

The White Papers

A Successful Performance Tuning Methodology

By Azeem Mohamed

Contents

<i>Introduction</i>	3
<i>Overview</i>	3
<i>The Challenge</i>	4
<i>The Performance Tuning Methodology</i>	4
<i>Detect</i>	7
<i>Diagnose</i>	7
<i>Resolve</i>	9
<i>Quest Central® for Oracle</i>	11
<i>Applying Quest Central for Oracle to the process</i>	12
<i>Conclusion</i>	16
<i>About the Author</i>	16
<i>About Quest Software</i>	17

A Successful Performance Tuning Methodology

By Azeem Mohamed

Introduction

Oracle database tuning can be compared to running a marathon: you must train just to attempt the effort. The training is very extensive and many who start it find the effort so intense that they give up. Crossing the finish line takes tremendous energy, time, and physical and mental stamina. Yet just when you start feeling good about your accomplishment, you begin to consider a higher goal. You train more and the cycle begins again. Despite your training, unless you are one of the world's best, unfortunately you experience diminishing returns. This analogy can be directly applied to the efforts and attempts of tuning Oracle for maximum performance.

One common mistake made in the effort to increase the performance of Oracle is to limit your efforts to a single area. Many reports from the last four years have applied the Pareto Principle (aka “the 80/20 rule”) to Oracle performance issues, citing that 80% of the performance problems originate from SQL and should be solved with *SQL tuning* alone. While it is true that many performance problems surface when the SQL is executed, many other factors can cause SQL statements to perform poorly. A request for data can come from many sources, including adhoc queries, applications calls, packages, PL/SQL and materialized views. Essentially they are all running SQL against the database. Generally, when a performance issue is called to attention, it is because a request for data was not serviced in a reasonable period of time. Perhaps a user calls, an alert is sent, or a report indicates contention as a different issue. This often misleads DBAs into thinking the SQL needs to be tuned.

This is a problem because the common mistake of always blaming it on SQL results in a myopic approach to tuning Oracle. The administrator must be aware of all other areas of tuning that need attention and apply some type of tuning methodology that covers each area beyond the insular area of SQL tuning alone. SQL tuning can be a very successful method of increasing performance, but must be used as a part of a larger and more effective performance tuning methodology for Oracle.

Overview

What is tuning anyway? A good place to start is to define a baseline so we have a starting ground from which to improve. Secondly we need to understand what our goal is. What exactly are we trying to achieve with performance tuning? The industry's many performance books, papers, best practices, scripts and seminars generally share a set of performance tuning goals:

- Identify and react to performance problems
- Proactively prevent performance issues
- Meet services levels

- Avoid hardware upgrades
- Ensure the application responds to customer needs

Having the definition of performance tuning will greatly guide us on a path of where we need to start looking when we have this goal of performance tuning in mind.

The Challenge

The Oracle database is arguably the most adaptable and configurable relational database engine available. It has many releases and versions and as a result supports everything from simple personal installations to the most highly scaled application supporting global business architectures. This is a positive and a negative. The positive is that Oracle can support and scale to just about any business model. It can be manipulated to support many of the most complex applications and do so with features that just do not exist in other RDBMS engines. The negative here is that the administrator is tasked with ensuring that these manipulations serve up the data in the most efficient manner possible, even when supporting complex application environments. Over time as the application grows, the tuning efforts only become more difficult. Applications grow and require modifications or upgrades, the user base changes over time and their business needs shift. Additionally, the underlying infrastructure of servers, databases, and operating systems also change over time, adding to the complexity of maintaining sound performance.

Some of the largest problems a database encounters when serving up the data it houses are:

1. Poorly tuned application code: SQL and PL/SQL
2. Contention for internal Oracle resources: locks, latches and buffers
3. I/O Bottlenecks: logical, datafile and disk level
4. Inadequate hardware resources: Disk, CPU, RAM

R. Paquet from Gartner Group [1] summed it up by saying, “at the core of business data for most production applications is a relational database management system (RDBMS). RDBMS monitoring and administration has evolved into a highly specialized market...” This is a highly specialized market because performance tuning is not easy; it is a disciplined practice that should not be done without a strategy. The mission of this paper is to outline a successful performance tuning methodology that is supported by software, which was written by the experts who authored many of the books on performance tuning of Oracle.

The Performance Tuning Methodology

A tried and tested method of managing issues in Oracle is outlined with a simple approach called ‘Detect, Diagnose and Resolve’. This method encompasses all of the steps a DBA needs to take in finding a problem, pinpointing the source of that problem, and resolving the problem. This can be done reactively, in a real-time fashion or

proactively with a more methodical and analytical approach using historical information. Both methods are effective as performance bottlenecks can develop suddenly or can build over time. The goal in the end is to be 100% proactive in your approach to tuning, but getting close to this is a great victory in managing the complexities of applications, systems, disks, I/O, users, jobs, network and other such factors as they interact with the Oracle engine.

The only proper way to implement a successful performance tuning methodology is to follow this set of steps:

Plan:

1. *Profile each database instance* Understand each Oracle instance and what type of application this instance supports. Data warehousing applications will have completely different goals from an OLTP environment. As a result of differing goals, each instance may need to be tuned using a different approach.
2. *Set a baseline for each database instance and* know how your database is performing today. This baseline may be acceptable or unacceptable in terms of your goals, but you need to have a starting point to understand how your tuning efforts are impacting performance.
3. *Set the goal of your tuning efforts.* Going back to the goals mentioned earlier, are you trying to solve an immediate problem, proactively prevent problems, meet service levels, or ensure users response time? You should know the specific goal and the level you wish to achieve with that goal, for example if your SLA guarantee's 90% availability or end user response time will be less than 5 seconds.

Now that you've initialized your tuning environment, it's time to start tuning. At this point your methodology will vary slightly based on whether you are tuning in a proactive or reactive mode. However, the techniques will be similar and the end results will be the same. So let's start tuning:

Detect:

4. *When reacting to immediate performance issues* you need a mechanism that allows you to determine where in the environment your performance problems are occurring. This could be the operating system, the Oracle database, or a SQL statement that requires tuning. If you don't have a monitor to identify these problems, handwritten scripts can be used, however, it may slow down your ability to detect the issue. Look for thresholds that have been exceeded to determine where the performance problem is arising.

5. *When proactively tuning to determine potential performance issues* you will need to reference historical data to determine performance trends and ensure you are not seeing a decline in performance that will fall below accepted levels. If you own a monitor it will typically contain a historical repository. If not, create a repository where you can store your script results, so you can review your history and compare different timeframes for trending.

Diagnose:

6. *Drill into real-time and historical information.* Typically, the immediate issue is not the root cause of the problem and analysis of additional metrics will find the root cause. An example of this is buffer cache. If the buffer cache falls below acceptable levels, typically it is not the size of the buffer cache, but perhaps a user running a long running transaction consuming the buffer cache that is the issue. It's important to find the root cause of the problem.
7. *List your tuning options and determine the benefit of each one.* Your analysis will provide you with a list of tuning efforts; however, you must choose the tuning effort that provides the biggest benefit to your environment. In addition to providing the biggest benefit, you may also want to implement some tuning changes that may not have as big a benefit, but are fairly trivial in terms of effort.

Resolve:

8. *Select the tuning option you wish and implement* If this tuning is being done reactively, begin with the real-time issues and make certain you resolve those issues that are affecting the user immediately. If this tuning is being done proactively, choose the tuning options that provide the highest benefit in terms of achieving the goals you have defined for this instance. Details on the different tuning options you may want to implement are covered later in this paper.
9. *Measure the performance of your system* after applying the tuning change. This will validate your tuning solution has improved the overall performance of your Oracle instance. Once you've determined this change has made a positive impact on your system and you decide to retain this change, you should set a new baseline. This will also help in analyzing new performance issues.
10. Repeat steps 4 through 9 on a regular basis. You can almost schedule this as a task. When the application or environment changes, you may need to set new baselines. You also need to be aware that every time you resolve issues through tuning, the change you make also affects the baseline of measurement that you are now tuning against. Tuning, in and of itself, gives reason to set new goals. Repeat the measurement of performance and tune for new issues found.

Now that the roadmap used for the overall tuning methodology has been laid out, the following section will drill into each step and describe what to look for and how to perform each step successfully.

Detect

The detect portion of the tuning methodology includes the gathering of performance metrics and analyzing those metrics looking for thresholds being exceeded. You must first determine where in your environment the problem is occurring. There are 3 types of collections that are suggested for Oracle database tuning.

- Operating system metrics - You will need to collect information about both the machine resources such as memory and CPU, as well as I/O usage. A monitor can be used to gather this information or standard O/S commands can be issued to gather this information. Suggested statistics are:
 - I/O – Disk Time, Disk reads per second, disk writes per second, disk queue, file space used
 - CPU - % of processor busy time, interrupts
 - Memory – Available memory, pages swapped per second, swap time, queues
 - Network Collisions, network utilization
- SQL collection - For accurate SQL and transaction metrics, you will need to capture SGA information at sub-second intervals. The smaller the interval the more SQL you will capture. Suggested statistics are: SQL Text, OS User, Oracle User, originating program, logical reads, disk reads, CPU.
- Database metrics - Database related metrics would also need to be collected from the V\$ tables and alert logs in Oracle. Suggested metrics are:
 - SQL*Net statistics – active users, active sessions, average response time
 - Background processes – Redo log writer, DBMS writer, archiver
 - SGA – Buffer cache utilization and hit ratio, keep and recycle pool usage, redo buffer usage, shared pool usage, sorts
 - I/O - Redo log statistics, I/O events, wait events, object and file growth, locks, latches
 - Log - alert logs messages

This information should be collected for both immediate problem resolution and also stored in a repository for historical trending. Look for and tracking performance metrics that have exceeded defined thresholds. In addition to understanding when and why a threshold has been exceeded, it is also important to track the trend of those events and the interval and frequency of those occurrences.

Diagnose

The diagnostic portion of the tuning methodology is specifically designed to drill into additional metrics to understand why a particular threshold has been exceeded. This white paper is focused on Oracle database tuning, so the diagnose portion will not

include diagnostics necessary on the operating system. This phase is more difficult than detecting the issue, since you now need to understand where and how to drill into contributing metrics to speed up your time to recovery. When diagnosing the source of the problem you need to understand the cause and effect that each component of the Oracle database has on other components. This knowledge can only be gained with experience or by referencing one of the many tuning guides available either online or published in hard copy. When trying to diagnose your Oracle database, the following metrics should also be analyzed to determine the root cause of the problem:

Here are just a few examples of the information you should be looking for and trying to correlate for the root source of a database related problem:

Latches and Locks - Blocking locks, Latch activity, Session locks

I/O - Logical I/O, Physical I/O

Oracle wait information - Session wait events

Session Information - Session SQL, Session activity

Rollback activity - Rollback segment information

Network activity - Oracle Net status and user activity

Caching - Library cache, Dictionary cache, Buffer cache hit and miss ratios

Redo logs - size and number

Memory – Sorts, Memory usage and allocation, SGA detail

Disk - Sorts, reads, writes

Alert logs, Parallel Server activity, Cursor usage

Space Management - Space allocation, Space used and available, Extent information, Init.ora parameters, Object location and usage, Indexing and key relationships

And others...

Correlating the above gathered information can be time consuming, full of effort and without reward, unless you organize your efforts and step through them in a certain order.

Let's outline the three main categories of diagnostics

1. Diagnosing SQL issues

There are many ways to diagnose SQL related issues. You should begin with examining the V\$SQL view, which shows all of the cached SQL statements. You should look for SQL statements which consume a large portion of the buffer gets. You should also be looking for SQL statements, which have a relatively high buffer_gets/executions ratio. When trying to diagnose your SQL statement syntax for problems, use the Explain Plan and associated analyze statistics to determine whether the optimizer has chosen the most efficient path for your SQL statement. In this process you should use TKPROF and identify operations that have a particularly high 'rowcount'. If the rowcount exceeds the number of rows the table or index, you can be certain there is a problem that needs to be resolved.

2. Diagnosing Contention

When examining contention, you should begin by looking for certain values the V\$System_Event table. The columns in the v\$system_event dynamic performance view are:

EVENT: This is the name of an event such as **enqueue wait, buffer busy waits, latch free, db file scattered read, db file sequential read, free buffer waits, etc.**

TOTAL_WAITS: This is the total number of waits for a given event.

TOTAL_TIMEOUTS: This column provides the total number of wait time-outs for the specific event.

TIME_WAITED: This is the total wait time in centiseconds (1/100ths of a second) for all sessions for a given event.

AVERAGE_WAIT: This is the average wait time in centiseconds (in 1/100ths of a second) for all sessions for a given event. $Average_Wait = (time_waited/total_waits.)$

As you examine the information contained within this table, you should eliminate any idle events such as SQL*NET waiting for client, etc. You should then calculate the amount of time spent waiting for other remaining events. The examples of wait events listed above should not be contributing to total wait time and can be tuned accordingly.

3. Diagnosing I/O problems

IF you have tuned SQL and resolved contention but still have performance issues, the logical I/O on your database will be near normal, but the I/O is not optimized. This presents us with the opportunity to reduce the percentage of I/O that requires a disk read. Start by identifying which disks have a disproportionately heavy amount of activity. For example, if one disk in a twelve-disk system experiences 25 percent of the I/O (measured by the number of reads, writes, or both) that disk is hot and the database needs to have the I/O optimized. Once you've identified the hot disks, search for the files and tables on the disks that experience most of the activity. The goal is to then move these files and tables to less-active disks as needed. To properly identify hot disks and locate the associated files and tables, you will need to run data-collection utilities, such as Oracle's utlstat and utlstat and the UNIX iostat utility. This collected information should then be used to pinpoint the root source of excessive I/O.

This may seem like a lot of information to go through when understanding a single issue. Once you have a monitor or the right scripts, it's a matter of associating the identified performance issue with corresponding detailed statistics to determine the root cause of the problem. With the right scripts and the mechanisms in place to execute those scripts, it's a matter of reviewing those script results looking for the exceeded threshold.

Resolve

By this phase, tuning should be focused on the exact resolution for a specific issue that was not only detected, but also thoroughly diagnosed and investigated. Resolution may be done at the operating system level, database parameter level, user level, space level, object level or by tuning the individual SQL statement. Some of the best practices in Oracle tuning include:

1. **Operating system tuning** should include focus on CPU cycles, RAM allocation and Network utilization
 - **CPU Cycles** - The amount of CPU cycles available can contribute to slow SQL execution times. Whenever the run-queue exceeds the number of CPUs on an Oracle server, it becomes CPU bound.
 - **Available RAM memory** - The amount of available RAM memory for Oracle will affect the performance of SQL, especially in the data buffers and in-memory sorts. The addition of dedicated RAM will help improve performance to a certain degree.
 - **Network** - Large amounts of Oracle Net traffic will contribute to slow SQL performance. This can be monitored and managed accordingly.
 - **Monitor and manage Disk I/O** – Object access and location of those objects are directly related to the use of disk devices. Proper usage of RAID configuration will help to balance I/O operations. Datafile, OS and Oracle wait information is needed to accurately pinpoint issues and subsequently manage and resolve such issues.

2. **Database tuning** will most likely focus on Init.ora parameters and space management including reorganizations as well as user and object usage patterns.
 - **Initialization Parameter settings** – The init.ora parameters are static upon installation of Oracle and need to be configured for the specific use case of that particular database. These parameter settings change with the different versions of Oracle, but more importantly change with the different types of applications that run on the database. As the database grows or changes form, we must be cognizant of the possibilities of modifying these parameters.
 - **Space Management** - When reorganizing a table, we remove or reduce extent fragments, coalesce chained rows, re-build the freelist chain, change object parameter and size and re-sequence table rows. These activities can all increase the performance of the Oracle database, by reducing I/O and increasing the efficiencies of SQL.
 - **I/O Optimization** – Some key techniques in this area are to optimize the size of your buffer cache, use effective keep and recycle pools and keep your scanned tables as small as possible. Additionally, when optimizing I/O, you should also look to make certain that there is a sufficient number of disks allocated for the workload and spread the I/O evenly

among these disks. Typical disk devices can do about 50 I/O's per second, but you should get a reading on what your specific devices can handle. In this case, if you divide your physical I/O rate by 50, you will have an estimate of the number of disk devices you will need for optimum I/O.

3. SQL Tuning will focus on either rewriting the SQL statement, index tuning, or session parameter changes.

- **SQL Tuning** – The rewrite of SQL statements or usage of an index to change the access pattern of that SQL, will greatly improve the performance of the SQL, while returning the same result set.
- **Index tuning** – Inclusive of identifying full table scans and other indexing opportunities as well as unused indexes. Adding an index where needed can significantly reduce the time spent reading a large object for a limited amount of information.
- **User and Object usage patterns** – There are many user and session settings that can be enabled to improve performance. Additionally, there are many ways to manipulate the object usage patterns by Oracle, such as indexing, rewriting SQL, location of objects, tablespace usage and other such variable settings either permanent or temporary.

It is important to include tuning of the operating system since the performance issue may never be resolved by tuning solely at the Oracle database level or simply with rewriting the individual SQL statement. Any operating system issues should be eliminated first prior to any focus on database or SQL tuning. Once you are ready to tune the Oracle database or the individual SQL statement, you need to be aware of all of the areas that can increase performance and then list and rank those that will bring you to your performance goals the quickest.

Quest Central™ for Oracle

To employ the best practices of tuning and the step-by-step approach of *Detect, Diagnose and Resolve*, Quest Software has packaged a solution to simplify and facilitate the execution of a successful performance tuning methodology for every DBA. Quest Software offers Quest Central for Oracle, an integrated database management tool that provides all the functionality that Oracle DBA's need to proactively manage their database environments all day, every day.

Quest Central for Oracle is a workbench for managing database administration, as well as performance and availability of Oracle. The components found within Quest Central

fall into four categories that are tightly integrated for maximum efficiency of employing a tuning methodology.

These four components are:

1. Performance Diagnostics – Real-time and historical information
2. Database Administration - common database management tasks, data browsing, user and object management and script editing.
3. SQL tuning – Automated collection of SQL, Auto-scripting, advice, scenario testing, execution plans, comparisons and impact analysis
4. Space Management – Object analysis, reorganization scripting, exception reporting and capacity planning with object trend analysis

In the process of managing an environment with multiple databases, a DBA must keep information on each database and the specific use for that instance. With several databases to manage and several configurations with the many users and programs that are affected, a DBA is challenged with tuning each environment for its' specific needs and doing so efficiently. Quest Central for Oracle is built to provide maximum efficiency in database management by providing a single console to manage multiple databases across a company's entire architecture.

Applying Quest Central for Oracle to the process

This section will uncover the successful performance tuning methodology through the use of Quest Central for Oracle.

Planning - Step number 1 in the successful performance tuning methodology is to profile your environment. Quest Central contains a component to analyze your database, and assist with diagnosing and resolving performance problems. The Database Analysis component can be used for both immediate, but more importantly, proactive performance tuning. Built into this Database Analysis component is a 'profiling process' that allows a DBA to input different variables for each database they are managing in each of these configuration settings. The settings included in the profile are: Alternated Redo devices, Available RAM, Datafile RAID status, Datafile disk devices, Dedicated archive devices, Dedicated redo devices, Raw devices, Third-party applications and what type of application is running in this database. As a result of modifying the various parameters for each database, Quest Central will take into account various operating system metrics when producing Oracle and SQL tuning diagnostics and resolutions.

Planning - Step number 2 in achieving your highest performance in a tuning methodology is to set a baseline by which you can measure performance over a period of time. In an effort to understand the desired performance of a database and expose issues, small and large, there must be a baseline of standard performance. If a baseline is not set, you have no way of ensuring performance is 'normal' or experiencing a slowdown. With Quest Central for Oracle, a component exists to manage Performance

Diagnostics, known as Spotlight on Oracle. Upon initializing Spotlight, you will be asked to ‘calibrate’ your system. It is recommended that you calibrate during normal processing time. Calibrating your system, equates to setting a baseline. This calibration process automatically sets the thresholds against which performance will be measured. This baseline will be used to determine when performance metrics have exceeded what’s considered ‘normal’ for that environment.

Planning - Step number 3 in route to success in performance tuning is to set the goal of your tuning efforts. The key to setting goals is to understand what the expectations are of the users on that system. If this is a data warehouse application, slower response time may be acceptable as long as the data retrieved is accurate. In an OLTP application, response time may be the number 1 goal. The Oracle database parameters can be tuned to meet whatever goal it is you are trying to achieve. The Database Analysis component will setup a list of goals for each Oracle instance based on the type of application running in that instance. The profile will help better define what those goals should be. These tuning goals will help set the standard by which a database should run and how it should be tuned.

Now that you have completed the planning phase of your tuning methodology you are ready to start tuning by identifying potential performance issues.

Detect - Step number 4 is to identify real-time issues on your instance. If you have a 24 x 7 alerting system, you should have the ability to receive alerts from a threshold being exceeded well before a user recognizes that there is an issue. This can be done with scripts or with an automated solution. If you do not have either of these in place, your detection will be done via phone calls from users complaining of response time.

In addition to some type of alerting system, it is important that you are able to get an overall view of your entire database and how the data is flowing through that database. Quest Central Performance Diagnostics or Spotlight is the perfect tool to get this view. In addition to seeing the database engine running, the color-coded areas indicate a bottleneck and allow you to drill into specific areas to assist with the diagnosis. Quest Central for Oracle provides 2 methods by which to gather information for real-time problem detection. The Diagnostics (or Spotlight) component uses a client-based solution, creates a connection to the database, retrieves the information, and sends that information back to a client machine for viewing. In addition Quest Central for Oracle also offers agent-based technology that polls the System Global Area of Oracle. This allows the user to view real-time (see figure 10) and historical events without intrusiveness to Oracle itself. This agent has sub-second sampling that can capture much more detail with less overhead than a script running inside of the database. Additionally, since this agent runs outside of the database, it can continue to gather information even when your database is locked up due to heavy usage.

Detect - Step number 5 of the performance tuning methodology provides another method to detect problems and that is to analyze historical data looking for any trends that may lead to poor performance.

The Database Analysis component of Quest Central can be used to capture performance metrics on a regularly scheduled basis, store those statistics in a repository, and format the output so trending information can be easily viewed. Additionally, the agent-based technology included in the SQL tuning component with Quest Central for Oracle, a user has the ability to collect SQL text and their statistics without any impact on the database engine itself.

Diagnose - Step numbers 6 in successful performance tuning is drill into your performance metrics and locate the root cause of the problem at hand. If you are tuning in a reactive mode because a user called complaining about response time, you may wish to take advantage of the Spotlight component and drill into the Top Sessions display for a list all users and their related performance statistics, additionally you may also want to pull up the SQL Text to see exactly what they are running.

If you are tuning reactively or trying to determine a user problem that occurred earlier in time, use the SQL Tuning component to view all user and SQL history to determine trends or pinpoint a user consuming heavy resources. The SQL Tuning display can find users consuming CPU, memory, obtaining many locks or latches, or experiencing heavy I/O.

But remember, not all performance issues are SQL related. If you are tuning your system proactively or just trying to get an overall view of the health of your Oracle system, you will need to view your historical data to find trends within your system. The Database Analysis component can be setup to collect statistics in regular intervals and store that information in a repository. You can choose the collections you wish to include for historical viewing and generate reports indicating the health of your Oracle system for that time period.

Diagnose – Step 7 Once you've chosen the collections you wish to analyze, the Database Analysis component will analyze your statistics and produce a list of tuning options, along with their benefits. Additionally helpful tuning guide information is included to help you understand this tuning option and what the optimal parameters are for that tuning option. Since there may be several ways to tune in order to resolve a problem, the may want to consult the advice of experts to choose the right tuning option. Built into Quest Central for Oracle, is expert advice from best selling authors of Oracle performance tuning books (Eyal Aronoff and Guy Harrison).

Resolve - Step number 8 while engaged in the successful performance tuning methodology is to resolve your issues using the tuning options and implementing the option you feel would give you the most benefit. When using the ranking system

provided in Quest Central for Oracle, a user has the ability to decide how to approach tuning efforts.

Once you've determined the option you wish to implement, you will need to determine whether this tuning effort will positively benefit your Oracle environment. In the example below, the Performance Diagnostics (Spotlight) identified a poor performing SQL statement. That SQL statement can be brought into the SQL Tuning component and the resolution tested out immediately to see the results it would have on your system.

In our working example, a SQL statement was found to be the issue and is placed into a tuning lab environment to view the execution plan along with statistical information on the objects involved in the execution.

In another example, the Database Analysis component suggested we modify the `SESSION_CACHED_CURSORS` parameter to 20 based on the activity found within the statistics collections. The script was generated from that component and run within Quest Central.

In yet another example, the Database Analysis component pointed out that a reorganization of certain objects would provide more efficient access to the data. The Space Management component not only creates reorganization scripts, but can also rank each object within your system so you can wisely choose the object that would gain the most benefit from reorganization.

Here we have a script that uses the CTAS method of reorganizing to re-sequence the rows in a table and reduce the row chaining as well as recover any wasted space. There are several ways to modify the script to use options for efficiency, including changing of the object size, block size, location of the object and other various parameters during the reorganization.

In a final example of tuning your database you may come to the realization that your database is trying to satisfy too many needs and the best resolution would be to offload your read-only activity to a replicated instance of Oracle. This is done to alleviate the contention between OLTP and reporting activities. Using Quest's SharePlex product, your users will immediately benefit, as the contention for resources will be eliminated. OLTP processing will not have additional buffer busy wait events and the reporting database will not have to wait on any locked objects. Quest Central for Oracle also provides a 'plug-in', which allows you to view the flow of data that SharePlex is replicating.

Resolve – Step number 9 in successful performance tuning is to measure the performance of your system against the baseline you had set in the planning phase. This step is often skipped, simply because attention to the tuning efforts just implemented are often moved to the next issue that arises. It is entirely possible that the tuning effort

implemented either did not yield the desired results or caused a secondary issue to surface. Using the Performance Diagnostics (Spotlight) component, you will immediately determine by the color-coded display whether you still have a bottleneck in your system.

Using Quest Central for Oracle, you have the ability to view your database immediately in a graphical interface to diagnose the newly tuned environment and seek the desired results of the tuning efforts.

At this point you've resolved the current performance issue, *however*, tuning is an ongoing effort that continues as any and all changes to a database take place. Without creating a list of tuning options and analyzing those, you may believe that your tuning effort has fixed the largest issue found, but it may have only been one of several tunable parameters of Oracle that needed to be adjusted for better performance. Once you have implemented one piece of tuning advice and viewed the results of your efforts, remember to set a new baseline. Next you are ready to move to the next piece of advice on your tuning options and determine what kind of performance boost that will give your system. With Quest Central for Oracle, you simply move to then next item on the list and follow the recommendation and use your goals as a guideline for future tuning efforts. The end goal is to have proactively achieved your tuning goals, maintained maximum availability and built a service level that exceeds the users' expectations.

Proactively following the 'detect, diagnose, resolve' paradigm makes you aware of new issues as changes occur in your environment.

Conclusion

Tuning of the Oracle database is not an easy task and requires a great deal of interaction and knowledge of the environment that surrounds the Oracle engine. Tuning involves diagnosing the operating system, the oracle database, and the SQL running within that Oracle database. When it comes to managing many of the performance tuning initiatives, Quest Central for Oracle offers a complete workbench of best-of-breed components that will help automate collection of information, provide guidance, goals, recommendations, advice, resolutions, scripts and best practices with a comprehensive reporting system to facilitate a successful performance tuning methodology. From profiling to collection of data to reports and resolution, Quest Central for Oracle will provide the accuracy you need to feel confident as you employ the repeatable and successful performance tuning methodology as outlined within this document.

About the Author

Azeem Mohamed is a Product Marketing Manager for the Oracle solutions at Quest Software Inc. He has over 8 years technical experience working with Oracle, holding positions from DBA, to Senior Systems Engineer and High Availability Architect. As an Oracle Certified Professional, Azeem has both applied database management



experience as well as acquired knowledge from working closely with the experts and well-known authors (Eyal Aronoff, Guy Harrison, Dan Hotka, Gaja Vaidyanatha, Mark Gurry and Steve Feuerstein).

About Quest Software

Quest Software, Inc. (NASDAQ: QSFT) is the leading provider of application management solutions. Our software gives businesses confidence that their vital applications will be available and performing well — while simultaneously driving down the costs of managing them. By focusing on the people and technology that make applications run, Quest Software enables IT professionals to achieve more with fewer resources and get the most out of existing application investments. Founded in 1987 and based in Irvine, California, Quest Software has offices around the globe and more than 1,650 employees. And approximately 10,000 business customers worldwide.