Exploring Mass Storage Concepts to Support Exascale Architectures

Dave Fellinger, *DataDirect Networks*

ABSTRACT: There are many challenges that must be faced in creating an architecture of compute clusters that can perform at multiple petaflops and beyond to exascale. One of these is certainly the design of an I/O and storage infrastructure that can maintain a balance between processing and data migration. Developing a traditional workflow including checkpoints for simulations will mean an unprecedented increment in I/O bandwidth over existing technologies and designing a storage system that can contain and deliver the product of the computation in an organized file system will be a key factor in enabling these large clusters.

Potential enabling technologies will be discussed which could achieve dynamic scaling in both I/O bandwidth and data distribution. Concepts will also be presented that could allow the distribution elements to act upon the data to simplify post processing requirements and scientific collaboration. These technologies, if effected, would help to establish a much closer tie between computation and storage resources decreasing the latency to data assimilation and analysis.

Current Technology

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. The end result is that the failure of either a server or storage controller will not affect the availability of the file system. Figure 1 *(see page 2)* illustrates the connection of two file system service nodes in redundant service. Each service node can access both storage controllers and both are on a common network such that data requests can be routed to either in the case of node failure.

A typical checkpoint operation consists of a series of data writes from the cluster nodes to the file system. In the case of Lustre the write request is first made to the metadata server then to the specific server assigned to that operation. The node executes the write to the storage controller through a SCSI process and the storage controller in turn executes the write to the disk system through another SCSI process adding redundant elements in the execution of a RAID algorithm. The interface from the storage controller to the disk is generally a native disk physical connection and protocol such as Fibre Channel, Serial Attached SCSI, or Serial ATA. The node acts to aggregate transfers through the use of cache and organize the write operations to the storage controller. The controller again caches and reorganizes the data to minimize the mechanical process of disk seeking. The typical SCSI transaction to the storage controller is from 1 to 4 megabytes for data and much smaller requests for journal entries. Data reads are simply executed in reverse order with the cluster client first accessing the metadata server then accessing the server maintaining the specific file requested.

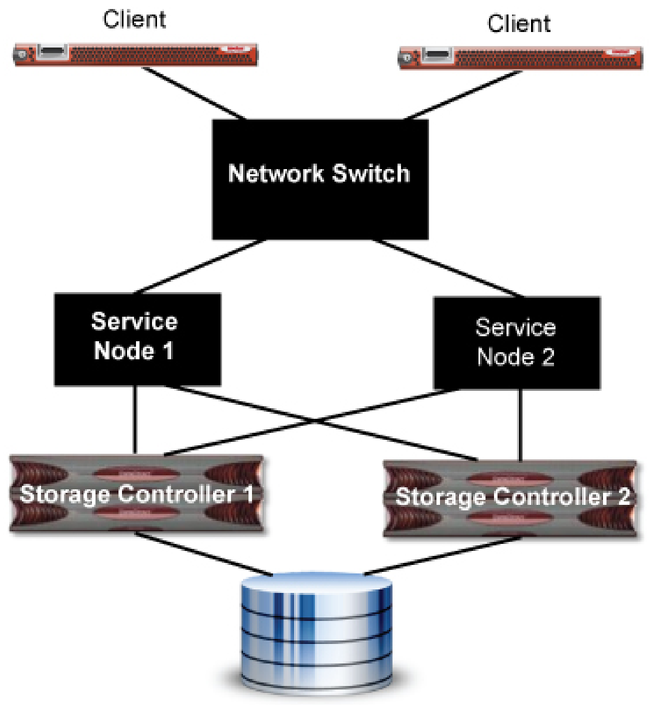

Figure 1

Bottleneck Analysis

Examining the sequence of events for any data movement we see that a TCP request from the client opens two layers of SCSI protocol, the first from the server to the storage controller and the second from the storage controller to the disk drive pool. A system bottleneck analysis requires the examination of each transition individually.

The physical connection from the server to the storage controller is implemented by some bus adaptors that can reside on the server's internal bus. As mentioned, this can be Infiniband, Fibre Channel, or some implementation of Ethernet. The storage controller must contain a similar adaptor to enable the data transitions to its internal bus. The interconnect is generally implemented through two optical fibres for bidirectional operation each stimulated by a semiconductor laser and decoded through the use of a PIN diode on the receiving end. The external data path is usually directly coupled; however, it is possible to add a switch layer to enable multiple paths to the storage controller. The software interface to the adaptors is a driver that maps data by maintaining a vector memory location in either the server or the storage controller servicing the data buffer until the confirmed completion of the data transition. Inefficiencies can be found in the execution of this data transition. First, since the bus can be operated in a switched environment, a protocol must be maintained for each transaction. The bus must be opened, serviced, and closed for each SCSI command. In addition the SCSI command also requires a protocol including a command descriptor and semaphores to enable efficient data reorganization. This can result in payload to protocol ratios as small as one for metadata transactions.

Inefficiencies can also be found in the second SCSI layer that is used to connect the storage controller to the disk subsystem. Again, a physical connection supported by control electronics must be utilized to transition data to the attached disk drives. The storage controller contains an interface that allows its native bus to be connected to the native bus of the disk drives. This is a serial connection that can be implemented by a copper or fibre physical layer with the decision usually driven by the distance between the controller and the drive enclosures. The disk drive enclosure has a similar interface which is usually attached to a switch allowing data steering to a specific target drive. As in the connection to the server, this bus must also accommodate both bus and data protocols. In addition, the disk drives have limited cache and it is never used for data writes. In all but sequential operations any disk access requires a positioning of the heads to a specific sector of the platter. This is a very slow process compared to the peak sequential performance of any disk drive so the process is mitigated by the implementation of a command queue. In other words, the disk drive can reorder its read or write commands to maximize efficiency. While this does save a great deal of bus attachment time it infers additional protocol delays in both bus acquisition and maintenance as well as SCSI protocol to allow the queue maintenance.

Data Acceleration Techniques

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 the 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 the 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.

Conclusion

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 check pointing 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.

Storage controllers with virtual servers and solid state memory are being characterized by DataDirect Networks in laboratory environments and in the field in beta environments. Test results will be published upon the conclusion of this testing.

About the Author

Dave Fellinger brings over thirty years of experience in engineering including film systems, ASIC design and development, GaAs semiconductor manufacture, RAID and storage systems, and video processing devices. He has architected high performance storage systems for the world’s fastest supercomputers. Mr. Fellinger attended Carnegie-Mellon University and holds patents in optics, motion control, video processing, and pattern recognition.