Market Requirements Document

Feature Name: In-Memory Mode

Version 1: Date Submitted: 05/18/2010 Completed By: Leon Guzenda

Description of the Problem

Objectivity/DB and its derivatives (Objectivity/SQL++ etc.) provide classic database management system capabilities. The product can be configured in many ways, including using large caches to speed up access to data that is primarily read-only. However, some applications need true in-memory speed and referentially correct transactions without the need to always persist changed data to disk. Others may also need the ability to asynchronously persist transactions and still guarantee ACID-like behavior. In-Memory Databases and hybrids that can work in memory, but also persist data to disk, address this problem and perform better than Objectivity/DB if scalability isn't an issue.

The core problem is that Objectivity/DB guarantees ACIDity, but at the cost of always committing changes to disk and preventing a thread/process from starting a new transaction until the previous one has finished. The I/Os and commit operations increase the duration of transactions.

Description of the Requested Feature

1. A version of Objectivity/DB that can be configured, or defaults, to a true In-Memory Mode, meaning that:

- Data is retained in the memory cache and is never written to disk unless the thread/process requests that it be.
- Rolling back a transaction (before it is committed) behaves correctly.
- Opening an existing read-only database causes a one-time load into cache, followed by the required In-Memory Mode operation. There could be an option to do this on demand, at the container level.
- Opening an existing updatable database causes containers to be read in their entirety and cached. This should happen on demand. Committing a transaction that has updated the container in any way must occur asynchronously, with an exception being thrown if the process/thread subsequently tries to access the containers being committed before the data has been safeguarded.
- An alternative, non-ACID transaction mode should be supported. In this case, the data is not committed to disk during the “commit” operation. It can be used and reused until a conventional (persistent) transaction is performed. However, other threads/processes must be prevented from updating containers that are held “open” in this mode.
- A page server cached mode should be supported in order to provide faster access to data that is being shared between processes. Alternatively, threads within the

same process should be able to access a shared cache in a controllable, safe (ACID) way.
- All unnecessary locking, journaling and tidying operations would be eliminated or minimized for threads running in the In-Memory Mode.
- SQL++ and Infinite Graph should be enhanced to be able to run in the In Memory Mode.
- PQE query agents should be configurable to exploit the In-Memory Mode.
- Full and incremental backups should be able to asynchronously work with threads and processes running in the In-Memory Mode. This is necessary to compete with commercial products that take snapshots of the cache periodically.
- Fully redundant failover would be desirable but is a low priority.

Part of an existing feature or does it require another feature, if so, which one?

- In-Memory Mode would be an enhancement of an existing feature, but it could be priced as an optional product.

How is this problem being solved now, and why isn't that acceptable?

Some telecom customers have used large caches and volatile or non-volatile RAM disks to achieve in-memory speeds for read-only data that they have cached themselves. However, updating transactions always cause journal and database file I/Os to occur, so a true object-oriented IMDB can outperform Objectivity/DB.

What languages must support this capability?

- C++ (Essential)
- Java and .Net for C# (Soon)
- SQL++ (Highly desirable)

Which platforms must be supported?

- Linux, Windows and Solaris.

Do any competitors already have this feature?

- None of the ODBMSs have a true In-Memory Mode. However, there are numerous In-memory products, such as eXtremeDB (which we believe beat us in an evaluation at the NSA), IBM's SolidDB and Oracle's TimesTen. We beat TimesTen in a benchmark at Glenayre many years ago, but Oracle may have improved its performance since then.

- There is also a class of open source and commercial distributed cache/grid products, such as Gemfire's Hydra, Gigaspace XAP and Oracle's Coherence.

Customers who require this feature

- The online gaming community. Zynga would be a good target prospect.
- The telecom and process control equipment markets
- Embedded intelligence, security and defense applications.
- Social networking and relationship analysis applications.

Revenue at risk, or which could be won

- This capability could open new markets (see above).
- We may have lost one NSA project to a true IMDB.

When is this required?

- Release 10.2, or at some point prior to R11.

Additional Notes

1. Related Material

We will also need:

- New Quality Assurance material.
- Updated documentation, training and web based training.
- Colateral.