Architecting iSCSI-based I/O Systems for High Performance Computing Clusters

Jacob Liberman [1], Li Ou [2], Suneet Chandok [3]

Dell, Inc.
1 Dell Way, Round Rock, Texas, 78682
[1] Jacob_Liberman@dell.com
[2] Li_Ou@dell.com
[3] Suneet_Chandok@dell.com

***Abstract*— The iSCSI protocol communicates over existing IP infrastructure, making it an intriguing storage alternative for cost-conscious HPC users. Two approaches have emerged for using iSCSI in an HPC context. With the first approach, I/O servers – also called data movers -- access iSCSI storage at the block level and export it to the cluster nodes via a distributed file system. With the second approach, cluster nodes connect directly to the storage at the block-level via local iSCSI initiators. This paper describes both architectures and compares their performance and scalability across several workloads at increasing cluster sizes. Our results show that cluster and workload characteristics dictate the suitable approach for each cluster. We conclude with design recommendations based on our measured performance results.**

I. Introduction

Over the course of the last decade clusters have emerged as the dominant architecture for technical computing. This trend is primarily due to the extraordinary price for performance they can deliver. However, many question the sustainability of this approach. The rapid expansion of multi-core processors and special purpose processing elements such as hardware accelerators are driving a deep discontinuity between floating point capability and communications efficiency. [1] Efficient use of dense floating point capability requires fast and reliable access to data through the memory and I/O subsystems. In industry, these aims are often achieved through the use of Storage Area Networks (SANs) that provide block-level access to remote storage via redundant, high bandwidth network links. The research community has been slow to adopt SAN technologies for HPC applications because the substantial acquisition and maintenance costs associated with specialized SAN hardware and software can quickly erode clustering's favourable economics.

Small Computer Systems Interface (SCSI) is a popular family of protocols for communicating with storage devices. [2] The iSCSI protocol defines a method for transporting SCSI packets over TCP/IP. iSCSI allows block-level storage access over existing IP infrastructure. iSCSI clients – also called initiators – can be instantiated in either hardware or software, meaning that client's equipped with standard Ethernet Network Interface Controllers (NICs) do not require additional hardware to access iSCSI devices. For these reasons, iSCSI is an intriguing alternative for cost-conscious HPC users. Furthermore because iSCSI provides an open architecture for block level I/O, it allows the user to architect a scalable storage solution tailored to meet their budgetary and performance requirements.

There are two architectures for using iSCSI storage in an HPC context. With the first approach, I/O servers – also called data movers -- access iSCSI storage at the block level and export it to the cluster nodes via a distributed file system. We call this approach an *indirect* SAN, as shown in Figure 1.

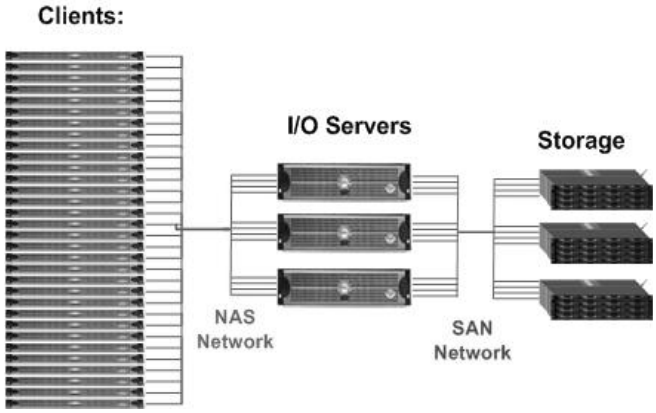

Figure 1 -- Indirect SAN

With the second approach, cluster nodes connect directly to the storage via local iSCSI initiators. We call this approach a *direct* SAN, illustrated in Figure 2.

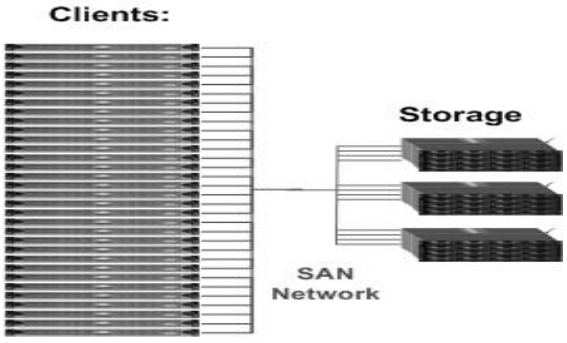

Figure 2 -- Direct SAN

In theory, the relative benefits of both approaches are clear. The indirect SAN should scale well because data movers can

978-1-4244-5012-1/09/$25.00 ©2009 IEEE

be added when the cluster grows. The direct SAN should be more cost effective because it eliminates the data movers. However, empirical, performance based comparisons of these architectures do not exist. This study compares the performance and scalability of both architectures using the same hardware test bed at increasing cluster sizes. Our results can be used as architecture selection criteria for iSCSI based storage systems for HPC clusters.

II. Architecting iSCSI Storage Solutions

Indirect and direct SANs require distinct upper layer software stacks to utilize iSCSI storage. For this study we selected two file systems that are representative of the typical software stacks used with each approach. For the indirect SAN architecture, we selected the Lustre file system. We selected Red Hat Global File System (GFS) for the direct SAN architecture. Although we recognize that the characteristics of both file systems influence our results, we believe they accurately represent other file systems within their class.

A. Lustre

Lustre is a scalable parallel file system with high availability features. [6] Lustre is an open source files ystem, It was originally developed by Peter Braam at Carnegie Mellon University under the GNU GPL license. It is now maintained by Sun Microsystems.

Lustre clusters contain three kinds of systems: File system clients -- which can be used to access the file system; data movers -- called Object Storage Servers (OSS) -- which provide file I/O service; and Metadata servers (MDS) which manage the names and directories and keep a global view of the file system metadata.

B. Red Hat GFS

Red Hat GFS is a 64-bit cluster file system that allows multiple clients simultaneous block-level access to a single file system namespace. GFS was originally developed at the University of Minnesota under the GPL license. Most recently GFS has been acquired and productized by Red Hat, Inc., and remains open source. [4]

GFS clients mount shared storage as a local file system. Data are stored as blocks on the underlying Logical Volume Manager (LVM) subsystem, which means GFS can be used in conjunction with the features provided by Linux LVM. A lock manager coordinates file access among the attached systems in order to ensure disk cache coherency. Although GFS supports several lock managers, we used the Distributed Lock Manager (DLM) for I/O synchronization across cluster nodes. [5]

III. Test Methodology

The test methodology subjected both architectures to typical HPC workloads at increasing cluster sizes. In order to ensure a fair architectural comparison, we used the same hardware and software whenever possible.

A. Environment

Both SANs used Dell EqualLogic PS6500 iSCSI storage arrays for shared storage. The PS6500 is a 4-U storage array with 48 SATA disks and redundant active/passive I/O controllers. Each I/O controller has 4 iSCSI interfaces accessible through a single target IP address. Multiple iSCSI storage arrays are organized logically into groups. Each group contains multiple arrays, called group members. Volumes can be distributed across multiple members or pinned to a single member within the group. The PS6500 storage arrays were used for this study because they can scale – both in terms of bandwidth and capacity – as additional members are added to the group. This feature ensured architectural differences were highlighted, not storage limitations. One to three member PS6500 groups were used for each study.

The cluster used to generate client I/O was comprised of 32 Dell PowerEdge 1950 III servers with two quad-core Intel Xeon X5450 processors and 4 GB of DDR-2 667 memory. Each server also had two Broadcom NICs. One interface was used for cluster administration and the other was used as an iSCSI software initiator during the direct SAN study or a TCP/IP NIC for the indirect SAN study. The servers were running Red Hat Enterprise Linux with the 2.6.9-53 kernel.

For the indirect SAN study, the Lustre OSS used were 3 Dell PowerEdge 2970 servers with two quad-core AMD Opteron 2300 CPUs and 32 GBs of DDR-2 667 memory. Each segment server had two onboard Broadcom BCM5708 NICs and three Intel 82571EB dual-port Gigabit Ethernet NICs. Four ports were bonded into a single load balanced team for sharing with the file system clients. The other four ports accessed the storage as iSCSI initiators.

The core network switch was a Cisco Catalyst 6500-E with two 48 port WS-X674 line cards. Each line card was configured as a separate VLAN in order to prevent broadcast traffic between the iSCSI and Ethernet interfaces. All switch ports, iSCSI initiators, and client NICs were configured with a Maximum Transmission Unit (MTU) size of 9000 bytes.

B. Workload Characterization

This study evaluated two I/O patterns. The first was comprised of sequential writes and reads. Sequential I/O of large files is a typical workload for computational clusters. In some cases – such as seismic image processing – the large files are the data. In other cases, large data sets are generated during program execution that must be moved en masse for archival purposes or visualization. This I/O pattern utilized very large files in order to saturate the network bandwidth and overwhelm client data caching. In all cases, file sizes were at least twice as large as each client's memory capacity. We used IOzone to generate the sequential I/O. IOzone is a file system benchmark tool that generates and measures a variety of file operations. [3] IOzone supports a clustered mode of operation, used in this study, that can launch multiple, simultaneous operations on different files.

The second I/O pattern was parallel I/O. During parallel I/O tests, multiple clients wrote to or read from the same file at the same time. This benchmark simulated the type of I/O commonly employed by cluster nodes executing parallel codes for distributed memory systems, such as MPI-2 based

applications. We used the IOR HPC benchmark utility to generate parallel I/O workloads. The IOR benchmark, developed by Lawrence Livermore National Labs, generates parallel/sequential read/write operations typical of scientific applications. [7]

IV. Results

This section shares the results of the benchmark performance tests at increasing cluster sizes.

A. Performance

IOzone was used to generate the sequential workload. Each client wrote and then read an 8GB file to the storage system at each cluster size. The total workload scaled with the number of clients. At the largest test size, all 32 clients ran two simultaneous threads writing 8GB files each, for a total output size of 512 GBs.

Figure three illustrates the indirect SAN results of the sequential I/O test. Client read performance scaled up to 1100 MB/s across three data movers and three iSCSI storage arrays. The write performance peaked at ~1150 MB/s with sixteen clients before dropping slightly with 32 clients. For the direct SAN, 32 clients accessed the three iSCSI storage arrays directly. Sequential performance for both read and write scaled up to 32 clients. Aggregate write performance reached ~1000 MB/s while read performance peaked above 1200 MB/s. Both the direct SAN and the indirect SAN scaled very well for sequential I/O, nearing 1500 MB/s, the peak theoretical performance of the iSCSI group.

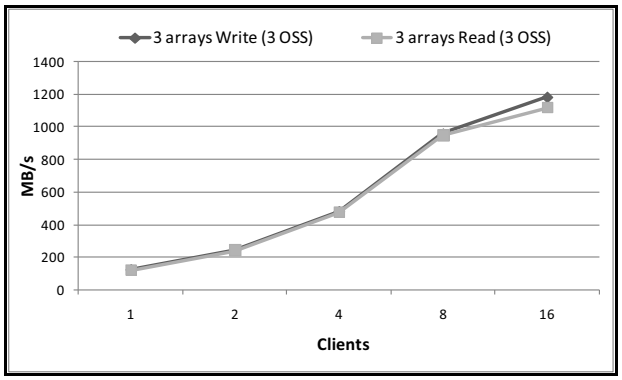

Figure 3 -- Indirect SAN Sequential Performance [Lustre]

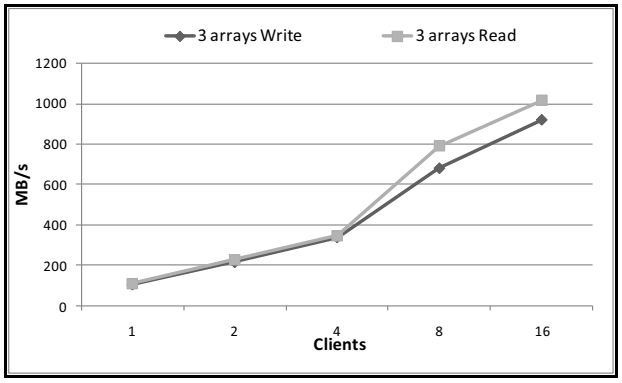

Figure 4 -- Direct SAN Sequential Performance [GFS]

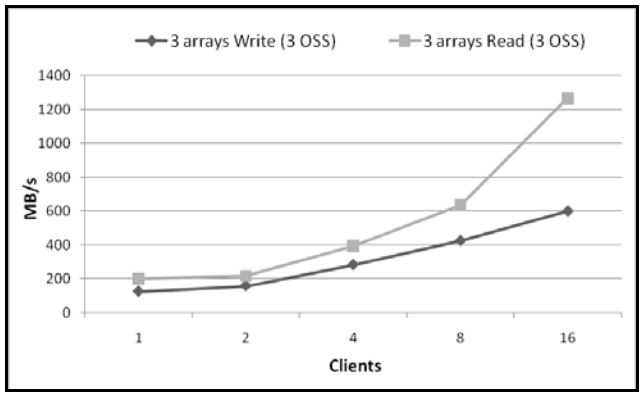

Figure 5 -- Indirect SAN Parallel Performance [Lustre]

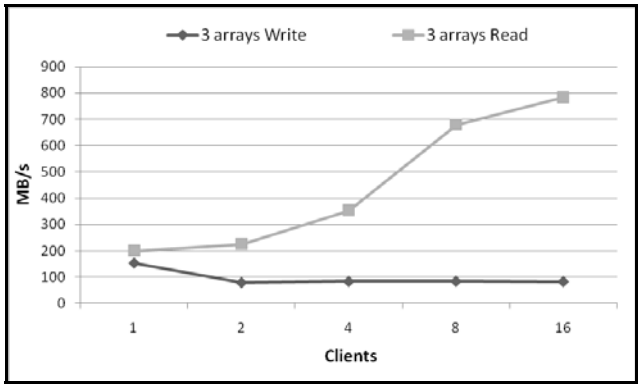

Figure 6 -- Direct SAN Parallel Performance [GFS]

The parallel read result for the direct architecture was similar to the sequential read result. The parallel write results also differed across architectures. For the direct SAN, write performance never scaled beyond 90 MB/s due to an architectural limitation. The distributed lock manager for the file system enforced exclusive lock during write so the aggregate bandwidth of the test was bound by the maximum link speed of single clients. Each client accessed the shared storage via a single network interface. Both architectures achieved higher read performance than write performance because parallel read requests do not require exclusive file locks. All hosts can read from the same file simultaneously.

The indirect SAN showed better scalability for parallel writes. With this architecture client parallel write requests were stripped crossing the data movers. Therefore, the maximum parallel write performance for the indirect approach was dictated by the bandwidth available to the data mover and the file system overhead.

B. Scalability

For sequential read, both approaches scaled as clients were added to the cluster as illustrated in figures 7 and 8. The read performance of both architectures scaled linearly as the iSCSI array group size increased from one to three members.

Theoretically, the scalability of indirect SAN architecture can be maintained indefinitely so long as additional OSS servers are added with increased client count.

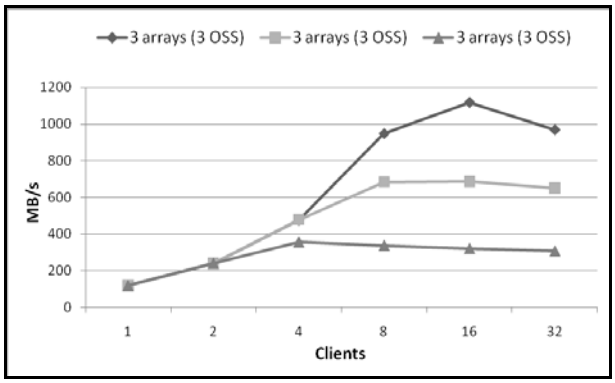

Figure 7 -- Indirect SAN Read Scalability [Lustre]

The direct architecture not only scaled well, but also maintained high network bandwidth utilization as the group increased from one to three members. As depicted in figure 8, the aggregate bandwidth of the single member storage group was ~400 MB/s, or 80% of the theoretical bandwidth for the array. When the iSCSI group increased to three members the bandwidth per client tripled.

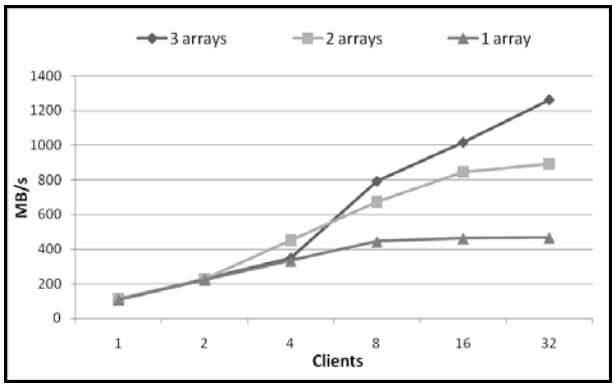

Figure 8 -- Direct SAN Read Scalability [GFS]

V. Conclusion

When comparing the performance and scalability of the indirect and direct iSCSI SAN architectures, important distinctions emerged. First, the direct SAN architecture made high efficient use of the networked storage on sequential workloads. However, several factors limit the continued scalability of this approach to large node counts. First, the maximum bandwidth of the storage limits the effective bandwidth per client. This is true for both approaches, but for the indirect approach, data movers provide an I/O buffer by responding to certain requests with cached data. Second, iSCSI storage solutions are limited to finite simultaneous iSCSI sessions. Data movers aggregate iSCSI sessions that could exceed these limitations. Finally, file system limitations can cap the upward scalability of the direct SAN approach. GFS, for example, has not been tested on more than 128 nodes. GFS distributed file locking overhead would also curtail continued scalability. This is a limitation of the file system implementation, and not the direct architecture.

Although the indirect SAN may make less efficient use of the available network bandwidth, additional data movers can be added when the cluster size increases. Reasonable bandwidth per client should be sustained even when the cluster grows to thousands of nodes in size, provided the ratio of data movers to clients to storage is maintained. Additionally, the indirect architecture also limits the maximum number of concurrent iSCSI sessions maintained by the storage. This eliminates the lock management overhead associated with the direct SAN approach by utilizing a meta-data server.

From a design standpoint, the direct SAN approach places a larger burden on the storage. The direct SAN does not provide a mechanism for future expansion. Therefore, when designing a direct iSCSI SAN, the I/O capabilities of the iSCSI storage must be a prime consideration. The advantage of the direct SAN approach is that it eliminates the cost and performance overheads associated with the data movers. The direct approach is most suitable for relatively small clusters that are not anticipated to grow.

The indirect SAN approach adds performance overhead for traditional and sequential workloads. The higher bandwidth between the data mover and the storage improves parallel I/O performance for the indirect SAN. When designing an indirect SAN, careful thought must be give to the optimal ratios of clients to data movers to storage arrays. The indirect approach – while perhaps more costly – provides flexibility in terms of future scalability and expansion.

Directions for future study include evaluating the impact of iSCSI software initiator overhead on direct SAN cluster clients. Initial application studies show excellent speedup can be maintained even during peak I/O periods. iSCSI over 10 Gigabit Ethernet can benefit both architectures and requires further evaluation. Finally, the study should be broadened to include other cluster file systems such as Lustre that are frequently used with HPC clusters.

References

[1] A. Buttari, J. Dongarra, J. Kurzak, J. Langou, P. Luszczek, and S. Tomov, *The Impact of Multicore on Math Software*, 2006.

[2] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, & E. Zeidner. *Internet Small Computer Systems Interface iSCSI.* (2003). Retrieved: http://www.ietf.org/rfc/rfc3720.txt

[3] D. Capps, & T. McNeal. *Analyzing NFS Client Performance with IOzone.* Retrieved: www.iozone.org/docs/NFSClientPerf_revised.pdf

[4] M. O'Keefe. *Red Hat GFS vs. NFS: Improving performance and scalability.* 2006. Retrieved: http://www.redhat.com/magazine/008jun05/features/gfs_nfs/

[5] DLM Project Page. 2007. Retrieved: http://sources.redhat.com/cluster/dlm/performance.html

[6] Lustre. Retrieved: http://www.lustre.org

[7] H. Shan & J. Shalf. *Using IOR to Analyze the IO Performance for HPC Platforms.*(2006).