S3 - 82

[illegible]

Cooperative High-Performance Storage in the Accelerated Strategic Computing Initiative

Mark Gary, Barry Howard, Steve Louis, Kim Minuzzo, Mark Seager
Lawrence Livermore National Laboratory[1]
P.O. Box 808
Livermore CA 94550
mgary@llnl.gov
Tel: 510-422-7527
Fax: 510-423-8715

Abstract

The use and acceptance of new high-performance, parallel computing platforms will be impeded by the absence of an infrastructure capable of supporting orders-of-magnitude improvement in hierarchical storage and high-speed I/O. The distribution of these high-performance platforms and supporting infrastructures across a wide-area network further compounds this problem. We describe an architectural design and phased implementation plan for a distributed, cooperative storage environment (CSE) to achieve the necessary performance, user transparency, site autonomy, communication, and security features needed to support the Accelerated Strategic Computing Initiative (ASCI). ASCI is a Department of Energy (DOE) program attempting to apply terascale platforms and problem-solving environments (PSEs) toward real-world computational modeling and simulation problems. The ASCI mission must be carried out through a unified, multi-laboratory effort, and will require highly secure, efficient access to vast amounts of data. The CSE provides a logically simple, geographically distributed, storage infrastructure of semi-autonomous cooperating sites to meet the strategic ASCI PSE goal of high-performance data storage and access at the user desktop.

Introduction

The Accelerated Strategic Computing Initiative (ASCI) [1] is a critical element of the Department of Energy (DOE) response to recent decisions ending nuclear testing. In the past, a large part of integrating fundamental science into nuclear weapon safety, reliability, and performance was accomplished through underground testing. In the future, the simulation capabilities provided by ASCI will provide much of that integration. The DOE Science-Based Stockpile Stewardship (SBSS) program, which encompasses ASCI, plans to build upon a strong foundation of simulation capabilities, non-nuclear testing, and basic science to assess the performance of nuclear stockpile systems, predict their safety and reliability, and certify their functionality.

A primary ASCI strategy is to facilitate an unprecedented level of cooperation among the three DOE Defense Programs (DP) laboratories (Lawrence Livermore, Los Alamos, and Sandia National Laboratories). These three DP labs (Tri-Lab) will share ASCI code development, computing, storage systems, and communication resources across laboratory boundaries in joint development efforts, while still maintaining a high degree of local

[1] This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract number W-7405-Eng-48, and supported by DOE Defense Programs Accelerated Strategic Computing Initiative funding.

autonomy. Together with a philosophy of creating a cooperative management environment, there are several technical strategic thrusts within ASCI. Each of these technical strategies influences the need for a high-performance, cooperative storage environment.

ASCI is developing new software applications for virtual testing and prototyping that integrate 3-D capabilities, fine spatial resolution, and accurate physics. These applications will generate much larger quantities of simulation data (multiple petabytes) than ever before. Access to other large data collected from non-nuclear testing and historical underground tests will also be needed to verify and validate results of new applications. ASCI is pushing the computing industry to develop high-performance computers with processing speeds and memory capacities thousands-of-times greater than currently available computers, and tens-to-several-hundred-times greater than the largest computers likely to result from current trends. Much larger capacities and data transfer rates will be required from ASCI storage resources to keep up.

ASCI will require an effective problem-solving environment (PSE) to support application code teams. The PSE provides a robust computing environment where codes may be rapidly developed. The PSE also provides an infrastructure to allow applications to execute efficiently on ASCI compute platforms and to allow easy accessibility to terascale resources from the desktop. This computational infrastructure will consist of local-area and wide-area high-speed networks, software development and visualization tools, and advanced storage facilities (see Figure 1).

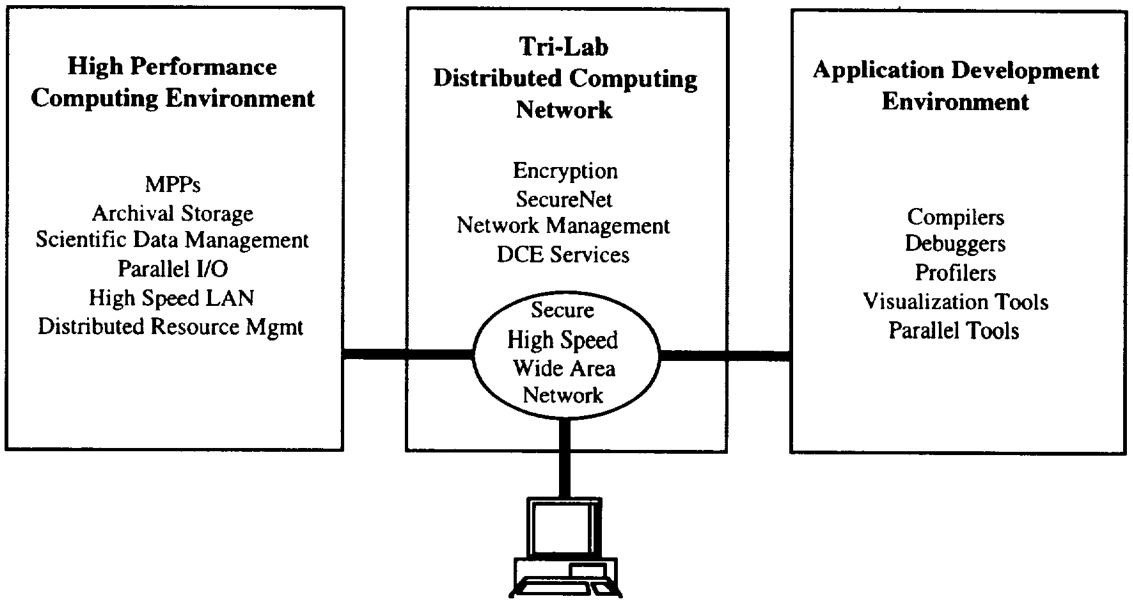
High Performance Computing Environment
MPPs
Archival Storage
Scientific Data Management
Parallel I/O
High Speed LAN
Distributed Resource Mgmt
Tri-Lab Distributed Computing Network
Encryption
SecureNet
Network Management
DCE Services
Secure High Speed Wide Area Network
Application Development Environment
Compilers
Debuggers
Profilers
Visualization Tools
Parallel Tools

Figure 1. The ASCI Problem Solving Environment

Motivation

To make effective use of terascale platform and application capabilities being developed in ASCI, scientists will need to store, access, and manipulate unprecedented amounts of data. The inability of existing data storage systems and I/O capabilities to perform these tasks are now being recognized. Recent projects, such as the San Diego Supercomputer Center initiatives in data-intensive computing and massive data analysis, have started to explore better architectures for multi-teraFLOP computing, multi-petabyte storage, integrated databases and archives [2]. Without improvements, users will spend significant time and effort working around storage problems and I/O bottlenecks, and therefore be unable to realize gains from the ASCI terascale environment. The ASCI PSE concentrates heavily on high-performance archival storage, hierarchical storage management, and scalable I/O to address successful use of data. Other important areas in the PSE, closely related to storage but outside the scope of this paper, are scientific data management and visualization (i.e., tools to aid locating and understanding data), parallel file system advancements (i.e., mechanisms for intelligent application use of scalable I/O), and high-speed interconnect fabrics (i.e., networks to provide efficient local and remote transmission of data).

ASCI is an application-driven effort, and as such, an overall PSE goal is to develop needed software and deploy necessary hardware to intelligently present designers and developers with all of the resources they will need. This view of convenient one-stop, ***black-box*** desktop access implies that individual components comprising the overall resource environment must cooperate. Requirements of cooperative storage differ from those of the scientific data management and visualization areas of the PSE. Cooperative storage is primarily a data logistics problem, concerned with how data is moved, shared, supplied, and stored. In contrast, scientific data management is more accurately a data semantics problem, concerned with finding and understanding meaningful data. Both are needed to provide a true *end-to-end* data management solution within ASCI.

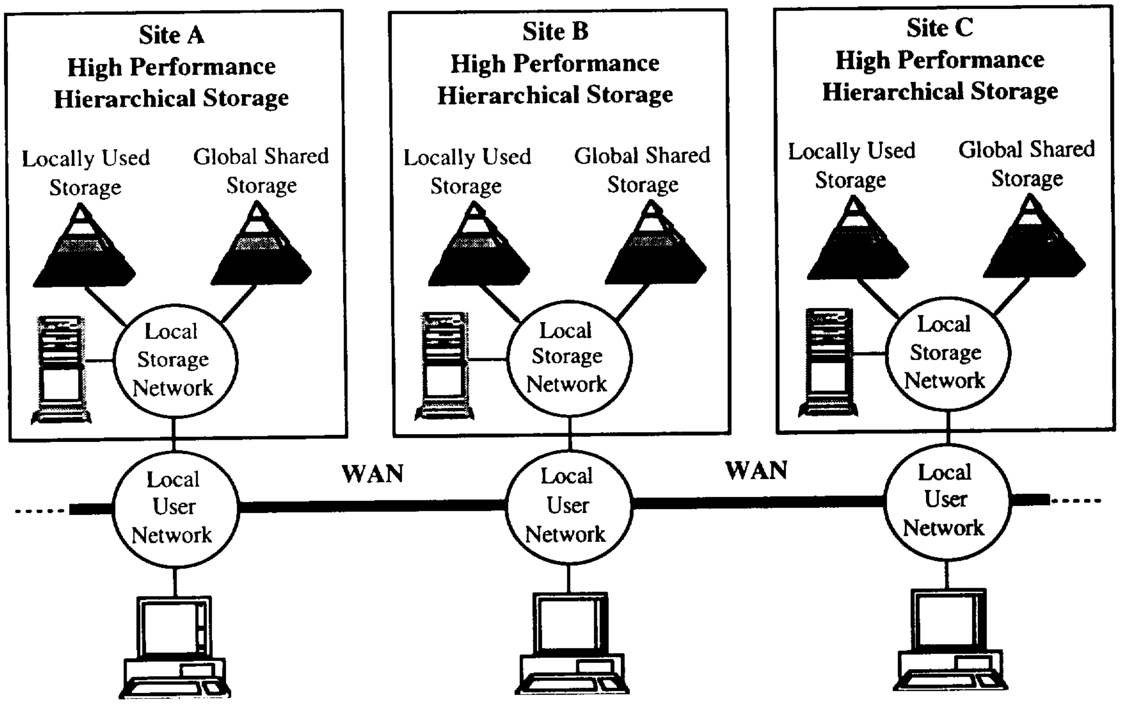

Figure 2. Integration of Autonomous HSM Systems

In this paper, we concentrate on the data logistics problem. We define cooperative storage as an *interworking* group (sometimes called a *federation*) of autonomous, hierarchical storage management systems (HSMs), joined to meet terascale computational needs through common shared vision and coordination, but employing local integration and administration (see Figure 2). HSMs provide management of a hierarchical set of storage devices and file *stewardship* services to its clients. Individual systems tend to view their resources as local and other systems' resources as remote. Within ASCI, systems will contribute part of their storage resources to the cooperative ASCI whole in compliance with well-defined Tri-Lab global policies. At the same time, systems can also provide local resources that are not sharable, again in agreement with management policy.

Coordination of networked block-data servers to implement a distributed, parallel data storage system has been investigated in other work, notably the Image Server System network-distributed data server [3]. Still other projects have explored integration of mass storage with file systems [4,5]. Unlike those efforts, ASCI attempts to integrate autonomous HSMs into a cohesive whole across a high-speed WAN, each HSM composed of multiple servers distributed across a local network. The NASA EOSDIS Project [6] has similar HSM requirements in its attempt to build a globally distributed, heterogeneous, autonomous data system for the Earth Sciences community. EOSDIS is based on a cooperative architecture where data archiving and distribution services can be configured to achieve a balance between centralized management and distributed authority.

Requirements

There exists a rich literature on the requirements of distributed file systems [7]. In this section, we investigate critical requirements that an ASCI cooperative model places on HSMs. Many distributed file system and cooperative HSM requirements overlap due to common issues of resource sharing and remote access. ASCI will require its terascale resources to be viewed in uniform ways (i.e., as part of a seemingly single, local, scalable resource space that can be globally optimized). Scalability becomes especially important as systems grow more complex. One of the ASCI PSE goals is to bring what appears to be unlimited storage and processing capacities to the desktop.

Transparency

In a black-box view, users should not be concerned with how, where, or in what format data is stored. Various transparencies can be provided to reduce or hide the complexity of system interaction and internal detail, but there may be disadvantages if efficiency is lost in resource utilization or performance. We discuss some important transparencies for ASCI below.

Access and location transparency are related. Access transparency (sometimes called network transparency) hides whether objects or services are local or remote. Location transparency hides where objects or services are physically located. Both help make a system easier to use. A related transparency is migration transparency, where users are unaware that an object or service has moved. These are key requirements of a cooperative storage design. In many systems, a user often must explicitly know where objects (typically files) are stored. An example is FTP access to a remote site where, to obtain data, a user must 1) know where the data is, 2) contact that site explicitly, and 3) login to view or access files. ASCI's black-box philosophy will require that the burden of multiple access methods and needing to specify data location be eliminated or made easier for users and applications.

A related transparency is data representation transparency, where users are not aware that different data representations may be used in local and remote parts of the system. FTP may require the user to specify binary or ASCII for proper data transfer. Although standard access interfaces such as FTP are required and will be supported by ASCI, distributed file system interfaces that can better support these transparencies will also required. An important ASCI requirement is that data stored using a specific type of interface be accessible by another. For example, a file stored using a local parallel file system interface should be accessible to the user (at the desktop) through a distributed file system interface. Note that, although the data may have been stored in parallel, actual distributed file system access may be serial. The data may also be physically stored in several different formats throughout the HSM.

Another transparency is naming. We define naming transparency as the ability to obtain a single namespace view, typically as a directory structure, independent of data location or access interface. Storage objects should have globally unique identifiers independent of resource location and access mechanism. Satisfying the naming transparency requirement presents users (logged into any part of the ASCI cooperative environment) with identical namespace views of storage resources.

Two other transparencies are replication and administration. Replication transparency hides the fact that objects or services may be replicated within the system (usually for performance or reliability reasons). Clients may not necessarily know if objects or services are replicated, and services may not know if clients are replicated. Administration transparency provides a management process with the same view of manageable resources, independent of the location of those resources. This allows administrators to assemble resources to fulfill user needs without an obligation to know what resources were local or remote. Such administrative transparency will, of course, be constrained by global management policies.

Site Autonomy

While transparency eases access and use, maintaining local autonomy or control of local resources is also important. Some sites will have local policies that they may be required to enforce, such as security, accounting, scheduling, and system administration. A site may also need to restrict access to particular data or hardware resources on a *need-to-know* basis, or may simply desire a greater degree of control over local assets. There are failure scenarios where a site may need to isolate itself from the cooperative whole while still providing service to local users. Likewise, sites need to be able to operate, possibly in a degraded fashion, when other sites or components of the cooperative are unavailable. To support these situations, interworking sites must strike a balance between local management of resources through site-specific policies and continuing to function as a usable part of the cooperative environment. The introduction of greater autonomy may result in much more visible component boundaries to users.

Communication and Security

Because extending the reach of scientists from the desktop to Tri-Lab resources is key, an ASCI cooperative storage infrastructure must not be isolated from the communication and security infrastructure shared by other ASCI elements. The ASCI communication and security infrastructure provides services and tools to support creation, use, and management of distributed applications in a heterogeneous environment. It also provides services that allow distributed applications to interact securely with a collection of possibly heterogeneous computers, operating systems, and networks, as if they were a single system. It is a requirement that cooperative storage work with, and be manageable within, this infrastructure. It would also be prudent for the HSM's internal component interactions to use this same communication and security infrastructure.

Performance
Even if all of the requirements listed above are satisfied, a system must meet minimum performance requirements for its users. Providing transparency, autonomy, security, and communication is valueless if the software and hardware resources providing them are unable to provide data storage and access in acceptable time. For ASCI, performance will be a dominant requirement.

Architecture

To satisfy the above requirements, the ASCI PSE effort has established a preliminary architectural vision for a cooperative high-performance storage environment (CSE). This vision brings together distinct elements of storage to form a cohesive, unified, orderly whole. The desired result is a logically single, geographically distributed, storage system of semi-autonomous cooperating sites. The system will be designed and implemented to meet the goal of high-performance data access at the desktop. In this section, we describe aspects of a cooperative architecture that target our requirements. We address each area and discuss the planned architectural approach to satisfy the requirement. ASCI is a multi-year effort, and therefore a phased implementation approach is planned.

Achieving Transparency
To achieve the transparencies described in the requirements section, our approach uses interfaces that naturally provide transparency to users, and tie those interfaces through additional linkage of Tri-Lab name spaces and data spaces. This provides ASCI users of non-transparent interfaces many of the beneficial properties afforded users of transparent ones. The CSE will use DFS [8] as a key file system interface. This interface provides users with a serial interface to storage that satisfies many transparency requirements. Interfaces to local NFS systems will also exist and be supported by CSE HSMs. We chose to concentrate on DFS because it is better able to support the wide-area requirements of ASCI and the PSE middleware infrastructure.

DFS uses a set of cooperating clients and servers to provide geographically separated users with a single, seamless view of a distributed name and data space. DFS also has enhanced data caching capability and consistency guarantees. While DFS satisfies several transparency requirements of ASCI, it lacks an archival back-end needed to support the petabytes of data generated by ASCI. To address this problem, we plan to exploit an extended definition of the DMIG DMAPI [9] specification to link DFS with an appropriate back-end HSM capable of satisfying the large storage requirements of ASCI. An HSM-DMAPI gateway will provide interoperability and communication between the DFS Server and the HSM (see Figure 3).

Another deficiency of DFS is that it does not currently satisfy the high-performance, parallel-access requirements of many ASCI applications and users. Overcoming the serial performance shortcomings of DFS will be challenging. We plan to investigate ways to expand the DFS interface for parallel transfer of data, where possible. While we hope to enhance DFS data transfer performance, we also need to provide other user interfaces capable of satisfying the performance requirements of ASCI. To satisfy transparency requirements, these storage interfaces will all need to operate within a common namespace shared by all user storage interfaces.

The strategy to allow all storage interfaces to view, and operate within, a single namespace is grounded in two architectural decisions. The first is that the namespaces of interworking sites will be linked together, thus allowing users to navigate a single global namespace. This linking will be accomplished by allowing the software that implements naming at each site to *link* leaves of its directory structure to points in other site namespaces. This approximates the idea of a UNIX™ mount, but more accurately mimics a UNIX soft link.

The second architectural decision concerns namespace consistency. Storage interfaces that have their own namespaces by design (e.g., DFS), must be integrated into the CSE such that the namespace provides a global namespace view. If interfaces are implemented as multiple namespaces joined to provide a single view, they must be kept consistent with other views. This presents a significant challenge. For example, back-ending DFS with an archival HSM as was done in [5] will not be enough. We require that the DFS namespace provide users with access to all data (given proper authorization) created by every interface in the CSE. This means that a file created by a non-DFS parallel interface will need to appear in the DFS interface's namespace.

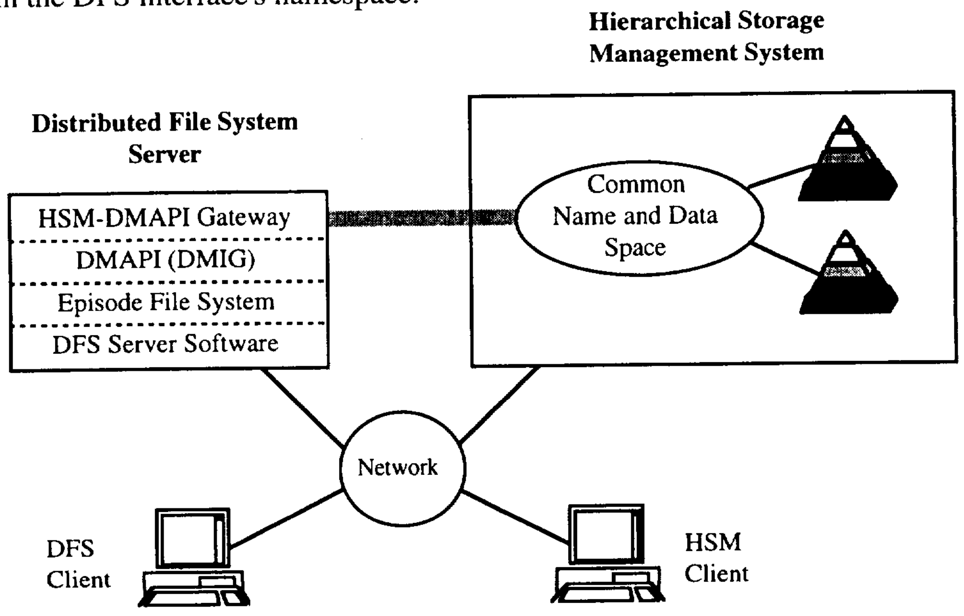
Hierarchical Storage Management System
Distributed File System Server
HSM-DMAPI Gateway
DMAPI (DMIG)
Episode File System
DFS Server Software
Common Name and Data Space
Network
DFS Client
HSM Client

Figure 3. Integration of DFS and HSM Name and Data Space

The above architectural decisions lead to a design with location, access, and name transparencies, but do not necessarily satisfy requirements for administration or replication transparency. Administrative transparency implies the ability to assemble resources located across interworking sites into useful configurations, taking advantage of the strengths and bypassing the weaknesses of each site. Later phases of CSE design encompass linking the storage system management (SSM) internals of the server components of each site together and with all server components. This enables construction of hierarchies that span geographically separated sites. There are also plans for introducing future file and physical volume replication capabilities into the CSE that can span sites.

Achieving Site Autonomy

Satisfying site autonomy requirements in an architecture that links name spaces and joins administrative domains is difficult. Our design maximizes site autonomy within a single system through incorporation of policy modules, and by allowing a site to continue functioning, albeit in degraded fashion, when disconnected from other participants in the CSE. Similarly, sites remaining in the CSE are allowed to operate in the absence of one or more sites.

Policy modules allow enforcement of SSM concerns that are likely to differ from site to site by isolating local rules or decisions into separate modules that may be implemented or modified by each site to suit its own needs. The CSE design provides for integration of policy modules by establishing well-defined interfaces to and from storage service components. Modification of the main components of the system is not required; only the policy modules need to change. An example is implementation of accounting. The CSE maintains a well-defined interface to an accounting policy module. As accounting events are encountered, policy module interfaces are used to communicate accounting information. The module that accepts this information will be customized by the site and will then perform site-specific actions appropriate to log, report, or ignore the information. The CSE design currently includes policy interfaces for accounting, scheduling, security, migration and purge and system management. The modeling of policies as general objects in distributed systems is explored in considerable detail in [10].

Architecting the system to allow independent operation, in the face of communication failure or other adverse events, relies on how server components of the CSE are linked. If servers are connected in a way that requires all information to be successfully exchanged over inter-site communication lines for a server to function, then autonomy will be lost. If servers are instead connected through simple remote location linking information, more autonomy is possible. The CSE uses the latter approach. Imagine two name server components residing within separate sites (see Figure 4). Each name server represents an autonomous directory structure that may have leaf nodes that are links or pointers to directory or file resources located in the other name server's structure. When traversing a pathname that crosses one of these links, the CSE is designed to realize that the next component is actually managed on a remote name server and contact that server for continued parsing of the path name.

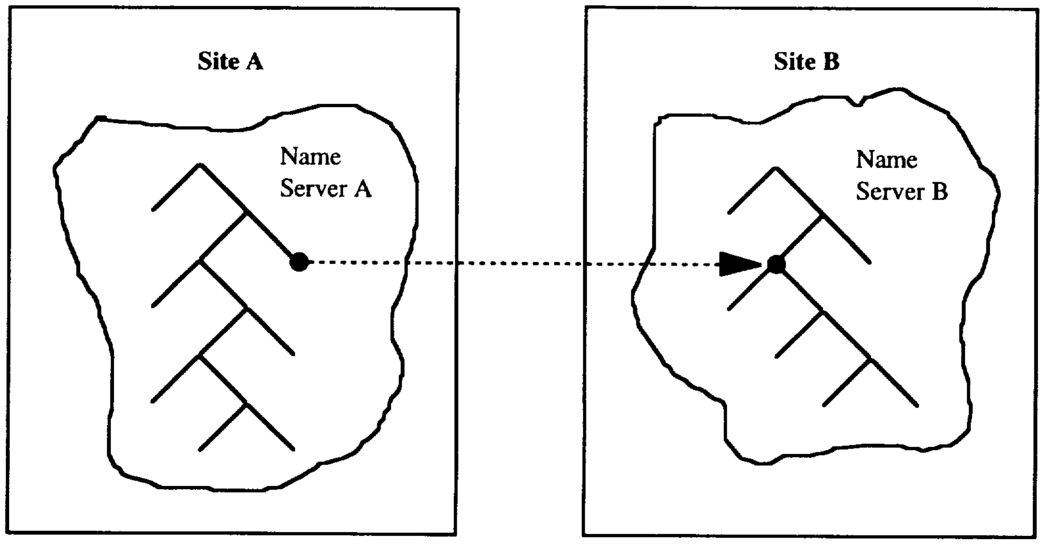

Figure 4. Linked Name Servers at Multiple Sites

Under the above scenario, one can imagine a situation where Name Server B is not reachable. Site A still has the ability to traverse all of its own namespace, but cannot access names located in Site B's space. In this case, depending on the distribution of data across sites, the absence of Site B will degrade access to data, even if names are accessible. This demonstrates a tradeoff between autonomy and administration transparency. Typically, as administrative transparency increases, site autonomy decreases. Striking a balance between these two requirements will affect the eventual implementation. Note that if an HSM is well designed and thus able to scale within a single site, many difficulties linking multiple sites will be moot. In the linked name server example above, one could have easily considered the two name servers to be at a single site, scaled because the namespace was too large to be supported by a single name server. Single-site name server scalability requires the ability of pathnames to span local name servers. Thus, solving this problem addresses many issues concerning multi-site name server cooperation.

Communication and Security Infrastructure

Because the ASCI environment is built on a wide-area fabric of distributed services, it is important that the CSE function as an integral part of this fabric. In our design, the HSM is constructed using this fabric to tie together its various distributed components. The ASCI PSE uses OSF's Distributed Computing Environment (DCE) as a convenient middleware layer to tie ASCI PSE components together. DCE provides services and tools that support the creation, use, and maintenance of distributed applications and servers in a heterogeneous computing environment. DCE also provides services that allow distributed applications and servers to interact securely with a collection of possibly heterogeneous computers, operating systems, and networks as if they were a single system. The CSE uses these services as the glue between each of its server components as well as for the mechanism linking HSMs with their clients and other components of the ASCI environment.

Because much of the work to be done within ASCI is classified and requires the use of *need-to-know* boundaries, authorization techniques will be critical to successful information sharing. The DCE Security Service maintains a replicated registry database that includes principals, users, groups, organizations, accounts and administrative policies. Sites within the CSE can make use of this security service by implementing calls to the service within their site-specific policy modules. Access Control List (ACL) entries in the database are used to define and grant permissions to an object. Any request by a user or application to use an object, such as an archival storage device, is accompanied by the requester's credentials which are checked against the ACL for that service.

Of equal importance to the authentication and authorization mechanisms provided by DCE is the security of data itself as they traverse communication networks. The security of WAN communications between distributed ASCI sites is provided by networks front-ended by high-performance encryptors, providing NSA-approved encryption of all data passing between classified ASCI sites.

Achieving Performance
As stated above, the transparencies inherent in the CSE architecture are not really useful if minimum performance needs of applications and users are not met. If performance is lacking, users will recognize the difference in speed when accessing data on remote systems, and as a result will find ways of working around the system to avoid delays, often at the expense of the system as a whole.

To support the performance requirements of ASCI, the CSE uses a scalable, parallel architecture to support high-performance data access. Under this design, sites can enhance performance by adding server components and/or wider stripes of data until a target performance level is achieved. For example, if a site requires a 100 megabyte/sec transfer rate for a file, but only has 20 megabyte/sec devices, the CSE allows the site to attain the desired transfer rate by storing data using a five-wide stripe across multiple devices at a site.

While the architecture we have chosen provides, in theory, the performance necessary to satisfy the requirements of ASCI, realizing this goal is highly dependent on additional advances in networking hardware, protocols, and encryption technology. Without gains in these areas matching the I/O performance gains seen in storage devices and platforms, the functionality required by ASCI will not be easily obtainable. The CSE architectural design and implementation will closely track these disciplines and react as necessary to integrate advances in these areas as they become available.

Implementation

The ASCI implementation strategy for a CSE focuses heavily on the High Performance Storage System (HPSS), an on-going effort to develop a scalable, high-performance, HSM for data-intensive applications and large-scale computers. Coordinated by IBM Government Systems, three of the principal developers for HPSS are also the three ASCI DP laboratories: Lawrence Livermore, Los Alamos, and Sandia National Laboratories. Oak Ridge National Laboratory, other federal agencies, and universities also participate in HPSS development. From the beginning, a major motivation for HPSS was to develop a high-speed storage system providing scalability to meet demands of new high-performance computer applications where vast amounts of data are generated, and to meet the needs of a national information infrastructure [11].

HPSS has a scalable, networked-centered architecture [12] and is based on the concepts of the IEEE Mass Storage System Reference Model, Version 5 [13]. The architecture includes a high-speed network for data transfer and a separate network for control. The control network uses OSF/DCE Remote Procedure Call technology. In an actual implementation, control and data transfer networks may be physically separate or shared [14]. Another key feature of HPSS is support for both parallel and sequential I/O. The parallel I/O architecture is designed to scale as technology improves by using data striping and multiple data movers [15]. HPSS was designed to support data transfers from hundreds of megabytes up to multiple gigabytes per second. File size scalability must meet the needs of billions of data sets, some potentially terabytes in size, for total storage capacities in petabytes.

The scalability features of HPSS are important for enabling distribution and cooperation of storage resources. The ability to introduce multiple, distributed servers as needed into an HPSS implementation is critical for both performance and autonomy reasons.

Distributable HPSS servers allow us to obtain a geographically distributed single-system view required by ASCI applications and users. Because HPSS supports network-attached peripherals [16,17] and third-party data transfer capabilities, new hardware can easily be added to provide incremental performance scalability. Each ASCI site maintains a degree of local autonomy and can tailor data rates and capacities needed for its local requirements and for resources shared across the Tri-Lab environment. The tailoring of storage resources and the access to them can be controlled and used through the HPSS Class of Service structures [18] and device hierarchy structures. Class of Service provides the control and management flexibility needed to implement a truly distributed storage hierarchy. HPSS also supports policy modules and uses a management-by-policy philosophy in its storage system management design.

The ASCI PSE working group defines a four-phase approach to realize a fully cooperative storage environment. These phases, described in the next sections, are:

- Independent HPSS systems linked via an encrypted WAN
- Independent HPSS systems with a single OSF/DCE DFS name space
- Cooperative distributed systems linked at the internal server level
- Cooperative distributed systems with single system/management view

Phase I

The first phase establishes independent autonomous HPSS systems at each of the Tri-Lab sites. Each site will operate a stand-alone HPSS system with the only linkage being a high speed encrypted WAN connecting each site (see Figure 5). Over this WAN, users will be able to use explicit interfaces such as FTP (assuming they are granted remote privileges) to communicate with remote sites. No transparency will be provided for remote access, but site autonomy will be more or less total. During this phase, significant planning and coordination will be undertaken to ensure that the site policies and DCE configurations at each site do not preclude any cooperation necessary when migrating to future implementation phases.

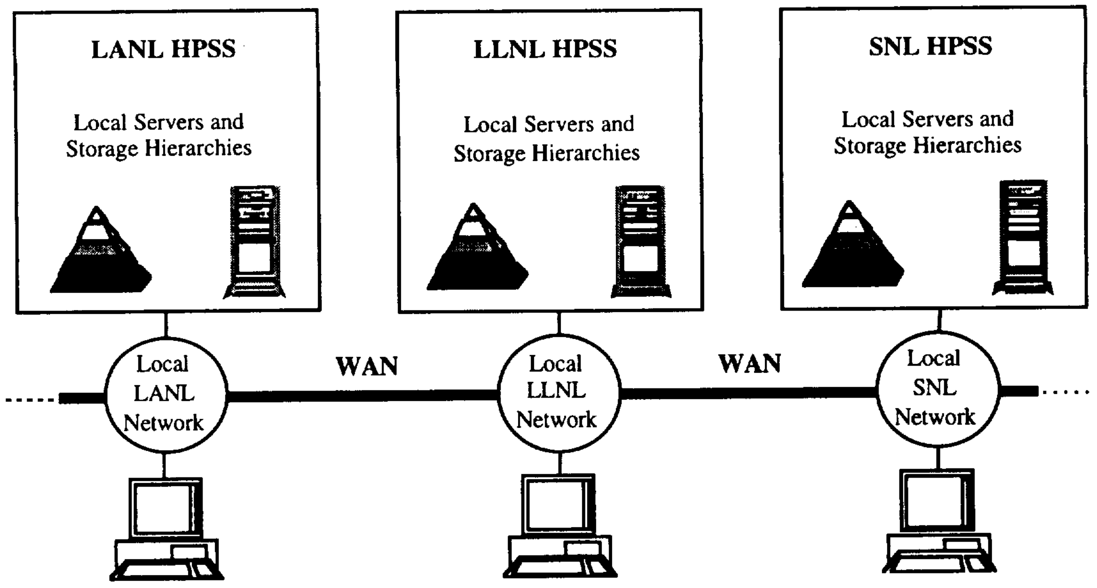
LANL HPSS
Local Servers and Storage Hierarchies
LLNL HPSS
Local Servers and Storage Hierarchies
SNL HPSS
Local Servers and Storage Hierarchies
Local LANL Network
WAN
Local LLNL Network
WAN
Local SNL Network

Figure 5. Phase I - Independent HPSS Systems

Phase II

In Phase II, the Tri-Lab sites will attain linkage of name spaces and gain some access transparency, using DFS. Each of the sites will support a DFS server or servers that are back-ended by HPSS (see Figure 6). The DFS namespace will export the view of each site's files to other sites that have linked their DFS servers to the global DFS space. Files created by DFS clients will be created and stored in the HPSS system that back-ends the DFS server where the new file is placed. Users will be able to access files at all sites using DFS, but will have to explicitly contact the location that stores a file to access the resource using any interface other than DFS. Thus in Phase II, the only interface that can see and access all of the storage resources at all sites without explicit login to remote sites will be DFS. Although the DFS interface provides functional transparency to users in this phase, users will be able to discern latency when accessing remote files. Note that data stored in parallel at a remote site will be accessible serially through DFS and other serial interfaces. Phase II introduces the first transparency aspects to the CSE while maintaining all of the autonomy provided in Phase I.

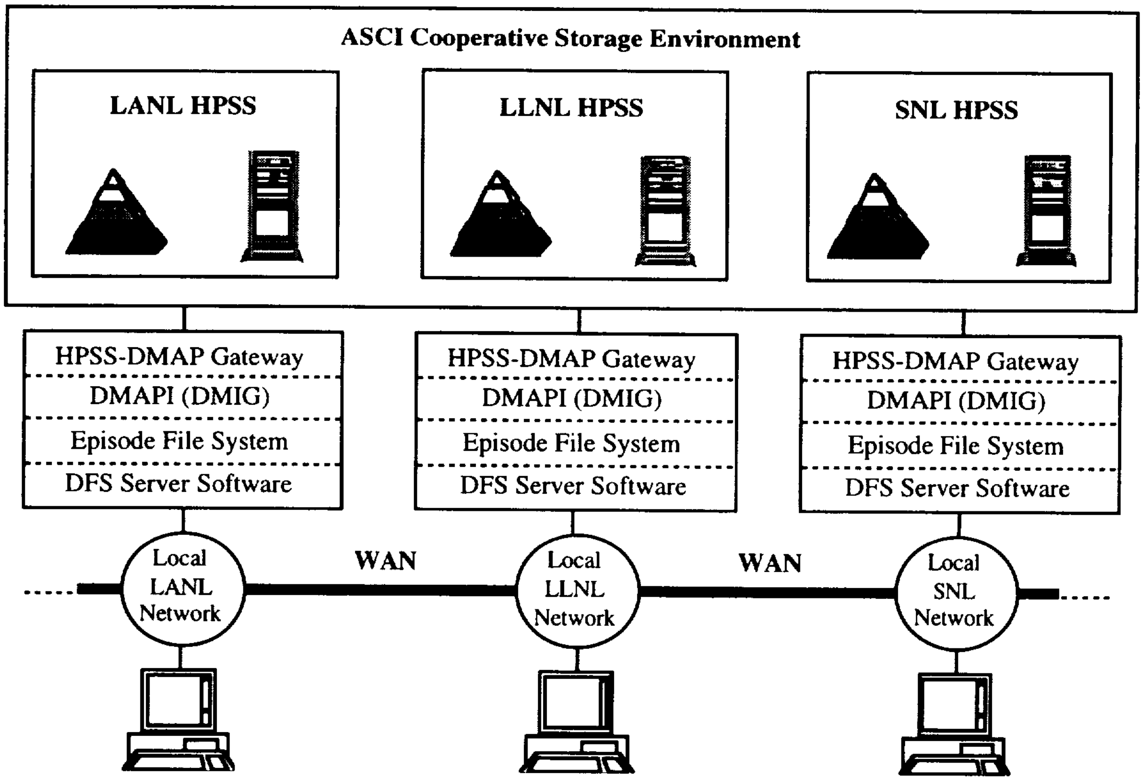

Figure 6. Phase II - Linked HPSS Namespaces with DFS

Phase III

Phase III builds on Phase II by linking the Name Servers of each site's HPSS system and allowing clients to seamlessly communicate with other sites for data access when necessary. The Phase III feature making this possible is implementation of links within a namespace pointing to a directory or file in the namespace of a remote HPSS system (see Figure 7). Parsing a pathname that crosses a link between two sites will be performed transparently, as will remote data access. This provides the ability for all CSE user interfaces to automatically and transparently view or access the namespace of all Tri-Lab sites.

In Phase III, given adequate networking performance, many of our transparency requirements are met while still maintaining a high level of site autonomy. Because each site still runs its own HPSS system, it can function independently, possibly in degraded fashion, when other Tri-Lab sites are unreachable. In such a failure scenario, all accesses to files or names stored at remote sites will fail. Because users will be allowed to use remote resources, Tri-Lab sites will have some common aspects of storage system management. While this in-common management does not preclude use of unique policy modules at each site, it will be necessary to work out issues such as accounting for remote user accesses. While Phase III goes far toward the final vision of a single distributed HSM, it does not achieve administrative transparency.

Phase IV

Phase IV maintains all functionality of Phase III, but adds administrative transparency across the Tri-Lab sites. This addition will be made possible through linking and coordination of the Storage System Management components of each site. This allows an administrator at one site, given adequate privileges, to assemble storage resources existing at multiple sites into storage hierarchies that take optimum advantage of the array of capability and capacity devices distributed throughout the Tri-Lab environment. The addition of administrative transparency capabilities yields a single logical HSM from a management view, but results in some sacrifice of site autonomy as hierarchies are built requiring resources at multiple sites to all be available in order to provide service. The level of autonomy lost will be related to how much inter-site sharing of resources occurs.

How far the CSE implementation should proceed is a matter of debate. Accomplishing Phase III may satisfy those CSE requirements deemed most critical without sacrificing too much site autonomy. On the other hand, Phase IV, or a slightly enhanced version of Phase III implementing rudimentary administrative transparency, may provide ASCI with more optimal capabilities. A third perspective is to de-emphasize site autonomy. This position argues that a geographically distributed, yet single HPSS system is best for ASCI (note that Phase IV as presented is a *logically* single combination of independent HPSS systems). Time, experience, and the performance of remote data transfers will determine the proper choice.

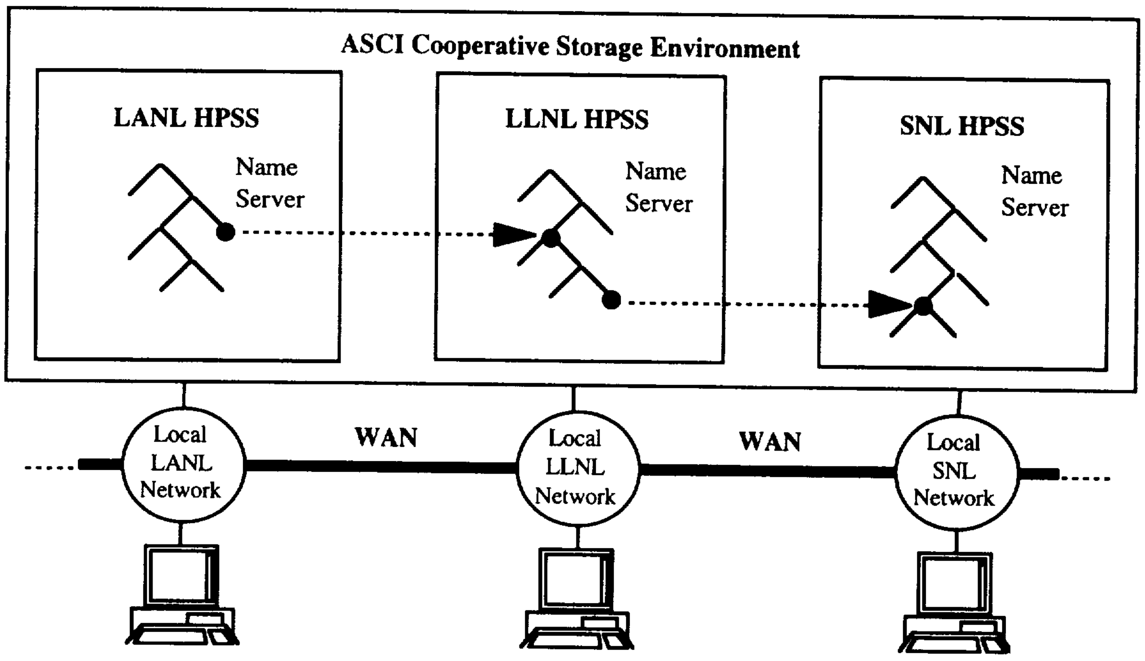

Figure 7. Phase III - Linked HPSS Servers Across Sites

Current Status

We plan to continue the long-standing DOE collaboration to develop HPSS with assistance from ASCI funding. HPSS Release 3, containing over 500 integration and system tests and approximately 200,000 lines of executable code, was released on June 30, 1996. Work continues in 1996 on the next deliveries of HPSS (known as Release 3+ and 4). In 1997, all three ASCI DP sites plan to deploy Release 3+ in production classified and unclassified computing environments. These three systems will initially function independently as described in the Phase I description above. Work on DFS/HPSS integration is on-going to meet the implementation described above as Phase II, as well as work to meet some of the scalability requirements of Phase III. Work is also underway at Lawrence Livermore to provide support in HPSS for the emerging MPI-IO interface standard [19,20] as a basis for parallel I/O portability in ASCI applications.

Conclusions

ASCI is an application-driven program. The fundamental goal of ASCI's Problem Solving Environment program is to give users the tools they need to do their job. This means ensuring that the power of the ASCI application/platform combination can be readily applied to the challenges of science-based stockpile stewardship. This will require the development of a balanced, supporting infrastructure far more capable than any available today. In particular, hierarchical storage management, archival storage, and parallel I/O must scale together with the growth in platform capability.

We have discussed the critical requirements that an ASCI cooperative model places on a storage infrastructure. ASCI will need terascale storage resources to be available from the desktop and viewed as a black-box. Users should not be concerned with how, where, or in what format data is stored. Several transparencies must be provided to reduce or hide the complexity of system interaction and internal detail, but must be balanced such that efficiency is not lost in resource utilization or performance. There are critical site autonomy, communication, and security issues that must also be considered.

We described an architectural design and phased implementation plan for a distributed, cooperative storage environment (CSE) to achieve the necessary performance, user transparency, site autonomy, communication and security features needed to support ASCI requirements. The CSE provides a logically single, geographically distributed, storage infrastructure of semi-autonomous cooperating sites. The ASCI implementation strategy focuses heavily on integration of the High Performance Storage System (HPSS), a scalable, high-performance system for data-intensive applications and large-scale computers, and OSF/DCE DFS, a popular distributed file system. While DFS satisfies several transparency requirements of ASCI, it lacks the archival back-end and performance needed to support the petabytes of data generated by ASCI. We have shown how we plan to exploit the DMIG DMAPI interface to provide DFS with an appropriate back-end capable of satisfying ASCI's large storage requirements, and a phased implementation approach satisfying the cooperative storage requirements of ASCI.

References

[1] Accelerated Strategic Computing Initiative (ASCI) Program Plan, Final Draft, DOE Defense Programs, May 1996.

[2] R. W. Moore, "A Petabyte/Teraflops System: Meeting Future Data Needs," San Diego Supercomputer Center Gather/Scatter, Vol. 11, No. 4, December 1995.

[3] B. Tierney, W. Johnston, L. Chen, H. Herzog, G. Hoo, G. Jin, J. Lee and D. Rotem, "Distributed Parallel Data Storage Systems: A Scalable Approach to High Speed Image Servers," *Proceedings of ACM Multimedia '94*, October 1994.

[4] C. Antonelli and P. Honeyman, "Integrating Mass Storage and File Systems," Technical Report 93-2, University of Michigan Center for Information and Technology Integration, Ann Arbor, MI, April 15, 1993.

[5] J. Nydick, K. Benninger, B. Bosley, J. Ellis, J. Goldick, C. Kirby, M. Levine, C. Maher and M. Mathis, "An AFS-based Mass Storage System at the Pittsburgh Supercomputer Center," *Proceedings of the Eleventh IEEE Symposium on Mass Storage Systems*, Monterey, CA, October 7-10, 1991.

[6] B. Kobler, J. Berbert, P. Caulk and P. Hariharan, "Architecture and Design of Storage and Data Management for the NASA Earth Observing System Data and Information System (EOSDIS)," *Proceedings of the Fourteenth IEEE Computer Society Mass Storage Systems Symposium*, Monterey, CA, September 11-14, 1995.

[7] E. Levy and A. Silberschatz, "Distributed File Systems: Concepts and Examples," *ACM Computing Surveys*, Vol. 22, No. 4, December, 1990.

[8] K. Kazar, B. Leverett, O. Anderson, V. Apostolides, B. Bottos, S. Chutani, C. Everhart, W. Mason, S. Tu and E. Zayas, "DEcorum File System Functional Overview," *Summer USENIX Conference Proceedings*, Anaheim, CA, June 1990.

[9] Data Management Interfaces Group, "Interface Specification for the Data Management Application Programming Interface," Version 2.3a Draft X/Open Submission, January 1996.

[10] J. Moffett and M. Sloman, "The Representation of Policies as System Objects," *Proceedings of the Conference on Organizational Computer Systems (COCS '91)*, Atlanta, GA, November 5-8, 1991.

[11] R. Coyne, H. Hulen, and R. Watson, "The High Performance Storage System," *Proceedings Supercomputing '93*, Portland, OR, November 15-19, 1993.

[12] D. Teaff, R. Coyne and R. Watson, "The Architecture of the High Performance Storage System (HPSS)," *Fourth NASA GSFC Conference on Mass Storage Systems and Technologies, College Park*, MD, March 28-30, 1995.

[13] R. Garrison, et al. (eds.), Reference Model for Open Storage Systems Interconnection: Mass Storage Reference Model Version 5, IEEE Storage System Standards Working Group (P1244), September 1994.

[14] R. Hyer, R. Ruef and R. Watson, "High Performance Direct Network Data Transfers at the National Storage Laboratory," *Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems*, Monterey, CA, April 26-29, 1993.

[15] R. Watson and R. Coyne, "The Parallel I/O Architecture of the High Performance Storage System," *Proceedings of the Fourteenth IEEE Computer Society Mass Storage Systems Symposium*, Monterey, CA, September 11-14, 1995.

[16] R. Van Meter, "A Brief Survey of Current Work on Network Attached Peripherals," abstract published in ACM Operating Systems Review, January 96, or available at http://www.isi.edu/~rdv/netstation/nap-research/napr-extab/napr-extab.html.

[17] D. Wiltzius, "Network-attached peripherals (NAP) for HPSS/SIOF." http://www.llnl.gov/liv_comp/siof/siof_nap.html.

[18] S. Louis, and D. Teaff, "Class of Service in the High Performance Storage System," *Third IFIP TC6 International Conference on Open Distributed Processing*, Brisbane, Australia, February 21-24, 1995.

[19] T. Jones, R. Mark, J. Martin. J. May, E. Pierce and L. Stanberry, "An MPI-IO Interface to HPSS," *Fifth NASA GSFC Conference on Mass Storage Systems and Technologies*, College Park, MD, September 17-20, 1996.

[20] MPI-IO Committee, MPI-IO: A Parallel File I/O Interface for MPI, Version 0.5. http://lovelace.nas.nasa.gov/MPI-IO.