Vipin V. Gokhale

Design of the 64-bit Option for the Oracle7 Relational Database Management System

Like most database management systems, the Oracle7 database server uses memory to cache data in disk files and improve the performance. In general, larger memory caches result in better performance. Until recently, the practical limit on the amount of memory the Oracle7 server could use was well under 3 gigabytes on most 32-bit system platforms. Digital Equipment Corporation's combination of the 64-bit Alpha system and the DIGITAL UNIX operating system differentiates itself from the rest of the computer industry by being the first standards-compliant UNIX implementation to support linear 64-bit memory addressing and 64-bit application programming interfaces, allowing high-performance applications to directly access memory in excess of 4 gigabytes. The Oracle7 database server is the first commercial database product in the industry to exploit the performance potential of the very large memory configurations provided by DIGITAL. This paper explores aspects of the design and implementation of the Oracle 64 Bit Option.

Introduction

Historically, the limiting factor for the Oracle7 relational database management system (RDBMS) performance on any given platform has been the amount of computational and I/O resources available on a single node. Although CPUs have become faster by an order of magnitude over the last several years, I/O speeds have not improved commensurately. For instance, the Alpha CPU clock speed alone has increased four times since its introduction; during the same time period, disk access times have improved by a factor of two at best. The overall throughput of database software is critically dependent on the speed of access to data.

To overcome the I/O speed limitation and to maximize performance, the standard Oracle7 database server already utilizes and is optimized for various parallelization techniques in software (e.g., intelligent caching, data prefetching, and parallel query execution) and in hardware (e.g., symmetric multiprocessing [SMP] systems, clusters, and massively parallel processing [MPP] systems). Given the disparity in latency for data access between memory (a few tens of nanoseconds) and disk (a few milliseconds), a common technique for maximizing performance is to minimize disk I/O. Our project originated as an investigation into possible additional performance improvements in the Oracle7 database server in the context of increased memory addressability and execution speed provided by the AlphaServer and DIGITAL UNIX system. Work done as part of this project subsequently became the foundation for product development of the Oracle 64 Bit Option.

Of the memory resource that the Oracle7 database uses, the largest portion is used to cache the most frequently used data blocks. With hardware and operating system support for 64-bit memory addresses, new possibilities have opened up for high-performance application software to take advantage of large memory configurations.

Two of the concepts utilized are hardly new in database development, i.e., improving database server performance by caching more data in memory and improving I/O subsystem throughput by increasing data transfer sizes. However, various conflicting factors contribute to the practical upper bounds on

performance improvement. These factors include CPU architectures; memory addressability; operating system features; cost; and product requirements for portability, compatibility, and time-to-market. An additional design challenge for the Oracle 64 Bit Option project was a requirement for significant performance increases for a broad class of existing database applications that use an open, general-purpose operating system and database software.

This paper provides an overview of the Oracle 64 Bit Option, factors that influenced its design and implementation, and performance implications for some database application areas. In-depth information on Oracle7 RDBMS architecture, administrative commands, and tuning guidelines can be found in the *Oracle7 Server Documentation Set*.[1] Detailed analysis, database server, and application-tuning issues are deferred to the references cited. Overall observations and conclusions from experiments, rather than specific details and data points, are used in this paper except where such data is publicly available.

Oracle 64 Bit Option Goals

The goals for the Oracle 64 Bit Option project were as follows:

- Demonstrate a clearly identifiable performance increase for Oracle7 running on DIGITAL UNIX systems across two commonly used classes of database applications: decision support systems (DSS) and online transaction processing (OLTP).
- Ensure that 64-bit addressability and large memory configurations are the only two control variables that influence overall application performance.
- Break the 1- to 2-GB barrier on the amount of directly accessible memory that can practically be used for typical Oracle7 database cache implementations.
- Add scalability and performance features that complement, rather than replace, current Oracle7 server SMP and cluster offerings.
- Implement all of the above goals without significantly rewriting Oracle7 code or introducing application incompatibilities across any of the other platforms on which the Oracle7 system runs.

Oracle 64 Bit Option Components

Two major components make up the Oracle 64 Bit Option: big Oracle blocks (BOB) and large shared global area (LSGA). They are briefly described in this section.

The BOB component takes advantage of large memory by making individual database blocks larger than those on 32-bit platforms. A database block is a basic unit for I/O and disk space allocation in the Oracle7 RDBMS. Large block sizes mean greater density in the rows per block for the data and indexes, and typically benefit decision-support applications. Large blocks are also useful to applications that require long, contiguous rows, for example, applications that store multimedia data such as images and sound. Rows that span multiple blocks in Oracle7 require proportionately more I/O transactions to read all the pieces, resulting in performance degradation. Most platforms that run the Oracle7 system support a maximum database block size of 8 kilobytes (KB); the DIGITAL UNIX system supports block sizes of up to 32 KB.

The shared global area (SGA) is that area of memory used by Oracle7 processes to hold critical shared data structures such as process state, structured query language (SQL)–level caches, session and transaction states, and redo buffers. The bulk of the SGA in terms of size, however, is the database buffer (or block) cache. Use of the buffer cache means that costly disk I/O is avoided; therefore, the performance of the Oracle7 database server relates directly to the amount of data cached in the buffer cache. LSGA seeks to use as much memory as possible to cache database blocks. Ideally, an entire database can be cached in memory (an "in-memory" database) and avoid almost all I/O during normal operation.

A transaction whose data request is satisfied from the database buffer cache executes an order of magnitude faster than a transaction that must read its data from disk. The difference in performance is a direct consequence of the disparity in access times for main memory and disk storage. A database block found in the buffer cache is termed a "cache hit." A cache miss, in contrast, is the single largest contributor to degradation in transaction latency. Both BOB and LSGA use memory to avoid cache misses. The Oracle7 buffer cache implementation is the same as that of a typical write-back cache. As such, a cache miss, in addition to resulting in a costly disk I/O, can have secondary effects. For instance, one or more of the least recently used buffers may be evicted from the buffer cache if no free buffers are available, and additional I/O transactions may be incurred if the evicted block has been modified since the last time it was read from the disk. Oracle7 buffer cache management algorithms already implement aggressive and intelligent caching schemes and seek to avoid disk I/O. Although cache-miss penalties apply with or without the 64-bit option, "cache thrashing" that results from constrained cache sizes and large data sets can be reduced with the option to the benefit of many existing applications.

The Oracle7 buffer cache is specifically designed and optimized for Oracle's multi-versioning read-consistency transactional model. (Oracle7 buffer cache is independent of the DIGITAL UNIX unified buffer cache, or UBC.) Since Oracle7 can manage its

own buffer cache more effectively than file system buffer caches, it is often recommended that the file system cache size be reduced in favor of a larger Oracle7 buffer cache when the database resides on a file system. Reducing file system cache size also minimizes redundant caching of data at the file system level. For this reason, we rejected early on the obvious design solution of using the DIGITAL UNIX file system as a large cache for taking advantage of large memory configurations—even though it had the appeal of complete transparency and no code changes to the Oracle7 system.

Background and Rationale for Design Decisions

The primary impetus for this project was to evaluate the impact on the Oracle7 database server of emerging 64-bit platforms, such as the AlphaServer system and DIGITAL UNIX operating system. Goals set forth for this project and subsequent design considerations therefore excluded any performance and functionality enhancements in the Oracle7 RDBMS that could not be attributed to the benefits offered by a typical 64-bit platform or otherwise encapsulated within platform-specific layers of the database server code or the operating system itself.

Common areas of potential benefit for a typical 64-bit platform (when compared to its 32-bit counterpart) are (a) increased direct memory addressability, and (b) the potential for configuring systems with greater than 4 GB of memory. As noted above, application performance of the Oracle7 database server depends on whether or not data are found in the database buffer cache. A 64-bit platform provides the opportunity to expand the database buffer cache in Oracle7 to sizes well beyond those of a 32-bit platform. BOB and LSGA reflect the only logical design choices available in Oracle7 to take advantage of this extended addressability and meet the project goals. Implementation of these components focused on ensuring scalability and maximizing the effectiveness of available memory resources.

BOB: Decisions Relevant to On-disk Database Size

Larger database blocks consume proportionately larger amounts of memory when the data contained in those blocks are read from the disk into the database buffer cache. Consequently, the size of the buffer cache itself must be increased if an application requires a greater number of these larger blocks to be cached. For any given size of database buffer cache, Oracle7 database administrators of 32-bit platforms have had to choose between the size of each database block and the number of database blocks that must be in the cache to minimize disk I/O, the choice depending on data access patterns of the applications. Memory available for the database buffer cache is further constrained by the fact that this resource is also shared by many other critical data structures in the SGA besides the buffer cache and the memory needed by the operating system. By eliminating the need to choose between the size of the database blocks and buffer cache, Oracle7 on a 64-bit platform can run a greater application mix without sacrificing performance.

Despite the codependency and the common goal of reducing costly disk I/O, BOB and LSGA address two different dimensions of database scalability: BOB addresses on-disk database size, and the LSGA addresses in-memory database size. Application developers and database administrators have complete flexibility to favor one over the other or to use them in combination.

In Oracle7, the on-disk data structures that locate a row of data in the database use a block-address–byte-offset tuple. The data block address (DBA) is a 32-bit quantity, which is further broken up into file number and block offset within that file. The byte offset within a block is a 16-bit quantity. Although the number of bits in the DBA used for file number and block offset are platform dependent (10 bits for the file number and 22 bits for the block offset is a common format), there exists a theoretical upper limit to the size of an Oracle7 database. With some exceptions, most 32-bit platforms support a maximum data block size of 8 KB, with 2 KB as the default. For example, using a 2-KB block size, the upper limit for the size of the database on DIGITAL UNIX is slightly under 8 terabytes (TB); whereas a 32-KB block size raises that limit to slightly under 128 TB. The ability to support buffer cache sizes well beyond those of 32-bit platforms was a critical prerequisite to enabling larger sized data blocks and consequently larger sized databases. Some 32-bit platforms are also constrained by the fact that each data file cannot exceed a size of 4 GB (especially if the data file is a file system managed object) and therefore may not be able to use all of the available block offset range in the existing DBA format. The largest database size that can be supported in such a case is even smaller. Addressing the perceived limits on the size of an Oracle7 database was an important consideration. Design alternatives that required changes to the layout or an interpretation of DBA format were rejected, at least in this project, because such changes would have introduced incompatibilities in on-disk data structures.

It should be pointed out that on current Alpha processors using an 8-KB page size, a 32-KB data block spans four memory pages, and I/O code paths in the operating system need to lock/unlock four times as many pages when performing an I/O transaction. The larger transfer size also adds to the total time taken to perform an I/O. For instance, four pages of memory that contain the 32-KB data block may not be physically contiguous, and a scatter-gather operation may be required. Although the Oracle7

database supports row-level locking for maximum concurrency in cases where unrelated transactions may be accessing different rows within a given data block, access to the data block is serialized as each individual change (a transaction-level change is broken down into multiple, smaller units of change) is applied to the data block. Larger data blocks accommodate more rows of data and consequently increase the probability of contention at the data block level if applications change (insert, update, delete) data and have a locality of reference. Experiments have shown, however, that this added cost is only marginal relative to the overall performance gains and can be offset easily by carefully tuning the application. Moreover, applications that mostly query the data rather than modify it (e.g., DSS applications) greatly benefit from larger block sizes since in this case access to the data block need not be serialized. Subtle costs such as the ones mentioned above nevertheless help explain why some applications may not necessarily see, for example, a fourfold performance increase when the change is made from an 8-KB block size to a 32-KB block size.

As with Oracle7 implementations on other platforms, database block size with the 64-bit option is determined at database creation time using db_block_size configuration parameter.[1] It cannot be changed dynamically at a later time.

LSGA: Decisions Relevant to In-memory Database Size

The focus for the LSGA effort was to identify and eliminate any constraints in Oracle7 on the sizes to which the database buffer cache could grow. DIGITAL UNIX memory allocation application programming interfaces (APIs) and process address space layout make it fairly straightforward to allocate and manage System V shared memory segments. Although the size of each shared memory segment is limited to a maximum of 2 GB (due to the requirement to comply with UNIX standards), multiple segments can be used to work around this restriction. The memory management layer in Oracle7 code therefore was the initial area of focus. Much of the Oracle7 code is written and architected to make it highly portable across a diverse range of platforms, including memory-constrained 16-bit desktop platforms. A particularly interesting aspect of 16-bit platforms with respect to memory management is that these platforms cannot support contiguous memory allocations beyond 64 KB. Users are forced to resort to a segmented memory model such that each individual segment does not exceed 64 KB in size. Although such restrictions are somewhat constraining (and perhaps irrelevant) for most 32-bit platforms—more so for 64-bit platforms—which can easily handle contiguous memory allocations well in excess of 64 KB, memory management layers in Oracle7 code are designed to be sensitive and cautious about large contiguous memory allocations and would use segmented allocations if the size of the memory allocation request exceeds a platform-dependent threshold. In particular, the size in bytes for each memory allocation request (a platform-dependent value) was assumed to be well under 4 GB, which was a correct assumption for all 32-bit platforms (and even for a 64-bit platform without LSGA). Internal data structures used 32-bit integers to represent the size of a memory allocation request.

For each buffer in the buffer cache, SGA also contains an additional data structure (buffer header) to hold all the metadata associated with that buffer. Although memory for the buffer cache itself was allocated using a special interface into the memory management layer, memory allocation for buffer headers used conventional interfaces. A different allocation scheme was needed to allocate memory for buffer headers. The buffer header is the only major data structure in Oracle7 code whose size requirements are directly dependent on the number of buffers in the buffer cache. Existing memory management interfaces and algorithms used prior to LSGA work were adequate until the number of buffers in the buffer cache exceeded approximately 700,000 (or buffer cache size of approximately 6.5 GB). Minor code changes were necessary in memory management algorithms to accommodate bigger allocation requests possible with existing high-end and future AlphaServer configurations.

The AlphaServer 8400 platform can support memory configurations ranging from 2 to 14 GB, using 2-GB memory modules. Some existing 32-bit platforms can support physical memory configurations that exceed their 4-GB addressing limit by way of segmentation, such that only 4 GB of that memory is directly accessible at any time. Programming complexity associated with such segmented memory models precluded any serious consideration in the design process to extend LSGA work to such platforms. Significantly rewriting the Oracle7 code was specifically identified as a goal not to be pursued by this project. The Alpha processor and DIGITAL UNIX system provides a flat 64-bit virtual address space model to the applications. DIGITAL UNIX extends standard UNIX APIs into a 64-bit programming environment. Our choice of the AlphaServer and DIGITAL UNIX as a development platform for this project was a fairly simple one from a time-to-market perspective because it allowed us to keep code changes to a minimum.

Efficiently managing a buffer cache of, for example, 8 or 10 GB in size was an interesting challenge. More than five million buffers can be accommodated in a 10-GB cache, with a 2-KB block size. That number of buffers is already an order of magnitude greater than what we were able to experiment with prior to the LSGA work. The Oracle7 buffer cache is organized as an associative write-back cache. The mechanism for

locating a data block of interest in this cache is supported by common algorithms and data structures such as hash functions and linked lists. In many cases, traversing critical linked lists is serialized among contending threads of execution to maintain the integrity of the lists themselves and secondary data structures managed by these lists. As a result, the size of such critical lists, for example, has an impact on overall concurrency. The larger buffer count now possible in LSGA configurations had the net effect of reduced concurrency because the size of these lists is proportionately larger. LSGA provided a framework to test contributions from other unrelated projects that addressed such potential bottlenecks to concurrency, as it could realistically simulate relatively more stringent boundary conditions than before.

Scalability Issues

Engineering teams at Oracle have worked very closely with their counterparts in the DIGITAL UNIX operating system group throughout this project. The data collected in the course of the project was useful in analyzing and addressing the scalability issues in the base operating system as well as in the Oracle7 product. Examples of this work are in the base operating system granularity hint regions and in the shared page tables.[2,3]

For every page of physical and virtual memory, an operating system must maintain various data structures such as page tables, data structures to track regions of memory with certain attributes (such as System V shared memory regions, or text and data segments), or data structures that track processes which have references to these memory regions. Ancillary operating system data structures such as page tables grow in size proportionately to the size of physical memory. Changes to page table management associated with System V shared memory regions were made such that processes that mapped the shared memory regions could share page tables in addition to the data pages themselves. Prior to this change, each process mapping the shared memory region used a copy of associated page tables. A change like this reduced physical memory consumption by the operating system. For example, on an Alpha CPU supporting an 8-KB page size, it would take 8 KB in page table entries to map 8 MB of physical memory. For an SGA of 8 GB, it would take 1 MB in page table entries. It is not uncommon in the Oracle7 system for hundreds of processes to connect to the database, and therefore map the 8 GB of SGA. Without shared page tables, 100 such processes would have consumed 100 MB of physical memory by maintaining a per-process copy of page tables.

A granularity hint region is a region of physically contiguous pages of memory that share virtual and physical mappings between all the processes that map them. Such a memory layout allows DIGITAL UNIX to take advantage of the granularity hint feature supported by Alpha processors. Granularity hint bits in a page table entry allow the Alpha CPU to use a single translation look-aside buffer (TLB) entry to map a 512K physical memory space. Using one TLB entry to map larger physical memory has the potential to reduce processor stalls during TLB misses and refills. Also, because of the requirement that the granularity hint region be both virtually and physically contiguous, it is allocated at system startup time and is not subject to normal virtual memory management; for example, it is never paged in or out, and subsequently the cost of a page fault is minimal. Since pages in granularity hint regions are physically contiguous, any I/O done from this region of memory is relatively more efficient because it need not go through the scatter-gather phase.

Summary of Test Results

One of the project goals was to demonstrate clear performance benefits for two common classes of database applications, DSS and OLTP. The Transaction Processing Council (TPC) provides an industry-standard benchmark suite for both applications, that is, TPC-C for OLTP and TPC-D for DSS. An industry-standard benchmark would have been a logical choice for a workload that would demonstrate performance benefits. However, the enormous time, resources, and effort required to stage an audited TPC benchmark and the strict guidelines for any direct comparison of published benchmark results were major factors in the decision to develop a workload for this project that matched the spirit of the TPC benchmark but not necessarily the letter.

In late 1995, Oracle Corporation ran a series of performance tests for a DSS-class workload of the Oracle7 system, with and without the 64-bit option on the AlphaServer 8400 system running the DIGITAL UNIX operating system with 8 GB of physical memory. A detailed report on this test is published and available from Oracle Corporation.[4] These results, shown in Figure 1, clearly demonstrate the benefits of a large amount of physical memory in a configuration with the 64-bit option. A summary of the tests conducted is presented here along with some data points and key observations.

(Readers interested in performance characteristics of an audited industry-standard OLTP benchmark are referred to the *Digital Technical Journal,* Volume 8, Number 3. Two papers present performance characteristics of Oracle7 Parallel Server release 7.3 using 5.0 GB SGA, and a TPC-C workload on a four-node cluster.[5])

The test database consisted of five tables, representing approximately 6 GB of data. The tests included two separate configurations:

- A "standard" configuration with a 128-MB SGA with a 2-KB database block size
- A 64-bit option-enabled configuration with a 7-GB SGA and 32-KB database block size

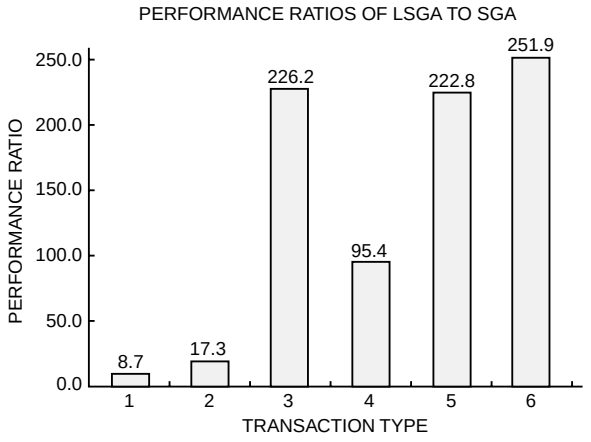

Figure 1
Performance Improvements for a DSS-class Workload, Ratios of LSGA to SGA

The evaluation included running six separate transaction types against these two configurations:

1. Full table scan against a table with 42 million rows (without the Parallel Query Option)
2. Full table scan against a table with 42 million rows (with the Parallel Query Option)
3. Set of ad hoc queries against a table with 42 million rows
4. Set of ad hoc queries involving a join against three tables with 10.5 million, 1.4 million, and 42 million rows, respectively
5. Set of ad hoc queries involving a join against four tables with 1 million, 10.5 million, 1.4 million, and 42 million rows, respectively
6. Set of ad hoc queries involving a join against five tables with 70,000, 1 million, 10.5 million, 1.4 million, and 42 million rows, respectively

Each bar in Figure 1 represents a ratio of execution time (elapsed) between a large SGA (64-bit option) and a small SGA ("standard" configuration) for each of the six transaction types. In every case, the configuration with the 64-bit option enabled consistently outperformed a "standard" configuration. In some cases, the performance increase with the option enabled was over 200 times that of the standard configuration.

The transaction mix chosen for this test represents database operations commonly used in DSS-class applications (e.g., full table scans, sort/merge, and joins). The test also uses a characteristically large data set. Transaction types 1 and 2 are identical except for the use of the Parallel Query Option. The Parallel Query Option in Oracle7 breaks up some database operations such as table scans and sorts/merge into smaller work units, and executes them concurrently. By default, these operations are executed serially, using only one thread of execution. The Parallel Query Option (independent of the 64-bit option) is a standard offering in the Oracle7 database server product since release 7.1. Use of parallel query in this test illustrates the effect of the 64-bit option enhancements on preexisting mechanisms for database performance improvement.

All other things being equal, if the only difference between a standard configuration and a 64-bit-option–enabled configuration is that the entire data set is cached in memory in the latter configuration and that typical times for main memory accesses are a few tens of nanoseconds whereas times for disk accesses are a few milliseconds, only the six to seven times performance increase in transaction type 1 would seem far below expectation. For a full table scan operation, the Oracle7 server is already optimized to use aggressive data prefetch. Before the server begins processing data in a given data block, it launches a read operation for the next block. This technique significantly reduces application-visible disk access latencies by overlapping computation and I/O. Disparity in access time for main memory and disk is still large enough to cause the computation to stall while waiting for the read-ahead I/O to finish. When data is cached in memory, this remaining stall point in the query processing is eliminated.

It is also important to note that a full table scan operation tends to access the disk sequentially. It is typical for disk access times to be better by a factor of at least two in sequential access as compared with random access. Keeping block size and disk and main memory access times the same as before in this equation, a faster Alpha CPU would yield better ratios in this test because it would finish computation proportionately faster and would wait longer for the read-ahead I/O to finish. Follow-on tests with faster CPUs supported this observation. Overlapping computation and I/O as in a table scan operation may not be possible in an index lookup operation. The sequence of operations for accessing a row of data using a B-tree index, in the best case, involves an I/O to read the index block matching the key value first, followed by another I/O to read the data block; a second I/O cannot be launched until the first finishes because the address of the data block to be read can only be determined by examining the contents of the index block read in the previous operation. Unlike table scans, these I/Os are nonsequential. Latencies of the disk I/O for an index lookup, as seen from the application perspective, are consequently greater than latencies for a table scan. Minimizing or eliminating I/Os in the index lookup, therefore, has the potential for even greater increases in speed. Index lookups are typical in OLTP workloads.

The test using transaction type 2 illustrates a cumulative effect because performance benefits for a single thread of execution extend to all the threads when the workload is parallelized.

Unlike full table scans, the sort/merge operation generates intermediate results. Depending on the size of these partial results, they may be stored in main memory if an adequate amount of memory is available; or they may be written back to temporary storage space in the database. The latter operation results in additional I/Os, proportionately more in number as inputs to the sort/merge grow in size or count. The 64-bit option makes it possible to eliminate these I/Os as well, as illustrated in transaction types 4 through 6. Performance improvements are greater as the complexity of queries increases.

Conclusion

The disparity between memory speeds and disk speeds is likely to continue for the foreseeable future. Large memory configurations represent an opportunity to overcome this disparity and to increase application performance by caching a large amount of data in memory. Even though the Oracle 64 Bit Option improves database performance—two orders of magnitude in some cases—specific application characteristics must be evaluated to determine the best means for maximizing overall performance and to balance the significant increase in hardware cost for the large amount of memory. The Oracle 64 Bit Option complements existing Oracle7 features and functionality. The exact extent of the increases in speed with the 64-bit option varies based on the type of database operation. Faster CPUs and denser memory allow for even more performance improvements than have been demonstrated. Factors of importance to new or existing applications, particularly those sensitive to response time, are an order of magnitude performance in terms of speed increases and the ability to utilize memory configurations much larger than previously possible in Oracle7 or for applications that use moderate-size data sets. With sufficient physical memory, the databases used by these response-time–sensitive applications can now be entirely cached in memory, eliminating virtually all disk I/O, which is often a major constraint to response time. In-memory (or fully cached) Oracle7 databases do not compromise transactional integrity in any way; nor do such configurations require special hardware (for example, nonvolatile random access memory [RAM]).

Because a 64-bit AlphaServer and DIGITAL UNIX operating system transparently extends existing 32-bit APIs into a 64-bit programming model, applications can take advantage of added addressability without using specialized APIs or making significant code changes. Performance levels equal to or better than previously possible with specialized hardware and software can now be achieved with industry-standard, open, general-purpose platforms.

Acknowledgments

Many people within several groups and disciplines at both Oracle and DIGITAL have contributed to the success of this project. I would like to thank the following individuals from Oracle: Walter Battistella, Saar Maoz, Jef Kennedy and David Irwin of the DIGITAL System Business Unit; and from DIGITAL: Jim Woodward, Paula Long, Darrell Dunnuck, and Dave Winchell of the DIGITAL UNIX Engineering group. Members of the Computer Systems Division's Performance Group at DIGITAL have also contributed to this project.

References

1. *Oracle7 Server Documentation Set* (Redwood Shores, Calif.: Oracle Corporation).
2. *DIGITAL UNIX V4.0 Release Notes* (Maynard, Mass.: Digital Equipment Corporation, 1996).
3. R. Sites and R. Witek, eds., *Alpha Architecture Reference Manual* (Newton, Mass.: Digital Press, 1995).
4. *Oracle 64 Bit Option Performance Report on Digital UNIX* (Redwood Shores, Calif.: Oracle Corporation, part number C10430, 1996).
5. J. Piantedosi, A. Sathaye, and D. Shakshober, "Performance Measurement of TruCluster Systems under the TPC-C Benchmark," and T. Kawaf, D. Shakshober, and D. Stanley, "Performance Analysis Using Very Large Memory on the 64-bit AlphaServer System," *Digital Technical Journal,* vol. 8, no. 3 (1996): 46–65.

Biography

Vipin V. Gokhale
Vipin Gokhale is a Consulting Software Engineer at Oracle Corporation in the DIGITAL System Business Unit where he has contributed to porting, optimization, and platform-specific features and functionality extensions to Oracle's database server on DIGITAL's operating systems and servers. He was responsible for delivering the first Oracle7 port to the DIGITAL UNIX platform. Prior to joining Oracle in 1990, Vipin was a Senior Software Engineer in India, developing telecommunications software. He received a B.Tech. in Electronics and Telecommunications from the Institute of Technology, Banaras Hindu University, India, in 1985.