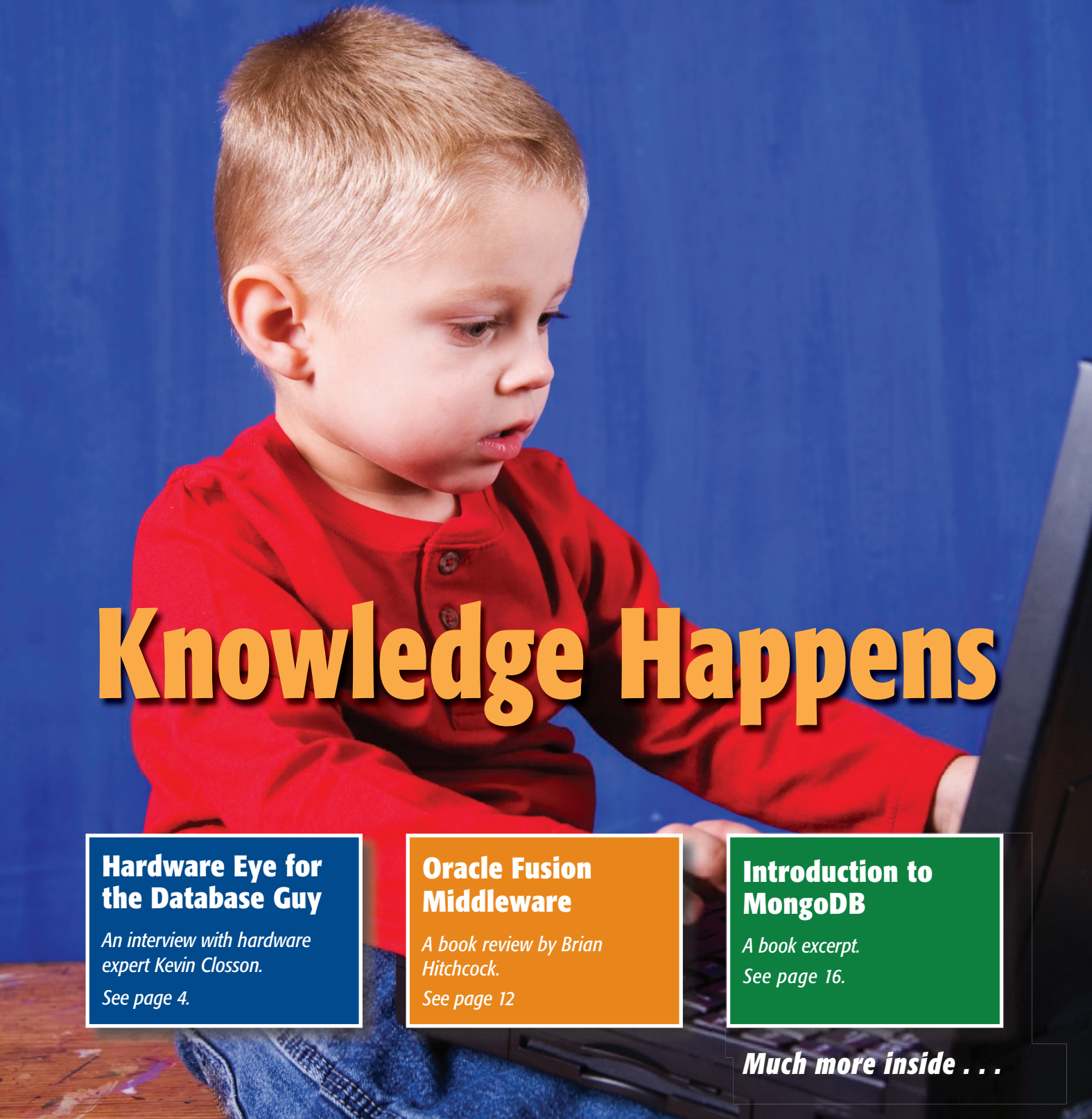
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 26, No. 3 · AUGUST 2012 $15



Knowledge Happens

Hardware Eye for the Database Guy

An interview with hardware expert Kevin Closson.

See page 4.

Oracle Fusion Middleware

A book review by Brian Hitchcock.

See page 12

Introduction to MongoDB

A book excerpt.

See page 16.

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

ADVERTISERS

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 November 2012 issue is August 31, 2012. Article submissions should be made in Microsoft Word format via email.

Hardware Eye for the Database Guy

with Kevin Closson

Kevin Closson

Kevin Closson is currently a technology director and performance architect in the Data Computing Division of EMC, focused on the Data Computing Appliance, a DW/BI/Big Data Analytics appliance based on the Greenplum Database. From 2007 through 2011, Kevin was a performance architect in the Exadata development team at Oracle Corporation's Server Technology Group. His work prior to Oracle, at HP/PolyServe, Veritas, and IBM/Sequent, was focused on throughput, scalability, and availability of the Oracle server. His Oracle porting work at Sequent led to U.S. patents in SMP/NUMA locking and database caching methods.

Kevin has collaborated on a number of books and contributed to Oracle Magazine, IBM Redbooks, and the IOUG SELECT Journal. He is a member of the OakTable Network and an Oracle ACE (retd). Kevin maintains a very popular technology blog at **http://kevinclosson.wordpress.com/.**

Is hardware the new software? Why bother with indexing, clustering, partitioning, and sharding (not to mention application design) if hardware keeps getting bigger and faster and cheaper every day. Can we just "load and go?"

A saying comes to mind: "*It's just software; anything is possible.*" No matter how capable computing platforms become, software will always find a way to demand more processing power.

As for your specific set of examples, I cannot envision a day when hardware can make up for the performance of a well-placed index. Imagine OLTP/ERP build on full scans. That just isn't going to happen. But no matter, because the design center for Exadata is Data Warehousing.

When bringing Exadata to market we were very vocal about removing indexes. That's because Oracle Data Warehousing—in an I/O-bound environment—absolutely requires the *misuse* of indexes. By misuse I mean the excruciatingly complex layering of indexes and the associated cost-based optimizer ramifications (bad plans) that come with such an approach. Essentially, indexes were used to mitigate poor storage provisioning and under-configured storage bandwidth. However, if you have a hardware platform that can scan storage at high rates of bandwidth, then, by all means, remove indexes and scan storage. If you need an index, create an index. Clustering and partitioning? Of course, why not?—use them where they solve problems. No matter how *fast* your hardware is you will still have to use the right tools for the right job. However, maybe *fast* isn't the important word!

There is a wave of *different hardware* in our future. There is a war between high core count and low power consumption. The concern for the next 10 years isn't rudimentary stuff like plumbing storage. Those are old, tired Y2K-era challenges. I think the real challenge is right-sizing applications (including the database) to the right power model. We can already see x64 SoCs (systems-on-chip) that draw single-digit wattage. Please don't laugh. Yes, I'm talking about a tiny little SoC with a single (possibly threaded) core and, perhaps, 4GB DRAM. Let's not focus so much, for a moment, on the clock frequency or maximum DRAM but, instead, let's focus on the *ratio* between CPU gigahertz and gigabytes of DRAM.

The ratio of DRAM to processor clock frequency on these up-coming server options is actually favorable for database processing. Indeed, current x64 flagship servers commonly mate, say, 20 gigabytes of DRAM to each gigahertz of CPU (i.e., 80 Xeon E7 cores at 2.4 GHz with 4TB DRAM). Building really big systems is no longer a problem with industry-standard (x64) parts. It hasn't always been that way.

Some of us remember the *card-cage rage* of the 1990s when one could buy a server with either a lot of CPU or a lot of DRAM but very seldom both. With those bad memories in mind one might conclude that the largest DRAM-to-CPU ratio would guarantee happiness. That's not always the case.

Idle processors do not speed up database processing! In database processing, the main purpose of DRAM is to drive up processor utilization—removing waits for high-latency storage accesses. However, the design trade-off for connecting large numbers of processors to very large memories is paid in CPU efficiency. What do I mean by CPU efficiency? Well, generally speaking, the *bigger* a system is, the *slower* it is. Allow me to explain by drawing upon TPC-C benchmark results.

You asked, "Is hardware the new software?"
No, it's the other way around.

Consider Oracle's two latest x64 TPC-C results. They are both very good results and they'll help me make my point nicely. These two results offer a classic comparison between a big system and a little system, each processing the same workload with the same instruction set. The "big system" was a Sun x4800 with 80 Xeon E7 processor cores and 4 terabytes of memory. The "little system" was a Cisco UCS server with 12 Xeon 5600-series cores and 384 gigabytes of memory. The big system delivered roughly 63,000 TpmC/core, whereas the little system delivered roughly 88,000 TpmC/core—40% more on a per-core basis. That's a significant increase, especially when you consider the fact that we license Oracle Database by the core. But there is more to be said on the matter.

Another way to look at this is TpmC per DRAM. The big system delivered roughly 1,200 TpmC per gigabyte of RAM. On the other hand, the little system squeezed out about 2,700 TpmC per gigabyte of DRAM. Finally, when we multiply the base clock rate times the number of cores in each configuration, we can calculate TpmC per gigahertz. As the math works out, they both generate roughly 25,000 TpmC/GHz. Simple facts like this tell us that right-sizing core processing power to DRAM is not always a more-is-better proposition. To that end, I'll come full circle and actually answer the question—sort of. You asked, "Is hardware the new software?" No, it's the other way around.

I no longer see a solid line between hardware and software. I see *platform technology*. Whichever platform technology is most capable of placing an application *where* it should be *when* it should be, *there* is the new hardware. I think of the flexibility and mobility of modern virtualization platforms like those from VCE (Vblock) and IBM (PureFlex System). Simply put, the "new hardware" is a software/hardware combination that *increases CPU-efficiency*. Any consolidation approach that does not offer this degree of flexibility is simply not state of the art.

So what about sharding and my mention of low-wattage SoC options as it relates to database? This is a delicate topic. I believe the architecture wars of shared-nothing versus shared-disk are old and tired. The fact is a scalable, massively parallel processing solution can be engineered from either of these storage architecture approaches. Speaking about products I'm intimately familiar with, I'll point out that Oracle is shared-disk and Greenplum Database is a "sharded," shared-nothing MPP—yet *both* are MPP. That matters very little. What matters is whether an MPP architecture is *symmetrical*; of these two, only Greenplum is symmetrical—and symmetry matters!

In Greenplum Database all code executes on all CPUs. That's symmetrical. One cannot drive up the utilization of processors unless there is symmetry. With Greenplum Database there is no "delineation of processing" with the associated bottleneck that entails. An example of asymmetrical architecture is Exadata. Compute-intensive code such as joins, aggregation, and sorting can only execute on the host processors, while filtration, projection, and HCC decompression *can*, and usually do, execute on other physically distant processors in the storage grid. I've written on this topic at great length. It is a simple problem to understand but a very difficult architectural problem for Oracle to mitigate—and I say *mitigate* because Oracle is quite stuck with the "chopped in not-quite-half" architecture. And now is when I come full circle to my mention of large-scale SoC platforms.

Greenplum Database is "just software." The proximity of the cores matters none. I'll expound on that point by drawing a mental picture built on huge node-count SoC—but I'm in no insinuating future product. Greenplum has vastly fewer architectural roadblocks than Oracle to prevent it from exploiting tens of thousands of gigahertz of SoC processing power—but that would require thousands of instances of the database. Greenplum Database is routinely deployed with extremely large instance-counts. I'm not in a field organization but I am aware of deployments that have over 1,000 segments (aka, shards) of the Greenplum Database. I cannot picture even half that many database instances with Oracle Real Application Clusters. It's for this reason that I think future platform technology will be exploited by the knowledge that software is, sort of, the new hardware.

In 2008, Exadata was a traditional-style (meaning software-only with ports).

Exadata v/s the World

You were an architect in Oracle's Exadata development organization. With all the excitement about Exadata in the Oracle community, what motivated you to leave?

That is a great question to get things started. There is a lot of confusion about what the word "architect" means in Oracle Server Technologies. Architect is an HR career level, not a title. Many household names hold that rank, including the author of Automatic Storage Management, Bill Bridge, and fellow OakTable Network members Graham Wood and Tom Kyte. Architects can have varying focus. Mine was a very narrow focus on the performance of Exadata.

The wherefore and why, as it were, behind my resignation from the Exadata organization is rooted in the path that lead me to join the organization in the first place.

The decade of the 1990s found me working as a lead performance engineer in the Advanced Oracle Engineering group of Sequent Computer Systems, where I focused on port-level optimizations of the Sequent port of Oracle. At that time Oracle Server Technologies had a team of performance engineers that focused on high-end multi-node scalability. That testing was conducted on Sequent gear, so I was involved in a lot of their efforts.

Fast-forward about 15 years to a time frame that found me working for HP after its acquisition of PolyServe, a startup where I was working. PolyServe got absorbed into the StorageWorks Division of HP.

Soon after, I learned of interesting joint-development work between Oracle and HP on a project called SAGE—an acronym that stands for Storage Appliance for Grid Environments.

SAGE was the project name that begat the Exadata product. For this reason I would have been involved with Exadata—one way or another. However, Oracle phoned me up to see if I was interested in joining the development effort from the Oracle side. What does this have to do with my ramblings about the mid-1990s? Well, the folks that recruited me to Oracle were the same folks that I worked with way back in the '90s in my role at Sequent. So we've gone full circle, but that only sets the stage for why I eventually left Oracle.

In late 2008, SAGE became the HP Oracle Database Machine based on Exadata Storage Server software. In that time frame Exadata was a traditional-style (meaning software-only with ports) Oracle product with a specific design center to make Oracle Database a more competitive solution in the Data Warehousing space. Oracle was losing a bit of business to niche players such as Netezza and Greenplum and, of course, Teradata—the most long-of-tooth in this category of vendors.

In 2010, Oracle bought Sun Microsystems, and the Exadata ports died.

In 2010 Oracle bought Sun Microsystems, and the Exadata ports died—along with the typical Oracle-style Open Systems mindset. I joined Oracle to work for the software company I had known for 20 years but wound up working for a company with the pressing need to push hardware and eschew the Open Systems mentality that made Oracle Database the dominant market leader.

The acquisition changed the project beyond recognition in ways I didn't agree with. So I resigned and eagerly joined EMC to focus on Greenplum Database—a software solution for very high-performance Data Warehousing/BI/Analytics. My specific focus is platform performance engineering of our appliance-embodiment of Greenplum Database—called the DCA (Data Computing Appliance).

You've been referring to it as "Exaggerdata." Why so harsh? You've got to admit that there's plenty of secret sauce under the covers.

Yes, in social media I take a rather irreverent approach to what I see as extreme over-marketing of Exadata. To make my point I'll draw upon a couple of examples. I'm not sure your readers are aware that the adoption of NAND Flash in Exadata Storage Servers had nothing to do with OLTP, and yet I'm sure that all of your readers are aware that the presence of this read cache is the sole characteristic that supposedly elevates Exadata to the status of The World's First OLTP Machine.

Exadata Smart Flash Cache was originally seen as a way to improve data flow for Data Warehousing. It's quite simple. Current-generation Exadata Storage Servers have five PCI slots over which database I/O can flow—but only 12 hard disk drives. The 12 hard disk drives are attached via a single OEM'ed LSI PCI controller. Therefore, a quarter-rack Exadata Database Machine—without Exadata Smart Flash Cache—is limited to roughly 5.5 GB/s of storage scan bandwidth with the high-performance, low-capacity SAS hard disk drives. The story changes, however, when four more PCI slots are populated and filled with cached table and index data. With all five slots actively flowing data, the storage scan rate is increased to about 16 GB/s for a quarter-rack. It was never about OLTP or ERP, because these use cases need to scale up writes along with reads. A platform that cannot scale up writes along with reads is not a good OLTP platform—much less the World's First OLTP Machine. Indeed, since a quarter-rack X2 model has but 24 total WSM-EP cores and capacity for only about 5,000 random writes (less with high redundancy) I see no reason to go that route for OLTP/ERP, thus missing out on a 21st-century technology injection by going totally integrated, virtual, and manageable via something like VCE Vblock. Yes, I'm bullish on this integrated solution and the many management and flexibility characteristics it offers. I suppose I should get back to the point about Exaggerdata.

It's really quite simple and I'll take Exadata out of it just to make the point. If one attributes, say, a 100-fold performance increase to any technology solution, one of two things must be true: either (1) whatever the bottleneck was on the slow side of the comparison is 100-fold faster on the fast side, or (2) the workload is not the same.

One online interview with an Oracle executive comes to mind. In the interview the executive (incorrectly) attributed "10x" faster data flow to Infiniband and touted 10x data compression (which is possible but not certain). Now in spite of the fact that Infiniband is not 10x faster than alternative, modern storage plumbing (e.g., 16GFC Fibre Channel), and in spite of the fact that Infiniband can be employed without Exadata and in spite of the fact that 10x compression is certainly not guaranteed, the executive simply multiplied the two tens and suggested Exadata is 100x faster for the same Oracle Database processing. This is junk science and we all make mistakes, so perhaps that what it was—a mistake. However, I see too much of this sort of mistaken junk science.

See, Oracle's marketing is full of non-Exadata-to-Exadata testimonials where Oracle database (unsurprisingly) remains a constant ingredient. Unless one migrates from something like Oracle 5 on 20 MHz processors to Exadata, astronomical improvements are only partially attributable to hardware.

Any hardware refresh that also includes a database upgrade is an effort that invariably includes SQL tuning. An improved query plan can easily result in a hundreds-fold reduction in processing, and SQL tuning doesn't come in a 42U engineered system. SQL tuning is science available to all Oracle customers. Too often IT shops are overspending for Exadata and mitigating that pain by basking in the multiple-orders-of-magnitude improvement—improvement that, all too often, would have come to them naturally with a hardware update and the same dutiful SQL tuning that took place during their Exadata migration. My point is there are thousands of highly-skilled Oracle tuning specialists whose skills are not obviated by Exadata (I presume some of them read this publication). OK, so please bear with me for another point on this exaggeration theme.

Exaggeration is often borne of excitement and I under-

Two Million Database Professionals Count on One Solution.

Simply the best for Oracle database professionals - Toad 11. Supported by over a decade of proven excellence, only Toad combines the deepest functionality available, extensive automation, and a workflow that enables database professionals of all skill and experience levels to work efficiently and accurately. Countless organizations empower their database professionals with Toad. The best just got better.

Watch the video at **www.quest.com/Toad11SimplyBetter**.

QUEST SOFTWARE®
Simplicity At Work™

stand that. However, I see the need for some corporate responsibility in the matter. Allow me to explain. In a recent Oracle Technical Case Study White Paper **https://bitly.com/ExadataTechnicalCaseStudy** (page 2), Exadata was credited with improving an Oracle report by 475% compared to the pre-Exadata processing completion time. An improvement of nearly five-fold *sounds* great; however, the white paper clearly details the improvement as reducing an MRP planning process from 302 minutes to 85 minutes—which is, of course, a 72% reduction in processing time. However, I wanted to use this vein of the Exaggerdata question to draw out an important fact—a fact that I needed this particular Technical Case Study to make.

In spite of the Data Warehousing design-center of Exadata, it appears the most popular use case for adopting the technology is OLTP/ERP.

In spite of the Data Warehousing design-center of Exadata, it appears the most popular use case for adopting the technology is OLTP/ERP. Many of the Oracle customer testimonials for this use case are not actually of the Exaggerdata caliber. In fact, most of what I've read for the OLTP/ERP use case on Exadata are the sorts of performance improvements one would naturally *expect* from a hardware refresh and database upgrade. How many Oracle shops bother going through a hardware refresh that doesn't roughly double their performance? That's expected. If you read the above-cited white paper you'll see mention of performance improvements that fall right in the normal/expected range of a hardware refresh. In particular, the paper calls out how the customer's *"top 20 critical concurrent batch jobs now run on average 46% faster"* and *"20% to 50% performance gain in their Advanced Supply Chain Planning reporting cycles"* and *"month-end related jobs now run on average 67% faster."* May I ask who *wouldn't* expect *any* hardware refresh to improve performance to this degree? Those improvements are natural for a hardware refresh—moreover, we would presume this particular customer benefitted from at least some intrinsic Oracle Database improvements because they migrated to Exadata with Oracle Database 11*g* from 10*g*. Oracle still has database kernel engineers working on the base product, so we can assume it is naturally getting better, no?

Thus far I've used the referenced technical case study to show a mild version of Exaggerdata. I've also used it to show that a migration to Exadata for workloads that fall outside the design center—specifically OLTP/ERP—do not enjoy the fabled hundreds-fold improvement we constantly hear about, but I'm not done with this train of thought. I'm addressing your question about my use of the silly word Exaggerdata. Well, no Oracle-produced Exadata collateral would be complete without mention of *something* running upwards of 100x faster.

In table 1 (page 9) of the report there is more detail to help account for the average 46% improvement for the top 20 critical concurrent batch jobs. The table calls out three of the 20 reports, each enjoying an average of 103-fold performance increase. These sorts of citations deserve scrutiny.

The Exadata system in question was a half-rack that offers table scan rates up to about 38 GB/s if the tables/indexes fit in the Exadata Smart Flash Cache. If the cited 100-fold improvement is attributable to the half-rack scan rate, then the old system was delivering only about 350 MB/s scan I/O—or a single 4GFC Fibre Channel path. That wouldn't be a very interesting hardware comparison at all. If, on the other hand, these specific jobs got their 100-fold improvement from CPU, then the old system would have had less than 4 GHz total CPU power, because the *entire* half-rack has 132 cores at roughly 3 GHz—or roughly 396 GHz total processing power. When you look at these claims critically it takes no time at all to realize that what's really happening here is a few of the reports improved significantly (e.g., 100-fold) but it's Oracle on both sides of the equation and therefore I cannot accept the notion that the customer was running Oracle Database 10*g* on a system of 1/100th the processing power of a half-rack.

There is much to be learned from these case studies—not only in terms of exaggeration, but they also underwhelm. Allow me to explain. Page 8 of that paper shows us that in spite of all the heralded performance improvements, *"peak CPU usage across all nodes of the cluster was observed to be 50%."*

In this day and age I see little value in IT overspending and even less value in waste. For these reasons I'm bullish on the established, best-of-breed 21st-century approach to platforms—such as those I've already mentioned.

"Do-it-yourself Exadata-level performance?" Really? We're all ears.

I presume the question is geared towards DIY (do-it-yourself) Exadata for data warehousing, because the question is actually opposite for non-DW. Let me put that another way. If the question happened to be "DIY non-Exadata OLTP performance with Exadata" I'd have said no, too difficult! Allow me to explain. Exadata has limited random write bandwidth that is further exacerbated by the best-practice ASM high redundancy. Exadata offload processing features do not come into play in OLTP. Oracle helps me prove the point about non-Exadata OLTP by routinely publishing record-breaking TPC-C and E-Business Suite benchmarks *without* Exadata. Moreover, Oracle's very own E-Business Suite Global Single Instance is not on Exadata. So, obviously, Oracle Database handles OLTP/ERP quite nicely without Exadata and even better with modern storage that improves service times for *both* reads and writes (unlike Exadata Smart Flash Cache). So what about DW/BI?

As I've said, DW/BI is the design center for Exadata and it obviously has value in that space—but that doesn't mean it is absolutely required. This gets a little complicated. We know that without Exadata Smart Flash Cache the full-rack scan rate is 25GB/s—so Flash Cache is extremely important. That said, think about the fact that a full-rack has but 5.3 TB of flash. If your database doesn't fit entirely in that cache, your

scan starts out fast and finishes slow. So if you are ecstatic about Exadata it is sort of implied that these databases either (a) fit in the limited capacity of the Flash Cache, or (b) that a 25 GB/s scan rate is a sufficient complement to 96 Xeon 5600 CPUs (or 160 Xeon E7 cores in the X2-8 model). Who is buying full racks? No matter: 25, 12.5, and 5.4 gigabytes per second (full-, half-, quarter-rack respectively) are certainly not impossible scan rates with modern storage. But that is not exactly my point.

The processors in the host grid of Exadata are the bottleneck for complex, concurrent query; data loading; transformations; gathering statistics; etc. Remember, the most CPU-intensive processing in Exadata (joins, aggregations, sorting, data loading, etc.) is not offloadable to storage. I encourage Oracle customers to take advantage of the "reverse PoC" program that Oracle has where they send you a dataset and queries along with the timings for quarter-, half-, and full-rack. You'd be quite surprised just how little physical I/O it takes to saturate 96 Xeon 5600-series processors when enduring CPU-intensive processing. Yes, Exadata has storage offload processing, which frees up some cycles, but until you actually study Exadata you'd be surprised just how little is offloaded. It is, after all, only filtration, projection, and HCC decompression, and while the latter is very CPU intensive it merely amplifies the rate at which data is sent to the easily bottlenecked host CPUs. My words on this fact appear in my blog, in *Expert Oracle Exadata* (Apress), and also in my "Exadata Meets Critical Thinking" video presentation. But that is not the end of the train of thought on this question.

I have asserted that I/O bandwidth is not that high without Exadata Smart Flash Cache. We know the capacity of this cache is 5.3 TB. Don't we realize how easy it is to build very large memory systems? Why not build a cluster with, say, 10 TB aggregate DRAM and use Oracle In-Memory Parallel Query and attach it to conventional storage to take advantage of modern data management capabilities? You might say, "But, Kevin, there would be no HCC in that model"—and you would be correct because Oracle has issued patches that disable this generic feature for customers unless Oracle-branded NFS storage is discovered. My view on the matter is that HCC really shouldn't be updated, so why not just settle on the lower compression ratio of the non-HCC compression type and magnify your DRAM capacity in that manner? Indeed, it isn't that tricky to configure, say, four servers of 4 TB DRAM each and magnify that 3-fold with a non-HCC compression ratio. That is a large data warehouse, and last I checked scanning DRAM is faster than the full-rack datasheet scan rate of 75 GB/s—which, again, is only possible when the data fits in the flash cache.

The point I'm trying to make is that Exadata, as it has been said, is "just Oracle," but that opens a wide array of options for solving problems as I see it. Now, to close that train of thought I have to say that I'd still have no qualms pitting Greenplum Database against that In-Memory Parallel Query model—but to go into that would probably take another interview.

Partitioning (shared everything) or sharding (shared nothing)? Is there a clear winner? The EMC Data Computing Appliance uses sharding, doesn't it?

Your readers will breathe a sigh of relief because this is a simple question with a short answer. As I see it, the question is moot. In Data Warehousing tuples need to flow through CPUs—full stop! So all that really matters is CPU. All the rest of this conversation is about getting data through the CPUs. As long as the processing architecture is symmetrical I really don't see how the presentation of storage matters. If the flow of data is not impeded and your CPUs are fully, and effectively, utilized due to symmetrical architecture, you have a good DW/BI solution.

I am a long-time advocate of Oracle over NFS (even more so with dNFS) because of the incredible ease of use—especially with Real Application Clusters.

Are TPC benchmarks relevant today? The PostgreSQL wiki says:* "The TPC-H Benchmark is a popular one for comparing database vendors. The results [of running the TPC-H data set on PostgreSQL] have generally been disappointing, for reasons that aren't necessarily relevant in the real world. PostgreSQL is missing some of the things needed to do well on this benchmark, whereas commercial database vendors are so focused on it they will 'game' TPC-H runs (add optimizations specifically aimed at it) to make absolutely sure they do well."** ***Sounds like a bad case of sour grapes to me. However, it's curious to me that we don't see Exadata benchmarks or TPC-E benchmarks from Oracle. Perhaps TPC (or a big part of it) is no longer relevant after all. Would you agree?

I still think TPC-C is relevant. I don't see TPC-H as particularly relevant mostly because the schema and queries are a 20-year-old mentality of data warehousing. I also think customers would still be compelled to conduct proof-of-concept (PoC) testing even if there happened to be Exadata, Greenplum, and/or Netezza results. I'm not seeing any end to the PoC trend—but I'm not in a field organization. I do, however, get pulled into PoCs, which leads me to another point.

You spoke of features that "game" the benchmark. Features like Results Cache fall into that category. Some would argue that Materialized Views should fall into that category as well. I actually disagree on both accounts. These sorts of features are often the brain child of these benchmarks and are features that have value in real life. Speaking of materialized views specifically, I know of a recent Exadata versus Greenplum Database PoC where the Oracle engineers employed materialized views to help out their result. Now this is interesting if you think about it. If it was done in a PoC, that means it was *needed* in the PoC and materialized views are not an Exadata feature. I'm only bringing this up to make a point. I firmly believe that if Oracle Database is going to work at all for a particular use case, then it can be done with or without Exadata. Let me add another thought to that. If DRAM cost the same, by weight, as your favorite carbonated beverage, would any-

one buy Exadata? Does Exadata solve any in-memory processing problems? No. You don't offload processing to storage if the database fits entirely in host DRAM. What is the future? More mechanical disk or less mechanical disk? Flash treated like storage or Flash treated as memory? I think I know the answer to those questions, and for that reason I believe Exadata is transient technology.

> ***I do not think the RAID5 argument is the same as it was early last decade, nor do I think it is an all-or-none proposition.***

Storage in the Spotlight

Can NAS match the performance of SAN? Can SAN match the performance of locally attached storage?

Yes. Don't confuse protocol with bandwidth. One no longer dictates the other. I am a long-time advocate of Oracle over NFS (even more so with dNFS) because of the incredible ease of use—especially with Real Application Clusters. The storage presentation model of NFS for Oracle is outstanding: Mount file systems and go. One type of storage for anything related to Oracle. Complexity makes me crabby.

As a long-time NetApp fan, I like to stay in my comfort zone but perhaps I'm biased. Does NetApp still offer anything that the other NAS vendors (such as EMC) don't?

I respect companies that compete honorably. I've always liked NetApp, and although I work for a competitor I don't have to trash competitors.

NetApp pioneered so many things, but I don't see huge gaps between NetApp and EMC feature-wise any longer. If you would have asked me about NetApp versus EMC in about 2006, I would have sided with NetApp (for NFS filer use case) for one simple reason: ease of use. There are a lot of good filers out there now, and VNX from EMC is absolutely one of them. I think the important comparison between filer vendors is in the huge portfolio of data management (backup, DR, etc.) that EMC and others offer. There is more to total data management than get getting data in and out of a filer. In short, I'm not one of these death-to-competitor sorts. EMC is focused on NFS and has a very good dNFS-supporting offering (VNX).

Do you recommend that I use ASM? I worry that a sleepy system administrator will unplug my disks someday because "there were no files on them." I do like to have my files where I can count them every day!

ASM is a feature of the database—you paid for it so use it to solve problems! So, yes, of course! But don't use it if it doesn't solve problems. Let me give you an example. ASM was created to help deal with raw disk. I am a staunch advocate of Oracle on NFS (dNFS). Oracle fully supports creating large files in your filer and using them as ASM disks. The I/O will be direct so the overhead is nil. Some folks also use ASM via dNFS to multiple filers as a way to get more scalability out of their NFS model. This is mostly a concern for single-headed filers, of course. So, yeah, use it to solve problems.

Where do you stand in the "Battle Against Any Raid Five (BAARF)?"

You know what? I think much of what we "know" in IT has a shelf life. A personal friend of mine, James Morle, started the BAARF movement early last decade. Let me start by sharing two nuggets of wisdom. First, not all databases deployed in non-RAID5 enjoy good performance, and second, not all RAID5-based databases suffer poor performance! Profound, eh? I think quite often RAID5 is blamed for performance problems that should be pinned on multiple applications sharing the same mechanical storage, storage provisioning based on capacity instead of IOPS, and—to some extent—array storage processors that don't live up to their duty.

I have not read all of EMC's literature on the matter. However, the material I have seen doesn't suggest dumping an entire database on RAID5 LUNS. In fact the literature generally suggests RAID10 for certain parts of the database and RAID5 for the other parts. Indeed, different parts of the database have different access patterns and read/write ratios, so that doesn't seem all that silly to me. Further, products like EMC FAST (automated tiering) go a long way to mitigate the service times from the varying tiers.

> ***Not long ago EMC IT migrated from large SMPs into a virtualized environment with VMware.***

I'm not going to put my EMC sales hat on because I'm a terrible salesman. But the fact is a lot has changed since the mid-'90s array technology that drove legions of skilled Oracle-savvy professionals to sign the BAARF initiative in the early 2000s. To summarize, I do not think the RAID5 argument is the same as it was early last decade, nor do I think it is an all-or-none proposition.

Do we need yet another "silly little Oracle benchmark"?

Yes. Your readers can Google "Oracle SLOB" to get all the information on the matter.

Modern Database Topics

Do you have an opinion on the NoSQL movement? Will the relational mothership ever catch up with the speedboats?

People don't want to solve 21st-century problems in the precise same manner as they solved 1990 problems. I can't blame them. It's really not a comparison between relational versus non-relational. I'm sure it is possible to retrofit RDBMS technology to do about anything. I'm just not sure every new *thing* that comes along warrants a retrofit. It'll be fun to watch.

(continued on page 15)

Third International

NoCOUG SQL & NoSQL Challenge

Sponsored by Pythian—*Love Your Data*™

BE IT KNOWN BY THESE PRESENTS that the Wicked Witch of the West needs your help to create a magic spell to ensure that the Third Annual Witching & Wizarding Ball is a grand success. A great tournament has been organized for practitioners of the arts of Es-Cue-El & No-Es-Cue-El to demonstrate their magic powers.

Mind-Boggling Puzzle

The Wicked Witch of the West has invited six friends to the Third Annual Witching & Wizarding Ball at Pythian Academy of Es-Cue-El & No-Es-Cue-El. Burdock Muldoon and Carlotta Pinkstone both said they would come if Albus Dumbledore came. Albus Dumbledore and Daisy Dodderidge both said they would come if Carlotta Pinkstone came. Albus Dumbledore, Burdock Muldoon, and Carlotta Pinkstone all said they would come if Elfrida Clagg came. Carlotta Pinkstone and Daisy Dodderidge both said they would come if Falco Aesalon came. Burdock Muldoon, Elfrida Clagg, and Falco Aesalon all said they would come if Carlotta Pinkstone and Daisy Dodderidge both came. Daisy Dodderidge said she would come if Albus Dumbledore and Burdock Muldoon both came.

The Wicked Witch of the West needs an Es-Cue-El or No-Es-Cue-El spell to determine whom she needs to persuade to attend the wizarding ball in order to ensure that all her invitees attend.

Awesome Prizes

The *August Order of the Wooden Pretzel* will be conferred on the winner, in keeping with the celebrated pronouncement of the supreme wizard of Pee-El/Es-Cue-El that *"some people can perform seeming miracles with straight Es-Cue-El, but the statements end up looking like pretzels created by somebody who is experimenting with hallucinogens."* As if that singular honor were not enough, a fiery wizarding device that magically displays Oracular tomes will be bestowed upon the champion.

May the best witch or wizard win!

RULES: Pythian will award a Kindle Fire or an Amazon gift card of equal value to the winner. Prizes may be awarded to runners-up at the discretion of NoCOUG. Submissions should be emailed to **SQLchallenge@nocoug.org**. Contestants may use any SQL or NoSQL technology to create the magic spell. The competition will be judged by Oracle ACE Director Gwen Shapira and the editor of the *NoCOUG Journal*, Iggy Fernandez. Judging criteria include correctness, originality, efficiency, portability, and readability. The judges' decisions are final. The competition will end at a time determined by the organizers. The judges and organizers reserve the right to publish and comment on any of the submissions, with due credit to the originators. More information about the problem and additional rules can be found at **http://www.nocoug.org/**.

Pythian
love your data

Oracle Fusion Middleware 11g Architecture and Management

A Book Review by Brian Hitchcock

Details

Authors: Reza Shafii, Stephen Lee, and Gangadhar Konduri

ISBN: 978-0-07-175417-0

Pages: 560

Year of Publication: 2011

Edition: 1

List Price: $60

Publisher: Oracle Press

Overall Review: Excellent coverage of Fusion Middleware.

Target Audience: Anyone new to Fusion Middleware and Fusion Applications.

Would you recommend this book to others: Yes.

Who will get the most from this book? Administrators.

Is this book platform specific: No.

Why did I obtain this book? See overall review below.

Overall Review

I need to be prepared to support Fusion Applications. I found many resources and among them were three books from Oracle Press. The first was *Managing Oracle Fusion Applications*, the second was *Oracle WebLogic Server 11g Administration Handbook*, and this is the third. Since I have reviewed other Oracle Press books, they sent me copies of each to read and review.

Fusion Applications is built on top of Fusion Middleware. Understanding the components of Fusion Middleware and how they are managed is a critical part of supporting Fusion Applications.

I learned things in this book that have really helped me understand other parts of Fusion Applications. Of the three books, I thought this one wouldn't tell me as much due to its focus on Fusion Middleware. I was wrong. Because Fusion Middleware is the foundation of Fusion Applications, I learned many things in this book that were not mentioned in the other two. I think that resources that focus on Fusion Applications or on WebLogic Server tend to assume you know all about Fusion Middleware. It is also true that Fusion Applications and Fusion Middleware are such big topics, spanning so many areas, that it just isn't practical to cover Fusion Middleware topics and anything else in one volume.

Introduction

The introduction explains the book's organization, objective, and target audience. This is all fine, but I want to react to a specific comment by the author: "*Fusion Middleware 11g products take advantage of industry open standards to ensure that the new skills and knowledge needed for their use are minimized.*" Really? My experience is that Fusion, both Fusion Applications and Fusion Middleware, involve many, many new concepts that I had not seen anywhere before. It may well be true that if industry standards had not been used throughout all Fusion products, there would be even more different pieces and parts to learn about. Still, the amount to learn, in my opinion, is huge. Do not let this statement make you think you can quickly pick up Fusion Middleware or Fusion Applications. You will need to follow a learning curve that is both long and, in parts, steep.

This book includes a Use Case. The Use Case starts at the end of Chapter 2. The scenario described covers deploying applications to an evaluation environment. At the end of most chapters, the Use Case environment is extended to include the Fusion Middleware components discussed in that chapter. The Use Case adds context to the material covered in each chapter. It helped me to follow along as each component was added to the Use Case environment.

Chapter 1—Fusion Middleware Primer

This chapter provides a much needed starting point by covering not only what middleware in general is, but also, more importantly, why it has come to be needed. The need for middleware is described as "*the need for software applications that exist not to provide actual business logic for the enterprise but to facilitate and optimize the execution of software that does.*" This is a much better explanation of what middleware is than what I've seen before. Middleware exists *in the middle*, between the machine operating system and the application software. Fusion Applications is an example of enterprise software that is built on top of middleware, specifically Fusion Middleware. The Oracle Fusion Middleware product set is described. Identity management, service-oriented architecture, and other high-level aspects of enterprise applications are reviewed.

Chapter 2—Oracle WebLogic Server

This chapter covers details of the WebLogic Server. Since I had already read and reviewed *Oracle WebLogic Server 11g Ad-*

ministration Handbook, a lot of the material was familiar. I still found many things that I hadn't seen before, and some were explained in more detail. I didn't know that WLS was the first application server to meet full-specification compliance with the J2EE standard. The explanation of exactly what WLS is and does is very good. It is explained that servlets are stateless, and running them in a container maintains the state information. Java provides MBeans to track the configuration and state of components, and this is all part of the JMX extension to Java. The WLS Admin Console and the WebLogic Scripting Tool (WLST) are both examples of JMX clients; that is, they both read and modify MBeans to view and change the configuration of a domain. WLST can be run in both online and offline modes, with WLST connected or not to the servers of the domain. When running WLST in offline mode, the servers should be down. Another feature of WLST that I didn't know about is the ability to create a WLST script while navigating the Admin Console. This way you will have the needed WLST commands if the Admin Console is not available.

We are told that the embedded LDAP supplied as the default authentication, authorization, and role-mapping provider for Fusion Applications is not usually used due to scalability limitations. Also interesting are JDBC data sources, which are an abstracted connection to one specific database schema and a pool of active connections. Both the multidata source and the Gridlink data source are described as JDBC options when using a RAC database.

The Java Messaging System (JMS) is explained well, as is WLS request management, where incoming requests are associated with a thread. Work managers provide different priorities for different threads.

The chapter ends by continuing the Use Case. The Node Manager is added to the environment, and it is explained that it is not part of any domain; rather, it is needed on each machine where WLS instances run.

Chapter 3—Fusion Middleware Common Infrastructure

Here we find a good example of why I am glad I read all three books. While the concept of a machine is not covered in detail, the explanation of a domain that I found in this chapter is new and helpful. Another example is the definition of a farm, which I did not find in the other two books. I also found good information on OPMN: what it is, how it works, and why it is needed.

A detail about WLST that I hadn't seen before is described—namely that each Fusion Middleware component has some additional WLST commands unique to that component. If you want to use these commands, you must start WLST from the ORACLE_HOME/common/bin directory of that specific product. Since this book is covering Fusion Middleware and not Fusion Applications, Enterprise Manager Middleware Control is described as the utility to stop, start, and monitor the WLS instances and other components in the farm. Note that for Oracle Applications, the EM utility is EM Fusion Applications Control, which incorporates all the features of EM Middleware Control.

I learned that the metrics data viewed in EM Middleware Control is not "persisted;" i.e., it isn't stored and is lost when the Admin Server for a WLS domain is restarted. In this chapter, the Use Case continues by extending the existing WLS domain to include a second managed server for Service-Oriented Architecture (SOA). A web front end is also added, and Oracle Access Manager (OAM) is integrated into the environment to provide Single Sign-On (SSO) functionality.

Chapter 4—Oracle Fusion Middleware Platform Security Services and Identity Management

I find the Oracle Identity Management product suite to be very confusing. There are so many products with similar acronyms, and some of the products provide overlapping capabilities. This chapter helped me understand what Oracle Internet Directory (OID), Oracle Access Manager (OAM) and Oracle Identity Manager (OIM) actually do in Fusion Middleware, and, by extension, what they do in Fusion Applications. The discussion of Oracle Platform Security Services was also helpful.

I learned that OID is Oracle's implementation of LDAP and that OAM uses LDAP to store user data and also uses a database backend to store policy and user session data. The way OIM works is discussed as well.

The Use Case is extended by adding OID (LDAP) to the environment and integrating it with OAM for SSO. What we have here is not enough acronyms!

Chapter 5—Oracle Service-Oriented Architecture Suite

Service-Oriented Architecture (SOA) is something I have not had any experience with. Therefore, the discussion of SOA in this chapter helped me a lot. The specific reasons that SOA is wanted and needed are covered as well as the components involved. Then the concept of a composite application is introduced. I found the following to be amusing: when the state of a composite is stored in the database for later use and is then retrieved, we say the composite has been "hydrated" and the database is the "hydration store." It sounds silly, but apparently this is a serious term. So many new things to learn about Fusion and J2EE applications!

The process of deploying a SOA composite is explained, and I learned another new term, "upsert," which means to insert into a table if the row does not exist and to update the row if it does exist. While not directly related to Fusion Middleware or Fusion Applications, this is another example of what I learned from this book.

The Use Case is extended by deploying a SOA composite application.

Chapter 6—Oracle Application Development Framework

This chapter discusses the Application Development Framework (ADF). This includes the Model-View-Controller architecture, consisting of the following layers: the model layer, which manages the behavior and data of the application; the view layer, which is the user interface; and the controller layer, which handles input and navigation, and integrates the model and view layers. The administration of ADF applications is covered and the Use Case is extended by deploying an ADF application.

Chapter 7—Oracle WebCenter

WebCenter is Oracle's product to develop portals and dashboards to present information and services from multiple applications. The main development tool is the WebCenter Composer. The various WebCenter services are covered and the Use Case is extended by building a WebCenter Portal application.

Chapter 8—Deploying Fusion Middleware Enterprise Applications

Up to this point, the Use Case has been building the "Pricing Application," and here a detailed review is made of all the steps needed to create the environment and deploy the application. The managed server is extended to an additional node within its WLS cluster

Chapter 9—Securing Fusion Middleware Enterprise Applications

The discussion in this chapter covers the detailed steps to secure the Fusion Middleware environment. This starts with the physical host and includes securing accounts and file-system access. The two types of keystore needed by Fusion Middleware (and by extension, needed by Fusion Applications) are explained. The J2EE applications use a Java keystore to store passwords and other credentials. The other components of Fusion Middleware, such as the Oracle HTTP Server and OID, use the Oracle wallet. The management of both kinds of keystore is covered. This is another example of how I benefitted from reading all three of these books related to various aspects of Fusion Applications. In this chapter I learned why both types of keystore are needed. How to secure web services is also explained.

Chapter 10—Optimizing Fusion Middleware Enterprise Applications

This is the chapter I got the most out of. The focus is on middleware performance issues, not the entire environment. This means the performance of the hardware and the operating system are not discussed. The development and design of an application are identified as large contributors to performance issues. As a support person, I can agree, but by the time the performance issues get to me, the development and design decisions are long gone.

The optimization (or performance) of Fusion Middleware Applications is broken down into two main topics: the Java Virtual Machine (JVM) and the WebLogic Server (WLS). Starting with the JVM, the reasons behind the two versions of the Java Development Kit (JDK) that come with Fusion Middleware are explained. I've always been confused by this. The two versions are the Oracle Java HotSpot VM that came from Sun and the JRockit VM that came from BEA. The HotSpot VM was part of the original Java language and is widely used. The JRockit VM was developed specifically for enterprise application servers.

The optimization of JVM performance is further broken down into two parts: processing Java programs and memory management. For processing Java programs, the optimization involves adaptive bytecode execution where compiled methods that are used often are cached. Memory management optimization is further split into two issues: garbage collection (GC) and optimal heap size.

While I'd heard about garbage collection, which is the process of reclaiming memory that is no longer being used by a Java program, I didn't know that there were two ways to configure this process. The garbage collection algorithm is explained as well as the two configuration options, concurrent and parallel. For concurrent, some of the CPUs on the server are dedicated to garbage collection all the time. For parallel, all of the CPUs stop processing at the same time and all are dedicated to garbage collection for a small period of time. I find it kind of scary to think that all of the processors in an application just stop processing and go looking for memory to reclaim. I assume this is not the way things get done in the systems flying airplanes!

The discussion of optimal heap size was also a learning experience. It turns out there is no optimal heap size. A larger heap size reduces latency but causes increased startup time. This reminds me of the optimal SGA size: bigger is better, but at some point you spend more time searching through a larger SGA than you save by not needing to move things in and out of the SGA. The recommendation is to set the initial and maximum heap size to be the same value and take the longer startup time. This will result in lower latency after the longer startup time, and most enterprise servers will run for a long time after starting up.

The other main issue to be covered is the optimization of WLS performance. This discussion starts by explaining the difference between WLS development and production mode. When configured for production mode, the WLS server doesn't check for updates to JSP files that would need to be recompiled. While the development mode is great for a development environment where you want WLS to automatically detect and compile any code changes, this would slow down a production environment.

WLS server uses a threading model to process user requests. Each request is assigned to a Work Manager for processing. The Work Manager can be configured to give different priorities to different tasks. Configuring JDBC data sources within WLS is covered next. We are told that there really isn't any way to tell what the optimal number of connections will be; only testing will establish this.

The various performance optimization issues that have been covered are then applied to the Use Case. Finally, the optimization of other Fusion Middleware components is covered. This includes Web Services Manager, SOA Suite, ADF, and WebCenter.

Chapter 11—Monitoring and Diagnosing Fusion Middleware Enterprise Applications

The focus here is mainly on the various log files generated by Fusion Middleware components. It is no surprise that this topic can be broken into two parts: the first is the log files for WLS and the second is the log files for other components.

There is a very good discussion of the actual content of WLS log messages. I found some good trivia in this section. It turns out that the RAW-TIME-VALUE in the WLS log messages does

not include leap seconds! Further evidence, as if any were needed, that it really is later than you think! I found the following statement to be a good observation about Fusion Middleware (and Fusion Applications): "*. . . the sheer number of log messages created within the domains of an enterprise application can sometimes complicate the direct analysis of log files.*" Indeed.

Each WLS domain has a log file that contains messages from all the J2EE components in a farm; this can be a good place to start looking for the cause of problems. We also learn that the standard logging service is provided by Oracle Diagnostics Logging (ODL). Had enough acronyms yet?

The Node Manager can be used to remotely manage and diagnose WLS issues. For example, Node Manager can automatically restart WLS instances that fail. It can also retrieve log information via WLST. The configuration of Node Manager is applied to the Use Case.

Chapter 12—Virtualizing Fusion Middleware Enterprise Applications

This chapter discusses the Oracle Virtual Machine (OVM). This consists of a hypervisor that replaces the operating system. This allows a single physical host computer to support multiple isolated virtual machines, each of which operates as a separate isolated computer.

The architecture of the Use Case is shown after deployment to a virtualized environment.

The virtual versions of JDK and WLS are covered. The next topic is Oracle Exalogic. Similar to Exadata, which is hardware and software specifically optimized for Oracle database, Exalogic is specifically optimized for WebLogic Servers and Fusion Middleware. The chapter concludes with a review of the Exalogic hardware and software.

Conclusion

I'm glad I read this book in addition to the books on WLS administration and managing Fusion Applications. It would have been easy to dismiss this book because I thought I needed to focus only on Fusion Applications. Middleware has become so important to the development and operation of modern enterprise applications that it is critical to give more attention to this area. Fusion Applications is completely dependent on Fusion Middleware. The components of Fusion Middleware such as WebLogic Server are also appearing in other Oracle products, not just Fusion Applications. I would suggest that anyone using Oracle software should become familiar with Fusion Middleware, and this book is a great way to do so. ▲

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.

INTERVIEW *(continued from page 10)*

My management is pushing me to take my databases virtual. I can appreciate the advantages (such as elasticity) of virtualization, but I'm not so sure about performance. Also, I cannot rely on guest O/S statistics. Is it just the Luddite in me? Can you help me push back on my management?

Not long ago EMC IT migrated from large SMPs into a virtualized environment with VMware. The E-Business Suite deployment in that virtualized environment was one of the 10 largest in the world. I think that example is more than sufficient to put to rest this notion that large-scale ERP cannot be handled by virtualized Oracle Database. While I wasn't involved in that project, I am quite proud of the fact that EMC "eats their own dog food"—as the saying goes. EMC has since migrated to SAP (with Oracle Database on the backend), and that environment is totally virtualized as well.

I like the Oracle Database Appliance (ODA) because RAC raises the bar higher than most IT organizations can handle, and ODA lowers it. What would you improve about the ODA if you had a wizard wand?

Oracle Database Appliance has 24 Xeon 5600-series cores and 4 TB (net) preconfigured storage that is triple mirrored so it is extremely limited in IOPS. It really bothers me that the software is so complex that one feels compelled to relegate the power of 24 modern cores to such a limited storage configuration. I am aware that the licensing model on ODA allows for licensing fewer cores than are physically available. That aspect of ODA is very intriguing. Perhaps that shows us how flexible Oracle licensing *should* be since it is clearly quite possible and, after all, ODA is just Oracle Database and OEL.

The storage expansion solution for ODA is NFS (via dNFS). If you know where you're going to end up, why not just start there. Get a good dNFS supporting filer and you're on your way.

Introspection

Is NoCOUG a dinosaur, a venerable relic of the pre-information age that has outlasted its usefulness? Our membership and attendance have been contracting for years, and a particular difficulty is finding volunteers to help with our conferences and publications. What should we do to keep NoCOUG relevant and vibrant for another 25 years? Where is the mother lode of new members (and volunteers?)

I've only spoken at your user group gathering a couple of times so I don't have a lot of context. However, I think attending physical gatherings is becoming too difficult for folks to fit into their schedules. Personally, I think webcasts are quite valuable.

Twenty-five years? My, oh my, your guess is as good as any! For all we know that distant horizon would include a name change for the organization! ▲

Introduction to MongoDB

An excerpt from *The Definitive Guide to MongoDB* by Eelco Plugge, Tim Hawkins and Peter Membre

This chapter is an excerpt from the book The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing *published by Apress, Oct. 2010, ISBN 1430230517;*

Imagine a world where using a database is so simple that you soon forget you're even using it. Imagine a world where speed and scalability *just work*, and there's no need for complicated configuration or setup. Imagine being able to focus only on the task at hand, get things done, and then—just for a change—leave work on time. That might sound a bit fanciful, but MongoDB promises to help you accomplish all these things (and many more).

MongoDB (derived from the word *humongous*) is a relatively new breed of database that has no concept of tables, schemas, SQL, or rows. It doesn't have transactions, ACID compliance, joins, foreign keys, or many of the other features that tend to cause headaches in the early hours of the morning. In short, MongoDB is probably a very different database than what you're used to, especially if you've used a relational database management system (RDBMS) in the past. In fact, you might even be shaking your head in wonder at the lack of so-called "standard" features.

Fear not! In a few moments, you will learn about MongoDB's background, guiding principles, and why the MongoDB team made the design decisions that it did. We'll also take a whistle-stop tour of MongoDB's feature list, providing just enough detail to ensure you'll be completely hooked on this topic for the rest of the book.

We'll start things off by looking at the philosophy and ideas behind the creation of MongoDB, as well as some of the interesting and somewhat controversial design decisions. We'll explore the concept of document-orientated databases, how they fit together, and what their strengths and weaknesses are. We'll also explore JSON and examine how it applies to MongoDB. To wrap things up, we'll step through some of the notable features of MongoDB.

Using the Right Tool for the Right Job

The most important of the philosophies that underpin MongoDB is the notion that one size does not fit all. For many years, traditional SQL databases (MongoDB is a document-orientated database) have been used for storing content of all types. It didn't matter whether the data was a good fit for the relational model (which is used in all RDBMS databases, such as MySQL, PostgresSQL, SQLite, Oracle, MS SQL Server, and so on); the data was stuffed in there, anyway. Part of the reason for this is that, generally speaking, it's much easier (and more secure) to read and write to a database than it is to write to a file system. If you pick up any book that teaches PHP (such as *PHP for Absolute Beginners* (Apress, 2009)) by Jason Lengstorf, you'll probably find that almost right away the database is used to store information, not the file system. It's just so much easier to do things that way. And while using a database as a storage bin works, developers always have to work against the flow. It's usually obvious when we're not using the database the way it was intended; anyone who has ever tried to store information with even slightly complex data, had to set up five tables, and then tried to pull it all together knows what I'm talking about!

The MongoDB team decided that it wasn't going to create another database that tries to do everything for everyone. Instead, the team wanted to create a database that worked with documents rather than rows, was blindingly fast, massively scalable, and easy to use. To do this, the team had to leave some features behind, which means that MongoDB is not an ideal candidate for certain situations. For example, its lack of transaction support means that you wouldn't want to use MongoDB to write an accounting application. That said, MongoDB might be perfect for part of the aforementioned application (such as storing complex data). That's not a problem though because there is no reason why you can't use a traditional RDBMS for the accounting components and MongoDB for the document storage. Such hybrid solutions are quite common, and you can see them in production apps such as Sourceforge.

Once you're comfortable with the idea that MongoDB may not solve all your problems (the coffee-making plug-in is still in development), you will discover that there are certain problems that MongoDB is a perfect fit for resolving, such as analytics (think a realtime Google Analytics for your website) and *complex data structures* (e.g., as blog posts and comments). If you're still not convinced that MongoDB is a serious database tool, feel free to skip ahead to the "Reviewing the Feature List" section, where you will find an impressive list of features for MongoDB.

Note: *The lack of transactions and other traditional database features doesn't mean that MongoDB is unstable or that it cannot be used for managing important data.*

Another key concept behind MongoDB's design: There should always be more than one copy of the database. If a single database should fail, then it can simply be restored from the other servers. Because MongoDB aims to be as fast as possible, it takes some shortcuts that make it more difficult to recover from a crash. The developers believe that most serious crashes are likely to remove an entire computer from service anyway; this means that, even if the database were perfectly

restored, it would still not be usable. Remember: MongoDB does not try to be everything to everyone. But for many things (such as building a web application), MongoDB can be an awesome tool for implementing your solution.

So now you know where MongoDB is coming from. It's not trying to be the best at everything, and it readily acknowledges that it's not for everyone. However, for those who do choose to use it, MongoDB provides a rich document-orientated database that's optimized for speed and scalability. It can also run nearly anywhere you might want to run it. MongoDB's website includes downloads for Linux, the Mac, Windows, and Solaris; it also includes various unofficial versions of the program that enable you to install it on Fedora or CentOS, among other platforms.

MongoDB succeeds at all these goals, and this is why using MongoDB (at least for me) is somewhat dream-like. You don't have to worry about squeezing your data into a table—just put the data together, and then pass it to MongoDB for handling.

Consider this real-world example. A recent application I worked on needed to store a set of eBay search results. There could be any number of results (up to 100 of them), and I needed an easy way to associate the results with the users in my database.

Had I been using MySQL, I would have had to design a table to store the data, write the code to store my results, and then write more code to piece it all back together again. This is a fairly common scenario and one most developers face on a regular basis. Normally, we just get on with it; however, for this project, I was using MongoDB and so things went a bit differently.

Specifically, I added this line of code:

```
request['ebay_results'] = ebay_results_array
collection.save(reqest)
```

In the preceding example, request is my document, ebay_results is the key, and ebay_result_array contains the results from eBay. The second line saves my changes. When I access this document in future, I will have the eBay results in exactly the same format as before. I don't need any SQL; I don't need to perform any conversions; nor do I need to create any new tables or write any special code—MongoDB just worked. It got out of the way, I finished my work early, and I got to go home on time.

Lacking Innate Support for Transactions

Another important design decision by MongoDB developers: The database does not include transactional semantics (the bit that offers guarantees about data consistency and storage). This is a solid tradeoff based on MongoDB's goal of being simple, fast, and scalable. Once you leave those heavyweight features at the door, it becomes much easier to scale horizontally.

Normally with a traditional RDBMS, you improve performance by buying a bigger, more powerful machine. This is scaling vertically but you can only take this so far. Horizontal scaling is where, rather than having one big machine, you have lots of less powerful small machines. Historically, clusters of servers like this were excellent for load balancing websites, but databases had always been a problem due to internal design limitations.

You might think this missing support constitutes a deal breaker; however, many people forget that one of the most popular table types in MySQL (MYISAM) doesn't support transactions, either. This fact hasn't stopped MySQL from becoming the dominant open-source database for well over a decade. As with most things when developing solutions, using MongoDB is going to be a matter of personal choice and whether the tradeoffs fit your project.

Note: *MongoDB offers durability when used in tandem with at least two servers, which is the recommended minimum for production deployments. It is possible to make the master server wait for the replica to confirm receipt of the data before the master server itself confirms the data has been accepted.*

Although single server durability is not guaranteed, this may change in the future and is currently an area of active interest.

Drilling Down on JSON and How It Relates to MongoDB

JSON is more than a great way to exchange data; it's also a nice way to store data. An RDBMS is highly structured, with multiple files (tables) that store the individual pieces. MongoDB, on the other hand, stores everything together in a single document. MongoDB is like JSON in this way, and this model provides a rich and expressive way of storing data. Moreover, JSON effectively describes all the content in a given document, so there is no need to specify the structure of the document in advance. JSON is effectively schemaless because documents can be updated individually or changed independently of any other documents. As an added bonus, JSON also provides excellent performance by keeping all of the related data in one place.

MongoDB doesn't actually use JSON to store the data; rather, it uses an open data format developed by the MongoDB team called BSON (pronounced Bee-Son), which is short for Binary-JSON. For the most part, using BSON instead of JSON doesn't change how you will work with your data. BSON makes MongoDB even faster by making it much easier for a computer to process and search documents. BSON also adds a couple of features that aren't available in standard JSON, including the ability to add types for handling binary data. We'll look at BSON in more depth later in the chapter when we cover the feature list.

The original specification for JSON can be found in RFC 4627, and it was written by Douglas Crockford. JSON allows complex data structures to be represented in a simple, human-readable text format that is generally considered to be much easier to read and understand than XML. Like XML, JSON was envisaged as a way to exchange data between a web client (such as a browser) and web applications. When combined with the rich way that it can describe objects, its simplicity has made it the exchange format of choice for the majority of developers.

You might wonder what is meant here by complex data structures. Historically, data was exchanged using the comma-separated values (CSV) format (indeed, this approach remains very common today). CSV is a simple text format that separates rows with a new line and fields with a comma. For example, a CSV file might look like this:

```
Membrey, Peter, +852 1234 5678
Thielen, Wouter, +81 1234 5678
```

A human can look at this information and see quite quickly what information is being communicated. Or maybe not—is that number in the third column a phone number or a fax number? It might even be the number for a pager. To combat this, CSV files often have a header field, where the first row defines what comes in the file. The following snippet takes the previous example one step further:

```
Surname, Forename, Phone Number
Membrey, Peter, +852 1234 5678
Thielen, Wouter, +81 1234 5678
```

Okay, that's a bit better. But now assume you have more than one phone number. You could add another field for an office phone number, but you face a new set of issues if you want several office phone numbers. And you face yet another set of issues if you also want to incorporate multiple e-mail addresses. Most people have more than one, and these addresses can't usually be neatly defined as either home or work. Suddenly, CSV starts to show its limitations. CSV files are only good for storing data that is flat and doesn't have repeating values. Similarly, it's not uncommon for several CSV files to be provided, each with the separate bits of information. These files are then combined (usually in an RDBMS) to create the whole picture. As an example, a large retail company may receive CSV files from each of its stores at the end of each day. These files must be combined before the company can see how it performed on a given day. This process is not exactly straightforward, and it certainly increases chances of a mistake as the number of required files grows.

XML largely solves this problem, but using XML for most things is a bit like using a sledgehammer to crack a nut: it works, but it feels like overkill. The reason for this: XML is highly extensible. Rather than define a particular data format, XML defines how you define a data format. This can be useful when you need to exchange complex and highly structured data; however, for simple data exchange, this often results in too much work. Indeed, this scenario is the source of the phrase "XML hell."

JSON provides a happy medium. Unlike CSV, it can store structured content; but unlike XML, JSON makes it easy to understand and simple to use. Let's revisit the previous example; however, this time you will use JSON rather than CSV:

```
{
   "forename": "Peter",
   "surname": "Membrey",
   "phone_numbers": [
      "+852 1234 5678",
      "+44 1234 565 555"
   ]
}
```

In the preceding example, each JSON object (or document) contains all the information needed to understand it. If you look at phone_numbers, you can see that you have a list of different numbers. This list can be as large as you want. You could also be more specific about the type of number being recorded, as in this example:

```
{
   "forename": "Peter",
   "surname": "Membrey",
   "numbers": [
      {
         "phone": "+852 1234 5678"
      },
      {
         "fax": "+44 1234 565 555"
      }
   ]
}
```

The preceding example improves on things a bit more. Now you can clearly see what each number is for. JSON is extremely expressive, and, although it's quite easy to write JSON by hand, it is usually generated automatically in software. For example, Python includes a module called simplejson that takes existing Python objects and automatically converts them to JSON. Because JSON is supported and used on so many platforms, it is an ideal choice for exchanging data.

When you add items such as the list of phone numbers, you are actually creating what is known as an *embedded document*. This happens whenever you add complex content such as a list (or *array*, to use the term favored in JSON). Generally speaking, there is also a logical distinction too. For example, a Person document might have several Address documents embedded inside it. Similarly, an Invoice document might have numerous LineItem documents embedded inside it. Of course, the embedded Address document could also have its own embedded document inside it that contains phone numbers, for example.

Whether you choose to embed a particular document is determined when you decide how to store your information. This is usually referred to as *schema design*. It might seem odd to refer to schema design when MongoDB is considered a schemaless database. However, while MongoDB doesn't force you to create a schema or enforce one that you create, you do still need to think about how your data fits together. We'll look at this in more depth in Chapter 3.

Adopting a Non-Relational Approach

Improving performance with a relational database is usually straightforward: you buy a bigger, faster server. And this works great until you reach the point where there isn't a bigger server available to buy. At that point, the only option is to spread out to two servers. This might sound easy, but it is a stumbling block for most databases. For example, neither MySQL nor PostgresSQL can run a single database on two servers, where both servers can both read and write data (this is often referred to as an active/active cluster). And although Oracle can do this with its impressive Real Application Clusters (RAC) architecture, you can expect to take out a mortgage if you want to use that solution—implementing a RAC-based solution requires multiple servers, shared storage, and several software licenses.

You might wonder why having an active/active cluster on two databases is so difficult. When you query your database, the database has to find all the relevant data and link it all together. RDBMS solutions feature many ingenious ways to im-

(continued on page 22)

SQL Plan Management for Performance Testing

by Maris Elsins

Maris Elsins

This article is the third in a series by Pythian experts that is being published as the "Performance Corner" column in the NoCOUG Journal.

SQL Plan Management (SPM) is a feature introduced with Oracle Database 11*g*R1 that effectively addresses the long-standing problem of unpredictable effects caused on the performance of SQL statements by the changes of optimizer statistics, system configuration, and schema objects, or the patching and upgrading of the database software. In many cases the cause of the degraded performance is the different execution plan produced by the optimizer for the same statement, compared to the one used before.

SPM solves the problem by capturing the execution plan history and creating plan baselines that contain all accepted execution plans for the statement; the optimizer is aware of SPM and allows only accepted plans in the baseline to be used for execution of the SQL statement. This mechanism prevents the use of any unverified execution plan, thereby eliminating the unpredictable degradation of the performance. Maria Colgan has explained how SPM works in a series of articles starting here:

https://blogs.oracle.com/optimizer/entry/sql_plan_management_part_1_of_4_creating_sql_plan_baselines.

In addition to the main purpose of SPM, it also can be used in different scenarios, such as preserving the execution plans of SQL statements while upgrading the database (**http://docs.oracle.com/cd/E11882_01/server.112/e23633/preup.htm#UPGRD00232**) or forcing the use of a different execution plan by creating a "fake baseline" (**http://jonathanlewis.wordpress.com/2011/01/12/fake-baselines/**).

But take a look at the problems you may run into if you are about to start using SPM:

http://intermediatesql.com/oracle/oracle-11g-sql-plan-management-the-dark-side-of-spm-part-4/.

The Problem

Testing the application is a complicated process, and sometimes even a tested change causes performance issues after it's implemented in production. There are a lot of reasons why that may happen; for example, the test environment does not accurately represent the production—the configuration, the data, or the optimizer statistics are different. Because of that, the optimizer produces different execution plans in each database. This problem is partially addressed by SPM, but the risk still remains if the implemented change does not allow reproducing any of the SQL plans in the baseline (e.g., an index was dropped) or the change introduces new SQL statements that don't have any baselines yet.

Another possible reason for unexpected behavior after implementation of the change is incomplete testing— only the changed functionality was tested or testing was done using automated testing tools that don't have the test cases for testing the full scope of functionality. In such cases, the testing can miss problems introduced in the parts of the application that were not tested. It is also possible the testing did not miss the change, but it was not noticed because the overall workload in the test system was lower and the issue did not reveal itself.

It is often hard to assess the quality of testing and, as the DBAs are unaware of all of the changes being implemented, it's nearly impossible to estimate the performance impact of the change using tools like AWR or ASH. It is hard to involve DBAs in the active testing process, and usually the involvement stops with troubleshooting of the discovered performance issues. On the other hand, DBAs are usually the first to be called for help when the change causes a severe performance issue in production, but by then it's too late—the business is already suffering.

In this article, I will describe a method for tracking down SQL statements introduced with the application change and SQL statements that have different execution plans compared to ones used in production before the change. This method allows verification of the execution plans and the performance metrics of these queries before installing the change in production. The method also estimates the completeness of the testing and helps to identify the functional areas that are not tested. It also gives an opportunity to involve the DBAs in the testing process—preparing them for the upcoming changes in the production.

The Solution

The information captured in SQL Management Base can be used to assess the quality of testing and the impact of the application changes by comparing the snapshots of DBA_SQL_PLAN_BASELINES taken before and after the testing to

identify the differences.

The DBA_SQL_PLAN_BASELINES columns CREATED, LAST_EXECUTED, ACCEPTED, and REPRODUCED allow tracking the usage and changes of SQL Plan Baselines. Note that the REPRODUCED column was introduced in 11gR2, and as it is needed to assess the impact of the tested changes, the method fully works from 11gR2 onward.

The full process contains the following steps:

- Capture SQL Baselines in production.
- Import SQL Baselines from production into the test database.
- Implement the changes in the test environment.
- Execute performance tests.
- Compare the snapshots of DBA_SQL_PLAN_BASELINES.
- Assess the results.

The following sections of the article will provide details and examples of how to execute each step of the plan.

I have set up two 11.2.0.3 databases named PROD and TEST to demonstrate the process. Hammerora version 2.11 (**http://hammerora.sourceforge.net/**) will be used to generate some workload and to simulate the testing of the application. The demo schema in the PROD database was built using the following TPC-C Build Options: User=TPCC, Default Tablespace=USERS, Number of Warehouses=1, Virtual Users to Build Schema=1.

The new TPCC schema was imported in TEST using data pump over database link, this way simulating the creation of the test database.

```
create database link PROD
connect to admin identified by password
using 'PROD';

impdp SCHEMAS=TPCC NETWORK_LINK=PROD
```

Capturing SQL Plan Baselines

The easiest way to capture SQL Plan Baselines is to set the initialization parameter OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES to TRUE at system level, but be aware of a few potential issues before you do that.

Bug 9910484 causes high performance overhead, excessive redo generation, and unexpected growth of SYSAUX tablespace. The bug is fixed in 11.2.0.3 for all platforms or where the patch 9910484 is available for versions 11.1.0.7 and higher.

If the application does not use bind variables SPM will not work, as it uses the signature of SQL (hash value calculated on normalized SQL text) to look up the present baselines. If bind variables are not used the signatures for the SQL statements will be different. If the SQL Plan Baseline capture is enabled it may also cause excessive growth of SYSAUX tablespace as large amount of statements are added to the SQL Management Base.

There are also other options to create the SQL Baselines, such as loading them from SQL Tuning Sets or collecting them from the Shared SQL Area using procedures in DBMS_SPM package.

For simplicity of this demonstration I'm setting the OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES to TRUE.

```
alter system set optimizer_capture_sql_plan_baselines=true;
```

The application activity in PROD is simulated using the Hammerora default driver script for TPC-C workload and one virtual user.

DBA_SQL_PLAN_BASELINES can be queried to confirm that the SQL Plan Baselines have been captured for the application schema.

```
select count(*) from DBA_SQL_PLAN_BASELINES
where PARSING_SCHEMA_NAME='TPCC';
```

Importing SQL Plan Baselines into the Test Database

Transferring the baselines to another database is a four-step process: create the staging table in the source database, pack SQL baselines into the staging table, copy the staging table to the target database, and unpack baselines from the staging table into the SQL Management Base. However, as the LAST_EXECUTED value in DBMS_SQL_PLAN_MANAGEMENT is updated only once every 6.5 days (it is the intended functionality due to the performance overhead updating it on each execution would cause, as explained in Bug 8864741), an additional step of resetting the values of LAST_EXECUTED has to be done in the staging table before unpacking the baselines; otherwise the LAST_EXECUTED values of the plan baselines may not be updated and the tracking of executed SQL statements will not be accurate. It is also required to purge all baselines in the test database to have a clean look at what is happening during the testing.

The following statements are used to extract the SQL Plan Baselines from the production database:

```
exec DBMS_SPM.CREATE_STGTAB_BASELINE('STAGE_
SPM');

exec :n:=DBMS_SPM.PACK_STGTAB_BASELINE('STAGE_
SPM');
```

The next set of commands prepares the SQL Management Base for testing in the test environment:

```
create table STAGE_SPM as
select * from STAGE_SPM@PROD;

update STAGE_SPM
set LAST_EXECUTED=NULL;
commit;

var n number
begin
  for c in (select distinct sql_handle from dba_sql_plan_baselines)
  loop
    :n:=DBMS_SPM.DROP_SQL_PLAN_BASELINE(c.sql_
handle);
  end loop;
end;
/

exec :n:=DBMS_SPM.UNPACK_STGTAB_BASELINE('STAGE_
SPM');
```

I'm creating the first snapshot from the DBA_SQL_PLAN_BASELINES after the baselines are unpacked; it will represent the production state of the SPM. Additional filters can be

added to limit the scope of the analysis using this method; in this case only enabled baselines for the application schema TPCC are collected.

```
create table SPM_SNAP_P as
select * from DBA_SQL_PLAN_BASELINES
where enabled='YES'
and parsing_schema_name = 'TPCC';
```

Implementing the Changes

This is the time to implement all the planned changes in the test system. For the purpose of the demonstration, the following changes will be done in TEST:

```
DROP INDEX tpcc.orders_i1;

CREATE INDEX tpcc.c_fn ON tpcc.customer(Upper(c_first));

ALTER TABLE tpcc.orders ADD (ocn NUMBER);

CREATE OR REPLACE TRIGGER tpcc.cust_ord_counter
BEFORE INSERT ON tpcc.orders FOR EACH ROW
BEGIN
   SELECT Count(*) INTO :new.ocn
   FROM tpcc.orders WHERE o_c_id = :new.o_c_id;
END;
/
```

One of the Hammerora files was also modified to simulate a situation in which a query is not executed during the testing.

```
sed -r -i 's,max\(w_id\),max\(w_id\) mwhid,g' \
  ./hora-components/hora_tpcc.tcl
```

Running the Performance Tests

Any method of testing can be used. The tests can be performed manually or automated tools executing predefined unit tests or load testing tools can be used, or it can be a mix of test methods. It is important to remember that the purpose of the method is to assess the quality of the testing and the overall impact of the introduced changes, so the target should be testing the full functionality of the application. For the best results, each of the test cases should be executed twice—otherwise the SQL Plan Baselines are not captured for the new statements. Keeping this in mind, the method may seem more appealing in cases where automated testing tools are available or the application is undergoing a bigger change, which would require complete functionality testing.

Parameters OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES and OPTIMIZER_USE_SQL_PLAN_BASELINES have to be set to true before the testing is started.

```
alter system set optimizer_capture_sql_plan_baselines=true;
alter system set optimizer_use_sql_plan_baselines=true;
```

For the simulation of testing I'm running Hammerora on the TEST database the same way it was done in PROD. When the testing completes, another snapshot from DBA_SQL_PLAN_BASELINES needs to be created.

```
create table SPM_SNAP_T as
select * from DBA_SQL_PLAN_BASELINES
where enabled='YES'
and parsing_schema_name = 'TPCC';
```

Assessing the Results

Based on the snapshots representing two states of SPM—the production state and the testing state—comparison can be done to observe the differences and address following questions:

- **Completeness of the testing:** The expectation is to test full functionality of the application to make sure that none of the possible changes are missed during testing. This query provides a list of statements that were not executed during testing. It should be reviewed to find out if the queries were removed from the application because of the implemented changes or if they were listed because the testing had missed some part of the functionality. The number of queries returned can be compared to the total number of statements in the SQL Management Base to assess the quality of testing and the level of risk; e.g., if the total is 1000 statements and 400 of them were not tested, it might be worth checking to determine why the difference is so big.

```
SELECT sql_handle, sql_text
FROM SPM_SNAP_P P
WHERE NOT EXISTS (
  SELECT 1 FROM SPM_SNAP_T T
  WHERE p.signature = t.signature
  AND (t.last_executed IS NOT NULL or t.created > p.created)
);
```

- **SQL statements introduced by the implemented change:** Having a list of queries that were executed during the testing but were not executed in production provides the opportunity to check the performance of the new SQL statements using the information available in AWR by looking at the statements that are related to the application change. For the best results the new baselines created for these queries should be imported into production together with the application change to ensure that the exact same tested execution plans are used.

```
SELECT sql_handle ,sql_text
FROM SPM_SNAP_T T
WHERE NOT EXISTS (
  SELECT 1 FROM SPM_SNAP_P P
  WHERE T.signature = P.signature
);
```

- **Non-reproducible execution plans:** This is the list of statements in which the performance is definitely affected by the implemented change, as the previous execution plan is no longer reproducible. The execution plan used is different even if SPM is used, so make sure the performance of the new execution plan is acceptable.

```
SELECT sql_handle, plan_name, sql_text
FROM SPM_SNAP_T T
WHERE T.reproduced = 'NO'
AND T.accepted='YES'
AND EXISTS (
  SELECT 1 FROM SPM_SNAP_P P
  WHERE T.signature = P.signature
  AND P.plan_name=T.plan_name
  and P.reproduced = 'YES'
);
```

- **New execution plans:** This is the list of all queries for which a new execution plan was found and added to the plan history during the testing. As the SPM is in use, the new execution plan may not have been actually used during the testing; therefore, the baselines for these queries should be evolved, and if the new plans perform better than the old ones, they should be imported into production together with the change to use the best possible execution plan immediately after the installation of the change.

```
SELECT sql_handle, plan_name, dbms_lob.Substr(sql_
text,80,1)
FROM spm_snap_t t
WHERE EXISTS (
  SELECT 1 FROM spm_snap_p p
  WHERE T.signature = P.signature
)
AND NOT EXISTS (
  SELECT 1 FROM spm_snap_p p
  WHERE P.signature = T.signature
  AND P.plan_name = T.plan_name
);
```

Calibrating the Test Database

Configuration differences between production and test environments can severely affect the results of testing. The general recommendation is to have the test environment exactly the same as production: the same hardware, the same architecture, the same configuration parameters, the same version of software, the same optimizer statistics, etc. In reality this is rarely possible, so how do we know if the results observed during the testing are introduced by the application change or by the fact that the test environment differs from production?

The question can be answered by running an additional round of "calibration" testing, done exactly the same way as described above but just before the planned changes are installed in the test environment. In this case we would have three snapshots of DBA_SQL_PLAN_BASELINES: the production snapshot, the calibration run snapshot, and the test run snapshot. Comparing the production snapshot to the calibration run snapshot reveals the impact of the differences between the production and the test systems. Comparing the calibration run snapshot to the test run snapshot will show the impact of the implemented changes.

Summary

SQL Plan Management is one of the greatest new performance features introduced in 11*g*. It not only does what it has been built for, but it also offers a lot of versatility, allowing its usage for nonstandard purposes.

In this article I have described a method for assessing the results of testing the changes in the application, configuration, and database software using the information collected in the SQL Management Base. The main idea of this technique is to capture snapshots of DBA_SQL_PLAN_MANAGEMENT and compare them to evaluate the quality of testing and to observe the impact that the installed changes have on the SQL statements and their execution plans. The information obtained is sufficient to be used for checking the execution plans and performance of the affected statements in the test database.

The use of this method allows active involvement of DBAs in the application testing, as a detailed view of the impact that the application change has at the database level becomes available. The tighter cooperation between DBAs, developers, and testers helps to move closer to the common goal of minimizing the risk of negative performance impact after the installation of the change in production. ▲

Maris Elsins is an Oracle Applications DBA currently working at Pythian, the global industry leader in remote database administration services and consulting for Oracle, MySQL, and SQL Server. Maris has extensive experience in troubleshooting and performance tuning of Oracle databases and Oracle E-Business Suite systems; his certifications include: 10g DBA OCM, 9i/10g/11g DBA OCP, and 11i E-Business Suite DBA OCP. He has led or taken part in multiple Oracle Database and Oracle E-Business Suite 11i/R12 implementation, maintenance, migration, and upgrade projects and has been a speaker at multiple UKOUG, LVOUG, and OUGF events.

BOOK EXCERPT *(continued from page 18)*

prove performance, but they all rely on having a complete picture of the data available. And this is where you hit a wall: this approach simply doesn't work when half the data is on another server.

Of course, you might have a small database that simply gets lots of requests, so you just need to share the workload. Unfortunately, here you hit another wall. You need to ensure that data written to the first server is available to the second server. And you face additional issues if updates are made on two separate masters simultaneously. For example, you need to determine which update is the correct one. Another problem you can encounter: someone might query for information on the second server that has just been written to the first server, but that information hasn't been updated yet on the second server. When you consider all these issues, it becomes easy to see why the Oracle solution is so expensive—these problems are extremely hard to address.

MongoDB solves this problem in a very clever way—it avoids it completely. Recall that MongoDB stores data in BSON documents, so the data is self-contained. That is, although similar documents are stored together, individual documents aren't made up of relationships. This means everything you need is all in one place. Because queries in MongoDB look for specific keys and values in a document, this information can be easily spread across as many servers as you have available. Each server checks the content it has and returns the result. This effectively allows almost linear scalability and performance. As an added bonus, it doesn't even require that you take out a new mortgage to pay for this functionality.

Admittedly, MongoDB does not offer *master/master replication*, where two separate servers can both accept write requests. However, it does have sharding, which allows data to split across multiple machines, with each machine responsible for updating different parts of the dataset. The benefit of this design is that, while some solutions allow two master databases, MongoDB can potentially scale to hundreds of machines as easily as it can run on two. ▲

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:

- Amazon Web Services
- Confio Software
- Database Specialists
- 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		
April 1, 2012		**$ 69,706.67**
Revenue		
Membership Dues	1,570.00	
Meeting Fees	150.00	
Vendor Receipts	6,000.00	
Advertising Fee	–	
Training Day	–	
Conference Sponsorship	–	
Interest	3.89	
Paypal balance	–	
Total Revenue		**$ 7,723.89**
Expenses		
Regional Meeting	4,687.50	
Journal	5,135.44	
Membership	56.26	
Administration	1,349.23	
Board Meeting	313.62	
Marketing	–	
Insurance	–	
Vendor expenses	103.30	
Membership S/W subscription	–	
Training Day	250.01	
IOUG-Rep	–	
Server and S/W Fees	635.00	
Bank Service Charges	11.80	
Total Expenses		**$ 12,542.16**
Ending Balance		
June 30, 2012		**$ 64,888.40**

The Last Word

by Dr. Bert Scalzo

In *Oracle on VMware,* Dr. Bert Scalzo makes a case for "solving" performance problems with hardware upgrades. Here's the full quote from Dr. Scalzo's book: "*Person hours cost so much more now than computer hardware even with inexpensive offshore outsourcing. It is now considered a sound business decision these days to throw cheap hardware at problems. It is at least, if not more, cost effective than having the staff [sic] tuned and optimized for the same net effect. Besides, a failed tuning and optimization effort leaves you exactly where you started. At least the hardware upgrade approach results in a faster/better server experiencing the same problem that may still have future value to the business once the fundamental problem is eventually corrected. And, if nothing else, the hardware can be depreciated, whereas the time spent tuning is always just a cost taken off the bottom line. So, with such cheap hardware, it might be a wiser business bet to throw hardware at some solutions sooner than was done in the past. One might go so far as to make an economic principle claim that the opportunity cost of tuning is forgoing cheap upgrades that might fix the issue and also possess intrinsic value. Stated this way, it is a safe bet that is where the business people would vote to spend.*"

We asked a number of performance experts to comment on Dr. Scalzo's suggestion, and they were quite dismissive. To quote one critic: "*I would guess that most of the readers of your magazine would think of themselves as software professionals. To me that title comes with a requirement to deliver quality product, not just hope that hardware will bail you out.*" Strong words indeed, and so we went back to Dr. Scalzo for a rebuttal. Here is his reply.

■ ■ ■

I fault the *NoCOUG Journal* for not making it sufficiently clear both to its readers and to the various commentators that my suggestion was made in the context of database virtualization as implied by the title of my book—*Oracle on VMware*. In the virtual world, DBA commonsense should now include this great new concept: *hardware is simply a dynamic resource.*

I suggest that truly successful DBAs need to work more like doctors in an emergency room triage–type mode—like the TV show MASH. Of course the first rule is to do no harm, but the second rule is to get an acceptable solution in the minimum amount of time. So a tourniquet might be a logical first step on the way to a more permanent solution such as proper surgical correction. When Dr. Winchester first arrived at MASH he had to unlearn/learn how to operate given the cold business realities of working at a front-line hospital. My belief is that DBAs are in the same boat—the cold business realities must rule the day.

The real-world business goal for any database application is to get the job done, not to have the world's most perfect database as measured by lots of techno-geek metrics. So when the business calls because screens are taking too long to process, reports are running slow, the website user sessions are timing out, or batch jobs are missing their must-complete-by times, the DBA has two choices. You could say it will be a while (possibly a long while) while you optimally correct the true underlying problem and just suffer until then or, if your database is on a virtual platform, that you'll immediately apply a tourniquet-type short-term fix to keep them working and later you'll find and fix the real problem. So depending on your virtualization platform, you might live and without downtime relocate the database server to a less-stressed machine; allocate additional resources such as CPU, memory, or network interfaces; or a combo of both. Now the DBA has time to find the real issue or even to choose that virtualization change as a medium-term fix if they feel they have more important things to do at that time. That's the beauty of virtualization—it simply transforms hardware configurations into a software component that can be tweaked much like the Oracle init.ora parameters. It's not a silver bullet to kill any werewolf, but rather a legitimate tool on the successful DBA's tool belt to be used with care and consideration.

In the virtual world, DBA commonsense should now include this great new concept: hardware is simply a dynamic resource.

Note that I am not now nor ever before suggesting that legitimate and proper tuning is to be avoided. I merely suggested that the opportunity cost of doing what's technically ideal at the onset of a problem resolution might not be the best choice from the business perspective—you know, the guys who pay the bills. I do believe that there will be times when adding hardware resources will not fix the problem. But if the cost to try it is 5 minutes with no downtime, why not add it to the list? To ignore this new option is to be a DBA with your head stuck in the sands of the past—where adding hardware was costly and thought to be the hack's solution. But times change and so should the ways DBAs attack problems. Who hasn't patched a flat tire and then decided to just run on that patch rather than replace the tire? So DBA common sense in the virtual world should now include this great new concept: *hardware is simply a dynamic resource.* ▲

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 Summer Conference Schedule

Thursday, August 16, 2012—Chevron, 6101 Bollinger Canyon Road, San Ramon, 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:** Iggy Fernandez, NoCOUG president
9:30–10:30	**Keynote:** *Creating Tests*—Jonathan Lewis
10:30–11:00	**Break**
11:00–12:00	**Parallel Sessions #1**
	Room 1220: *New Strategies for Statistics in 11*g—Jonathan Lewis
	Room 1240: *From Zero to 60 with Oracle Application Express (APEX)*—Brad Brown, InteliVideo
	Room 1150: *Oracle is Listening: Documentation Feedback Session*—Eric Paapanen, Oracle Corp.
12:00–1:00 p.m.	**Lunch**
1:00–2:00	**Parallel Sessions #2**
	Room 1220: *Big Data Analysis Using Oracle R Enterprise*—Vaishnavi Sashikanth, Oracle Corp.
	Room 1240: *A CTO's Perspective on Agile Programming: What It Is and What It Ain't*—Brad Brown, InteliVideo
	Room 1150: *Building Business Intelligence Applications Using MongoDB*—Shafaq Abdullah
2:00–2:30	**Break and Refreshments**
2:30–3:30	**Parallel Sessions #3**
	Room 1220: *To Be Announced*
	Room 1240: *Oracle Advanced Queuing for DBAs and Developers*—Jeff Jacobs, PayPal
	Room 1150: *Is Your Database Stressed? Load Testing for Peak Performance*—Noelle Stimely, UCSF
3:30–4:00	**Raffle**
4:00–5:00	**Parallel Sessions #4**
	Room 1220: *Network Graph Databases, Semantic Modeling, SPARQL Queries and Social Network Analysis*—Dave Abercrombie, Tapjoy
	Room 1240: *To Be Announced*
	Room 1150: *Tips for the Accidental Oracle-on-Linux DBA*—Arijit Das, Naval Postgraduate School
5:00–	**NoCOUG Networking and No-Host Happy Hour at Izzy's Steaks & Chops, 200 Montgomery Street, San Ramon, CA**

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

The *NoCOUG Journal* design and production: Giraffex, Inc., S.F.
Front cover photo: A Young Child Works on a Laptop by Jami Garrison/photos.com.