloged in the `V$STATNAME` view; approximately 480 statistics are available in Oracle Database.

Displaying Systemwide Statistics

Example:

```
SQL>  SELECT name, class, value FROM v$sysstat;
NAME                                CLASS      VALUE
-------------------------------  ------ ----------
...
table scans (short tables)            64     135116
table scans (long tables)             64        250
table scans (rowid ranges)            64          0
table scans (cache partitions)        64          3
table scans (direct read)             64          0
table scan rows gotten                64   14789836
table scan blocks gotten              64     558542
...
```

Systemwide statistics are classified by the tuning topic and the debugging purpose. The classes include general instance activity, redo log buffer activity, locking, database buffer cache activity, and so on.

Troubleshooting and Tuning Views

Instance/Database V$DATABASE V$INSTANCE V$PARAMETER V$SPPARAMETER V$SYSTEM_PARAMETER V$PROCESS V$BGPROCESS V$PX_PROCESS_SYSSTAT V$SYSTEM_EVENT	**Disk** V$DATAFILE V$FILESTAT V$LOG V$LOG_HISTORY V$DBFILE V$TEMPFILE V$TEMPSEG_USAGE V$SEGMENT_STATISTICS
Memory V$BUFFER_POOL_STATISTICS V$LIBRARYCACHE V$SGAINFO V$PGASTAT	**Contention** V$LOCK V$UNDOSTAT V$WAITSTAT V$LATCH

In the slide, some of the views that can help you determine the cause of performance problems or analyze the current status of your database are listed.

For a complete description of these views, see the *Oracle Database Reference*.

Quiz

Automatic Memory Management allows the Oracle instance to reallocate memory from the _______ to the SGA .

a. Large Pool
b. Log Buffer
c. PGA
d. Streams Pool

Answer: c

Quiz

SGA_TARGET may not be sized greater than __________ .

a. LOG_BUFFER
b. SGA_MAX_SIZE
c. STREAMS_POOL_SIZE
d. PGA_AGGREGATE_TARGET

Answer: b

Summary

In this lesson, you should have learned how to use:

- Enterprise Manager to monitor performance
- Automatic Memory Management (AMM)
- The Memory Advisor to size memory buffers

Practice 21

- 21-1: Managing Performance
- 21-2: Using Automatic Memory Management
- 21-3: Monitoring Services