LA-UR- 10-01765

Approved for public release; distribution is unlimited.

Title:	Integration Experiences and Performance Studies of A COTS Parallel Archive System
Author(s):	Hsing-bung Chen, Gary Grider, Cody Scott, Milton Turley, Aaron Torrez, Kathy Sanchez, John Bremer HPC division Los Alamos National Laboratory Los Alamos, New Mexico 87545, USA
Intended for:	IEEE Cluster 2010 Conference

Form 836 (7/06)

Integration Experiences and Performance Studies of A COTS Parallel Archive System

Hsing-bung Chen[3], Gary Grider[1], Cody Scott[2], Milton Turley[2], Aaron Torrez[2],
Kathy Sanchez[2], John Bremer[2]
HPC-DO[1], HPC-3[2], HPC-5[3]
Los Alamos National Laboratory
Los Alamos, New Mexico 87545, USA
hbchen@lanl.gov

Abstract

Current and future Archive Storage Systems have been asked to (a) scale to very high bandwidths, (b) scale in metadata performance, (c) support policy-based hierarchical storage management capability, (d) scale in supporting changing needs of very large data sets, (e) support standard interface, and (f) utilize commercial-off-the-shelf(COTS) hardware. Parallel file systems have been asked to do the same thing but at one or more orders of magnitude faster in performance. Archive systems continue to move closer to file systems in their design due to the need for speed and bandwidth, especially metadata searching speeds such as more caching and less robust semantics. Currently the number of extreme highly scalable parallel archive solutions is very small especially those that will move a single large striped parallel disk file onto many tapes in parallel. We believe that a hybrid storage approach of using COTS components and innovative software technology can bring new capabilities into a production environment for the HPC community much faster than the approach of creating and maintaining a complete end-to-end unique parallel archive software solution. In this paper, we relay our experience of integrating a global parallel file system and a standard backup/archive product with a very small amount of additional code to provide a scalable, parallel archive. Our solution has a high degree of overlap with current parallel archive products including (a) doing parallel movement to/from tape for a single large parallel file, (b) hierarchical storage management, (c) ILM features, (d) high volume (non-single parallel file) archives for backup/archive/content management, and (e) leveraging all free file movement tools in Linux such as copy, move, ls, tar, etc. We have successfully applied our working COTS Parallel Archive System to the current world's first petaflop/s computing system, LANL's Roadrunner, and demonstrated its capability to address requirements of future archival storage systems.

Keywords: Archive Storage System, Parallel File System, Storage Hierarchy, Parallel Data Movement, Hierarchical Storage Management, Parallel Archive, Cluster Computing, Parallel I/O

Integration Experiences and Performance Studies of A COTS Parallel Archive System

Hsing-bung Chen[3], Gary Grider[1], Cody Scott[2], Milton Turley[2], Aaron Torrez[2],
Kathy Sanchez[2], John Bremer[2]
HPC-DO[1], HPC-3[2], HPC-5[3]
Los Alamos National Laboratory
Los Alamos, New Mexico 87545, USA
hbchen@lanl.gov

Abstract

Current and future Archive Storage Systems have been asked to (a) scale to very high bandwidths, (b) scale in metadata performance, (c) support policy-based hierarchical storage management capability, (d) scale in supporting changing needs of very large data sets, (e) support standard interface, and (f) utilize commercial-off-the-shelf(COTS) hardware. Parallel file systems have been asked to do the same thing but at one or more orders of magnitude faster in performance. Archive systems continue to move closer to file systems in their design due to the need for speed and bandwidth, especially metadata searching speeds such as more caching and less robust semantics. Currently the number of extreme highly scalable parallel archive solutions is very small especially those that will move a single large striped parallel disk file onto many tapes in parallel. We believe that a hybrid storage approach of using COTS components and innovative software technology can bring new capabilities into a production environment for the HPC community much faster than the approach of creating and maintaining a complete end-to-end unique parallel archive software solution. In this paper, we relay our experience of integrating a global parallel file system and a standard backup/archive product with a very small amount of additional code to provide a scalable, parallel archive. Our solution has a high degree of overlap with current parallel archive products including (a) doing parallel movement to/from tape for a single large parallel file, (b) hierarchical storage management, (c) ILM features, (d) high volume (non-single parallel file) archives for backup/archive/content management, and (e) leveraging all free file movement tools in Linux such as copy, move, ls, tar, etc. We have successfully applied our working COTS Parallel Archive System to the current world's first petaflop/s computing system, LANL's Roadrunner, and demonstrated its capability to address requirements of future archival storage systems.

Keywords: Archive Storage System, Parallel File System, Storage Hierarchy, Parallel Data Movement, Hierarchical Storage Management, Parallel Archive, Cluster Computing, Parallel I/O

1 Introduction

Modern high performance computing/data intensive computing involves computing, organizing, moving, visualizing, and analyzing massive amount of data from various scientific application domains. The unbounded increase in the computation and data requirements of scientific applications has necessitated the use of widely distributed compute and storage resources to meet the demand. Efficient and reliable access to data sources and archiving destinations in such an environment brings new challenges [1][2][3][6].

The existing solution for HPC archive storage systems (Figure-1) is far from catching up with the growing requirements of archive I/O bandwidth and archive system's scalability [1][3][6][9].

Currently the number of extreme highly scalable parallel archive solutions is very small especially those that will move a single large highly striped parallel disk file to many tapes in parallel [2]. We believe that a hybrid approach of using commercial-off-the-shelf (COTS) components and innovative software technology can bring new usable capabilities to the HPC community much faster than the approach of creating and maintaining a unique software parallel archive solution [1][2].

Disk-based parallel file systems for clusters are increasingly using multiple software "mover" components to accomplish parallel data transfers [7][10]. These data movers, most often, function by exporting access to unique, independent data stores. Classic HSM methodologies also employ multiple data movers, but curiously, usually to support more connections, not parallel connections. HSM is a data storage technique, which automatically moves data between high-cost and low-cost storage media [6]. ILM is the practice of applying certain policies to the effective management of information throughout its useful life [6]. Lessons learned from recent file systems work could be used to simplify the back-end data path in HSMs by using a metadata service to maintain tape and location layout information. Perhaps a realistic core set of requirements for archive products for science use might be a stepping-off

point to an acceptable interface to HSM software with a usable lifetime greater than a decade. The marriage of modern parallel file system designs with a subset of classic HSM software could yield a seamless infinite global parallel file system solution, which could eliminate the need for a separate parallel or serial archive capability.

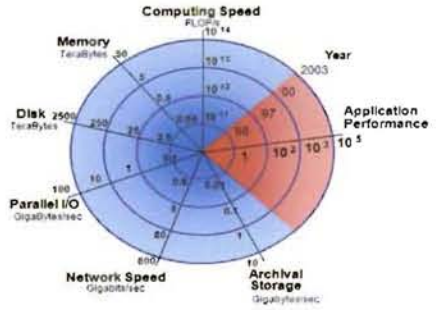

Figure 1: The DOE Advanced Strategic Computing Initiative Program published this Kiviat diagram that shows parallel file systems scaling performance at an order of magnitude faster than parallel archives

The integration of Archive/HSM (Hierarchical Storage Management) / ILM (Information Lifecycle Management) / Parallel File Systems functions is a possible solution to meet the above requirements. HSM transparently handles the data movement between storage hierarchies. ILM comprises the policies, processes, practices, and tools used to align the business value of information with the most appropriate and cost effective IT infrastructure from the time information is conceived through its final disposition. Parallel File Systems provide a fast, efficient, and scalable I/O capability.

In this paper we try to integrate COTS global parallel file systems and a standard backup/archive product with a very small amount of additional user space code to provide a scalable & parallel solution that overlaps highly with current niche parallel archive product(s) including (a) doing parallel movement to/from tape for s single file, (b) hierarchical storage management, (c) ILM features, (d) high volume (non-single parallel file based) archives for backup/archive/content management, and (e) leveraging all free file movement/management tools in Linux such as copy, move, ls, tar. etc.

The rest of this paper is organized as follows. Section 2 introduces some background information. We address issues, motivation and leverage in Section 3. Our proposed COTS Parallel Archive System is presented in Section 4. In Section 5, we discuss performance data from the Roadrunner Open Science projects. Then, we present experience and observed issues of our COTS Parallel Archive System on the current world's second fastest supercomputer, the Roadrunner cluster, in Section 6. Finally, we conclude our contribution and future works in Section 7.

2 Background

In order to understand parallel archive systems, it is important to understand some basic concepts in the parallel archive area. Parallel file systems are normally not used as archives in HPC environments. Typically parallel file systems are used as a fast storage place to stage data to and from a supercomputer. Often parallel file systems are considered to be scratched in nature meaning that this storage is not intended for long-term storage, it is only used as temporary staging and working storage. Parallel archives are used typically as the long-term storage for an HPC site. The largest of HPC sites need parallel archives as opposed to non-parallel archives to allow for fast enough data movement of large files to and from the archive. The parallel archive is usually considered more stable and highly dependable.

2.1 Parallel File Systems & Parallel I/O

Parallel I/O means the operation of multiple file read/write at the same time. It is the common feature of modern Parallel File Systems. One particular instance is parallel writing of data to disk; when file data is stored across multiple disks, for example in a RAID array, one can store multiple parts of the data at the same time, thereby achieving higher write speeds than with a single device. The functionality and capability of Parallel data I/O movement is proven to be the key factor for the success of high performance computing systems [1][3][22].

2.2 HSM

Hierarchical Storage Management (HSM) is a data storage technique which automatically moves data between high-cost and low-cost storage media. HSM systems exist because high-speed storage devices, such as hard disk drive arrays, are more expensive (per byte stored) than slower devices, such as optical discs and magnetic tape drives. While it would be ideal to have all data available on high-speed devices all the time, this is prohibitively expensive for many organizations. Instead, HSM systems store the bulk of the enterprise's data on slower devices, and then copy data to faster disk drives when needed. In effect, HSM turns the fast disk drives into caches for the slower mass storage devices. The HSM system monitors the way data is used and makes choices based on specific criteria as to which data can safely be moved to slower devices and which data should stay on the fast devices [19].

2.3 ILM – Information Life cycle Management

Information life cycle management (ILM) is a comprehensive approach to managing the flow of an information system's data and associated metadata from creation and initial storage to the time when it becomes obsolete and is deleted. Unlike earlier approaches to data storage management, ILM involves all aspects of dealing with data, starting with user practices, rather than just automating storage procedures, as a hierarchical storage manager product like the one HSM does. Also in contrast to older systems, ILM enables more complex criteria for

storage management than data age and frequency of access [20]. ILM typically organizes data into separate tiers according to specified policies, and automates data migration from one tier to another based on those criteria. As a rule, newer data, and data that must be accessed more frequently, is stored on faster, but more expensive storage media, while less critical data is stored on less expensive and slower media. However, the ILM approach recognizes that the importance of any data does not rely solely on its age or how often it is accessed. Users can specify different policies for data that declines in value at different rates or that retains its value throughout its life span [20].

2.4 Non-Parallel vs. Parallel Archive Systems

Non-parallel hierarchical storage systems provide parallelism for multiple files through the storage hierarchy, but single files do not move in parallel. In parallel archives, single files move in parallel all the way through the storage hierarchy. The market for non-parallel archives is huge with tens of thousands of sites using these solutions. The market for parallel archives is extremely small with tens of sites needing these solutions. The market for parallel file systems is larger than the market for parallel archives by at least one or two orders of magnitude. Due to the market demands, the number of extreme highly scalable parallel archive solutions is very small. The cost of maintaining specialized highly scalable parallel archive is high due to the very small market at least for those institutions that are paying for doing this maintenance [2].

2.5 Parallel Archives That Do Not Leverage Parallel File Systems as Their First Tier of Storage

All parallel archives and even non-parallel archives utilize HSM as described above to store data and move it to cheaper or more appropriate storage devices over time. Some archive solutions utilize file systems and even parallel file systems as their first tier of storage, others do not. Access methods for utilizing these types of parallel archives that do not use file systems as their first tier typically are full file movement based and require custom interfaces or utilize file transfer interfaces like "ftp" or "scp".

2.6 Parallel Archives That Leverage Parallel File Systems as Their First Tier of Storage

There are parallel archives and even non-parallel archives that utilize file systems and parallel file systems as their first tier of storage in their HSM. Due to the fact that the top tier of storage is a file system, most sites utilize this fact to use the very nice file system interface as their interface to the archive. This is unlike non file system leveraging archives where access methods are typically data transfer programs like "ftp" and "scp." NFS is utilized to export the archive interface to client machines in the parallel file-system paradigm. This makes for a very rich user interface to the archive which is nice for users but comes with its own set of issues for managing the archive. The typical problem this rich interface presents the archive is that it makes it simple for users to use tools like "grep", which scans files for strings, which would be very difficult to do for data that is on removable media.

3 Issues, Motivation, and Leverage of using COTS Parallel Archive System

3.1 Issues When Using a Parallel File System as the First Tier in a Parallel Archive Storage System

There are several issues when we use a parallel file system as the first tier in a parallel archive storage system. We have itemized those issues as follows [1] [2] [3] [4][5] [6][8] [11] [12][13][14] [17] [18] [19] [20]:

1) Due to NFS access you have "the grep from &*&(*&",
2) No way to get immense file from HSM disk to parallel tapes and back (single stream of tapes),
3) No parallel copy/tree walker to copy scratch file system to/from archive storage system,
4) Due to NFS, no way to query target storage pools for archive placement,
5) Due to NFS, no way to do efficient ordered retrieval from many tape files when copying back from archive to scratch,
6) Need ILM/storage pool management/policy on archive parallel file system (multiple copies, smart placement etc.),
7) Need powerful ILM/storage pool management/policy on tape back end system (multiple copies, remote copies, smart placement),
8) Need data parallelism on tape back end system (data cannot flow through single HSM server etc),
9) Need excellent metadata handling on tape back end system,
10) Need excellent scalable parallel file system data/metadata for archive,
11) Need good integration between archive parallel file system and backend tape storage system,
12) Eliminate garbage collection etc. to avoid having to sync archive tape back end and archive parallel file system metadata,
13) Need robust HSM (file system and backend) metadata backup system, and
14) Need ability to handle massive amounts of small, millions of medium, and few enormous files.

Those classical issues mentioned above post a new challenge for designing and developing a parallel archive storage system.

3.2 Motivation

As was mentioned above in the introduction, parallel archives and parallel file systems are being asked to scale in similar ways but parallel file systems have been forced to scale to one or more orders of magnitude faster performance. Additionally, file systems and archives are growing together in design due to the scalability needs, especially in metadata management, integrity, and loosened metadata semantics. These facts combined with marketing

and cost realities such as the high cost of maintaining a parallel archive capability for an extremely small market compared to a similar cost for parallel file systems for a much larger market, gives us cause to wonder if more leverage of parallel file systems to provide parallel archive is possible and makes sense. Ultimately, the question we are trying to answer in this pilot project is: can we leverage parallel file system and non parallel archive COTS solutions that are highly leveragable to build a highly leveraged parallel archive with very little unique code needed to provide the parallel archive service. If this can be done, a large savings in providing this service could possibly be realized.

3.3 Leveraging

Our premise is that the parallel archive systems can benefit from leveraging the functions and capabilities of the parallel file-systems and appeal to a broader market. The following is a list of possible technologies and economic facts that could be utilized in our pursuit to build a more leveraged parallel archive:

1) Disk is becoming more competitive with tape over time for a larger portion of archival data [7][10][14],
2) Moderate and growing volume Global Parallel File Systems market,
   a. Scalable bandwidth and metadata
   b. Growing use of Global Parallel File Systems for moderate scale HPC
3) HSM and ILM features in file systems and archives (driven by huge industry mandates like HIPPA/S-0x etc.),
4) High volume (non single parallel file) archives for backup/archive/content mgmt, and
5) Leverage al free file movement/management tools in Linux, copy, move, ls, tar, etc.
   a. a well known file management environment
   b. get scp, sftp, and web/gui file management for free etc.

4 Proposed COTS Parallel Archive System

Over the last five years, the DOE's Advanced Simulation Computing (ASC) Program at LANL has wanted to run a pilot program to demonstrate and study the viability of a high performance commodity based parallel archive. In order to test the new parallel archive at the scale of HPC environments, the new parallel archive project was tested with LANL's Roadrunner cluster [22] while the cluster was in the initial testing phase. We chose IBM GPFS for the parallel file system because of the new ILM features. We chose Tivoli Storage Manager (TSM) because we were already using it in house and liked the existing functionality with GPFS. The following features were designed, developed, and integrated into the COTS Parallel Archive System (Figure 2):

1) Build a parallel tree walker and copy user space utility,
2) Add storage pool (stgpool) [17][18][19] support (using file system API),
3) Create an efficient ordered file retrieval utility (using dmapi API and back end tape system query),
4) Add support for ILM stgpool features,
5) Add support for ILM stgpool and co-location features in the archive back-end, and
6) Use FUSE to break up enormous files into pieces that can be migrated and recalled in parallel to/from the back end tape system[15][23]

Our proposed COTS Parallel Archive System consists of the frontend system and the backend system.

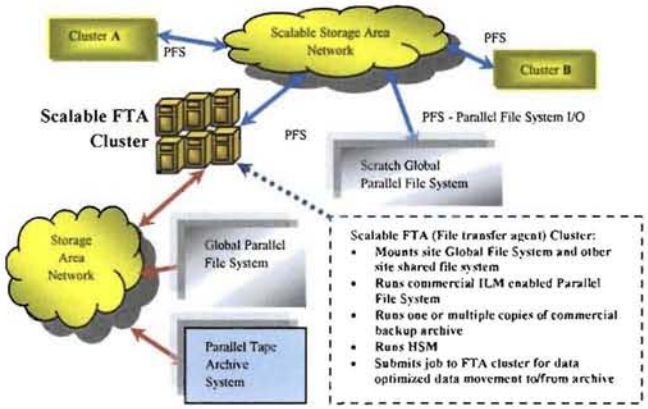

Figure-2: The proposed COTS Parallel Archive System

4.1 Frontend System

Although the archive's base COTS components provide a lot of functionality out of the box, extra tools were necessary to combine GPFS with TSM to create a fully functional high performance parallel archive. A utility to copy data in parallel was introduced to the new parallel archive. In addition, the new parallel archive also needs to be tape aware. It should not treat tapes like it does disk, as this could cause unnecessary tape mounting and seeking.

There can also be problems if users use standard file-system utilities that indiscriminately access files without being tape aware. Deleting files can lead to issues since part of a file is on the file system and another part is on tape. Finally, we have to expect that a user will eventually want to archive extremely large files and we must be able to support to archive those files in parallel. To fulfill all these needs, we have designed and implemented a parallel file/archive software tool as the front-end system.

The frontend system is made of components from PFTool (**P**arallel **F**ile software **Tool**) software and PFTool runtime environment.

4.1.1 PFTool Software System

The PFTool software system diagram is illustrated in Figure- 3. PFTool is built upon MPI and consisted of one Manager process, one OutPutProc process, at least one ReadDir processes, one WatchDog process, at least one Worker process, and some TapeRestore processes (only for

restoring direction). The total number of MPI processes used in PFTool is dynamically adjustable during runtime. The function of each MPI processes is :

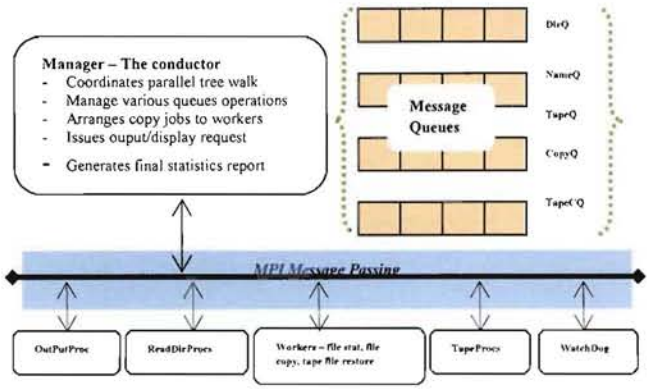

Figure 3: PFTool Software System Diagram

1) Manager process: The Manager is the conductor of PFTool run-time activities: it (a) starts the parallel tree walk, (b) puts exposed directories in the directory queue (DirQ) and assigns a directory traversal job to an available ReadDir process, (c) receives file stat request from Worker processes, (d) puts source files for stating in the name queue (NameQ) and assigns them to available Worker processes for stating, (e) receives stated file information from Worker processes, (f) puts stated file/tape file information into the copy queue (CopyQ) or the tape copy queues (TapeCQ), (g) arranges and lines up the tape restore file information into TapeCQs for tape restoring optimization, (h) assigns regular file copy jobs from CopyQ to available Worker processes, (i) sends tape restoring file copy request to TapeProc processes, (j) receives additional restored tape file copy request from TapeProc processes and assigns them to Workers for further copying from archival parallel file system to scratch parallel file system, (k) asks the OutPutProc to display status and results of PFTool operations, (l) periodically updates PFTool runtime status with the WatchDog, and (m) finalizes parallel archive/restore operations by killing all running MPI processes.
2) OutPutProc process: The OutPutProc handles the output of PFTool operation status and results. Both operations of on-screen display and re-direction to the file are supported.
3) WatchDog process: The WatchDog is a run-time PFTool progress indicator that runs periodically. The WatchDog (a) records the current and historical statistics of PFTool such as total number of files copied, number of files copied in the past "T" minutes, total number of bytes moved, number of bytes copied in the past "T" minutes, (b) indicates the data movement status, and (c) forces the termination of PFTool runtime activities if the "data copy" is stalled without any further progress for a specific amount of time
4) ReadDir process: The ReadDir (a) receives requests from the Manager, (b) exposes directory information, (c) collects exposed directory information, and (d) sends collected file/dub-directory information back to the Manger for further stat processing.
5) TapeProc process: The TapeProc (a) receives requests from the Manager, (b) restores migrated files from tapes to the archival GPFS parallel file system, and (c) sends additional restored tape file copy request to the Manager.
6) Worker process: The Worker (a) receives copy-request jobs from the Manager, (b) moves data to and from the COTS Parallel archive system, (c) sends copy status and results back to the Manager.

All available processes except the Manager keep sending request messages to the Manager and ask for more works. The Manager finalizes the archive/restore operations when there are no jobs in the queues and all processors become available. A performance report is generated after finishing each parallel archive job.

4.1.2 PFTool Runtime Environment

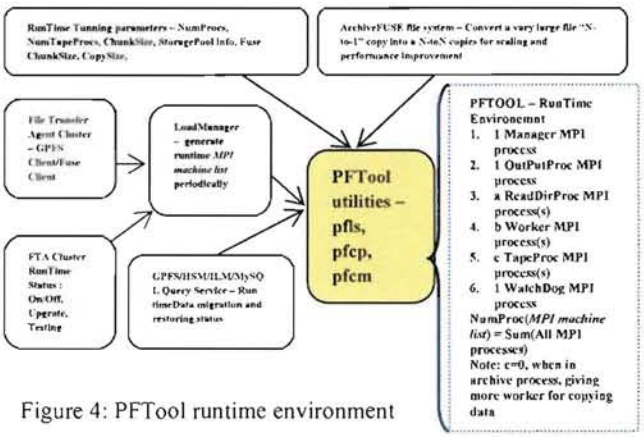

Figure 4: PFTool runtime environment

The PFTool runtime Environment consists of

1) LoadManager – The LoadManager runs periodically for (a) collecting FTA scalable cluster machine CPU workload status, (b) sorting available MPI machine list in ascending order based on current machine CPU workload, and (c) generating a timely MPI machine list.
2) Tape optimization – When files are concurrently archived to multiple tape drives, one of the most important performance problem is how to line up the file and tape resources such that unnecessary tape mounting and un-mounting overhead is minimized. This is a "tape drive thrashing problem." That is the dominant factor of degrading tape restoring performance. When hundreds or thousands files are restored from many tapes concurrently, we try to arrange tape files based on their tape sequential numbers and unique Tape-IDs. The tape files with the same Tape ID are put into a corresponding TapeCQ based on their ascending tape sequential number. We then assign them to any available TapeProc so we can drastically reduce tape drive thrashing overhead and enforce sequential tape read when we are restoring many midsize files.
3) A single large file parallel copy – The size of a single large file is in the range of 10GBs to 100 GBs. We divide a single large file into "N" equal-size sub-chunks and assign them to available Workers such that we can utilize concurrent read/write capabilities of the parallel file system and speedup data movement. Each Worker is copying one chunk of a single large file and N workers copy data in parallel. This is a typical parallel N-to-1 data copy.
4) Very large file parallel copies – A file of size greater than 100 GB is considered a very large file. When archiving very large

files in parallel on many tapes, we encounter problems of (a) N-to-1 parallel I/O overhead [23] and (b) performance impact from tape sequential write operation. To overcome these problems, we built an ArchiveFUSE file system on top of the GPFS file system, and can successfully transfer very large files broken down in to N equal size chunk files and assign them to M workers for parallel write operation. We have successfully converted an N-to-1 parallel I/O operation into an N-to-N parallel I/O operation.

5) Runtime tunable parameters for adjusting PFTool commands runtime performance – We manipulate a list of runtime tunable parameters when issuing each PFTool command. Tunable parameters are (a) number of processes created, (b) number of tape drives used, (c) basic file copy size, (d) storage pool information, (e) Fuse file chunk size used, and (f) tape restoring optimization flag.

4.1.3 Parallel Archive commands supported in PFTool

PFTOOL supports three parallel archive commands. They are

- **pfls** – *using parallel file tree walker and list files in parallel*
- **pfcp** – *using parallel file tree walker and copy files in parallel, and*
- **pfcm** – *using parallel file tree walker and compare source and destination files in terms of byte content comparison. Users use it to verify data integrity of files after data copy.*

4.2 Backend System

4.2.1 GPFS – General Parallel File System

GPFS is a fast, scalable, and parallel file system from IBM that runs on commodity hardware and the Linux operating system. Based on performance testing in our environment, GPFS can scan one million inodes in ten minutes. This indicates that GPFS scales well under a heavy load in production environments and is a good fit in a parallel archive. GPFS can be configured to use storage pools. A storage pool can be thought of as a class of service where data can be stored. For example, all fast fiber channel disks can be in one storage pool. In our archive, we have a fast fiber channel disk storage pool where all files are initially written and a "slow" disk pool used to store small files. In addition to these, GPFS version 3.2 introduced the idea of external storage pools, which extends the GPFS pool metaphor to offline media. The external storage pool was necessary to merge the GPFS file system with tape software to construct the archive [17][18].

4.2.2 TSM - Tivoli Storage Manager

TSM is a backup/archive product from IBM that can move data to and from tape while maintaining a database of all objects that it manages. Like GPFS, TSM has excellent metadata handling capabilities and has been tested on hundreds of millions of files. Another important addition to TSM is Hierarchical Storage Management (HSM). GPFS supports HSM through the Data Management API (DMAPI). HSM is a TSM tool that manages moving files between the file system disk and the TSM tape system. HSM breaks any file that is passed to it into metadata and data. The metadata is stored on file system disk while the data is moved to tape (Figure 5). ILM external storage pools allow a GPFS policy to generate and pass HSM a list of files. HSM then processes every file in the list. This tiered storage is transparent to the user, and he or she are unaware as to whether a file is stored locally on disk or if it has been migrated to tape [19][20].

In addition to simple migration of data from disk to tape, we need a way to make movement parallel across the various machines. For standard TSM operations, all data is passed to a central server via the network, making the TSM server's network connection the bottleneck for transferring data. Fortunately, TSM has a feature called LAN-free that can move data directly from a client machine to a tape drive via the SAN. Metadata is still sent to the TSM server via the network. If you have multiple machines running LAN-free, they can read and write to different tapes independently of each other. This allows for parallel data movement to and from tape (Figure 6). HSM can use this LAN-free feature across multiple machines to migrate files to tape and recall files from tape in parallel.

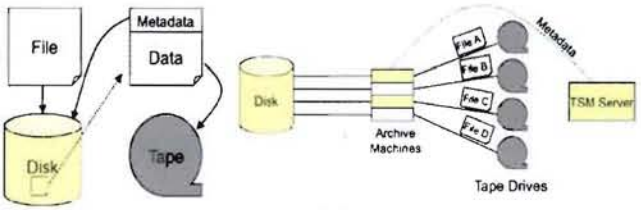

Figure 5

Figure 6: Parallel data movement

4.2.3 Controlling User Commands

If the archive is left as a standard UNIX environment, user can make use of any tool available on the UNIX platform. This becomes a dangerous problem when some files may be on tape. A simple example of this would be "grep" looking for a pattern across a set of files. In order to do so, the archive must recall any files to be searched from tape. This recall has no order and can result in a tape rewinding and seeking repeatedly to find files. This becomes especially problematic when we consider "grep" commands across machines. The machines may try to access files scattered across a single tape causing the tape to be mounted and dismounted repeatedly. While this is not a problem on a standard disk, tape complicates things because of its sequential nature.

One solution to this problem is to restrict the commands available to users by creating a unique environment using the UNIX "chroot" utility. "chroot" prevents users from seeing outside a specific subset of directories and lets administrators restrict available commands. While avoiding dangerous uses of commands like "grep," we encourage the

use of PFTool, which executes in parallel and is tape aware, for copying files.

4.2.4 Parallel Migrator

Although the GPFS policy engine supports parallel execution of migration policies, the migration does not take into account load balancing regarding file size or the number of GPFS machines. One process may be responsible for all of the large files in the list while another has nothing but small files. Additionally, although the GPFS policy engine may start multiple migrations to tape, all of these processes may be created on a single machine despite multiple machines being available.

Rather than use a GPFS migration policy, we use a list policy to generate lists of candidate files to migrate to tape. We combine, sort, and distribute the candidate files by file size evenly across machines. This allows the migrations to tape to complete at the same time across machines and can greatly speed up the process of migrating to tape in parallel. After migrating the data to tape in parallel, we also need a parallel way to recall it back to disk.

4.2.5 Parallel Recall from Tape

We strongly encourage users to use PFTool so we can optimize operations for tape archive/restore processing such as file recall ordering. PFTool can place the files being recalled in the order they are stored on tape. This means the tape can be read front to back without rewinding, but it also means we need a way of figuring out what tape and where on the tape (the sequence ID) a given file is.

It is difficult to query TSM directly for the tape and sequence ID for a given file. TSM version 5.5 and below make use of a proprietary database. The fields in question are not indexed, and we are unable to add our own indices. To solve this we export the necessary parts of the TSM database to a MySQL database, which we can then index. PFTool queries this database to get tape and sequence ID for files that are migrated to tape. It can then sort the files in tape order for efficient recall. It can also execute recalls in parallel across multiple tapes. In the course of transferring data to and from tape, we need to guard against data becoming stale on tape.

4.2.6 Synchronous Delete

When a migrated file is deleted from GPFS, the data on tape is orphaned because the delete from the file system only deletes the metadata. Traditionally, reconciliation is used to clean up these orphans on tape. The reconcile agent compares the file system and the tape system to make sure that they are synchronized. Unfortunately, the reconcile agent does a directory tree-walk and compares each file one by one rather than take advantage of the GPFS metadata system. For an archive with tens to hundreds of millions of files, the overhead is unacceptable.

To avoid reconciliation, we can synchronously delete the file from disk and tape. The process to do this is to first query GPFS for the file ID, a unique identifier generated by GPFS for each file. Then the MySQL database is queried to get the TSM object ID, a different unique file identifier created by TSM for the same file. Once the GPFS file ID and TSM object ID are known for a given file, the process can issue a delete to the file system and TSM at the same time. Unfortunately, only an administrator may do the GPFS file ID lookup and the TSM delete. Thus, we need a way to track files that users delete so that an administrative process can issue synchronous deletes. A trashcan is our solution.

4.2.7 Trashcan

From a user's perspective, the trashcan is identical to the Windows Recycle Bin. In our case, we use the GPFS policy engine to generate lists of all files located in the trashcans of various users based on age or size. These lists are passed to the synchronous deleter, thereby deleting data without leaving orphans on tape or requiring a costly reconciliation process. Before this policy is run, we can also un-delete in case a user accidentally deletes a file. Both the movement of data to and from tape, as well as the deletion of that data becomes more difficult when files become very large [21].

4.3 Integration of backend and frontend

4.3.1 Archive Setup

Our COTS archive consists of fifteen x64 machines: 10 nodes for data movement and 5 nodes with internal disk arrays totaling 100 TB (Figure 7). There is also an IBM pSeries machine for the TSM server. Each of these machines has a fiber channel card (FC4) for SAN connectivity and a 10-gigabit Ethernet card for network. We also have 100 TB of fast FC4 disk and twenty-four LTO-4 tape drives connected to the SAN. Software consists of Red Hat Enterprise Linux 5.2, GPFS 3.2, and TSM 5.5.

We integrate the archive parallel file system and the back end tape to eliminate garbage collection etc. to avoid having to sync archive tape back end and archive parallel file system metadata:

- The "chroot" jail environment setup allows us to implement our own delete function via a trash can which allows us to implement a delayed synchronous delete from the file system and archive
- The Fuse layer allows us to implement truncate/unlink any way we want in conjunction with the back end (synchronously or asynchronously), and
- We have learned over time that tight two way integration of file system and back end name spaces is difficult with and leads to poor performance, so asynchronous garbage collection is nice – but we cannot afford to walk both file systems and compare (reconcile).

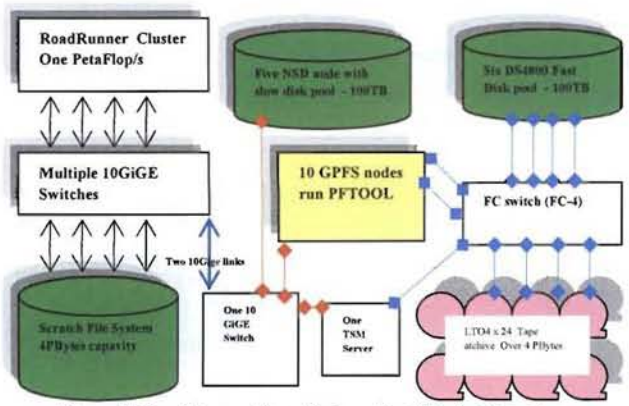

Figure 7: RoadRunner Open Science Parallel Archive setup

4.4 Handling various file sizes

Very small files can be backed up but medium sized files (millions of them) may need to be migrated. Need aggregation capability for medium sized file migration. Enormous files broken up by using FUSE.

4.5 Restart-able File Transfer

Occasionally, a network or other problem will stop a file transfer from file system to archive. Since we are transferring very large files, we needed a way to restart file transfer from the point the process last left off. Multiple single files can be restarted by using the date on the destination archive file when it is equal to or more than the source file. What about restarting a 40 Terabyte file, we don't want to start it from the beginning. To get around this, we mark regular file chunks or FUSE file chunks as good or bad so that we don't have to re-send known good chunks. This is a unique incremental parallel archive feature that can reduce unnecessary data copy and increase performance.

5 Performance studies on LANL's Roadrunner Open Science Projects

Ten science projects were chosen as the Open Science Projects to run on "Roadrunner", the world's first petaflop/s computer. These projects were completed within a six-month period following the installation of Roadrunner at LANL. This resulted in significant breakthroughs in materials, astronomy, and laser plasma science [21].

5.1 COTS Parallel Archive System Deployment on LANL's Roadrunner Cluster

We setup the proposed COTS Parallel Archive System with the Roadrunner cluster (Figure 7). Two 10-Gigabit Ethernet links were used for moving data between the scratch parallel global file system, Panasas, and the archive parallel file system(GPFS). PFTool software and utilities were running on ten FTA cluster nodes. The PANFS, GPFS, and ArchiveFuse file systems were mounted on each FTA node. Users use MOAB both interactively and in batch modes to launch parallel archive commands such as "pfls", "pfcp", and "pfcm." Over four peta bytes of data were archived within a six months period from the seven Open Science Projects.

5.2 LANL's Open Science Project – COTS Parallel Archive experience and performance data

PFTool runtime performance data was collected over a continuous eighteen operation day periods from LANL Open Science projects in summer 2009. We recorded 62 parallel archive jobs during this monitoring period.

Figure 8 shows the number of files archived per job (using $\log_{10}$ base). The range is from 1 file/job to 2920088 file/job. The average number of file per job is 167491 file/job. This demonstrates the scaling capability of our PFTOOL software system.

Figure 9 shows amount of data archived per job (using $\log_{10}$ base). The range is from 4GB/job to 32593GB/job. The average data archived per job is 2442GB.

Figure 10 shows the data rate (MB/sec) recorded per job. The variety of performance is dependent on the file size, number file archived, and overall system run-time status (bandwidth sharing and machine sharing among multiple users). The range is from 73 MB/sec to 1868MB/sec. It shows that we can reach almost ~75% bandwidth utilization from two 10Gigabit Ethernet trunk. The average data rate is about 575^{+}MB/sec which is a very good performance number compared to non-parallel archive storage systems with about 70MB$^{+/}$sec archival bandwidth.

Figure 11 shows the average file size archived per job. The range is from 4^{+}KB/file to 4,220^{+}MB/file. The average file size per job is 596MB/file. The diversity of the LANL Open Science projects can be seen by the various size of files created and archives. This is a valuable observation of data characteristics from different scientific computing problems.

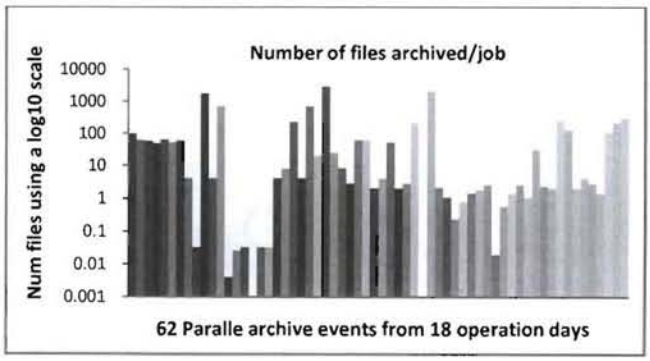

Figure 8: Number of files copied per job over 18 operation days

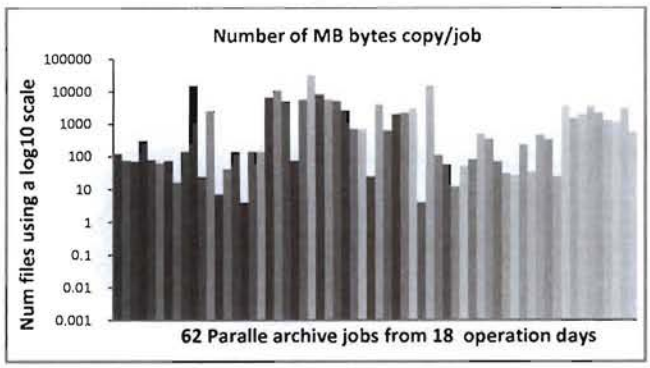

Figure 9: Number of data copied per job over 18 operation days

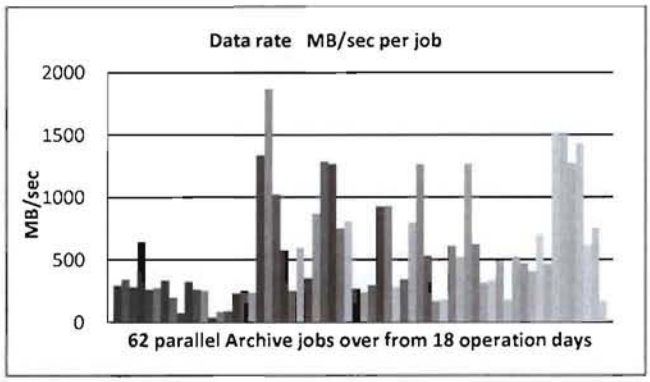

Figure 10: Archived Data rate per job over 18 operation days

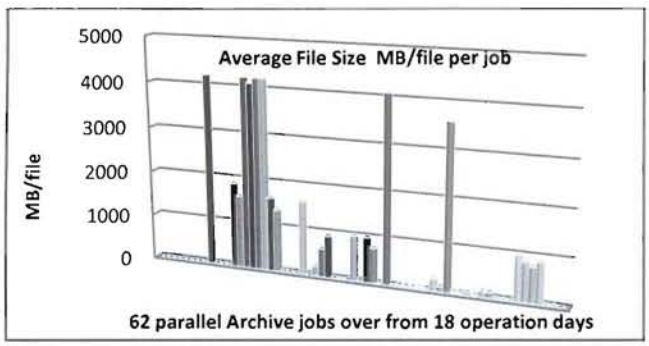

Figure 11: Archived job size over 18 operation days

6 Experience and observed issues of our COTS Parallel Archive System

6.1 Small File Tape Performance

For HSM tape operation, one file is one transaction. This results in poor performance for small files because the tape drive stops writing after each file. In one local case, a user copied millions of 8 MB files to GPFS disk. Migrating these files to tape was an order of magnitude slower than migrating large files at a rate of 4 MB/s instead of 100 MB/s, the rated performance of LTO-4 tapes. As a result, it took an entire weekend to migrate those files off of disk using 24 tape drives. One solution to this problem is aggregation, which consists of bundling these small files into larger aggregates better suited to getting the tape drive up to full speed, and then writing the aggregate to tape. Interestingly, the TSM backup client does this now, but currently we have no solution for migration.

6.2 Tape Optimization/Smart Recall

An HSM recall daemon runs on every machine in the cluster. When a request to recall a file from tape comes in, it is assigned to a machine. When a list of files to recall from tape is passed to HSM, even if we put that list in tape order, there is no guarantee the same machine will be responsible for all of the recalls for a given tape.

When the TSM server is reading from a tape, this is not a problem because it is the only machine to do so and does not need to rewind and seek or recheck the tape label. Since we use LAN-free, this does not hold true for our environment. The result is that HSM will send the recalls to different machines in the cluster that then causes the tape to rewind and verify its label every time the tape is passed between machines. This causes a massive performance hit even though the tape is not physically dismounted. A way to ensure that all files in a recall request are handled by the same machine, at least in a LAN-free environment, would correct this issue.

6.3 Limitations of the Synchronous Deleter

The synchronous deleter correctly handles users deleting files (unlink). However, it cannot detect when users overwrite files (truncate), and this behavior still requires a reconcile process to clean up. This is not the case when using our FUSE plug-in to correctly handles this by intercepting overwrite requests and move chunks into a trashcan that can be synchronously deleted before creating new chunks. However, having a built-in synchronous delete function between GPFS and TSM would catch all delete events and go further to avoiding reconciliation between file system and tape.

6.4 Single TSM Server

Having a single TSM server creates a single point of a failure in a generally redundant system. It also creates a limitation when we need to scale beyond what a single TSM server can provide. In our current archive, scalability is not an issue, but could be in future archives that have more than hundreds of millions of files. By leveraging the remote file system feature of GPFS, it might be possible to tether multiple archive file systems together thus allowing for multiple TSM servers. However, native support for multiple TSM servers would be beneficial to maintain a single namespace.

7 Summary and Future Works

We have proposed and built a hybrid Parallel Archive Storage System approach of using COTS components and

innovative software technology. We used the same solution as many others, but we have proved that our COTS Parallel Archive System can bring new capabilities into a production environment for the HPC community much faster than the approach of creating and maintaining a complete end to end unique parallel archive software solution.

We have successfully integrated a COTS global parallel file system and a standard backup/archive product with a very small amount of additional user space code to provide a scalable and parallel solution that overlaps highly in functions with current niche parallel archive products including (a) doing parallel movement to/from tape for a single large parallel file, (b) hierarchical storage management, (c) ILM features , (d) high volume (non-single parallel file) archives for backup / archive / content management, and (e) leveraging all free file movement/management tools in Linux such as copy, move, compare, ls, tar, etc.

We have some unique solutions to some of the shortcomings of HSMs for archives others could borrow

- Extremely small amount of code to maintain, extremely low development costs (maybe 1.5 man years compared to 100+ man years for non COTS solution)
- Highly leveraged use of COTS parallel file system with ILM features, COTS archive/backup products with LAN free movement and ILM features, ILM features will only get richer and add functionality
- Leverages Linux/Unix data management tools, a familiar environment with GUI's etc. (all for free)
- Why is now a good time to take another look at using HSM as an archive?

We plan to enhance the proposed COTS Parallel Archive System with the multi-dimensional metadata searching capabilities and provide an efficient solution for archiving vary large number of small files in parallel (i.e. very large number grass files parallel copy problem). We also plan to have a code development schedule to accommodate most of current COTS parallel file systems such as Lustre, PVFSv2, pNFS, cloud file systems, and cluster based file systems into the PFTool software system.

8 Acknowledgements

We would like to thank the NNSA ASC program, LANL's Institutional Computing, IBM GPFS and TSM teams, and everyone at LANL that helped throughout this COTS parallel archive project.

9 Reference

1. P.L. Bradshaw, K.W. Brannon, T. Clark, K. Dahman, S. Doraiswamy, L. Duyanovich, B.L. Hillsberg, W. Hineman, M. Kaczmarski, B.J. Klingenberg, X. Ma, R. Rees, "Archive storage system design for long-term storage of massive amounts of data, IBM J Research and Development, Vol 52, No. 45, July/Septerber 2008
2. Gary grider, "COTS Parallel Archive Integration Experiences, LANL LAUR XXXX-2009
3. Dingshan He, Xianbo Zhang, David H.C Du, Gray Grider, "Coordinating Parallel Hierarchical Storage management in Object-based Cluster File System", 2006 IEEE MSST Conference on Mass Storage Systems and technologies
4. L.L. You, K.T. Pollack, D. Long, "Deep Store: An Archival Storage System Architecture, Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)
5. M. Roschke, B. Parliman, D. Cook, D. Sherril, "Parallel Processing of Data, Metadata, and Aggregrates Within an Archival Storage System User Interface Toward Archiving a Million Files and A million Megabytes per Minutes), 2008 IEEE MSST Conference on Mass Storage Systems and technologies
6. Gary Grider, J. Nunez, R. Ross,j. Bent, L. Ward, S. Poole, E. Felix, E. Salmon, M. Bancroft, "Coordinating Government Funding of File System and I/O Research through the High End Computing University Research Activity, ACM SIGOPS Operating Systems Review Volume 43 , Issue 1 (January 2009)
7. M.W. Storer, K.M. Greenan, E.L. Miller, K. Voruganti, "Pergamum: Replacing Tape with Energy Efficient, Reliable, Disk-based Archival Storage, "2008 USENIX, FAST – File And Storage Technologies Confcrence
8. M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, O. Reshef, J. Satran, "The Nced for Preservation Aware Storage,
9. John McKnight, "Digital Archiving; End-User Survey & Market Forecast 2006-2010, " ESG- Enterprise Strategy Group, Research report, January 2006
10. J, Gray, W. Chong, T. Barclay, A Szalay, J. Vandenberg, "TeraScale SneakerNet: Using Inexpensive Disks for backup, Archiving, and Data Exchange". Microsoft Research, Technical Report MS-TR-02-54
11. R. Haynes, W.R. Johnson, "The Data Service Archive", 2004 IEEE MSST Conference on Mass Storage Systems and technologies
12. Technology Brief, "Archiving Beyond File System: Object Storage EMC Centera And Disk Archival", The TANEJA Group, Inc., 2009
13. Sun Micro, "The Right Data, in the Right place, at the Right Time", Technical Brief, Sun Micro System, 2009
14. Z. Ali, Q. Malluhi, "Performance Analysis of Distributed Parallel Archive Data Retrieval with Reliable Data Delivery", 2004 DASD Conference.
15. Sumit Singh, "Develop your own file systems with FUSE", IBM online developer works library information.
16. Technical Brief, "The New Metrics of Disk-based Data Protection", www.sresearch.com
17. "An Introduction to GPFS", IBM High Performance Computing, July 2006
18. "A GPFS Primer", IBM online document, Oct. 2005
19. "User Guide for Tivoli Storage manager for Space management for UNIX and Linux", Versiion 6.1, IBM
20. "Managing the Life of your Data – The Role IBM Tivoli Storage manager plays in the ILM model", Stephen Firmes, IBM System Magazine, May/June 2005 Issue
21. LIBNTRASH – A Trash Can for GNU/Linux, Free Software written by Manuel Arriaga http://pages.stern.nyu.edu/~marriaga/software/libtrash/
22. RoadRunner's Open Science - LANL's Open Science Project on RoadRunner, http://www.lanl.gov/orgs/hpc/roadrunner/
23. John Bent, garth Gibson, Gary Grider, Ben McClelland, Paul Nowoczynski, James Nunez, Milo Polte, Meghan Wingate, "PLFS – A Checkpoint File system for Parallel Applications", Proceedings of the 2009 Supercomputing conference, Portland, Oregan