Selecting a Scalable Cluster File System

Cluster File Systems, Inc.

Nov 2005

Introduction

This short paper introduces considerations which are important in selecting a **scalable cluster file system** for compute clusters. After we introduce the storage challenges facing cluster architects we compare various solutions. Finally, we describe how Lustre™[1] can meet requirements.

Challenges

Managing many servers

Almost always the storage capacity required in an HPC cluster exceeds what can be handled by a single server. This introduces the issue of managing many, possibly hundreds of servers storing the data. Many installations still use collections of NFS servers and delegate management to the users and administrators, not to the file system software. The time taken by I/O operations often significantly affects the overall cluster performance.

Aggregate and single node I/O performance

One of the most prevalent I/O patterns is a collective I/O performed by all nodes participating in a job. This is common for checkpoint data dumps, and when a single file is used by the entire job. The second common pattern is a single node writing or reading data. The cluster architecture will take into account the aggregate bandwidth available, and in well-tuned installations this is the sum of the bandwidth offered by all servers. The bandwidth available to individual nodes is generally limited not by the servers, but by the network and achievable client file system bandwidth.

Capacity

A continued increase of the storage capacity of a cluster after deployment is extremely common. Storage solutions must allow for growth, through adding servers and resizing file systems. File systems should not introduce limitations on file sizes or file system sizes.

Client count and multiple clusters

Many storage solutions work well with small numbers of clients, but very few work well when the client count increases. Typical issues of concern are stable performance characteristics when many clients create files in a single directory, and the ability of the file system to handle a true storm of I/O or metadata requests which can be initiated by all clients simultaneously. Clusters are rarely used in isolation. File systems should scale well enough to join the file systems of possibly dozens of clusters into globally available file systems. This means that initially modest client counts can increase by an order of magnitude.

High availability and rolling upgrades

Each server in a cluster is often equipped with a daunting number of storage devices and serves between dozens and tens of thousands of clients. The file system should handle server reboots or failures completely transparently, through a high availability mechanism, such as failover. Applications should merely perceive a delay in the execution of I/O commands for failed servers. A robust failover mechanism in conjunction with software that offers interoperability between versions can be used for rolling upgrades of file system software on active clusters. The absence of a robust failover mechanism leads to hanging or failed jobs, requiring restarts and cluster reboots, which are extremely undesirable.

Overview of solutions

Shared-disk filesystems

Shared disk file systems were introduced prominently by the VAX VMS[2] clusters in the early 80's. They rely on a storage area network, which can be supplied by a fibre-channel, iSCSI or IB SAN. GPFS, Polyserve, CxFS, GFS and Terrafs all fall in this category. The architecture of these file systems mirrors that of local disk file systems, and performance for a single client is extremely good. However, concurrent behavior suffers from an architecture that doesn't target scalability. Typically these systems offer failover, of varying degrees of robustness. GPFS has been very successful on clusters up to a few hundred nodes. Typically SAN performance on fibre-channel is reasonable, but it cannot compete with clients that use IB, Quadrics, or Myricom networks with native protocols.

Exporting Shared File Systems with NFS

To limit the scalability problems encountered by shared disk file systems these systems are often used on an I/O sub-cluster, which exports NFS. This is commonly done with GPFS, CxFS, GFS, and Polyserve's Matrix Server. Isilon offers an appliance for this purpose. Each of the I/O nodes then exports the file system through NFS version 2 or 3. For NFSv4 such exports are much more complex due to required management of shared state among the NFS servers. While the scalability of NFS improves, the layering introduces further performance degradation, and NFS failover is rarely completely transparent to applications. NFS offers neither POSIX semantics, nor good performance.

[1] Lustre is a trademark of Cluster File Systems, Inc.

[2] All product names are the trademarks or registered trademarks of their respective owners.

Object architectures
Several systems offer a novel architecture to address scalability and performance problems. Ibrix offers a symmetric solution, but little is publicly known about its architecture, semantics and scalability. Panasas offers a *server hardware* solution, combined with client file system software. It makes use of smart **object** iSCSI storage devices and a metadata server, which can serve multiple file sets. Good scaling and security are achievable, even though all file locking is done by a single metadata server. The Panasas system uses TCP/IP networking. Lustre's architecture is similar, but is an Open Source, software-only solution running on commodity hardware. Lustre will be discussed in the next section.

Lustre – ultimate scalability

Lustre is based on novel storage architecture, initiated at CMU in 1999, going beyond the older NASD architecture in exploiting better storage protocols. A Lustre cluster has clients (up to tens of thousands), object storage servers (up to thousands) and metadata servers (currently a failover pair, in the future a cluster comprising up to dozens of nodes). All servers are organized in failover pairs, and export a large file system. Servers can be added dynamically, and the file system is completely POSIX compliant. Although Lustre offers ADIO interfaces, can disable locking, can perform direct I/O suitable for advanced databases, and has many other tunable settings, the default values are typically used and the performance benefits derived from these hooks are minimal.

Lustre currently runs on Linux where it can interoperate among all supported processor architectures. Versions for other operating systems are under development.

Lustre networking & routing
Lustre was designed to use multiple networks simultaneously using native protocols. It achieves unparalleled performance over TCP/IP, supports all flavors of Infiniband, Myrinet (GM), Quadrics Elan and various proprietary networks from Cray. Typically achievable bandwidths are >90% of the native speed, and except on TCP/IP many tens of thousands of RPCs can be executed per second. Moreover, Lustre can perform routing between networks of different types. This truly enables site wide access to cluster file systems used on all clusters.

Lustre performance
Lustre performance, both for data and metadata operations is very high, and scales to tens of thousands of nodes. Figures 1 and 2 below, taken from an independent study done at NCAR and the University of Colorado[3], show some comparisons with other file systems for parallel writes and creates in a single directory.

Lustre's I/O engines are tuned to handle requests from tens of thousands of clients without decay. Figures 3 and 4 below show the throughput of an I/O server, writing concurrently to 32 files per service thread with up to 32 active service threads. This server reaches bandwidths exceeding 2.4 GByte/sec.

Lustre Scalability
The largest Lustre installation is the Red Storm computer at Sandia National Laboratories, which has 11,500 Lustre clients. Individual clients in this cluster have demonstrated concurrent read/write operations with a total throughput of 2 GByte/sec. Because protocols are carefully tuned, a relatively inexpensive metadata server can easily handle this installation, but future Lustre versions will have clustered metadata, which will scale metadata throughput by another order of magnitude or more. Another large-scale installation is the IBM BlueGene/L computer, where 64,000 CPUs use the Lustre file system via the 1,024 BlueGene I/O nodes, which subsequently use some 400 Lustre servers to export a file system close to 1 PB in aggregate size. Here, 64,000 file system benchmarks running in parallel complete successfully, writing at a throughput of approximately 22 GByte/sec. Other large installations are found elsewhere in government, the oil and gas industry, rich media, and pharmaceuticals.

Lustre High Availability
Lustre failover delivers completely application-transparent system call completion. Lustre Metadata Servers are configured in an active/passive pair, while Object Storage Servers are typically deployed in an active/active configuration that provides redundancy without extra overhead. Although a file system checking tool (lfsck) is provided for disaster recovery, journaling and sophisticated protocols resynchronize the cluster within seconds, without the need for a lengthy fsck. CFS guarantees version interoperability of Lustre between successive minor versions of the software and, as a result, failover is now regularly used to upgrade the software without experiencing any cluster downtime.

Conclusion

This paper discussed options for cluster file systems. We discussed how scalability, high performance and robust failover play an increasingly important role in HPC deployments. Lustre is specifically designed for these environments.

[3] Shared Parallel Filesystems in Heterogeneous Linux Multi-Cluster Environments, Jason Cope*, Michael Oberg*, Henry M. Tufo*†, and Matthew Woitaszek*, (* University of Colorado, Boulder, † National Center for Atmospheric Research)

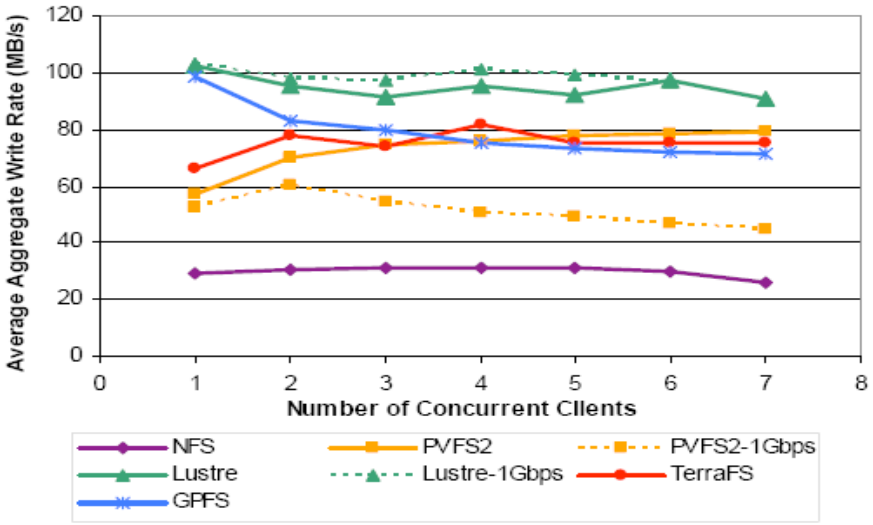

Fig 1: Xeon cluster average aggregate write bandwidth by number of clients

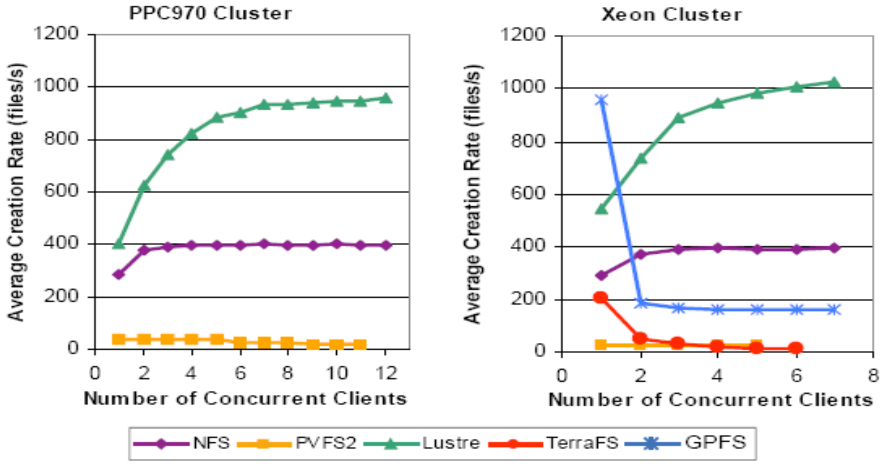

Fig 2: Average aggregate file creation in a single directory by number of clients

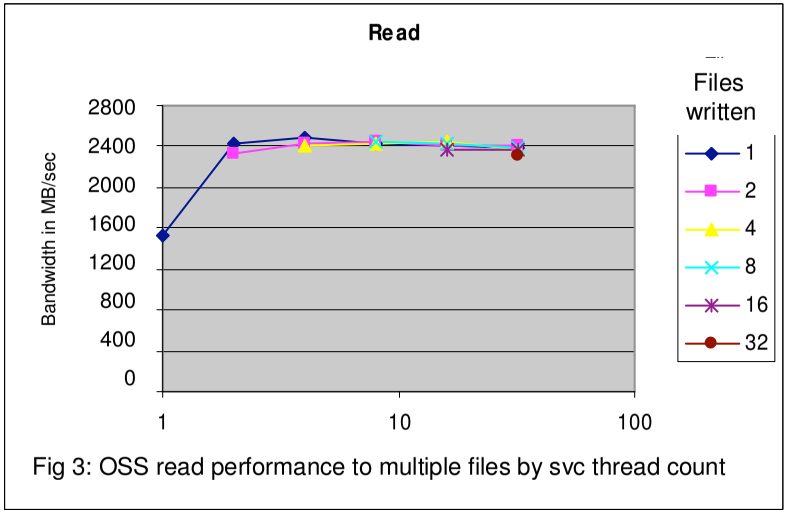

Fig 3: OSS read performance to multiple files by svc thread count

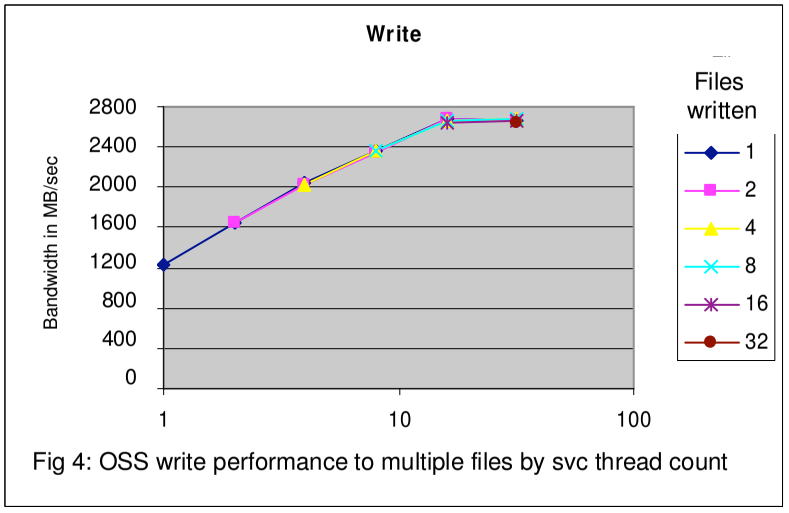

Fig 4: OSS write performance to multiple files by svc thread count