# VLM Capabilities of the Sybase System 11 SQL Server

**T.K. Rengarajan**
**Maxwell Berenson**
**Ganesan Gopal**
**Bruce McCready**
**Sapan Panigrahi**
**Srikant Subramaniam**
**Marc B. Sugiyama**

**Software applications must be enhanced to take advantage of very large memory (VLM) system capabilities. The System 11 SQL Server from Sybase, Inc. has expanded the semantics of database tables for better use of memory on DIGITAL 64-bit Alpha microprocessor-based systems. Database memory management for the Sybase System 11 SQL Server includes the ability to partition the physical memory available to database buffers into multiple caches and subdivide the named caches into multiple buffer pools for various I/O sizes. The database management system can bind a database or one table in a database to any cache. A new facility on the SQL Server engine provides nonintrusive checkpoints in a VLM system.**

The advent of the System 11 SQL Server from Sybase, Inc. coincided with the widespread availability and use of very large memory (VLM) technology on DIGITAL's Alpha microprocessor-based computer systems. Technological features of the System 11 SQL Server were used to achieve record results of 14,176 transactions-per-minute C (tpmC) at $198/tpmC on the DIGITAL AlphaServer 8400 server product.[1] One of these features, the Logical Memory Manager, provides the ability to fine-tune memory management. It is the first step in exploiting the semantics of database tables for better use of memory in VLM systems. To partition memory, a database administrator (DBA) creates multiple named buffer caches. The DBA then subdivides each named cache into multiple buffer pools for various I/O sizes. The DBA can bind a database or one table in a database to any cache. A new thread in the SQL Server engine, called the Housekeeper, uses idle cycles to provide free (nonintrusive) checkpoints in a large memory system.

In this paper, we briefly discuss VLM technology. Then we describe the capabilities of the Sybase System 11 SQL Server that address the issues of fast access, checkpoint, and recovery of VLM systems, namely, the Logical Memory Manager, a VLM query optimizer, the Housekeeper, and fuzzy checkpoint.

## VLM Technology

The term very large memory is subjective, and its widespread meaning changes with time. By VLM, we mean systems with more than 4 gigabytes (GB) of memory. In late 1996, personal computer servers with 4 GB of memory appeared in the marketplace. At $10 per megabyte (MB), 4 GB of memory becomes affordable ($40,000) at the departmental level for corporations. We expect that most of the mid-range and high-end systems will be built with more memory in 1997. Growth in the amount of system memory is an ongoing trend. Growth beyond 4 GB, however, is a significant expansion; 32-bit systems run out of memory after 4 GB.

DIGITAL developed 64-bit computing with its Alpha line of microprocessors. Digital is now

well-positioned to facilitate the transition from 32-bit to 64-bit systems. Sybase, Inc. provided one of the first relational database management systems to use VLM technology. The Sybase System 11 SQL Server provides full, native support of 64-bit Alpha microprocessors and the 64-bit DIGITAL UNIX operating system. DIGITAL UNIX is the first operating system to provide a 64-bit address space for all processes. The System 11 SQL Server uses this large address space primarily to cache large portions of the database in memory.

VLM technology is appropriate for use with applications that have stringent response time requirements. With these applications, for example, call-routing, it becomes necessary to fit the entire database in memory.[2,3] The use of VLM systems can also be beneficial when the price/performance is improved by adding more memory.[4]

## Main Memory Database Systems

The widespread availability of VLM systems raises the possibility of building main memory database (MMDB) systems. Several techniques to improve the performance of MMDB systems have been discussed in the database literature. Reference 5 provides an excellent, detailed survey. We provide a brief discussion in this section.

Lock contention is low in MMDB systems since the data resides in memory. Hence, the granularity of concurrency control can be increased to minimize the overhead of lock operations. The lock manager data structures can be combined with the database objects to reduce memory usage. Specialized, stable memory hardware can be used to minimize latency of logging. Early release of transaction locks and group commit during commit processing can be used to increase concurrency and throughput. Since random access is fast in MMDBs, access methods can be developed with no key values in the index but only pointers to data rows in memory.[6] Query optimizers need to consider CPU costs, not I/O costs, when comparing various alternative plans for a query. In an MMDB, checkpointing and failure recovery are the only reasons for performing disk operations. A checkpoint process can be made "fuzzy" with low impact on transaction throughput. After a system failure, incremental recovery processing allows transaction processing to resume before the recovery is complete.[7]

As memory sizes increase with VLM systems, database sizes are also increasing. In general, we expect that databases will not fit in memory in the next decade. Therefore, for most of the databases, MMDB techniques can be exploited only for those parts of the database that do fit in memory.[5]

In addition to the capability of caching the entire database in buffers, the Sybase System 11 SQL Server provides technological advances that take advantage of VLM systems. These are the Logical Memory Manager, VLM query optimization, the Housekeeper thread, and fuzzy checkpoints. We discuss the significance of these advances in the remaining sections of this paper.

## Logical Memory Manager

The Sybase SQL Server consists of several DIGITAL UNIX processes, called engines. The DBA configures the number of engines. As shown in Figure 1, each engine is permanently dedicated to one CPU of a symmetric multiprocessing (SMP) machine. The Sybase engines share virtual memory, which has been sized to include the SQL Server executable. The virtual memory is locked to physical memory. As a result, there is never any operating system paging for the Sybase memory. This shared memory region also uses large operating system pages to minimize translation lookaside buffer (TLB) entries for the CPU.[8] The shared memory holds the database buffers, stored procedure cache, sort buffers, and other dynamic memory. This memory is managed exclusively by the SQL Server. One SQL Server usually processes transactions on multiple databases. Each database has its own log. Transactions can span databases using two-phase commit. For further details on the SQL Server architecture, please see reference 9.

The Logical Memory Manager (LMM) provides the ability for a DBA to partition the physical memory available to database buffers. The DBA can partition the memory used for the database buffers into multiple caches. The DBA needs to specify a size and a name for each cache. After all named caches have been defined, the system defines the remaining memory as the default cache. Once the DBA partitions the memory, it can then bind database entities to a particular cache. The database entity is one of the following: an

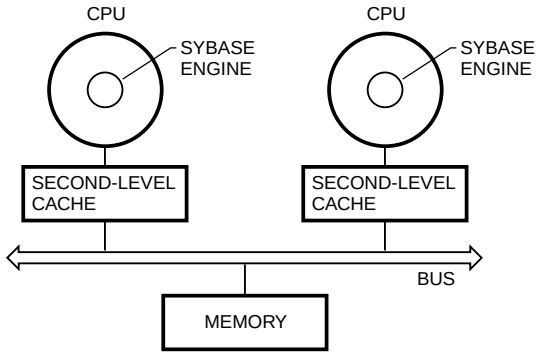


**Figure 1**
SQL Server on an SMP System

entire database, one table in a database, or one index on one table in a database. There is no limit to the number of such entities that can be bound to a cache. This cache binding directs the SQL Server to use only that cache for the pages that belong to the entity. Thus, the DBA can bind a small database to one cache. In a VLM system, if the cache were sized to be larger than the database, an MMDB would result.

Figure 2 shows the table bindings to named caches with the LMM. The procedure cache is used only for keeping compiled stored procedures in memory and is shown for completeness. The item cache is a small cache of 1 GB in size and is used for storing a small read-only table (item) in memory. The default cache holds the remaining tables. Figure 2 shows one table bound to the item cache and the other tables bound to the default cache. By being able to partition the use of memory for the item table separately, the SQL Server is now able to take advantage of MMDB techniques for only the item cache.

Each named cache can be larger than 4 GB. The size is limited only by the amount of memory present in the system. Although we do not expect such a need, it is also possible to have hundreds of named caches; 64-bit pointers are used throughout the SQL Server to address large memory spaces.

The LMM enables the DBA to fine-tune the use of memory. The LMM also allows for the introduction of specific MMDB algorithms in the SQL Server based on the semantics of database entities and the size of named caches. For example, in the future, it becomes possible for a DBA to express the fact that most of one table fits in one named cache in memory, so that SQL Server can use clock buffer replacement.

## VLM Query Optimization

The SQL Server query optimizer computes the cost of query plans in terms of CPU as well as I/O. Both costs are reduced to an estimate of time. Since the number of I/O operations depends on the amount of memory available, the optimizer uses the size of the cache in the cost calculations. With LMM, the optimizer uses the size of the named cache to which a certain table is bound. Therefore, in the case of a database that completely fits in memory in a VLM system, the optimizer choices are made purely on the basis of CPU cost. In particular, the I/O cost is zero, when a table or an index fits in a named cache.

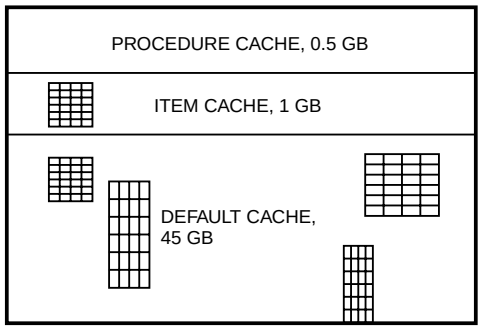


**Figure 2**
Table Bindings to Named Caches with Logical Memory Manager

The Sybase System 11 SQL Server introduced the notion of the fetch-and-discard buffer replacement policy. This strategy indicates that a buffer read from disk will not be used in the near future and hence is a good candidate to be replaced from the cache. The buffer management algorithms leave this buffer close to the least-recently-used end of the buffer chain. In the simplest example, a sequential scan of a table uses this strategy. With VLM, this strategy is turned off if the table can be completely cached in memory. The fetch-and-discard strategy can also be tuned by application developers and DBAs if necessary.

## Housekeeper

One of the motivations for developing VLM was the extremely quick response time requirements for transactions. These environments also require high availability of systems. A key component in achieving high availability is the recovery time. Database systems write dirty pages to disk primarily for page replacement. The checkpoint procedure writes dirty pages to disk to minimize recovery time.

The Sybase System 11 SQL Server introduces a new thread called the Housekeeper that runs only at idle time for the system and does useful work. This thread is the basis for lazy processing in the SQL Server for now and the future. In System 11, the Housekeeper writes dirty pages to disk. At first, it writes pages to disk from the least-recently-used buffer. In this sense, it helps page replacement. In addition to ensuring that there are enough clean buffers, the Housekeeper also attempts to minimize both the checkpoint time and the recovery time. If the system becomes idle at any time during transaction processing, even for a few milliseconds, the Housekeeper keeps the disks (as many as possible) busy by writing dirty pages to disk. It also makes sure that none of the disks is overloaded, thus preventing an undue delay if transaction processing resumes. In the best case, the Housekeeper automatically generates a free checkpoint for the system, thereby reducing the performance impact of the checkpoint during transaction processing. In steady state, the Housekeeper continuously writes dirty pages to disk, while minimizing the number of extra writes incurred by premature writes to disk.[10]

## Checkpoint and Recovery

As the size of memory increases, the following two factors increase as well: (1) the number of writes to disk during the checkpoint and (2) the number of disk I/Os to be done during recovery. The Sybase System 11 SQL Server allows the DBA to tune the amount of buffers that will be kept clean all the time. This is called the wash region. In essence, the wash region represents the amount of memory that is always clean (or strictly, in the process of being written to disk). For example, if the total amount of memory for database buffers is 6 GB and the wash region is 2 GB, then at any time, only 4 GB of memory can be in an updated state (dirty). The ability to tune the wash region reduces the load on the checkpoint procedure, as well as recovery.

The Sybase System 11 SQL Server has implemented a fuzzy checkpoint that allows transactions to proceed even during a checkpoint operation. Transactions are stalled only when they try to update a database page that is being written to disk by the checkpoint. Even in that case, the stall lasts only for the time it takes the disk write to complete. In addition, in the SQL Server, the checkpoint process can keep multiple disks busy by issuing a large number of asynchronous writes one after another. During the time of the checkpoint, the Housekeeper often becomes active due to extra idle time created by the checkpoint. The Housekeeper is self-pacing; it does not swamp the storage system with writes.

## Commit Processing

The SQL Server uses the group commit algorithm to improve throughput.[8,11] The group commit algorithm collects the log records of multiple transactions and writes them to the disk in one I/O. This allows higher transaction throughput due to the amortization of disk I/O costs, as well as committing more and more transactions in each disk write to the log file. The SQL Server does not use a timer, however, to improve the grouping of transactions. Instead, the duration of the previous log I/O is used to collect transactions to be committed in the next batch. The size of the batch is determined by the number of transactions that reach commit processing during one rotation of the log disk. This self-tuning algorithm adapts itself to various speeds of disks. For the same transaction processing system, the grouping occurs more often with slower disks than with faster disks.

Consider, for example, a system performing 1,000 transactions per second. Let us assume the log disk is rated at 7,200 rpm. Each rotation of the disk takes 8 milliseconds. Within this duration, we expect (on the average) 8 transactions to complete, assuming uniform arrival rates at commit point. This indicates a natural grouping of 8 transactions per log write. For the same system, if the log disk is rated at 3,600 rpm, the same calculation yields 16 transactions per log write.

The group commit algorithm used by the SQL Server also takes advantage of disk arrays by initiating multiple asynchronous writes to different members of the disk array. The SQL Server is also able to issue up to 16 kilobytes in one write to a single disk. Together, the group commit algorithm, large writes, and the ability to drive multiple disks in a disk array eliminate the log bottleneck for high-throughput systems.

## Future Work

When a VLM system fails, the large number of database buffers in memory that are dirty need to be recovered. Therefore, database recovery time grows with the size of memory in the VLM system, at least for all database systems that use log-based recovery. In addition, since there are a large number of dirty buffers in memory, the performance impact of checkpoint on transactions also increases with memory size. To minimize the recovery time, one may increase the checkpoint frequency. The checkpoints have a higher impact, however, and need to be done infrequently. These conflicting requirements need to be addressed for VLM systems.

When a database fits in memory, the buffer replacement algorithm can be eliminated. For example, for a single table that fits in one named cache, this optimization can be done with the LMM. In addition, if a table is read-only, it is possible to minimize the synchronization necessary to access the buffers in memory. These optimizations require syntax for the DBA to specify properties (for example, read-only) of tables, as well as properties of named caches (for example, buffer replacement algorithms).

These two areas as well as other MMDB techniques will be explored by the SQL Server developers for incorporation in future releases.

## Summary

The Sybase System 11 SQL Server supports VLM systems built and sold by DIGITAL. The SQL Server can completely cache parts of a database in memory. It can also cache the entire database in memory if the database size is smaller than the amount of memory. System 11 has facilities that address issues of fast access, checkpoint, and recovery of VLM systems; these facilities are the Logical Memory Manager, the VLM query optimizer, the Housekeeper, and fuzzy checkpoint. The SQL Server product achieved

SMP TPC performance of 14,176 tpmC at $198/tpmC on a DIGITAL VLM system. The technology developed in System 11 lays the groundwork for further implementation of MMDB techniques in the SQL Server.

## Acknowledgments

We gratefully acknowledge the various members of the SQL Server development team who contributed to the VLM capabilities described in this paper.

## References and Notes

1. For more information about audited tpmC measurements, see the Transaction Processing Performance Council home page on the World Wide Web, http://www.tpc.org.
2. S.-O. Hvasshovd, O. Torbjornsen, S. Bratsberg, and P. Holager, "The ClustRa Telecom Database: High Availability, High Throughput, and Real-Time Response," *Proceedings of the 21st Very Large Database Conference,* Zurich, Switzerland, 1995.
3. H. Jagadish, D. Lieuwen, R. Rastogi, A. Silberschatz, and S. Sudharshan, "Dali: A High Performance Main Memory Storage Manager," *Proceedings of the 20th Very Large Database Conference Conference,* Santiago, Chile, 1994.
4. M. Heytens, S. Listgarten, M.-A. Neimat, and K. Wilkinson, "Smallbase: A Main-Memory DBMS for High-Performance Applications" (1995).
5. H. Garcia-Molina and K. Salem, "Main Memory Database Systems: An Overview," *IEEE Transactions on Knowledge and Data Engineering,* vol. 4, no. 6 (1992): 509–516.
6. D. Gawlick and D. Kinkade, "Varieties of Concurrency Control in IMS/VS Fast Path," *Database Engineering Bulletin,* vol. 8, no. 2 (1985): 3–10.
7. E. Levy and A. Silberschatz, *Incremental Recovery in Main Memory Database Systems* (University of Texas at Austin, Technical Report TR-92-01, January 1992).
8. J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach,* Second Edition (San Francisco: Morgan Kaufmann Publishers, Inc., 1995).
9. S. Roy and M. Sugiyama, *Sybase Performance Tuning* (Upper Saddle River, N.J.: Prentice Hall Professional Technical Reference, 1996).
10. *Sybase System 11 SQL Server Documentation Set* (Emeryville, Calif.: Sybase, Inc., 1996).
11. P. Spiro, A. Joshi, and T. Rengarajan, "Designing an Optimized Transaction Commit Protocol," *Digital Technical Journal,* vol. 3, no. 1 (Winter 1991): 70–78.

## Biographies

**T.K. Rengarajan**
T. K. Rengarajan has been building high-performance database systems for the past 10 years. He now leads the Server Performance Engineering and Development (SPeeD) Group in SQL Server Engineering at Sybase, Inc. His most recent focus has been System 11 scalability and self-tuning algorithms. Prior to joining Sybase, he contributed to the DEC Rdb system at DIGITAL in the areas of buffer management, high availability, OLTP performance on Alpha systems, and multimedia databases. He holds M.S. degrees in computer-aided design and computer science from the University of Kentucky and the University of Wisconsin, respectively.

**Maxwell Berenson**
Max Berenson is a staff software engineer in the Server Performance Engineering and Development Group in SQL Server Engineering at Sybase, Inc. During his four years at Sybase, Max has developed the Logical Memory Manager for System 11 and has made many buffer manager modifications to improve SMP scalability. Prior to joining Sybase, Max worked at DIGITAL, where he developed a relational database engine.

**Ganesan Gopal**
Ganesan Gopal is a senior member of the Server Performance Engineering and Development Group at Sybase, Inc. He was a member of the team that implemented the Housekeeper in System 11. In addition, he has worked on a number of projects that have enhanced the performance and scaling of the Sybase SQL Server. At present, he is working on a performance feature for an upcoming release. He holds bachelor degrees in advanced physics and in electronics and communication engineering from the Indian Institute of Science, Bangalore, India.

**Bruce McCready**
Bruce McCready is an SQL Server performance engineer in the Server Performance Engineering and Development Group at Sybase, Inc. Bruce received a B.S. in computer science from the University of California at Berkeley in 1989.

**Sapan Panigrahi**
A senior performance engineer, Sapan Panigrahi works in the Server Performance Engineering and Development Group at Sybase, Inc. He was responsible for TPC benchmarks and performance analysis for the Sybase SQL Server.

**Srikant Subramaniam**
A member of the Server Performance Engineering and Development Group at Sybase, Inc., Srikant Subramaniam was involved in the design and implementation of the VLM support in the Sybase SQL Server. He was a member of the team that implemented the Logical Memory Manager in System 11. In addition, he has worked on projects that have enhanced the performance and scaling of the Sybase SQL Server. At present, he is working on performance optimizations for an upcoming release. He holds an M.S. in computer science from the University of Saskatchewan, Canada. His specialty area was the performance of shared-memory multiprocessor systems.

**Marc B. Sugiyama**
Marc Sugiyama is a staff software engineer in the SQL Server Performance Engineering and Development Group at Sybase, Inc. He was the technical lead for the original port of Sybase SQL Server to the DIGITAL Alpha OSF/1 system. He is coauthor of *Sybase Performance Tuning,* published by Prentice Hall, 1996.