Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

VOL. 18, No. 4 · NOVEMBER 2004 $15

Take a Tour of Computer History

at NoCOUG's Fall Conference

Renew Your NoCOUG Membership for 2005

Membership runs on the calendar year. That means that it's time to renew your NoCOUG membership for 2005. Just go to www.nocoug.org/join.html today!

Grids Are Not Just for Data…

and Other Insights, featuring Craig Shallahamer. You'll also find a review of his popular class on Oracle performance.

Oracle 10g Upgrade Experiences

Learn what happened when a local DBA took the plunge.

Oracle Performance Troubleshooting

Check out this book review by Brian Hitchcock.

Much More Inside . . .

It's a Wrap!

Working on the *NoCOUG Journal* was especially fun this year. Now that the year is coming to an end, I'd like to send out a big "thank you" to all the Oracle professionals who have taken the time to share their knowledge and experience by writing for the *NoCOUG Journal*. I have to especially thank a couple fellow NoCOUG members—Brian Hitchcock and Chris Lawson—who contributed to almost every issue this year. They have offered an enormous contribution to the *Journal* in the past year. I'd also like to thank the NoCOUG Board and the members I meet at the conferences who offer their ongoing feedback.

My wonderful co-editor, Laurie Robbins, continued to be part of the NoCOUG volunteer team. Working together, we've made some positive changes to the Journal, such as:

- more original articles from great writers
- feature interviews with Oracle luminaries
- a more professional layout with added color (without an increase printing costs)

As always, your comments and suggestions are welcome. Please send them to journal@nocoug.org. We look forward to seeing you in 2005!

—Lisa Loper, *NoCOUG Journal* Editor

Correction: The interview article of Tom Kyte in the August, 2004 issue incorrectly stated he was a member of BAARF. Tom Kyte is NOT a member of BAARF.

Table of Contents

Publication and Submission Format

The *NoCOUG Journal* is published four times a year by the Northern California Oracle Users Group approximately two weeks prior to the quarterly regional meetings. Please send your questions, feedback, and submissions to: Lisa Loper, *NoCOUG Journal* Editor, at journal@nocoug.org.

The submission deadline for the upcoming February 2005 issue is December 20, 2004. Article submissions should be made in electronic format via email if possible. Word documents are preferred.

NoCOUG does not warrant the NoCOUG Journal *to be error-free.*

NOCOUG BOARD

ADVERTISING RATES

Too Busy for Networking?

"Waiting until you are without work to think about networking is like waiting for a disk crash on your server before developing a backup and recovery plan."

Roger Schrag

I had lunch with a sysadmin friend of mine the other day, and he asked me how I stay so busy with work. I told him that networking is the key for me. My friend was asking because his project had ended recently and he found himself without work for the first time in many years. But unfortunately for my friend, he rarely took the time for networking. "I was too busy to stay in touch with people," he explained.

I've heard this one before, and it makes me sad every time. Waiting until you are without work to think about networking is like waiting for a disk crash on your server before developing a backup and recovery plan. It's one of those things that many of us "techies" don't particularly enjoy, yet it is so important for our career development. Peers are a great source of valuable information, whether you want to learn when to use bitmap indexes or what skills are the top priorities for that DBA job opening at Acme Widgets. Plus, having those established relationships with people in your field makes work more enjoyable.

Whether you are a consultant or a full-time employee, it is always the right time to be building and maintaining relationships with colleagues in the industry. It is always the right time to keep up with others on trends, best practices, and new developments. And if you should find yourself without work, you'll have a network to draw upon for referrals, interviewing tips, support, and words of wisdom.

So how do you go about networking? How do you meet other Oracle DBAs and application developers? I bet you can guess *my* answer: NoCOUG, of course! At a NoCOUG conference you can spend the day with over 200 Oracle professionals who live in your region. Exchange business cards while you exchange technical tips and tricks—this is an easy way to build your personal network. And maybe someday you'll find yourself on the inside track because a colleague you met at NoCOUG threw a great lead your way.

And how do you get started? How about at NoCOUG's Fall Conference on November 4? This event will be held at the Computer History Museum in Mountain View, an exciting new venue for NoCOUG. The day will be filled with technical sessions led by Oracle luminaries such as Craig Shallahamer of OraPub and Peter Koletzke of Quovera. And as an extra bonus, free tours of this fascinating new museum will be available. It should be a great day.

As I Ride Off into the Sunset…

I have been at the helm of NoCOUG since the start of 2003, and term limits tell me it is time to step down and let another able volunteer take the lead. I definitely plan to continue as an active member of the all-volunteer board of directors. Together, we will continue to bring you the high-quality conferences, publications, and web resources that you have come to expect from NoCOUG.

Looking back on 2004, it has truly been a great year for NoCOUG members. For a modest membership fee of $70, members received:

- Four full days of education and networking spread throughout the year.
- Free admittance to sessions presented by the likes of Tom Kyte, Jonathan Lewis, Ken Jacobs, and Craig Shallahamer.
- Hefty discounts on registration to heavy-duty training days led by Gaja Vaidyanatha and Jonathan Lewis.
- Access to valuable articles, tech tips, and slide shows published in the *NoCOUG Journal* and presented on the NoCOUG website.

How else could you boost your career so significantly for just $70? I am proud to have led NoCOUG during the last two years, during a time when the organization has brought so much value to its members. But I certainly haven't done it alone.

I would like to take this opportunity to thank the volunteers on the NoCOUG board for all of their hard work. Thank you Darrin Swan, Joel Rosingana, Colette Lamm, Randy Samberg, Lisa Loper, Jen Hong, Les Kopari, Eric Hutchinson, Judy Lyman, Eric Buskirk, Vilin Roufchaie, and Hamid Minoui for all of your effort and dedication in 2004. Let's keep it going for next year!

And 2005 is shaping up to be another great year at NoCOUG. There will be more great training and educational events, four more quarterly issues of a great publication, and more web resources. So here is an excuse for you to contact those colleagues you haven't touched base with in awhile and jumpstart your networking all over again: *Tell them about NoCOUG!*

See you November 4! ▲

Grids Are Not Just for Data . . . and Other Insights

An Interview with Craig Shallahamer

I first met Craig Shallahamer in February 2004, at the Rocky Mountain Oracle Users Group (RMOUG) Conference in order to interview him for the *NoCOUG Journal*. We thought it would be a great way for members to get to know more about Craig, especially since he was scheduled to speak at an upcoming NoCOUG conference.

I had the opportunity to meet Craig again in September 2004 while attending one of his training seminars (Advanced *Reactive* Performance Management for Oracle Based Systems).

Craig is probably already a familiar name to many, since he's spoken at NoCOUG conferences before and is a popular speaker at Oracle Conferences worldwide. In addition, Craig founded OraPub, Inc., in 1998. OraPub's mission is "*Doing and helping others do Oracle Performance Management.*" Specifically, OraPub offers Oracle consulting, training, and sells a rightsizing product called HoriZone. However, the OraPub website is NOT just for paying customers. The site also holds a wealth of white papers that are available for anyone who registers (free). Some of my favorite white papers from the site are *Grid Computing—The Next Big Thing, Response Time Analysis for Oracle Based Systems, Conquering Oracle Latch Contention,* and *Capacity Planning Using Roller Coaster Tycoon.* You can also download the OraPub System Monitor (OSM) Tool, which is a collection of scripts to help you manage Oracle System Performance.

I thoroughly enjoyed meeting Craig and attending his training class. I hope you enjoy this interview equally as well.

From your resume, it shows you worked at Oracle Corporation from 1989–1998. You held the position of consultant, educator, manager, and director. What motivated you to leave Oracle?

Craig: Well, I started out working in consulting and enjoyed it for many years. But, I sort of got burned out going to customer sites. Within consulting, it was also important to develop educational material for the consultants. It was nice to combine the consulting work with the education work. Then, for about my last year at Oracle, I worked in the Education group and focused on developing and delivering training to other consultants globally. After that, I was ready to move on, but I didn't want to go back to the consulting group within Oracle, so I left and started OraPub, Inc.

How do you think your educational background contributes to your success tuning Oracle systems?

Craig: In college I started out majoring in math, but switched to economics, so my background is half technical/science and half economics/business. I really enjoyed the math, but once it started getting into proofs and becoming really abstract, I realized it wasn't for me. I find having an economic and math background to be useful since it there is always a business aspect to most technical problems.

Craig Shallahamer

Why did you decide to start your own company after leaving Oracle, instead of working for someone else?

Craig: Ever since I was a kid, I wanted to have my own company. My dad left the corporate world to start his own company, and throughout college, I started multiple little businesses. In fact, I tried to start a small business before I graduated so I could roll into that, but it didn't work. So I ended up working for a few smaller companies before I started at Oracle. Looking back, I realize working for a big company like Oracle had its advantages as well.

How did OraPub get started?

Craig: The consulting work I do at OraPub was a natural transition from what I did at Oracle. I really like to solve problems. The training courses started out slowly. Looking back, I realize I didn't have a great strategic plan. I just began to offer classes, and it started from there. I also had experience developing courses and training from my days at Oracle. But at OraPub, I can update my class material more easily and target it more toward what students are asking about in the classes. I also think any teacher will tell you when you notice that someone "*gets it*" it really feels good. You can take something really complicated and simplify it and kind of deliver it to them in pieces until they get the whole concept or message. The more you teach, the more you notice and watch people learn.

You are also a regular speaker at many conferences and user group meetings. What do you enjoy about those experiences?

Craig: I've always liked conferences; it really allows you to get out of your box. When you don't venture from your working environment, you start to lose perspective on what's going on around you. Besides networking with others and attending other presentations, I also enjoy speaking at conferences. It can be a real energy boost when you are talking to a couple hundred people.

Can you tell us a little about your experience with grid computing?

Craig: There are two main types of grids, data grids and computational grids. Computational grids are where you have CPU-focused processes like if you are going to do number crunching such as financial analysis or drug simulation and testing. These are really CPU-intensive, so you just want as much computational power as you can get. The other type is a data grid where you are doing something like market research and you need to go through tons and tons of data. Most grids are computational grids; they are actually much easier to create. Oracle grids are very datacentric, so they are built for a different purpose. Oracle grids also have all the things most other grid companies don't have, which is security, availability, and reliability for the more data-related issues. *Editor's note: For more information on Craig's thoughts on grid computing, download his white paper,* Grid Computing—The Next Big Thing, *from OraPub (URL at the end of this article).*

What motivated you to pursue the computational side of grid computing versus the data-type grids?

Craig: That goes into another part of my life, which includes starting a company named BigBlueRiver. The company is a grid computing company focused on helping people in third world countries. I wanted to find a way to work with people in developing countries where I could help them get a real job. I had many ideas, and one included using PCs hooked together through a network where people would get paid for that computer time. They wouldn't need extensive technical expertise or need to have business knowledge, but they could still do real work in a real job and get paid. Grid computing seemed like the right fit.

So, I began investigating and researching ideas. I have a central controlling program that takes work and distributes it throughout the grid. I have PCs in developing countries where they just need to make sure the computers are running and connected to the Internet, and, because these are third world countries, that nobody steals the PCs. The company pays them for the computational work the PCs do. So, it's purely a computational grid company built for CPU-intensive processes. The purpose wasn't a traditional business model, and the company has had success, which is rewarding.

Can you explain some of the aspects of proactively supporting an Oracle database?

Craig: As far as performance, forecasting provides a means to be proactive. It's where we want to live instead of living by reacting all the time or putting out fires. Proactively managing your database system is sort of like preventive healthcare. If you can, you don't want to wait until the problem becomes a crisis to solve it.

Capacity planning and forecasting will look at the system workload and by understanding the business forecast come up with how much capacity is needed. One way to accomplish this is to gather some workload, and develop a forecast model. You can then test out different scenarios such as what will happen if you increase the workload, CPU, IO, etc. This allows a more scientific approach rather than guessing or a gut feel.

I believe in forecasting so much that I created a product called HoriZone. There's information on my website if anyone wants to learn more, as well as a few white papers.

In the reactive tuning class, you offered a lot of information about some of the internals of the database. How do you research these topics?

Craig: I rely on what I call my "trusted advisors." There are many folks I've worked with through the years, and for specific areas of the database, I have a group of people I trust. If I have a SQL question, or an internals question, or a tuning question, I run down my list and try to get the answers to a question relating to a new feature or new parameter that isn't explained clearly elsewhere.

Besides the technical specifics of Oracle tuning, in the class you also define some important terms such as response time, queue time, and service time. Can you explain a bit more about these terms and how they relate to Oracle tuning?

Craig: Sure, understanding these terms, and having a definition that all agree on, aids in the tuning process. I take these definitions from the mathematical subject of queuing theory. Queuing theory can be complex or simple; for the purposes of understanding these terms and relating them to Oracle performance, it's simple.

The formula for this is: Response Time = Queue Time + Service Time or $R_t = Q_t + S_t$

You first need a queue, or a line (as we say in the U.S.). A queue is like a list of transactions waiting to be serviced. Queue time is how long a transaction waits before being serviced. It's important to add that queue time is unlimited, so a transaction can wait in a queue forever. Service time is how long it takes to service a transaction; an example would be the seconds consumed for CPU time servicing the transaction. In contrast to queue time, service time is fixed. So in this example, that refers to the finite amount of CPU time on the system. Response time is the sum of the queue time plus the service time. *Editor's note: For more information on this topic, download Craig's white paper* Response Time Analysis for Oracle Based Systems *from OraPub (URL at the end of this article).*

What interests do you have outside your business?

Craig: There is also a spiritual aspect to who I am, my relationship with Jesus. I tend to separate the business from my religious side, but it's always a challenge of how much of a spiritual aspect I should put into my work. For example, what do I do with my website, how far do I go in terms of that part of who I am. Oracle is great, but it's still just lines of C code when you get right down to it. Connecting with people, "heart stuff" instead of "head stuff," is really an important part of my life.

And, just for fun, how would you spend your ideal vacation?

Craig: I'd escape to an island in the South Pacific, and go fly fishing. My wife and daughters would also enjoy the island, but probably not the fishing (laughs).

To download white papers or the OraPub System Monitor (OSM) tool or to learn more about OraPub seminars and OraPub's rightsizing product, HoriZone, visit ***http://www.orapub.com/cgi/genesis.cgi.*** ▲

Interview conducted by Laurie Robbins, assistant editor, NoCOUG Journal.

Oracle 10g Upgrade Experiences

by Ignatius Fernandez

Citius! Altius! Fortius!
Faster! Higher! Stronger!

—The Motto of the Olympic Games

Introduction

We are a fast-growing provider of web-based business software to mid-size businesses with thousands of users across North America and overseas. Unlike traditional big-box software, our solution was developed with web-hosting in mind. The "shared everything" architecture consists of a powerful farm of web servers and database servers housed in a state-of-the-art co-location center and fed with redundant power and internet connections. Fine Grained Access Control is successfully used in conjunction with Demand Management techniques such as Query Governors to create a highly leveraged and scalable hardware and software solution that gives mid-sized businesses the feel of an expensive big-box solution at a fraction of the cost and in a fraction of the time. In this paper, I will share some of our experiences of upgrading our production platform from Oracle 9i to Oracle 10g.

Why Did We Upgrade?

The architecture team had been tasked with choosing and installing more powerful database servers to ensure that there would be enough database capacity for the foreseeable future. We decided to use Oracle 10g in the new server design. Since the application servers and web servers were not part of the project, we decided to leave the clients on Oracle 9i.

Research

The first thing to do was to learn all we could about Oracle 10g. I recommend that everybody start with a 20-article series by Arup Nanda, Oracle Magazine's "DBA of the Year" for 2003. It can be found at *http://www.oracle.com/technology/pub/articles/10gdba/index.html*. It's a quick way to get your feet wet and doesn't cost a penny. Check out amazing features like "flashback database," which allows you to perform fast database recovery without using backups! Also check out "grid control," which puts the "g" into Oracle 10g. It allows you to do things like compare two database configurations or two server configurations.

If you really miss the new book smell, consider investing in *Oracle Database 10g New Features* by Robert Freeman, brought to you by Oracle Press. If you want to burnish your resume with the Oracle Certified Professional logo, you can turn to Sybex, who have beaten everybody else to market with *OCP: Oracle 10g New Features For Administrators*, by Bob Bryla and Biju Thomas.

Ignatius Fernandez

If you want to go for the gold, download the Oracle documentation from *http://www.oracle. com/technology/documentation/ database10g.html* and start with the *New Features Guide*. To make things a trifle challenging, the *New Features Guide* is a pretty minimalist listing of the new features. You will have to use the Search tool provided with the documentation to search the other manuals for the information you need. However, this approach is highly recommended for the serious student, for several reasons. The Oracle documentation cannot be beat for correctness and sheer volume of detail. Finally, some of the new features in Oracle 10g are minor improvements on major new features introduced in Oracle 9i. For reasons of space, the books we mentioned assume a familiarity with Oracle 9i features. As an example, Oracle 10g offers improvements on the Virtual Private Database (VPD) feature introduced in Oracle 9i. Chances are that you have not encountered this feature before and you will need to review several long chapters in the *Security Guide* before you can claim to fully understand the Oracle 10g improvements.

A Few of My Favorite Things

Outstanding reasons why you should rush to upgrade to Oracle 10g include AWR (Automatic Workload Repository) and ADDM (Automatic Database Diagnostic Monitor) and the new and improved *web-based* Enterprise Manager interfaces dubbed "Database Control" and "Grid Control." AWR and ADDM represent the new and improved STATSPACK introduced in Oracle 8i. AWR is a data collection infrastructure and is hardwired into the Oracle engine. A special Oracle process called MMON handles data collection. A "snapshot" is generated at configurable intervals and ADDM is invoked to identify problems and suggest solutions. The ADDM report is vastly superior to the STATSPACK report because it does not simply spew a dry collection of metrics. Instead it identifies problems, estimates their impact on performance, and suggests solutions. You have to see it in action to believe it.

The developers of Oracle 10g have done a superb job in precisely articulating the goal of database tuning and creating tools to further that goal. The goal of database tuning is to minimize "total database time," which is defined as the

total time spent by database connections working *or* waiting, i.e., the total *active* time spent by all database connections. Oracle 10g tracks database time in the WRH$_SYS_TIME_MODEL table.

This clear definition of database tuning highlights the fact that database tuning requires a two-pronged approach. You need to reduce the amount of resources consumed by database sessions (perhaps by SQL tuning) as well as identify and remove bottlenecks that cause sessions to wait. Note however that reducing "working time" can also have an impact on "waiting time" and vice versa.

Most importantly, this clearly articulated goal of database tuning also provides a clear way to measure and quantify the effect of any tuning measures. In fact, ADDM attaches an estimate of the reduction on database time to every suggestion it makes.

The stunning highlight of AWR is the "Sessions Waiting and Working" graph displayed by Oracle Enterprise Manager. Every *active* Oracle session is either working or waiting. There are many different reasons why sessions find it necessary to wait and these can be classified into major classes (e.g., I/O, Concurrency, Application etc.). The number of sessions working and the number of sessions waiting in each major wait class can be plotted on a graph, creating a picture that is worth a thousand words (see Figure 1). After you have looked at this graph often enough, any atypical or abnormal pattern will stick out like a sore thumb, prompting you to immediately check the ADDM report for the time period in question.

Don't miss our Fall Conference!

November 4, 2004, at the Computer History Museum in Mountain View, CA!

Note that the *area* of the various bands is an approximation of each component of total database time.

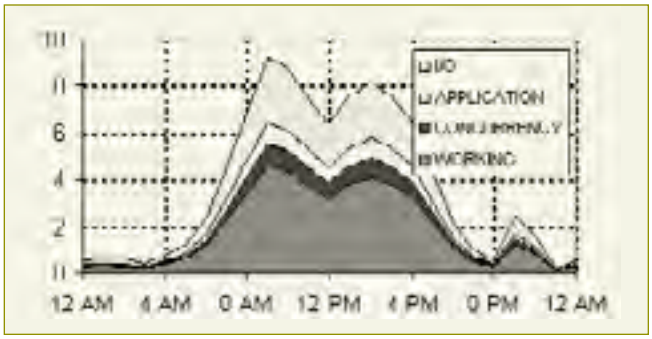


Figure 1: Graph of Sessions Waiting and Working

A longer overview of AWR, ADDM, and the new and improved graphical tools in Enterprise Manager can be found in Arup Nanda's lecture series.

Taking the "g" out of Oracle 10g!

Oracle touts the new manageability features of Oracle 10g, which help you manage the computing "grid." These manageability features put the "g" into Oracle 10g. Unfortunately, most of the new manageability features including "Database Control," "Grid Control," AWR, and ADDM are separately licensed, and only if you have previously licensed Enterprise Edition, which, at $40,000 per CPU, costs about three times more than Standard Edition and eight times more than Standard Edition One, the low-end offering for servers with one or two CPUs.

Interestingly, these manageability features are automatically installed even with Standard Edition and cannot be disabled. However, the Licensing Information manual clearly forbids their use. If the language is to be believed, you may not click on dozens of links or menu items in "Database Control" and "Grid Control" nor may you directly query the database tables in which information is being automatically collected by the MMON process. Chapter 2 of the Licensing Information Manual covers Options and Packs. All the new manageability features of Oracle 10g are grouped into four "Management" packs (Diagnostics Pack, Tuning Pack, Change Management Pack, and Configuration Management Pack), which have their roots in older versions of Oracle. Here is representative language from the Licensing Information manual.

"The entire Database Performance page, which you reach by clicking the Performance subtab from the Database home page, is part of this pack."

Further confirmation of Oracle's position is available in Metalink Document 271886.1: *Differences Between Different Editions of Oracle Database 10g*. Note that some stunning features of Oracle 10g, such as "flashback database" are indeed disabled in Standard Edition and Standard Edition One. It is possible that in a future patch release, Oracle will disable the manageability features now conveniently enabled.

Upgrade Methodologies

There are three ways you can upgrade the database. You can either use the interactive Upgrade Assistant, upgrade the database manually, or export the database out of the old database and import it into a freshly created Oracle 10g database. We decided to use the import method for the following reasons even though it would take substantially longer than the other methods:

- There are usually one-time performance gains to be obtained by rebuilding tables and indexes. Cache effectiveness is increased by packing rows tightly into data blocks, lowering the high-water mark, eliminating chained rows, and rebalancing indexes. Our testing also showed that a complete rebuild would shrink the database size by almost one-third. This would significantly reduce database backup times, which had begun to exceed the Service Level Objectives.
- Twice before, with Oracle 8i and with Oracle 9i, we had upgraded "in place" and continued to live with the old space management technology. The import method is the only way to convert to locally managed tablespaces, uniform allocation, and automatic segment space management (ASSM).
- We took the opportunity to perform all kinds of housecleaning. Some data files had grown larger than the size we were comfortable with. Permissions were out of whack in some cases. Some schemas could be dropped. For good measure we decided to increase the database block size to accommodate the rapidly growing complexity of the application.

It is true that the above reasons are really justification for a database rebuilding exercise, not really for our decision to not upgrade the database in place. Here are two additional reasons why we finally decided to use the longer route:

- It is often politically more expedient (and logistically more efficient) to add an additional task to an existing project than to undertake it as a second project. Business users are uncomfortable with frequent requests for downtime. One must take advantage of windows of opportunity such as long weekends. It is easier to piggyback on an already approved downtime window rather than to make a request for additional downtime. We decided to piggyback the database rebuild onto the Oracle upgrade just as we had already piggybacked the Oracle upgrade onto the hardware upgrade.
- The main attraction of Oracle 10g was the improved management and monitoring tools that we will highlight later in this article. We found that it was procedurally less complex to import the application data into a fully configured Oracle 10g database than to bolt on all the necessary management and monitoring pieces to an Oracle 9i database that had been upgraded in place.

The rebuilding route is not for everybody, especially when the source databases get very large, although it is possible to speed the process using network "pipes" and by creating multiple export and import streams. The upgrade manual contains a discussion of all three approaches, i.e., the Database Upgrade Assistant (DBUA), manual upgrades, and complete rebuilds.

Pre-Upgrade Experiences

Because we are an application developer, we have comprehensive regression test suites. Full-scale automated regression testing on full-sized data sets established that Oracle 10g operated at similar if not better performance levels than Oracle 9i. We did encounter a few subtle differences in SQL behavior that the developers were able to work around. We found no incompatibilities between Oracle 9i clients and Oracle 10g so we stuck with our decision to leave the web servers and Oracle 10g on Oracle 9i.

After creating pristine databases on the new hardware configuration, we shut down the databases and backed up the Oracle filesystems. We then repeated the import and export exercise several times to practice our approach. We handled the largest tables separately to reduce the total amount of time to perform the export and import.

Post-Upgrade Experiences

Thanks to all our testing and preparation, the upgrade went smoothly but we did encounter a couple of problems over the next few weeks. For some reason, we began to encounter ORA-04031 errors every afternoon when application usage peaked. To make a long and complicated story short, ORA-04031 errors result when a session cannot find enough space in the shared pool. The whole database freezes (and the DBAs go into panic mode) while the database attempts to find its footing. Oracle Support recommended we try the new "Automatic Shared Memory Management" (ASMM) feature, but that was no help. Mysteriously the problem returned to manageable levels when we eliminated CPU hyper-threading. Prior to the 10g upgrade, we had not encountered an ORA-04031 in several years, so we are convinced that the return of these errors was no coincidence.

We also encountered a problem with the nifty new compressed RMAN backups. Our backup strategy includes disk backups followed by tape backups, and Oracle 9i required us to compress the disk backups as a separate step. Oracle 10g compresses as it goes, which reduced the total time for the backup by almost 50%. However, paging and swapping went through the roof and application response plummeted during backups. Our system administrators solved the problem by tweaking the system parameters that relate to page caching.

On the positive front, our measurements showed a 25% improvement in application performance, though we cannot separate the contributions of the hardware upgrade, OS upgrade, Oracle upgrade, and database rebuild! We were very pleased with the 50% reduction in backup times. We were also able to eliminate the job that collected optimizer statistics, because Oracle 10g does it automatically and efficiently.

Recommendations

If you are still on Oracle 8i and have only recently begun considering moving to Oracle 9i, you could evaluate the option of skipping Oracle 9i completely and going directly to Oracle 10g. ▲

Iggy Fernandez is the Lead DBA for a Silicon Valley startup and is Oracle 10g certified. Previously, he was the manager of database administration for Corio, an Application Services Provider (ASP) and was responsible for a mixed portfolio of nearly one thousand Oracle and SQL Server databases. He is interested in best practices for Oracle database administration and is writing a book called A Structured Approach to Oracle Database Administration using Oracle 10g, *which seeks to apply I.T. Service Management (ITSM) techniques to Oracle database administration. You can contact him at* iggy_fernandez@hotmail.com.

Oracle Performance Troubleshooting

Book reviewed by Brian Hitchcock

Oracle Performance Troubleshooting with Dictionary Internals SQL & Tuning Scripts
Oracle In-Focus Series
by Robin Schumacher
published by Rampant TechPress 2003

Summary

Overall review: OK.

Target audience: Anyone who needs a quick overview of the causes of performance issues within the Oracle database.

Would you recommend to others? Yes, if you are new to performance tuning. Experienced DBAs will find it lacking.

Who will get the most from this book? Readers who have a general knowledge of the components of the Oracle database but haven't done performance tuning before.

Is this book platform specific? No, the book covers the specifics of the Oracle database and doesn't cover any operating systems issues. Note that this book does cover 9i features.

Why was this book read/chosen for review? I needed to understand the details of shared pool tuning, and I was looking for SQL scripts that would help me examine the relevant system tables.

Overall Review

The Technique You Will Learn

The back cover of the book promises to focus on how to quickly troubleshoot and correct Oracle performance problems. The discussion and scripts in the book do support this claim. However, the subject of Oracle tuning is much broader than what this book covers. Many readers may find that their database, and the environment in which real-world applications live, will have performance issues that aren't addressed by this book. The performance troubleshooting approach described in this book relies on using SQL scripts exclusively. This will work for many—but not all—database performance issues. The more complex the environment, the less likely it is that this approach will provide a comprehensive solution. There are many performance issues that will affect overall application performance that aren't visible from within the database. Using only SQL scripts means your view of what's going on is going to be limited.

To make best use of the approach presented, you have to have already determined—from some other process—that the database performance is the primary performance issue for your application. For example, if the user's client machine is the primary reason for slow application response time, the database performance issues that can be detected and resolved by this approach may be irrelevant.

Chapter by Chapter

Chapter 1 – Accurately Measuring Performance – is a generic discussion of how to assess performance and what the various factors are that affect the end-user perception of overall application system performance.

Chapter 2 – Performance Methodologies – reviews the use of ratios, wait events, and looking for bottlenecks in the system. The comments on the limitations of each of these methodologies are valuable and should be given more weight in the mainstream DBA community. For example, it can be difficult to gauge the extent of sorts to disk from wait state analysis alone, while a ratio of in-memory sorts to disk sorts can be very useful. The author goes on to clearly explain the benefits and limitations of both ratio and wait based analysis.

Chapter 3 – Foundational Flaws – discusses the impact of physical database design errors. This is refreshing, as it doesn't get discussed much in the general DBA press. For those who haven't done much tuning work before, this foundation is needed. It is easy to spend lots of time tuning SQL when most of the problem is in the physical layout of the database. The author points out that application tuning can't be successful unless the physical layout of the database is done properly first.

Chapter 4 – Optimizing Storage – covers disk layout and fragmentation issues. The author expresses concern that bigger disks, which means fewer physical drives for a given database size, will cause performance problems. I haven't seen this in practice, so I think this would need to be supported with more evidence. The discussion of fragmentation is good, and provides a detailed example demonstrating that fragmentation does not cause performance problems.

Chapter 5 – Maximizing Memory – reviews the value (yes, the value) of buffer cache hit ratios, issues relating to the shared pool and sorting. A number of features new to Oracle 9i are covered. The author provides a good discussion of the use and misuse of the buffer cache hit ratio. This leads into a discussion of the new 9i feature db_cache_advice that provides some insight into how a larger buffer cache might help performance.

Chapter 6 – I/O Hotspots – covers how to find database objects, data files, and users that are causing the most I/O, chained rows and rollback issues. Scripts are presented to determine the I/O activity of the overall database, ratios for physical to logical I/O, wait events relating to I/O, how to see how objects are being accessed (table scan, row fetch), as well as the user sessions that are currently using the most I/O.

Chapter 7 – Workload Analysis Part 1, Problem Sessions – oddly, brings up security at the beginning of the chapter. The general issue raised is valid, but security should have been included in the high-level discussion of performance issues in Chapter 1. This section covers how to find users, processes, and individual SQL statements that are consuming the most resources. A personal favorite of mine is the script that will identify any SQL that is currently in the cache that is using a Cartesian join. I haven't seen this before, and it is a good way to find problematic SQL.

Chapter 8 – Workload Analysis Part 2, Problem SQL – a good high level review of how to find and fix problem SQL statements.

What Was Good

By using the scripts in the book, you can look into many Oracle database performance problems without installing any other tools. These scripts only require a SQL*Plus connection and the privileges needed to select from the system tables. For someone who can't get any more access, or someone who doesn't want to install any further tools such as STATSPACK or any of the many third party products available for Oracle tuning, this approach is appealing. It is the simplest, lowest cost and most universal way to approach Oracle database tuning.

The book was also inexpensive (relative to other Oracle books), and doesn't try to cover everything. As long as the reader understands what the book does and doesn't cover, the value is very good.

A fast read—some might say it's too short, but that can be viewed as a plus. This book doesn't waste your time with huge reprints of the Oracle manuals, like many tuning books do.

Small, targeted books like this one are to be encouraged, with the understanding that it may take more effort, and more editorial oversight, to make this work well. To be successful, a quick read needs to be very carefully targeted to deliver the best value to the reader.

What Could Be Better

The process described in the book is presented as Oracle performance troubleshooting, and covers performance issues within the database. If you are sure the performance issue is in the Oracle database, that's fine. However, there are many performance issues that involve the operating system, network, client machine, etc. I think this should be pointed out more clearly in the early sections of the book.

The book doesn't mention the tools available to help find and resolve Oracle database performance issues. Specifically, I refer to STATSPACK and 10046 tracing. I have made clear my support of short, targeted books, and I don't want this book to expand to try to cover all possible database performance tools. However, short sections describing these tools would be good since these tools are included with the Oracle database software. Since I think this book is best for people who haven't done a lot of Oracle tuning, describing the existence and value of the tools that come with Oracle would be of great benefit.

A Review Relative to Tuning Books/Methods in General?

There are many sources for performance tuning methodologies. Because of the importance and popularity of this material, I like to look at any given approach with the following questions (along with my answers specific to this book):

How to approach tuning?
— Assumes the database is the performance issue, uses SQL scripts exclusively, doesn't use other tools available within Oracle (STATSPACK, 10046 trace)

What is the value of tuning?
— Not addressed

How to know when you are done?
— Not addressed

Can your users understand what you are doing?
— Yes, absolutely, process is clear

Is the process well documented?
— Yes, very much so

Is the process repeatable?
— Yes, definitely

Editorial Details

Normally, the physical layout and presentation of an Oracle book doesn't get my attention. Unfortunately, this one did. While the following comments don't detract from

the value of the technical material in the book, the book itself appears to have been hastily prepared. It would be good if simple things had been done better. Each chapter should have a title, if only to show that the author and editor have a theme for the chapter and to show what the intended flow of the book was. (I know what it's like to have very good editors; they make you write better because you know someone, a very demanding someone, is always watching!). Further, it would be good if each page of the book showed which chapter it was part of. This book simply repeats the book title at the bottom of each page.

On page 13, halfway down the page, there is a mysterious box that says, "The code depot key is rudy". What is this about? I have no clue what this is telling me. Tight editing could have prevented this.

As mentioned previously, one of the specific reasons I spent my own money on this book was for the scripts, but I don't want to even try to manually type them into my system from the book. The book comes with a CD, but the CD doesn't have the scripts. On page 1 of the book we are told about the Online Code Depot, the URL is given and we are told that this is where to go for the sample tests and answers. I don't see any sample tests in this book, and there is no mention of whether the scripts are available from any source. I assumed that the reference to "sample tests" really meant the scripts in the book, so I tried the URL. But, you have to have a userid and password for the URL. I had to email the publisher asking for the userid and password (and the password isn't "rudy"). They replied quickly, but this process could have been much simpler.

Conclusion

Overall, this book could be ideal for developers that are supporting their own development databases. In this environment, the developers will have full access to the database and the database machine and may not have a full time DBA to monitor database performance. With the efficient presentation of the performance issues, and the scripts, it would be easy to monitor and resolve performance issues as the developers modify their code. The book is quick read which makes it a good value for anyone that wants a quick, high-level view of the primary performance issues within the Oracle database. ▲

About the Book Author

Robin Schumacher is vice-president of Product Management for Embarcadero Technologies, Inc., a leading supplier of database software tools. Robin has over fourteen years experience in database administration, development, monitoring, and tuning with Oracle, DB2, Teradata, Sybase, and Microsoft SQL Server. He has authored countless performance-related articles for many database-centric magazines as well as serving as a database software reviewer and feature writer for the likes of Intelligent Enterprise, eWeek, DM Review, and others.

(See page 27 for information about the book's reviewer, Brian Hitchcock.)

A Little NoCOUG History

In honor of our upcoming visit to the Computer History Museum for our Fall Conference on November 4, we thought we'd publish a few facts about NoCOUG's history. It's interesting to see how we've grown.

- NoCOUG started in the late 1980s.
- Our Board member with the most knowledge of NoCOUG history is Joel Rosingana. He joined NoCOUG in about 1991 and has been on the Board of Directors since about 1993. He has held many positions, including president (more than once). He has a little history in his office, too. As many of his friends know, he has Oracle v6 on DOS. But don't worry, that's for nostalgic purposes. He's also running Oracle 9i.
- Back in the early days, 60 attendees at a conference was a good turnout. At the Summer NoCOUG Conference in August 2004, we had over 250 attendees.
- In 1991, individual membership dues were $50. For 2005, they are $70.
- At the Spring Conference in 1991, there was one track and four technical sessions, and the meeting ran from 9:00 a.m. to 2:30 p.m. At the Spring Conference in 2004, there were three tracks and 12 technical sessions, and the meeting ran from 8:00 a.m. to 4:45 p.m.
- NoCOUG's bank balance as of September 30, 1991, was $6,592.80. On September 30, 2004, NoCOUG's balance was $60,908.56. ▲

Stayin' Alive: High Availability Without Breaking the Bank

by Jeremiah Wilton, Independent Oracle Professional

Part Two: Unplanned Outages*

* **Editor's Note:** ***This is the second of a two-part series. Part One, published in the August 2004 NoCOUG Journal, provided an overview of high availability and dealing with planned outages.***

Unplanned Outages

Unplanned outages are the sexiest area of availability, understandable not only to all DBAs, but also to managers and executives. Because such failures are inherently unpredictable, DBAs often feel the need to be prepared not only for the most probable and easily preventable failures, such as disk faults and instance crashes, but also for large-scale natural calamities. The challenge in planning such strategies is walking the thin line between prudence and overzealousness.

When discussed in conference rooms, the topic of availability is fraught with scenarios of calamity and apocalypse of biblical proportions. While I by no means seek to diminish the likelihood of Armageddon, DBAs and corporate executives alike must seriously consider that a disk failure or even a bad motherboard is statistically a great deal more likely, by several orders of magnitude. Complex solutions that hold the promise of near-100% infallibility must not be pursued at the expense of commonsense measures that provide basic protection against the most common failures. While great ships may be unsinkable, common sense reminds us not to underestimate the necessity of lifeboats. Or lifejackets. Or a paddle.

Monitoring

The most basic measure you can take to maintain high availability is to monitor your systems. To this end, many companies have produced software for monitoring logs, disk space, CPU, memory utilization, and database statistics. Some sites have built their own monitoring mechanisms in Perl or using shell scripts. When considering whether to buy or build, be sure to take into account the extensive time and effort required to customize any off-the-shelf package to meet the business requirements of your system.

At a minimum, monitoring scripts must have the ability to notify staff of critical events, so that appropriate measures can be taken to avert an outage or rectify a critical problem.

Alert file monitoring

Many critical events such as block corruption, memory, and space problems can be detected by monitoring the alert file. A good monitor will allow a user-definable list of normal messages (beginning checkpoint, checkpoint complete, alter tablespace) to be excluded from the notification system, leaving only unusual messages to be reported to the DBA. Significant downtime can be shaved off an outage if the DBA receives a page such as "Instance terminated by PMON," rather than waiting for users to complain and then logging in to investigate.

Jeremiah Wilton

Additionally, it is helpful if low-priority messages can also be differentiated from critical ones, so that a single ORA-7445 or ORA-600 in a shadow process does not wake the support staff at 3:00 a.m. Such messages should instead be reported to an email address where the issue can be investigated at lower priority at a later time.

Host monitoring

It is typically the responsibility of a systems administrator to monitor critical aspects of system performance, including CPU, disk, and memory utilization, as well as the system messages file. Such monitoring, however, is absolutely critical to the successful continued operation of an Oracle database. So if such monitoring does not exist, it is manifestly in the interest of the DBA to make sure that it is established.

The LOG_ARCHIVE_DEST directory, for instance, must never fill to 100% capacity, or the archiver will suspend operations on the Oracle instance once all online redologs have been filled but remain unarchived. Similarly, if listener logging is enabled, listeners will cease to accept connections if the logging directory is full. For this reason, a DBA must either work very closely with the systems administrators or monitor critical systems resources independently.

Availability monitoring

Probably the single most important monitor is one that will determine if a system is unavailable. Many DBAs write this monitor to look for background processes such as PMON and SMON, and to alarm if they are missing. Unfortunately this method does not determine availability. True, you can be reasonably sure that without PMON the service is or will very soon be unavailable. But the presence of PMON does nothing to guarantee that people can access a service.

A better method for guaranteeing availability is to write a program that runs on an application server where a typical user application runs. The program should use the same database API that a typical application uses, and should attempt to connect to a service and fetch a row of data. If an availability monitor can't do that, it isn't really measuring availability.

Availability monitors are important record keepers. If you build one, you should have it record availability in a file or a database. This allows an accurate record to be kept of a system's availability over time, and provides a metric for the DBA and management to determine if current availability strategies are adequate or not.

Wait monitoring

Along the same lines of "the proof is in the pudding," comes the practice of monitoring for excessive waits in an instance. If too many sessions are waiting, then the service isn't truly available to those sessions stuck waiting. It is technically an outage (however brief) for those users. An effective wait monitor will alarm when the percentage of sessions waiting on nontrivial wait events exceeds a threshold percentage. Trivial events that can be ignored include "rdbms ipc message" and "SQL*Net message from client." This type of monitoring is the next step up from availability monitoring as discussed above.

Frequent monitoring of system statistics and wait events allow DBAs to identify the cause of slowness problems and also allow accurate projections to be made of resource utilization based on historical trends. The critical views to monitor and record are as follows:

- v$system_event — *for all-time greatest wait events on the instance*
- v$sysstat — *for all-time greatest resource utilization on the instance*
- v$session_event — *for each logged-in session's greatest wait events since login*
- v$session_wait — *for each logged-in session's current wait event*
- v$sesstat — *for each logged-in session's greatest resource utilization since login*

STATSPACK does a good job of recording these and other tables over time, and presenting trends and data from the historical information. In Oracle 10g, Active Session History (ASH) provides a superior fine-grained mechanism for recording wait and resource information on a per-session level over time.

By recording the v$session_wait session event information, outages and general slowness problems can be easily diagnosed and prevented, even if a DBA did not actually log in and witness the problem at the time it happened. By going back through logs of v$session_wait output, the exact event that a particular session was waiting on at a given time can be determined. The v$session_wait view provides further information that a good monitoring script would be able to decode.

- If v$session_wait indicates that a session is waiting on an enqueue, then the monitoring script should also look and record the blockers and waiters.
- If a session is seen waiting on "latch free," then the monitor should record the name of the latch from v$latch and the session holding the latch at the time identified from v$latchholder.
- If a session is performing a "db file sequential read," then the monitor should record the table or index being scanned from the p1 and p2 columns in v$session_wait and joining to dba_extents.

On a database-wide basis, statistics and events should be monitored and recorded on a periodic basis using Statspack or your own scripts. Using this information, generally increased resource utilization can be narrowed down to the time it started and correlated with the specific changes or applications that might have triggered a problem. Also, gradual increases in resource utilization can be tracked to accurately determine when the current system will run out of capacity.

Oracle tablespace monitoring

It is unacceptable for users of a high-availability site to encounter "ORA-01653: unable to extend table . . ." or an "ORA-01654: unable to extend index . . ." Effective monitoring should be able to determine if the NEXT extent of any segment in a tablespace is bigger than the largest available piece of free space in that tablespace, and raise an alarm if it is. Similarly, any table close to reaching MAX_EXTENTS should be noticed and the appropriate people notified. This kind of monitoring is fundamental to the continued smooth operation of a mission-critical system.

The most advanced monitoring of this type will maintain a daily log of the growth of individual segments, and project growth into the future. In this way, effective monitoring will notify the DBA well in advance of space being exhausted. It is much easier to understand the impact of five days worth of space remaining in a tablespace than 512Mb of space.

Lock monitoring

Often, complaints of slowness in the database are traced back to users contending for the same rows in tables, when one user is holding a row lock for too long and blocking others. Preventing locking problems in the first place relies upon good application design. A well-designed OLTP application should not hold row locks for more than a few seconds. But even the best applications, using optimistic locking approaches, can exit abnormally while holding a lock, and PMON may take some time to clean it up.

Monitoring can be implemented to detect blocking locks and to automatically take a series of steps to resolve the problem. One good approach is time-based escalation. With time-based lock notification escalation, a blocking lock is initially detected by a monitoring script, and notification is sent to the blocking user, either by email or terminal warning. Unfortunately, some end users do not use terminals that can receive server-initiated messages (such as a web browser), may not be reading their email, or may in fact not be people but rather batch processes. If a lock is not released by the user, then the notification level escalates, and a page is sent to a DBA. In extremely critical applications, or in applications that require lights-out operation, the blocking session can simply be terminated if it blocks for too long, allowing waiting sessions to proceed.

Resource limits

Certain limits do not make sense in a high-availability environment. Limitations on extents, sessions, processes, and similar parameters should be set at theoretical maximum system capacity levels. In many cases, setting artificial limits can result in a degradation of service that has no basis in true system capacity.

A good way to be sure that availability problems will not be caused by artificially low settings on initialization parameters is to comb thoroughly through v$resource_limit, seeking parameters that are near their limits.

Monitoring should be deployed to check v$resource_limit periodically, and to notify if any resources approach their limit.

MAXEXTENTS UNLIMITED

If good monitoring practices, such as those outlined above, are maintained, it does not make sense to set MAXEXTENTS on tables in application schemas to anything other than UNLIMITED. Even if a DBA wants to keep extents below a certain number, it makes more sense to monitor extent allocations and take corrective actions when it is least disruptive. To cause degradation or loss of service as a result of reaching MAXEXTENTS during peak usage periods would be a senseless outage. It is useful to note that tables and indexes can be composed of many thousands of extents without any significant performance or manageability problems. The classic problem of extent deallocation on drop or truncating of segments has been almost entirely alleviated through the use of locally managed tablespaces.

Rollback and temporary segments, on the contrary, should ***not*** be set to grow with MAXEXTENTS UNLIMITED. Unbounded growth of a single rollback or temporary segment can cause denial of service for other sessions, and benefits only the session causing the growth. On high-throughput OLTP systems, no single session should ever need massive amounts of temporary or rollback space. Transactions on such systems should be small and commit quickly, and sorting should be tuned to a minimum for best application performance.

Initialization parameters

Several initialization parameters can impose needless limits on availability. Because the default values of many parameters are derived from the specified values of other parameters, some limits can exist without being expressly specified in the initialization file. A good example is the SESSIONS parameter, which has a default value of (1.1 x PROCESSES + 5). This derived value makes sense as long as sessions grow at roughly the same rate as processes. However, at a site using Shared Server (formerly MTS), sessions can grow at a much greater rate than processes, because Shared Server consolidates processes. When using Shared Server as a scalability measure, care must be taken to account for the large number of sessions, transactions, DML locks and other resources that can be supported on a relatively small number of processes.

Practical limits

Some resource limits are not covered by v$resource_limit, which only includes initialization parameters whose values might be exceeded. There are numerous other practical limits that can be exceeded, and should be individually monitored. Some of these include:

- Queue depths for AQ
- Multimaster replication queue depths/lag
- Streams queue depths
- Materialized view log depth for fast refresh MVs
- Limits for Shared Server such as dispatchers, shared servers and circuits.

A good example of this is the shared pool.

Shared/large pool monitoring: ORA-04031

By monitoring v$sgastat, a DBA can make a simple determination ahead of time if an instance is going to run out of shared or large pool space. In systems with diverse and heavy workloads, the shared pool can easily become fragmented. In this situation, there appears to be sufficient free space according to v$sgastat, but when you look at x$ksmsp, the chunks of free space that are available are too small to use. A good shared pool monitor will not only monitor the percent free space, but will also compare the sizes of common objects in the shared pool with the number and size of free space chunks.

Corrective actions

Many monitoring scripts can be designed to take corrective actions on their own, rather than immediately paging a DBA.

- Scripts monitoring the job queue can try restarting a job.
- Jobs that detect blocking locks held by long-idle sessions can terminate the session.
- Upon finding that the shared pool is badly fragmented, selected objects can be unpinned, and the shared pool flushed.

Oracle's user profile feature allows SMON to perform many resource-limiting actions along certain parameters, but more fine-grained control can be exercised by writing simple scripts to do such tasks.

Redundancy

Redundancy is one of management's favorite buzz words when high availability is discussed. It is a word that brings to mind very expensive solutions, such as complete

remote replicas of all critical systems, triple disk mirrors, and redundant diesel generators. Such solutions may help guarantee availability for mission-critical systems in the direst of circumstances, but may be beyond the means of the average organization, or even many established companies.

For the greatest benefit for the least capital expenditure, redundancy may be applied not only to whole installations, such as hosts and datacenters, but also to small pieces of architecture, such as redologs and listeners. Many individual components in the Oracle architecture can be made redundant within the scope of a single host. At little or no additional expense, greater availability and peace of mind can be achieved with little effort.

Redundant storage

One of the most basic measures you can take to improve availability is to invest in fault-tolerant storage. This means disks should be mirrored, or protected with a fault-tolerant RAID level. Disk controllers can be made redundant as well, and disk arrays can be split across multiple controllers. Files that Oracle mirrors such as controlfiles and redologs can be mirrored on different controllers. Although volume management software products offer the capability to perform software striping and mirroring of storage, it is best to use disk controllers for this purpose. Disk controllers operate as independent CPUs, and can perform mirror resyncs and other I/O and CPU intensive operations without consuming host resources, as volume manager mirroring does.

RAID levels

Different RAID levels provide different levels of fault tolerance and speed. Choosing a RAID level is dependent on the type of access patterns and the availability requirements of the system.

SAME

Oracle recommends that customers "stripe and mirror everything" (SAME). This recommendation that everyone use RAID 0+1 is fine for people who don't want to think at all about their specific needs, but others may choose to put more thought into their storage configuration. For example, some instances may have a low rate of writes. Their best RAID level might be RAID5, since they can save money on equipment and space.

Precious logfiles

Other customers might have set an extremely high write rate on their online redologs, or don't want online logs to become a bottleneck during periods of heavy transactional activity. Because redologs are sequentially written, those customers might prefer to assign dedicated physical disks to redologs, and to alternate such that the archiver reads from a disk other than the ones containing the current active logfiles. If log multiplexing is to be used, then the disk mirroring would be superfluous. Sustained heavy write activity on redologs can also render the RAID's write cache superfluous for those writes.

There is some disagreement over whether it is necessary or advisable to mirror logs at both the hardware and Oracle levels. Complicating this question is the aforementioned common practice of placing redologs on dedicated disks. A level of fault tolerance higher than just storing logs on a RAID 0+1 array can be achieved by using Oracle log multiplexing to maintain logfile members on two or more separate disk subsystems with independent controllers. Consider that although redundant disk controllers can be used to maintain a single mirror set, any controller-based corruptions that occur in such a configuration will be applied to all copies of a log. If a separate controller is responsible for each copy of a logfile, then one controller's corruptions can only damage one copy of any particular log, and never both. Also, by using multiplexed redologs, a system can be protected against many filesystem or volume manager-based faults. This desire not to rely on a single disk array for all redologs drives many organizations in the direction of multiplexing over hardware mirroring for logs.

Using dedicated redo drives also assures that heavily accessed database blocks will not be accidentally placed on the same disks as redologs, causing I/O contention.

RAID Level	RAID 0	RAID 1	RAID 0+1	RAID 3	RAID 5
Description	Data is striped across a set of disks	Data is mirrored among a set of disks	Data is striped and mirrored	Data is striped across a set of disks, with one disk reserved for parity information for data recovery	Data and parity information for data recovery are striped across a set of disks
Advantages	Access is distributed across many disks, reducing contention	Data is duplicated in case of a disk failure	Data is duplicated in case of a disk failure and evenly distributed for improved access speed	Fewer disks are required for data distribution and fault tolerance	Fewest disks are required for data distribution and fault tolerance
Disadvantages	No fault tolerance	Data is not evenly distributed; writing to two drives takes extra time	Data must be written to two drives	Parity drive is a bottleneck; many I/O operations are necessary for each write	Many I/O operations are necessary for each write
Speed	Fast read Fast write	Fast read Slower write	Fast read Normal write	Fast read Slow write	Fast read Slow write

Redundant listeners

Redundancy at the Net8 listener level is another critical element for high availability. Although the days of Oracle listeners hanging are largely gone, you can still use multiple listeners to achieve redundancy across network interfaces, or to allow shutdown of one listener at a time for maintenance or troubleshooting.

To achieve redundancy with two or more listeners, configure client tnsnames.ora entries with the failover=on parameter and multiple addresses. When a client is establishing a connection, it will try the first address, then the next, and so on. If one listener fails, connection attempts will automatically try to use the other listener. In this way, one listener at a time can even be intentionally stopped and restarted without any effect on client connections.

When running in a multiple listener configuration, connections can be randomized among the listeners by specifying the load_balance=on parameter in the client tnsnames.ora. This allows you to realize the scaling benefits of using both listeners.

Address List example with failover and load balancing:

```
orcl.world=
(description=
(failover=on)
(load_balance=on)
(address=(protocol=tcp)(host=207.60.86.67)(port=1521))
(address=(protocol=tcp)(host=207.60.86.67)(port=1522))
)
(connect_data=(sid=orcl))
```

Cost vs. rule

Most DBAs think of the choice of optimizers as a tuning issue, not a high-availability issue. But ask anyone who has run in a large environment with the CBO, and they will tell you that they have had more than their share of downtime related to the CBO. Even if the CBO can't crash the database, it can render unusable a critical query performed by a critical application. In many cases, this means that the service is technically down until the query is fixed.

A popular approach has been to ignore the CBO and use rule exclusively. Oracle Applications were dependent solely on the RBO through 11i. Soon, however, there will be no choice between cost and rule when it comes to choosing an optimizer mode. Oracle's stated direction for future versions is to gradually phase out use of the rule-based optimizer. This is good news, because it means they will necessarily have to improve the stability of the CBO.

Because of the infamous instability of the CBO, it goes against the instinct of many DBAs that overall greater stability could be achieved through its consistent and standardized use. After all, using rule guarantees that query plans will not change wildly from one day to the next for the same SQL statement, which is a hallmark of the CBO.

The advantage of the CBO is that it is dynamic and adaptable. With the CBO, if tables are consistently analyzed on a regular basis, and histograms maintained on skewed columns, then rapid changes in data size or distribution of a particular table can be handled with the most efficient query plan for that data. With the RBO, changes in data distribution in a table will result in no change in query plan, so no additional efficiencies will be introduced to handle such changes.

Oracle's improvement of the CBO since Oracle v.7 has been a slow process, but with 10g it is reaching some kind of maturity. Increasingly, the task of analyzing tables has been integrated into everyday tasks that read the data, such as index rebuilds. In 10g, statistics are maintained automatically—after initial collection with DBMS_STATS—via the normal reading of data by server processes. In 9i, this was also possible with the MONITORING segment attribute.

The use of segment monitoring should mitigate some of the instability associated with the CBO. Previously, it was necessary to collect statistics periodically. Suddenly analyzing tables that had never before been analyzed, or that had run without statistics for a long time often produced unexpected results.

A variety of features have been developed over the life of the CBO to improve stability. In Oracle 8i, query plan stability was introduced, allowing desirable query plans to be frozen and preserved. Optimizer statistics for a segment can also be moved around starting in 8i using DBMS_STATS.EXPORT . . . This allows DBAs to save prior statistics when analyzing tables. It also allows the application of production statistics to a smaller development database for more realistic testing of applications and query plans prior to rolling out new or changed applications into production.

Nevertheless, be prepared to closely monitor query plans on any instance that undergoes an Oracle version upgrade. So many changes go into each new revision of the CBO that query plans can change dramatically from version to version.

Sometimes, despite your best efforts, a statement that performs well under the RBO will perform poorly with the CBO. If tables are properly analyzed, and no cause for inefficient CBO behavior can be determined, then users should contact Oracle Support to report and thoroughly document the issue. Oracle cannot improve a product if bugs go unreported.

Filesystem maintenance and cleanup

If you have ever experienced a few core dumps (ORA-7445) over a short period, you know that Oracle's core files can be large, because they dump the whole SGA. An Oracle process that dumps core should not take down a whole system, but in many environments it easily fills all available free space in a filesystem. This can cause a variety of availability problems including the inability of the listener to write to its logfile, which actually causes the listener to hang in some versions. A full filesystem could also prevent the archiver from working, which would eventually cause an instance to stop accepting transactions.

One good solution is to place the user_dump_dest and background_dump_dest on their own filesystems, separate from the database files, listener log, and archived redologs. Alternately, some Unixes allow quotas to be imposed on certain directories so that they cannot exceed a specified size. This approach is preferable to setting max_dump_file_size, which can truncate trace files that are needed for debugging a problem.

To prevent filesystems from filling up in the first place, jobs should be scheduled to run periodically to clean up

dumps and archivelog directories. Such scripts should be smart enough not to delete needed files. For trace files, a good script will delete from oldest to newest, bringing the filesystem utilization down to a threshold percentage, such as 50%.

For archivelogs, a cleanup script should be smart enough never to delete logs that have not been backed up. For RMAN backups, determining which logs have been backed up is a matter of an easy lookup in v$backup_redolog. For your own non-RMAN archivelog backups, some kind of separate state file or table has to be maintained by the backup jobs in order to track log sequences that have been backed up.

In addition to cleaning up trace, core, and archive files, filesystem maintenance scripts must also be able to trim listener logs and the alert file. The alert file is unproblematic to manage, because Oracle's logging functions used to write to it do not hold the file open. If the file is renamed, Oracle will just start a new alert file named alert_<sid>.log.

The listener logs are not so dynamic. The listener holds its log file open, and if the file is moved, it will continue writing to the renamed file. This makes operating the Net8 listener in a high-availability environment a challenge. One way to truncate the listener log in Unix is to cat the null device into the listener log, as in the following example:

```
$ cat /dev/null > listener.log
```

This clears the log file, and the listener continues writing to it. You may prefer to retain the data from listener logs for capacity planning and problem resolution. If so, the listener log should be copied periodically to a dated file, and the log truncated. This operation should be conducted during a low usage period, because you will lose any entries made in the log between the copy and the truncation.

Redo-related pauses

When an Oracle instance runs out of available online redologs, it will pause, and no new operations or logins will be permitted until the situation has cleared. That constitutes an availability problem, even if it is brief. Even very brief downtimes detract from the overall experience of users.

When Oracle fills an online redolog, it cannot use that logfile again until it has been archived (assuming the database is in ARCHIVELOG mode), and the database has been checkpointed. If the instance proceeds through all redologs before archival or a checkpoint has occurred, Oracle will hang because it does not have any logs available in which to record transactions.

If the archiver cannot keep up with the rate of redo generation, multiple archiver processes may be used to archive faster. However, steps should be taken to determine the cause of and reduce the rate of redo generation if possible. The best way to determine the source of heavy redo generation is to look in v$sesstat, where the redo size of every session is recorded by the timed statistics feature. The following query displays redo generation (per unit time logged in) by all currently logged-in sessions, ordered by volume:

```
select module, osuser, sql_hash_value, value / (sysdate -
logon_time) redo
from v$session s, v$sesstat ss, v$statname sn
where s.sid = ss.sid
and ss.statistic# = sn.statistic#
and name = 'redo size'
order by redo;
```

Often the archiver only falls behind during periods of heavy transactional activity associated with large data loads or index rebuilds. The UNRECOVERABLE option is a good way to mitigate excessive redo generation for repeatable data loads and for index builds. You need to understand your own recovery scenarios and path before using UNRECOVERABLE, but for repeatable activities, it can be an important tool for achieving scalability.

If Oracle cannot complete a checkpoint ("checkpoint not complete" in the alert file) before using all available online redologs, then there may be any of a number of issues at work. While the boilerplate solution for this problem is to add more redologs, this may not be the correct solution. The most basic reason for checkpoints not completing is that the I/O subsystem is too slow to keep up with the demand in writes from the database. Many Unix filesystems do not allow asynchronous I/O, so every write operation attempted by the database writer must complete before it can go on to the next write. Furthermore, some Unix filesystems serialize write operations to a single file, defeating any attempts to improve I/O performance by using multiple database writer I/O slaves. This problem actually causes one I/O slave to block others for writes to a particular datafile, ultimately slowing down overall operation. Asynchronous I/O allows write operations to be issued much more rapidly than synchronous operations, dramatically decreasing the overall time required to complete a checkpoint. Most platforms support asynchronous I/O on raw device datafiles, and some filesystems also have their own software implementations of asynchronous I/O. Many checkpoint slowness problems can be overcome by switching to asynchronous I/O.

If asynchronous I/O is not a possibility, the database writer can be set to checkpoint more aggressively. From 8i forward, you can switch to fast-start checkpointing. This will increase the frequency of checkpoints, but will ultimately allow checkpoints to complete in a more timely manner. In 10g, the aggressiveness of the checkpointing activity is automatically adjusted. If you provide no log_checkpoint… parameters, 10g will use fast-start checkpointing by default and automatically adjust the fast_start_io_target.

Checkpoint currency and recovery

In the event Oracle must perform instance (crash) recovery, the speed with which it completes depends largely upon how recently the last checkpoint occurred. When Oracle performs instance recovery, it must apply all redo entries from the online redologs from the time the database last checkpointed each datafile, forward to the time the database went down. Therefore, to minimize downtime in the event of an instance failure, the database should be kept checkpointed to as recent a point in time as possible, preferably with fast-start checkpointing.

If a database host has multiple processors, instance recovery can be performed in parallel to minimize the time required to open the database. By appropriately setting the values of parallel_max_servers and recovery_parallelism, any recovery will be performed in parallel, and should complete faster than a single-threaded recovery.

When deciding on values for these settings, some experimentation should be performed in order to determine the best recovery times for any particular architecture. Some papers have stated that no benefit from parallel recovery can be realized with less than eight parallel processes, but since architectures vary widely, so will any given result.

Recovery from user oopses

Even with strict access policies and privilege controls, it is still likely that someone will accidentally drop a table, update rows or delete critical data. This may be data that a critical application needs just to run. Such user oopses can result in extended downtimes for an application or service.

One of the best new features in Oracle 10g for mitigating the impact of a DML-related oops is flashback query. Instances that use server-managed undo instead of rollback segments can perform consistent reads of data as it appeared in the past. Using the same multiversion consistent read mechanism that allows Oracle's writers never to block readers, undo is applied to data to provide output as of a time in the past.

Flashback query is simple to use, and allows previous versions of data to be queried from a select statement, using the "versions between" clause. After a user oops, it is critical to perform the flashback query as quickly as possible to get the data back, or the undo retention time may expire. To minimize the time spent writing the results, you can "create table . . . nologging as select . . . versions between . . . ;"

The flashback drop feature which is supposed to protect from accidental table drops, is less useful, and is only related by name to flashback query. When you drop a table in a database with flashback drop enabled, it actually just renames it, until the db_flashback_retention_target expires, at which point it is dropped.

Avoiding reliance on DNS

When setting up listener.ora and tnsnames.ora addresses, either the IP address or DNS name of the target machine must be specified. Some prefer to use the DNS name rather than the IP address for reasons of transparency. The problem with using DNS names for Net8 addresses is that it causes you to rely on name resolution via DNS (prone to outage) or a local hosts file (prone to corruption).

In order to eliminate the hosts file and the DNS as points of possible failure for an Oracle Net8 implementation, IP addresses should be used in listener.ora and tnsnames.ora entries for critical services. Although the transparency benefits of using DNS are lost, mistakes in hosts file or DNS configuration cannot negatively impact the ability of clients to establish connections via a Net8 listener.

A larger lesson can be gleaned from this tip. When considering configurations for any critical service, make sure you rely on as few points of failure as possible. The more external services you make a system rely on, the more points of potential failure you create.

Tips for Humans

Although availability seems like a technical issue, it is also largely a human issue. A great deal of downtime can be traced not to technical failures, but rather to human failures. Just a few of these are:

- Lack of diligence or followup on previous problems
- Panicking under pressure of an outage
- Laziness
- Failure to grasp the financial impact of the DBA's role in a company
- Lack of planning
- Failure to focus on the most probable problems

My last few tips therefore address the DBAs themselves, and how they can prepare themselves for the inevitable major failure over which they will one day preside.

Remote access/effective communications

Essential to any 24/7 operation is a dependable remote access capability. DBAs and other key support staff need to be able to quickly log in from home or the road in order to resolve problems. In many IT organizations, several different people may be called on to assist in problem solving, so remote access modem pools or VPNs should be large and dependable enough to handle many connected users. Depending on the criticality of operations, some larger organizations may decide to keep DBA and systems staff on site at all times.

I often hear that DBAs are not provided with company-funded pagers or mobile phones. Any company that expects their DBAs to respond to emergencies must provide a means to do so. It is up to the DBA to insist on this. A couple of unattended major outages will be all that it takes to clear up this misunderstanding between DBAs and management.

On-site operators

Even if all staff is not on site, a staff of one or two systems operators located in the same place as the physical systems can be indispensable in an emergency. They can page and call people, maintain a conference call, and visually check systems. DBAs and sysadmins should be able to devote 100% of their time to fixing the problem in an emergency. If they have to waste time paging and calling people, it just extends the downtime. System operators who perform this support role, such as those at Amazon.com, are the true heroes of high availability.

Bridge line

There is no substitute for a commercial conference system that can bridge together 20 to 50 people on a single call. When all the minds come together, ideas for resolution are more likely than with one person pecking away in the dark.

Application failure tolerance

Despite the best efforts of any staff, outages will take place from time to time, and software must be prepared to

handle that eventuality gracefully. By designing applications in such a way that they display a friendly message, such as "This service is temporarily unavailable—Please try again later," the impact of outages on users can be lessened. The advantage of presenting a clear, friendly failure message to users is that it makes the outage less cryptic and confusing to users. Without such functionality, users see error messages that mean nothing to them and give no indication that anyone is aware of a problem. With a "service unavailable" message, users can be told that the problem is known, and that people are working to correct it. Furthermore, the DBA rushing to solve the problem does not have to answer as many calls asking if she is aware of the problem.

Dealing with new and unfamiliar problems

It is a good idea to develop a game plan for unexpected outages. In most cases, the original root cause of a problem will require extensive analysis to uncover. For this reason, decide ahead of time on how a severe problem will be approached and managed. A diagnostic plan should be developed and made well known to all DBA staff.

Often the corrective action for a particular problem will have the side effect of eliminating any evidence of the cause of the problem. For example, a problem with the SGA will be completely cleared by restarting the instance. DBAs should therefore be able to rapidly perform some cursory diagnostic actions, then proceed to concerted efforts toward the task of restoring service.

Taking this approach is especially effective upon encountering the first occurrence of a particular problem. After restoring service, ample time can be spent on thinking through the root cause, and developing strategies for better diagnosing or correcting the problem if it ever happens again. The DBA's ability to perform post-event root cause analysis is completely dependent on the quality of data recorded during the event.

Chief among the tools for obtaining such data is the systemstate dump. Logged in as a user with DBA privileges, a systemstate dump must be performed when connected to the database through a dedicated server process.

The SQL statement for generating a systemstate dump is:

```
SQL> alter session set events 'immediate trace name systemstate level 10';
```

A systemstate dump will create a trace file in the instance's user_dump_test, containing the wait states and resource dependency hierarchies of all sessions logged in. systemstate dumps are primarily for hanging situations and resource pileups. Oracle has written a useful awk script that parses systemstate dumps and provides a clear, readable output. Ask your Oracle Support analyst for a copy of ass.awk (I'm not kidding). Many other diagnostics can be easily and quickly dumped, depending on the type of problem that a system is encountering. The information contained in the resulting trace files can be valuable in diagnosing a wide variety of database problems, and can also be helpful when working with Oracle Support.

In addition to database diagnostics, it is also important to check host resources, such as memory, swap, and disk space. Many of these emergency diagnostic techniques can also be run routinely at periodic intervals, as part of a meticulous monitoring regime. If particular areas cause problems, additional diagnostic queries and tests can be added to monitoring scripts to provide early warning of an impending problem or better post-outage diagnostics.

These guidelines are limited in that they cannot take into account the particular nature or circumstances of any particular critical problem. The best diagnostic methods are specific to a problem, and successful root cause analysis depends on the ability of the DBA to investigate. In a nutshell, postpone as much analysis as possible for after the system is back up. Get your diagnostics fast and return the system to service as best you can. Remember, availability means having the host up as much as possible, not necessarily understanding why it last went down.

Conclusion

Much of the availability revolution consists of a shift in attitude. Those who have previously regarded downtime as inevitable, or as a necessary evil, will adapt to increasing availability requirements, or fall by the wayside. Over time, more and more DBAs will lose their scheduled weekly or monthly outage windows as their companies conclude that any and all downtime is unacceptable. As the volume of business transacted on the Internet grows, so will the amount of revenue lost per second of downtime.

Very visible public outages can be damaging to a company's credibility. They not only have the potential to be expensive in terms of lost revenue, but they also diminish customer confidence, projecting an incompetent image to the public. Certain sites have experienced notorious outages that people remember and discuss long after the event has passed. Because lengthy downtimes are so preventable, an attitude must be adopted among IT management and staff that treats availability issues very seriously. Adopting a laissez-faire attitude toward availability leaves a company's success up to luck, and outages resulting from such an attitude may paint a deservedly incompetent image of a company.

Oracle's focus on availability has increased with each successive version of the RDBMS, reflecting user demand for such features. While some sites will not need all of the new availability features and will not need to employ every technique outlined here, other sites will find them indispensable, and all would be wise to consider them. ▲

Jeremiah Wilton was the founding DBA and a 6-year veteran of Amazon.com's database group. Appearing frequently at the lecterns of U.S. and overseas Oracle conferences, he enjoys sharing his novel approaches to availability and scalability with the world community of Oracle DBAs. He was a recipient of an honorary Oracle Certified Master certificate in 2000, and a keynote speaker for IOUG-A at Oracle Openworld 2000. Jeremiah now teaches seminars and classes to groups on crisis management and Oracle administration. He is also available for remote emergency DBA services. For more information, visit his website at http://www.speakeasy.net/~jwilton.

Many Thanks to Our Sponsors

NoCOUG would like to acknowledge and thank our generous sponsors for their contributions. Without this sponsorship, it would not be possible to present regular events while offering low-cost membership dues. If your company is able to offer sponsorship at any level, please contact NoCOUG president Roger Schrag at rschrag@dbspecialists.com. ▲

Long-term event sponsorship:

LOCKHEED MARTIN

CHEVRONTEXACO

ORACLE CORP.

Thank you! Year 2004 Gold Vendors:

- Churchill Software
- Confio
- Database Specialists, Inc.
- DataMirror
- Embarcadero Technologies
- EMC Software
- Quest Software
- Quovera
- Verican
- Veritas

For information about our Gold Vendor Program, contact the NoCOUG vendor coordinator via email at: vendor_coordinator@nocoug.org

TREASURER'S REPORT

Judy Lyman, *Treasurer*

Beginning Balance		
July 1, 2004		**$ 73,545.93**
Revenue		
Membership Dues	2,830.00	
Meeting Fees	1,600.00	
Vendor Receipts	3,000.00	
Training Day	307.95	
Advertising	---	
Interest	24.66	
Total Revenue		**$ 7,762.61**
Expenses		
Regional Meeting	13,534.51	
Journal	5,806.15	
Membership	---	
Administration	---	
Website	150.00	
Board Meeting	506.95	
Training Day	307.95	
Marketing	---	
Paypal	72.42	
Miscellaneous	22.00	
IRS	---	
FTB	---	
Total Expenses		**$ 20,399.98**
Ending Balance		
September 30, 2004		**$ 60,908.56**

OraPub Class Review

Class reviewed by Brian Hitchcock

Class Title: Reactive Performance Management
Instructor: Craig Shallahamer
September 2004, Cupertino, CA
OraPub, Inc.
www.orapub.com

Summary

Overall review: Excellent, possibly the best training experience I have had.

Target audience: Anyone who needs to solve application performance issues where an Oracle database is involved.

Would you recommend it to others? Yes, if you are experienced with Oracle architecture and tuning.

Who will get the most from this class? The more experience students have with their operating system and actual hands-on Oracle tuning, the more they will get from this class. This is not the class for students who are new to Oracle and have not dealt with performance issues before.

Is this class platform specific? The focus is on applications using an Oracle database running on a Linux or UNIX operating system. However, the techniques and overall philosophy can be productively applied to any database on any operating system. Certainly, the course is targeted to systems using Oracle databases, and the database server used in class for the hands-on exercises was a Linux server.

Overall Review

The class focused on Oracle performance tuning by including the operating system, the Oracle database, and the SQL of the application. This approach allows you to find the real bottlenecks that may exist in one or more of these areas. The traditional approach to Oracle tuning is to look only at the database, and perhaps the application SQL that is moving through the database.

The class was three days long, with about 80% instructor-led presentations and 20% in-class, hands-on exercises. The other participants were all full-time Oracle DBAs, and the way the instructor ran the class encouraged everyone to participate. Compared with other classes, I got much more time with the instructor and the other students, which makes this class a better value than most.

I attended this class specifically for its focus on the bottlenecks that can exist in the operating system layer of Oracle database systems. In my own work experience, I haven't had much success in the past looking at the OS level, and I've seen cases where this limited what I could do. Even when I found the database bottleneck, I wasn't confident I could relate it to what was going on at the OS level.

The general approach taught in this class is that you have to be confident in your abilities to find performance issues in all three areas of the overall system: application SQL, database, and operating system .

This approach is very different from what I've seen before in other classes and in the popular Oracle press. While it was very familiar as far as the details of how Oracle works, the integration of the operating system was refreshing. I think many DBAs don't want to see this level of complexity, preferring to stay in their comfort zone, focused on the details of how they think the Oracle database works. (I'm not sure a DBA can truly understand how Oracle works without understanding the interactions with both the application and the operating system, but I see a lot of DBAs who attempt this.)

Now that I have more experience with looking at all three aspects of performance, I will be more adamant that all three areas must be examined at once. In many cases, I am asked to verify that there *isn't* a performance problem, which can be much harder than dealing with a situation where there is at least the perception of a specific performance issue. After attending the class, I am much more confident that I can offer my customers detailed explanations of why I don't think their system is having a problem and, where I would expect the problems to come from as their application grows over time.

The Course Materials

The course materials are presented in two volumes, the **Core Material Note Set** and the **Reference and Optional Evening Sessions Note Set**. Additionally, each student was supplied with a tar file containing all the scripts needed to perform the in-class hands-on exercises.

In addition to the two volumes of course materials, the class had a Linux server running Oracle 9i. We used this for the three in-class hands-on exercises that involved performance troubleshooting

What Was Good

The focus of the course was examining all three areas of overall performance management, namely the application (specifically the SQL that is being processed by the database), the Oracle database and the operating system that the database is running on. This focus on all three areas is rare and very valuable. The operating system is usually forgotten when Oracle DBAs start in on a performance issue.

Whether this is due to ignorance or fear or lack of experience, the result is the same. The odds of producing an accurate, verifiable diagnosis of the real performance issue are reduced. Further, without an understanding of what is going on at the OS level, the ability to come up with the best options for resolving the performance issue are also reduced.

While you may find that the specific performance bottleneck is in the Oracle database, if you have also been looking at the operating system level you would know if you have spare CPU, memory, or I/O resources available. This knowledge gives you a head start on selecting possible solutions. If you know you don't have spare CPU capacity, you don't want to pursue tuning solutions that require more CPU if that isn't feasible for the system you are working on.

Identifying performance issues

The class emphasized being able to confidently identify the performance issues. The standard tuning course puts the emphasis on how to solve performance issues. In a complex, globally deployed application environment, it can be very difficult to identify the true performance bottleneck. If this isn't done correctly, there isn't much value in being able to solve any performance issue.

The instructor made a very important point as the class was getting started. DBAs who understand the OS issues involved with overall performance management are more valuable. This is something to think about as outsourcing—and the more real threat of automation—continue to change the nature of the DBA job. If you have an interest in staying employed, this point must not be ignored.

Gathering data

Before we got to the details of overall performance management, the difference between summary data and data gathered over a specific interval was examined. This was very valuable, and based on the questions asked by some of the students, it is not something that we can assume everyone has dealt with before. (This is another example of how a focus on details can cause you to lose sight of the bigger picture.)

A graph of buffer cache hit ratio was shown with a red line plotting the cumulative buffer cache hit ratio. A blue line plotted the same data, but it plotted the buffer hit ratio for each smaller interval within the overall time span. The red line was much smoother since the data, over time, tends to settle to some average value. The blue line showed what the data was doing over much smaller time intervals, and it showed much more variation. After this, students would routinely discuss what they saw during the in-class exercises and ask each other if the data was "red line" or "blue line." This concept is very worthwhile, and the fact that this was brought out early, in such an easily understood way, shows how much this class is focused on *understanding* versus simply teaching techniques.

Teaching an approach

It was refreshing that the instructor was not trying to sell a particular tool or product. Instead, the instructor was teaching an approach that isn't dependent on any tool or product. For the class itself, the instructor provided a set of scripts that students used to identify the performance issues on the database server that was available in class. These scripts were given to each student to use indefinitely.

The instructor also explained that what was being taught was largely focused on the performance of the Oracle database, the SQL passing through it and the OS supporting it. In many real-world applications, there are many more components that can affect overall performance than just the Oracle database. This class does not address SQL tuning, or how an application server can affect overall performance, for example. This assumption needs to be understood by the general DBA community. Having expertise in the details of how Oracle works may not have any value to your business if you can't help the business identify the true performance problem. Many times, in my experience, the DBA will be called on to verify that the Oracle database is not the performance problem. This can be difficult for DBAs who have been trained to believe that any wait event is evidence of a poorly performing database.

Performance bottlenecks

The course included detailed examinations of the cause of the major types of Oracle database performance bottlenecks such as contention for latches, shared pool, and redo resources. These explanations went a lot deeper than

TECH TIPS

DBA Click Resource

Here's yet another place to click your way to more Oracle resources. DBAClick.com offers Oracle scripts, installation guides, manuals, tips, white papers, books, tools reviews, news, and links. There's also a FAQ section, Oracle DBA forums, and instructions on how DBAs can upload and share their own favorite scripts and resources on the site. You'll find it all at ***http://www.dbaclick.com***. Some of the categories on the site include:

- Oracle Backup/Recovery Scripts
- Oracle DBA Scripts
- Oracle Export/Import Scripts
- Oracle Monitoring Scripts

what I had seen over the years in the Oracle manuals and in many of the books and white papers available.

In-class exercises

During the class, there were three in-class exercises using the in-class Oracle system. These in-class, hands-on examples were great because they really gave a feel for what it is like to have to look at lots of data and figure out what is happening. The source of the performance problem was not obvious and multiple scripts had to be used to review the OS, the application SQL, and the database before the student could determine what to do next. Overall, this was the single most valuable part of the course, and I believe it is this unique approach to getting students to actually experience the approach being taught that makes this class so valuable.

OS-level I/O issues

A specific topic that was very meaningful to me was that of OS-level I/O issues. The instructor was very clear that he was teaching us how to identify I/O issues but that after that, you must engage the experts that support the I/O subsystem for the application. This is something that means a lot to me because I support many systems where the I/O subsystem really is a black box. I have no control over the configuration of the disk, arrays, controllers, fiber channels etc., and therefore I can't provide any value in resolving I/O subsystem issues. The same thing applies to network issues. Being able to identify the performance issue is the key task. Once identified, the DBA may need to let other experts resolve the issue depending on where the problem is. Too many DBAs think they are experts in all areas, and this wastes time. Once the DBA has identified an I/O issue from the Oracle database perspective, for example, the experts that handle the I/O subsystem must be the ones to resolve the issue.

Overall, [the in-class exercise portion] was the single most valuable part of the course, and I believe it is this unique approach to getting students to actually experience the approach being taught that makes this class so valuable.

What Could Be Better

When looking at the Oracle database, this class still assumes that the response time is between the Oracle client and the Oracle server. This may be true in some systems, but as more and more businesses are connected to customers that are farther away, the true meaning of response time must expand to include everything between the end user and the Oracle database. The response time as seen by the end user starts when they hit Return and ends when their screen is completely updated ready for the next user action. While this is typical of Oracle classes, it avoids a whole class of performance issues that can't be found or solved within the Oracle world. I would strongly encourage the instructor to make this point very clear on the first day.

Similarly, I would like to see more emphasis on what is the real goal of any performance management task, namely, to help the business make money. It doesn't matter how well a database is tuned if the overall business process is not making money. Focusing on the performance management at the application, database, and OS levels is very, very good, but I think the context should be set early on that this is all within the broader picture of a business process that needs to serve the business. No one cares if Oracle or an application is slow or fast; what really matters is moving the business forward. I think the question needs to be asked at every step of the performance tuning process, "What is the business value of the work we are doing?" Everyone working on the project must be able to answer this question before any of the detailed performance tuning work begins. I believe DBAs need to look at the value to their business of everything they do. I believe DBAs that ignore this do so at their own peril.

Scripts

The scripts supplied as part of the course were great; they worked as advertised and were a valuable part of the learning experience. However, as tools for the students to use long term, these scripts require maintenance. Since the structure of the scripts was not examined, the amount of maintenance needed as we move from one Oracle database version to the next is unknown. Students who depend on these scripts to perform their tuning tasks back on the job may find that the scripts fail at the most inopportune time. I would like to see more emphasis placed on what the scripts were doing for us so we would be better able to support them in the future. More understanding of what the scripts were doing would also allow us to decide if each script was the best way to gather the relevant data in our specific environment.

The course makes no mention of STATSPACK. This seems odd to me. Granted, I focus on STATSPACK because it is what I can use when I support production performance issues in my work environment. Because of this, some of the data supplied by the scripts used in class will be easier to get, in my work environment, from STATSPACK.

The scripts also require some amount of installation. This was done for us on the in-class server before class. This was great for us during the class, but it meant that when I arrived back at work, I didn't know ahead of time what I would have to do to install the set of scripts. A single page in the class notes that showed the steps to install the scripts would have solved this problem. I would have liked to know, before I left class, which Oracle database user would own the objects created by the scripts and what privileges this user would need. This would have allowed me to determine which of the scripts I could use in my work environment where I have to work within some very specific security restrictions.

More in-class exercises

The greatest strength of this class also points to the biggest area where I think the value could be increased substantially. I'm referring to the in-class exercises. As I've stated previously, these were the most valuable part of the class and they set this class apart from the typical tuning courses available. I would like to see more of the class dedicated to more of these exercises. I realize that it may not be practical to set up more cases where the students actually run the scripts to determine the problem. For those cases where it isn't practical, it would be valuable to have the instructor walk through output from the scripts showing the results.

Specific tuning issues I'd like to see addressed in class, with real data from real systems, would include the following situations: database machines that have multiple CPUs, systems that have application servers, multiple databases on a single machine, remote databases with database links used between them, distributed transactions, and standby or other disaster recovery systems in place. I believe most of the students returned to work environments where these issues are present. I would like to see discussion of how to manage performance when you have to look at the performance on the local database/app/OS and the remote database/app/OS at the same time.

I realize that it may not be practical to bring a 6-way box to class. However, I know from my work that the issues listed above come up all the time. I suggest that they be addressed in future versions of this class, or perhaps as a separate class. I'm not asking for in-class hands-on exercises for two databases on two continents. These issues could be covered with charts showing data from tuning work done on systems with multiple databases on multiple machines, etc. Perhaps the class material could show how the data for these cases differ from what is seen in class for the simpler cases.

Since I am advocating more in-class focus on more complex environments, I'm also advocating a reduction in the class time devoted to the details of how Oracle works. I did learn more about how the database does what it does, but I would have gained more useful knowledge from seeing more case studies. I know that the details of Oracle internals are very popular, but I don't think it has nearly the business value of a review of more tuning cases, especially cases that involve issues that are very likely to come up in modern application systems.

Finally, some of the videos were simply awful. Just say "No" to *9i Man*. Whoever produced these must have been an agent from Redmond.

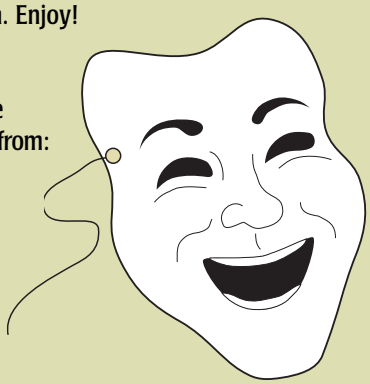
What Was It Like in Class?

Yes, you had to buy your own lunch, and dinner after

Comedy Corner—Laughter Is the Best Medicine

By Laurie Robbins

With the stresses of work and family, we all need to take a break and have a good laugh. Enjoy!



DBA Bloopers

Eleanor Roosevelt said it best when she said, "Learn from the mistakes of others." On the SearchDatabase.com, there are a few DBA bloopers that should be fun to read *and* to learn from:

True DBA blooper #2: Small slips, big data wipeout
A simple typo and a misplaced double click was all it took for these two DBAs to wipe out entire directories of critical files. They recount their horror here.
True DBA blooper #3: Wildcard runs wild
While implementing a new file path naming standard at his company, DBA Ron Horjus thought he would take advantage of the wildcard.
True DBA blooper #5: Mind your `s and 's
Oracle DBA Brian Bientz relates this harrowing tale of what happens when one confuses a ' with a ` . . .
You'll find these stories and more at ***http://searchdatabase.techtarget.com/bestWebLinks/0,289521,sid13_tax292843,00.html.***

The Computer Guy

Do you remember the *Saturday Night Live* skit "Nick Burns the Computer Guy"? Well, if you don't or just want to watch again, try one of the links below. The file will download to a temporary folder, then immediately start to play or right-click on the link to download to your system.
http://mrt300.ods.org/snl/view.php?computerguy
Jennifer Aniston; November 20, 1999 (5:14 minutes)
OR
Jackie Chan as Wang; May 20, 2000 (3:42 minutes)

Acknowledgements

Thanks to Craig Shallahamer for "Nick Burns the Computer Guy" links.

the second day of class, if you choose to attend. I highly recommend you attend lunch and dinner. It was at the dinner that I learned more than I should have about the many personal conflicts going on behind the scenes in the highly charged world of Oracle consulting. Who knew? It sounds quite exciting. The lunch and dinner gatherings provided lots of time to discuss many Oracle questions and find out what others are doing in the field. This is a very good complement to the formal instruction in class and adds to the value of the training experience.

Day 1 – the Instructor impressed us with the fact that the covers to both volumes had been printed in color. One wonders if color printing is just now making its way to Oregon. One wonders what other marvels will follow. One assumes the instructor was attempting to be humorous. The afternoon brownies were great.

Day 2 – Lunch caused an embarrassing situation. Our group of highly experienced technical professionals decided to try the restaurant next to the hotel where the class was being held. Too bad the restaurant turned out to be a bank. (Maybe we should have studied foreign languages after all.)

Day 3 – Due to watching too many videos, time ran short, so we had pizza in class for lunch. Pizza was great—doing the math when the bill had to be paid was challenging.

The Instructor: Craig A. Shallahamer, from OraPub, Inc.

Mr. Shallahamer's 18-plus years of experience in the IT marketplace brings a unique balance of controlled creativity to any person, team, or classroom. As the president of OraPub, Inc., his objective is to empower Oracle performance managers. His specializations include "doing" and teaching others to "do" whole system performance optimization (reactive and proactive) for Oracle-based systems.

The website has lots more details on Mr. Shallahamer if you wish to know more about him and his many experiences.

Conclusion

Overall, this class was great, one of the best training experiences I have ever had. It is rare that I am excited about the daily DBA tasks that I face most of the time, but this class made me look forward to tuning tasks. This is not something I was expecting and speaks to the quality of the instructor and the class content. Note that unlike most, if not all, training offerings, OraPub classes are written and delivered by the same person. Craig teaches all the classes himself; no one else will show up to lead the class after you have paid your money. This means the OraPub instructor is highly motivated to deliver.

I don't get to go to training very often. This was the first class I've been sent to in over three years. Therefore, I had a very definite reason for wanting to go to this class in particular; I wasn't sent just to satisfy some corporate training requirement.

Many of the training classes I have attended were led by instructors who simply read from a manual and rarely had any real-world experience. Craig doesn't run his classes this way. He prepares the class materials and provides in-class hands-on exercises that force you to practice what is presented. The in-class exercises are the most valuable part of this class because they can't be faked; you have to understand what you have been taught to work through the exercises.

This class was very valuable to me, and allowed me to go to work the very next day and be more productive, a very good and all-too-rare training value. If you are looking for an Oracle performance training class, I don't know of anything that comes close.

Finally, how do I know that this class was valuable? The Monday after I took the class I was tasked with finding the bottleneck in a performance test system that had suddenly started to run poorly. I applied what I had learned and indeed, I did find the bottleneck quickly, without lots of trial and error. The conclusion is clear, the focus on the OS aspects of Oracle tuning allowed me to be more productive the very next day I was at work. It is hard to top this for proof of the value of this class. ▲

Brian Hitchcock has worked at Sun Microsystems in Newark, California, for the past nine years. Before that he worked at Sybase in Emeryville, California. Even further into the past he spent 12 years at Lockheed in Sunnyvale, California, designing microwave antennas for spacecraft. He supports development databases for many different applications at Sun, handling installation, tuning, NLS and character set issues as well as Unicode conversions. Other interests include Formula One racing, finishing a Tiffany Wisteria lamp, Springbok puzzles, Marklin model trains, Corel Painter 8, and watching TV (TiVo® rules!). Brian can be reached at **brian.hitchcock@aol.com** *or* **brhora@aol.com**.

NoCOUG Fall Conference

Thursday, November 4, 2004 · Computer History Museum, Mountain View, CA

Please visit www.nocoug.org for session abstracts, directions to the conference, and to submit your RSVP.

Session descriptions can be found at www.nocoug.org

8:00 a.m. **Registration and Continental Breakfast**

9:00–9:30 **General Session and Welcome**

9:30–10:00 **Keynote:** *Learning by Fire* by Craig Shallahamer, President, OraPub, Inc.

10:00–10:30 **Keynote:** Computer History Museum Welcome and Introduction

10:30–11:00 **Break**

11:00–12:00 **Parallel Sessions #1**
Track 1: *Conquering Oracle Latch Contention* by Craig Shallahamer, President, OraPub
Track 2: *Use EXPLAIN PLAN and TKPROF to Tune Your Applications* by Roger Schrag, President, Database Specialists, Inc.
Track 3: *Visual and Declarative J2EE Development with Oracle JDeveloper 10g and ADF* by Blaise Ribet, Principle Product Manager, Oracle Corporation

12:00–1:00 **Lunch**

1:00–2:00 **Parallel Sessions #2**
Track 1: *Resource Mapping: A Wait Time Based Methodology for Database Performance Analysis* by Matt Larson, CTO and founder of Confio Software, Confio Software
Track 2: *Common Performance Monitoring Mistakes to Avoid* by Virag Saksena, CEO, Auptyma
Track 3: *I Love the Java Jive: J2EE Overview for Oracle Technologists* by Peter Koletzke, Technical Director and Principal Instructor, Quovera

2:00–2:15 **Break**

2:15–3:15 **Parallel Sessions #3**
Track 1: *Optimizing Oracle9i Instance Memory* by Lenka Vanek, Sr. Product Manager, Quest Software
Track 2: *Indexing Strategies in Oracle—An Overview* by Scott Martin, President, Terlingua Software
Track 3: *Introduction to Java—PL/SQL Developers Take Heart!* by Peter Koletzke,Technical Director and Principal Instructor, Quovera

3:15–3:45 **Break & Raffle**

3:45–4:45 **Parallel Sessions #4**
Track 1: *Managing Oracle Application with Oracle Grid Control* by Valerie K. Kane, Group Manager, and Lars Ewe, Senior Principal Product Manager, Oracle Corp.
Track 2: *Oracle 10g RAC on Red Hat Enterprise Linux 3 with Direct I/O* by Sunil Mahale, Network Appliance
Track 3: TBA

4:45 -?? **NoCOUG Networking and Happy Hour**

Cost: $40 admission fee for non-members. Members free. Includes lunch voucher.

RSVP online at www.nocoug.org/rsvp.htm

(COVER PHOTOS COURTESY OF THE COMPUTER HISTORY MUSEUM.)

NoCOUG
P.O. Box 3282
Danville, CA 94526

FIRST-CLASS MAIL
U.S. POSTAGE
PAID
OAKLAND, CA
PERMIT NO. 1770

Don't forget to renew your NoCOUG membership for 2005!