

# Exploring Mass Storage Concepts to Support Exascale Architectures

**David Fellingner**

DataDirect Networks Inc., Chatsworth, California, [dfellinger@ddn.com](mailto:dfellinger@ddn.com)

## **EXTENDED ABSTRACT**

It has become common practice to operate a “scratch” file system in conjunction with a simulation environment. This file system is generally used to store checkpoint data as the simulation is run so that a recovery from an application error, power failure, etc. can be accomplished. The problem is that the checkpoint must be synchronous and the simulation must be paused to allow this function. This infers a duty cycle of compute and an I/O operation and, of course, the I/O operation is interrupting the computation. As supercomputer architectures have grown in node count the requirement for increased file system bandwidth has also grown to maintain machine balance. There are concerns that future file system implementations will not scale to satisfy the needs of multiple petaflop machines and that this critical checkpoint process may not be possible considering the time required to execute the cycle.

Current file systems are implemented through servers attached to the switching backbone of the compute cluster. The physical attachment is generally either Ethernet, Infiniband, or some proprietary bus depending on the cluster interconnects. These servers run a scalable file system such as Lustre or GPFS and the checkpoint files are stored with associated metadata. In the case of Lustre this metadata is stored on a separate service node and is therefore asymmetrical to the data with respect to the switching paths. Each of the service nodes is connected to block level storage with some high performance physical interconnect that can support a SCSI protocol. The latest implementations consist of block devices directly connected to the servers through the use of Infiniband at either double or quad data rate. The connection is additionally complicated by the need for redundancy at both the server and block storage levels. This redundancy is usually accomplished by requiring that two servers are connected to redundant storage control systems that can share a common disk drive pool.

Future file systems must have far greater efficiencies in executing data push and pull operations. The elimination of these SCSI and external bus layers would certainly eliminate two large bottlenecks that exist in today’s implementations.

An entire SCSI layer can be eliminated by moving the service node into the storage controller. This could be accomplished in an SMP environment with dedicated resources for both the applications.

This implementation must be accomplished without changing the service node code in any way since it is always necessary to allow field updates of this code. The interface must be generalized such that any storage socket can be accommodated supporting any file system service. The preferred means for establishing this socket interface is to operate the service node as a virtual machine hosted on the storage controller. This virtual environment can share a common memory map with the storage controller.

A typical write operation would be accomplished by the service node executing either a direct I/O operation or a kernel controlled operation to a SCSI socket. In this architecture, however, the socket does not service an external SCSI bus but rather becomes a cache segment for the storage controller. Rather than the typical bus transitions to execute a serial operation over an external bus the cache can be utilized for a data write operation to the storage system as soon as the memory lock is released on the socket. The entire SCSI bus operation including bus commands and the generation of a SCSI command descriptor is completely eliminated resulting in a significant increase in performance efficiency. Laboratory testing of this interface has shown an efficiency of 96% for the virtual machine operations. The loss of 4% is more than compensated by the elimination of the steps required to execute a SCSI operation.

This virtual interface is so efficient that it is possible to host several virtual machines in the same storage controller to allow shared storage access from service nodes or applications.

The complete elimination of an external SCSI bus and its related physical infrastructure and protocol represents a major step toward increasing the efficiency of data operations in a cluster environment. A data transaction to non-volatile media is concluded by the storage controller opening a SCSI socket with an external SCSI bus transaction executed to the media. If this media is a spinning disk the operation includes disk armature operations and a physical write or read to the media with a subsequent acknowledgement of the operation.

Another potential efficiency would be realized by populating the storage controller node with non-volatile RAM such that a storage write operation would act to simply populate data to a specific RAM segment that is in the memory space of the storage controller. An ideal implementation of this architecture would be to populate PCI Express physical sockets with NVRAM media completely eliminating rotating media. The result would be a storage architecture that could transfer data from a file system service node to non-volatile memory without an external SCSI interface. The data transfer is simply a PCI operation first to the storage controller where redundant elements are added then to non-volatile memory through another efficient PCI operation. There are no Host Bus Adaptors or Host Channel Adaptors required for this data operation.

The most cost efficient storage technologies are still rotating disk drives so the implementation of an entire storage system on solid state media may be cost prohibitive utilizing available devices. A hybrid implementation may be the best approach where the storage controller still maintains an external SCSI interface to rotating disk drives but also accommodates some amount of solid state non-volatile memory. This architecture would allow very high data transfer bandwidth to the storage system for a period until the solid state memory was filled then a reduced bandwidth to rotating media.

This architecture would be ideally suited for scratch storage environments accommodating fast data transfers to a storage system with solid state media sized to match the cache size of a typical cluster application. During the compute cycle of the cluster data could be migrated by the storage controller from the solid state memory to rotating media in preparation for the next I/O cycle.

Elimination of data transition and file system bottlenecks at a system level must be a goal in the implementation of large scale compute clusters. The necessity for checkpointing calculations is clear but this must be done while maintaining a very high compute duty cycle. The reduction of SCSI layers and protocol is a clear advantage in the overall data mobility architecture. Additional bottlenecks in file system operations must also be studied to streamline data transactions