HPC File Systems and Scalable I/O: Suggested Research and Development Topics for the fiscal 2005-2009 time frame

DOE Office of Science

Rob Ross ANL
Evan Felix PNL

DOE NNSA

Bill Loewe LLNL
Lee Ward SNL
Gary Grider LANL

DOD

Rob Hill, NSA

Executive Summary

The need for immense and rapidly increasing scale in scientific computation drives the need for rapidly increasing scale in storage for scientific processing. Individual storage devices are rapidly getting denser while bandwidth is not growing at the same pace. In the past several years, Research and Development (R&D) into highly scalable file systems, high level I/O libraries, and I/O middleware was done to provide some solutions to the problems that arise from massively parallel storage. This document primarily concentrates on file systems and I/O middleware since high level I/O libraries have been addressed in many Data Management discussions and calls for R&D. The purpose of this document is to present areas of needed research and development in the HPC file systems and scalable I/O area which should be pursued by the government.

In the last five years, supercomputers with thousands of nodes and over ten thousand processors have been deployed. Additionally, thousands of small clusters have been deployed worldwide. File systems and I/O has come a long way since the simple cross mounted NFS that was used with early clusters. File systems and I/O middleware have been developed and deployed for these supercomputers and clusters systems which have enabled bandwidths beyond ten gigabytes/sec and metadata performance beyond one thousand file inserts/sec. It is now possible to scale bandwidth, and there is competition in the scalable global parallel file systems market space through products that span the gamut from completely proprietary to open source. I/O Middleware is maturing which is enabling many applications to get good performance from the underlying file systems.

Recently several new I/O R&D efforts have begun to try to address future needs in HPC file systems and I/O middlware. Efforts in the areas of:

- Relaxation of POSIX Semantics for parallelism
- Scalable metadata operations in a single directory
- NFSv4 security and the pNFS effort to allow NFS to get more native file system performance through separation of data and control which enables parallelism
- I/O Middleware enhancements to enable dealing with small, overlapped, and unaligned I/O
- Tighter integration between high level I/O libraries and I/O middlware
- Initial autonomic storage management
- Sharing of global parallel file systems between multiple clusters
- Initial active storage research

In the near future, sites will deploy supercomputers with tens of thousands or even hundreds of thousands of processors. Immense bandwidth, metadata, security, and management needs are emerging. Work flows for efficient complex science will begin to approach the exabyte range, and the ability to handle a more varied I/O workload including small to extremely large I/O operations, extremely high metadata activities, and multiple simultaneous workloads will be required. File systems will be so large that complete checks or rebuilds, entire tree walks, and other large operations will not be able

to be contemplated. Management of these large storage systems will become increasingly difficult.

To meet the demands posed by the future HPC environments, investment in R&D and standards work need to be undertaken. The following are key areas to consider investment in:

- Scaling of metadata , security, reliability, availability, and management to deal with the enormity of future systems
- Enhancements in the POSIX I/O API and I/O middleware in the areas of ordering, coherence, alternate metadata operations, shared file descriptors, locking schemes, and portability of hinting for layouts and other information.
- Support for machine architectures which do not have full operating systems on each node and exploitation of hierarchies of node types
- Additional work in active storage concepts, including use in the LAN/SAN, WAN, application integration, and object archives
- Continued support for development and standards work on NFSv4 and pNFS
- Tracing, Simulation, and benchmarking, including application realistic benchmarking and simulation
- New metadata layouts other than tree based directories

More focused and complete government investment needs to be made in the file systems and I/O middlware area of HPC, given its importance and its lack of sufficient funding levels in the past, compared to other elements of HPC. Scalable I/O is perhaps the most overlooked area of HPC R&D, and given the information generating capabilities being installed and contemplated, it is a mistake to continue to neglect this area of HPC. Many areas in need of new and continued investment in R&D and standardization in this crucial HPC I/O area have been summarized in this document.

Background

The need for immense and rapidly increasing scale in scientific computation is well understood and documented. To keep up with the need for immense scaling, Moore's Law, which maps how individual processors get faster, and the idea of ever increasing parallelism are combined. In providing storage for scientific processing, similar and well documented concepts apply. It is well known that individual storage devices are getting denser at an amazing rate, keeping up with the ever faster speeds of processors. It is also well known that bandwidth to/from individual storage devices is getting faster but at an alarmingly slower rate than density of the devices. To deal with the ever increasing need for bandwidth to storage, massive parallelism is required.

In the past several years, Research and Development (R&D) into highly scalable file systems and I/O, was done to provide some solutions to the problems that arise from massively parallel storage. The I/O software stack is primarily composed of 4 basic layers: higher level I/O libraries, I/O Middleware, file systems, and storage devices. Higher level I/O libraries, for the most part, provide the needed high level abstraction for application programmers who are familiar with the science involved in the application. I/O Middleware, for the most part, provide abstractions that are computer science based, and deal with distributed/parallel nature of the memories involved and the distributed/parallel access to the storage devices. File systems, of course, provide the standard interfaces for organization, storage and retrieval of data and deal with coordination of the storage devices. The goal of the I/O software stack is to maintain performance and correctness from the high level I/O interface all the way to the storage devices and to fully leverage innovations in storage devices and the connectivity to those devices.

We define HPC file systems and scalable I/O as a subset of the overall scientific data management function, that subset of the I/O stack dealing with the storage devices, networks to reach those storage devices, data movement and metadata protocols, file systems, and I/O Middleware functions.

Purpose

The purpose of this document is to present areas of needed research and development in the HPC file systems and scalable I/O area which should be pursued. This document does not heavily delve into R&D needs in the high level I/O libraries area, because that topic is better addressed by Scientific Data Management (SDM) experts and scientific application designers and developers, although it is understood that coordination with high level I/O libraries is necessary and good. Nor does this document address R&D needs for advancement of physical storage devices. The R&D at the storage device level is largely driven by the needs of the multi-billion dollar storage industry and is done at a scale that is well beyond the proposed R&D investments in this document. The focus area of this document is the I/O Middleware and file system layers of the I/O software

stack, primarily, with some extension upwards into high level I/O libraries and down into storage devices.

Additionally, this document does not address breakthrough storage technologies that would fundamentally change I/O paradigms. This document assumes an evolution of storage devices with no extremely disruptive technologies. While there are new storage technologies nearly ready to enter the market place like Micro-Electro-Mechanical Systems (MEMS) (http://www.memsnet.org/mems/what-is.html) and Magnetic Random Access Memory (MRAM) (http://computer.howstuffworks.com/mram.htm), given the market drivers behind these technologies, for the HPC market, it is very unlikely these technologies will be disruptive enough, in the next several years, to cause radical or wholesale changes in the I/O stack. While it is important to remain vigilant in exploiting new evolutionary technologies and watching for disruptive technologies, this document only addresses evolutionary technologies and ideas.

This document is also based on the idea that high end supercomputing will continue to be based on physically distributed memories. Much of the I/O software stack assumes and deals with this current distributed memory reality.

Frequently used terms

I/O – input/output
File system – A combination of hardware and software that provides applications access to persistent storage through an application programming interface (API), normally the Portable Operating System Interface (POSIX) for I/O.
POSIX - Portable Operating System Interface (POSIX), the standard user interfaces in the UNIX based and other operating systems (http://web.access.net.au/felixadv/files/output/book/x1164.html)
Global – refers to accessible globally (by all), often implies all who access see the same view
Parallel – multiple coordinated instances, such as streams of data, computational elements
Scalable – decomposition of a set of work into an arbitrary number of elements, the ability to subdivide work into any number of parts from 1 to infinity
Metadata – information that describes stored data, examples are location, creation/access dates/times, sizes, security information, etc.
Higher level I/O library – software libraries that provide applications with high level abstractions of storage systems, higher level abstractions than parallelism, examples are the Hierarchical Data Formats version 5 library (HDF5) (http://www-rcd.cc.purdue.edu/~aai/HDF5/html/RM_H5Front.html) and parallel Network Common Data Formats library (PnetCDF) (http://www-unix.mcs.anl.gov/parallel-netcdf/sc03_present.pdf)
I/O Middleware – software that provide applications with higher level abstractions than simple strings of bytes, an example is the Message Passing Interface – I/O library (MPI-IO) (http:www-unix.mcs.anl.gov/romio)

WAN – Wide Area Network, refers to connection over a great distance, tens to thousands of miles
SAN – Storage Area Network, network for connecting computers to storage devices
QoS – Quality of Service

Historical perspective

To put into context the rationale for the proposed set of research and development topics, a high level summary of more than the last half decade of HPC file systems and Scalable I/O related research and development is presented here.

The early 1990's

In the early 1990's, most cluster based supercomputers were relatively small by today's standards, typically with tens of nodes, and for the most part used Network File System (NFS) servers for both home (program/source code/etc.) and scratch space. Sometimes only one NFS server was used, but frequently multiple cross mounted NFS servers were used. Given the small nature of these clusters, this approach provided sufficient performance and acceptable access. Given the primitive and limited scale of the applications in this time frame, parallel access methods to I/O typically were not needed.

The mid 1990's

In the mid 1990's, large scale cluster based supercomputers, hundreds to a few thousand nodes, began to show up. In order to serve the storage bandwidth needs of these clusters, hundreds of redundant array of independent disks (RAID) controllers with over a thousand individual disks used in a coordinated manner were required. Given that the Linux cluster phenomenon had not occurred yet, these clusters were for the most part proprietary operating system based clusters. NFS or cross mounted NFS was simply not an option on these machines due to their need for scalable bandwidth and manageability at scale. The file systems used on these clusters were first generation client/server based proprietary cluster file systems. Examples include IBM's General Purpose File System (GPFS) (http://www.almaden.ibm.com/StorageSystems/file_systems/GPFS/), Meiko's Parallel File System (Meiko PFS), and Intel's PFS (Intel PFS). Additionally, due to the large scale parallelism in the clusters of this era, middleware libraries like the MPI-IO library to enhance parallel access were begun. Projects to develop Storage Area Network (SAN) based file systems, which relied, at the time, on every file system client having direct access to the storage devices, typically via a fibre channel SAN, were also begun. Examples of SAN approaches at that time are the University of Minnesota/Sistina Global File System (GFS) (http://www.redhat.com/software/rha/gfs/), the Silicon Graphic Incorporated (SGI) Clustered-XFS (C-XFS) (http://www.sgi.com/products/storage/cxfs.html), the Advanced Digital Information Corporation (ADIC) Clustered Virtual File System (CVFS) (http://www2.adic.com/stornext/). These SAN file systems were primarily intended for use in smaller node count clusters (tens to a few hundred) due to the difficulty and expense in building large secure Fibre Channel SAN's. Fibre Channel SAN's lack the

ability to share data at a granularity smaller than a volume (disk or virtual disk) and all nodes/machines that have direct access to the SAN that are sharing a volume have root style access to that entire volume, so there is no concept of sharing at the file or object level with authentication based security. Another concept that took shape in this timeframe was the use of an old idea, the separation of metadata activity from data movement activity. Some file systems deployed this concept though the use of separate metadata servers and others shared the metadata responsibility among the file systems clients through the use of lock mechanisms.

The late 1990's to present

In the late 1990's to the present, cluster supercomputers are now routinely in the thousands of nodes with tens of thousands being contemplated. Given the slow improvement of individual disk device bandwidth improvement compared to processor speed increases, the number of individual disk devices needed to be coordinated to achieve the needed bandwidth for these super clusters is currently several thousand. Open source Linux clusters are common place, implying that other, more open and Linux based, solutions for global parallel file systems are needed and past proprietary solutions for global parallel file systems are diminished in value.

Both SAN based and client server file systems have grown in popularity as clustered computing has become more prevalent in thousands of computing centers worldwide. Pure SAN file systems, again, where all file system clients have access to the disk devices, still suffer from scalability for very large clusters due to the difficulty in building scalable Fibre Channel SANs, also have security issues due to clients having direct access to storage. Most SAN based file systems have begun to offer ways to extend access to beyond clients that have direct access the disk devices through a variety of mechanisms like SCSI protocol over IP (ISCSI), gateway functions, and others. In part, due to government sponsored R&D, new and less proprietary client server file systems have become popular. Examples include: the DOE Office of Science/Argonne National Laboratory/Clemson Parallel Virtual File System (PVFS) (http://www.parl.clemson.edu/pvfs/desc.html), the DOE/NNSA/tri-labs/HP/CFS/Intel Lustre file system (http://www.lustre.org), and the Panasas PanFS (http://www.panasas.com) file system. These new and less proprietary client server file systems leverage heavily separation of data and control, abstracting the storage device functions like allocation and date-write operations for less than full block updates away from the file system proper.

Recently the ANSI T10/1355-D Object based Storage Devices (OSD) (http://www.t10.org/ftp/t10/drafts/osd/osd-r10.pdf) standard was introduced, putting the industry on a course to standardize the interface to storage devices that provide allocation, read-update-write on partial block writes, and transactional security for the storage devices. Additionally, using separate metadata servers for metadata operations has become even more popular. The MPI-IO library and other parallel access methods became popular in parallel applications during this time.

Additionally, in this time frame several important HPC file systems and I/O related R&D efforts were begun including:

Relaxation of Posix Semantics

A realization that the POSIX semantics had to be broken to enable scaling occurred. Ordering semantics to ensure last writer wins on overlapped I/O operations and lazy updates of last update times/dates and file sizes for files being written to by many clients concurrently are both examples of where relaxations have been made. File systems like PVFS, Lustre, and Panasas have all implemented options for relaxed POSIX semantics where absolutely necessary for scaling. These three file systems and others have implemented special (non POSIX standard) I/O controls (IOCTL's) which allow applications to control widths, depths, and other striping oriented layout information for files. The PVFS is probably the best example of extending POSIX semantics for scaling. This is accomplished by providing a user space library interface to the file system which is well integrated with the MPI-IO parallel I/O library. To assist with situations where overlapped I/O is necessary and control over last writer wins semantics is important for the application, Northwestern University has done important work in enabling detecting and handling these overlapped I/O operations within the MPI-IO library, where multiple clients can coordinate these activities in an intelligent way.

Scaling

Extremely scalable parallel data movement bandwidth has been achieved by client server based file systems. Both Lustre and Panasas have shown coordinated bandwidths in excess of 10 GigaBytes/sec and both have plans of exceeding 30-50 GigaBytes/sec in fy05. For the most part, the data movement bandwidth scaling problem has been solved at least for non-overlapped, large buffer, parallel I/O operations to/from file systems. Some initial research has begun at the University of California Santa Cruz (UCSC) in the area of scalable metadata. The problem here is handling tens to hundreds of thousands of inserts, deletes, and queries per second in file systems that will manage billions of files. Some file systems like the IBM SAN file system and others have introduced scalable metadata systems, but only for scalability across multiple directories. Scalability within a single directory is a very hard problem especially when posed with the possibility that mass operations like tens of thousands of inserts could be requested into the same directory or subdirectory all within a few milliseconds. UCSC has come up with some novel ideas in trying to address this problem. Additionally, the Lustre and Panasas file systems are building first generation engineering approaches to the scalable metadata problem for scalability across multiple directories as well as within one directory.

NFS version 4

The Internet Engineering Task Force (IETF) NFS version 4 (NFSv4) (http://nfsv4.org) effort has seen progress in the last two years. The NFS has been riddled with security problems since its inception. The NFSv4 effort is addressing this issue through the use of the General Security Services (GSS) infrastructure. Many government sites see this as a

very useful move for the NFS. Additionally, the NFSv4 effort has new compound operations capability which allows for multiple operations to be coalesced to allow for efficiencies never before possible with the NFS. Additionally, there is a new parallel NFS effort (pNFS) (http://www.ietf.org/proceedings/04mar/slides/nfsv4-1.pdf) which is a part of the overall IETF NFSv4 project. This pNFS effort promises to allow NFSv4 clients to perform metadata operations on file systems via the NFSv4 server and then bypass the NFSv4 server for data movement operations. There have been many non-standard modifications to the NFS over the years to bypass the NFS server for data movement operations, going directly to the storage devices. This pNFS effort legitimizes these approaches via the IETF standardization process. This pNFS effort promises to make NFS clients first class clients to parallel file systems for performance. This allows for a large variety of heterogeneous clients to have high performance clients to parallel file systems. Important work by the University of Michigan is enabling a Linux implementation of these NFSv4 features. Many vendors are participating in building NFSv4 releases as well as working on the NFSv4 and pNFS IETF standards effort. Garth Gibson of Panasas and Carnegie Mellon University (CMU) has been instrumental in working on the pNFS standards effort.

Higher level I/O libraries and integration with I/O Middleware

The I/O software has to be able to extract the available performance from the underlying storage devices. The high level I/O library must be well integrated into the overall I/O stack so that performance can be attained. It is vital to recognize important performance work done by assisting with tighter integration between I/O Middleware and higher level I/O libraries like the Hierarchical Data Formats version 5 (HDF5) from the National Center for Supercomputer Applications (NCSA) and parallel Network Common Data Format (PnetCDF) Argonne National Laboratory (ANL) and Northwestern University. It is important to tune these higher level libraries and tune how applications utilize these libraries. Work by ANL to utilize the PnetCDF library integrated with the MPI-IO library and the PVFS achieved promising performance results. The joint work by Los Alamos National Laboratory and NCSA using the Unified Data Models (UDM) library in conjunction with the HDF5 library which uses the MPI-IO I/O Middleware has also achieved promising performance for applications.

Enterprise Class Global Parallel File System (GPFS – not to be confused with the IBM GPFS – General Purpose File system)

Supercomputer sites have been deploying more than one computing cluster for many years. Sometimes sites will have a very large cluster for simulation with smaller clusters for pre/post processing of data, reduction, analysis, and visualization. In the last few years this has become common for larger supercomputer sites to have more than one very large cluster for simulation. This often arises from sites installing a new large cluster every year or two but keeping the past generation or two before decommissioning. Sites with more than one cluster increasingly want to share access to the same data between clusters. In the past it was very common to have a separate parallel or high bandwidth file system on each cluster with some means to move the data between clusters for

sharing, either through direct data movement or via a common archive capability. Many supercomputer sites are now expressing the desire to have and deploy parallel and scalable file systems that are shared between all the supercomputers at the site. We refer to these deployments as Enterprise Class Global Parallel File Systems (GPFS). Often sites desiring this Enterprise Class GPFS capability even want to provide access not only to just the clusters in the enterprise, but for workstations and other servers in the enterprise. Giving access to multiple clusters and workstations to a common file system gives rise to new issues like Quality of Service (QoS) guarantees to more important clients, differing security between different kinds of clients, and heterogeneous access. Space allocation on a filesytem according to various policies, which relate to which system, project, or user is creating the data, will also become important, as multiple funding sources will be dictating how the resources are used. Relevant work in this area includes the NFSv4 and pNFS work already mentioned, as well as some early design work for QoS in the Lustre file system and in the ANSI T10/1355-D Object based Storage Device standardization effort.

Utilize processing power near the storage devices – Active Storage

With the success of client server oriented file systems, the opportunity to utilize server side processing power near the disk storage device was recognized. Many organizations have contemplated using this power near the disk. Early research at the Intelligent Storage Consortia (ISC) at the University of Minnesota has looked at ways to utilize the power near the storage device to provide functions like hierarchical storage management (HSM), indexing, and mining. Other universities have also done preliminary research into how the power near the storage device could be used. Proof of Active Storage concepts has been done at PNNL(http://www.emsl.pnl.gov/), by augmenting the Lustre filesystem. Additionally, industry has begun to deploy storage systems with processing capabilities, probably the most noteworthy are the Content Addressable Storage released recently by the EMC^2 Corporation (http://www.emc.com/products/systems/centera/) which allows access to data objects by their content, and the Data Appliance also released recently by the Netteza Company (http://www.netezza.com) provides database query operations. This area of R&D represents great promise, but only initial work has been done. The advent of the ANSI T10/1355-D OSD standard which provides a standard and secure way to request actions from an intelligent storage device is an important step to being able to utilize processing power near the storage device.

Summary of current state

It would be appropriate to call the late 1990's to the present the era spent enabling extremely scalable data movement bandwidth. Multiple client/server based file systems have achieved greater than 10 GigaBytes/sec and have plans to exceed 30-50 GigaBytes/sec in fy05. Supercomputing sites have deployed scalable global parallel file systems for individual extremely large clusters as well as for multiple clusters in an enterprise. It is now possible to scale bandwidth, and there is competition in the scalable global parallel file systems market space through products that span the gamut from completely proprietary to open source. Standards are emerging, like the ISCSI standard, the ANSI T10/1355-D OSD standard, and others, which will lead to even more

competition. Some work has been done to deal with POSIX limitations; the NFSv4/pNFS efforts appear to be headed in a good direction for security, heterogeneous access, and WAN access; tighter integration of the I/O software stack has yielded good results; and some promising initial work on metadata scaling and how to utilize the power near the disk has been started. Much progress has been made in the HPC file systems and scalable I/O area in the last decade.

Demands of the next 5 years

In the near future, sites will deploy supercomputers with tens of thousands of processors routinely, perhaps even hundreds of thousands. Bandwidth needs to storage will go from tens of GigaBytes/sec to TeraBytes/sec. Online storage needs to support work flows for efficient complex science will begin to approach the exabyte range. The ability to handle a more varied I/O workload including small to extremely large I/O operations, extremely high metadata activities, and multiple simultaneous workloads will be required. Additionally, new access methods, such as access methods to files/objects in arrangements other than tree based directories, will be required. Global or virtual enterprise and wide sharing of data with flexible and effective security will be required. Current extreme scale file system deployments already suffer from reliability and availability issues including recovery times from corruption issues and rebuild times. As these extreme scale deployments grow larger, these issues will only get worse. It will possibly be unthinkable for a site to run a file system check utility, yet it is almost a given that corruption issues will arise. Recovery times need to be reduced by orders of magnitude and these types of tools need to be reliable, even though they may rarely be used. The number of storage devices needed in a single coordinated operation could be in the tens to hundreds of thousands, making the need for integrity and reliability schemes to be far more scalable than available today. Management for enterprise class global parallel file/storage systems will become increasingly difficult due to the number of elements involved and the extreme varied workloads. The challenges of the future are formidable.

Key areas of possible future research, development, and standards work

As indicated by the challenges for the future, the needed R&D activities need to shift from scaling of data movement bandwidth for large I/O operations to scaling performance for high volumes of small I/O operations, high volumes of metadata operations, and extreme mixing of a variety of workloads. Improving reliability, integrity, availability, and manageability by orders of magnitude must be addressed. Additionally, R&D into new access methods, issues for enterprise wide sharing, WAN and heterogeneous access, security, as well as extreme scaling issues for metadata operations, and security will be required. Novel approaches to these issues such as leveraging the power near the storage devices, tighter integration of the I/O software stack, as well as other approaches should be studied.

It is important that the natural evolution from ideas to prototypes, from prototypes to user level library implementations, from user level library implementations to standards, standard implementations and even products, be supported and managed. Thus, all efforts throughout this evolution should be undertaken where promising results are indicated.

Below, categories of future R&D and standards work are discussed.

The POSIX I/O API

The POSIX API is "unnatural" for high-end computing applications. Opportunity abounds to make the POSIX I/O API more friendly to HPC and parallelism. The entire set of operations should be combed over and carefully, consistently, altered for high-end computing needs. Then, the result must be re-integrated in such a way that it is enabled by all applications in a positive fashion. The resulting semantics, after all, are less than useful for legacy applications as well as single platform applications.

Ordering

One of the prime reasons the POSIX I/O API is awkward for HPC stems from the "stream of bytes" paradigm in which POSIX I/O is based. POSIX I/O was developed to provide an interface from a single machine with a single memory space to a streaming device with some simple random access capabilities. HPC/parallel I/O applications are based on distributed memories being mapped to many storage devices. A re-interpretation towards the concept of a "vector of bytes" would be more appropriate for HPC applications versus a "stream of bytes" model. This would entail a careful reexamination of the POSIX I/O-related interfaces to eliminate or relax stream-oriented semantics.

Coherence

Probably the worst offense for really high bandwidth I/O is the fundamental read and write calls. They have two glaring problems. The first is coherency. The last-writer-wins semantic of the write call without cache-snooping is difficult, perhaps impossible, to achieve in a scalable fashion. Similarly, again without cache-snooping, the overhead of cache invalidation is enormous for reads. Especially if attempted for regions for which an application might never have interest. Additionally, block boundaries can present coherence issues for the application. A standard way for applications to assume all responsibility for coherency is needed, which implies that application control of cache flushing/invalidation at some level is also needed. Additionally dealing with block boundaries and alignment needs to be dealt with in the API in a consistent manner which takes alignment/block boundary issues away from the application. This problem is particularly bad in the implementation of O_Direct today.

Extension issues

The POSIX I/O API is organized as a set of mandatory functions and sets of extensions to that set of mandatory functions. Current extensions of the API are awkward for use in

HPC. For instance, the real-time extensions for list-based I/O (listio) are useful, however they are awkward in that they have the restriction that the memory regions of interest must coincide exactly with a region in the file. For many applications, relationship of memory regions to regions in the file is often not possible. Instead, two separate vectors, one of memory regions and one of file regions, could be passed and the two lists reconciled by the implementation. Such a concept allows scatter/gather-gather/scatter semantics.

Missing capabilities

The POSIX I/O API is also not as complete as one would like. For instance, there is a call to position and write a buffer of data (pwrite) but no call to position and write a vector of described memory regions, like a pwritev.

Metadata

The classic "wish" in this area is for the support of "lazy" attributes. These are results to the stat call, where some values may not be maintained at the same granularity normally expected. The most obvious fields are those that record timestamps for last update and modify. Many file systems implement these but no two in the same way. A standard, portable set of semantics would be useful. Explorations into a more descriptive API for metadata management and query to allow applications to deal with the needed information could be helpful in this area. For years, the backup/archive industry has needed a portable bulk metadata interface to the metadata of file systems. There are many opportunities for R&D in the area of an overhaul of how metadata is handled and the API's with which it is accessed given the extremely limited implementations in today's file systems.

Locking schemes

Locking schemes that support cooperating groups of processes is also necessary. The current POSIX semantics assume only a single process would ever want exclusive write access. In the HPC/parallel world, groups of processes may want to gain exclusive access to a file. Perhaps a mandatory version of the fcntl and flock is needed. It is necessary that more options for how locks can be requested and revoked be provided. Legacy codes must continue to work as expected, so current locking semantics must be maintained.

Shared file descriptors

Shared file descriptors between nodes in a cluster would be of great value, not just between processes on a node. The component count for supercomputers is going up in the next generation of supercomputers. Full lookups in the name space are going to have to become a thing of the past for parallel codes. Some mechanism decreasing the need for mass name-space traversal is desperately needed, even if it requires new semantics to accomplish it. It is possible to implement this via higher level function in the I/O stack. Implementation of shared file descriptors at the file system level might be difficult but

none the less would be quite useful, if achievable with a reasonable amount of R&D investment. As mentioned in the metadata section above, an alternate API for name space traversal as well as alternate file organization (something other than a tree) might also be a way to assist in this area.

Portability of hinting for layouts and other information

There is a need for proper hinting calls. Things like stripe, stride, depth, raid layout options, etc. need to be accomplished in some portable way. Additionally, there needs to be mechanisms for adding standard hinting without major new standardization efforts. Perhaps the MPI-IO Info approach for hinting can serve as a prototype, particularly in terms of the semantics, like ignoring of unknown hints and the mechanism for getting the values to use. For users to understand and use these hints effectively, they need to be as easy to use as things like umasks, shell variables, or file permissions.

Necessary determinism

Additionally, all operations done on the basis of time are awkward to deal with on supercomputers with light weight operating systems due to the inability to respond via asynchronous signaling to call back mechanisms. Supercomputing applications need more deterministic behavior and more control over the hardware throughout the entire computation and I/O hardware stacks. Operations with the ability to be driven from clients that can't listen for call backs is vital. It is quite likely that some variants of supercomputers with hundreds of thousands of processors simply won't be able to be bothered with call back mechanisms at all.

Active storage concepts

Active storage concepts are those ideas where CPU power near the storage device is utilized for better overall application performance or machine throughput. The work on active storage and associated concepts is interesting, although it is important that the value proposition be examined closely. Just because processing power near the storage device can be used to participate in the problem solution does not mean there is value in doing the processing near the disk as opposed to on other hardware. Many examples of particular applications and classes of applications enjoying significant benefits from this technology are available. However, to date, no pursuit of a generally useful, secure interface has been made. Research into a library, or hardware/firmware, interface supporting "sandboxes" and rich programming support could go a long way toward motivating applications to make use of the scheme. Without proper interfaces, this proposed new function will always be a "one-off" for any application, generally useful and portable for that application, but extremely hard to reuse, especially in an environment where more than one application might like to leverage active storage paradigms simultaneously. This R&D area is in its infancy, still mostly in prototypes. More work needs to be done to understand better the value proposition this idea brings and how it might be used in a more standard way.

Application of active storage concepts – Network Topology

In the local, machine, or system area network setting, the processing power near the storage device has no real advantage in bandwidth or latency to the storage device over any other processor, it is unclear that applications actually benefit from the processor near the disk versus just adding another general purpose processor to the application pool or a co-processor near the disk used only for application specific code. There is risk, availability/reliability risk, in putting application oriented function directly in the path of a highly shared item like a storage device. The processing power near the storage device does enjoy one advantage in this setting, that being location. All accesses to the storage device go through this processor. It is possible that this advantage could be exploited in some manner.

In the wide area network setting, the processing power near the storage device offers at least a latency advantage and possible also a bandwidth advantage over a general purpose processor. In this setting, there are many advantages to exploit, including smart batching of requests to hide latency, retrieving only the data needed for transmission over the WAN, etc.

Application of active storage concepts – enhancing the I/O stack function

Another important aspect is the ability to utilize the processing power near the storage device to simplify the higher layers in the I/O stack. Pushing more function nearer the storage device could have the benefit of allowing more innovation to occur for file systems, I/O Middle Ware, and high level I/O libraries. Exposing data layout information to the processor near the storage device could help that processor better map I/O operations to the geometry of the underlying storage and open up new possibilities for I/O stack exploitation of this concept. Database systems live in this world, so it is likely that many ideas can be formulated by studying database technology. R&D in this area could pay big dividends for some applications.

Leveraging active storage based file system technology, Archive/HSM, Other approaches

One example of utilizing active storage to allow for enhancement of the I/O stack is the possibility of integration of Archive/HSM function. Disk-based file systems for clusters are increasingly using multiple software "mover" components to accomplish parallel data transfers. These movers, most often, function by exporting access to unique, independent data stores.

Classic hierarchical storage management (HSM) methodologies also employ multiple movers, but curiously, usually to support more connections, not parallel connections. 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 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.

Further, there are doubtless additional approaches not yet considered for using active storage. There needs to be an effort to pursue other ways one might design a scalable file system with computational power near the storage devices to contribute to providing solutions for data intensive related problems?

Application integration with Active Storage

As has been mentioned earlier, tighter integration in the I/O stack is becoming important for applications to effectively tap the performance of the I/O Middleware, file systems, and storage devices. Extending this integration from the application all the way to the storage device through the use of processing power near the storage device, which has been shown in the past to have promising performance, warrants a closer look. As mentioned above, if the processing power near the storage device had information about data layout, much could be done to exploit that fact higher in the I/O stack. A generic capability for applications to securely and effectively utilize processing power near the disk should be explored, prototypes of this environment need to be developed and tested to determine if the performance value proposition is worth the risks of destabilizing a highly shared device like a storage device. More work in understanding the performance payoff for applications, the API(s) needed to accomplish this capability, and the risks in providing this capability needs to be done.

To date, though, all research in this area has only been applicable to a single application at any given moment. To be really useful, sandboxes or other technology must be applied to allow independent applications access simultaneously.

NFSv4

As has been mentioned earlier in this document, the NFSv4 effort has yielded a much more secure NFS capability. There is still good work in the pipeline from the NFSv4 effort which must continue to be supported. Additions of directory leasing capabilities for WAN access performance, load balanced NFS serving, and the important pNFS effort which promises to allow heterogeneous NFSv4 clients to access file systems more natively by bypassing the NFS server for data movement operations, are all vital parts of the NFSv4 effort that need to be accomplished. In order for these efforts to be successful, development and standards work need to continue. The IETF requires two interoperating implementations, so this requires persistence in funding and oversight to see these projects through to completion and insertion officially into the IETF.

Enterprise Class Global Parallel File System (GPFS – not to be confused with the IBM GPFS – General Purpose File system)

The use of a global parallel file system by multiple clusters within an enterprise and extending access to the desktop workstations of an enterprise causes a set of issues to

arise. These issues have mostly to do with treating one set of clients differently than others. There may be a need for security or other services to behave differently based on file system client or sets of clients. Additionally, this idea applied to performance implies a QoS solution is needed to enable one set of applications/clients to be treated differently than others applications/clients. R&D and standards work need to occur to enable these capabilities to support this enterprise class sharing concept in a portable way. Further, when connecting multiple clusters of different technologies and workstations to an Enterprise Class GPFS, scalable backbone technologies that allow heterogeneous interconnection at extremely scalable bandwidths with high reliability and availability are needed. Normally, single cluster interconnects are designed to scale to very high cross sectional bandwidth, but intra-cluster networks are not designed to scale that broadly. This multi-heterogeneous-cluster to common GPFS scalable network is a new development and needs to be studied. It is possible that Internet Protocol version 6 (IPv6) may be of assistance with this issue.

Scaling

As mentioned before, clusters of unprecedented scale are on the drawing boards with tens to hundreds of thousands of processors. Given that data movement scaling has been accomplished, R&D to address scaling other attributes of file systems and I/O is desperately needed.

Metadata

Clustered file systems seem to be converging on an architecture that employs a centralized metadata service to maintain layout and allocation information among multiple, distinct movers. While this has had significant, positive impact on the scalability in the data path, it has been at the expense of the metadata service. Due to scale up, the transaction rates against the metadata service have increased. As well, the amount of information communicated between the metadata service component and the clients has increased. There is some belief that such a file system design is problematic. The reason for this is seek-latency in disk media. Additionally, alternate metadata access methods, like bulk metadata access and perhaps alternative to tree shaped access of the metadata might provide both new needed function and relieve some of the metadata scaling issues. It is vital that continued R&D investments be made in this vital scaling of metadata performance.

Data movement bandwidth with small and unaligned I/O operations

With the incredible success in scaling data movement bandwidth using large I/O operations, it is now time to concentrate on dealing well with the scaling of small I/O operations. Many applications have not been able to take advantage of the enormous improvements in file system scalability in the last several years due to the small I/O operation sizes used by these applications. It is vital that all applications be able to have scalable I/O available to them. Often, it is inconvenient, or impossible, for the applications to be altered. They are expensive, proven codes and, in some cases, the host machines do not have the memory it would require to efficiently rearrange working sets for efficient data transfer. For non dusty deck applications, R&D in areas of more

aggressive caching in high level I/O libraries, I/O middleware, and alternate file system consistency close-to-open semantics could pay off, particularly in applications with lots of small I/O operations to independent files. It is also possible that active storage pursuits could assist in this area significantly by providing data layout information to the processor near the storage devices to allow for better mapping of the workload to the underlying storage geometry. For the more dusty deck oriented applications, this is a very difficult and perhaps nearly intractable problem, however, if R&D could assist these applications in their ability to use global parallel file systems more effectively and efficiently, there is a win for users.

High Level library exploitation of scalability enhancements

As has been mentioned several times in this document, integration up and down the entire I/O software stack has yielded good performance benefits. If file systems become better at scaling of metadata and small I/O operations as mentioned above, further re-integration of higher level I/O libraries to take advantage of these file system improvements may be possible and should be explored. As an example, it is possible that formatting libraries, which currently put all data and metadata for a single application run into a single file, might leverage scalable metadata operations by keeping a family of related files associated with a single application run. It is important to not resort to thousands of files per application or one file per process in this endeavor, but some modest number of files based on access patterns could be exploited. In the storage management field, the term "collections" is used to describe this concept. All advancements in the file system layer and below should be exploited if possible by higher layers in the I/O software stack.

Security

Another dimension for file systems that must be addressed in a scalable fashion is security. Allowing file system clients to access storage devices in a scalable way may require transactional oriented security so storage devices can trust that clients are authorized to perform the request. Additionally, with metadata services becoming more scalable, security related workload for authentication and authorization, which the metadata server must do, is increasing. The necessity for security even as scaling increases means security services must be scalable too. For this reason, R&D investments in security scaling must be undertaken.

Reliability and availability

As has been discussed, clusters of unprecedented scale are being planned. To provide the needed file system bandwidth to these clusters, unprecedented numbers of storage devices will need to be used in a coordinated way. Striping data from a single application over enormous numbers of disks will eventually lead to difficulties in protecting against data unavailability or loss. Current RAID protection and availability technologies are not designed to provide sufficient protection for such immense scale. One concept that could be pursued is lazy redundancy, producing redundant data at specific points in time, perhaps associated with checkpoints or snapshots. This concept allows for variable redundancy on a per file basis which allows for trading off reliability for performance

based on expected usage. Another concept that could be pursued is the ideas related to raiding memories in compute nodes. This concept works quite well for pure defensive I/O and dovetails nicely with MPI-2 features of dynamic process/communicator growth. There are no doubt other redundancy concepts that could be pursued as well.

Management

Yet another area affected by immense scale is management. The number of devices needed to provide the needed scalable file system service in a demanding and mixed workload environment of the future will be extremely difficult to manage given current technology. Advances must be made in massive scale storage management to enable management survival with future file system deployments.

Autonomic Storage Management concepts

The storage industry is currently working on management solutions that are fully automated. Ideas like, storage that self configures, self heals, self migrates, and self tunes are all being pursued. These ideas are all good ideas and the related projects need to be at least followed by the HPC I/O community. Additionally if these features are to be useful in the HPC environment, where things like determinism in parallel are important, it would be very useful for the HPC community to be involved in this Autonomic Storage Management R&D to ensure that these good ideas are implemented in a way that the HPC community can benefit. As an addition to this thinking, automated mining of data is also being pursued. These features also could be useful in the HPC environment but must be developed with consideration for the HPC I/O environment.

Hierarchical I/O architectures

Many supercomputers are arranged with compute nodes, I/O/routing nodes, and storage. I/O. All I/O operations on behalf of the compute nodes are routed in some manner through the I/O/routing nodes. Some of the newer architectures emerging, like the Red Storm and Blue Gene/Light architectures, require this type of I/O arrangement. An excellent research questions is: “Can part of the I/O function be placed at these I/O/router nodes to assist in performance for the user, especially in the areas of caching and aggregation”?

Tools for scalable I/O tracing and file system simulation

In parallel application building and tuning, there are a multitude of correctness and performance tools available to applications. In the area of scalable I/O and file systems, there are few generally applicable tools available. Tools and benchmarks for use by application programmers, library developers, and file system managers would be of enormous use for the future.

Tracing

Tools to quickly get tracing information from parallel applications, analyzing these traces to characterize the applications I/O footprint, and even being able to replay the traces in parallel against real or simulated parallel file systems would be of great use.

Simulation

Tools for simulation of portions or entire global parallel file systems would also be of great use to assist in understanding design trade-offs for I/O performance for applications, libraries, and file systems. Most other areas of the computer industry rely heavily on simulation tools. While asking for a parallel file system simulator is a tall order, the value it would have could be enormous.

Benchmarking

As in other areas of computing, benchmarking is a vital part of the I/O professional's toolkit. Benchmarks in the areas of interactive benchmarks (simulating user experience items like ls, cp, rm, tar, etc.), throughput benchmarks (including both peak performance and I/O kernels from real applications), and metadata benchmarks (collective opens, creates, resizes, etc) are needed. Some of these benchmarks exist and others do not. R&D in the benchmarking area is needed to collect and index current benchmarks and design and build new benchmarks to fill any gaps. At the very least, a clearing house for all I/O benchmarks and their usage could be of benefit. It currently is quite difficult to determine if a benchmark exists for a particular function or workload.

New metadata layouts other than tree based directories

In order to assist applications with managing enormous amounts of data, application programmers and data management specialists are calling for the ability to store and retrieve data in organizations other than the age old file system tree based directory structure. The data formats libraries currently provide some of this function, but it is not at all well mated to the underlying file system capabilities. Databases are often called upon to provide this capability but they are not designed for petabyte or exabyte scale stores with immense numbers of clients. Exploratory work in providing new metadata layouts is vital to address this identified need.

Kernel/user space data movement and control passing

There are many benefits to providing I/O related services completely in user space. Due to the kernel based file system layer paradigm on which most all operating system function rests, it is necessary to continue providing file system services through the Unix kernel. If a zero-copy data path from user space to kernel and back were standardized in Unix/Linux, it would be possible to implement more file system services in user space without a penalty in bandwidth or latency performance. This development could open up new and innovative I/O and file system services never before possible due to the abundance of user space developers. As well, user-space implementations tend to be more highly portable and cheaper to implement.

IPv6

If machines continue to grow in power at the same rate – Moore's law again – then the number of components in the I/O system must increase dramatically. Most of these components are uniquely addressable. While the internet protocol (IP) is ubiquitous, its address space is partitioned into only very small remaining chunks. The advent of IPv6 presents an opportunity to cleanly craft addressable sub-units in the I/O system. Unfortunately, little attention has been paid to this relevant streamlining.

Support of Storage Centers of Excellence

As part of the overall investment strategy, consideration should be given to supporting the approximately 3 university storage centers of excellence within the US at some base level. These centers provide ongoing file systems, storage, and scalable I/O research funded by industry partners. Supporting and being involved in these centers leads to more access to industry planners, more leverage over research topics, and more access to students and faculty. Additionally, supporting these centers produces the next generation of researchers in the field.

Conclusion

More focused and complete government investment needs to be made in this area of HPC, given its importance and its lack of sufficient funding levels in the past, compared to other elements of HPC. Scalable I/O is perhaps the most overlooked area of HPC R&D, and given the information generating capabilities being installed and contemplated, it is a mistake to continue to neglect this area of HPC. Many areas in need of new and continued investment in R&D and standardization in this crucial HPC I/O area have been summarized in this document.