# Non-Volatile Memory for High Performance Software

*Bill Bridge, Oracle*

Byte addressable Non-Volatile Memory (NVM) on the processor's memory bus is a new technology that has the promise to dramatically change the way data is persistently stored and accessed. New NVM technologies are now available that support terabytes of NVM on a processor memory bus. This allows software to access persistent data at memory bus speeds, 1000 times faster than disk. This eliminates the need to organize data into fixed size blocks accessed sequentially to optimize performance. NVM introduces a new storage paradigm that will require significant software changes to take full advantage of its potential.

Since NVM is persistent storage, it needs to be managed like storage. NVM aware file systems, such as XFS on Linux, can build a file system on a pool of NVM. Using mmap(), the NVM holding a file is placed directly in the application address space for load/store access. Once this is done, the persistent data is directly accessible to the application without any OS operations such as paging.

NVM in an application introduces new problems related to managing consistent application state. To solve these problems Oracle developed an NVM API consisting of an open source C library and a set of C language extensions: https://github.com/oracle/NVM-Direct. It provides region file management, atomic transactions, locking, and NVM heaps for allocation. Prevention and early detection of software corruption of NVM data structures is a major goal of the API.

Since NVM is on the memory bus, it is only available to a single failure domain. Replication between servers is required for high availability. Replication needs to be at a higher level than byte for byte copying to avoid replicating corrupt data. NVM corruption due to software bugs will be a serious problem.

There are a number of ways databases can use NVM to improve performance:

- Multi-terabyte database files can be kept in NVM. They can be mapped into the database instance to avoid any file I/O for queries or block writes. There is no lengthy cache warm up time at startup since the entire file is visible through load and store instructions.
- Logging to NVM can be very low latency and very high throughput. Log replication through a high-speed network can provide redundancy.
- In memory databases can be kept in NVM so there is no need to checkpoint them to disk or reload from disk at startup.
- Persistent data formats no longer need to be formatted into blocks that require CPU time to marshal and unmarshal.
- Highly available metadata can be maintained in NVM by a group of servers using a consensus algorithm. Such a group can provide much lower latency and higher throughput using NVM than is possible using disks.