In Search of Data Durability and High Performance:

Benchmarking In-Memory & On-Disk Databases With Hard-Disk, SSD and Memory-Tier NAND Flash

In-memory database systems (IMDSs) eliminate much of the performance latency associated with traditional on-disk database management systems (DBMSs), but some applications require a higher level of data durability (i.e. an ability to recover if someone "pulls the plug" or otherwise disrupts volatile memory). As a solution, IMDSs offer transaction logging, in which changes to the database are recorded on persistent media. But critics object that logging re-introduces the storage-related latency that builds slowness into on-disk DBMSs.

Will an IMDS with transaction logging outperform a traditional DBMS? Will type of storage – hard disk drive vs. SSD vs. state-of-the-art memory-tier device – affect any performance difference? McObject's benchmark report answers these questions with tests using IMDS and on-disk DBMS technology. It also examines the impact of transaction length (long vs. short) on database system performance and the interplay of storage device type with this effect.

McObject LLC
33309 1st Way South
Suite A-208
Federal Way, WA 98003
Phone: 425-888-8505
E-mail: info@mcobject.com
www.mcobject.com

In the past decade, in-memory database systems (IMDSs) have emerged as the database solution for many real-time applications. In-memory databases work with data entirely in main memory, eliminating the file I/O, cache management, data transfer and other overhead that is hard-wired into traditional disk-based database management systems (DBMSs).

Some applications require data durability beyond what RAM-based storage can provide. As a solution, IMDS vendors offer transaction logging, which keeps a record of changes to the database, enabling recovery in the event of system failure. But this logging requires writing to persistent storage. When these "database writes" are reintroduced via transaction logging, do IMDSs retain their performance advantage?

McObject and Fusion-io collaborated on a benchmark report to answer these questions, with tests comparing on-disk database system performance vs. that of an IMDS with transaction logging. To highlight the effects on performance of today's diverse data storage options, we ran separate tests using hard disk drive (HDD), solid state drive (SSD) and high performance NAND flash memory-tier storage for on-disk database records and the IMDS transaction log.

Test hardware and software

The testing platform consisted of the following off-the-shelf elements:

- **Server**. Dell PowerEdge T110 Tower Server with 4GB of 1333 mhz memory. The original equipment hard-disk storage was not used.

- **Hard Disk Drive**. Western Digital VelociRaptor, 600 GB.

- **Solid State Drive**. SanDisk Extreme Solid State Drive, 240 GB.

- **NAND Flash Memory tier**. Fusion ioDrive2, 1.2 TB, installed via PCI Express within the test server. Fusion ioMemory products differ from SSD storage in that they present flash to the host system as a new memory tier, using the Fusion-io Virtual Storage Layer (VSL) software to integrate flash close to the host CPU, eliminating hardware and software "layers" that otherwise introduce latency by standing between CPU and the flash.

- **In-memory database system**. McObject *eXtreme*DB® IMDS.

- **Disk-based DBMS**. McObject *eXtreme*DB Fusion. With this hybrid DBMS, the developer can combine in-memory and on-disk database storage, or specify entirely on-disk storage. The 100% on-disk storage used in these tests implements caching, file I/O and all other aspects of a disk-based DBMS that affect performance.

Test 1 – Database Operations

Methodology

The test application examined five database operations: record inserts, record updates, record deletes, index searches and table traversals. It measured performance on each operation while looping, with each loop constituting a database transaction and each loop containing at least two instances of the operation, as follows:

Insert records test – 2 record inserts/loop
Update test – 2 record updates/loop
Delete test – 2 record deletes/loop
Index search test – 8 searches/loop
Table traversal test – 8 traversals/loop

Total loops executed for each test ranged from 20,000 to 400,000. The benchmark application recorded the number of loops accomplished per millisecond for each of the two database types (on-disk DBMS and in-memory database system with transaction logging, or "IMDS+TL") with each of the three storage devices (HDD, SSD and ioDrive).

The *e*X*treme*DB database system supports multiple application programming interfaces (APIs) including SQL/ODBC/JDBC and native interfaces for the C/C++, C# (.NET) and Java languages. Code for the operations described above was implemented using the native C/C++ API.

Results

Inserts, Updates & Deletes

The test showed that for insert, update and delete operations, the IMDS performing transaction logging maintained its speed advantage over the traditional on-disk database system. This advantage increased as the IMDS moved from using hard disk, to SSD, and finally to the Fusion ioDrive2 as storage for the transaction log.

Figure 1 shows results in loops/ms for each of the configurations, as well as the performance multiple exhibited by each of the IMDS+TL configurations over the baseline on-disk DBMS with hard disk storage. For example, in the test of database deletes, the IMDS+TL using the solid state drive was 11.87 times faster than the on-disk DBMS using HDD. For database inserts, the IMDS+TL using the Fusion ioDrive2 for storage was 20.05 times faster than the on-disk DBMS using HDD.

Insert	Loops/ms	Perf. Multiple
HDD - On-disk	1.60	1.00
HDD - IMDS+TL	5.11	3.20
SSD - IMDS+TL	15.49	9.69
ioDrive2 - IMDS+TL	32.05	20.05

Update	Loops/ms	Perf. Multiple
HDD - On-disk	3.00	1.00
HDD - IMDS+TL	5.32	1.77
SSD - IMDS+TL	17.30	5.77
ioDrive2 - IMDS+TL	38.25	12.75

Delete	Loops/ms	Perf. Multiple
HDD - On-disk	1.50	1.00
HDD - IMDS+TL	5.31	3.55
SSD - IMDS+TL	17.77	11.87
ioDrive2 - IMDS+TL	34.72	23.19

Figure 1.

Why is an in-memory database system with transaction logging so much faster than a disk-based DBMS for inserts, updates and deletes? First, on-disk DBMSs cache large amounts of data in memory to avoid disk writes. The algorithms required to manage this cache are a drain on speed and an IMDS (with or without transaction logging) eliminates the caching sub-system.

Second, widely-used b-tree indexes can greatly improve random lookups, and retrieval of database content in sorted order, but they are also expensive for an on-disk DBMS to maintain during insert/update/delete operations, and become more burdensome as database size increases. (B-tree lookups are less costly with an IMDS because they impose no cache processing, accesses happen at in-memory speed, and trees are shallower because they contain no duplicate index data.)

Third, the writes imposed by transaction logging are sequential, i.e. they append to the end of the log file, adjacent to the previous write. DBMSs write to a transaction log file sequentially, too, but whenever pages must be flushed from a DBMS cache, they are written to random locations on disk. Thus the disk head potentially travels far between write locations, adding mechanical overhead during these flushing operations.

Index Searches & Table Traversals

Database index searches and table traversals showed no significant performance change when moving from on-disk DBMS to IMDS+TL, or when changing storage device from hard disk to either SSD or the Fusion ioDrive2. This result was expected, because such database "reads" are typically much less costly, in performance terms, than insert, update and delete operations. Because they do not change the database contents, reads impose little overhead, and hence have less to gain from moving from disk to main memory for data storage, or from disk to SSD or to the Fusion ioDrive2 to store the transaction log. The server's ample memory ensured that all or most of the on-disk database was cached, so there were few (if any) cache misses that would cause a read from the database storage device.

Storing the *Entire* Database on SSD or Fusion ioDrive2

So far this paper has compared storing the entire DBMS on hard disk drive, to scenarios using an in-memory database system with transaction logging enabled and the log stored to HDD, SSD or Fusion ioMemory. Since SSDs outperform HDDs, and the ioDrive2 greatly exceeds the SSD performance, another option worth examining is to use the on-disk DBMS and store the *entire* database (not just the transaction log) on either the SSD or ioMemory. McObject performed such tests. In Figure 2, below, these results are inserted, in red, into the chart shown earlier.

Insert	Loops/ms	Perf. Multiple
HDD - On-disk	1.60	1.00
SSD - On-disk	**3.00**	**1.88**
ioDrive2 - On-disk	**6.12**	**3.83**
HDD - IMDS+TL	5.11	3.20
SSD - IMDS+TL	15.49	9.69
ioDrive - IMDS+TL	32.05	20.04

Update	Loops/ms	Perf. Multiple
HDD - On-disk	3.00	1.00
SSD - On-disk	**7.23**	**2.41**
ioDrive2 - On-disk	**15.99**	**5.33**
HDD - IMDS+TL	5.32	1.77
SSD - IMDS+TL	17.30	5.77
ioDrive - IMDS+TL	38.25	12.75

Delete	Loops/ms	Perf. Multiple
HDD - On-disk	1.50	1.00
SSD - On-disk	**2.98**	**1.99**
ioDrive2 - On-disk	**6.17**	**4.12**
HDD - IMDS+TL	5.31	3.55
SSD - IMDS+TL	17.77	11.87
ioDrive - IMDS+TL	34.72	23.19

Figure 2.

Again, the performance multiple in the chart represents the speed advantage lent by the combination of a particular database type and storage device over using a conventional on-disk DBMS with “spinning disk” storage. From these results, it is clear that both database type and storage matter when optimizing system performance. The traditional DBMS performance is markedly better when using the more advanced storage devices (SSD and ioDrive2) instead of HDD. But these results cannot compare with using the same storage device with an IMDS and transaction logging (though always the ioDrive2/DBMS combination, and in one instance the SSD/DBMS combination, beats the HDD with IMDS+TL).

Test 2 – Database Transactions

Methodology

Database operations are typically grouped in transactions, or units of work that complete, or fail, together. To some extent, the content of a transaction is determined by the task: when a bank customer transfers funds, one account must be debited and one credited in a single stroke. But developers often have some choice whether to design their application with longer transactions (more operations per transaction) or shorter transactions (fewer operations per transaction). To accomplish the same work, application code with fewer, longer transactions should outperform code with many shorter transactions. This is the case due to the larger amount of transactional "housekeeping" (processing associated with data being flushed to storage) at the end of each transaction.

The second benchmark test sought to confirm this performance difference and to illustrate the impact of storage device type (HDD, SSD or ioDrive2) on it. The test application implemented loops, using *eXtreme*DB's SQL ODBC interface, to perform the following five SELECT and UPDATE statements per loop:

```
UPDATE accounts SET Abalance=Abalance+value WHERE Aid=value

SELECT Abalance FROM accounts WHERE Aid=value

UPDATE tellers SET Tbalance=Tbalance+value WHERE Tid=value

UPDATE branches SET Bbalance=Bbalance+value WHERE Bid=value

INSERT INTO account_history(Tid, Bid, Aid, delta, htime, filler) VALUES
(value1,value2,value3,value4,now,value5)
```

In one test variation, each of these statements was implemented as a single, very short transaction (resulting in five transactions per loop). A second variation used long transactions containing 1,000 statements (or 200 loops per transaction). The benchmark measured the performance (loops accomplished per second) of both test variations using each of the three storage devices: hard disk drive (HDD), solid state drive (SSD), and the Fusion ioMemory tier, flash-based ioDrive2. Results are shown in Figure 3.

	Long Transactions - loops/sec.	Short transactions - loops/sec.	Performance Multiple long vs. short trans.
On-disk - HDD	28,818.44	416.63	69.17
On-disk - SSD	31,133.25	811.84	38.35
On-disk - ioDrive2	40,683.48	1,839.01	22.12
IMDS+TL - HDD	41,806.02	915.23	45.68
IMDS+TL - SSD	44,072.28	2,206.77	19.97
IMDS+TL - ioDrive2	54,406.96	4,910.63	11.08

Figure 3.

As in the test of database operations discussed in the first section of this report, the IMDS+TL and ioDrive2 combination is fastest, while DBMS/HDD is slowest. This is true whether transactions are long or short.

The performance multiple column in Figure 3 shows the speed advantage lent by using long rather than short transactions for each of the combinations of database type and storage device. The expectation of superior performance from longer transactions is clearly met, with these delivering anywhere from 11.08x to 69.17x more speed than short transactions.

Note that while the Fusion ioDrive2 exhibits the fastest performance (often by a wide margin) in absolute terms, across the board, the performance boost from moving from short to long transactions is highest when using the slowest storage device (HDD). The reason for this is that each transaction commit requires that the transaction be durable (recoverable), requiring it to be flushed to the media. Flushing to a HDD means scribbling the bits on the surface of the disk (moving the disk head, waiting for the platter to rotate to the write location, etc). That mechanical process is much slower than the data recording process with the other two devices. Because each write to a hard disk takes longer, there is greater benefit to grouping the changes into fewer big transactions, as reflected by the over 69X performance improvement.

Conclusions

The first and most obvious conclusion from these tests is that if an application demands both fast database processing and data durability, and is expected to modify the database (i.e. the software is not read-only), then an in-memory database system with transaction logging outperforms an on-disk DBMS. Both technologies use persistent storage to secure database durability; the IMDS+TL's greater speed rests on its streamlined design (no cache management, for example) and more efficient use of resources.

Equally clear is that type of storage device matters greatly. In the test implemented using the database system's C/C++ interface (a more likely choice of tool than SQL in high performance

applications), the IMDS+TL/hard drive combination accelerated processing by as much as 5.33x over the traditional on-disk DBMS with HDD. However, pairing the IMDS+TL with state-of-the-art storage (Fusion ioMemory) pushed the in-memory database system's performance advantage when using transaction logging to as high as 23.19x the speed of a traditional DBMS storing records on hard disk.

The second test, using *eXtreme*DB's SQL API, confirmed the advantages mentioned above as well as the superior performance of long transactions compared to short ones. It suggests that if a developer is locked into using hard disk storage for an application, he/she should strive to use long transactions. Storage device choice remained an important determinant of performance in the second test, whether paired with long or short transactions, or with either type of database. The absolute high and low results – 54,406.96 loops per second using long transactions, IMDS+TL and Fusion ioMemory vs. 416.63 loops/second using short transactions, on-disk DBMS and HDD – show a *combined* performance effect of more than 130x (13,000%), highlighting the extent to which the blend of database type, storage device and programming technique can determine application performance.