GPFS on a Cray XT

R. Shane Canon, Matt Andrews, William Baird, Greg Butler, Nicholas P. Cardo, Rei Lee
Lawrence Berkeley National Laboratory

May 4, 2009

Abstract

The NERSC Global File System (NGF) is a center-wide production file system at NERSC based on IBM's GPFS. In this paper we will give an overview of GPFS and the NGF architecture. This will include a comparison of features and capabilities between GPFS and Lustre. We will discuss integrating GPFS with a Cray XT system. This configuration relies heavily on Cray DVS. We will describe DVS and discuss NERSC's experience with DVS and the testing process. We will conclude with a discussion of future plans for NGF and data initiatives at NERSC.

1 Introduction

As High-Performance Computing systems continue to scale, the demands placed on the file system continue to increase. As the computational systems have increased in parallelism, so too have the file systems which support them. Two of the most common parallel file systems used in large scale HPC systems are IBM's General Parallel File System (GPFS) [5] and Sun Microsystems Lustre File System [4]. This paper will briefly examine these two file systems including their history, general architecture, and capabilities. Some of the differences between the two file systems are highlighted. The steps required to integrate GPFS on a Cray XT system are explained. Cray's Data Virtualization System (DVS) are described since it is a key key technology for using GPFS on a Cray XT system.

2 GPFS Overview

GPFS is IBM's flagship scalable file system. It is a mature product with over a decade of research and development invested in it. GPFS originated from Tiger Shark, a file system designed to support interactive multimedia systems first conceived in the early 1990s. Since that time it has evolved to support large HPC systems across various platforms including AIX cluster, Blue Gene systems, and various flavors of Linux Clusters. Some of the largest GPFS file systems include the file system on Lawrence Livermore National Lab's Purple [3] and Argonne National Lab's Intrepid [2]. Purple with over 1500 clients has a 2 PB file system and has demonstrated 122 GB/s of file system bandwidth.

2.1 Architecture

GPFS employs a shared disk model with a distributed lock manager to insure consistency. Disks can be shared via a Network Shared Disk (NSD), through a common storage area network (SAN), or both models concurrently. In an NSD mode, each disk is accessed by a specific server or set of servers and then projected over a network to the clients. Additionally, AIX supports Virtual Shared Disk (VSD) mode. This is similar in design to an NSD, but the sharing is performed by the Operating System versus GPFS. The distributed lock manager maintains consistency. A central manager issues tokens that permit the token holder to issue byte-range locks.

GPFS also supports a multicluster configuration. In this type of configuration, the client clusters and server clusters can be placed in separate GPFS clusters. This changes the communications patterns for lock negotiations. If a client possesses the token for a file and another client in the same GPFS cluster needs to negotiate a lock, the token will remain on the client cluster and the clients will communicate directly with each other to negotiate the locks. On

the other hand, if a different GPFS cluster needs to negotiate a lock for a file, the token will shift back to the "owning cluster" and the lock will be negotiated between the various clusters. A multicluster configuration affects other processes as well, but the impact on the lock negotiations is the most significant.

3 Lustre Overview

Lustre was conceived as a response to the Advanced Simulation and Computing Initiative (ASCI) Path-Forward File System. The contract for the development was awarded to Hewlett Packard who subcontracted Cluster File Systems, Inc to perform the development. This contract expired in 2005. In 2007, Sun Microsystems acquired Cluster File Systems, Inc. While the development and overall control of Lustre is maintained by Sun, the majority of the file system is licensed under a GPL license and can be freely downloaded from Sun. Lustre has broadened in its reach while continuing to scale to the largest HPC systems. The largest Lustre file system is deployed at Oak Ridge National Labs and supports its Jaguar system. The file system has over 25,000 clients, is 10 PB in size, and has demonstrated over 200 GB/s of file system bandwidth.

3.1 Architecture

Lustre is an object-based file system. Metadata is stored on a single Metadata Target and extend data is stored on Object Storage Targets (OSTs). File data can be striped across multiple OSTs to achieve high-bandwidth. Lustre provides its own transport layer called LNET. LNET supports many networks and can efficiently route between networks. Routing can even be striped across many routers and can detect and avoid failed routers. Locks are maintained by the OSTs. Clients will optimistically obtain a large byte range lock and then renegotiate the locks with other clients as new write requests arrive.

4 NGF

The NERSC Global File system (NGF) was initially deployed in 2004 to provide a global file system across all NERSC production systems. The file system is based on GPFS and currently provides over 400 TB of space. The file system was intended to enable collaborative data sharing as well as simplify running complex work-flows that span multiple NERSC systems. The ultimate vision of the file system was to replace the islands of storage coupled to the large systems with a global scratch file system that would facilitate data sharing and simplify data management.

4.1 NGF Architecture

An overview of the NGF architecture is illustrated in Figure 1. NGF makes use of many of the advanced capabilities of GPFS including a mixture of SAN and NSD access, storage pools, file sets, and mixed hardware architectures. For systems geared towards data intensive applications, SAN mode mounts are used to ensure the best possible bandwidth to the storage. For many of the clusters, private NSD servers are used. These servers are directly connected to the high-speed interconnect of the cluster (i.e. InfiniBand or Federation). This allows better utilization of the bandwidth and reduces the load on the center-wide Ethernet network. For the Cray Systems, the GPFS client is deployed on specific service nodes which access the storage via the SAN. An I/O forwarding layer called DVS (described below) projects the NGF file system into the compute nodes.

4.2 NGF in Operation

NGF relies heavily on the fail-over and reliability features of GPFS. Many of the upgrades and configuration changes are made live without taking the file system out of production. GPFS' many advanced management capabilities make this possible. The fail-over mechanism allows rolling upgrades of the GPFS software to be performed without requiring an outage. In the past year, NGF has made numerous changes where new storage was added, older storage was drained and migrated out, and space was rebalanced across devices. All of these were done while the file system was online and accessible by users. Scheduled outages are taken only to perform the few tasks that require the file system to be offline. For example, changes to the list of servers handling a storage device must be done with the device removed from the file system or with the file system offline. Therefore, adding a new private NSD server requires that the file system be down.

NGF also leverages many of the data management features of GPFS. For example, the project file system contains directories for many of the major projects at NERSC. In addition, some projects requests sub-project directories to address specific

details of their collaborations. Each of these project directories uses GPFS file sets to control the overall allocation of space. This is more logical than using user or group based quotas since group ownership can get intentionally or accidentally changed within a directory which leads to confusion about where space is being consumed for a group. GPFS also has the ability to create storage pools. A set of policies allows the administrator to control how these storage pools are used. These policies can be based on the path of a newly created file, as well as file characteristics such as age or size. NGF currently places older files which have not been accessed in several months on a pool of SATA based storage, thus freeing up high-performance FibreChannel storage for more active data. NGF also uses storage pools for certain data sets that are intended for access through web portals thus having more relaxed requirements for performance.

5 Cray's Data Virtualization System (DVS)

An I/O Forwarding layer ships I/O commands from a client host to a proxy host where the actual commands are performed. The results from the operation are then shipped back to the originating client. I/O forwarders offer several interesting advantages on large scale HPC systems. They reduce the number of clients that the file system must directly manage. This can reduce the complexity of the lock management across thousands of clients. Secondly, it eliminates the need for the file system client to run directly on the compute node. This can result in a smaller memory foot print on the compute node for file system support and can be more amendable to light-weight kernels. The chief disadvantages to an I/O forwarder are: they introduce additional latency as the data and commands move through the additional layers the I/O forwarder introduces: and they shift the scaling problem from the file system to the I/O forwarding system. An I/O forwarding system has been demonstrated to effectively scale to very large system as demonstrated by various IBM BG/L and BG/P systems.

Cray offers an I/O forwarding layer based on software originally developed at Unlimited Scale, now Cassatt. It was originally developed to support large Linux clusters. DVS layers on top of Portals [1] and makes effective use of the SeaStar's remote put/get capabilities. DVS is a client/server based system. The clients typically run on the compute nodes and ship the I/O operations to the DVS servers where the operations are performed. The DVS servers would normally run on SIO nodes running the native file system client. Initial versions of DVS only supported a single DVS server. However, Cray has extended DVS to support multiple DVS servers including striping operations for a single file across multiple DVS servers. This is critical for using DVS on top of parallel file systems such as GPFS, since this allows individual client to leverage the parallelism of the file system. DVS provides variables and options to control how the IO is decomposed across the servers. They include mount options as well as run-time options expressed through environment variables. When striping is used, a list of servers is specified. A block size parameter (blksize) controls how large of a chunk is directed to a

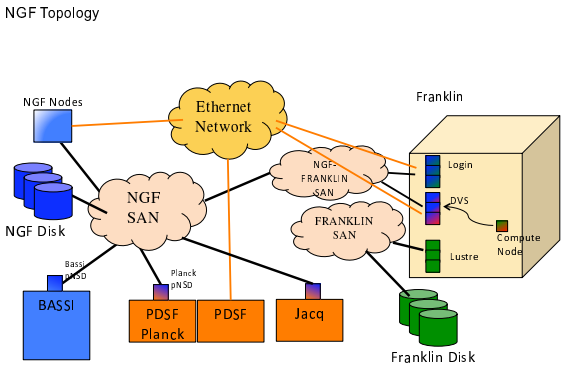

Figure 1: NGF architecture

specific server before going to the next server in the list. Another parameter (maxnodes) determines how many serves from the list will be used for a specific access. The client will wrap back around to the initial server based on a modula (maxnodes x blksize) of the offset. To insure that all clients utilize the same subset of servers in the same order, a hashing algorithm based on the filename is used. This helps minimize the overhead of the lock management for the underlying parallel file system.

DVS consists of a set of kernel modules that are loaded on both the client and server. The client modules plug into the standard Linux Virtual File System (VFS) layer and a standard mount command is used to mount the file system. A sample mount command is shown below. The options include the on the DVS server to be mounted on the client, a list of server nodes, the block size, maxnodes, among others. Some of these options can also be controlled with environment variables (i.e. DVS_MAXNODES and DVS_BLOCKSIZE). This enables users to tune the behavior of DVS if the application has certain access patterns that conflict with the defaults. To convert between the names specified in the nodename list, a node-map is loaded into the kernel memory through a file (*proc/fs/dvs/ssi-map*). The map consists of a simple list of space separated pairs of names and NIDs (portals Network Identifier).

```
mount -t dvs -o path=/mnt,nodename=
   c0-0c2s0n0:c0-0c2s0n3:c0-0c2s1n0:
   c0-0c2s1n3,blksize=1048576 /mnt /mnt
```

The authors had planned to include early performance results for DVS. Some of these tests have been conducted. However, the back-end GPFS file system for the tests was not of sufficient scale to extract meaningful performance numbers. The impact of adding additional DVS servers was studied and shows good scaling but is heavily constrained by the back-end file system. Table 1 summarizes the results of those tests. The back-end file system was capable of around $2GB/s$. The results demonstrate that DVS is able to provide performance consistent with the underlying file system.

6 GPFS on a Cray XT System

GPFS can be easily ported to run on the Cray XT SIO nodes. The process is simplified by the fact that the SIO nodes run a full Linux OS based on SuSE Linux Enterprise Server (SLES) and SLES is a supported distribution for GPFS. The primary changes revolve around the locations for the kernel headers which are required to build the GPFS compatibility layer and subtle aspects of how the IP addresses are configured for the SeaStar interface. After installing the correct kernel source in /usr/src, it is still necessary to create symbolic links from the appropriate /lib/modules/*kernel_version* to the */usr/src* directory. Once this is done, the normal steps for installing GPFS can be performed within the Cray shared root environment. It was also found that two files (*mmremote* and *mmsdrfsdef*) had to be modified to correctly extract the IP address for the node. By default, these utility use output from *ifconfig*. However, Cray uses the *ip* utility to configure the address.

7 Comparison of GPFS and Lustre

While GPFS and Lustre are both used to support large HPC systems, their roots and development history have lead to significant differences in their capabilities and development focus. GPFS's roots as a file system to support interactive video has lead to a design that favors reliability, predictable performance, manageability, and good metatdata access characteristics. These features are emphasized over scaling and peak performance which are also important to GPFS. Lustre's roots from the ASCI Path Forward project focused on massive scaling and performance. While Lustre has always aimed to support tolerance to single points of failure and recovery from various failure modes, it relies on timeouts and external heartbeat programs to initiative fail-over and recovery. Furthermore, some aspects like clustered metadata have been on the Lustre road map for several years but have been delayed in many cases to permit improvements in performance and scaling.

In contrasting the architecture of the two file systems, there are several fundamental differences that are evident. First, GPFS is based on a shared disk model, whereas Lustre uses an Object Storage model. Second, GPFS uses a distributed lock manager whereas in Lustre, lock management as handled by the OSSs in a semi-centralized manner. While these approaches can be viewed as two ways of accomplishing the same things (scalability and consistency), these differences have significant ramifica-

tions in the capabilities of the file system and the pressure points in scaling.

In the GPFS shared disk model, the clients allocate the blocks and the block is the fundamental unit. In the OST model, the Object storage servers own the block device. The clients read and write to objects. The object model adds additional context to a request and the OSS can check an operation to see if it is permissible. This has several consequences. For example, supporting a SAN access model in an object based file system is fundamentally at odds with the design, since the trust would essentially have to be shifted to the clients. Stated another way: in the shared disk model the client handles many of the key decisions; whereas in the object model the OST (handled by the OSS) is responsible for managing many of the details of lock management and disk allocation.

The consistency model of the two file systems also has an impact. GPFS's distributed locking model depends heavily on the clients have a consistent view of who possesses tokens and locks in the file system. GPFS designates quorum nodes that are responsible for deciding what the consistent view is.

Table 2 summarizes some of the features and capabilities for GPFS and Lustre. The table is based on the latest release for both file systems and data from currently deployed systems.

Acknowledgments

The authors wish to thank Dean Roe, Terry Malberg, Brian Welty, and Kitrick Sheets for sharing results for DVS testing, as well as, details about the workings of DVS. This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

About the Authors

Shane Canon is the Group Leader for the Data Systems Group at the National Energy Research Scientific Computing Center. Matt Andrews, William Baird, Greg Butler, and Rei Lee are members of the Data Systems Group. The author can be reached via e-mail at SCanon@lbl.gov.

References

[1] Ron Brightwell, William Lawry, Arthur B. Maccabe, and Rolf Riesen. Portals 3.0: Protocol building blocks for low overhead communication. In *IPDPS*, 2002.

[2] Argonne National Laboratory. Argonne leadership computing facility, May 2009. `http://www.alcf.anl.gov/`.

[3] Lawrence Livermore National Laboratory. ASC purple, May 2009. `http://asc.llnl.gov/`.

[4] Sun Microsystems. Lustre, May 2009. `http://www.lustre.org/`.

[5] Frank Schmuck and Roger Haskin. GPFS: a shared-disk file system for large computing clusters. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, page 19, Berkeley, CA, USA, 2002. USENIX Association.

clients	servers	read (MB/s)	write (MB/s)
1	1	520	485
1	2	916	922
1	4	1345	1462
1	8	1765	1470
1	16	1754	1388
1	20	1870	1439

Table 1: Table summarizing DVS performance as the number of DVS servers is increased. The backend file system was limited to around $2GB/s$. *Courtesy of Cray*

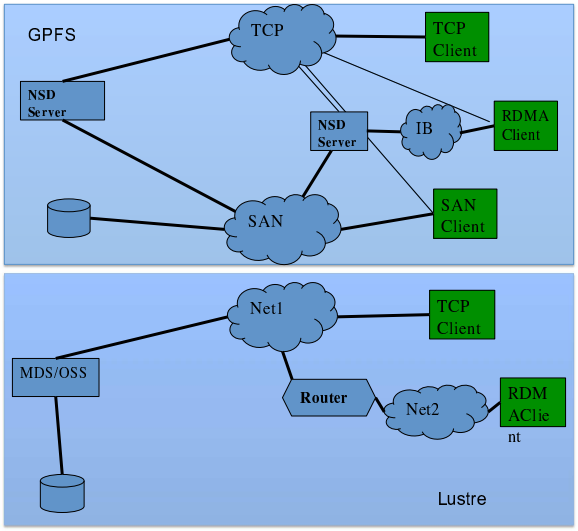

Figure 2: GPFS and Lustre network architectures.

Capability	GPFS	Lustre
Demonstrated Scalability		
Number of clients	1,500	25,000
File System Size (PB)	2	10
Bandwidth (GB/s)	120	200
Management Features		
Online Adding of Storage	✔	✔
Online Removal of Storage	✔	
Online Re-balancing of Storage	✔	
Online Fsck	✔	Partially supported
User and Group Quotas	✔	✔
Filesets	✔	✔
Storage Pools	✔	(Due in 1.8)
Quotas	✔	✔
Snapshots	✔	
Networking Features		
TCP/IP	✔	✔
RDMA InfiniBand	✔	✔
IBM Federation	✔	
SeaStar		✔
Quadrics		✔
Routing across Networks Types	Only if using TCP/IP	✔
Metadata Features		
Clustered/distributed Metadata	✔	Scheduled for 3.0

Table 2: Table summarizing a comparison of features and capabilities of GPFS and Lustre.