Course Transcript

Oracle 12c Performance Tuning: Introduction

Ground Rules for Performance Management

Basic Tuning Diagnostics

Dynamic Performance Views

Performance Diagnosis Tools

Scoping Performance Issues

Practice: Basic Tuning Diagnostics

Course Introduction

Learning Objective

After completing this topic, you should be able to

- *start the course*

1. Introduction to the course

Hi, my name is Ian Baugaard. In this course, I'll be introducing the principles that should be employed when attempting to resolve a performance problem with an Oracle database. I'll also explain some of the core concepts that are available in Oracle Database 12c, such as database time. And I'll briefly introduce some of the diagnostic tools which will become a part of your arsenal. I hope you'll enjoy the course and get some benefit from it.

Introduction to Performance Management

Learning Objective

After completing this topic, you should be able to

- *define the components of performance management*

1. Components of performance management

In this lesson we'll be defining the concept of performance management and briefly looking at some of the topics which are not included in this course. Some of the components of performance management include defining Service Level Agreements, or SLAs. And this could be a commitment to have a task completed before a certain time of day. Setting goals and objectives which are normally inputs when defining SLAs, but are typically desirable performance targets. Ensuring regular monitoring is in place as this allows the establishment of a performance baseline. Having this in place allows for quick detection of changes and bottlenecks. And finally, ensuring that you have the correct tools available when troubleshooting various performance tuning scenarios. Managing performance is an ongoing task and involves good planning such as ensuring any new application deployments provide the same or potentially better performance than the current deployment. Anticipating change is useful in forecasting performance needs. If it is known, for example, that the transaction volume on this system will be increasing over a period of time.

[Heading: Introduction to Performance Management. Some performance management components include defining service level agreements (SLAs), setting goals and objectives, ensuring regular monitoring, and ensuring the availability of appropriate troubleshooting tools. Managing performance involves good planning and anticipating change.]

It is important to note that diagnosing performance issues in any configuration of an Oracle database starts with the same basic techniques covered in this course. What is not covered are issues specific to certain features and options such as 24x7 availability, online operations, which include the segment shrink commands and index rebuilds, backup performance – specifically covered in the backup and recovery workshop if you're planning to use RMAN to backup your database – parallel operations which are typically covered in data warehousing environments, Oracle Streams and Data Guard performance issues, Real Application Clusters, Operating Systems specific issues such as misconfigured kernel parameters and finally application specific issues such as LOB handling, or PL/SQL performance tuning. In this lesson we defined the concept of performance management.

[There are a number of performance management related topics that aren't included in this course. These are 24x7 availability, online operations, backup performance, parallel operations, Oracle Streams and Data Guard performance issues, Real Application Clusters, operating system specific issues, and application specific issues, such as LOB handling.]

Who Does Performance Management

Learning Objective

After completing this topic, you should be able to

- *identify who is responsible for performance management of the Oracle database*

1. Performance management responsibilities

In this lesson, we'll have a look at which resources are involved with performance management. The database is typically the first component of a system that starts presenting symptoms of a problem. And as a result, the database administrator is typically the first respondent. Application architects and designers are typically involved during major performance tuning initiatives, whereas application developers should continuously be involved as part of the software development lifecycle. System and storage administrators may also be called upon when focus needs to be given to hardware-related issues. Finally, network administrators also have a role to play since most applications have to communicate with the database via the LAN or WAN. If we look at specific performance tuning areas, application tuning is a shared responsibility with developers. Tuning the SQL statement performance and managing change as the application matures, are important parts of application tuning. Instance tuning, however, is the sole responsibility of a DBA. It entails optimally setting memory parameters, such as SGA_TARGET, ensuring database structures, such as the redo logs are optimally configured, and verifying that the instance configuration is ideally suited for the system. This could involve checking parameter settings, such as DB_FILE_MULTIBLOCK_READ_COUNT. Operating system tuning is shared with the system administrators. Typically, this includes checking that the I/O SAP systems is optimal, ensuring that enough Swap capacity is available, and making sure that the kernel parameters for the system are adequately set. In this lesson, we looked at which resources are involved at performance management.

[Heading: Who Does Performance Management. There are a number of different resources typically involved in performance management. These are database administrators, application architects, application designers, application developers, system administrators, storage administrators, and network administrators. There are three different performance tuning areas - application tuning, instance tuning, and tuning operating system interactions. Application tuning is shared with developers and involves SQL statement performance tuning and change management. Instance tuning involves optimizing memory, database structures, and instance configuration. The tuning of operating system interactions is shared with the system administrators and typically includes optimizing the I/O subsystem, ensuring the availability of swap capacity, and adequately setting system parameters.]

Tuning Methodology

Learning Objective

After completing this topic, you should be able to

- *describe the Oracle tuning methodology*

1. Oracle tuning methodology

In this lesson, we'll have a look at the tuning methodologies which can be applied when a performance tuning exercise exists. Essentially, there are two types of tuning. Firstly, proactive tuning which occurs at regular intervals and should be seen as an overall capacity planning process. Secondly, bottleneck elimination which is a more reactive approach and involves donning your fire fighting suit to ease the utilization of an overused resource. Some of the tuning steps include properly identifying the scope of the problem – and this is key as there are potentially many avenues of investigation when a performance tuning problem exists. Then tune from the top down; check the design is optimal before looking at the application code. An example that I've dealt with before was we had developed a user table with a single column and row and this was used to produce sequential numbers. Unfortunately this introduced a huge amount of serialization and was resolved by using an Oracle sequence instead. Tuning the code before tuning the instance – and typically you would look at reducing logical and physical reads before changing table space layouts or changing disk configurations. Then tune the area which will yield the biggest potential benefit by properly identifying the problem, analyzing the problem, and using appropriate tools to tune the implicated components. Finally, there has to be a point of diminishing return, so one should remember to stop once the tuning goal has been met. In this lesson, we looked at the tuning methodologies which can be applied.

[Heading: Tuning Methodology. There are two types of tuning - proactive monitoring or tuning and bottleneck elimination. Proactive monitoring or tuning involves examining performance statistics at regular intervals to identify whether the system behavior and resource usage has changed. Bottleneck elimination involves identifying an overused resource in the system and determining potential fixes. There is a logical sequence of steps to follow when tuning. First identify the scope of the problem, such as the OS, database, and so on. Second, tune from the top down. Apply this rule and tune the design before tuning the application code, and the code before tuning the instance. Third, tune the area with the greatest potential benefit. This is done by identifying the performance problem, analyzing it and looking for skewed and tunable components, and then using appropriate tools to tune the implicated components. Finally, stop tuning once the goal is met.]

Effective Tuning Goals

Learning Objective

After completing this topic, you should be able to

- *describe effective tuning goals*

1. Effective tuning goals

In this lesson we'll discuss what effective tuning goals are. Quite often the DBA is told that the database is slow. Unfortunately this just doesn't provide sufficient information on where to start looking for a problem. Tuning goals are often related to SLAs. And as such they should be specific, measurable, achievable, and cost effective, from both an effort delivered as well as infrastructure cost point of view. As an example this scenario presented a tuning exercise from an online portal which was suffering from response times occasionally exceeding agreed SLAs. This table illustrates the output of an AWR SQL report and has been modified to show the before and after results. The problem statement was narrowed down to a particular piece of SQL, which accounted for half of the logical I/Os or reads from cache over the problem period. By adding an index to the source table, the performance significantly improved as the SQL was originally issuing full table scans, several times per second. The important thing to note here is that the goal was specific; the gain was measurable and completely achievable.

[Heading: Effective Tuning Goals. Effective tuning goals are specific, measurable, achievable, and cost effective. A table containing results of an Automatic Workload Repository (AWR) report displays. It compares per execution and % snap total figures for various stats before and after an index was added as part of a tuning exercise. The table has five columns: Stat Name, BEFORE INDEX Per Execution, BEFORE INDEX % Snap Total, AFTER INDEX Per Execution, and AFTER INDEX % Snap Total. There are 13 rows but only 10 contain data. The first three data rows are: Elapsed Time (ms): 256.72: 43.77: 0.54: 0.01. CPU Time (ms): 117.06: 46.39: 0.05: 0.00. Buffer Gets: 26,851.58: 51.56: 4.15: 0.00 The problem statement for this tuning exercise is "Improve online query performance."]

This screenshot shows the CPU utilization of the system in question which is coincidentally a two-node RAC cluster. As mentioned before, even though RAC-specific tuning is not part of this course the foundation tuning principles of the course still apply regardless of the configuration used. With no other changes taking place, it is evident to see at which point the index was created and subsequently used by the offending SQL. By reducing the logical I/Os there was a subsequent reduction in overall CPU utilization and thus allowing for more effective use of resources. This will be explained in more detail in course 5. In this lesson we discussed what effective tuning goals are.

[This performance tuning exercise also resulted in reduced CPU utilization. A line graph titled "Metric Value History" provides CPU Utilization (%) information. It contains two data sets representing the two node Real Application Clusters. The x axis measures time in hours, starting at 00:00 and ending at 23:00. The y axis measures CPU utilization as a percentage, starting at 8.6 and increasing to 48.6. The average CPU utilization for both data sets averages at 23.6 % from 00:00 to 16:00. Both data sets peak at 09:00 and 16:00. The higher of the two is 48.6% at 09:00 and 41% at 16:00. At 16:01 the tuning exercise is implemented and both data sets drop significantly. One drops to 13.6% and the other to 10%. The average CPU utilization from 16:01 to 23:00 is approximately 14%.]

General Tuning Session

Learning Objective

After completing this topic, you should be able to

- *list the steps of a general tuning session*

1. Steps of a general tuning session

In this lesson we'll look at the components of a general tuning session. In general, tuning sessions are generic and applied to any performance monitoring tool. You will start with defining the problem and goal, which is the analysis step. Collecting current statistics which could include host and database statistics. If a baseline is available, it would be good to compare to the current statistics. Consider common performance errors as we saw in the previous lesson with a missing index. Build a trial solution, looking at why the performance degraded and how to resolve the problem. Implement and measure the change, again as we saw in the lesson before. Ending with confirming whether or not the solution met the goal. If not, loop back to step 3 otherwise make the current set of statistics your new baseline and stop tuning. In this lesson we looked at the components of a general tuning session.

[Heading: General Tuning Session. All tuning sessions follow the same procedure. The first step is to define the problem and state the goal. The second step is to collect current performance statistics. The third step is to consider common performance errors. The fourth step is to build a trial solution. The fifth step is to implement and measure the change. The sixth step is to decide whether the solution met the goal. If not then go back to the third step and repeat the procedure. If yes, the create a new baseline.]

Tuning Objectives

Learning Objective

After completing this topic, you should be able to

- *identify tuning objectives*

1. Identifying tuning objectives

In this lesson, we'll look at some of the common tuning objectives. Typically, this would include minimizing response time or reducing user-wait time; increasing throughput – which means decreasing the time to perform a job or set of jobs – increasing load capabilities, which equates to scalability and ensuring that there is sufficient capacity to allow for more tasks; and finally, reducing recovery time, which potentially bares a trade-off with ensuring the shortest recovery window in the event of a failure versus the general performance of a system. Diagnostics play an important role in defining tuning objectives. And diagnostic tools gather and display the following types of performance data. Cumulative statistics – where your wait events with time information and the time model are key in identifying where time is being spent or lost. Metrics, which provide the basis to proactively monitor performance. And finally, sampled statistics with Active Session History or ASH as an example. It provides statistics by session, statistics by SQL, statistics by service, and other dimensions.

[Heading: Tuning Objectives. Tuning objectives typically include minimizing response time, increasing throughput, increasing load capabilities, and reducing recovery time. Diagnostic tools are used to gather and format several types of performance data. These include cumulative statistics; metrics; and sampled statistics. Cumulative statistics use wait events with time information and the time model to identify where time is spent or lost. Metrics, such as statistic rates, provide the basis to monitor performance. Finally, an example of sampled statistics is Active Session History, which is included as part of the Oracle Diagnostics Pack. It provides statistics by session, by SQL, by service, and other dimensions.]

There are several features and tools which can be used during a performance tuning exercise. Some of the automated features include AWR, ADDM or the Automatic Database Diagnostic Monitor, server-generated alerts, Advisors, and the Database Performance page of Enterprise Manager. Some additional tools include the dynamic performance views, the instance alert.log file, trace files – which could either be extended SQL trace files or error files – and Statspack, which is similar to AWR but available and supplied with all editions of the Oracle database software. In this lesson, we looked at some of the common tuning objectives.

[There are several features and tools that can be used for performance tuning. Automatic performance tuning features include Automatic Workload Repository (AWR), Automatic Database Diagnostic Monitor (ADDM), server-generated alerts, advisors, and the Database Performance page of Enterprise Manager. Some additional tools include V$ performance views, the alert log, trace files, and the Statspack.]

Top Timed Events

Learning Objective

After completing this topic, you should be able to

- *examine top timed events in an AWR report*

1. Examining top timed events

In this lesson we'll look at the Top Timed Event section of an AWR report. When reviewing an AWR report, the Top 10 Foregrounds by Total Wait Time section is generally a good starting point when diagnosing a generic performance issue. This section is available in both AWR and Statspack reports, and at a glance allows you to see the Top Timed Events. In this example, free buffer waits and buffer busy waits are consuming more than 50% of DB time when combined. These events provide a clue as to what the underlying problem could be, and gives insight into which avenue to pursue for further investigation. If the instance is not performing as well as expected, these Top Timed Events indicate that there may either be a problem in the database buffer cache, or that there is a SQL statement that needs tuning. This final example is from a live transactional system and, while taken from an 11g Release 2 environment, you will note that the structure is virtually the same as the 12C equivalent. In most cases you would expect I/O related wait events to be amongst the Top Timed Events, as this has historically been the slowest component from a hardware perspective. Typical average I/O wait time should not exceed greater than 20 milliseconds. So at a glance there is no I/O related problem in this example. If DB CPU is the Top Timed Event, you may wish to consider looking at SQL that is requiring more than necessary logical I/Os or reads from cache. In this lesson we looked at the Top Timed Event section of an AWR report.

[Heading: Top Timed Events. The Top 10 Foreground Events by Total Wait Time section of the Automatic Workload Repository (AWR) report displays the 10 timed events or operations that consumed the highest percentage of total database time, or DB time, in the snapshot set. A table titled Top 10 Foreground Events by Total Wait Time displays. It provides an example of the section available in AWR and Statspack reports. The table has six columns: Event, Waits, Total Wait Time (sec), Wait Avg(ms), %DB time, and Wait Class. There are ten data rows. The first three rows are: buffer busy waits: 45,078: 1907.9: 42: 41.2: Concurrency. write complete waits: 766: 995: 1299: 21.5: Configuration. free buffer waits: 4,947: 597.9: 121: 12.9: Configuration. A second table titled Top 5 Foreground Events by Total Wait Time displays. It provides a specific example taken from an 11.2 Oracle database environment. The table has six columns: Event, Waits, Time(s), Avg wait(ms), %DB time, and Wait Class. There are five data rows: db file sequential read: 1,084,054: 2,243: 2: 65.28: Userl/O. DB CPU: blank: 699: blank: 20.34: blank. log file sync: 762,556: 504: 1: 14.67: User I/O. db file parallel read: 5,872: 22: 4: 0.65: User I/O. direct path read: 2,873: 21: 7: 0.60: User I/O. The presenter switches back to the first table titled Top 10 Foreground Events by Total Wait Time. It provides an example of the section available in AWR and Statspack reports. The table has six columns: Event, Waits, Total Wait Time (sec), Wait Avg(ms), %DB time, and Wait Class. The presenter refers to all ten data rows at a glance, which are: buffer busy waits: 45,078: 1907.9: 42: 41.2: Concurrency. write complete waits: 766: 995: 1299: 21.5: Configuration. free buffer waits: 4,947: 597.9: 121: 12.9: Configuration. DB CPU: blank: 281.3: blank: 6.1: blank. log file sync: 2,058: 208: 101: 4.5: Commit. enq: TX - index contention: 3,206: 145.5: 45: 3.1: Concurrency. library cache lock: 466: 137.9: 296: 3.0: Concurrency. log file switch (checkpoint incomplete): 36: 36.4: 1012: .8: Configuration. latch: cache buffers chains: 12,965: 30.2: 2: .7: Concurrency. db file sequential read: 464,369: 23.2: 0: .5: User I/O.]

DB Time

Learning Objective

After completing this topic, you should be able to

- *define the concept of DB time*

1. Defining DB time

In this lesson we'll define the concept of DB time. By definition, the term DB time is the total time spent by user processes, either actively working or actively waiting, in a database call. In other words, it can be seen as time spent in the database. The following graphic displays this concept nicely. It shows the example of a user browsing a web site and the end-to-end response time of the user's experience, showing exactly how much time was spent within the database. It also shows that tuning should not simply be about reducing waits, but rather at improving the end-user response time. The next graphic shows the components of DB time. Any request to the database is composed of two distinct segments – a wait time and a service time. The wait time is the sum of all the waits for various database instance resources. The CPU time is the sum of the time that is spent actually working on the request. Tuning consists of reducing or eliminating the wait time and reducing the CPU time. This definition applies to any application type, be that OLTP or data warehousing. It is important to note that DB time excludes idle wait time. And that will be covered in later lessons. In this lesson, we defined the concept of DB time.

[Heading: DB time. DB time is the amount of elapsed time – in microseconds – spent performing database user-level calls, not including the elapsed time spent on instance background processes, such as PMON. A basic diagram representing time spent in the database displays. A long bidirectional arrow titled "User Response Time" points left and right. A section in its center is highlighted and labeled DB time. Including this section, the User Response Time arrow is divided into nine sections. From left to right, these are Browser, WAN, Apps Server, LAN, DB time, LAN, Apps Server, WAN, and Browser. A second diagram displays. A horizontal arrow labeled "DB time" points from left to right. It is divided into two parts. The left-hand section of the arrow is labeled "DB Wait Time" and the right-hand portion is labeled "DB CPU Time."]

CPU and Wait Time Tuning Dimensions

Learning Objective

After completing this topic, you should be able to

- *describe CPU and wait time tuning dimensions*

1. CPU and wait time tuning dimensions

In this lesson, we'll look at the CPU and wait time tuning dimensions. When tuning your system, it is important that you compare the CPU time with a wait time of your system. By comparing CPU time with wait time, you can determine how much of the response time is spent on useful work and how much is spent waiting for resources potentially held by other processes. As a general rule, systems where CPU time is dominant usually need less tuning than the ones where wait time is dominant. However, heavy CPU usage can be caused by badly written SQL statements. The proportion of wait time to CPU time always tends to increase as the load on the system increases. Steep increases in wait time are a sign of contention and must be addressed for good scalability. When contention is evidenced by increased wait time, adding more CPUs to a node, or nodes to a cluster, would provide very limited benefit. Conversely, a system where the proportion of CPU time does not decrease significantly as load increases can scale better and would most likely benefit from adding CPUs or Real Application Cluster instances. Do note that Automatic Workload Repository and Statspack reports display CPU time in the Top 10 Foreground Events section, if the CPU time portion is among the top events. In this lesson, we looked at the CPU and wait time tuning dimensions.

[Heading: CPU and Wait Time Tuning Dimensions. A basic formula and chart display. The formula states that DB time is equal to the sum of DB CPU time and DB wait time. The chart's x axis measures DB wait time and its y axis measures DB CPU time. The chart is divided into three areas. In the first area, there is a significant increase in CPU time without a correlating increase in wait time. In this case, the application possibly needs SQL tuning. In the second area, there is a significant increase in wait time without a correlating increase in CPU time. In this case, the application needs instance/RAC tuning and there is no gain achieved by adding CPUs or nodes. Finally, in the third area, the increase in CPU time and wait time correlates and divides the other two extremes. When an application's increase in CPU time correlates with an increase in wait time then it is scalable.]

Time Model Overview

Learning Objective

After completing this topic, you should be able to

- *define the time model*

1. The time model

In this lesson we'll define the Time Model concept. When looking at the system as a whole, time is the only common denominator for comparison across various components. In the Oracle database server, most of the advisories report their findings in time. Also statistics called Time Model statistics appear as the V$SYS_TIME_MODEL and V$SESS_TIME_MODEL dynamic performance views. This instrumentation helps the Oracle database server to identify quantitative effects on the database operations. Since DB time represents the total time spent in database calls, it is an excellent starting point when beginning a general tuning exercise of an Oracle database. The objective for tuning an Oracle database system could be stated as reducing the time that users spend in performing some action on the database, or by simply saying we want to reduce DB time.

[Heading: Time Model Overview. The time model is a set of statistics that give an overview of where time is spent inside the Oracle database. All statistics use the same dimension: time. Time model statistics are accessible through the following dynamic performance views: V$SYS_TIME_MODEL and V$SESS_TIME_MODEL. DB time is a good place to start a general tuning exercise since it represents the total time spent in database calls by user sessions. The objective of an Oracle database tuning exercise is to reduce DB time.]

DB time extends to looking at the performance impact of any database entity such as logon operations, hard and soft parses, PLSQL execution time, and Java execution. This graphic is taken from an AWR report, though the Time Model information is also available in the Statspack reports. The statistics are ordered by the percentage of DB time value, so the area that is taking the most time, and its children, are first on the list. In this example, SQL execute elapsed time is at the top. Note that the sum of percentage of DB time of the individual statistics is more than 100%, and that is because there is an overlap between statistics. For example, you need a bit of DB CPU time in order to execute SQL statements. And those obviously are shared. In this lesson we defined the Time Model concept.

[By using DB time you can gauge the performance impact of any entity of the database. A basic pie chart titled "DB time" displays. It contains the segments: SQL, PLSQL, Connect, Parse, and Java. SQL, accounts for 57%. Parse accounts for 20%, Connect accounts for 10%, and Java and PLSQL each account for 6.5%. A table containing time model statistics is provided as a time model example. The total time in database user-calls (DB time) is 24321.8s. Statistics including the word "background" measure background process time and all statistics are ordered by % or DB time descending, Statistic name. The table contains three columns: Statistic Name, Time (s), and % of DB time. There are 15 data rows. Examples include: sql execute elapsed time: 19,564.83: 80.44. connection management calls elapsed time: 2,901.18: 11.93. DB CPU: 2,394.16: 9.84. PL/SQL execution elapsed time: 807.35: 3.32. parse time elapsed: 88.90: 0.37. Note that the sum of the "% of DB time" of the individual statistics is more than 100%.]

Dynamic Performance Views

Learning Objective

After completing this topic, you should be able to

- *work with the dynamic performance views*

1. Dynamic performance views

In this lesson, we'll discuss the Dynamic Performance Views. The Oracle database server maintains a dynamic set of data about the operation and performance of the instance. These dynamic performance views are based on virtual tables that are built from memory structures inside the database server. In other words, they are not conventional tables. Also, dynamic performance views include the raw information that is used by AWR and Statspack. They are commonly referred to as V$ views and they externalize metadata contained in memory structures of an Oracle instance. Where RAC environments are concerned, you can query GV$ views or Global V$ views, which contain information for all the instances in the cluster. Since Enterprise Manager uses dynamic performance views, DBAs can also query these views as needed. The examples shown in the slide answer the following questions. Block A asks, what are the SQL statements and their associated number of executions where the CPU time consumed is greater than 200,000 microseconds? Block B asks, what sessions are logged in from the ED9P1 computer within the last day? And finally, what are the session IDs of any sessions that are currently holding a lock that is blocking another user, and how long has that lock been held?

[Heading: Dynamic Performance Views. Dynamic performance views provide access to information about changing states and conditions in the instance. They are continuously updated while a database is open and in use, and their contents relate primarily to performance. Some examples of dynamic performance views are Session data, Wait events, Memory allocations, and Running SQL. They are commonly referred to as v-dollar (v$) views. Global v-dollar, or gv$ views are specific to Real Application Cluster environments. Three SQL queries display as usage examples. The first example, labeled "a", is: SELECT sql_text, executions FROM v$sqlstats WHERE cpu_time > 200000; The second example, labeled "b", is: SELECT * FROM v$session WHERE machine = 'ED9P1' and logon_time > SYSDATE -1; The third and final example, labeled "c", is: SELECT sid, ctime FROM v$lock WHERE block > 0;]

So let's look at some of the considerations when using V$ views. First of all, all these views are owned by SYS or the database super user. Second, some dynamic performance views contain data that is not applicable to all states of a database instance. In other words, some views are only applicable when the instance is started, when the database has been mounted, or when the database has been opened. The V$FIXED_TABLE view lists all the dynamic performance views. Dynamic performance views are based on memory structures and the data in them is cumulative since startup. The data is reset at startup, and that's an important point to remember. In other words, you can't know, for example, how many physical reads happened between two times by calculating the delta, if the instance had been restarted during those two times. Finally, because all the reads on these views are current reads, there's no locking mechanism on these views. So there's no guarantee that the data would be read-consistent. In this lesson, we discussed the Dynamic Performance Views.

[There are several considerations to bear in mind when using v$ views. These views are owned by SYS. Different views are available at different times, such as when the instance has been started, the database is mounted, or the database is open. You can query V$FIXED_TABLE to see all the view names. Since v$ views contain cumulative data since startup, they are reset at startup. Finally, all reads on these views are current reads.]

Wait Events

Learning Objective

After completing this topic, you should be able to

- *describe database wait events*

1. DB wait events

In this lesson, we'll discuss wait events. When a session has to wait while processing a user request, the database records the wait by using one of a set of predefined wait events. Wait events are incremented by a session to indicate that the session had to wait for an event, to complete before being able to continue processing. The events are then grouped into wait classes, such as user I/O or network. Each event can have additional parameters returned with the event, columns PARAMETER1 through PARAMETER3 show the meaning of these parameters. And all of this information can be retrieved from the V$EVENT_NAME view. Do note that time information columns for wait events are populated only, if the time statistics initialization parameter is set to true. A wait event can have up to three parameters. The meaning of each of these parameters is listed in the parameter n columns. In the slide below, you can see the different meanings that each parameter n value has, for each corresponding wait event. If we were to take the db file sequential read wait event, parameters 1 to 3 indicate the file that was being read as part of the wait, the block number that was being retrieved, and finally the number of blocks read. For this particular wait event, which is typically associated with indexed access paths, this value will always be 1 as it is a single block read. All in all, there are over 1,650 wait events associated with Oracle release 12.1.0.2.0. And this number usually increases with each release. So it isn't practical to know all of the wait events, but it is a good idea to understand the ones you will typically see in your environment. In this lesson, we discussed wait events.

*[Heading: Wait Events. A collection of wait events provide information about the sessions that had to wait, or must wait, for different reasons. These events are listed in the V$EVENT_NAME view, which have the columns: EVENT#, NAME, PARAMETER1, PARAMETER2, and PARAMETER3. Heading: Wait Events: Using the V%EVENT_NAME view. The SQL statement displays: SELECT name, parameter1, parameter2, parameter3 FROM v$event_name;. Below this is a table that lists various wait events and shows the meaning of each of their corresponding parameter values. The table has four columns: NAME, PARAMETER1, PARAMETER2, and PARAMETER3. There are 1118 rows selected in this example. There are five data rows: PL/SQL lock timer: duration: blank: blank. buffer busy waits: file#: block#: id . library cache pin: handle addr: pin address: 0*mode+name. db file sequential read: file#: block#: blocks. transaction: undo seg#: wrap#: count.]*

Wait Classes

Learning Objective

After completing this topic, you should be able to

- *describe database wait classes*

1. DB wait classes

In this lesson, we'll discuss wait classes. The many different wait events possible in Oracle database are categorized into wait classes, on the basis of the solutions related to that event. Each event is related to only one wait class. This enables high-level analysis of the wait events, and provides a general indication of where the performance problem exists. We've already seen that all wait events are catalogued in the v$event_name view. And some of the other dynamic performance views that are relevant here are v$session_wait_class, v$system_wait_class, and v$service_wait_class. The following slide shows some of the most commonly occurring wait classes. And some of the most relevant from field experience are administration, which relates to DBA commands that cause other users to wait such as index rebuilds. Application, which are usually lock waits caused by row-level locking or explicit lock commands. Commit, which are related to waits for redo log write confirmation after a commit. Concurrency, which is concurrent passing and buffer cache latch and lock contention. Configuration, which is typically something like an undersized log buffer or buffer cache. User I/O, which are waits for blocks to be read off disk by foreground processes. And network communications, which are waits for data to be sent over the network. Do note that the other wait class contains waits that should not typically occur on a system. For example, wait for EMON to spawn. In this lesson, we discussed wait classes.

[Heading: Wait Classes. Every wait event belongs to a specific class of wait event. The wait_class gives you an approximate idea of where the performance problem will be found. The following dynamic performance views are relevant: v$session_wait_class, v$system_wait_class, v$service_wait_class, and v$event_name. A SQL statement and corresponding table provide a distribution of waits by wait class. The SQL statement is "select wait_class, count(*) from v$event_name group by wait_class order by 1 ;". The table has two columns: WAIT_CLASS and COUNT(*). There are 13 data rows: Administrative: 57. Application: 17. Cluster: 64. Commit: 4. Concurrency: 38. Configuration: 26. Idle: 121. Network: 28. Other: 1,186. Queueing: 9. Scheduler: 9. System I/O: 35. User I/O: 56.]

Wait Event Statistics

Learning Objective

After completing this topic, you should be able to

- *use SQL to display wait event statistics*

1. Wait event statistics

In this lesson, we'll discuss wait event statistics. Wait events can be categorized as System, Service, or Session specific waits. Cumulative wait events statistics for all sessions are stored in V$SYSTEM_EVENT, which shows the total waits for a particular event since instance start-up. V$SERVICE_EVENT shows the cumulative wait event statistics for each service. And V$SESSION_EVENT shows cumulative wait event statistics for current sessions, and contains a SID column for matching waits to a particular user session. Most of the wait event statistics are the same for each view, and the differences can be seen in the table on the right-hand side. The V$SYSTEM_EVENT view includes a breakout of statistics for foreground processes. V$SERVICE_EVENT includes the service and V$SESSION includes the session identifier. Note that V$SERVICE_EVENT does not include the three wait class columns, but they are easily obtained by joining to V$EVENT_NAME. When you are troubleshooting, you need to know if a process has waited for any resource. The output depicted on this slide shows all the events that a session has waited on. And how many times the session waited on them, where the time waited in microseconds was greater than zero. In this lesson, we discussed wait event statistics.

[Heading: Wait Event Statistics. There are three wait event statistics levels: system, service, and session. These statistics levels correspond respectively to the following views: V$SYSTEM_EVENT, V$SERVICE_EVENT, and V$SESSION_EVENT. Many of the wait event statistics columns are the same for each view but there are others that vary by view. The V$SYSTEM_EVENT view includes the following wait event statistics columns: EVENT, TOTAL_WAITS, TOTAL TIMEOUTS, TIME_WAITED, AVERAGE_WAIT, TIME_WAITED_MICRO, EVENT_ID, TOTAL_WAIT_FG, TOTAL_TIMEOUTS_FG, TIME_WAITED_FG, AVERAGE_WAIT_FG, TIME_WAITED_MICRO_FG, WAIT_CLASS_ID, WAIT_CLASS#, AND WAIT_CLASS. The V$SERVICE_EVENT view includes the following wait event statistics columns: EVENT, TOTAL_WAITS, TOTAL TIMEOUTS, TIME_WAITED, AVERAGE_WAIT, TIME_WAITED_MICRO, EVENT_ID, SERVICE_NAME, and SERVICE_NAME_HASH. Finally, the V$SESSION_EVENT view includes the following wait event statistics columns: EVENT, TOTAL_WAITS, TOTAL TIMEOUTS, TIME_WAITED, AVERAGE_WAIT, TIME_WAITED_MICRO, EVENT_ID, SID, WAIT_CLASS_ID, WAIT_CLASS#, AND WAIT_CLASS. A SQL statement and corresponding output table shows aggregated session wait event statistics. The SQL statement is "select event, count(*) from v$session_wait where wait_time_micro > 0 group by event ;". The output table has two columns: EVENT and COUNT(*). There are eight data rows: jobq slave wait: 1. db file sequential read: 3. PL/SQL lock timer: 4. db file scattered read: 1. external table read: 1. rdbms ipc message: 1. log file sequential read: 1. Streams AQ: waiting for time management or cleanup tasks: 1.]

Commonly Observed Wait Events

Learning Objective

After completing this topic, you should be able to

- *describe commonly observed wait events*

1. Common wait events

In this lesson, we'll look at some of the most commonly observed wait events. This slide shows a list of wait events and component areas that could be the source of these waits. The internal definitions of these wait events can change from version to version, and may cause other events to become more common. For a description of wait events for a particular database version, it is best to check the Oracle database reference associated with that version. Do note that not all wait events are documented. As the first two events contained the buffer keyword, they can easily be associated with the buffer cache as a starting point. Db file scattered and sequential reads are I/O related. And Enqueue waits refer to Locks. Library cache waits typically require you to consider looking at Mutexes or Latches, and those are serialization mechanisms to protect memory areas. Log buffer space could be indicative of an undersized log buffer parameter. And finally, Log file sync is typically associated with commit operations.

[Heading: Commonly Observed Wait Events. A table lists various wait events and their potential component area sources. The table has two columns: Wait Event and Area. There are seven data rows: Buffer busy waits: Buffer cache, DBWR. Free buffer waits: Buffer cache, DBWR, I/O. Db file scattered read, Db file sequential read: I/O SQL Tuning. Enqueue waits (enq:): Locks. Library cache waits: Mutexes/latches. Log buffer space: Log buffer I/O. Log file sync: Over-commit, I/O.]

This slide shows a TKPROF extract and this is a topic, which will be discussed later in course 4. Having profiled an extended SQL trace file, this output provides a good indication of which wait events a particular piece of SQL waited for, how many times per event it had to do so, the maximum wait time per event, and also what the total time waited per event was. Note that all times are listed in seconds, so it is important to remember where this applies and where the timings are in microseconds. In this lesson, we looked at some of the most commonly observed wait events.

*[A sample output from the Transient Kernel PROFile (TKPROF) utility displays in table format. It shows elapsed times waiting on various events. The table has four columns: Event waited on, Time Waited, Max. Wait, and Total Waited. There are six data rows: db file scattered read: 76561: 4.49: 1766.57. latch free: 958: 0.12: 4.46. db file sequential read: 3648: 0.12: 3.42. SQL*Net message from client: 672330: 1.26: 174.94. SQL*Net message to client: 672329: 0.00: 0.81. log file sync: 3: 0.01: 0.02.]*

Enterprise Manager Overview

Learning Objective

After completing this topic, you should be able to

- *describe Oracle Enterprise Manager*

1. Oracle Enterprise Manager

In this lesson, we'll provide an overview of Oracle Enterprise Manager. There are two primary Enterprise Manager products, which are referred to as EM for short, available for use. EM Database Express, which is a web-based management tool, built inside the database, and supports basic database administration and key performance management functions. There are no middle-tiered components required. And since the database has to be open in order to use EM express, it cannot perform actions outside of the database. Examples of this include, initiating a database start-up or changing the archive log mode of the database. Then there is EM Cloud Control, which provides a holistic environment, from which you can manage and monitor your infrastructure and software landscape. Some of the components include a repository database, a management service, management agents, and the Cloud Control console. There are additional features related to cloud deployment and management, as well as software provisioning and patching to name but a few. This slide shows the landing page for an EM Cloud Control console, and there is no significant difference when compared to EM Database Express. This final screenshot shows the Performance Hub page of an EM Express console. This particular page is not present as is in EM Cloud Control, but it shares similar content to the database homepage of EM Cloud Control. In this lesson, we had a brief overview of Oracle Enterprise Manager.

[Heading: Enterprise Manager Overview. There are two Enterprise Manager products available: Enterprise Manager Database Express and Enterprise Manager Cloud Control. Enterprise Manager Database Express is a web-based database management tool built inside the Oracle Database. It supports basic database administration and key performance management functions. Enterprise Manager Cloud Control is an end-to-end solution for managing and supporting application middleware, database, and hardware tiers. It enables cloud deployment integration with metering and chargeback capabilities. An image illustrates the landing page for Enterprise Manager Cloud Control 12c console open in the web browser. The page contains a login box with User Name and Password fields and a Login button. There are also three sections listed at the bottom of the landing page. These are Enterprise Manager Key Features, New in this Release, and Did you know. A second image displays the Performance Hub page within the Oracle Enterprise Manager Database Express 12c console. A line chart displays at the top of the Performance Hub page and shows real time data for the last hour. Below this are six tabbed pages: Summary, Activity, Workload, Monitored SQL, ADDM, and CurrentADDM Findings. The Summary tabbed page is selected. It contain four sections, each containing a bar chart. The four sections are Host: Runnable Processes, Memory: Host Total 17.3GB, Active Sessions, and I/O. The Active Sessions section contains two tabbed pages - Activity and Services. The Activity tabbed page is selected. The I/O section contains three tabbed pages - Requests, Throughput, and Latency. The Requests tabbed page is selected.]

Oracle EM Architecture

Learning Objective

After completing this topic, you should be able to

- *describe the Oracle EM architecture*

1. EM Express architecture

In this lesson, we'll look at more detail around EM Database Express. Oracle EM Database Express is a lightweight administration tool. It provides an out-of-the-box, browser-based management solution for a single Oracle database or database cluster, including performance monitoring, configuration management, administration, diagnostics, and tuning.This slide shows the EM Database Express architecture, which uses a web-based console communicating with a built-in web server available in XML DB. As the requests from the console are processed, the EM Database Express servlet handles the requests including authentication, session management, and caching. The servlet passes requests for reports to the common reporting framework, and actions requiring shell files to the file manager. As indicated previously, EM Database Express is available only when the database is open. Also, EM Database Express is configurable with a single click in DBCA, or Database Configuration Assistant, and requires that the XML DB components are installed. All Oracle version 12.1 databases and higher have XML DB installed by default.

[Heading: Oracle EM Architecture. A flow diagram representing the Oracle EM database architecture displays. It is comprised of seven main components, which are Console, Listener, Dispatcher, SharedServer, Oracle Web Server, Common Reporting Framework, and File Manager. The console sends a request to the listener, which sends the request via the dispatcher to the SharedServer. The SharedServer sends the request to the Oracle Web Server. At this point the EM Express Servlet handles the request by performing the following tasks: authentication, session management, compression, and caching. The EM Express Servlet passes requests for reports to the Common Reporting Framework, and requests for shell files to the File Manager.]

To activate EM Database Express in a database, verify that the DISPATCHERS initialization parameter has at least one dispatcher configured for the XML DB service with the TCP protocol. Use the setHTTP procedure in the DBMS_XDB_CONFIG package to configure a port on the server and connect to the EM Database Express console with the URL shown in the slide. Substitute the hostname of the server, and the port number you set using the setHTTPPort procedure in the URL. If you have multiple instances to monitor on the same machine, set a different port for each. EM Database Express uses Shockwave Flash files, so the web browser must have the Flash plug-in installed. In this lesson, we looked at more detail around EM Database Express.

[Configuring EM Database Express requires configuring an HTTP listener port for each database instance. This involves three tasks. First verify the DISPACTHERS parameter using the code: dispatchers=(PROTOCOL=TCP) (SERVICE=sampleXDB) Next use the DBMS_XDB_CONFIG.setHTTPPort procedure by using the code: exec DBMS_XDB_CONFIG.setHTTPPort (5500) Finally, launch EM Database Express using the following URL: http://hostname:5500/em. Make sure to use a different port for each instance and note that the web browser requires the Flash browser plug-in to be installed.]

Viewing the Performance Hub Page

Learning Objective

After completing this topic, you should be able to

- *use OEM to view the performance hub page*

1. The Performance Hub page

In this lesson we'll look at navigating around EM Database Express. You can access EM Database Express by opening your web browser and entering the following URL: hostname is the name or IP address of your computer, and port number is the EM Database Express HTTP port number that is specified during installation – the default port is 5500. The EM Database Express Home page is your starting point to monitor and administer your database. You can use the Database Home page to determine the current status of the database by viewing a series of metrics, or access the Configuration, Storage, Security, and Performance pages via menus. This slide shows the Database Home page, which enables you to monitor the status of your database and the workload it is experiencing. The various sections of the Database Home page provide detailed information and menu options can be used to get more detail around problem areas, and in some cases to obtain recommendations for resolving the problems. The Database Home page contains the following charts and sections: the Activity Class chart, which shows the average number of database sessions active for the past hour, including the type of activity for each session; Services chart, which shows the average number of database sessions active for the past hour for database services; the Host CPU chart shows the percentage of CPU time used by the database instance and other processes during the last minute, including the percentage of CPU used by foreground and background instance processes...

[Heading: Navigating around EM Express. You can access EM Express by entering the following the URL in the following format in your browser: http://host name:5500/em. An image of the Oracle EM Express 12c login page displays. It contains User Name and Password fields and a Login button. An arrow points from the login button to a second image, which shows the Database Home screen of EM Express 12c once logged in. It contains various sections, such as Status, Incidents - Last 24 Hours, and Resources. A larger, more in depth image of the Database Home page displays. Four options are listed in the top-level menu - Configuration, Storage, Security, and Performance. In the main page area are four different sections - Status, Incidents - Last 24 Hours, Performance, and Resources. The Status section contains a list of fields and corresponding values. For example, Up Time, Type, Version, Database Name, Instance Name and Platform Name. The Incidents - Last 24 Hours section contains a table with five columns: Instance, Time, Incident, Problem, and Error. There are currently no incidents listed in the table. The Performance section contains two tabbed pages - Activity Class and Services. The Activity Class tabbed page is selected and contains a bar chart indicating Wait, User I/O, and CPU levels for a given time period. The Resources section contains four subsections. These are Host CPU, Active Sessions, Memory, and Data Storage. Each of these subsections contains a thermometer chart indicating various metric level.]

...and finally the Active Sessions chart, which shows the average number of active sessions during the last minute, broken out by Wait, User I/O, and CPU. Finally, here we can see the Performance Hub page. It is designed in a master/detail format, where the time picker at the top of the page drives the contents of the details displayed in the tabs below. The time picker displays average active sessions over time, and if there are peaks in the time picker, the user can move the selected time range to the period of interest to get more information. The Performance Hub can be used to view both historical and real-time data. In real-time mode, performance data is retrieved from in-memory views. The time picker shows data for the past hour, and the user can select any time range from within this period. The default selection is the past five minutes. In historical mode, data is retrieved from the Automatic Workload Repository, or AWR; the user can select any time period provided there is sufficient data in AWR. The Performance Hub organizes performance data by dividing it into

different tabs, with each tab addressing a specific aspect of database performance. In this lesson, we looked at navigating around EM Database Express.

[An image of the Performance Hub page within the EM Express 12c console displays. A line chart titled "Real Time - Last Hour" displays at the top of the page. It shows performance related data for the time period 1:15PM to 2:15PM. Below it are six tabbed pages: Summary, Activity, Workload, Monitored SQL, ADDM, and CurrentADDM Findings. The Summary tabbed page is selected. It contains four sections, each of which present performance related data in a bar chart for the time period 1:15PM to 2:15PM. The four sections are Host: Runnable Processes, Memory: Host Total 7.7GB, Active Sessions, and I/O. The Active Sessions section contains two tabbed pages - Activity and Services. The Activity tabbed page is selected. The I/O section contains three tabbed pages - Requests, Throughput, and Latency. The Requests tabbed page is selected.]

Using the Alert Log in Tuning

Learning Objective

After completing this topic, you should be able to

- *describe the instance alert log*

1. The alert log

In this lesson, we'll look at using the instance alert log. The alert log of a database is a chronological log of messages and it is written out by the Oracle database server. Typical messages found in this file are: any non-default initialization parameters used at startup, all internal errors, block corruption errors, log switches, and the like. Also included are various administrative operations, such as the SQL statements, CREATE, ALTER, DROP DATABASE and TABLESPACE; and the STARTUP, SHUTDOWN, ARCHIVELOG, and RECOVER statements. Some of the information contained in the alert log that is useful when tuning a database, is the time to perform archiving, which could relate to either a poorly performing I/O subsystem or insufficient archival background processes; instance recovery start and complete times; deadlock and timeout errors; incomplete checkpoints; and checkpoint start and end times.

[Heading: Using the Alert Log in Tuning. An image displays the top right-hand portion of the EM Cloud Control 12c console. Four options are visible on the top-level menu - Enterprise, Targets, Favorites, and History. Directly below this is section titled "Orcl". It contains its own menu where five options are visible: Oracle Database, Performance, Availability, Schema, and Administration. The Oracle Database menu is expanded and contains a number of options, such as Home, Monitoring, Control, and Logs. The Logs submenu is expanded and it contains four options - Text Alert Log Contents, Alert Log Errors, Archive/Purge Alert Log, and Trace Files. The Text Alert Log Contents option is highlighted and an arrow points from it to a separate snapshot containing a list of typical text alert logs output by the Oracle DB server. These include a list of Sweep entries, errors, and incident details. For example, one output line states "Errors in file /u01/app/orcle/diag/rdbms/orcl/trace/orcle_j004_17702.trc (incident=12461):". The alert log file contains information that can be used when tuning a database. This information includes the time to perform archiving, instance recovery start and complete times, deadlock and timeout errors, incomplete checkpoints, and checkpoint start and end times.]

Those last two points are most often prevalent on systems generating large amounts of redo, and the FAST_START_MTTR_TARGET initialization parameter and size of the redo log files would be good starting points to look at in resolving this. Because the file can grow to an unmanageable size, you can periodically backup the alert log and delete the current one. This does not present a problem, as a new file is created when the database attempts to write to the file again. Do note that there is an XML version of the alert log in the directory with the format shown. To determine the location of the alert log with SQL*Plus, connect to the database with SQL*Plus, or any other query tool such as SQL Developer, and query the V$DIAG_INFO view, or execute the show parameter background command. Both of these options will provide the paths to the diagnostic directories. ADRCI can also be used to view the alert log. In this lesson, we looked at using the instance alert log.

[The alert log file can be backed up and deleted if it grows to an unmanageable size. This doesn't create any problems because the database creates a new alert file when it attempts to write to it again. Also note that there is an XML version of the alert log file in the following directory: $ORACLE_BASE/diag/rdbms/<db_name>/<SID>/alert.To determine the location of the alert log with SQL*Plus, you can either query the V$DIAG_INFO view or execute the "show parameter background" SQL command. Both of these options provide the path to the diagnostic directories. ADRCI can also be used to view the alert log file.]

User Trace Files

Learning Objective

After completing this topic, you should be able to

- *describe user trace files*

1. User trace files

In this lesson, we'll look at user trace files. Server processes can generate user trace files at the request of the user or DBA, and can be traced at the session or instance level. Do note that instance-level tracing should only be enabled when absolutely necessary, as tracing all sessions will create an I/O load and can fill up the file system quite quickly. User trace files are created on a per-server process basis, and can be created by enabling SQL TRACE, typically by calling the DBMS_MONITOR package – we will cover this in more detail in course 4 – executing the BACKUP CONTROL FILE TO TRACE command, and finally by any processing error that may occur. This slide shows a sample SQL trace file. The header is comprised of information such as the database version, the Oracle home, and information pertaining to the machine hosting the Oracle instance.

[Heading: User Trace Files. Server-process tracing can be enabled or disabled at the session or instance level. User trace files are created on a per-server process basis. They can be created by enabling SQL TRACE. A trace file contains statistics for traced SQL statements in that session. They can also be created by executing the BACKUP CONTROL FILE TO TRACE command and by processing errors.]

The next section includes information on the parent server process; that is the session that was being traced. It will indicate detailed session information, including timestamps, service information, module name, et cetera. The detailed wait events that the session waited on while tracing was enabled, is then displayed. Finally, background processes also create trace files. In general, these files contain diagnostic information, and not information regarding performance tuning. However, by using events, information regarding performance can be written to these files. Database events can be set by the DBA, but it is usually done under the supervision of Oracle Support. An exception to this rule is the 10053 event that can be used to trace the optimizer choices. This event will be touched on later in course 4 as well. In this lesson, we looked at user trace files.

[A sample SQL trace file displays. It contains three sections. The header section contains details such as ORACLE_HOME, system name, node name, version, machine, and instance name. The ORACLE_HOME is /opt/apps/oracle/database/12.1.0.2, the system name is Linux, The node name is localhost, the version is #1 SMP Sun Jul 27 15:5546 EDT 2014, the machine is x86_64, and the instance name is orcl. The second section contains details such as SESSION ID, CLIENT ID, SERVICE NAME, MODULE NAME, CLIENT DRIVER, and CONTAINER ID. For example, the SERVICE NAME is (orcl.localhost.com) 2015-07-14 03:06:30.865. The third section contains detailed wait events that the session waited on. These include BEGIN, END, EXEC, WAIT, and CLOSE entries. Background processes also use trace files. The Oracle database server dumps information about errors detected by any background process into trace files. Oracle Support uses these trace files to diagnose and troubleshoot. These files do not usually contain tuning information.]

Defining the Scope of a Problem

Learning Objective

After completing this topic, you should be able to

- *describe how to define and limit the scope of a problem*

1. Defining and limiting a problem's scope

In this lesson, we'll clarify how to define the scope of a performance problem. Problems can arise at any time. A proactive DBA watches for problems and corrects them before they are noticed by users. In the past, the discovery and definition step has been tedious and frequently dependent on listening to user feedback. While user feedback is important, it is often subjective and not reproducible. To quickly identify the problem, the DBA will monitor the current state of the database instance and compare it to a previous state, and also examine AWR or Statspack reports and instance files carefully. The next step is identifying the location of the performance issue. Does it originate in the application SQL, the database instance, or the operating system? This question is not always easy to answer. Poorly performing SQL can cause excessive physical reads and writes appearing to be an I/O issue. Improperly sized memory components, which is an instance configuration issue, can lead to excessive swapping in the operating system.

[Heading: Defining the Scope of a Problem. A basic flow diagram for defining the scope of a problem displays. It shows a database administrator (DBA) receiving monitoring and reports and files information from a specific database instance. The DBA also receives feedback from users. The next step is identifying the location of the problem. Problems can originate from the SQL application, the database instance, or the operating system.]

In this case, one has to eliminate possibilities. When the instance appears to have I/O problems, compare the instance file I/O statistics with operating system level statistics; the differences can guide you to the actual problem. For example, a higher than normal average wait time on a particular tablespace could be due to: hardware, where the file is on a slower drive or has an improper RAID configuration; the operating system, where it is busy with other files on the same drive or partition; the instance, where the tablespace in question was created with properties different to other tablespaces; or the application, where excessive I/O is the result of poor access path choices, stale statistics, or inefficient indexes. Determine the scope of the problem to focus your efforts on the solutions that provide the most benefit. Look at the system performance in terms of service and wait times. The service time is the time spent actively working on the request or CPU time, and the wait time by definition, is the time waiting for any reason. Then look at which component is consuming the greatest amount of time, and drill down to tune that component. To tune the service time, something has to change, either the processing, the SQL, the access path, or the data storage structures. Wait times can be tuned by reducing contention for the resource where the wait is occurring. In this lesson, we clarified how to define the scope of a performance problem.

[Choose the problem that has the greatest impact. This is done by first analyzing system performance in terms of work done, that is CPU or service time, versus the time spent waiting for work, referred to as wait time. Next determine which component consumes the greatest amount of time. Finally, drill down to tune that component, if appropriate.]

Common Tuning Problems

Learning Objective

After completing this topic, you should be able to

- *list common tuning problems*

1. Identifying common tuning problems

In this lesson, we'll look at some of the most common tuning problems. Tuning inefficient or high-load SQL statements could relate to poorly written SQL, ineffective use of indexes, or suboptimal access paths. Problems such as establishing new database connections repeatedly, excessive SQL passing, and high levels of contention can degrade application performance significantly. These are all poor use of the database by the application, and require a change in the application design. Proper sizing of the System Global Area, or SGA, and the Process Global Area, or PGA, reduces contention for memory resources. Many resources can be accessed only by one process at a time, so several processes attempting to access the same resource creates concurrency. In any database, I/O issues, such as database file layout on disk, can be a source of performance problems. In OLTP applications, the amount of redo and undo generated can create bottlenecks in memory or I/O. Some problems are reported by users, but are brief and not immediately apparent. The Oracle Database has additional tools such as Active Session History or Real-Time ADDM, that allows the DBA to view statistics and metrics over small segments of time in the recent past.

[Heading: Common Tuning Problems. There are a number of common tuning problems. These include inefficient or high-load SQL statements, problems relating to the suboptimal use of the Oracle database by the application, undersized memory structures, concurrency issues, I/O issues, database configuration issues, and short-lived performance problems.]

Many databases will have gradual changes which may lead to degradation in performance. The proactive DBA will capture and save statistics hits from when the database is performing acceptably, to compare with statistics when the database performance is poor to identify the differences. The environment of the database is seldom static, patches, upgrades, or hardware changes can change the performance of the database. Sometimes a change improves performance in one area and causes another area to degrade. While locking issues are not common problems, when you do have them they are important to have a look at. In this lesson, we looked at some of the most common tuning problems.

[Other common tuning problems relate to the degradation of database performance over time, unexpected performance regression following environment changes, and locking issues.]

Tuning Life Cycle Phases

Learning Objective

After completing this topic, you should be able to

- *describe the tuning life cycle phases*

1. Describing the tuning lifecycle

In this lesson, we'll look at the tuning life cycle phases. Tuning will follow the general methodology in all of the lifecycle phases, and different phases of the lifecycle will have somewhat different approaches. Whenever possible, you should start tuning at the application design and development level. A good design facilitates scalability and optimal use of resources, thus mitigating many common performance issues. The testing phase is a continuation of development, with more realistic tests that use production hardware and software. The deployment phase involves adding a new application to an existing system, which inevitably means the workload will change. You should accompany any major change in the workload with performance monitoring. When in production, continue the methodology, and use a test instance to determine whether a potential solution eliminates a bottleneck. Remember to always provide tangible and measurable results to support the solution. And finally, a change of operating system or database version requires thorough testing to avoid performance issues. Tuning during the lifecycle involves two courses of action, proactive or reactive tuning. During the design, development, and testing phases, tuning is mostly proactive. The results are measured and compared against other configurations, which allows for better problem identification and resolution. In deployment and production environments, tuning is mostly reactive. The need for hypothetical loads is removed because actual users and workloads are created, but the ability to anticipate problems also diminishes. You can monitor the database instance to observe changes and trends in performance metrics. From the information you gather by monitoring, you may be able to mitigate performance issues before they are noticed by users.

[Heading: Tuning Life Cycle Phases. An application's lifecycle can be divided into five different phases. The first phase is application design and development. The second phase is testing, which involves database configuration. The third phase is deployment, which involves adding a new application to an existing database. The fourth phase is production, which entails troubleshooting and tuning. The fifth and final phase includes migration, upgrade, and environment changes. Tuning is of two types: proactive and reactive. The aim of proactive tuning is to make improvements to ensure something doesn't break in the future. Proactive tuning activities include testing scenarios, finding problem areas, and resolving the problem. Reactive tuning entails waiting until something breaks and then fixing it. Reactive tuning activities include monitoring an active instance and tuning issues as needed.]

Frequently, the environment of the database instance must change in ways that impact performance. The hardware or operating system must be upgraded or changed, database software can be upgraded to a new version, the disk subsystem is modified or replaced, and even smaller changes such as operating system or database parameter changes can have significant effects. In general, you want to test the current workload on the target system before the change occurs in the production system. The difficulty comes in capturing a representative workload and replaying the workload on a test database. Then it is a question of collecting statistics, potentially resetting the environment to make a change, executing the workload once more, and capturing statistics to compare before and after snapshots. Oracle Database Enterprise Edition has the Real Application Testing option, which includes the SQL Performance Analyzer and Database Replay for overcoming these difficulties. This functionality will be covered in more detail in course 6. In this lesson, we looked at the tuning lifecycle phases.

[Migration, upgrade, and environmental changes can all impact performance. There are seven general steps to follow to better manage the impact of such changes. The first step is to create a test

system. The second step is to capture a representative workload. The third step is to execute the workload against the test system. The fourth step is to collect statistics. The fifth step is to reset the test system and make a change. The sixth step is to execute the workload. And finally, the eighth step is to capture and compare statistics.]

Performance vs. Business Requirements

Learning Objective

After completing this topic, you should be able to

- *evaluate the balance between performance and business requirements*

1. Performance and business requirements

In this lesson, we'll look at balancing performance versus business requirements. More often than not, requirements of the business can impact performance. Frequent checkpoints reduce the mean time to recover, but can increase the number of physical writes. Producing archived redo log files allows for point-in-time recovery, but there is an overhead of reading the online redo logs, writing out copies of those logs, and storing them on disk. And protecting against the unlikely event of block corruption, must weigh up the impact to business should this event occur, against the processing overheads in generating block check sums. Redundancy improves availability, but requires more I/Os. Frequent backups should be maintained in order to ensure recoverability in the event of a catastrophic failure. At the minimum, a single control file is required, but it is common to have two or three. However, more control files require more writes and thus more overhead. Multiple redo log members reduce the chance of loss of data due to a disk failure, but increase the overhead of writing redo. Security measures are often required, and each has some impact on performance, even if it is a very small impact. As an example, many companies have implemented requirements, such as Sarbanes-Oxley, in order to provide proof of accountability and accuracy of information. Auditing access of data, and ensuring sensitive data such as credit card numbers is encrypted, are key components in providing this. Ultimately, the question is not whether to use the security features, but what the business requirement is, and then only use those features that are necessary. In this lesson, we looked at balancing performance versus business requirements.

[Heading: Performance versus Business Requirements. There are a number of factors that can affect performance. These include, frequent checkpointing, performing archiving, generating block check sums, redundancy, and security. Redundancy's impact can be due to the need for frequent backups of data files, multiple control files, and multiple redo log members in a group. Security measures that can impact performance include auditing, encryption, and managing virtual private databases that require fine-grained access control.]

Filing a Performance Service Request

Learning Objective

After completing this topic, you should be able to

- *work with Oracle support to file a performance service request*

1. Filing a service request

In this lesson, we'll look at filing a performance service request. Oracle provides a large set of performance tuning resources. The first and most accessible of which is the database Documentation set. The Oracle Database Performance Tuning Guide covers a wide range of performance tuning issues, and the steps to find a solution. My Oracle Support, or MOS documents, are available to all who have subscribed to support services. A customer support identifier is required to set up a MOS account. And MOS has a large repository of documentation that describes best practices, how-tos, and the diagnosis of bugs and problems. Community discussion Forums are publicly available on the Oracle Technical Network. And is often a good place to look at because of the exchange of knowledge and ideas. Finally, Oracle support will accept Performance Service requests. Oracle Configuration Manager is a tool that is available through MOS, and integrated with Enterprise Manager, to assist customers and Oracle support to get problem solving information quickly.

[Heading: Filing a Performance Service Request. Oracle provides a large set of resources for problem solving. These include documentation, My Oracle Support documents, community discussion forums, and Performance Service Requests made via Oracle support.]

The performance problems you encounter, are often unique to your application and database configuration. Occasionally, the problem is something you cannot tune. My Oracle Support document 210014.1 guides you in logging a good performance service request, and asks questions such as – is the problem instance-wide or query specific? What is the root cause? Is the request to provide Statspack, or AWR reports, as well as operating system statistics during the time when the database is exhibiting the problem, and the baseline when the problem does not appear? If possible, take short period snapshots when the problem is evident. Provide Remote Diagnostic Agent, or RDA reports, and provide SQL_TRACE reports if applicable. Finally, this slide shows a quick screen capture of the aforementioned My Oracle Support note. It provides a straightforward and detailed breakdown of steps to follow, and information to gather and upload, in order for support to assist you in resolving your performance problem. In this lesson, we looked at filing a performance service request.

[There are several requirements for filing a good performance service request. First identify whether the problem is instance-wide or query specific and then identify the root cause. When submitting the service request, make sure to provide Statspack or AWR reports and OS statistics, Remote Diagnostic Agent, or RDA, reports, and SQL_TRACE reports if applicable. A screenshot of the My Oracle Support site displays. There are six tabbed page listed at the top. These are Dashboard, Knowledge, Service Requests, Patches & Updates, Community, and Certifications. The Knowledge tabbed page is selected. It contains a document titled "How to Log a Good Performance Request (DOC ID 210014.1)". Below it is a list of links to the various sections in the document. The four top-level sections are Purpose, Scope, Details, and References. The Details section contains two subsections - A. Database Wide problems and B. Query Tuning Problem. Subsection A contains three sections - "General Database Performance Problem", "The Database is hung", and "A session is hung, running slowly or spinning/A session hogs CPU". Each of these sections contains subsections, such as "What evidence do we require?", "Some good questions", and "Things you can do to help identify the Root Cause".]

Exercise: View Diagnostic Information

Learning Objective

After completing this topic, you should be able to

- *use SQL commands to query the dynamic performance views to view tuning information*

1. Querying dynamic performance views

Now that you've completed the introduction to Oracle Performance Management, it's time to put that knowledge to work. In this exercise, you will list all the SQL_IDs from the v$sqlstats dynamic performance view that have an elapsed time of more than 10 seconds. You will list the wait classes and total number of sessions currently waiting on events associated with those wait classes; identify, navigate to, and view the database instance alert.log file; and finally log in to your local EM Database Express console and navigate to the Performance Hub page. At this point, you can pause the video and perform the exercise. Once you've finished, resume this video and see how I would do it. Now that you've had a chance to complete the exercise, I'll take you through how I would have approached it. As with many such tasks, there is more than one way to achieve the same goal. So if you followed a different approach to what I am about to show you, but achieved the same results, well done. For our first task, we need to establish a connection to the database. In this instance, I'm already logged onto the machine hosting the database, so I'll just connect to the SYS user. Now that I'm connected, I can query the v$sqlstats dynamic performance view to get the applicable SQL_IDs.

[Heading: In this exercise, you will. In this exercise you will list all SQL_IDs from the v$sqlstats dynamic performance view that have an elapsed time of more than 10 seconds. Next you will list the wait classes and the total number of sessions currently waiting on events associated with those wait classes. You will then identify, navigate to, and view the database instance alert.log file. Finally, you will log in to your local EM Database Express console and navigate to the Performance Hub page. Now pause the video and perform the exercise. The presenter switches to a Linux workstation. He clicks Applications on the top-level Linux menu and selects Terminal. The Oracle terminal opens. The prompt is currently [oracle@gc-ol7 ~] At the prompt he enters the command sqlplus / as sysdba. SQL*Plus release, copyright, and connected to information is output. The connection is to Oracle Database 12c ENterprise Edition Release 12.1.0.2.0 - 64bit. The prompt is now SYS@orcl. Next he begins typing the command: select sql_id from v$sqlstats where elapsed_time >=10]

It's important to remember that the data stored in the elapsed time column is stored in microseconds, so we need to do some arithmetic in order to get to a second value. And as you can see, we've only had three SQL statements that have had an elapsed time of more than 10 seconds. Do remember that this view contains information since instant startup and also could potentially contain information about SQL statements that have already aged out of the shared pool. For the second task we will reuse our existing session, but because we are interested in the current waiting sessions we'll need to dive into the v$session dynamic performance view. I've just added the "`where username is not null`" for the predicate so that I can exclude the background processes from my query results. So as we can see on my little virtual machine, things are relatively quiet. We've got three sessions waiting on the idle wait_class and one waiting on network. For task three, we will continue in our sql command prompt and we need to find the value of the diagnostic dest parameter; we will use the show parameter command to do that. And since this accepts wildcard pattern matching we only need to enter in the first few characters. So there we have the value of diagnostic_dest. We can then go out to the file system and from that location, which equates to the Oracle base environment variable, we can follow the standard directory tree, which is opt/apps/oracle for the oracle base, diag/rdbms, db name, instance name, and trace.

[The presenter completes and then enters the command: select sql_id from v$sqlstats where elapsed_time >=1000000 ; The output is: SQL_ID cvn54b7yz0s8u 3un99a0zwp4vd 06g9mhm5ba7tt

Elapsed: 00:00:00:01 Next he enters the command: select wait_class, count(*) from v$session where username is not null group by wait_class order by 2 desc ; A table with two columns is output: WAIT_CLASS and COUNT(*). There are two rows: Idle: 3. Network: 1. He enters the command: show parameter diag A table with three columns is output: NAME, TYPE, and VALUE. There is one row: diagnostic dest: string: opt/apps/oracle. Next the presenter enters the exit command to disconnect from the Oracle database. The command prompt is now: [oracle@gc-ol7 ~] At the prompt, he enters the command: cd /opt/apps/oracle/diag/rdbms/orcl/orcl/trace/ The prompt changes to [oracle@gc-ol7 trace]]

The alert.log file also follows a standard naming convention and that is alert underscore instance name. So now we've located the alert.log file and it can be viewed with whichever OS command you wish. Finally, let's get our web browser up and running. I'm using Chrome in this case, but you could use Firefox or Internet Explorer as well, as long as the Adobe Flash plug-in has been installed. The format of the EM Database Express console page is https, host name, port name, forward slash, em. That brings up the login console and you can connect with a valid database account. Now that we've logged in, we can simply navigate to Performance and select Performance Hub from the drop-down list.

[The presenter enters the command: ls -l alert_orcl.log The output is: -rw-r-----. 1 oracle oracle 102468 Aug 26 15:32 alert_orcl.log Next he enters less alert_orcl.log A number of lines are output. Underneath "Dumping current patch information: and "No patches have been applied", the last three lines are: amount of physical memory in the system pga_aggregate_limit is 2048 MB limit based on physical memory and SFA usage is 771 MB The presenter clicks Applications on the top-level Linux menu and selects Internet: Google Chrome. A single tabbed page opens in Google Chrome and loads the "Introducing Oracle Linux 7" landing page. He enters the following URL in the address bar: http://localhost:5500/em. A page titled "Your connection is not private" loads and a red cross appears over the "https" portion of the page URL. The presenter clicks the Advanced link on the page. Next he clicks the "Proceed to localhost (unsafe)" link that appears in a new section at the bottom of the page. The Oracle EM Express 12c Login page loads. It contains a Login box with User Name and Password text fields, an "as sysdba" checkbox, and a Login button. The page's URL is http://localhost:5500/em/login. He enters a user name and password and presses Enter. The Oracle EM Express 12c Database Home page loads. It contains a number of different sections, such as a Status, Performance, and Incidents - Last 24 Hours. Above these is a top-level ORCL menu with four options - Configuration, Storage, Security, and Performance. The presenter clicks Performance on the top-level menu and selects Performance Hub.]