Abstract
McObject’s benchmark application tests the 64-bit eXtremeDB in-memory database system (IMDS) as it scales beyond one terabyte of managed data. It examines query speed as well as parallel processing efficiency.

The test platform consisted of an SGI Altix 4700 system with 80 dual-core 1.6 Ghz Itanium 2 processors (160 cores total) and 4 TB NUMA RAM, hosted by Louisiana Immersive Technologies Enterprise (LITE), and running SUSE Linux Enterprise Server 9.

Benchmark Application
The benchmark database consists of PERSONS and ORDER tables. These represent any transaction (such as a gift, a trade, etc.) in which there are two instances of a ‘person’ (one for each side of the transaction) and one instance of a ‘order’ between them.

The database is accessed using eXtremeDB’s native API and, separately, using its SQL ODBC API, called eXtremeSQL. Engineers created 3 billion PERSONS records (rows) and 12.54 billion ORDERS records (rows), resulting in a database size of 1.17 terabytes.

Performance – Select
The tests were run with varying numbers of threads (1, 80 and 160). The first test was a simple query:

```
SELECT name FROM persons WHERE person_id = ?
```

![Chart 1 – SELECT performance of the eXtremeDB native API (queries/second).](chart1.png)
Performance – Join
The second test query implemented a three-table join:

```sql
```

Chart 2 – SELECT performance of eXtremeSQL ODBC API (queries/second).

Chart 3 – JOIN performance of the eXtremeDB native API (queries/second).
Chart 4 – JOIN performance of the eXtremeSQL ODBC API (queries/second).

Performance – Subquery
The third and final test was a subquery:
SELECT * FROM persons
WHERE PERSON_ID IN
  (SELECT RECEIVER_ID FROM orders
   WHERE SENDER_ID IN
     (SELECT RECEIVER_ID FROM orders WHERE SENDER_ID = ?))

Chart 5 – SUBQUERY performance of the eXtremeDB native API (queries/second).
Observations
The SELECT operation using eXtremeDB’s native API shows less performance improvement between 80 and 160 threads compared to the improvement between 1 and 80 threads. This stems in part from the fact that virtually no computation (a strong suit for the native API) was required for this operation. Instead, the application spends nearly all its time accessing random locations in the NUMA RAM. NUMA memory is not optimized for this type of access. As a result, those random accesses were going almost as fast as they could go at 80 threads, so there was only incremental improvement as the application scaled to 160 threads.

In the JOIN and SUBQUERY operations (and with SQL in every case) relatively more time is spent in computation versus accessing the memory, so 80 threads were not yet taxing the system’s ability to fetch the data. In addition, the JOIN and SUBQUERY tests exploit “locality of reference”: when the database is provisioned, related rows are more likely to be placed on the same or adjacent pages, causing far fewer cache invalidations and leading to more uniform improvement in performance as the number of threads scales up.

Conclusion
The test delivered groundbreaking results in terms of in-memory database system scalability. Among its firsts: performance numbers from an IMDS (eXtremeDB) beyond 1 TB; confirmation that an IMDS can support an arbitrarily large number of concurrent processes/threads and deliver consistent performance; and proof that performance does not degrade significantly with increased database size.

To put McObject’s benchmark result of 87.7 million queries per second in perspective, consider that the standard currency of such comparisons is transactions per minute. For example, in late 2010, one database vendor pointed to a TPC- audited benchmark result of 30.25 million transactions/minute as documenting “the world’s fastest database.”

Comparing performance across different applications and operating environments is notoriously tricky. But viewed in this light, even the 1.82 million queries/second (109.2 million queries/minute) result achieved in McObject’s benchmark test, using eXtremeDB-64’s SQL ODBC API (rather than the faster native API) in a technically challenging sub-query operation, proves that IMDS technology is an excellent candidate for high-volume real-time database management. Further evidence is provided by the nearly linear scalability achieved in McObject’s benchmark, from a single processor core up to 160 processor cores, with some minor and unavoidable drop-off in scalability occurring for operations that involve a higher proportion of system “housekeeping.”

Copyright 2012 McObject