WHITE PAPER

Breaking the Disk Barrier with In-Memory DBMS Technology: Sybase Adds a Big Performance Boost for ASE 15.5

Sponsored by: Sybase

Carl W. Olofson

January 2010

IDC OPINION

Over the course of the past 10 years, in-memory database technology has emerged as a key means of boosting performance and scalability and containing storage costs. This technology has evolved from use only for caching, or for extremely high-speed data systems, to much more mainstream IT applications. In-memory DBMS (IMDB) usage is now applied in ways that deliver better performance and respond flexibly to increased user demand at low incremental cost. The reason for this is that the economics of computing have shifted in favor of this approach due to the following factors:

- Cheaper, faster processors in multicore, multiprocessor configurations
- 64-bit memory addressability combined with cheaper memory that makes large in-memory deployment affordable
- Newer DBMS internal data management software that better exploits memory

IN THIS WHITE PAPER

This white paper discusses the growing need for high-performance database management, without disrupting applications and the operations of the datacenter. It considers the ways in which in-memory DBMS technology has emerged and been developed to satisfy this need. It discusses issues of performance versus durability and how an in-memory database strategy can be used to provide better high-speed, high-volume data management without sacrificing data integrity. The document then explores how ASE 15.5 with its in-memory databases capability provides a means of meeting these challenges while operating in a manner that is familiar to Sybase ASE users and fully compatible with existing ASE applications and installations.

METHODOLOGY

In preparing this white paper, IDC has drawn upon many years of research in the area of in-memory DBMS technology — and the products and approaches of numerous vendors over the past 10 years in delivering this technology to the market. IDC has relied upon details from Sybase regarding the in-memory DBMS features of Sybase ASE 15.5, as well as conversations with users of this technology, in order to compile an overview of the product.

SITUATION OVERVIEW

A key requirement of datacenters today is that of managing, or even reducing, the physical footprint and the operational cost of databases while dealing with their continued growth in size and user demand for better performance. The new economics of computing, which derive from large memory models, 64-bit addressability, fast processors, and cheap memory, make it possible to design core database technology that is far faster and more scalable than was possible when the only option was to base data management on spinning disks. Technologies including clustering and virtualization have been employed to achieve scalability, high availability, and data consolidation. Another technology, which had been on the fringes of DBMS deployment until fairly recently, is in-memory DBMS.

Once thought of as useful only for caching, or for specialized computer configurations designed to meet the needs of extremely demanding, very high-performance applications, in-memory DBMS technology has evolved into a capability that can and should be an option for any database workload where performance is a key consideration. Also, because in-memory databases are managed in main memory instead of on disk, the disk is relegated to the role of a recovery, rather than data management, platform. This greatly reduces both the dependency on disk storage and the volume of disk storage required, especially when multiple copies of data are used in distributed architectures.

About In-Memory DBMS Technology

How In-Memory DBMS Works

An in-memory DBMS manages its data in main memory. It optimizes the internal structure on this basis. Indexes contain memory pointers instead of disk page and line references. Data is shifted around according to memory optimization rather than disk optimization strategies. An IMDB is not necessarily less recoverable than disk-based database technology because it may be configured to stream log data to nonvolatile storage, such as disk or solid state memory. Often, however, users choose not to stream a log because even that little bit of I/O activity can slow down the database.

Why In-Memory DBMS Is So Much Faster than Fully Buffered Disk DBMS

An IMDB is typically as much as 10 times faster than a disk-based database, depending on the workload. The reason for this has to do with its internal storage architecture.

A disk-based database is designed from the ground up to optimize its data management based on an optimal disk layout and I/O minimization strategy. For this reason, data is assigned to preallocated spaces on disk that are mapped to files or, if the access method is direct, to designated segments on designated disk volumes. Often, to minimize disk head movement and contention, the data is spread across many volumes. When data is required for a database operation, the DBMS must identify the database page required, determine where that page is stored on disk, access the appropriate volume, retrieve the data, allocate the data to a buffer, and update a page buffer table.

Even when required data is in buffer memory, and its database key (actual physical location) is already known, the system must dereference the database key, determine that the page is in memory, locate the page, and find the referenced location in the buffer where the requested data is located. By contrast, when an IMDB looks for data by its location, it simply uses a memory offset to load a pointer and gets the data directly.

When data is updated on a disk-based system, the buffer is updated and marked as "dirty." A lock is set to prevent another user from reading the old data from disk until the transaction can be committed. When the transaction is committed, a record is written to the log. In some systems, the buffers are flushed at this point. In others, the buffer page table is updated to indicate that requests for that data should be referred to the now committed data in buffer, which is eventually written to disk either when the buffer must be flushed to make way for another database page or when the I/O optimization algorithm finds an idle moment to perform the write. In an IMDB, most of these operations do not exist. A user has a temporary image of updated data until it is committed, at which point the temporary image replaces the permanent image.

In-Memory DBMS and Scalability

A fundamental barrier to scalability for a disk-based database is the bottleneck represented by the storage system. Most database tuning for scalability involves strategies around the distribution of data across disk volumes, the collocation of related data to reduce the possibility of cross-volume queries, the sparse scattering of frequently but randomly accessed data to reduce head contention, and so on. Even with all this, funneling data access through the I/O channels adds unavoidable overhead. Keeping data in memory, and avoiding not only all the I/O activity but also all the work that the DBMS core must do to map data to and from disk, find the right volume for each element of a query or update, and so on, results in a system that not only is much more efficient but also scales with increases in demand and increases in data volume, without requiring special storage tuning efforts by the DBA.

How In-Memory DBMS Saves Storage

A disk-based database usually requires scattering data across volumes to ensure good I/O throughput. Typically, volumes used in this way are kept at about 40–50% utilization to avoid head contention, excessive head movement, and redirection when data is written to disk as the result of an insert operation. Also, for good performance, production databases are generally stored on expensive tier 1 storage.

By contrast, an IMDB typically uses disk only for recovery. It writes the log (if there is one) to disk, and it dumps its memory contents to disk from time to time. Because the disk is not involved in database transactions, the data that is dumped to disk can be packed together. As a result, there is no problem with filling a volume right up to the brim. Also, because the disk has no impact on performance, cheap lower-tier storage may be used. In scale-out situations, where clustered databases use partitioned and redundantly stored data, this savings effect is compounded.

Why IMDB Now?

For decades, technologists have worked on a way to manage a general-purpose database in memory, but until recently, these efforts had been hampered by the economics of computing. Until early in this century, most computers had limited main memory and, in fact, could address only a limited amount of main memory. Main memory was expensive, and processors were not fast enough to make the trade-off between managing data in memory and managing data on disk attractive enough for main memory databases to catch on.

Today, however, processors are orders of magnitude faster than they were just a decade ago. Most systems have multiple processors and multiple cores per processor. Enterprise servers typically use 64-bit memory addressing and are stocked with multiple gigabytes of main memory. All this means that the economics of computing have swung in favor of in-memory databases for many workloads. The leading incumbent DBMS products were developed, and are still largely architected, to support disk-based databases. Although very large buffers provide significant performance benefits, they do not provide the overall power and performance of a true in-memory database.

About Sybase ASE 15.5 In-Memory Database

The Sybase Approach to In-Memory Databases

In order to ensure maximum compatibility between the IMDB and disk-based deployments of ASE, Sybase has implemented IMDB functionality by storage of data on a memory-based virtual disk volume. Internal optimizations enable the DBMS core to manage the relationship between buffers and the virtual volume in a clean and simple way. Still, from the perspective of the user, DBMS operation is the same on a virtual volume as it is on a physical volume so that all existing SQL and T-SQL is compatible with the IMDB. The user codes familiar, conventional SQL DDL to define the database device as a logical disk that resides in a special cache in memory and then allocates all the space required for the database on that device.

In this initial offering of IMDB, Sybase has also routed logging to memory in order to maximize throughput. This means that the database, while retaining the ACID (Atomic, Consistent, Isolated, Durable) properties of a database transaction, is not recoverable in case of system failure. A "relaxed-durability database" (RDDB) allows a measure of recoverability by providing for the data to be dumped to disk upon shutdown if it is meant to be retained. This configuration is discussed in greater detail later in this paper.

Sybase is offering the nondurable IMDB in recognition of the fact that most database applications use the database as a common clearinghouse for transitive data (temporary data used during long data-driven application transactions) that does not require recovery because any such transactions that might fail would need to be restarted anyway. By doing this, Sybase has been able to eliminate a number of internal operations that ensure transaction durability; these operations are necessary for recovery of transactional or reference data (which are the kept products of transactions), but not for transitive data.

If desired, the user can perform minimal logging, which can be used for roll forward operations from a snapshot but lacks the DML detail necessary for full rollback operations. This feature is called "ML-DML" and requires that the database option "select into/bulkcopy/pllsort" be enabled.

However, because the IMDB configuration may be used in combination with a conventional disk-based configuration, the user is advised to use the IMDB to manage the transitive data and the disk-based database for the transactional and reference data. Such combinations still result in substantial performance improvements because a very large amount (or even a majority, for some applications) of database activity involves transitive data.

Suggested Use Cases for ASE In-Memory Database

When should one consider using the ASE IMDB? Some use cases would include:

- Cases where transitive data can be segregated from durable data and allocated to a separate database. These cases would typically involve applications that perform complex, long transactions with many intermediate steps before the final results are stored.

- Cases where applications do a large amount of data reading from a large set of data, but write data to tables other than those being read. In such cases, the read-only data can be replicated from disk to the IMDB, while the updatable data remains in a disk-based database. The reads are then highly optimized.

- Reporting databases containing extracts from production databases. These databases should show spectacular performance improvements over the disk-based alternative, and, of course, they require no disk storage.

- Development and test databases where the primary motive to use the IMDB is to eliminate the need for any disk volumes to be dedicated to development and test.

Relaxed-Durability Database

Some databases that might fit the IMDB model may be too large to fit in memory. Others, while not requiring transaction recoverability, may still require some level of point-in-time recovery. For these use cases, Sybase offers its RDDB feature.

The RDDB is created and managed somewhat like the IMDB, except that it is also allocated disk storage. If the database is too large to fit in memory, the RDDB will balance the data between memory and disk, flushing buffers to disk as necessary. The RDDB may be told to save its in-memory data upon shutdown ("at shutdown durability") or not ("no recovery"), at the user's option.

Folding ASE In-Memory Database into an Existing ASE Installation

Because the IMDB operates in the same way as a disk-based database, except for its actual storage management, it can be mixed with other ASE databases in the same way as any ASE databases might be mixed together, sharing servers and serving the same applications. Users can gradually segregate transitive data from durable data with minimal application disruption. Users can move read-only database workloads to IMDB with no disruption at all.

Sybase supports full replication from disk-based databases to IMDB right now and plans to support IMDB replication to disk-based databases sometime in 2010. Replication allows easier integration of IMDB into database application operations. Sybase is also providing a new T-SQL command called "transfer table" that enables the transfer of committed data that has been modified since the previous "transfer table" was executed from the IMDB to a designated target. This supports the use of the IMDB as a data staging area, transferring results to a permanent disk-based database at the end of a transaction.

Customer Experience

IDC spoke with a Wall Street trading firm that has been working with Sybase ASE IMDB. For this firm, the speed with which trades can be processed is absolutely critical to its business. Delays of even milliseconds in processing the millions of trades it handles daily can cost thousands of dollars due to missed prices. A key cause of such delays is the processing that must take place before a trade can be committed. Such processing includes pretrade compliance checks that determine if the proposed trade violates federal regulations, exchange rules, business rules, or rules governing the securities being traded. These compliance checks are done by looking up data in a variety of tables in a database dedicated to that purpose.

In an effort to speed up this processing, the firm explored a number of IMDB technologies provided by different database software vendors. All the scenarios it considered involved migrating data to an unfamiliar database using tools that were either unfamiliar to the staff or not designed to support that DBMS. While the other options it considered did offer some performance improvement, the firm observed that other products had operational limitations and required special tools, training, and expertise in order to integrate them into its existing trading system.

Then it tried Sybase ASE IMDB. Because the ASE IMDB behaves just like the disk-based Sybase ASE, it was supported by all the regular database replication tools the firm had in-house. It was easy for the firm to replicate data from its Sybase, Oracle, and Microsoft SQL Server databases into the ASE IMDB. Because the firm's DBAs were familiar with ASE, they had no problem using the IMDB and required no special training. Best of all, without any special tuning or schema changes, the pretrading compliance checks, which are driven by previously developed Sybase T-SQL stored procedures, ran an average of nine to 10 times faster than they ever had before and required no code changes to the T-SQL. In addition, IMDB scales better in terms of connection concurrency, which will help achieve better transaction throughput with existing infrastructure.

This trading firm is thrilled with these results and plans to roll ASE IMDB into full production within the first couple of months of 2010. The securities trading business is highly competitive, and speed is the number 1 differentiator. This firm believes that ASE IMDB will give it a significant competitive advantage on Wall Street.

FUTURE OUTLOOK

In-memory DBMS is an important dimension of the DBMS landscape and will become more so in the coming years. The shift in computing economics that make processors and memory abundant will compel every DBMS vendor to move in this direction in the future. It is likely that as solid state memory drops in price, main memory will be seen as the primary "home" of a database, solid state memory will be seen as the overflow area, and disks will be relegated to recovery functions only.

We are not there yet, but the signs are clear. The developments from Sybase that provide transparent in-memory database management show clear leadership in the DBMS field, anticipating a key deployment model for databases in the future.

CHALLENGES/OPPORTUNITIES

Sybase is not the only DBMS vendor exploring in-memory functionality. All the leaders are investing in this area. In order to leverage IMDB as a competitive differentiator, Sybase must continue to expand functionality in this area.

CONCLUSION

The days of basing random-access database management on spinning disks with moving heads are coming to an end. The efficiency, combined with the growing affordability, of shifting this functionality to main memory will compel the database community to move to IMDB in the coming years. IMDB not only is much faster than disk-based database performance but also eliminates the need for the use of tier 1 storage in database deployment and drastically reduces the role of disk storage in database management, which results in dramatically lower equipment and operational costs associated with database storage.

In anticipation of this trend, users should:

- Consider what data they manage today that could be moved to an IMDB
- Plan for the adoption of durable IMDB in the future
- Watch what the major DBMS vendors are doing with respect to IMDB technology in the coming years

- Demand in-memory DBMS technology that they can adopt gradually by integrating it into their existing environments with a minimum of database or application code changes
- Evaluate ASE 15.5 and its IMDB capabilities to determine if they can deliver real benefits in terms of better performance and scalability and storage cost savings today