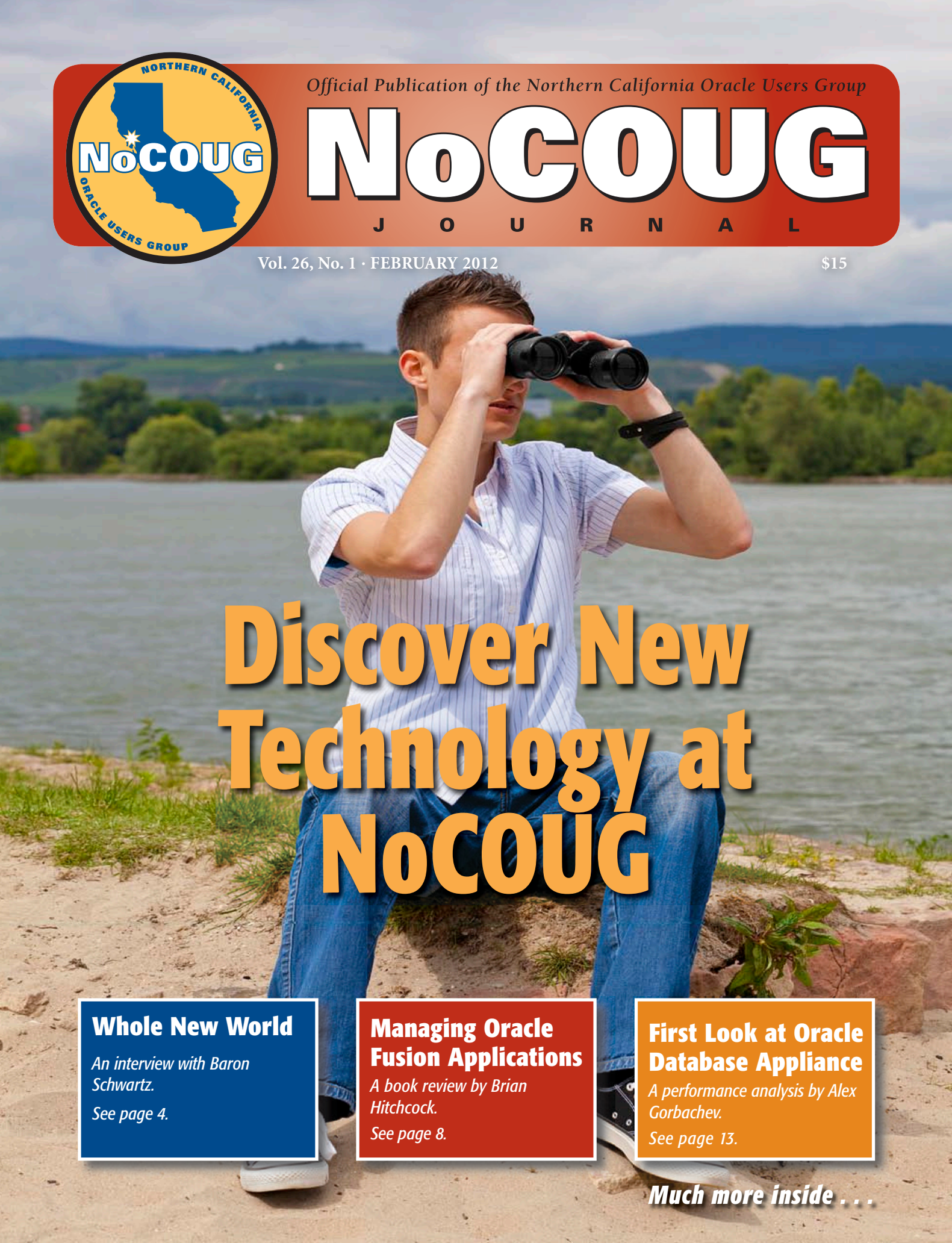
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 26, No. 1 · FEBRUARY 2012 $15



Discover New Technology at NoCOUG

Whole New World

An interview with Baron Schwartz.

See page 4.

Managing Oracle Fusion Applications

A book review by Brian Hitchcock.

See page 8.

First Look at Oracle Database Appliance

A performance analysis by Alex Gorbachev.

See page 13.

Much more inside . . .

Thanking the Team

Take a moment to think about the huge amount of effort that goes into this publication. Your first thought might be about the care and attention required of the authors. Yes, writing is hard work. Now consider each author's years of hard-won experience; then add it up. The cumulative amount of time spent to acquire the knowledge printed in each issue is decades—maybe even centuries.

But let's take a moment to thank the people who make it possible for us to share this knowledge with you. Without the dedication and skill of our production team, all that we'd have is a jumble of Word files and a bunch of JPEGs. Copyeditor Karen Mead of Creative Solutions transforms our technobabble into readable English. Layout artist Kenneth Lockerbie and graphics guru Richard Repas give the *Journal* its professional layout.

Finally, what really distinguishes this *Journal* is that it is actually printed! Special thanks go to Jo Dziubek and Allen Hom of Andover Printing Services for making us more than just a magnetically recorded byte stream. ▲

—*NoCOUG Journal* Editor

Table of Contents

Interview 4

Book Review 8

Performance Corner 12

Special Feature 17

President's Message 22

Sponsorship Appreciation 24

Session Descriptions 25

Conference Schedule 28

ADVERTISERS

GridIron Systems 2

Quest Software 11

Apress 15

HiT Software 21

Delphix 23

Embarcadero Technologies 23

Confio Software 23

Quilogy Services 23

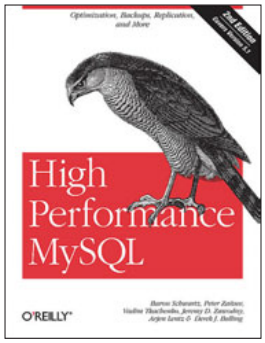
Database Specialists 27

Publication Notices and Submission Format

The *NoCOUG Journal* is published four times a year by the Northern California Oracle Users Group (NoCOUG) approximately two weeks prior to the quarterly educational conferences.

Please send your questions, feedback, and submissions to the *NoCOUG Journal* editor at **journal@nocoug.org**.

The submission deadline for the upcoming May 2012 issue is February 28, 2012. Article submissions should be made in Microsoft Word format via email.



Whole New World

with Baron Schwartz


Baron Schwartz

"A whole new world
A new fantastic point of view
No one to tell us no
Or where to go
Or say we're only dreaming."

—Oscar-winning song
from the movie Aladdin

Baron Schwartz is the chief performance architect at Percona. He is the lead author of High Performance MySQL *and creator of Maatkit and other open-source tools for MySQL database administration. He is an Oracle ACE and has spoken internationally on performance and scalability analysis and modeling for MySQL and PostgreSQL, and on system performance in general.*

When did you first hear of MySQL, and why did you choose to get involved? Which database technologies did you use before joining the MySQL camp? What have been your contributions to MySQL since then? What are your current projects?

I became acquainted with MySQL in 1999, when I was getting my undergraduate degree at the University of Virginia. I didn't know a lot about relational database technology at the time, and my experience was limited to a single very academic class taught by a graduate student whom I could barely understand. I finished that course with an understanding of sigmas and other funny words, but with absolutely no concept of how to really use a database server. The first time I used MySQL in a serious way is when I joined an outdoors club at the university. It was painfully obvious to me that clipboards and pieces of paper were never going to be able to meet the demand, and in fact it was all we could do to organize the events with about 30 people attending. I realized that if I built an online application for organizing the club, we could scale to several hundred members without much trouble. The club is still going strong about a decade later.

After graduating from university, I joined a company that used Microsoft SQL Server exclusively. There, I was fortunate to work with a very talented database administrator, who taught me how database servers work in the real world. I stayed there for three years, and when he left to join another company, I followed him. That company used MySQL, and the day that I walked in the door it was clear that the growing pains were severe. Together with several other people, we got past those hurdles, and the company is still running on MySQL today. Along the way, I began blogging, traveling to attend conferences, and meeting a lot of other people in the MySQL ecosystem. Something about relational database technology fascinates me—and I'm still not quite sure what that is, but I know I love working with open source and I love working with databases. The combination of those two loves has made my career very satisfying and rewarding.

To be clear, I think Microsoft SQL Server is also a fantastic product. In fact, it is superior in many ways to MySQL. I miss the instrumentation and the great tools. Many of the things that I have done in my career have been targeted at solving some of those shortcomings in MySQL.

This really began in 2006, when I started writing what eventually became Maatkit. That suite of administrative tools was pretty universally acknowledged as essential for MySQL users, and although I discontinued that project last year, all of the code lives on in Percona Toolkit, along with another open-source project that I started. I am a command-line geek first and foremost, but today I recognize that people also need graphical tools. That's why my newest project is a suite of web-based tools, which you can find at **http://tools.Percona.com**.

Have Sun and Oracle been good stewards of MySQL? Has the pace of development slowed or sped up under their stewardship? Can we expect Oracle to contribute its technology gems such as [pick any feature not currently available in MySQL such as parallel query and hash joins] to MySQL?

I think that Sun and Oracle have been fantastic for MySQL. Oracle, in particular, has really figured out how to get the product out the door. The MySQL 5.5 release is the best in MySQL's history, in my opinion. There's a little less visibility into how development is progressing than there used to be,

"There was a lot of speculation that Oracle would simply kill MySQL or that they would let it stagnate. History has not borne out those fears."

but they release milestone previews of the upcoming version at intervals, and what I have seen so far for the next version of MySQL is very promising.

I'm not sure how far Oracle is going to take MySQL. There was a lot of speculation that Oracle would simply kill MySQL or that they would let it stagnate. History has not borne out those fears, and I never thought they were well founded in the first place. It is such an emotional issue for some people that I tend not to participate in those conversations, and instead I simply thank Oracle for the good work that they're doing.

What are the best use cases for MySQL in the enterprise?

It's risky to paint enterprise applications with a broad brush, but in general I would say that large applications can be designed successfully in several different ways. Sometimes the application itself is complex, but the database is not expected to do a lot of heavy lifting. For example, many Java applications do a lot of computation outside the database. On the other hand, many applications do centralize the logic into the database, especially when there is a plurality of applications accessing the same data store. These two approaches really represent APIs to the data; in one case the API is built in application code, but in the other case it's built in stored procedures. When I used Microsoft SQL Server, we took the latter approach, and it's anyone's guess how many millions of lines of stored procedures we had in our databases. That is not an approach that I would advocate using for MySQL. Instead, the access to the database should go through an API that is designed and implemented separately from the database. Web services are the obvious example.

Why are there not TPC benchmarks for MySQL? Is MySQL unsuitable at extreme scales?

In fact, there are some TPC benchmarks for MySQL. We routinely use TPC benchmarks, or benchmarks that are similar to TPC, to measure and stress test the server. However, many of the TPC benchmarks are really focused on use cases outside of MySQL's sweet spot. For example, TPC-H is not a workload that you would ever run successfully on MySQL—at least, not on current versions of MySQL. MySQL is really more of an OLTP system. There are some third-party solutions, such as Infobright, that can work very well for analytic workloads. But those are not vanilla MySQL—they are a customized version of the server.

MySQL is suitable for use at extreme scale, but to achieve that, you have to choose a suitable application and database architecture. When MySQL is used on very large-scale applications, it simply must be sharded and scaled out across many machines. I don't think that this is an ideal architecture in general, and many small-, medium-, or even medium-to-large-scale applications can get away without doing this. But at extreme scale, there's no other viable architecture, and this is not a problem that's unique to MySQL.

The MySQL forks like MariaDB, Drizzle, and Percona Server confuse me. Why does Oracle encourage them? Are they a good thing?

You're not alone in your confusion. I do believe that these forks are a good thing, because they serve people's needs. However, I wouldn't say that Oracle encourages them. Oracle is playing an open-source game with an open-source database server, and those are the rules of the game. If you don't satisfy users, they might take the code and do what they want with it. The three major forks of MySQL represent three different sets of user and developer needs.

I would say that Drizzle and MariaDB are more focused on what the developers want to achieve with the product, and you might say that they even represent an agenda. Drizzle, for example, represents a desire to rip all of the old messy code out of the server and build a clean, elegant server from scratch. MariaDB, on the other hand, represents the original MySQL developer's vision of a database server that is very community oriented, and the desire to improve the server by leaps and bounds.

"You can consider Percona Server to be a fork that we modify only as needed to solve our customers' critical problems, which are not yet addressed in current versions of the official MySQL releases."

Percona Server is a little bit different. We try to stay as close to Oracle's version of MySQL as possible, and we're focused on solving users' needs not scratching our own itches. You can consider Percona Server to be a fork that we modify only as needed to solve our customers' critical problems, which are not yet addressed in current versions of the official MySQL releases. As a bonus, we distribute this to the public, not just our paying customers. Much of our customer base is extremely conservative and risk averse. As a result, we focus on small, high-value, low-risk improvements to the server. Many of the features and improvements we've implemented are reimplemented by Oracle as time passes, which is a nice validation of their usefulness.

In terms of real-world deployments, the official MySQL from Oracle is by far the most popular. This should not be surprising. I am biased, but from what I see, Percona Server has the overwhelming majority of the fork "market share," if you want to call it that.

When Sun acquired MySQL, a PostgreSQL developer, Greg Sabino Mullane, wrote: *"MySQL is an open-source PRODUCT. Postgres is an open-source PROJECT. Only two letters of difference between 'product' and 'project,' but a very important distinction. MySQL is run by a for-profit corporation (MySQL AB), which owns the code, employs practically all the developers, dictates what direction the software goes in, and has the right to change (indeed, has changed) the licensing terms for use of the software (and documentation). By contrast, Postgres is not owned by any one company, is controlled by the community, has no licensing issues, and has its main developers spread across a wide spectrum of companies, both public and private."* What licensing issues is Mullane referring to?

Open-source licensing is either a fascinating or annoying topic, depending on your point of view. PostgreSQL uses a very permissive license, which essentially boils down to *"do what you want, but don't sue us."* That serves the project very well. MySQL, on the other hand, uses the GPL, which is not only a pretty restrictive license but also a manifesto for a philosophical point of view. The reasons why many for-profit corporations choose to use the GPL often boil down to *"we don't want anyone else to own our intellectual property, because then our investors have nothing to sell."* As a result, companies that want to play in the open-source space, but still make a lot of money someday, often dual license the product, and that is what MySQL did during the days when they were venture-capital funded. Those days are over, and now it's time for the product itself to generate a steady stream of money for its owner, but the license remains. From a revenue-generation point of view, I think it makes perfect sense. My only wish is that the GPL were purely a license, without the preaching.

"PostgreSQL has always had a richer set of features than MySQL, but many myths about the real differences between the products persist. And frankly, there are zealots on both sides."

The just-released PostgreSQL 9.1 has an impressive list of features such as serializable snapshot isolation. Is PostgreSQL leaving MySQL behind in the technology race?

PostgreSQL has always had a richer set of features than MySQL, but many myths about the real differences between the products persist. And frankly, there are zealots on both sides. PostgreSQL is an amazing project and an amazing database server as well. But that doesn't mean that it's perfect. Recent releases have added many of the missing "killer features" that have made MySQL so popular for many years. One of the things that I think is going to make a huge difference in the upcoming release is the addition of index-only scans.

What do you think of the NoSQL movement? It is a threat to MySQL?

I think we have all learned a lot from it in the last few years, but I think that some of the strongest proponents have actually misunderstood what the real problems are. There was a lot of discussion that went something like "SQL is not scalable," when in fact that's not true at all. Current implementations of SQL databases indeed have some scalability limitations. That is an implementation problem, not a problem with the model. The baby got thrown out with the bathwater. I believe that some of the emerging products, such as Clustrix, are going to open a lot of people's eyes to what is possible with relational technology. That said, many of the approaches taken by nonrelational databases are worth careful consideration, too. In the end, though, I don't think that we need 50 or 100 alternative ways to access a database. We might need more than one, but I would like some of the non-SQL databases to standardize a bit.

I'm an Oracle professional who needs to learn MySQL fast. Where should I start? What books should I read? Where can I get help?

I would start by making friends. See if there are any user groups or meet-ups near you. Who you know is much more important than what you know. In particular, the annual MySQL users conference has always been the watering hole where the MySQL community, partners, and commercial ecosystem have come together. O'Reilly used to run the conference, but this year Percona is doing it instead. You can find out more at **http://www.percona.com/live/**.

In terms of books, I would recommend two books to an experienced Oracle professional. The first is simply titled *MySQL*, by Paul Dubois. This is probably the most comprehensive and accessible overall reference to MySQL. The second is *High Performance MySQL*, which I co-authored with Peter Zaitsev and Vadim Tkachenko, Percona's founders. There's no other book like this for learning how to really make MySQL work well for you. I'm finishing up the third edition right now. Finally, I would strongly recommend the MySQL manual and documentation. The documentation team at MySQL does an amazing job.

In addition to Oracle, several other service providers are good resources for consulting, support contracts, development, remote database administration, and the like. Percona is one, of course. There are also Pythian, SkySQL, and a variety of smaller providers.

NoCOUG membership levels have been stagnant for years. Are we a dinosaur in a connected world where all information is at available our fingertips? What does a user group like NoCOUG need to do in order to stay relevant?

I think that the unique value that a user group such as yours can offer is to bring people together face-to-face. My company is entirely distributed, with more than 60 people working from their homes worldwide. This has its benefits, but there is no substitute for meeting each other and working together. Direct personal interactions still matter, and technology cannot replace them. As an example, although I'm delighted that you interviewed me for this magazine, I would be even more pleased to speak to all of you in person someday. ▲

Interview conducted by Iggy Fernandez

"I think that the unique value that a user group such as yours can offer is to bring people together face-to-face. Direct personal interactions still matter, and technology cannot replace them."

Oracle Developer Training with Dan Hotka

Instructor-led training and hands-on lab exercises with Oracle ACE Director Dan Hotka

Oracle SQL Tuning for Developers: March 26–27

This course is designed for Oracle developers who wish to acquire SQL tuning skills. The course starts with an overview of Oracle database architecture and continues with an in-depth study of the EXPLAIN PLAN (how Oracle retrieves data and in what order). Students will work with a variety of SQL statements, review the EXPLAIN PLAN, and make changes to make these SQL statements perform better.

The course includes index design, using hints, and coding style to control plans. The course also takes a very close look at indexes: how Oracle selects them, why they are sometimes not used, and how to tell if they are being used.

This course emphasizes coding style. The instructor finds that tuning in this fashion maintains the performance of SQL when upgrading to new releases.

For more information and to register,
please visit **http://www.nocoug.org**.

Oracle Advanced PL/SQL Tips and Techniques: March 28–30

Oracle developers will learn a variety of advanced PL/SQL techniques including advanced cursor management/cursor sharing, code sharing, code debugging, transaction management, and more. This course also includes a variety of Oracle performance tuning techniques and a review of new PL/SQL features in Oracle Database 10*g* and Oracle Database 11*g*.

Topics include Oracle architecture, PL/SQL syntax and structure, user-defined and Oracle-defined packages, compilation options, cursor sharing and cursor variables, collections, dynamic SQL, autonomous transactions, large objects, debugging, and profiling.

Lectures and topics are enhanced with live illustrations and hands-on exercises.

For more information and to register,
please visit **http://www.nocoug.org**.

Managing Oracle Fusion Applications

A Book Review by Brian Hitchcock

Details

Authors: Richard Bingham

ISBN: 978-0-07-175033-2

Pages: 350

Year of Publication: 2011

Edition: 1

List Price: $50

Publisher: Oracle Press

Overall Review: Worthwhile overview of what Fusion Applications is all about and why it represents such a big change.

Target Audience: Those interested in a high-level look at Fusion Applications.

Would you recommend this book to others: Yes.

Who will get the most from this book? Anyone who needs to understand what it will mean to transition to Fusion Applications.

Is this book platform specific: No.

Why did I obtain this book? See Overall Review below.

Overall Review

Fusion Applications is a bigger change than you realize. It's a bigger change than I realized.

If I had seen this book on the shelf, I would have flipped through it and dismissed it. It looks like a lot of descriptions of acronyms made up of other acronyms—recursive acronyms. But I did read it, and I'm glad I did. The move to Fusion Applications requires moving to open standards, and that requires learning about all those acronyms.

I need to be prepared to support Fusion Applications. I found many resources, and among them were three books from Oracle Press. This is one of them, along with *WebLogic Server 11g Administration Handbook* and *Oracle Fusion Middleware 11g Architecture and Management*. Since I have reviewed other Oracle Press books, they sent me copies of each to read and review.

Reading this book really helped me understand just how big a change the move to Fusion Apps will be. The description of the history of Oracle Applications, both those developed by Oracle and those acquired by Oracle, is the best I've ever seen. It provides the context I've been looking for to understand why this is happening and why it has to be such a big change.

I understand that many Oracle professionals who currently support Oracle Enterprise Suite and other products in what is now called Oracle Applications Unlimited will be looking for resources that show them the technical details of implementing and supporting Fusion Applications. This book will not appear to be relevant, since it doesn't attempt to address those issues. However, Fusion Applications is not just Applications Unlimited with some new stuff bolted on the side. It is a complete overhaul and integration of all the various applications products with some new stuff mixed in. Wrapping your head around all this requires a high-level review, and that is what this book provides.

Introduction

Unlike most books I've reviewed, this one has some of the best information in the introduction. Don't skip it. In the first few pages, we find this statement: "The structure of this book is intended to help you first build a basic understanding of Fusion Applications . . . ." Indeed—and you need this high-level overview to get that basic understanding. From "A Brief History" on, this introduction helped me the most. It gave me the context I was looking for, explaining how the evolution of applications, access, and user experience all led to Fusion Applications.

Chapter 1—A Fusion Applications Product Overview

This chapter covers the size and scope of Fusion Apps. The product families are explained (Financial Management, Human Capital Management, etc.), followed by a discussion of the functional architecture.

Chapter 2—A Fusion Applications Technical Overview

The user interface is explained, as are dashboards, worklists, and the incorporation of social networking. The use of model-view-controller architecture is a good example of abstract concepts that I would not have thought I needed to know about. I looked it up online and read about it. This shows just how much different Fusion Apps is from the products we use now. Further examples include orchestration—which relates to SOA—and BPEL, all of which I would have dismissed as way too academic for me. Learning about all of this helps. I will be installing "artifacts" and patching "composites"—concepts that I never had to deal with in Enterprise Business Suite. Also new to me are web services. Fusion Apps has more than 1000 web services supporting all the functional objects and tasks.

The tech stack is discussed, and it includes many more acronyms. Also significant is the explanation of how Fusion Applications is built on top of Fusion Middleware. This had confused me for a long time, but it makes sense now. Again, this is important information to understand for anyone who has to figure out Fusion Apps. It is here that WebLogic Server makes its appearance. In many ways, Fusion Apps is WebLogic Server. I had been trying to understand why I needed to know about WebLogic Server, and this also makes sense now.

The "Processing Walkthrough" section made me appreciate how integrated Fusion Apps is. There are a lot of new concepts to become familiar with. How to extend and customize Fusion Apps is also discussed. This will be a big part of working with Fusion Apps. With so many applications integrated, the possibilities for extension are much greater, as are the challenges of supporting all of this.

Chapter 3—Successful Enterprise Application Management

Here the author describes the Enterprise Application Management role and why it is more important than ever due to the increased complexity of Fusion Applications. The simple definition of this role is that it has responsibility for the enterprise application and making sure it is available to users. This definition is then broken down into five areas, and individual chapters describe the relevant features of Fusion Applications that support each of these areas. These five areas are addressed in Chapters 6 through 10.

While it is implicit in this discussion of complexity and roles, it is also plainly stated: you will need dedicated resources for Fusion Applications. You may currently have resources that support various pieces of Applications Unlimited, but that won't be enough to take on Fusion Applications as well.

Each of the five areas of Enterprise Application Management is described. *Reliability* involves keeping users happy, completing business processes on time, resource monitoring, preventative support, and resolving issues as they arise. *Availability* means no downtime and providing multiple environments for tasks such as testing and development. *Performance* covers monitoring for failures, setting thresholds to detect low performance due to bottlenecks and high performance indicating underused resources, business process performances, core services performance, and hardware and operating system performance. *Optimization* involves finding ways to improve all of the areas already described. *Governance* includes security, data quality, and managing change.

You can dismiss this discussion, but you are already doing some form of all of these tasks, however informally, in your current environments. Fusion Applications will support (and, I think, *require*) all of these areas more formally, by design, with utilities built into the product.

This chapter concludes with a discussion of what it means to succeed at managing the enterprise application. I like the following statement: "[T]he modern enterprise application is no longer a collection of linear software programs overlain by a static set of front-end forms."

And there it is. You won't be able to manage Fusion Applications from the command line alone. The complexity requires more sophisticated tools to manage the environment. And yes, this means lots more GUIs. You don't have to like them, but you can't avoid them anymore.

Chapter 4—The Fusion Applications Lifecycle

Here we learn about the lifecycle models used by most IT departments. The alphabet soup is especially deep with RAD, ALM, ILM (no, not the special effects place), ITSM, ITIL, and finally COBIT. We are told that none of these is really applicable for Fusion Applications, and then we are presented with the Fusion Applications Management Lifecycle Model. This model is used to show how the five areas of Enterprise Application Management discussed earlier fit into an overall lifecycle model for Fusion Applications. This leads to chapters 6 through 10, each of which describes the specific features of Fusion Apps that support each area.

> ***"Fusion Applications is a big change. This means that people with experience in the current Oracle applications offerings will need to embrace change. Most people will be new to Fusion Applications, so this is a great time to get started."***

Chapter 5—Fusion Applications Management Toolbox

This chapter is less than six pages long. (When does a chapter become so small that it's no longer a chapter?) It's small because it's the introduction to the next five chapters. Each chapter discusses the tools available within Fusion Applications to address the specific application management area. The goal of this is to provide some structure to a large set of tools and utilities to make it easier to know what to use in which situation.

Chapter 6—A Reliability Management Toolbox

To help manage the reliability of Fusion Applications, you need to look at the Diagnostic Test Framework (DTF), which allows you to run diagnostic tests. These tests can be used for troubleshooting or for validating the health of application components. Functional Setup Manager is used to capture configuration information. Incidents are created when something goes wrong in the applications, the Fusion Middleware (which is the foundation that Fusion Applications is built upon), or the database. The content and management of incidents is discussed.

A big part of reliability management is seen in the extensions made to Enterprise Manager. We are told that EM is "the single most important tool for monitoring and managing the complete Fusion Applications technology stack." Time to get comfortable with EM. I have supported Oracle Applications for some time and have never needed EM before. This is symbolic of the big change that Fusion Applications brings. Now I need to use EM all the time.

Chapter 7—The Availability Management Toolbox

The important point here is that while all the application components may be running, not all the business processes may be. Business Service Availability is discussed, along with Fusion Applications Control, which is an extension of Enterprise Manager and has many dashboards to monitor the availability of components and the SOA-based composite applications as well.

An interesting point is made here, namely that each product in Fusion Applications comprises one or more Java 2 Enterprise Edition (J2EE) applications. This means that each product handles all aspects of rendering forms and pages. There is no other server that does this for the product applications. Once again, Fusion Applications is a big change from what has come before.

Fusion Applications Control is also used to monitor the availability of Fusion Middleware. As another example (are more examples needed?) of just how big a change Fusion Apps is, consider the MBeans Browser that is part of Fusion Apps Control. MBeans are the Java Management Extensions (JMX) that are used to contain the configuration of servers and data sources and other components. The MBeans Browser is used to examine and modify the MBeans. WebLogic Server (WLS) is the central component of Fusion Apps and is monitored in Fusion Apps Control as well. Database availability is monitored through Enterprise Manager Database Control or Grid Control.

Chapter 8—A Performance Management Toolbox

Fusion Apps Control is used to monitor the performance of products, SOA, and business process execution. Enterprise Scheduling Services (ESS), which is similar to Enterprise Business Suite Concurrent Manager processing, is also monitored for performance from within Fusion Apps Control. Yet another significant change: in Fusion Apps, each product family has its own ESS, instead of one Concurrent Manager processing environment. This means that you need to monitor, through Fusion Apps Control, the ESS performance for each family separately. Fusion Middleware, WebLogic Server, and the database all have separate areas within Fusion Apps Control for performance monitoring.

Chapter 9—An Optimization Management Toolbox

This chapter covers which Fusion Apps tools are used to get more done more efficiently and to find new ways to do things. A big part of this is documenting the configuration of the many components and services. The Oracle Enterprise Repository stores all this information and is useful when new applications need to be built on existing application services. The Diagnostic Test Framework (DTF) is used to configure tests that can validate the configuration and the data before an application is put into use. Data captured by DTF helps identify processes that should be optimized.

In Fusion Apps, most application components run largely independently of each other. This is why most application components are now referred to as a service that is called by other services to build an application. This means it can be more complex to make changes in one service and document the impact on all the other affected services.

Chapter 10—A Governance Management Toolbox

Here we see the features used to oversee security, configuration, data management, and patching to achieve various governance and regulatory requirements. SOA governance is a new concept, needed to ensure standards within SOA applications. Security management is provided by Oracle Access Manager (OAM), Oracle Identity Manager (OIM), Oracle Authorization Policy Manager (OAPM), and Oracle Web Services Manager (OWSM). There are separate GUIs for these products. It is interesting to note that Fusion Applications does not provide comprehensive change management functionality at this time. Given the complexity of Fusion Applications, this is certainly needed.

Chapter 11—Getting and Staying Healthy

This chapter covers a lot of topics. The following statement needs to be reviewed carefully: "Application support requires a specific set of abilities that is often overlooked . . . ." This is all too true. Fusion Applications is more complex and will require more resources dedicated to application support. Other areas that are covered include capacity planning; patch planning; data quality, including data fixes; system testing; and customization management. The flexibility inherent in a services-based enterprise application system also means that extensions and customization can be more extensive than in the past. This also means that more effort will be required to manage these customizations.

Chapter 12—Planning for the Future

Predicting what will come next is challenging. This chapter provides some thoughts on how Fusion Applications will change as it matures. For example, it discusses what business process might come along next, as well as possible changes to configuration management and the patching process.

Conclusion

Fusion Applications is a big change. This means that people with experience in the current Oracle applications offerings will need to embrace change. Most people will be new to Fusion Applications, so this is a great time to get started. This book is a good way to get exposed to just how much is new and different. You won't learn how to install Fusion Applications, but, whether you accept it or not, you really do need to become comfortable with all the new ideas, acronyms, and processes that make up Fusion Applications. This book is a very high-level view of what will be involved in managing Fusion Applications. I think everyone involved would benefit from reading this book. ▲

Brian Hitchcock worked for Sun Microsystems for 15 years supporting Oracle databases and Oracle Applications. Since Oracle acquired Sun he has been with Oracle supporting the On Demand refresh group and most recently the Federal On Demand DBA group. All of his book reviews, presentations and his contact information are available at **http://www.brianhitchcock.net**.

The statements and opinions expressed here are the author's and do not necessarily represent those of Oracle Corporation.

First Look at Oracle Database Appliance (ODA)

by Alex Gorbachev

This article is the first of a series by Pythian experts that will regularly be published as the "Performance Corner" column in the NoCOUG Journal.

The innovation of Oracle Database Appliance (ODA) is in its architectural simplicity—it's as simple as you can imagine a two-node Oracle Real Application Cluster (RAC) could be. Thanks to its simplicity, Oracle has designed the database appliance to be an inexpensive and reliable all-in-one database platform. Oracle Database Appliance is a single 4U device that contains all the required infrastructure for a RAC cluster, including two powerful database servers, shared storage, and cluster interconnect. While there is much to say about ODA, in this article we will focus on the performance characteristics of the appliance and its scalability.

To understand what a single database appliance is capable of, you need to understand its architecture. Each database server inside an appliance is represented by an independent mainboard with CPUs, memory, and interfaces either built into the mainboard itself or as PCI Express expansion cards. All this (plus cooling and two hard disks used for the operating system) is packaged in a small chassis called a "System Controller" by ODA creators or a "Server Node" for end users. You can think of them as server blades, because they are powered by the central power supplies and simply plugged into the appliance, connecting via a backplane.

Until Intel Sandy Bridge CPUs become available in enterprise servers, the fastest x86 cores are in Intel Xeon X5600 series processors. Note that I'm talking about core performance; this is what you should be looking at, since Oracle Database Enterprise Edition is licensed per core and not per socket. The most powerful processor in the 95-watt TDP range is X5675, with six cores running at 3.06 GHz. The next CPU in this series is X5680, which requires 130 watts, so it's more difficult to cool.

With a total of 24 powerful cores, ODA has plenty of processing capacity for the majority of database applications I see customers running these days, even if those applications are not very well written (regardless of the platform they run on) and tend not to burn CPU cycles very effectively. These are, unfortunately, the most common applications out there. With 96 GB of RAM in each server, it's possible to run a database with quite large SGA, allocating most of it for the buffer cache to minimize slow disk I/O.

Each server node of an ODA resembles a server node of the X2-2 Exadata Database Machine, except for the peripheral devices and the ability to expand server memory of an Exadata server node to 144 GB. But here all similarities to Exadata end.

Oracle Database Appliance doesn't use InfiniBand to perform disk I/O or as a cluster interconnect. Instead, it relies on simple SCSI and Ethernet technologies. There's nothing bleeding edge here, but it's a proven rock solid approach. Does the absence of InfiniBand sacrifice the performance of ODA? Hardly.

For the cluster interconnect, ODA uses two 1 Gbit Fiber Ethernet. There is no network switch in between, and two fiber ports on one server node are connected directly to two ports on another server node. In my experience with Oracle RAC deployments, a two-node RAC cluster generally does not saturate a single gigabit link processing OLTP workloads. If the workload is not RAC optimized and causes a lot of Cache Fusion traffic, a bottleneck is usually hit somewhere else before interconnect is saturated. Tracking "gc %" wait events on ODA and Exadata, I see they have comparable timings. InfiniBand really shines at higher scale, where saturated Ethernet latencies usually climb up rapidly. However, this usually goes beyond a two-node cluster.

When it comes to data warehousing workloads running parallel operations with slaves on both nodes of the cluster, saturating both gigabit links is possible. However, I normally don't see much benefit to spreading parallel operations across just two nodes—there is more overhead of managing inter-node parallel processing than gains, especially, when the bottleneck is not on the server but in the I/O subsystem. This is more likely the case for ODA, as we will see later. If you are processing parallel queries on the appliance, consider keeping each query on one node.

"The innovation of Oracle Database Appliance (ODA) is in its architectural simplicity—it's as simple as you can imagine a two-node Oracle Real Application Cluster (RAC) could be."

Before moving to I/O performance, let me summarize the simple parts: In ODA, CPU and memory performance is just what you would expect from two modern two-socket x86 servers. Cluster interconnect performance is as good as you would build yourself using 1 Gbit Ethernet. The network latency of the ODA cluster interconnect is actually slightly lower, because there is no network switch in the middle.

The storage subsystem of an Oracle Database Appliance resembles that of a standalone server with internal disks. Each server node has two Serial Attached SCSI (SAS) Host Bus Adapters (HBAs). These HBAs are connected to 24 disks via 2 SAS expanders. The difference is that each disk has two SAS ports, so each server node is connected to each disk via one of those ports. This is my favorite architectural feature of ODA—it solves the shared storage problem in such an elegantly simple way, as shown in Figure 1.

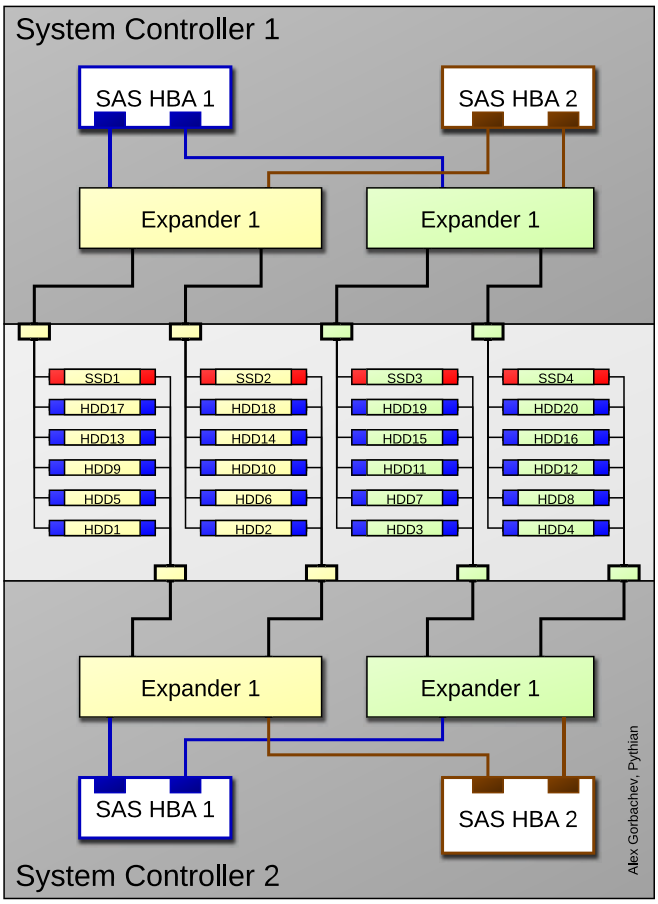


Figure 1: Storage Subsystem of ODA

Since each of the two SAS HBAs has two paths to each disk, one via each expander, multipathing is configured on the servers (don't worry—it's done by the installer, so you don't even need to know what it is).

As you can see, ODA doesn't really have a storage tier per-se like Exadata Storage Cells. That also means that there can't be any Exadata-like storage offload features, simply because there is no storage tier to offload to. One feature that first appeared in Exadata that could architecturally run on ODA is Hybrid Columnar Compression (HCC), and we already know that Oracle has opened up this feature to be used outside of Exadata if customers are using ZFS Appliance. I believe Oracle Database Appliance could easily support HCC if Oracle permits its use on that platform.

> ***"SSD vendors have mostly solved the write problem of flash these days by over-provisioning the space inside SSDs and doing some smart manipulation with writes by rearranging pages and ensuring that they are written sequentially in the same blocks."***

So, what are those 24 disks I mentioned earlier? Twenty disks are traditional spinning disks. They are 600 GB high-end 15K RPM SAS disks—the same disks as high-performance disks in Exadata. Four other disks are SSD from STEC (MLC version with SAS interface: **http://www.stec-inc.com/product/zeusiops.php**). SSD disks are dedicated to only one purpose in the database appliance, online redo logs.

When we initially evaluated the database appliance during the beta program, my colleagues and I were somewhat surprised by that decision, since flash storage is generally considered not to be very effective under heavy write workloads. The problem with flash writes is that in order to write to flash, they need to be deleted. While a small write request can be done as one page (typically 4K), the flash delete operation is done on a whole block (typically 512K) and it's very slow and wears off the media quickly. That said, SSD vendors have mostly solved the write problem of flash these days by over-provisioning the space inside SSDs and doing some smart manipulation with writes by rearranging pages and ensuring that they are written sequentially in the same blocks.

But let's get back to redo logs and ask an important question: why do we need SSDs at all? If you remember the basic architecture of an Oracle Database instance, you'll recall that each time a session commits a transaction, it requests the LGWR process to flush the redo log buffer to disk up to the transaction's commit marker. In order to guarantee data integrity, the Oracle process does not return control to the client until LGWR completes the write to disk. In a write-intensive application, multiple processes can be waiting for LGWR to flush redo blocks, and every millisecond of LGWR spent writing can contribute to a response time increase of multiple user sessions. Slow redo log write situations frequently manifest themselves as spikes of excessive "log file sync" waits.

One solution to ensure that critical writes are done in a very short amount of time is to use battery-backed cache, and that's what many disk controllers and storage arrays do these days: they complete a write to cache and report disk write as completed, all while still propagating the write down to the physical disk. In the case of Oracle Database Appliance, each server node has its own disk controller, but both are writing to the shared physical disks. Because we are running Oracle RAC, we

must ensure that if a block write has completed on one node, it can be immediately read on another node. Thus, it's impossible to use write cache (or read cache for that matter) that is local to one node of a cluster. This is why ODA is using LSI SAS9211-8i controllers without cache. As a result, the shared disk itself must be able to perform writes very fast consistently, regardless of how saturated (until a certain limit, of course) the I/O subsystem is.

Until solid state memory became affordable, in addition to write cache, storage administrators and DBAs used to craft storage carefully and allocate redo logs in a very specific way to minimize heads moves as well as avoid the impact of redo log archival to the performance of LGWR. With modern SSDs, this problem can be solved much more easily, and that's what Oracle did in ODA.

When studying technical overview of Oracle Database Appliance, I've read Oracle claims that no redo log write will take more than 0.5 ms. According to my ORION benchmarks doing sequential 32K writes (Figure 2), I have achieved around 6,000 writes to SSD disks accounting for ASM high redundancy (i.e., one redo write is written on three disks) with eight to ten parallel threads. This means four to five RAC databases with each instance aggressively pounding redo logs. In this situation, the average write time is still around 0.5 ms, and each instance is conducting 1,200–1,500 writes. Note that because of the piggyback effect of multiple commits, the effective achievable transaction rate is actually higher. This figure is rather theoretical, as I would not expect to find a reasonable deployment with five databases on a two-node RAC clustering doing so many commits, but it gives you an idea of the potential ceiling for the redo log scalability of ODA.

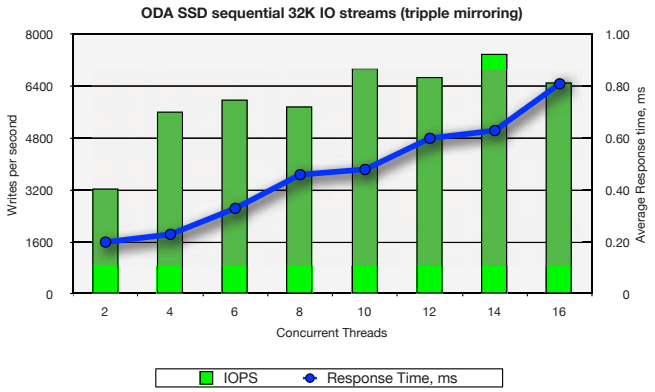


Figure 2: ORION Benchmark

Based on this ORION benchmark, a single database with both instances committing aggressively, you can achieve about 3,200 sequential (each thread writing in its own sequence) 32K writes—1,600 writes per instance. To simulate a similar situation with a real database, I crafted a small benchmark saturating LGWR with very small transactions as fast as possible. I scheduled multiple jobs, each inserting a single row in its own table, so there is no possible contention between jobs, and committing after each insert. With such a setup, I could reach a similar number of 3,200 writes per second on both nodes with average write time around 0.2 ms as well. This is 1,600 writes for a single LGWR process on each node. If you do the math, these 1,600 writes would account for only about one-third of a second (320 ms), where the rest of the time LGWR spends consuming CPU. No matter how hard I pushed, LGWR consumed up to 70% of a single CPU core while spending the rest of the time waiting for I/O completion.

Depending on the traffic patterns, this might translate easily into 2,000–3,000 commits per second, per node and even higher. Based on my experience, this covers the vast majority of real-life database workloads, but I expected to be able to squeeze a bit more from ODA. The best part of this setup is that the write time is very consistent and does indeed stay below 0.5 ms even when the archivers kick in to archive an active online redo log after a redo log file switch.

> ***"With a very attractive price, unique licensing flexibility, and benefits of an integrated appliance, Oracle Database Appliance stands out from the competition."***

I discovered that write time can go into the several millisecond range if I push LGWR really hard to perform writes in the hundreds of kilobytes size. You would normally see this during massive data load batches done without taking advantage of NOLOGGING operations. As you can see, ODA is reaching its limits again when we come to a heavy data warehousing workload that is more appropriate for the next class of Oracle-engineered systems—Oracle Exadata.

Before we move on, I should note that Oracle uses SSD strictly for redo logs. Four SSD disks of 73 GB each give you up to 73 GB of usable space in a single high-redundancy ASM disk group (accounting for potential failure of a single disk). With two database instances and exceptional redo performance, you would generally not use more than 5–10 GB for online redo logs. This leaves you with 60 GB of high-performance triple-mirrored SSD storage on the REDO disk group.

If you choose not to follow Oracle's instructions and use that space for something else, you can. However, think about potential benefits and whether it's worth turning your highly repeatable deployment into something unique. With 192 GB of RAM on two severs, you can easily allocate around 160 GB for buffer cache, automatically eliminating most of the single block I/O to "hot" areas. Placing datafiles on SSD would only make sense if you have to do some specific direct path reads to SSD (even then its usually sequential, and spinning disks deliver just fine) or if you can't oversize buffer cache high enough.

Using the 60 GB of SSD storage for high-performance temporary tablespace is an option. but you might be much better off allocating more RAM for PGA as opposed to SGA. Placing write-intensive undo tablespaces is rarely a justified need because dirty data blocks are flushed to disk by DBWR processes in the background and do not directly affect user response time. There might be rare cases of excessive delayed block clean-out activity that can potentially cause a high number of physical reads, but these situations are pretty rare. The use of SSD for Database Smart Flash Cache will likely not bring any significant benefit either, because you already have 192 GB of

real RAM, which is currently several orders of magnitude faster than flash memory and three times the size of 60 GB.

Again, there might be the few very specific scenarios when an additional 60 GB of SSD space is useful, but think twice (no, think five times) before choosing this path, as it will cause your database appliance configuration to become unique and potentially bring unique problems.

Enough talking about SSD; let's look at the performance of good old brown spinning disks—HDDs. I have covered some of it on the Pythian Blog in September 2011 (**http://bit.ly/ODAstorage1**). Here, I would like to include the following diagram (Figure 3).

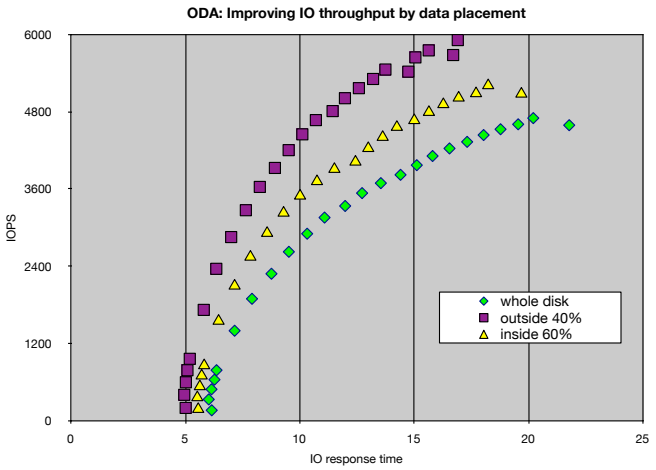


Figure 3: Improving IO throughput by data placement

The diagram visualizes the measurements of small (8K) random I/Os (read-only here) using the Oracle ORION tool. Orion scales I/O workload by increasing the number of concurrent I/O requests progressively while collecting average response time and I/Os per second (IOPS) metrics for each data point. To produce this chart, I ran three independent tests: (1) using whole physical disks, (2) using outside 40% capacity of the disks, and (3) using inside 60% capacity of the disks. I then built the data points with average response time on axis X and IOPS throughput on axis Y.

As you can see, using outside 40% of disk capacity leads to about a 50% increase in IOPS capacity while average response time stays constant. For example, if average I/O response time of 15 ms is acceptable, you can reach almost 6,000-read IOPS using 40% of disk capacity as opposed to almost 4,000 IOPS using the whole disks. You can turn it another way around: if you keep the same volume of IOPS, the average response time drops about 30%.

This should help you plan your ODA deployment balancing space and performance capacity. For writes scalability versus reads, refer to the blog post I mentioned above.

Let's look at the sequential I/O scans. I could reach 2 GB per second using a single node with 1 MB reads (Figure 4). Depending on the patterns of parallel scans, I was able to get up to 2.4 GBPS using ORION on a single node, but Oracle specs of ODA claim up to a 4 GB scan rate. I haven't tested loading it from both nodes yet, but keep an eye on my blog and I will post the results as soon as I have a chance to take the right measurements.

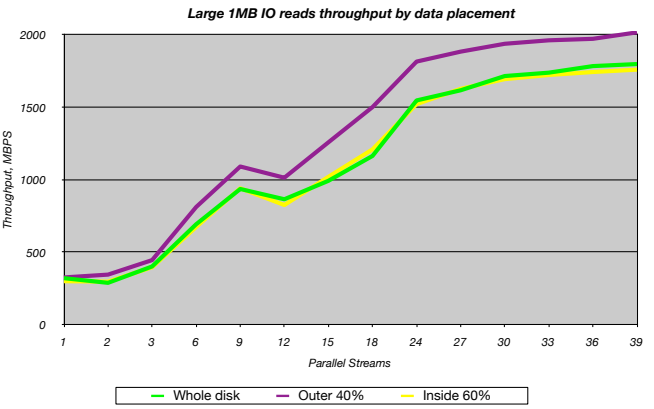


Figure 4: Large 1 MB reads throughput

I also tested RMAN performance backing up a small 42 GB database. First I backed up to the FRA on the database appliance itself, which caused disk heads to move aggressively between outside and inside parts of the disks all the time. By running multiple tests, I identified that the optimal number of RMAN channels is eight and backing up a 42 GB database takes one minute and 45 seconds, delivering an effective backup rate of about 400 MBPS. RMAN channels were spending a significant amount of time managing start and completion of backup pieces, so for a larger database, the effective backup rate should be higher. I didn't use any compression in this case to ensure I don't introduce CPU bottlenecks.

I didn't have any external storage attached to the appliance, so I ran BACKUP VALIDATE with 8 parallel channels to simulate potential backup to external media; BACKUP VALIDATE only reads and validates data blocks and then throws them away instead of writing to the backup destination. Thus, assuming your backup destination is not a bottleneck, BACKUP VALIDATE is a good way to measure the potential backup rate to an external backup destination. The same 42 GB database was backed up in only 45 seconds, which gives about a 1 GBPS backup rate. If you connect external storage using a single 10 Gbit Ethernet link, this would be close to its wire speed, so I didn't push any harder. But I think I could have easily doubled it by increasing the number of channels to about 20 (20 disk heads wouldn't need to move much, unlike in the scenario of reading from and writing to the same physical disks). Remember that this is an ideal scenario when there is no other concurrent database workload, which is normally not the case in real life.

With a very attractive price, unique licensing flexibility, and benefits of an integrated appliance, Oracle Database Appliance stands out from the competition. It also has some good horsepower packed inside, but you still need to know the limits of the platform, so I hope this article gives you some guidance in that area. ▲

Pythian CTO Alex Gorbachev is a respected figure in the Oracle world and a sought-after leader and speaker at Oracle conferences around the globe. He also regularly publishes articles on the Pythian blog. Alex is a member of the Oak Table Network and holds an Oracle ACE Director title from Oracle Corporation. He is the founder of the Battle Against Any Guess (BAAG) movement promoting scientific troubleshooting techniques.

Cost-Effective Oracle

by Jay Stanley

A fuller version of this article can be found on the Database Specialists website. (**http://www.dbspecialists.com/files/presentations/Cost-EffectiveOracle.pdf**)

There are quite a few companies, given the current global financial situation, that are investigating ways of lowering the cost of running their business using an Oracle database. This paper does not involve Oracle Applications or Oracle Financials; they are strictly tips on how to get the most out of the Oracle RDBMS itself.

Tip 1: Understand the amount of money spent on the software license v/s the hardware used, including both up-front and recurring costs over the life of the database

Typically, the cost of an Oracle license is far, far greater than the cost of hardware, hardware support, the operating system, and operating system support, especially over the life of the installation.

So, it makes sense to spend more on hardware and try to limit the cost of the Oracle license itself. Do an analysis that includes all of the factors involved, and include any others that may be particular to your case.

Generally, the separate items you pay for include:

- The hardware that will run the database, including any networking required
- The electricity needed to run that hardware
- The space that that hardware takes up (especially if in a co-located rented space)
- Hardware support from the manufacturer (e.g., Dell, HP, IBM, Oracle/Sun)
- The operating system itself (Red Hat Linux, AIX, Oracle Enterprise Linux, HP/UX, Windows)
- Operating system support (this includes security updates and bug fixes)
- Any external storage plus replacement disks/associated hardware as they fail
- Support for external storage
- Electricity and rack space for external storage
- The Oracle database software licenses, additional option packs (e.g., RAC, tuning pack), and Oracle Support for help and patches as required,,
- The administrators (system administrators, network administrators, programmers, database administrators) to support the system

Look at these costs over the entire time span of your target product.

When I have done these exercises in the past, excluding human resources, frequently the cost paid for the RDBMS license to Oracle is more than 70–80% of the total.

Tip 2: Buy the cheapest version of Oracle that will do the job

Oracle has several versions of its venerable RDBMS; they range in price from free to quite expensive. Carefully analyze which version makes the most sense for you; in fact, if you haven't designed your application yet, you may be able to target your product to one of the cheaper versions.

At the time of this writing, the different versions are as follows: (**http://www.oracle.com/us/products/database/product-editions-066501.html?ssSourceSiteId=otnen**)

Edition	CPU Limit	RAM Limit	Database Size	Notes
Express Edition	1	1 GB	4 GB	Free; only on Windows, Linux; quite limited
Standard Edition One (SE1)	2 sockets	OS max	No limit	Cannot use advanced features
Standard Edition (SE)	4 sockets	OS max	No limit	Cannot use advanced features
Enterprise Edition (EE)	No limit	No limit	No limit	Extra cost for each advanced option

Note that "Sockets" is the maximum number of processor sockets that are available on the server, regardless of whether

"Typically, the cost of an Oracle license is far, far greater than the cost of hardware. So, it makes sense to spend more on hardware and try to limit the cost of the Oracle license itself. Buy the cheapest version of Oracle that will do the job."

the sockets are filled. It doesn't make financial sense, in terms of the Oracle license, to buy a server that has empty sockets, unless there's a special agreement with Oracle.

The nice thing about these different versions is that it's usually quite easy to upgrade. On the other hand, downgrading, say from Enterprise Edition to Standard Edition, usually involves quite a lot of downtime and frequently a full export/import of all of the data, with associated downtime.

Unless your application is very, very small, Express Edition may not give you what you really need; be wary.

Standard Edition One will work extremely well in many applications today. It offers configurations up to 2 sockets, which, with processors today, can often be more than sufficient. You can buy SE1, and can have an unlimited amount of RAM and unlimited amount of storage and whatever quality of storage with this license. You can even use ASM (Oracle Automatic Storage Management) with SE1, and this combination can be very cost effective.

Standard Edition also works very, very well, especially if you don't need any of the advanced options and need more CPUs than SE1. Note, however, that parallel operations are not available; that's only enabled in EE. Standard Edition is usually one-third of the cost of Enterprise Edition.

Enterprise Edition has a lot going for it, especially when purchased with some of the more-used extra options; however, it's usually about three times (300%) the cost of Standard Edition. Of course, this ends up being not only a purchase cost but a recurring support cost as well for the life of the database.

There are quite a few advanced features that can be enabled only for Enterprise Edition, which may be absolutely required by some applications. For example, if you have a product that is under constant development with fast release cycles and really requires constant tuning, then the Enterprise Edition with the SQL Tuning Pack extra-cost option may save you time in the long run; similarly, for compliance reasons you may need the Advanced Security option. The Partitioning option is also quite popular.

Enterprise Edition has quite a few features that make disaster recovery better as well, such as tablespace point-in-time recovery and the ability to have a physical or logical standby slaved in real time with the primary database. There are many other advanced features as well—too many to include in this presentation.

If you are not sure, ask your Database Administrator for help.

Tip 3: Limit the number of CPU sockets in the database server to save licensing costs; investigate the named-user-plus license

There are several plans offered by Oracle; named-user-plus and processor-based are two of them. Take a very close look at their current licensing agreements and discounts available at **http://www.oracle.com/us/corporate/pricing/specialty-topics/ index.html**.

If your application faces the Internet with an unlimited number of end users, you'll need to purchase the CPU license. The CPU license is affected by the number of sockets or CPUs that you have. This means that you should get the fastest CPU that you can but also that you limit the number of sockets; a two-socket machine will cost twice as much (200%) for the Oracle RDBMS license as a single socket in most cases.

If you are using a processor-based plan (for unlimited users), see if you can get by with a less powerful processor. This can save quite a bit of money.

Some people consider RAC more reliable, but in my experience it most certainly is not; there is still a single point of failure—the storage subsystem—and the complexity of RAC makes it far less bugfree.

Tip 4: Don't use RAC unless you need it

Remember, when you use RAC, you have to buy a separate license for each box, and since this is very likely the biggest-ticket item, you will pay much more when you use RAC. In addition, RAC is an option that is purchased in addition to the Enterprise Edition license.

RAC is useful for when you cannot buy a machine with enough CPUs or memory at any cost for your target application, or that cost is extremely prohibitive; it allows you to scale your application horizontally. The main successful applications I've seen are generally data-warehouse-type applications with extremely large data volumes.

In my experience:

- Buying RAC for two nodes of one processor apiece is usually far more expensive than buying a non-RAC license for one two-processor box.
- Buying a machine with eight processors will cost less in overall costs than buying four servers with two processors apiece.
- In fact, buying RAC for any number of nodes is more expensive than buying one box that has the same amount of processing power/memory.
- Your database will run faster on a single box with, say, four processors and 64 G of RAM than it will on four boxes with one processor apiece and 16 G of RAM apiece, and you'll save on licensing costs as well.

Oracle RAC uses the memory in all of the instance nodes in order to cache database blocks for queries (this is called "cache fusion" by Oracle). When one instance requires a read-consistent block, it requests it from other nodes to see if it is already in memory. If the block is in memory on another node, the node that has the block will move it over the interconnect to the instance that requested it. The speed of the network interconnect between machines (used to move blocks between instances in a RAC configuration) is an order of magnitude or more slower than the speed of inter-processor communication on a single machine with multiple processors. In terms of obtaining a read-consistent block, a machine with all CPUs together will be able to obtain the block an order of magnitude

faster than obtaining that block over an interconnect.

Current CPUs are far, far faster than they have been historically; most database servers today are oversized in terms of processing power. If using the "unlimited users/CPU" license structure, RAC is far more expensive in licensing costs than a single machine with multiple processors.

In 2001 when Oracle first released RAC, servers were far, far less powerful than they are today. At the time, it was relatively easy to max out the amount of CPU and memory bandwidth used by an Oracle database. For example, in 2000 the fastest Intel processor (Pentium III) did about 2,054 MIPS. Today, the fastest Intel processor, the Intel Core i7 875k does 92,100 MIPS (**https://secure.wikimedia.org/wikipedia/en/wiki/Instructions_per_second**). High-end processors today are roughly 60 times faster than they were in 2001. Memory bandwidth and bus speed have increased quite a bit as well, which is a major part of the equation. In 2000, it was extremely expensive to buy multi-processor computers; today it is the norm, with some single sockets boasting 64 processors, and most laptops—and even smartphones—built with multiple CPUs in single sockets. It is actually quite rare in the author's experience to see CPU maxed out on a current high-end box except in very special applications; the author has witnessed single non-RAC machines being able to do in excess of 2 million logical reads per second! For this reason, with today's high-end hardware, RAC really doesn't make much sense when viewed in a cost-benefit, accounting-based analysis for most applications.

Some people consider RAC more reliable, but in my experience it most certainly is not; there is still a single point of failure—the storage subsystem—and the complexity of RAC makes it far less bugfree. It is true, though, that if a single node in a RAC cluster does go down and the storage and interconnect are not affected, then it does allow some failover ability. Database sessions that were connected to the bad node can reconnect, and if the underlying application is designed for it, they can do so semi-transparently (using TAF); they can typically reconnect far faster than using a standby database or using Data Guard under Enterprise Edition. So, RAC does help protect against CPU/memory/network issues, but it does not protect against physical or logical storage corruption.

Note that in order to really use RAC, you will be required to buy the Enterprise Edition. The only Standard Edition RAC setup supported is very small; only two nodes can exist, and they're limited to one socket apiece.

Unless you really don't care about uptime, never invest in RAC if you haven't already invested in a standby database; a standby is much more important. Also, if you are considering really using RAC to its fullest, e.g., using "instance fencing" or service-based connection types, remember that to reliably implement them, you will need a setup with the same number of nodes for your standby database, as well as for your development/QA team(s), as you are using in production.

Using RAC does require a higher level of skills from your DBA team.

RAC can provide more availability in some situations if properly configured, administered, and monitored, but it is not for everyone.

Tip 5: Investigate Oracle Enterprise Linux (if considering Linux)

If considering Linux, be sure to check the pricing of Oracle Enterprise Linux (OEL) compared to Red Hat Enterprise Linux, and include support in there. This is a moving target, but each time I check, OEL is much cheaper than Red Hat Enterprise Linux, especially when support is figured in. There is also the added advantage that when something is wrong between the operating system and the RDBMS software, you can point your finger at one place for a solution: Oracle. Oracle also claims that there is improved performance using their kernel rather than the stock Red Hat Enterprise Linux–supplied kernel. CentOS, the free alternative to Red Hat, is typically nine months behind in both performance and security updates at the time of this article (October 2011). It can save some money, but not much when compared with the actual cost of the Oracle license over the lifespan of the database, given the low cost of OEL.

Tip 6: Do buy a boatload of RAM

Oracle loves RAM for most applications. Especially if you expect your application to scale, RAM becomes paramount. Oracle doesn't cost more if you have more RAM in your system. Remember also that, like faster CPUs, RAM is a one-time cost. It is also quite cheap (for most applications) to increase throughput compared to other options, e.g., adding processors. It is particularly good for applications that re-read a lot of data, which in real life is most of them.

Tip 7: Do spend more money on top-quality storage

The Oracle license does not cost more if you have better storage.

In my experience, the main bottleneck most databases experience is almost never CPU; they are rarely CPU-bound. Instead, they are I/O bound, and in particular IOPS bound (input/output operations per second); that is to say, they are not limited by the bandwidth of the storage subsystem but rather by the latency of single-block reads. For this reason, don't focus on the disk bandwidth; focus on IOPS. There is no other program that I know of that can stress storage as much as Oracle at high loads.

Saving money on disks will ultimately cost more, as there is no substitute for disk subsystems that can handle very high IOPS reliably.

Oracle provides a utility called "Orion" that can help qualify storage subsystem choices. If you have the luxury of benchmarking them, use it.

Failing disks will happen; it is not if they will fail, it is when. If your database has no way for the disk subsystem to recover from this (and you do not have a standby database), then your database will go down for awhile; the bigger the database, the longer it will be unavailable during recovery. With today's larger databases exceeding a petabyte, recovery can take days or weeks. Better storage subsystems will make it much easier and more transparent to trade out failed disks and to add capacity.

Cheap disks, if the company scales, will ultimately cost a lot of money to replace. Not buying enough storage will ulti-

mately cost a lot of money in Oracle administration time, juggling storage, and this obviously adds risk to the equation as well. Quality disk subsystems of sufficient capacity, while not cheap, are very inexpensive compared to trying to make do with inadequate solutions.

Tip 8: Always use a standby database

If your application requires high availability, there is no substitute for a standby database. It will double your costs, as you will need equivalent hardware for the standby, including Oracle licenses. If you are not using Enterprise Edition, failover will very likely be a manual operation, but it will still be an order of magnitude faster than restoring it from a backup unless the size is trivially small.

Unless you really don't care about uptime, never invest in RAC if you haven't already invested in a standby database; a standby is much more important.

Investigate the cost if your application is down for the time it would take for a full recovery from a backup. Very frequently, this cost is far, far more than the cost of a standby database, including all of the costs mentioned in tip #1.

Tip 9: Do buy Oracle Support

In my experience, not purchasing Oracle Support is best described as "penny wise and pound foolish," even though it is a recurring cost that usually runs about one-third of the purchase price yearly for the life of the database.

If you don't require 24/7 access or don't have the money for it, go with one of the cheaper support options. Remember, Oracle releases security patches four times a year, and if you do not have support, you cannot download or legally apply them. It also releases bug fixes often, and again, if you have not bought support, you cannot legally apply those bug fixes or even discover them using the Oracle Support website.

Even the cheapest option gives you free patches and upgrades, which in reality are absolutely required. If you do not have support and you run into an insurmountable software issue with Oracle, you will not be able to fix it, period. Buying emergency support from Oracle can be terribly expensive.

If you are using the current version of Oracle, released within the past six months to a year, you will most certainly see benefit from purchasing Oracle support.

In 2011, Oracle really tightened their release and support cycle; as of July 2011, you could not download or apply the latest security patches for version 10 unless you had purchased extended support, which is an extra-cost option. This means in practice that release cycles are and will be shorter than they have been in the past, and the length of time that you can get Oracle support for a version without paying extra for the extended support option is shortened compared to what it was in previous years. This will encourage the adoption of the current version of Oracle; without purchased support for it, you may well find the database to have bugs that can affect production in a very fundamental way.

Tip 10: If your application is in-house, engage your development team

There are smart ways of using an Oracle database in an application. There are also many ways of using it that can dramatically increase the demands on the database. If the application is predicted to grow a lot, to scale, then, it's very important not to use the database for things that it doesn't need to be used for. Develop metrics that can measure how much effort the database needs to expend to satisfy a given requirement; if these are tracked and provided to the development team in an easy and timely fashion, it can make quite a big difference in the machine capacity required—and, therefore, the ultimate licensing costs.

If the application is in the development phase, it may be possible to target one of the cheaper licenses (SE1 or SE) instead of Enterprise Edition. If you become used to the EE extra features, it can be quite difficult to eliminate their usage.

Be careful what you use Oracle for: Oracle is among the best RDBMSs out there, but many problems just don't require ACID or relational algebra to solve. Using it as a syslog/application log destination for your data center, including web logs, may not be the best use of it. Using it to store and report various performance metrics of machines in your data center may be convenient, but it will increase the amount of work your database has to do; investigate RRDtool or similar specialty databases. If you are trying to use it to store documents, especially with schemas that evolve often, investigate Hadoop or a more appropriate document-centric data-store solution. If you're using it for a graph database with nodes, edges, and properties, again, investigate a more appropriate data store. Using Oracle as a key-value store usually is vastly overengineered; look into Oracle BDB or one of the other well-known key-value data store databases. Using your Oracle RDBMS for the parts of the application that really require ACID, work in sets of data, and require full transactional support will usually reduce the amount of CPU you need to buy and, therefore, the licensing costs.

Also, place your business logic on servers that are not on the database, unless they need to gather lots of data to perform their function; again, this will reduce the CPU requirements of your database server and, therefore, your Oracle licensing costs. ▲

Jay Stanley began his IT career in 1986 with Xerox, specializing in data analysis, system administration, programming, and database administration. Since then, he's held a variety of IT jobs in database administration, programming/development, and systems administration. He has been working with Oracle databases since 1995 and has been an Oracle Certified Professional (OCP) since 1997 (7.3.4, 8.0, 8i, and 10g). Since then, he has worked with a wide variety of businesses, databases, and applications, ranging from small web-based startups to medium-sized companies to Fortune 500 multinational companies. Today he works as a Senior Database Consultant for Database Specialists, Inc. He considers his colleagues at Database Specialists to be the best Oracle database administrators in the business.

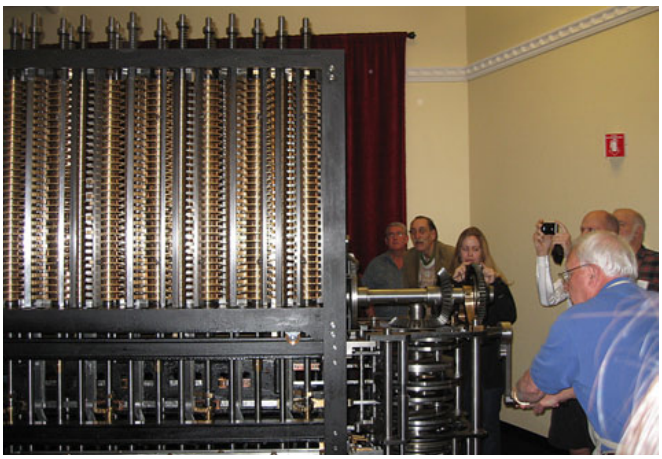
Pictures at an Exhibition

Conference #100 at the Computer History Museum was a real blast—*and* we smashed all attendance records. The sweetest accolade came from conference attendee Philip Rice of UC Santa Cruz, who insisted on giving us his membership check for 2012. Thank you, Philip! Please join me for a beer summit at Conference #200 in 2026! Another sweet accolade came from performance maven Craig Shallahamer, who wrote, *"I love speaking at NoCOUG; there are lots of really smart and inquisitive and full-of-energy attendees."* Please send us your pictures of Conference 100.

Thanks to everybody who made Conference 100 possible. Special thanks go to Quest Software, whose sponsorship dollars made it possible to hold Conference 100 at the Computer History Museum, and Hanan Hit, the star closer on the NoCOUG team, who competently handled logistics and catering, and secured the win for NoCOUG.

The 100th issue of the *NoCOUG Journal* is available at **http://bit.ly/rC2gRA**. It includes the 1970 paper on relational theory by Dr. Edgar Codd; a fascinating essay by Guy Harrison titled "How I Learned to Stop Worrying and Love Oracle"; a thought-provoking interview with the high-priest of relational databases, Professor Michael Stonebraker, titled "Words as Hard as Cannon-Balls"; and the results of the Second International NoCOUG SQL Challenge. All of the Conference 100 presentations are available at **http://www.nocoug.org/presentations.html**.

Conference 100 created a lot of positive energy for the years to come, and I hope that there will be even more attendees at Conference #101 on Thursday, February 23, at the Oracle conference center at 350 Oracle Parkway in Redwood City. Please mark your calendars; I hope to see you there. ▲



Static electricity crackles as a museum volunteer cranks the Babbage Difference Engine. The engine can compute mathematical formulas with more accuracy than Excel.

Close-up of one of the dessert trays provided by Extravaganza Catering. Can you say "petit four"?

Past and present board members reunite, including six NoCOUG presidents. Longest-serving board member Joel Rosingana models the Conference 100 commemorative tote while Dave Abercrombie waves the 100th issue of the NoCOUG Journal.

"Toad 11 is simply the best," say the signs at the Quest Software booth in the Hall of Fellows.

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 memberships. If your company is able to offer sponsorship at any level, please contact NoCOUG's president, Iggy Fernandez, at **iggy_fernandez@hotmail.com**. ▲

Long-term event sponsorship:

Chevron

Oracle Corp.

Thank you! Year 2012 Gold Vendors:

- Confio Software
- Database Specialists
- Delphix
- Embarcadero Technologies
- GridIron Systems
- Quest Software
- Quilogy Services

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

$ TREASURER'S REPORT

Naren Nagtode, *Treasurer*

Beginning Balance		
October 1, 2011		**$ 63,172.47**
Revenue		
Membership Dues	37.23	
Meeting Fees	1,050.00	
Vendor Receipts	2,000.00	
Advertising Fee	–	
Training Day	(4,043.40)	
Conference Sponsorship	10,000.00	
Interest	2.58	
Paypal balance	–	
Total Revenue		**$ 9,046.41**
Expenses		
Regional Meeting	15,118.52	
Journal	5,601.88	
Membership	0.90	
Administration	(20.00)	
Website	–	
Board Meeting	1,608.43	
Marketing	–	
Insurance	533.00	
Vendors	–	
Tax	–	
Training Day	191.70	
IOUG-Rep	–	
Miscellaneous	–	
Total Expenses		**$ 23,034.43**
Ending Balance		
December 31, 2011		**$ 49,184.45**

NoCOUG Winter Conference

Session Descriptions

For the most up-to-date information, please visit **http://www.nocoug.org**.

–Keynote–

Big Data—Are You Ready?
Mark Townsend, *Oracle Corporation* 9:30–10:30

As the world becomes increasingly digital, aggregating and analyzing new and diverse digital data streams can unlock new sources of economic value and provide fresh insights into customer behavior and market trends. But this data is often captured in new and different applications and formats, and the information density can be very low. To derive real business value from big data, you need the right tools to capture, organize, and analyze a wide variety of data types from different sources. This session focuses on how to unlock the business value of big data with an integrated set of products from Oracle.

Mark Townsend is responsible for leading Oracle Database product management activities, including requirement analysis, release planning, and coordination of field technical training. He is also responsible for Oracle XE. Mark has been with Oracle since 1991 and has specialized in the Oracle database for more than 15 years. He regularly talks to partners and customers, to better understand new technology and how it can be successfully applied to existing and emerging data-processing requirements.

–Auditorium–

Best Practices for Upgrading to Oracle Database 11*g*
Carol Palmer, *Oracle Corporation* 11:00–12:00

If you are planning to upgrade to Oracle Database 11*g* from Oracle Database 9*i* or Oracle Database 10*g*, this session is a rare opportunity to learn firsthand from an Oracle expert. In this session, we'll cover all the basics you need to know about upgrading to Oracle Database 11*g*, including all the required preparatory steps, which upgrade strategies best fit your environment, and how to address post-upgrade performance analysis, along with helpful tips and common pitfalls to watch out for.

Oracle Database Security Defense in Depth
Roxana Bradescu, *Oracle Corporation* 1:00–2:00

As the amount of digital data within organizations continues to grow at unprecedented rates, so does the value of the data and the challenges of safeguarding it. Two-thirds of sensitive and regulated data resides in databases, yet most IT security programs fail to address database security and the primary sources of threats that databases face today: insecure applications and privileged users. Learn about the evolving database attack landscape, adopting a defense-in-depth database security strategy with a strong focus on preventive controls as well as detective controls, and new solutions that offer transparent protection for Oracle and non-Oracle databases, ensuring the data privacy and integrity needed for sustainable regulatory compliance.

Private Cloud Database Consolidation
Mark Townsend, *Oracle Corporation* 2:30–3:30

Oracle Database 11*g*, Oracle Exadata, and Oracle Real Application Clusters enable consolidation of multiple applications on clustered server and storage pools—providing unbeatable fault tolerance, performance, and scalability. In this session, we'll discuss how to consolidate your databases onto a private cloud—and realize the efficiencies of mixed workload consolidation, workload and resource management, and dynamic provisioning for elastic scalability.

Introducing the Oracle Database Firewall: Your New First Line of Defense
Roxana Bradescu, *Oracle Corporation* 4:00–5:00

Millions of organizations worldwide have been attacked using SQL injection techniques. The reason these attacks are successful is that they exploit the relationship of trust between the application tier and the back-end database and, therefore, cannot be stopped using traditional methods. Learn about the recently introduced Oracle Database Firewall which provides unprecedented capabilities to defend against SQL injection and other threats by monitoring network activity and blocking unauthorized requests from even reaching your databases—all without impact to performance or costly and time consuming application changes.

–Room 101–

Accelerating Your Oracle Performance with SSD Technology
Michael Ault, *Texas Memory Systems* 11:00–12:00

In this presentation Michael Ault will show how SSD technology accelerates Oracle performance. Determining if SSDs are right for your application will be discussed and comparisons to other storage architectures shown.

Oracle Meets Fractals and Learns the Power of Power Laws
Neil Gunther, *Performance Dynamics Company* 1:00–2:00

Fractals are power laws in *space*. You're probably familiar with them as those infinitely recursive, self-similar, geometrical patterns made famous by the late Benoit Mandelbrot. Since then, power laws have been found to explain everything from forest fires to zip codes. It turns out that fractals can also impact computer system performance. However, to understand how that works, we need to consider fractals in *time*. That's

because time is the zeroth performance metric. Time fractals lead to self-similar clustering of events in time, and that can significantly affect things like the latency of packets in networks and (so it seems) queries in databases. This fractal clustering in performance data is a signal that certain large-scale correlations are at play. Since fractal or power law distributions, with their potentially infinite correlations, do not conform to standard statistical distributions (e.g., Poisson, exponential, normal), how can you possibly analyze such fractal performance data? That's what I will show you in this talk.

Hardware and Software Tools for the Oracle Fusion Middleware Architect

Arijit Das, *Naval Postgraduate School* 2:30–3:30

Oracle was known for its database product line but not for its middleware offerings. All that changed in the last decade, with major acquisitions and movements. For a small Oracle shop this has been a challenge, as one now has to deal with more patches, upgrades, testing, and support. To keep the customer seamlessly integrated, a small tech team has to come up with very savvy and cost-effective tools and techniques to provide the best production experience. In this presentation we look at how we put together a Research Collaboration Portal using Oracle fusion middleware products like Beehive, Oracle Portal, OID/SSO, reverse proxy servers, and the Oracle 11*g* database. When Oracle moved away from its shrink-wrapped product of Oracle Collaboration Suite, we had to stitch together new and upgraded products to provide the same user experience. In this effort we leveraged Dell hardware solutions, Redhat Linux software applications, and Oracle support to first test every version in the lab before rolling it out to production. We feel that the lessons learned can benefit a production shop of any size. We will give examples of reverse proxy servers to mask ports, breaking the RAID 1 mirror as a simple backup technique, memory tuning for Oracle Java containers to fully utilize physical memory, running several server applications on a single physical server using multiple network interfaces, and installing multiple fusion middleware schemas on a single Oracle 11*g* database, to name a few.

Increase Performance of Existing Oracle RAC up to 10X

Erik de la Iglesia, *GridIron Systems* 4:00–5:00

This session presents actual IT case studies and reference architectures for integrating Solid State Disk (SSD) into existing storage infrastructures to boost performance and enable virtualization especially in big data environments where data volumes are growing rapidly and applications and users are demanding high-speed concurrent access. Attendees will see how large Oracle environments can be more strategic in their deployment of SSD and reap tenfold performance acceleration without the need to reconfigure or move applications or data, without forklift hardware upgrades, and without changing data protection policies.

–Room 102–

Internals of Online Index Build

Saibabu Devabhaktuni, *PayPal* 11:00–12:00

Since the time Oracle introduced the online index build feature in 8*i*, it experienced issues from long index build times to contention during beginning and end phases of online index build. This presentation will explain how online index build was implemented internally in 10*g*, why contention can surface during beginning and end operations, and what shortcomings it had. I'll also explain how this behavior changed in 11*g*, along with internal implementation details; what shortcomings it still has; and how contention is still possible.

Just Get It Written: Deploying Applications to WebLogic Server Using JDeveloper and WLS Console

Peter Koletzke, *Quovera* 1:00–2:00

Deploying a Java EE web application consists of wrapping all application files into a single Java archive file (the EAR, enterprise application archive) and then copying this file to the application server. The Oracle WebLogic Server (WLS, formerly, Oracle Containers for Java EE-OC4J) is a standard Java EE runtime process to which you deploy Java-based applications for use in a web runtime environment.

Database Performance in a Virtualized World

Eric Jenkinson 2:30–3:30

More and more companies are utilizing virtualization in their environments to lower costs and effectively utilize resources. Placing Oracle databases in virtualized environments is not just for development and testing anymore. The rise of virtualization is introducing new performance challenges to the Oracle DBA that are not well documented at this time, and these issues are further complicated by the lack of knowledge of the virtualized configuration by the Oracle DBA. This presentation will bring the Oracle DBA up to speed on the virtualized configuration options in both VMWare and Oracle VM that impact Oracle Database performance. The presentation will also show new tools and procedures for Oracle DBAs to add to their toolbox, to help them better address performance issues in a virtualized environment. It is not the intent of this presentation to make the case for virtualization; its intent is to help the DBA identify and troubleshoot performance problems in a virtualized environment.

Do-It-Yourself Exadata: Is It Possible? **Editor's Pick**

Kerry Osborne, *Enkitec*. 4:00–5:00

The possibility of building a generic version of Exadata is attractive because the hardware components used in Exadata are well documented and readily available. One of the potential advantages of this approach is that the components can be purchased for less than the list price for Exadata. Building it yourself also allows you to modify the hardware specifications as you choose. For example, you can add faster CPUs or more memory. But is it really possible to reproduce (or come close to) the performance provided by Exadata? This presentation aims to answer that question. We'll compare the contribution of the various optimizations provided by the storage software v/s the contribution provided by the hardware components. ▲

Database Specialists: DBA Pro Service

DBA PRO BENEFITS

- *Cost-effective and flexible extension of your IT team*
- *Proactive database maintenance and quick resolution of problems by Oracle experts*
- *Increased database uptime*
- *Improved database performance*
- *Constant database monitoring with Database Rx*
- *Onsite and offsite flexibility*
- *Reliable support from a stable team of DBAs familiar with your databases*

CUSTOMIZABLE SERVICE PLANS FOR ORACLE SYSTEMS

Keeping your Oracle database systems highly available takes knowledge, skill, and experience. It also takes knowing that each environment is different. From large companies that need additional DBA support and specialized expertise to small companies that don't require a full-time onsite DBA, flexibility is the key. That's why Database Specialists offers a flexible service called DBA Pro. With DBA Pro, we work with you to configure a program that best suits your needs and helps you deal with any Oracle issues that arise. You receive cost-effective basic services for development systems and more comprehensive plans for production and mission-critical Oracle systems.

DBA Pro's mix and match service components

Access to experienced senior Oracle expertise when you need it
We work as an extension of your team to set up and manage your Oracle databases to maintain reliability, scalability, and peak performance. When you become a DBA Pro client, you are assigned a primary and secondary Database Specialists DBA. They'll become intimately familiar with your systems. When you need us, just call our toll-free number or send email for assistance from an experienced DBA during regular business hours. If you need a fuller range of coverage with guaranteed response times, you may choose our 24 x 7 option.

24 x 7 availability with guaranteed response time
For managing mission-critical systems, no service is more valuable than being able to call on a team of experts to solve a database problem quickly and efficiently. You may call in an emergency request for help at any time, knowing your call will be answered by a Database Specialists DBA within a guaranteed response time.

Daily review and recommendations for database care
A Database Specialists DBA will perform a daily review of activity and alerts on your Oracle database. This aids in a proactive approach to managing your database systems. After each review, you receive personalized recommendations, comments, and action items via email. This information is stored in the Database Rx Performance Portal for future reference.

Monthly review and report
Looking at trends and focusing on performance, availability, and stability are critical over time. Each month, a Database Specialists DBA will review activity and alerts on your Oracle database and prepare a comprehensive report for you.

Proactive maintenance
When you want Database Specialists to handle ongoing proactive maintenance, we can automatically access your database remotely and address issues directly — if the maintenance procedure is one you have pre-authorized us to perform. You can rest assured knowing your Oracle systems are in good hands.

Onsite and offsite flexibility
You may choose to have Database Specialists consultants work onsite so they can work closely with your own DBA staff, or you may bring us onsite only for specific projects. Or you may choose to save money on travel time and infrastructure setup by having work done remotely. With DBA Pro we provide the most appropriate service program for you.

DatabaseSpecialists

CALL 1-888-648-0500 TO DISCUSS A SERVICE PLAN

NoCOUG
P.O. Box 3282
Danville, CA 94526

NoCOUG Winter Conference Schedule

Thursday, February 23, 2012—Oracle Conference Center, Redwood Shores, CA

Please visit **http://www.nocoug.org** for updates and directions, and to submit your RSVP.
Cost: $50 admission fee for non-members. Members free. Includes lunch voucher.

8:00 a.m.–9:00	Registration and Continental Breakfast—Refreshments served
9:00–9:30	**Welcome:** Hanan Hit, NoCOUG vice president
9:30–10:30	**Keynote:** *Big Data—Are You Ready?*—Mark Townsend, Oracle Corporation
10:30–11:00	**Break**
11:00–12:00	**Parallel Sessions #1**
	Auditorium: *Best Practices for Upgrading to Oracle Database 11*g—Carol Palmer, Oracle Corporation
	Room 102: *Accelerating Your Oracle Performance with SSD Technology*—Michael Ault, Texas Memory Systems
	Room 103: *Internals of Online Index Build*—Saibabu Devabhaktuni, PayPal
12:00–1:00 p.m.	**Lunch**
1:00–2:00	**Parallel Sessions #2**
	Auditorium: *Oracle Database Security Defense in Depth*—Roxana Bradescu, Oracle Corporation
	Room 102: *Oracle Meets Fractals and Learns the Power of Power Laws*—Neil Gunther, Performance Dynamics Company
	Room 103: *Just Get It Written—Deploying Applications to WebLogic Server Using JDeveloper and WLS Console*—Peter Koletzke, Quovera
2:00–2:30	**Break and Refreshments**
2:30–3:30	**Parallel Sessions #3**
	Auditorium: *Private Cloud Database Consolidation*—Mark Townsend, Oracle Corporation
	Room 102: *Hardware and Software Tools for the Oracle Fusion Middleware Architect*—Arijit Das, Naval Postgraduate School
	Room 103: *Database Performance in a Virtualized World*—Eric Jenkinson
3:30–4:00	**Raffle**
4:00–5:00	**Parallel Sessions #4**
	Auditorium: *Introducing the Oracle Database Firewall: Your New First Line of Defense*—Roxana Bradescu, Oracle Corporation
	Room 102: *Increase Performance of Existing Oracle RAC up to 10X*—Eric de la Iglesia, GridIron Systems
	Room 103: *Do-It Yourself Exadata: Is It Possible?*—Kerry Osborne, Enkitec **Editor's Pick**
5:00–	**NoCOUG Networking and No-Host Happy Hour**

RSVP *required* at http://www.nocoug.org

The *NoCOUG Journal* design and production: Giraffex, Inc., S.F.
Front cover photo: Arnd Rockser/Photos.com