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
michael.oberg@colorado.edu

Abstract. In this paper, we examine parallel filesystems for shared deployment across multiple Linux clusters running with different hardware architectures and operating systems. Specifically, we deploy PVFS2, GPFS, Lustre, and TerraFS in our test environment containing Intel Xeon, Intel x86-64, and IBM PPC970 systems. We comment on the feature sets of each filesystem, describe our implementation and configuration experiences, and present initial performance benchmark results. Our analysis shows that all of the parallel filesystems outperform a legacy NFS system but with different levels of complexity. Each of the filesystems demonstrates the best performance under certain conditions. Three of the systems – GPFS, Lustre and TerraFS – depend on specific kernel versions that increase administrative complexity and can reduce interoperability.

1 Introduction

High-performance computing environments require parallel filesystems. Traditional single-host file systems (e.g., those exported via Network File System (NFS)) are unable to efficiently scale to support hundreds of nodes or utilize multiple servers. Parallel filesystems are typically deployed for dedicated high-performance storage solutions within clusters, usually as part of a vendor's integrated cluster solution; typically these parallel filesystems are so tightly integrated with a single cluster's hardware and software environment that sharing is impractical. Recently, several parallel filesystems have been introduced that are designed to make sharing a filesystem between clusters feasible in the presence of hardware and software heterogeneity. Our investigation has identified four parallel filesystems that support such heterogeneity and which, in our opinion, are currently deployable in a production computing environment: GPFS, TerraFS, PVFS2, and Lustre. The objective of this paper is to quantify the performance of these filesystems and identify their respective strengths and weaknesses.

We maintain two moderate-sized clusters with processor counts of 54 and 128 at the University of Colorado (CU). Both clusters currently share home directories and a large scratch space using NFS. We employ subsets of both clusters in this investigation. The primary attraction of implementing a shared parallel filesystem in our cluster environment is the increase in I/O performance and the addition of high availability features, such as redundant storage servers capable of serving files during server outages.

In addition to maintaining the CU environment, we work with members of the Scientific Computing Division (SCD) at the National Center for Atmospheric Research (NCAR) where there is current interest to deploy shared parallel filesystems in their heterogeneous production computing environment at the Mesa Lab facility. Most of the clusters maintained by SCD contain at least 256 processors and are configured to appear similar to their current IBM p690 based production supercomputing environment. NFS is currently used to provide shared home directories across all machines for code repositories, but large data files are stored on a centralized tape-based Mass Storage System (MSS). Each cluster has its own private scratch space as part of the vendor-provided installation. Users include stage-in and stage-out directives in their queue batch scripts to manually migrate large data files between mass storage, local cluster scratch space, and other clusters' local scratch space as their jobs execute. Some users are exploring using Grid-based tools such as the Storage Resource Broker [12] and DataMover [11] to move data between clusters in the same machine room. Other users prefix and suffix their queue scripts with commands to stage data through the mass storage system manually. This forces large quantities of data to move through a single host, introducing a bottleneck limited by a single host's I/O and network connectivity. In addition, the file staging operation occurs only on one node, but during this time the remainder of the nodes allocated for the parallel job is idle. Either solution requires that the user, usually an application scientist, directly manage data flow between multiple machines. A single shared filesystem, with sufficient performance to store model data between simulation and visualization execution stages while meeting reliability requirements, would substantially reduce the time, network bandwidth, and storage space consumed by routine bulk data replication while providing a more user-friendly computing environment, thereby allowing scientists to focus on the science.

Our interest is in parallel filesystems that can serve commodity clusters with a minimum of specialized hardware. In this environment, we envision a small storage cluster of servers running the filesystem, connected to disks via SCSI, iSCSI, or fibre channel. While many compute clusters utilize a specialized interconnect such as Myrinet, Infiniband, or Dolphin for MPI traffic, the multi-cluster installations we are familiar with use separate MPI interconnects for each cluster. Thus, we require that each parallel filesystem be able to serve clients using Ethernet and postpone an examination of specialized interconnect compatibility, which requires datacenter preplanning, for future work.

We examine IBM's General Parallel File System (GPFS) [6], Cluster File Systems' (CFS) Lustre [3], TerraScale Technologies' TerraFS [13], and Argonne and Clemson's Parallel Virtual File System 2 (PVFS2) [9], and compare these to the baseline NFS shared file system we current employ. We selected these systems because of their popularity in current high-performance computing environments, ability to function without specialized fibre channel hardware, and advertised capability to operate in the type of heterogeneous cluster environment we're targeting. In surveying these systems we evaluated the ability of each filesystem to fit into typical datacenters similar to those at CU and NCAR, with special attention to the staff effort required to deploy, configure, and maintain each solution. We intend to trade application-centric parallel I/O performance for ubiquity, but the centralized storage space must be of sufficiently high performance that users may read and write data files from it without staging, thus reducing reliance of cluster-specific filesystems and single-host NFS volumes.

Each filesystem is compared in terms of performance, usability, and stability, as well as additional features unique to each filesystem. For performance, we examine average aggregate transfer rates and metadata operation rates in order to confirm that each parallel filesystem outperforms NFS and rank the systems in similar configurations. For usability, we examine filesystem features such as security to determine if the system is appropriate for a large multi-user home directory data store. Finally, we approach stability from the perspective of administration overhead. Systems with control commands that routinely fail, or daemons that cease to execute without warning, are not welcome additions to a managed production computing environment.

The remainder of this paper is organized as follows: Section 2 introduces relevant related work. Section 3 describes our experimental setup and presents a summary of each filesystem's characteristics. We present our installation experiences in Section 4 and performance benchmarking results in Section 5. The final sections present future work and conclusions.

2 Related Work

The performance of individual filesystems has been studied as each filesystem matures, with some national laboratories actively partnering with corporations to produce filesystems to meet their high performance computing requirements. Lawrence Livermore National Laboratory (LLNL), for example, has worked with Cluster File Systems to develop Lustre [4]. The primary LLNL objectives are focused on solving their need for a highly parallel filesystem not restricted to specific hardware, such as IBM GPFS or Sistina's SAN-based GFS, deployed as a scalable and secure datacenter-wide filesystem.

NCSA is exploring the breadth of available parallel filesystems and is in the process of evaluating filesystems for integration with their mass storage and TeraGrid systems [5]. In the past few years, NCSA has tested GPFS, GFS, Panasas, SamFS, SGI CxFS, an ADIC solution, Lustre, and IBRIX. Each of their supercomputer systems supports one or more separate filesystems, but the aggregate petabyte of spinning disk is split into many different systems. NCSA requires a high level of reliability and stability, and is researching ways to move data throughout the datacenter in ways that are transparent to the user and provide high bandwidth.

Work on parallel filesystems for Linux clusters was recently performed by the San Diego Supercomputer Center (SDSC), which examined PVFS, Lustre, and GPFS on their IA-64 cluster [8]. The authors evaluated these filesystems from both the administration and performance perspective, examining the ease of installation and maintenance as well as each filesystem's performance and redundancy characteristics. The SDSC study focused on a homogeneous architecture with much larger fibre-channel equipped storage servers typical of high-performance clusters. The authors discuss the strengths and weaknesses of these filesystems and conclude that filesystem selection is strongly influenced by site requirements. For example, PVFS provides application support through ROMIO but with less performance than Lustre and GPFS.

3 Experimental Setup

Our experimental setup consists of a small storage cluster and two target compute clusters (see Fig. 1). The storage cluster connects to the computational clusters through dual gigabit Ethernet trunk links. The compute clusters are administered using a single authentication namespace, while the storage cluster machines are restricted to administrator access.

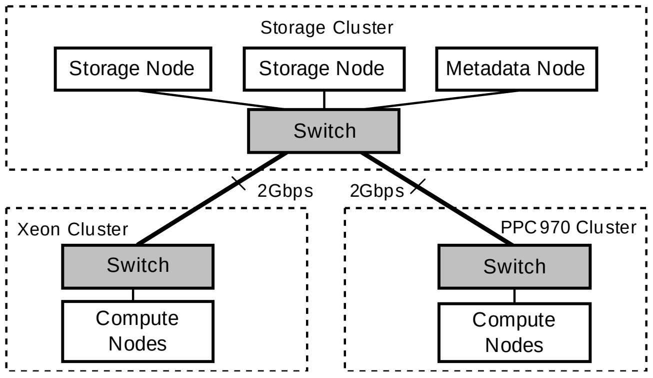

Fig. 1. Experimental cluster setup

Our storage cluster consists of two identical IBM x345 storage nodes. Each x345 system contains dual Xeon 3.06GHZ processors with 512KB L2 cache and 2.5GB of PC2100 RAM. Storage is provided by a direct attached IBM EXP400 RAID5 disk chassis containing 14 Ultra320 10k RPM drives controlled by a IBM ServeRAID 6M Ultra320 SCSI RAID card plugged into the 64-bit, 133MHz PCI-X slot. The RAID5 configuration consists of 13 disks with a single hot spare. Each filesystem was given a 400GB partition on each storage node using LVM except for GPFS. Because GPFS does not support LVM-based devices as storage targets, we configured it with block access to the entire RAID array instead. Some filesystems, such as Lustre and PVFS2, support a separate metadata controller. For these systems, an Intel x86-64 EM64T system with dual 3.4 GHz processors and 8GB RAM was utilized. The storage and metadata nodes are each connected to the core switch using two 1Gbps Ethernet links under the control of Linux channel bonding and switch-side trunking. We tested all filesystems under this channel bonded interface, but we also tested some filesystems with untrunked single gigabit Ethernet connections to the storage servers as well.

Our primary computational cluster, the Xeon cluster, consists of 64 nodes connected by an 8x8 Dolphin torus. Each node contains dual Xeon 2.4 GHz processors and 2GB RAM. This commodity cluster typically runs the kernel.org Linux 2.4.26 kernel, although some filesystems required installing specific kernels. Our secondary computational cluster consists of 27 IBM JS20 blades. Each blade contains two PowerPC 970 (PPC970) 1.6 GHz processors and 2.5 GB RAM. This cluster must run SuSE SLES9 Linux. Because we wanted to maintain production computing on our Xeon cluster, we dedicated 14 nodes to filesystem testing.

4 Experiences

Our experience with the filesystems covers installation, configuration, and management of each product. The first issue we encountered and which warrants special mention is the known vendor compatibility with each system in our environment (see Table 1). Some filesystems required specific kernels on the servers, some required specific kernels on the clients, and some of these requirements overlapped in a fashion that prevented us from installing the products! Most notably, Lustre and GPFS require specific kernel versions only available from commercial Linux vendors such as SuSE and RedHat. Use of these systems requires use of a specifically supported kernel variant. PVFS2, which requires only a kernel module, does not require a specific kernel release. Finally, TerraFS requires a specific kernel patch, but TerraScale currently builds patched kernels based on client specifications.

Table 1. Architecture and Operating System Kernel Support for Filesystem Clients and Servers

	GPFS 2.3	Lustre 1.4.0	PVFS2	TerraFS
Intel x86-64 Metadata server	Not Used	Restricted SLES .141	No Change	Not Used
Intel Xeon Storage server	Restricted SLES .111	Restricted SLES .141	No Change	No Change
PPC970 Client	Restricted SLES .111	Restricted SLES .141	All (Module Only)	N/A
Intel Xeon Client	Restricted SLES .111	Restricted SLES .141	All (Module Only)	Custom 2.4.26 Patch

Restricted indicates that only a small subset of kernels are acceptable. **Custom** indicates that the vendor builds custom kernels for sites at request. **All** indicates that all of the standard available kernels worked. The variant of the operating system we used is shown. The SLES kernels are 2.6.5-7.x with the specific build revision noted.

One other item of interest is the performance gain provided to each filesystem by Linux kernel-level Ethernet channel bonding. When we first obtained the storage servers, each server was connected to the core switch using a single gigabit Ethernet connection. We ran a full series of aggregate I/O bandwidth tests using NFS, PVFS2, and Lustre using this configuration. We then used Linux channel bonding to combine the two Ethernet devices into one interface, which also required configuring the Ethernet switch ports to function as a trunk. With the exception of specifically noted single-gigabit Ethernet tests, all of the tests were performed using Linux channel bonding. Our results show that Linux channel bonding can provide definite performance improvements for some filesystems, but others do not exhibit any performance gain. In the latter case, vendors may recommend alternate configurations to improve performance using multiple non-bonded Ethernet interfaces. We opted to use a consistent experimental networking setup for all filesystems and have not explored these configurations.

4.1 PVFS2

PVFS2 was easy to install and configure. PVFS2 was almost fully compatible with our current computational environment. The software provided useful configuration options, but did not provide some desirable performance tuning parameters and software fault tolerance configuration solutions.

The install process for PVFS2 was fairly straightforward. The build and install process for the software was similar to most GNU Linux programs compiled from source through the standard three step build process: configure, make, and make install. The version tested, 1.0.1, was successfully built for all computer architectures tested on and for both the 2.4.x and 2.6.x Linux kernels.

Configuration of PVFS2 was also an easy task. PVFS2 provides a question and answer based configuration tool that generates the configuration files for all nodes in the system. The generated configuration files are copied to the appropriate nodes once the configuration tool completes. A PVFS2 file system can be configured with a single storage and metadata server or several instances of the servers. The stripe pattern and size cannot be configured for PVFS2. Instead, the PVFS2 documentation recommends that the number of storage servers and metadata servers be configured as appropriate for the intended applications using PVFS2. The documentation recommends that applications that access many files should be configured with multiple metadata servers and applications that read or write large files should employ several storage servers. PVFS2 fails to provide software enabled server failover or redundancy solutions. Instead, the software documentation outlines a configuration for the file system to be redundant to some failures through the use of specialized SCSI hardware.

PVFS2 servers and clients successfully ran on all of our systems and architectures. Our preferred configuration included a single metadata server on the Xeon x86 64-bit node, two storage servers running on IBM x345 nodes, and clients on both Intel Xeon 32-bit nodes and IBM PPC970 64-bit nodes. Starting and stopping the servers and clients required more effort than other file systems. With the scripts and programs provided by the source build, starting the file system consisted of individually staring the metadata and storage servers as well as the clients. Storage servers are started with a single command that may need to be issued more than once if the filesystem has not been created beforehand. The clients require that the PVFS2 kernel module be loaded, the client application be started, and that the PVFS2 file system be mounted. Startup scripts are available to automate these tasks for Red Hat systems but must be configured and installed after the installation of PVFS2. Overall, we found that PVFS2 provides a stable and reliable parallel filesystem in our environment.

4.2 Lustre

Our experiences with the Lustre occurred in several phases. The install process was challenging but worthwhile because of the benefits gained from using this filesystem. Initially, we attempted to apply the Lustre kernel patches to the source of the kernels running in our environment. This was a difficult task and we were not able to successfully patch our stock kernels. Lustre supplies some useful tools to help with patch management, but we found that our stock kernels could not accommodate the Lustre patches.

Next, we decided to install the pre-patched, Lustre-enabled Linux kernels and kernel source supplied by Cluster File Systems. On our new Xeon servers running SLES 9, the kernels were easily integrated. The Xeon nodes running an older Linux distribution were able to run a pre-patched kernel intended for a newer distribution of Linux. We were able to successfully run the Lustre client on the new kernel version, but we experienced frequent system failures when many of the clients were working in parallel. We attribute these failures to the fact that we used a newer kernel version in conjunction with an outdated Linux distribution. We believe certain libraries and dependencies, such as glibc, were not interacting correctly due to the version differences.

Until recently, Lustre was not supported on the PowerPC architecture. Cluster File Systems graciously offered to port the Blue Gene/L version of Lustre to our platform. These new versions of Lustre were installed on our Xeon, Xeon EM64T, and PPC970 platforms. The new version of the software allows for the interoperability of the PPC970 clients with the Xeon based storage servers.

With the new expanded environment and the new version of Lustre installed on most of our systems, we revisited installing the Lustre client on our Xeon platform. Instead of using our outdated Linux distribution, we set up a portion of the cluster to network boot SLES9. These network boot nodes ran a Lustre client and the pre-patched kernel with no problems. With this last installation, we were able to run Lustre on all nodes in our computational and storage environments.

Configuring nodes in our environment to run the Lustre servers and clients was done through the lconf utility. The configuration parameters are individually passed to this utility and are written in XML-formatted files. Lustre supports several storage servers for a single file system including optional failover servers. Additional object servers can also be added as needed. Lustre supports up to two metadata servers: one primary metadata server and one failover server. Additionally, Lustre also offers several key configuration options not available in other solutions. Lustre can be configured with a custom stripe size which determines the amount of data from a particular file that is written to each object storage target (OST). The number of OSTs to stripe a single file across can also be set to a single OST, all OSTs, or any number in between. Limited support for a stripe pattern is available for Lustre as of version 1.4.0; currently only a single pattern is available.

Running the Lustre servers and clients is similar to other parallel file system products. The servers and clients are started individually on the desired systems. Lustre uses a single tool to start all clients and servers for a single filesystem. The tool loads the appropriate kernel modules, starts the appropriate executables, and mounts the file system if the node is a client. After our initial difficulties, we have found our Lustre installation to be high-performance and reasonably reliable.

4.3 TerraFS

TerraFS is a new parallel filesystem exclusively targeted for Linux clusters. Instead of a completely custom system, TerraFS uses the iSCSI protocol for client-server communication and leverages Linux md (multi-device) software RAID for parallelism. TerraFS storage servers run a custom iSCSI target daemon and stores object data using any underlying file system. Clients use a proprietary TerraFS initiator to create an iSCSI connection to each server. Thus, each client has a series of block devices representing every remote iSCSI storage device. A parallel block device is constructed at the client by aggregating the remote iSCSI block devices using Linux md software RAID. When the parallel block device is formatted using the TerraFS extz file system, the proprietary iSCSI targets implement a cache coherent parallel file system.

TerraFS is implemented on the server side as a userspace application but on the client side as a patched Linux kernel and client daemon. At the present time, TerraScale builds custom client kernels to their customers' specifications. A company engineer patched our source tree, installed, and tested the system on a single computer and then sent us instructions on how to install and configure TerraFS on our cluster. In our case, the kernel build stage required about 3 weeks because we were using a highmem kernel configuration that TerraScale had not encountered before and caused the operating system to halt when the filesystem was started. After TerraScale completed the initial testing process, installing and configuring TerraFS on our system proceeded very quickly and without problems.

We found TerraFS less tolerant of transients than other systems. An unexpected reboot of a storage server, for example, would flood client logs with error messages. After several weeks of testing with frequent client and server restarts, we received an error message indicating that the free block count was corrupted and the filesystem incorrectly reported that it was full. Other than this single error, it supported our entire test suite without difficulty. TerraFS fully supports UNIX permissions and ACLs, so it would be suitable for a large multi-user cluster deployment.

TerraFS does not currently support availability or reliability. Although md provides software RAID, including RAID 1 mirroring and RAID 5, TerraFS only supports RAID 0 striping at the present time. In addition, TerraFS does not support the PPC970 architecture, which makes it inadequate for our heterogeneous datacenter. While we found TerraFS to be a fully functional high-performance parallel filesystem, we will wait for the product's feature set and operating system support to mature before considering it for local use.

4.4 GPFS

IBM released GPFS 2.3, the first version that supports heterogeneous PPC970 and Xeon systems, in December 2004. Our experiences with GPFS were very positive, and we found it to be a very powerful filesystem with well-documented administrative tools. After about 2 days of initial experiences, we are now able to install and configure GPFS on every system in our datacenter in just a few hours.

The GPFS installation process was simple and straightforward. GPFS is implemented as a series of kernel modules referred to as the kernel compatibility layer and must be compiled by the end-user at the final stage of software installation. The documentation for this stage is slightly out of date, and we were surprised that the compilation configuration file we needed to edit did not contain sample lines for our officially supported platform. The comments provided sufficient explanation, however, and we were able to compile the kernel compatibility layer on all of our systems. It is important to note that GPFS is only guaranteed to work on certain commercial Linux variants. In our case, we used the SuSE SLES .111 kernel. We also tried to use the .141 kernel, and while GPFS compiled, it did not function at all. We now believe the documentation.

After installing the GPFS software, a quick series of command line utilities are used to configure GPFS. Setup steps include creating the GPFS cluster, creating network storage devices (NSDs), creating a filesystem, adding clients, and finally mounting the volume. One documented limitation that we were initially unaware of is that GPFS requires block access to devices and cannot function on devices managed by Linux LVM. The initial configuration stages appear to work with LVM and only the final filesystem creation step fails. We spent several hours trying to figure this out, but an IBM specialist identified the problem from a quick examination of the configuration files over e-mail. Once the software installation was complete, creating the filesystem required about 5 minutes.

GPFS has several features to support high availability and reliability. SAN and dual-head SCSI configurations can be used to provide high availability to storage devices, keeping disks accessible even if a single server fails. A quorum system allows continuing filesystem operation even during the failure of storage servers. Unfortunately, our simple hardware configuration cannot support these options. GPFS can also provide redundancy through software-based block replication. Overall, GPFS was straightforward to install on our systems and has provided excellent performance.

5 Results

The primary results of our work are functional in nature, and we were able to successfully install and configure each filesystem on our heterogeneous cluster test bed. All of these filesystems operated as expected and were capable of storing and retrieving data. We then examined the performance of each filesystem with both single processor and parallel benchmarks.

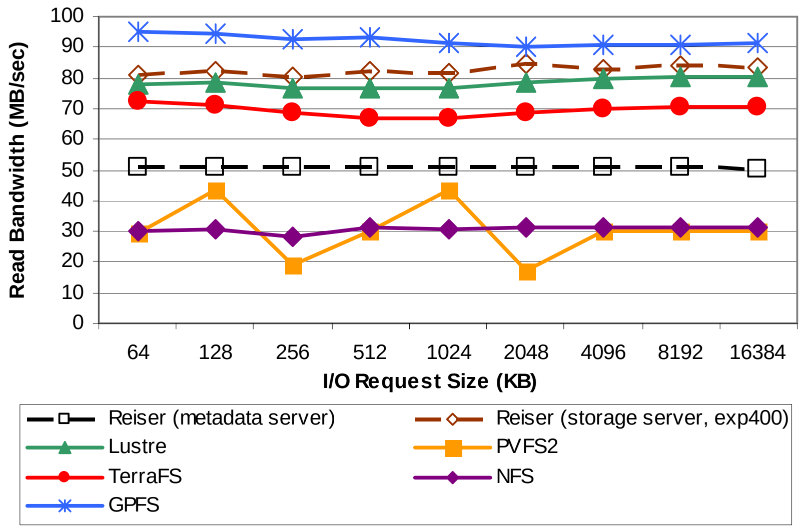

Fig. 2. Single Xeon node read bandwidth by request size

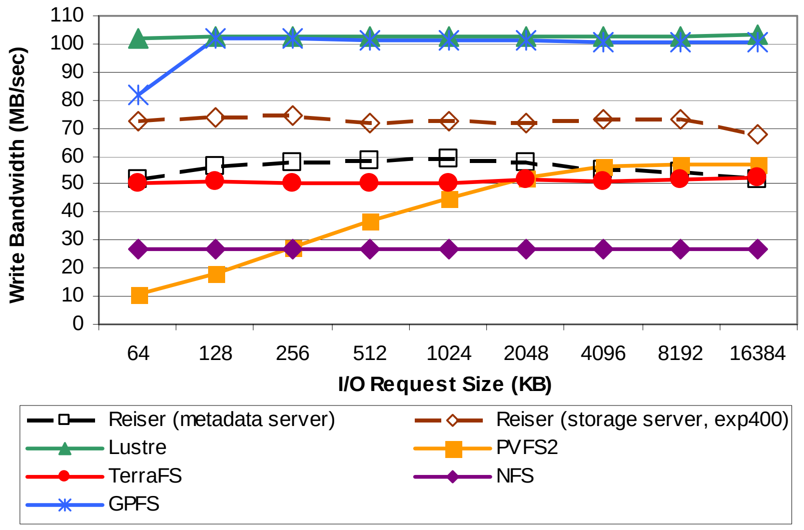

Fig. 3. Single Xeon node write bandwidth by request size

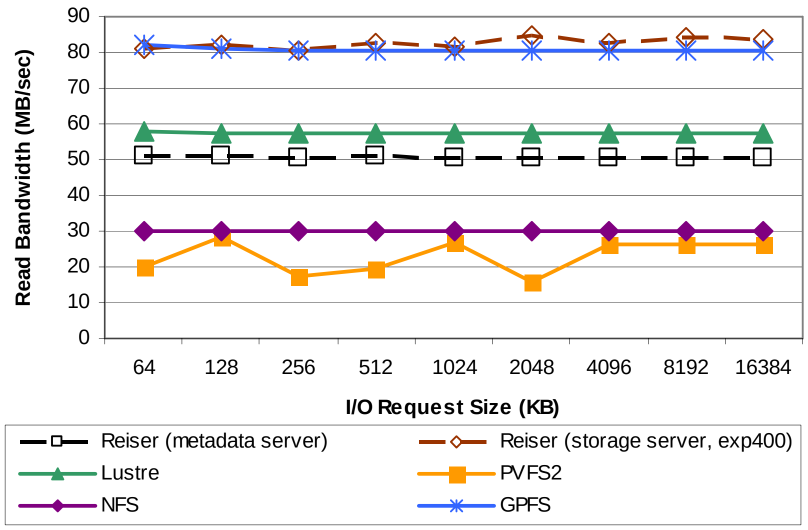

Fig. 4. Single PPC970 node read bandwidth by request size

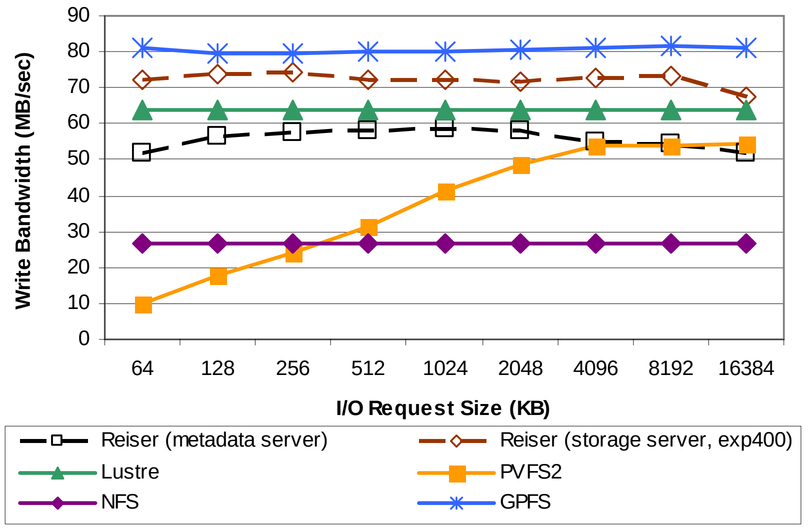

Fig. 5. Single PPC970 node write bandwidth by request size

5.1 Single Node Performance with IOZone

The single client bandwidth for each file system was obtained using the iozone filesystem benchmark [7]. The objective of these tests was to evaluate each filesystem's performance with varying I/O request block sizes for very large files. These tests were executed on client nodes of both architectures in our computing environment. Additional iozone tests were performed on the storage and metadata servers in our environment in order to determine the baseline performance for our supporting storage hardware. The hardware performance analysis was done by writing a file larger than the amount of RAM available for a system in order to avoid any caching effects. The client tests wrote 32 GB files.

The baseline performance of the individual storage system components, locally running the ReiserFS filesystem, is indicated on each of the figures for comparison to the parallel filesystem products evaluated. The IBM x345 storage server, with the attached IBM EXP400 RAID array, exhibited an average read bandwidth for all I/O request sizes of 82.5 MB/sec with a 1.44 MB/sec standard deviation and a write bandwidth of 72.3 MB/sec with a standard deviation of 2.0 MB/sec. The metadata server exhibited an average read bandwidth for all I/O request sizes of 50.8 MB/sec with a 0.19 MB/sec standard deviation and a write bandwidth of 55.6 MB/sec with a standard deviation of 2.7 MB/sec.

The read bandwidth we observed from each file system evaluated on the Xeon clients reveals several interesting points (see Fig. 2). The GPFS, Lustre, and TerraFS filesystems follow the bandwidth trends of the storage server to varying degrees. GPFS attained an average read bandwidth of 92 MB/sec with a 1.86 MB/sec standard deviation, Lustre attained an average read bandwidth of 78.4 MB/sec with a 1.5 MB/sec standard deviation, and TerraFS attained an average read bandwidth of 69.4 MB/sec with a 1.97 MB/sec. NFS exhibited a consistent read performance of 30 MB/sec, with a 0.96 MB/sec standard deviation, for reads of all sizes. PVFS2 exhibited a volatile read performance ranging from 43.75 MB/sec to 10.85 MB/sec. For the tests performed, PVFS2's read bandwidth eventually stabilized at 30.1 MB/sec.

While some filesystems running on Xeon clients were able to attain a write bandwidth faster than the baseline performance of a single storage server node, most were not able to break this threshold (see Fig. 3). Lustre and GPFS exceeded this threshold and achieved better single client performance than a single client writing directly to a single RAID array. GPFS produced an average write bandwidth of 100.66 MB/sec, with a standard deviation of 1.69 MB/sec. Lustre produced an average write performance of 102 MB/sec with a standard deviation of 0.2 MB/sec for all write sizes analyzed. TerraFS achieved an average write performance of 51.0 MB/sec with a standard deviation of 0.88 MB/sec. NFS achieved an average read performance of 27.0 MB/sec, with a 0.027 MB/sec standard deviation. PVFS2 performed poorly for small I/O request sizes, with a low bandwidth of 10MB/sec, but gradually increased to 51.0 MB/sec for block sizes of 4 MB to 16 MB.

In our experimental setup, one item of concern was the use of LVM with all filesystems except for GPFS. We prefer to use LVM on our RAID devices for ease of configuration in our occasionally reconfigured computing. We were concerned that using LVM with Lustre, TerraFS, and PVFS2 and not with GPFS would skew the results, so we ran additional bandwidth tests using Lustre to analyze this independent variable. As expected, when the partitions on our storage servers were configured with LVM, we noticed a slight decrease in performance. Comparisons we made between Lustre OSTs configured to use the raw storage device and Lustre OSTs configured to use LVM partitions indicate that LVM configuration incurs an approximate 2 MB/sec decrease in performance. Our results show all filesystems using LVM except for GPFS. In cases where GPFS outperforms Lustre by a small bandwidth amount, this configuration irregularity may be partially responsible.

Our next battery of tests evaluated each filesystem's read performance on the PPC970 architecture (see Fig. 4). The general performance trends are similar to the read performance of the Xeon clients. A noticeable difference between the read performance measured on the Xeon and PPC970 nodes is that the PPC970 clients do not attain the performance level of the Xeon clients in this set of experiments. The GPFS clients attained the best read performance on the PPC970 system, which is 80.77 MB/sec with a 0.47 MB/sec standard deviation. The GPFS results observed on the PowerPC system are 12 % slower than the Xeon clients. The Lustre clients on the PPC970 architecture attained an average read bandwidth of 57.4 MB/sec with a 0.17MB/sec standard deviation, which is 26.7% slower than those on the Xeon architecture. The PVFS2 performance fluctuated from 15.8 MB/sec to 26 MB/sec. The PVFS2 clients on the Xeon nodes also outperformed the PPC970 systems by 34.2% to 6.75%. NFS was the only file system tested where the read performance was consistent across platforms with an average read bandwidth of 29.9 MB/sec.

Our final test using the iozone benchmark measured the single-client write bandwidth for PPC970 clients for each product evaluated (see Fig. 5). As with the read performance, the measured performance trends were similar to those on the Xeon clients. Again, the Xeon clients outperformed the PPC970 clients. The write performance of the GPFS client is 80.5 MB/sec, with a standard deviation of 0.77 MB/sec. Our data indicates that GPFS clients on the PowerPC system are 20% slower than the Xeon clients. The Lustre clients demonstrated an average write bandwidth of 63.7 MB/sec, with a standard deviation of 0.1 MB/sec, and were 37.9% slower than the write performance on the Xeon system. Similar to the PVFS2 client on the Xeon system, the client on the PPC970 systems started with low performance and gradually increased to 53 MB/sec, which is 5% slower than the stabilized write bandwidth measured on the Xeon nodes. NFS's write performance on the PPC970 system is consistent with the Xeon nodes with an average write bandwidth of 26.6 MB/sec.

5.2 Parallel Performance with the NCAR caggreIO and metarates Benchmarks

For our parallel performance analysis, we employed two benchmarks used by NCAR for storage procurement. The first benchmark, caggreIO [1], measures aggregate file read and write rates with an increasing number of simultaneous clients. The benchmark uses direct system calls to read and write files in 128MB chunks; we read and write files at least 50% larger than the amount of RAM available in a node. The second benchmark, metarates [2], measures metadata manipulation rates. This application uses system calls to create, close, and stat exactly 10,000 files per node. We ran these tests on both the Xeon and PCC970 clusters using each filesystem.

The first sequence of average aggregate bandwidth tests was performed on the Xeon cluster. Lustre exhibits the best aggregate write performance followed by GPFS, PVFS2 and TerraFS, which show similar performance for larger processor counts (see Fig. 6). In terms of read bandwidth, GPFS, PVFS2, and Lustre alternately produced the best aggregate write bandwidth followed by TerraFS (see Fig. 7). All of the parallel systems considered here outperformed NFS.

We also examined the impact of using Linux channel bonding on the storage servers for both Lustre and PVFS2. The graph lines labeled "1Gbps" show performance with only a single Gigabit Ethernet connection instead of two bonded links. For these tests on the Xeon nodes, we also used the standard 2.4.26 and Lustre RHEL 2.4.x kernel instead of the SLES 9 2.6.x kernel. The introduction of Linux channel bonding drastically improved the performance of PVFS2 for both reads and writes, and even increased the bandwidth to be competitive with Lustre. Channel bonding did not substantially improve the performance of Lustre. CFS indicated that this is expected and suggested that Lustre's optimal bandwidth could be obtained by using dual unbonded interfaces with Lustre itself managing aggregation. As we wanted to perform all tests with a single network configuration, we did not fully investigate this option but intend to do so in the future.

The second sequence of aggregate bandwidth tests was performed on the PPC970 cluster. Lustre provided the best write performance (see Fig. 8). The read tests produced more interesting results. PVFS2 with channel bonding was directly competitive with Lustre from np=4 to np=7 and both exhibited a slightly decreasing read bandwidth with increasing clients in that range (see Fig. 9). We believe that this is due to the selection of blades from chassis units and the chassis switch performance. Otherwise, Lustre provided the best performance, and as the number of clients increased the performance of PVFS2 with channel bonding gradually reduced to the performance of PVFS2 without.

Our final analysis examined how well each filesystem handles metadata intensive applications using the metarates benchmark. We ran the benchmark in two configurations, with the first using a single directory for all clients and the second using a unique directory for all clients. The benchmark measures the amount of time for a certain number of clients to create 10,000 files. In both configurations, Lustre exhibits the best overall performance, while PVFS2 exhibits the slowest file creation rate.

When operating in a single directory, filesystem metadata operations are limited by the locking scheme protecting the structure used to store directory entries. As a server function, this appears to be generally independent of the client architecture (see Fig. 10). Lustre approached a file creation rate of 1027 files/s, NFS sustained an average of 374 files/s, and PVFS2 sustained an average of 25 files/s. Of particular note is the sudden drop in metadata performance demonstrated by TerraFS and GPFS when operating with multiple clients. When running on a single processor, TerraFS created files at around 204 files/s, but when running with more than one client, the file creation rate dropped to around 10 files/s. Similarly, GPFS dropped from 959 files/s to about 167 files/s. We believe that this is because both systems use inode-based filesystems to store metadata and require excessive synchronization.

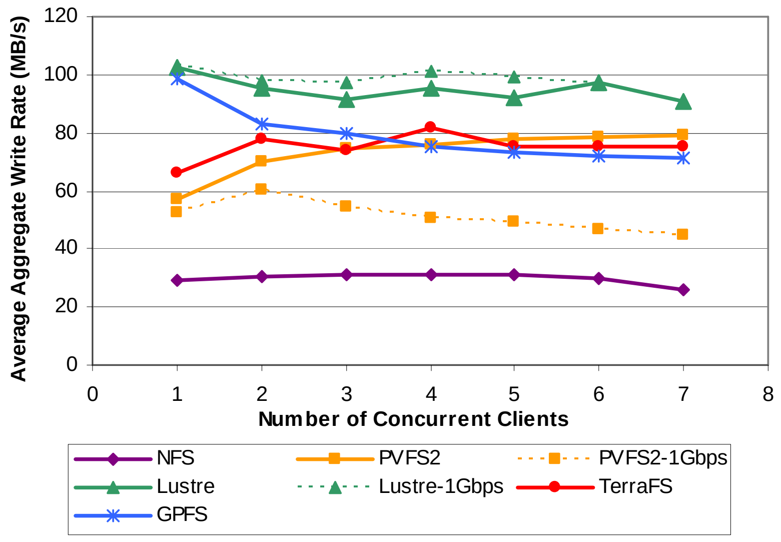

Fig. 6. Xeon cluster average aggregate write bandwidth by number of clients

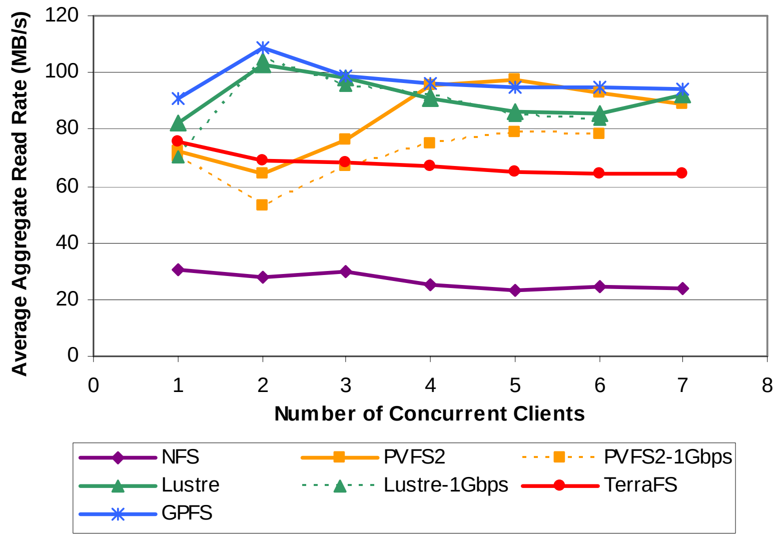

Fig. 7. Xeon cluster average aggregate read bandwidth by number of clients

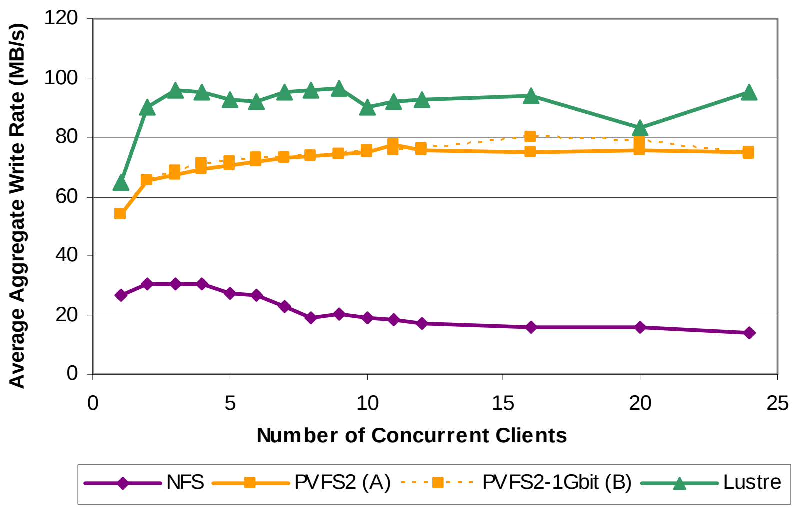

Fig. 8. PPC970 cluster average aggregate write bandwidth by number of clients

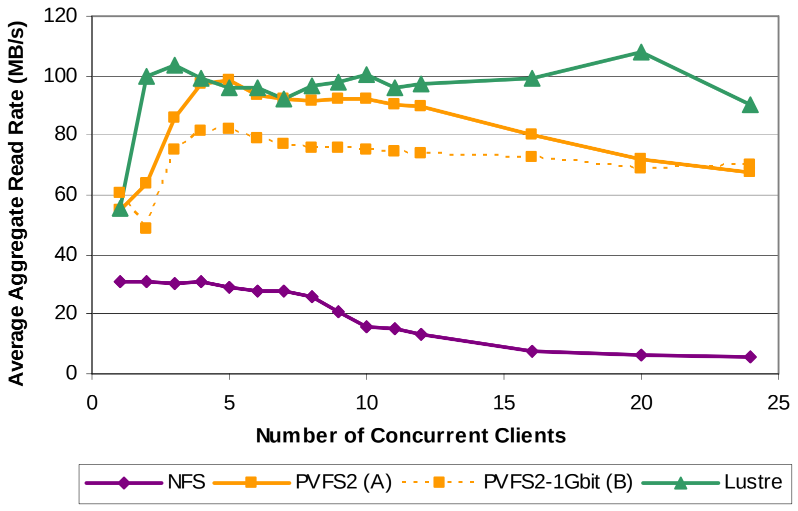

Fig. 9. PPC970 cluster average aggregate read bandwidth by number of clients

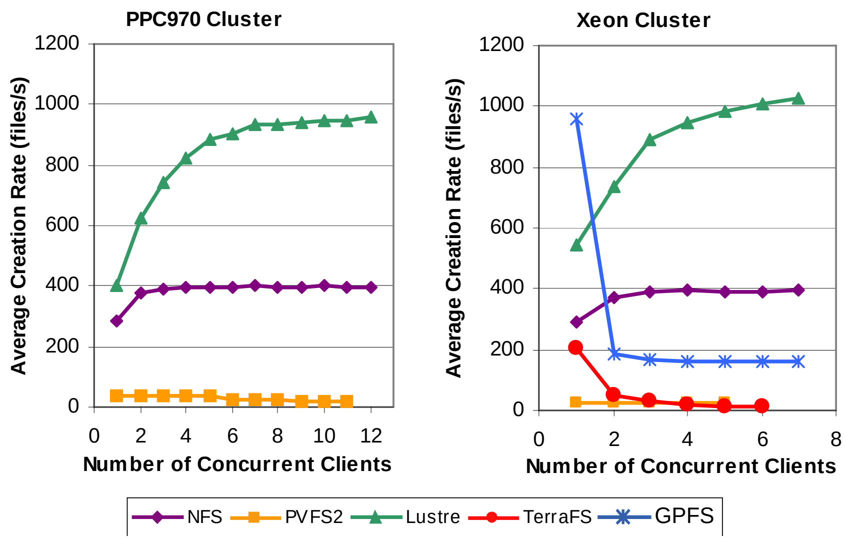

Fig. 10. Average aggregate file creation rate in a single directory by number of clients

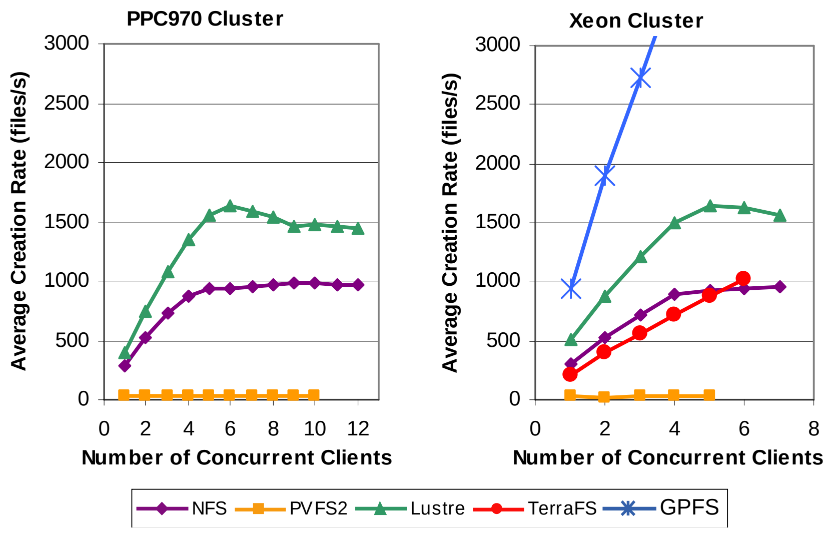

Fig. 11. Average aggregate file creation rate in a unique directory for each client by number of clients

All filesystems provide much better performance when performing metadata operations in unique directories instead of single directories (see Fig. 11). Lustre peaked at 1646 files/s, NFS peaked at 953 files/s, and PVFS2 maintained a creation rate of about 25 files/s. While the file creation rates demonstrated by Lustre, NFS, and PVFS2 appear to level out with increasing clients, TerraFS and GPFS scaled very well in our testing range. TerraFS started at 206 files/s and increased by an average of 186 files/s per processor. GPFS started at 945 files/s with an average increase of 855 files/s per processor. Just as its use of an inode-based filesystem may have reduced performance in the single-directory test case, here we suspect that this treatment of inodes as data blocks may have allowed for perfectly parallel operation with no synchronization.

6 Future Work

Having completed this initial examination of parallel filesystem products for use in our heterogeneous datacenter, our immediate plan is to select one or two of these filesystems and deploy them for production use. At the present time, Lustre and GPFS are the most attractive candidates, but we only have two storage servers and previously encountered problems attempting to run GPFS on the Lustre kernel. As the Lustre patches are being integrated into the SLES kernel and IBM supports GPFS on all official SLES kernel releases, we anticipate that this will be possible in the near future. We would also like to examine reliability through redundancy without specialized hardware, leveraging internal filesystem redundancy features such as block replication and parity.

After implementing a shared parallel filesystem for use in our datacenter at the University of Colorado, we would like to investigate the possibility of connecting to CU filesystems from the BlueGene/L system at NCAR’s Mesa Lab facility. This will involve extensive security policy and mechanism conversations between the two institutions. Nevertheless, as many NCAR BlueGene/L users frequently develop software on our CU systems, remote accessibility may be particularly beneficial. We intend to support Grid-based connectivity in the meantime.

As our goal is to provide a single centralized parallel filesystem for multiple clusters, we are currently interested in filesystems that function well in a least common denominator situation using gigabit Ethernet as the I/O interconnect. Later, we would also like to examine a subset of these filesystems using multiple high-performance interconnects (e.g., 10-GigE, Myrinet, and Infiniband) between the computational clusters and the storage clusters. Two issues make this more expensive and difficult. First, specialized interconnects are much more expensive per-port than gigabit Ethernet. Some organizations will require heterogeneous interconnect support because providing a specialized network connection to all hosts may not be financially feasible. Second, in order to multi-task a specialized interconnect for both MPI and storage traffic, the system must be adequately planned before installation. For example, many installations purchase specialized interconnect switches with exactly the capacity of their compute cluster, making adding I/O nodes or trunking to a storage cluster extremely cost prohibitive. While a single network solution would be ideal, the cost and requirements of different types of communication may make heterogeneous solutions more attractive in the near future.

7 Suggestions for Filesystem Developers

Linux systems produce no fury greater than when a block-based filesystem suddenly becomes inaccessible and the pending I/O operations cannot be fulfilled. In our testing, filesystem server daemons were frequently stopped and started as we concluded testing one system and commenced testing another. Even in production environments, servers occasionally crash unexpectedly, and gracefully surviving these unanticipated transients can only be provided by software developers who enjoy writing robust error handling routines. In all of filesystems we tested, we encountered a substantial amount of unpleasant filesystem behavior in the face of transients, and we hope that future filesystem releases are designed to handle error situations with finesse. We have compiled a wish list for future filesystem products that would provide for a more pleasant system administrator experience.

First, filesystems should never be unresponsive to standard system administration commands even in failure conditions. We encountered situations where executing "ls –la /mnt" or "df" would hang because a server hosting a parallel filesystem had been rebooted. Similarly, benchmark processes running at the time of the server crash would exist in the defunct state as zombies immortal even to "kill –s 9" commands until the clients were rebooted.

Second, filesystems should support clean normal and emergency termination, and never interrupt system shutdown procedures. Many of the filesystems we tested seem to support only "start" directives and provide various excuses why directories cannot be unmounted, client processes cannot be terminated, and the server daemons cannot be stopped. One filesystem we tested included kernel patches that made rebooting systems using "reboot" fail. With this filesystem running, the operating system could not be shut down normally, and we had to use remote power control to restart these systems!

Finally, like industrial control systems, filesystems should support an Emergency Stop feature. When invoked, Emergency Stop would simply drop pending I/O operations and kill the services. Instead of waiting for a failure condition to be resolved, just reject all pending I/O requests and fail gracefully. This way, administrators working on recovering from a parallel filesystem crash would not need to reboot client nodes to return a filesystem to production. Similarly, it would be very pleasant if a parallel filesystem could automatically detect critical failures and make the filesystem inaccessible on all client nodes in such cases. Current behavior is much more nebulous, occasionally leaving users and administrators wondering if the filesystem is available.

8 Conclusions

All of the parallel filesystems we tested outperformed our current NFS-based installation. However, this improvement came at a price. Perhaps the most frustrating aspect of the four parallel filesystems we tested is their dependence on specific kernel versions. Both Lustre and GPFS, commercially supported parallel filesystems intended for high performance computing, essentially require current commercially supported Red Hat or SuSE kernels. They do not support the default kernel.org Linux kernels. This is problematic because, though our newer equipment uses supported kernels, we prefer to continue to use noncommercial kernels on our previously purchased production cluster systems due to cost considerations. Although the TerraFS policy of custom-building kernels based on our specifications was a pleasant relief for the system administration staff, not being *allowed* to build our kernels in-house is unacceptable from a security, maintenance, and research standpoint. Specific kernel version requirements stall critical kernel exploit patches and make building kernels for specialized hardware difficult (and we are still working to return our Dolphin interconnect to production). In the case of Lustre, we are confident that CFS efforts to integrate their patches into the SuSE and kernel.org kernels will eventually eliminate this problem. However, until such patches are propagated to the Linux community by all vendors, the kernel dependencies introduced by these filesystems complicate administration and are quite inconvenient.

We examined the functionality of several currently available parallel filesystems on clusters in our heterogeneous Intel and PPC970 datacenter. Lustre, PVFS2, and GPFS functioned well on all of our systems and provided roughly equivalent performance on our suite of benchmark tests. PVFS2 and GPFS provide the best right-out-of-the-box administrator experience and did not require kernel changes. Lustre and TerraFS require more setup overhead and troubleshooting. In the near future we intend to select a parallel filesystem for use in our environment and subject it to a test much more difficult than any benchmark: real users.

9 Acknowledgements

We would like to acknowledge the assistance of the following organizations and individuals: CU Boulder – Theron Voran. NCAR – Bill Anderson, Rich Loft, and George Fuentes. IBM – Gautam Shah, Ray Paden, Barry Bolding, and Rajiv Bendale. CFS – Jeffrey Denworth, Phil Schwan, and Jacob Berkman. TerraScale and Western Scientific – Tim Wilcox and Dave Jensen. University of Colorado computer time was provided by equipment purchased under NSF ARI Grant #CDA-9601817, NASA AIST grant #NAG2-1646, NSF sponsorship of the National Center for Atmospheric Research, and a grant from the IBM Shared University Research program.

References

1. Anderson, Bill. caggreIO: A program that measures aggregate I/O rates to and from filesystems. Source code. 18 October 2004.
2. Anderson, Bill. metarates: A program that measures aggregate metadata transaction rates. Source code. 18 October 2004.
3. Braam, P. The Lustre Storage Architecture. http://www.lustre.org/docs/lustre.pdf, 2004.
4. Braby, R. L., Garlick, J. E., and Goldstone, R. J. "Achieving Order through CHAOS: the LLNL HPC Linux Cluster Experience." The 4th International Conference on Linux Clusters: The HPC Revolution 2003, San Jose, CA, June 2003.
5. Butler, M. "Petabyte Production Storage Environments and File Systems." NCSA Storage Enabling Technologies presentation, Supercomputing 2004, November 2004. http://www.ncassr.org/projects/storage-sec/papers/mbutlerSC04slides.pdf
6. IBM. General Parallel File System, 2005. http://www-1.ibm.com/servers/eserver/clusters/software/gpfs.html
7. Iozone filesystem benchmark. http://www.iozone.org
8. Margo, M. W., Kovatch, P. A., Andrews, P., and Banister, B. "An Analysis of State-of-the-Art Parallel File Systems for Linux." The 5th International Conference on Linux Clusters: The HPC Revolution 2004, Austin, TX, May 2004.
9. Ramachandran, H. *Design and Implementation of the System Interface for PVFS2*. M.S. Thesis, Clemson University, Clemson, South Carolina, December 2002.
10. Schmuck, F. and Haskin, R. "GPFS: A Shared-Disk File System for Large Computing Clusters." Proceedings of the First Conference on File and Storage Technologies (FAST), January 2002.
11. Sim, A. J. Gu, A. Shoshani, V. Natarajan. DataMover: robust terabyte-scale multi-file replication over wide-area networks. Proceedings of the 16th International Conference on Scientific and Statistical Database Management, 403, 21 June 2004.
12. Storage Resource Broker (SRB), 2004. http://www.npaci.edu/dice/srb
13. Terrascale Technologies. Terragrid Overview, 2005. http://www.terrascale.com/prod_over_e.html