

Chapter 10. Tuning Oracle and RAC on Linux (*excerpt*)

Publisher: Oracle Press

Title: Oracle Database 10g Linux Administration

ISBN: 0072230533

Authors: Edward Whalen, Wim Coekaerts

Tuning Oracle is as much an art as it is a science. Tuning involves scientific investigation, good note taking, intuition and experience in order to determine if there is a performance problem, what that performance problem is, and how to solve it. Performance Tuning is made easier by developing and sticking with a disciplined methodology. By having a methodology that you stick to, you can experience more predictable and reproducible results. In this chapter I will introduce you to the methodology that I use, but this is something that you should change in order to suit your own needs.

Performance tuning involves taking a system that is performing poorly and making it perform better. This sounds simple enough, but when there are numerous components involved and many layers of applications, hardware and databases, it can be a very complicated task. It usually involves monitoring and tuning at several layers including:

- Application Tuning
- Instance Tuning
- Hardware Tuning

In addition, architectural design changes might need to be made as well. In this chapter you will be introduced to all layers of tuning. In order to begin this introduction, you will first learn about my personal tuning methodology.

Performance Tuning Methodology

This tuning methodology involves a number of individual steps that leads to a final result. This result might be information or it might be a recommendation for a change to modify the way the system performs. The individual steps for the performance tuning methodology are:

1. Initial Assessment
2. System Monitoring
3. Results Analysis
4. Hypothesis Creation
5. Solution Proposal
6. Implementation
7. Testing and Monitoring
8. Go to Step 3

This methodology might not work for everyone, but it is a good starting point. In addition, there are a number of tips and techniques that can be used in order to improve the effectiveness of this methodology that are given below. Let's look at the methodology in a little more detail.

Step 1: Initial Assessment

The goal of step 1 is to understand the environment that you are evaluating. If you are tuning your own system, this step might be something that you are quite familiar with. If you have been brought in to tune someone else's system, this might involve a little more work. In either case, it is useful to document your assessment of the system. Some steps involved in the initial assessment are:

- Learn about the application. It is important that you have a basic understanding of what the system and application is and how it works.
- Ask about how it works and what it does. You should be able to find someone who can provide information about how the system operates, what the various components are and how they interact.
- Determine what the reported problem is. If a problem has been reported, determine exactly what the reported problem is. Often the reported problem is "Everything is slow". Getting more details could help the process.
- Document Database size, tuning parameters, etc. Before you start tuning the system you should document the size of the database, Oracle and Linux parameters, etc..
- Validate parameters. Rarely tuning problems are solved in the first few minutes of the engagement by finding a improperly set parameter. This is rare, but it does happen.

This is a very important stage of the tuning exercise, but the least glamorous. By getting your documentation and information lined up in the beginning, you might have already set in motion a successful project.

Step 2: Monitor the System

The goal of this phase is to monitor the system and gather data. This data can then be used to develop a theory on what might be causing the performance problems that you are experiencing. As with Step 1, it is important to document your results so that they can be properly analyzed and repeated as necessary. Some of the tasks involved include the following:

- Monitor the System using:
 - OS Tools: Sar, vmstat, iostat. These tools can give you valuable information on how the system itself is performing at the OS level. High level problems might be detected with these tools, such as CPU utilization at 100%, an overloaded I/O subsystem, etc..
 - Oracle Tools: Statspack, dbastudio, v\$views. These tools can provide information on the performance of the Oracle instance and wait events. These tools can provide information on problems within the Oracle instance and application.
 - 3rd Party Tools: Third party tools such as Veritas InDepth can provide information on how the application and SQL Statements are performing.
- Analyze OS and Oracle configuration parameters. Analyzing OS and Oracle configuration parameters might highlight a fundamental problem that can be easily solved.

Performance monitoring is much more exiting and interesting that documenting the system configuration, however, documentation is equally important here. If you are saving performance data on the system be sure to make notations on when that monitoring occurred and what was happening on the system at the time.

Step 3: Analyze Results

The goal of the analysis phase is to take the performance monitoring results in conjunction with the initial assessment and use this data to formulate a hypothesis in the next step. Some of the tasks involved are:

- Analyze data. The data that you have collected during the monitoring phase should be carefully analyzed. You might be surprised at what you find, or it might confirm some suspicion that you might have.
- Review error logs. This is one task that is often overlooked. The error logs can be a valuable source of information in the event of some type of configuration error or hardware failure.
- Look for performance data from customer sources. This is where the user and/or the tester have their opportunity to give you feedback on how the tests went. This might also be where the application testers can offer helpful information.

Once you have completed the analysis phase, you are ready to start forming a hypothesis on what the problem might be and how to solve that problem.

Step 4: Create a Hypothesis

The goal of this phase is to develop a theory or hypothesis on what is causing the performance problem. Creating a hypothesis could be very simple or quite complex. In the case of multiple problems, you might just be creating a hypothesis for one or two problems. Remember, you don't have to solve all of the problems in one shot; in fact, you are better off changing only a few, or one thing at a time. The tasks involved include the following:

- Determine if there actually is a problem. Some times you come to the conclusion that the system is running as designed, and there really isn't a problem. This could be the hypothesis that you come up with.
- Formulate a theory. Formulating a theory is the most important part of the tuning art. Here you have to take in all of the data that you have gotten and try to determine why you are having the problem, if there actually is a problem. The types of problems that you might theorize include I/O problems, locking problems, poorly formulated SQL statements, etc..
- Back that theory up with data. A theory with no supporting data isn't worth very much. Even if the data isn't conclusive, there should be some indications from the data to support your theory.

The hypothesis is essential for proposing solutions. If you can't determine what you think the problem is from the data that you have gathered, you need to go back and gather data that is more useful.

Step 5: Propose a Solution

The goal of this phase is to come up with a proposed solution or a test in order to determine what the actual solution should be. In some cases the solution might be to purchase more hardware, such as disk drives or memory. In other cases, the solution might be to tune some SQL statements or add an index. This phase is where you get to actually propose the tuning solution and it consists of these tasks:

- Develop a solution. This solution could involve adding indexes, changing SQL statements, changing the hardware layout or adding more hardware.

- Develop a validation plan. The validation plan should include some procedures that can validate that the change has been made correctly.
- Document expected results. It is easier to run a test when I have some expectations on what the results should be. This allows you to determine if the test might be flawed in some way. If you don't get the expected results, it is either the test is flawed or the solution is flawed.

When proposing a solution, you should always have the result of that solution in mind. If you don't know what the proposed solution is going to do, then why do it.

Step 6: Implementation of Solution

This step is the really fun part. The goal of this phase is to actually fix the performance problem. Here is where you get to make that changes that you have proposed and get ready to test the solution in order to validate your hypothesis. Implementing the solution could be as easy as adding an index, or as difficult as changing out the entire storage subsystem. In any case, this is where the action takes place and can involve some of the following tasks:

- Make the hardware change. This might involve upgrading hardware, changing the I/O subsystem or replacing the entire system and migration to a new system.
- Make the configuration parameter change. This is usually pretty easy and doesn't have too much risk involved in it.
- Add an index. Adding indexes can be quite time consuming on large tables. The time it takes depends on the data and the complexity of the index.

It is important when making changes to follow a few simple rules. These rules will make your job easier and give you better results in the end:

- Document your changes. By documenting your changes you will be better able to reproduce them and be able to determine which changes helped.
- Only change one thing at a time. Multiple changes at a time lead to unpredictable results.
- Document your changes. This is in the list twice since it is so important.

By being methodical and careful you can achieve great results.

Step 7: Test and Monitor

This phase involves running validation and performance tests against the changed system. The goal of this step is to validate the solution that you have executed in the previous step. This validation either involves confirming that the change made an improvement, the change made things worse, or in some cases that the test is flawed.

- If possible, test the change in a non-production environment. This requirement is sometimes impossible, but should be a goal of all tuning. In any case, the risk of the change should be evaluated and a determination of how to test and implement the change should be made. For example; adding or changing an index could affect performance, but not the resulting data returned by a query. A change to a query itself could return incorrect results.
- QA the change before deploying. Any code change should have quality assurance tests performed on it before it is deployed.

- If possible, run load tests. Load tests can help you to measure and document the performance gain from the change. The better the test, the better the results.
- Monitor the changes using the monitoring tools described in step 2. If you use the same tools and monitor in a similar fashion you can compare the results with those collected in step 2.
- Document the changes. If you don't document what you did, it might be very difficult to reproduce the change and to determine how it affected performance.

It is important that changes be made carefully and methodically. Too many changes at a time can lead to unpredictable results. The effect of one change might change the effect of another change.

Further Analysis and Testing

You should now pick up with step 3 and repeat until you are satisfied with the results of your tuning or you are out of time or budget. Unfortunately tuning is like cutting grass, it always grows back. The end point of tuning is when one of several events occur:

- The performance goals have been met. This could be a certain average response time, batch jobs running within a specific time window, etc..
- Time has expired. Sometimes you only have test equipment for a limited time frame and you must end testing when that time frame has been met.
- Budget has been used up. Many performance tuning exercises have a specific budget and once those funds have been used up, tuning must stop.

These and many other reasons could dictate the end of the tuning engagement. Hopefully once you are done the results are better performance or a recommendation on how to improve performance.

But how do you actually tune the system. That is the subject of the remainder of the chapter. This will start with monitoring and tuning the Oracle instance.