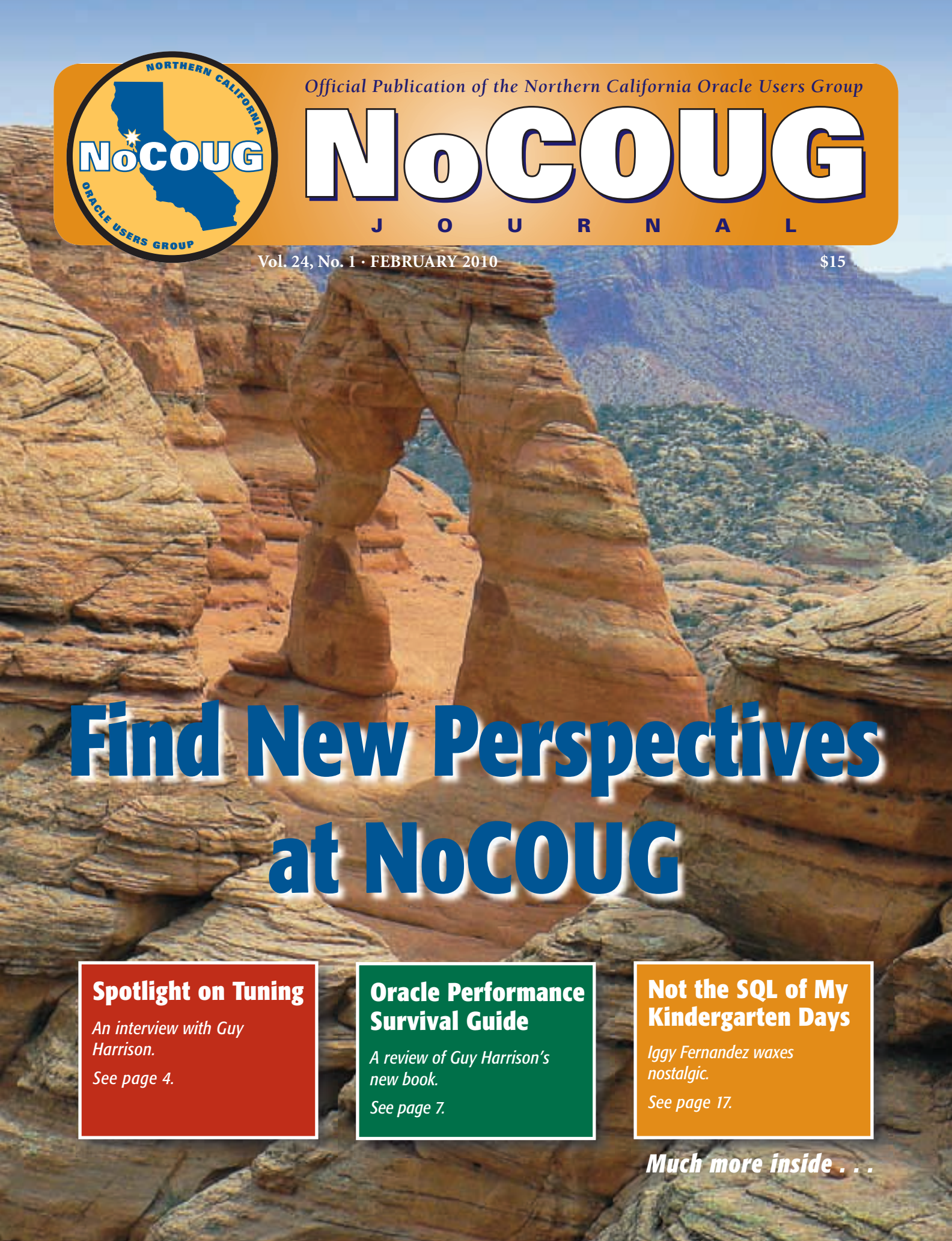
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 24, No. 1 · FEBRUARY 2010 $15



Find New Perspectives at NoCOUG

Spotlight on Tuning

An interview with Guy Harrison.

See page 4.

Oracle Performance Survival Guide

A review of Guy Harrison's new book.

See page 7.

Not the SQL of My Kindergarten Days

Iggy Fernandez waxes nostalgic.

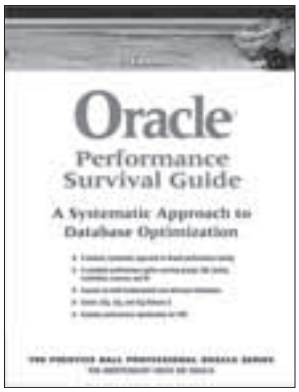
See page 17.

Much more inside . . .

Oracle Performance Survival Guide

A Book Review by Dave Abercrombie

Details



Author: Guy Harrison

ISBN-13: 978-0-13-701195-7

ISBN-10: 0-13-70119-54

Pages: 730

Publication Date: October 2009

Edition: 1

Price: $59.99 (list)

Publisher: Pearson Education (Prentice Hall)

Summary

Overall review: An encyclopedic overview of all aspects of Oracle performance, addressing all layers from high-level application design through SQL tuning, PL/SQL, locks, and contention, all the way down to memory configuration and disk layout.

Target audience: Application developers and DBAs

Would you recommend this to others? Yes.

Who will get the most from this book? Application developers who want to do the right thing, and performance experts or DBAs who want to broaden their knowledge.

Is this book platform specific? Yes: 10*g*, 11*g*, 11*g*R2 (with context from 8*i* and 9*i*)

Why did I obtain this book? Guy Harrison has been a respected author in this field for more than a decade. Back in the days of version 8, his writing opened my eyes to the world of Oracle performance tuning. When I learned that he published this new book—updated and greatly expanded in scope—I was determined to read it.

Overall Review

This book is an encyclopedic overview of all aspects of Oracle performance. Guy Harrison takes a layered approach, starting at the top with application and data model design, where the focus is minimizing the demand for database resources. He then moves down into database code internals, where the focus is maximizing concurrency through reduction of lock, latch, and mutex contention. The next step is to optimize memory usage to minimize the need for physical I/O. Finally, he moves to the bottom layer, where the focus is on optimizing physical I/O at the disk layer.

Each of these layers is worthy of its own book, so to combine all of these topics in a single book is an ambitious goal. Indeed, experienced readers will often want a bit more detail or wonder why their favorite optimization is not mentioned. However, Harrison strikes a very good balance between depth and coverage. He also provides a very useful bibliography, including Oracle documentation, books, and Internet sites and blogs.

For each chapter, Harrison provides extraordinarily clear, concise, and helpful introductions and summaries. He also uses boxed borders to highlight particularly important points within the text. You can learn a great deal by simply reading these summaries and boxed items, and these can also be used to help the reader find relevant sections, which is especially valuable in a book of this length.

Part I: Methods, Concepts, and Tools

In recent years, Oracle performance experts have moved away from "ratio-based" tuning techniques toward use of the wait interface, and this has brought radical improvements in our ability to diagnose and tune. Harrison begins this book by warning us not to rely too much on wait event analysis. He warns that by concentrating only on the largest wait events, we may fall into the trap of treating symptoms rather than causes, we may be tempted to waste money on hardware-based solutions, and we may not reach permanent, scalable solutions.

Harrison's methodical tuning-by-layers approach. which forms the foundation of this book, helps us to avoid these traps. Harrison describes the four layers of an Oracle application: Application, Database, Memory, and Disk. Tuning by layers starts at the top-level **Application** layer, where the goal is to minimize the application workload by tuning application code (optimizing SQL, client-side caching, reducing request rate, etc.) or modifying the physical implementation (indexes, denormalization, partitioning, etc.). Once the application workload has been reduced, we move down to the next layer, **Database**, where the focus shifts to reducing contention and bottlenecks (transaction locks, latches, mutexes, etc.). The next

step is to optimize Oracle's **Memory** usage (block buffer cache, PGA, etc.) to reduce physical I/O. Only after problems in the preceding layers have been addressed does focus shift to the **Disk** layer, where we work to ensure adequate bandwidth and balance I/O demands. Interestingly, Harrison's tuning-by-layers approach does not explicitly cover CPU utilization issues such as the number or speed of CPUs.

Harrison devotes the next chapter to an overview of Oracle's basic architecture. Obviously, this huge topic cannot be thoroughly described in a single chapter, so some subtleties are glossed over and many aspects are greatly simplified. Experts may quibble with some passages, but a concise overview like this is both welcome and essential.

Harrison's next chapter covers the basic tools used for assessing and improving Oracle performance. Although several commercial tools are available for purchase, he focuses on those that come with Oracle. Here, and throughout the book, Harrison does an admirable job of pointing out tools like Active Session History (ASH) and Active Workload Repository (AWR) that are built in and begging to be used, but that require Oracle's extra Diagnostic Pack license for legal use.

Harrison first shows how to use `EXPLAIN PLAN` to generate a SQL execution plan, and he briefly illustrates how to use `DBMS_XPLAN` to view the plan. However, he does not go into detail about how the plan produced by `EXPLAIN PLAN` may vary from the "real" plan nor how to deal with this limitation. Harrison briefly hints at the use of setting `STATISTICS_LEVEL` to `ALL` but he does not illustrate this powerful technique. He misses a wonderful opportunity here to introduce the amazingly useful "tuning by cardinality feedback" approach popularized by Wolfgang Breitling. Harrison also neglects to emphasize the need for representative datasets when tuning: I'm sure we've all seen queries that worked fine in development but were problematic in production due to the lack of test data that adequately resembles production.

Harrison next illustrates basic SQL execution tracing and shows how to format trace files with Oracle's `tkprof`. He also provides tips on finding your trace files, including the use of the nifty `TRACEFILE_IDENTIFIER` session parameter trick. He explains when you might need to merge multiple trace files using Oracle's `trcsess` tool and shows how to use it. Harrison also introduces using `DBMS_MONITOR` to initiate traces by using the end-to-end metrics (aka `DBMS_APPLICATION_INFO`) tags `MODULE` and `ACTION`, although he does not illustrate the many other wonderful benefits of using these session-level tags. Of course, Harrison devotes several pages to interpretation of trace data. He concludes this concise and useful overview of tracing with pointers to `tkprof` alternatives.

Harrison walks us through the use of the `AUTOTRACE` tool with in `SQL*Plus`. This tool combines execution plan information from `DBMS_XPLAN` with a subset of session-level statistics from `V$SESSTAT`. This useful `AUTOTRACE` summary would have been improved, however, through mention of the benefits of querying `V$MYSTAT`, which provides performance details ignored by `AUTOTRACE`.

Harrison spends only a couple of pages summarizing Oracle's `V$` view interface, with an emphasis on the subset that exposes the wait interface. These topics are huge and complicated, with hundreds of `V$` views, wait events, and performance statistics, so he is only able to scratch the surface within the scope of this book. This book is not primarily a troubleshooting book, so it is no surprise that these topics are covered so briefly. However, it is perhaps surprising that no mention is made of Oracle's free STATSPACK or its enhancement `AWR` (which requires a Diagnostic Pack license). Systemwide tools like STATSPACK and friends can be essential in finding and prioritizing problems that can be solved with Harrison's tuning-by-layers approach.

Part II: Application and Database Design

Harrison emphasizes the need for sound database design, since the data model determines the ultimate performance limits of the application. He makes a clear distinction between logical and physical database design. Traditional logical design ensures that "all necessary data is correctly, completely, and unambiguously represented" while generally ignoring performance considerations. Design then proceeds to the physical stage, where performance and scalability goals may cause the logical design to shift. Harrison walks us through many common physical design optimizations, including denormalization, star schemas, and materialized views.

Since the scope of this book allows only one chapter for this overall topic, some details are necessarily skipped. For example, he does not define "primary key" or "foreign key," so the novice could become confused by his discussion of "artificial keys." However, Harrison does an excellent job of pointing out potential risks of denormalization and provides a nice discussion of "vertical" partitioning (moving infrequently used columns of a frequently scanned table into a separate table). Still, Harrison devotes only a few pages to the very powerful techniques of range, hash, and list partitioning. His brief coverage of star schemas does a good job of whetting our appetites, and he kindly provides a reference to Bert Scalzo's book on the topic. Likewise, Harrison's coverage of `PCTFREE`, `PCTUSED`, and `LOB` storage is very brief but clear and concise.

Harrison's overview of B*-Tree indexing is aided by some very helpful graphics. He correctly points out that "creating widely applicable and selective concatenated (i.e., multi-column) indexes should be a top priority of your indexing strategy." However, he devotes only three pages to this topic, and the extraordinarily powerful trick of including enough columns to eliminate the need to read table blocks (i.e., a "covering index") receives only one sentence. In my experience, ignorance of these techniques is one of the biggest contributors to performance troubles, and just a few more pages here would have been very beneficial. However, he provides very helpful advice about finding unused indexes. He also provides a very useful overview of other index issues such as bitmap indexes, DML overhead, index organized tables, clustering, and nested tables. Interestingly, he does not discuss the index clustering factor; this factor is one of the main reasons an index might not be used, and it often contributes to differences in execution plan between test datasets and production. I would have preferred to see more discussion about the importance of data distributions and clustering factors.

Harrison provides a nice overview of application use of bind variables. His advice to use the FORCE_MATCHING_SIGNATURE column of V$SQL to find statements that should be using bind variables was quite welcome. He neatly describes how setting CURSOR_SHARING to FORCE will bring the benefits of bind variables to applications that cannot be rewritten to include them, but he does not warn about the plan instability that commonly arises from bind variables peeking into histograms in such an environment.

Of course, the most efficient SQL statement is one that is not executed, and so Harrison discusses caching, including many potential pitfalls. His overview of the 11*g* feature "client-side result caching" is very helpful.

Harrison's discussion of array fetches is most welcome. He shows how to set the array size within Java application code and claims that it "can provide approximately an order of magnitude improvement for bulk queries."

Harrison provides an excellent overview of isolation levels: read committed (Oracle's default), read only, and serializable. This is combined with an excellent introduction to locking strategies, pessimistic and optimistic, and how they can be implemented. For example, he provides a clear example of using the pseudo-column ORA_ROWSCN and the table DDL keyword ROWDEPENDENCIES to implement optimistic locking strategies. His advice on how to choose a locking strategy is most helpful, since this issue can be quite confusing.

Harrison suggests using PL/SQL-stored procedures to improve performance by reducing network roundtrips. However, this discussion could have benefitted from a few words on other benefits (e.g., more reliable transaction management) and management issues (e.g., dependencies and recompilations).

Part III: SQL and PL/SQL Tuning

Harrison introduces the Oracle optimizer, pointing out that it "makes good decisions across a wide range of circumstances, **but it has not become self-aware yet, and human intervention is often still required**." He neatly summarizes the optimizer's decision-making process. He mentions the static DBA_* views that expose table and column statistics, but curiously does not mention the USER_* versions that are often more available to the typical developer. He very briefly summarizes the optimizer cost calculations: although Jonathan Lewis's work on this topic is recommended in the bibliography, a reminder to the reader here too would have been nice. Harrison's discussion of histograms, 10*g* plan instability from bind variable peeking, and the 11*g* fix for this called "Adaptive Cursor Sharing" is very helpful.

Harrison provides very useful guidance for using DBMS_STAT to gather table, column, and index statistics. His description of stale statistics is very helpful (see the ALL_TAB_MODIFICATIONS view), as is his advice on how to set systemwide defaults. Harrison does a good job of explaining how to gather histogram statistics, including warnings about some of their pitfalls, but he does not explicitly describe how to remove a histogram or why you might need to do so. Harrison neatly discusses "multicolumn extended statistics," but unless you read very carefully, you might not realize that this is an 11*g* feature only.

Harrison reminds us that optimizer hints are actually strict directives, but they can only be obeyed if they arise in the proper context. He advises us that we should use them "only after you have exhausted less direct methods." The scope of this book does not allow a thorough discussion of hints, but Harrison does a good job of summarizing the most important points.

He also does a remarkable job of contrasting outlines and profiles. Outlines are used to guarantee plan stability despite changes in statistics or database configuration, while profiles "are intended to increase optimizer flexibility." Profiles are SQL-specific statistics that "are created by a SQL tuning task and that can then be used by the SQL tuning advisor to determine an optimum plan." Harrison points out many advantages of the SQL tuning advisor: it can spend more time optimizing than the real-time optimizer, it can actually run the SQL to help it optimize, it can tell when indexes are missing, and it can create a profile. Harrison points out that use of profiles requires a Tuning Pack license. He then provides a very nice summary of the new 11*g* feature called "SQL Baselines" that "are intended to supplement SQL profiles and eventually replace stored outlines."

Harrison then goes on to describe how the optimizer decides when to use an index and when to do a full table scan. He goes into a fair amount of useful detail, partially replicating optimizer math, but again seems to ignore CLUSTERING_FACTOR. However, his section on avoiding accidental full table scans is excellent, with relevant advice regarding NOT EQUALS conditions, searching for NULL values, and avoiding accidentally disregarding an index by incorrectly using a function. A useful supplement to this section would have been Tom Kyte's often repeated advice on using a function-based index to quickly search for a needle in a haystack.

Harrison wonderfully points out that an "inexperienced SQL programmer often uses EXPLAIN PLAN to determine that a full table scan has been avoided. If there is no full table scan, the programmer might conclude that the plan is a good one. In fact, there are usually a wide variety of index-based retrievals possible, and merely ensuring that one of these access plans is used does not mean that the SQL is optimized." He then repeats the excellent advice to use a concatenated index, that such an index is optimized if its column order "supports the widest range of queries" and, if possible, that it completely avoids the need for any table access.

Harrison then provides many useful tips on—and possible problems arising from—searching for ranges using the LIKE operator and multivalve single-column lookups. Sometimes we need to do full table scans, so Harrison provides many useful tips for optimizing necessary full table scans. These include lowering the high-water mark, optimizing PCTFREE and PCTUSED, reducing row length, compressing the table, and optimizing use of the block buffer cache. Especially intriguing is the idea of using the SAMPLE clause for queries where approximate answers are acceptable. For example, if you need to calculate an average value of an attribute, a sample of only 5% of the rows may provide a result that is just as useful as the correct answer, but it will run about 20 times faster.

Harrison provides an excellent and widely accessible over-

view of Oracle's join methods: nested loops, sort-merge, and hash. He follows this with a very good description of how the optimizer chooses a join method. The reader is then well prepared to understand how these methods can be optimized, for example by tuning indexes or properly sizing memory. Finally, he provides useful guidance on methods to avoid joins: denormalization, index clusters, materialized views, and bitmap join indexes.

Harrison brings up some interesting facts about special joins. For example, he points out that left and right outer joins will force a join order, which can greatly impact performance. He provides an excellent overview of STAR joins. He briefly discusses hierarchical join methods, but more details would be most welcome here. One very interesting trick he describes is how to use the PARTITION BY feature of analytic functions to avoid expensive correlated subqueries. He also has very interesting advice for anti-joins (finding rows in one table that do not match rows in another table): "you should almost never use a NOT IN sub-query when any of the columns involved is nullable—for this type of query, you either want to define the columns as NOT NULL, or add IS NOT NULL clauses to the WHERE clause."

Harrison provides an excellent overview of Oracle sorting, including optimal, one-pass, and multi-pass sorts. He describes how to estimate memory needs by looking at the TempSpc output from DBMS_XPLAN. He explains how to do a 10032 trace event to obtain sort details. He describes the advantages and disadvantage of indexes used to avoid sorts. Harrison's discussion on sort- and set-related topics is thorough and practical: maximums, minimums, top-N (RANK and DENSE_RANK), and counting rows. He provides a great discussion of GROUP BY, with useful advice that "you should never use a HAVING in place of a WHERE." He has useful advice about UNION and UNION ALL, although his discussion of MINUS and INTERSECT is a little skimpy.

Harrison is clearly a fan of the focused use of PL/SQL to improve performance, and his chapter on it provides an excellent overview of its advantages. He briefly discusses the time model views and V$SQL as ways of assessing PL/SQL resource consumption, but he does not mention other useful instrumentation techniques such as those based on DBMS_APPLICATION_INFO or V$MYSTAT, nor does he address potential dependencies and recompilation issues relevant for maintaining a live production system. Harrison introduces the use of DBMS_PROFILER and its 11*g* enhancement, the hierarchical profiler DBMS_HPROF. He provides a decent overview of BULK COLLECT array processing, pointing out, for example, that the 10*g* complier will default to an array size of 100 for a simple SELECT loop even if you do not code it that way. However, this automatic array processing does not happen with DML, since he suggests to "use the FOREALL statement to perform bulk inserts, updates, deletes and merges." He has charts demonstrating the performance benefit of array processing, but a general discussion of all advantages and disadvantages is lacking. For example, Tom Kyte has written that inserts done with periodic commits generate more redo than single-commit versions, so it would be interesting to have a fuller discussion on this topic.

Harrison provides excellent advice on tuning PL/SQL—insisting, for example, that the SQL first be optimized. He offers plenty of useful tips regarding loop optimization, short circuiting expressions, expression order in IF and CASE statements, and the NOCOPY clause.

Harrison provides an excellent chapter on parallel SQL, starting with an explanation of Oracle's parallel architecture, including its slave pool and its use of direct I/O (bypassing the buffer cache). He gives thorough advice on when to use parallel processing, how to configure it, and how to monitor it (e.g., V$PQ_TQSTAT). His section on optimizing parallel performance wonderfully ties all these concepts together. He concludes this chapter with welcome details on a miscellany of other parallel topics, including RAC and CREATE TABLE AS SELECT.

Harrison includes a wide-ranging chapter on DML tuning. Basic factors include optimization of the WHERE clause and the amount of index maintenance overhead. However, most of his discussion uses logical reads as the main metric, ignoring the amount of REDO or UNDO generated. For example, he does not mention the use of global temporary tables to eliminate REDO for those parts of the application where it is appropriate. Nevertheless, his discussion of the following topics is quite excellent: direct path inserts, multi-table inserts, deletions, update joins (aka "correlated updates"), MERGE, COMMIT (as seldom as possible!) and NOWAIT (use very cautiously!).

Part IV: Minimizing Contention

Harrison starts his section on contention with a very useful summary of transaction locks, including types and modes (e.g. TX "transaction" type in "exclusive" mode X). He continues with a neat summary of lock controls such as FOR UPDATE, SKIP LOCKED, NOWAIT, and WAIT. He mentions use of V$SYSTEM_EVENT to assess the impact of locks, but he suggests instead to use one of his downloadable scripts that takes snapshots and calculates deltas. He, of course, mentions that Oracle's built-in ASH will do this for you automatically, but requires special licensing. Although he does not mention it, the V$EVENT_HISTOGRAM view can also be very helpful too, especially if you compute periodic deltas, as STATSPACK does. Harrison provides several real-time queries that you can use during a transaction pileup. Especially useful is a CONNECT BY query into V$SESSION that shows the lock pileup "tree." This information is so valuable that it has been implemented as V$WAIT_CHAINS in version 11*g*.

Harrison's discussion of latches and mutexes is fairly short. He describes the use of one of his downloadable scripts to take snapshots and compute deltas for latch/mutex waits. He does, however, provide useful explanation and advice for some of the most commonly encountered latch and mutex problems. His bibliography suggests sources for more details on these topics. Harrison is a proponent of altering the hidden parameter _SPIN_COUNT, and he discusses a commercial product that can be used to optimize it. However, he clearly acknowledges the controversial nature of this advice while recognizing its limits: he states that "if the average CPU queue length is approaching or greater than 1, increasing _SPIN_COUNT is unlikely to be effective."

Harrison concludes this section with a brief overview of the buffer cache architecture. He discusses issues surrounding free buffer waits, recovery writer waits, buffer busy waits, and redo log buffer waits. Again, his bibliography suggests sources for more details on these topics.

Part V: Optimizing Memory

Proceeding down into the next layer, Harrison shifts his attention to minimizing physical I/O through optimization of memory allocation. He begins with a short summary of buffer cache principles. This sets the stage for buffer cache configuration and tuning, including dealing with Automatic Shared Memory Management (ASMM), introduced in 10*g*. His coverage of these topics is thorough, helpful, and obviously grounded in practical experience.

Harrison moves next to optimizing PGA memory, which is mostly used for sorts and hashes. The basic goal is to do these operations in memory, avoiding physical I/O. He thoroughly discusses PGA memory management and measuring its usage and efficiency. He provides complete advice on sizing the PGA with `V$PGA_TARGET_ADVICE`. He also does an excellent job of describing how to use the 11*g* Automatic Memory Management (AMM) feature. His discussion of shared pool sizing with `V$SHARED_POOL_ADVICE` is very helpful.

Part VI: I/O Tuning and Clustering

Proceeding down into his fourth and final layer, Harrison shifts his attention to optimizing physical I/O. He begins with an overview of disk I/O concepts and Oracle's I/O architecture. He has extensive and useful advice on measuring and monitoring Oracle I/O, but again neglects mention of STATSPACK.

Regarding optimizing file I/O, he brings up the interesting point that to reduce latency, disks should be run at less than full—perhaps only 50% to 70% full. He also bemoans the fact that disk vendors publish *throughput* metrics at 100% utilization (where *latency* suffers), and publish *latency* metrics at 0% utilization (which is unrealistic).

Harrison provides a nice overview of the various RAID levels. He advises against RAID5 due to its write penalty. Vendors often tout using their write caches to minimize this problem, but Harrison points out that these help only for bursts of write activity, with sustained write activity likely leading to performance degradation.

Harrison provides several useful tips for optimizing physical I/O, especially with the goal of maintaining predictable response time under bursts of activity such as sorts. He strongly suggests dedicated non-RAID5 for redo files.

Harrison moves on to discuss advanced I/O techniques such as Automatic Storage Management (ASM; this is a huge topic—see his bibliography), solid state disks, and Oracle's Exadata Storage Server. He thoroughly discusses changing the database block size from its default 8 KB, but advises against it.

Harrison concludes his book by discussing RAC. He starts with a wonderful overview of its architecture and provides useful advice on measuring cluster overhead. One major goal is to reduce global cache latency. He provides clear, basic advice on how to achieve this goal by optimizing the interconnect, balancing the cluster, and—especially—minimizing global cache requests.

Conclusion

Guy Harrison's new book has a very ambitious scope: it addresses all aspects of Oracle performance, from data model and application design through traditional query tuning and database configuration—all the way down to disk subsystem design. As it takes the reader step by step through these layers, Harrison's book demonstrates the wisdom of tuning the higher layers first before proceeding down into the lower layers. Due to its comprehensive scope, it cannot delve deeply into details at every step. Each chapter could potentially be expanded into its own book. In fact, books have been written around some of these chapters and Harrison kindly refers to them in the bibliography. Therefore, this book is not the last word on SQL tuning, optimizer internals, Oracle troubleshooting, the SGA, or latch contention. However, its coherent approach, useful summaries and highlights, and efficient organization, make *Oracle Performance Survival Guide—A Systematic Approach to Database Optimization* a valuable and essential guide to anyone wishing to expand and deepen their Oracle performance skill set. ▲

"Its coherent approach, useful summaries and highlights, and efficient organization, make [Harrison's book] a valuable and essential guide to anyone wishing to expand and deepen their Oracle performance skill set."

Links

Author's website and blog: **www.guyharrison.net**

Main book website, including scripts and tools: **www.informit.com/store/product.aspx?isbn=0137011954**

Dave Abercrombie has worked at Convio (with a "v," not an "f") for about ten years, having helped to found GetActive Software before its merger with Convio. This company's business model is rather like a distributed denial of service attack against itself. Its customers are nonprofit membership organizations who want to use the Web to engage and activate their members. So each day, Convio sends tens of millions of emails to these members, and then tracks the ensuing member transactions and activities, such as donations, advocacy, and clickthroughs. Dave has honed his troubleshooting and scalability skills by keeping these very busy databases happy. He has presented at Hotsos and is becoming a regular presenter at NoCOUG. He can be reached at **dabercrombie@convio.com**.