

ATTACHMENT A

STATEMENT OF WORK: SGS File System

DOE National Nuclear Security Administration
& the DOD National Security Agency



April 25, 2001

Table of Contents

| | |
|--|-----------|
| TABLE OF CONTENTS | 2 |
| 1.0 OVERVIEW | 4 |
| 1.1 INTRODUCTION..... | 4 |
| 2.0 MOTIVATION | 5 |
| 2.1 THE NEED FOR IMPROVED FILE SYSTEMS..... | 5 |
| 2.2 I/O CHARACTERIZATION OF IMPORTANT APPLICATIONS..... | 6 |
| 2.3 CURRENT AND PROJECTED ENVIRONMENTS AT LLNL, LANL, SANDIA, AND THE NSA..... | 6 |
| 2.4 SUMMARY OF FIVE TECHNOLOGY CATEGORIES..... | 9 |
| 3.0 MINIMUM REQUIREMENTS (GO/NO-GO CRITERIA) | 12 |
| 3.1 POSIX-LIKE INTERFACE [MANDATORY] | 12 |
| 3.2 INTEGRATION COMPATIBILITY [MANDATORY] | 12 |
| 3.3 NO SINGLE POINT OF FAILURE [MANDATORY] | 12 |
| 4.0 DESIRED PERFORMANCE FEATURES (THE 5 TECHNOLOGY TARGETS) | 13 |
| 4.1 GLOBAL ACCESS | 13 |
| 4.1.1 <i>Global Scalable Name Space [EXTREMELY DESIRED]</i> | 13 |
| 4.1.2 <i>Client software [EXTREMELY DESIRED]</i> | 14 |
| 4.1.3 <i>Exportable interfaces and protocols [EXTREMELY DESIRED]</i> | 14 |
| 4.1.4 <i>Coexistence with other file systems [EXTREMELY DESIRED]</i> | 14 |
| 4.1.5 <i>Transparent global capabilities [EXTREMELY DESIRED to DESIRED (see table 3)]</i> | 15 |
| 4.1.6 <i>Integration in a SAN environment [EXTREMELY DESIRED]</i> | 16 |
| 4.2 SCALABLE INFRASTRUCTURE FOR CLUSTERS AND THE ENTERPRISE | 16 |
| 4.2.1 <i>Parallel I/O Bandwidth [EXTREMELY DESIRED]</i> | 16 |
| 4.2.2 <i>Support for very large file systems [EXTREMELY DESIRED]</i> | 18 |
| 4.2.3 <i>Scalable file creation & Metadata Operations [EXTREMELY DESIRED]</i> | 19 |
| 4.2.4 <i>Archive Driven Performance [EXTREMELY DESIRED]</i> | 21 |
| 4.2.5 <i>Adaptive Prefetching (Desired)</i> | 21 |
| 4.3 INTEGRATED INFRASTRUCTURE FOR WAN ACCESS | 21 |
| 4.3.1 <i>WAN Access To Files [HIGHLY DESIRED]</i> | 21 |
| 4.3.2 <i>Global Identities [HIGHLY DESIRED]</i> | 22 |
| 4.3.3 <i>WAN Security Integration [HIGHLY DESIRED]</i> | 22 |
| 4.4 SCALABLE MANAGEMENT & OPERATIONAL FACILITIES | 22 |
| 4.4.1 <i>Need to minimize human management effort [EXTREMELY DESIRED]</i> | 23 |
| 4.4.2 <i>Integration with other Management Tools [HIGHLY DESIRED]</i> | 23 |
| 4.4.3 <i>Dynamic tuning & reconfiguration [EXTREMELY DESIRED]</i> | 24 |
| 4.4.4 <i>Diagnostic reporting [EXTREMELY DESIRED]</i> | 24 |
| 4.4.5 <i>Support for configuration management [EXTREMELY DESIRED]</i> | 24 |
| 4.4.6 <i>Problem determination GUI [DESIRED]</i> | 24 |
| 4.4.7 <i>User statistics reporting [EXTREMELY DESIRED]</i> | 25 |
| 4.4.8 <i>Security management [EXTREMELY DESIRED]</i> | 25 |
| 4.4.9 <i>Improved Characterization and Retrieval of Files [DESIRED]</i> | 25 |
| 4.4.10 <i>Full documentation [EXTREMELY DESIRED]</i> | 25 |
| 4.4.11 <i>Fault Tolerance, Reliability, Availability, Serviceability (RAS) [EXTREMELY DESIRED]</i> | 25 |
| 4.4.12 <i>Integration with Tertiary Storage [EXTREMELY DESIRED]</i> | 26 |
| 4.4.13 <i>Standard POSIX and MPI-IO [EXTREMELY DESIRED]</i> | 27 |
| 4.4.14 <i>Special API semantics for increased performance [HIGHLY DESIRED]</i> | 27 |
| 4.4.15 <i>Time to build a file system [EXTREMELY DESIRED]</i> | 28 |
| 4.4.16 <i>Backup/Recovery [EXTREMELY DESIRED]</i> | 28 |

4.4.17 Snapshot Capability [*HIGHLY DESIRED* -> *EXTREMELY DESIRED*].....28

4.4.18 Flow Control & Quality of I/O Service [*HIGHLY DESIRED*]29

4.4.19 Benchmarks [*EXTREMELY DESIRED*].....29

4.5 SECURITY.....29

4.5.1 Authentication [*EXTREMELY DESIRED*].....29

4.5.2 Authorization - [*EXTREMELY DESIRED*]30

4.5.3 Content-based Authorization - [*HIGHLY DESIRED*]30

4.5.4 Logging and auditing [*EXTREMELY DESIRED*].....31

4.5.5 Encryption [*DESIRED*]31

4.5.6 Deciding what can be trusted [*EXTREMELY DESIRED*].....31

REFERENCES.....32

GLOSSARY33

1.0 Overview

1.1 Introduction

This document is a Statement of Work for developing new high performance computer file system technology. The solicitation, which is managed from a government program called **PathForward**, anticipates funding several companies to accelerate research and development of potential commercial products. Companies with an interest in furthering the capabilities of high performance computer file systems are encouraged to seek funding in the manner described in this document. The document is composed of four sections: *first*, it begins with an introduction to PathForward; *second*, it describes our motivation for the solicitation; *third*, it provides a description of minimum requirements; and *fourth*, it provides a description of the specific desired technological components.

The goal of this File System PathForward effort is to accelerate file system development activities in five key areas, and to bring these critical technologies or technology enhancements to the marketplace more quickly than they might normally appear:

- ? Global Access
- ? Scalable Infrastructure for Clusters and Enterprise
- ? Integrated Infrastructure for Wan Access
- ? Scalable Management & Operational Facilities
- ? Security.

Promising global file system product development projects that could benefit from additional development funded by the U.S. Department of Energy Accelerated Strategic Computing Initiative (ASCI) and other high performance computing initiatives within other agencies of the U.S. Government will be considered for funding.

The required file system may be characterized as secure, extremely scalable and able to support complex multiple supercomputer sites. It is likely that most global file system development projects have not considered all the ramifications of such an environment, and it is hoped that one or more of these projects would desire to work with the File System PathForward project to add scalability and security capability to help the product in question to scale to enormous proportions in a secure and manageable way. For an example of the PathForward process, please see the computer interconnect technologies PathForward webpage <http://www.asci.doe.gov/scs/path.htm>

2.0 Motivation

2.1 The Need For Improved File Systems

Every year, the DOE meets to discuss barriers that are reducing the Defense Programs (DP) Laboratories' computational capabilities to perform their mission. According to the most recent meeting, the most critical computer science barriers faced by ASCI are I/O related. In fact, the top two barriers, and three of the top five barriers, recognized at Curves and Barriers 2000 are file system related: (1) Data access for visualization; (2) End-to-End I/O throughput between the platforms, visualization, and storage; and (5) Distributed File System deployment.

The DP Laboratories have a roadmap for future file systems and their required attributes. One defining characteristic of these file systems is that even though the data and metadata are available to a variety of machines (i.e., compute engines, visualization engines, archival systems, personal desktop computers), the bandwidth-limiting bottlenecks found in today's shared file systems are eliminated. Common problems of today such as the CPU-intensive nature of protocol stacks, serialization resulting from non-parallel metadata, and/or data servers are removed to permit the tremendous scalability requirements of ASCI.

According to the strategy, the file system should be independent of machine makes (Compaq, Cray, HP, IBM, SGI, Sun, ...); operating systems (AIX, HPUX, Irix, Linux, Solaris, Tru64, Unicos...); and storage device makes (Ciprico, EMC, IBM, Quantum, Seagate, ...). This "best of breed" independence will extend the file system life by allowing supercomputer centers to exploit new technologies while retaining old spindles/devices. Classically most ASCI compute platform procurements have included the purchase of both a platform and an integrated file system for that platform. The ASCI sites have stated a desire to separate the file system component of the next ASCI platform from the rest of the RFP. This file system independence opens the door to a much broader number of computer vendors and architectures.

File sharing will be key to the success of Sandia's machine delivery in 2003 and all machines that follow it. As file system capacities and investments continue to soar, we can ill afford to continue to rely on separate file systems for each compute engine, each visualization/analysis engine, and each archival storage cache. There is a clear need for a single shared file system ala NFS, but with parallel performance and ASCI-sized scalability.

Finally, future file systems should provide a minimal set of security capabilities. Since a shared network storage device grants access to file system blocks (or objects) via a remote request over some network, it is imperative that adequate measures ensure all unauthorized accesses (write, read, create, delete) are disallowed. This is especially true when the desire is to share the file system across multiple administrative domains such as LLNL, LANL, Sandia, and DOE headquarters. The tension between providing convenient file sharing and proper security makes for quite a challenging problem.

2.2 I/O Characterization of Important Applications

I/O intensive jobs at the DP Laboratories fall into 3 basic categories:

| Application Type | Description |
|-------------------|---|
| Simulation Type-1 | Very little reads. Very write intensive. Typically submitted to the biggest machines (hundreds to thousands of SMP nodes). One new file is created about every 30 minutes; each new file is write shared by multiple nodes. Each new file has extensive write activity resulting in a large size (multiple terabytes -- approaching 25% of the cumulative memory of the cluster). Sequential & random write access for each node, writes typically > 1KBytes. Most codes are MPI based message-passing, but OpenMP style and hybrid style applications exist. Furthermore, some codes use an uncoordinated (embarrassingly parallel) approach. |
| Simulation Type-2 | Very little reads. Very write intensive. Typically submitted to the biggest machines (hundreds to thousands of SMP nodes). File create performance is very important. One new file create for each processor about every 30 minutes, each file may be exclusive to its originating node. Each new file has extensive write activity resulting in a large size (multiple gigabytes – cumulatively approaching 25% of the cumulative memory of the cluster). Sequential & random write access for each file, writes typically > 1KBytes. Most codes are MPI based message-passing, but OpenMP style and hybrid style applications exist. Furthermore, some codes use an uncoordinated (embarrassingly parallel) approach. |
| Post Analysis | Very little writes. Very read intensive. Typically submitted on small clusters (32-128 SMP nodes). Data often represents values (pressure, temperature) within a 3-D mesh. The 3-D mesh is not static, but rather is adaptive (e.g., you need to access pointers to see where in the file a given mesh-point will be in the next time-step). Analysis relies heavily on tricks to speed up disk access for iso-planes or iso-contours (normally random access, difficult to predict – attempts are made to improve locality of data). Most codes are MPI based message-passing, but OpenMP style codes exist. |

Table 1

2.3 Current and Projected Environments at LLNL, LANL, Sandia, and the NSA

Currently each ASCI site consists of many different types of computational platforms, each with its own high performance tightly integrated file system. Additionally there are medium to low performance NFS/DFS/CIFS type services for small file sharing. This environment requires multiple copies of files moved using custom parallel tools. Figure 1 is a generic depiction of this current environment.

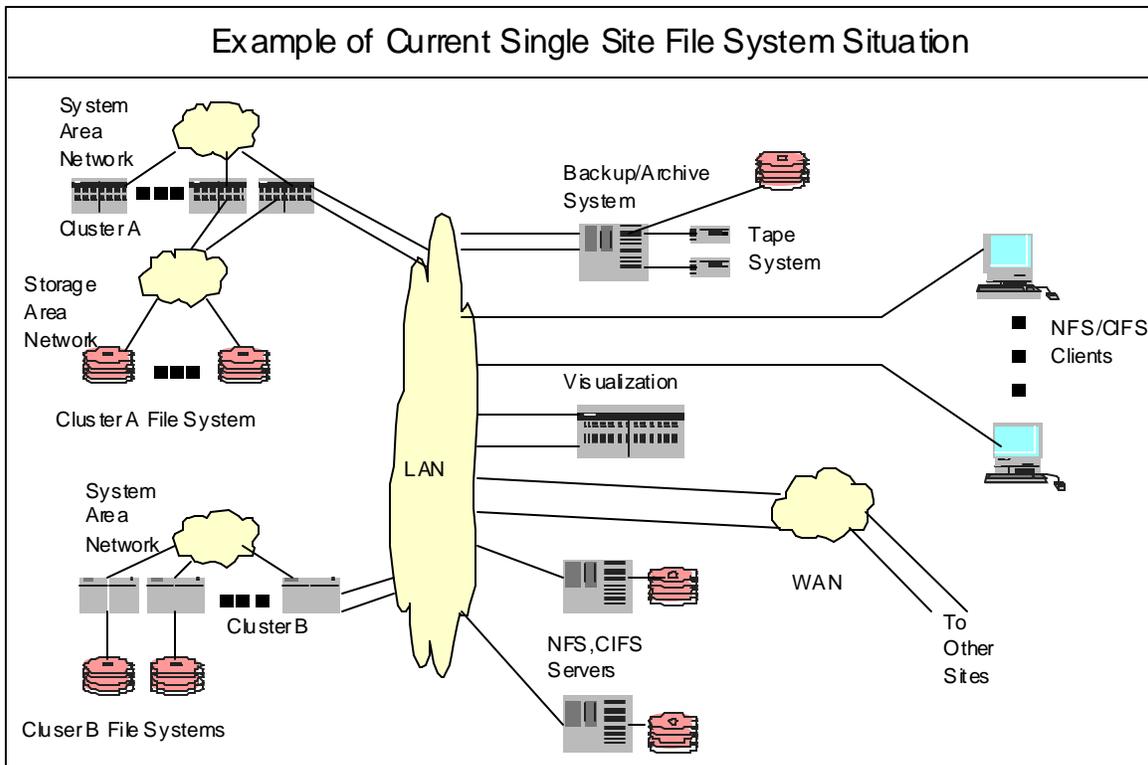


Figure 1: Current Site Environment

Figure 2 below shows an example of how one ASCI site (LLNL) looks currently.

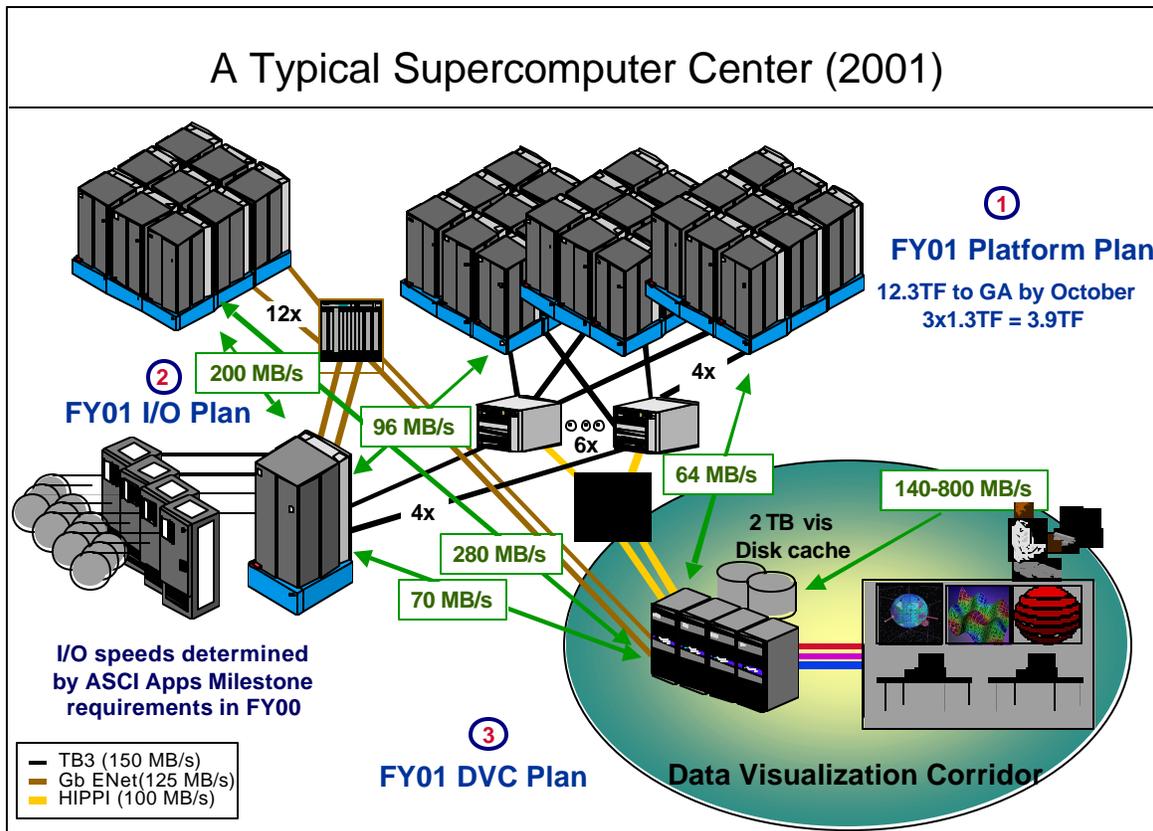


Figure 2: Current LLNL Computation Environment

Figure 3 below represents a possible SGS File System architecture for an ASCII site. Note that the multitudes of file systems are replaced by one global file system. This is only one example of an architecture for a site that utilizes a Storage Area Networked approach to provide the file system connectivity. This is not meant to imply that this is the only solution that would be considered, but it does illustrate the desire to move towards a common file system between platforms and sites.

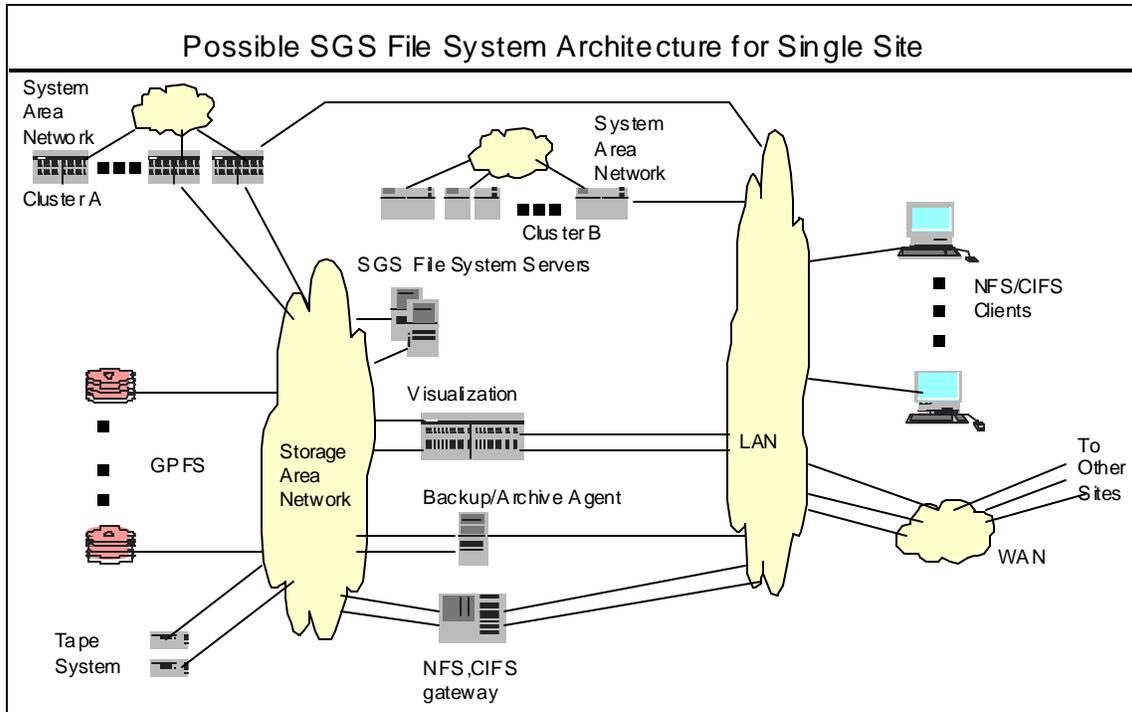


Figure 3: Possible generic architecture for SGS File System at a Site

Figure 4 below is a specific example of an envisioned Site architecture at LLNL in the 2004 time frame. Notice the SAN attached disk providing a file system infrastructure for an SGS File System for the entire site.

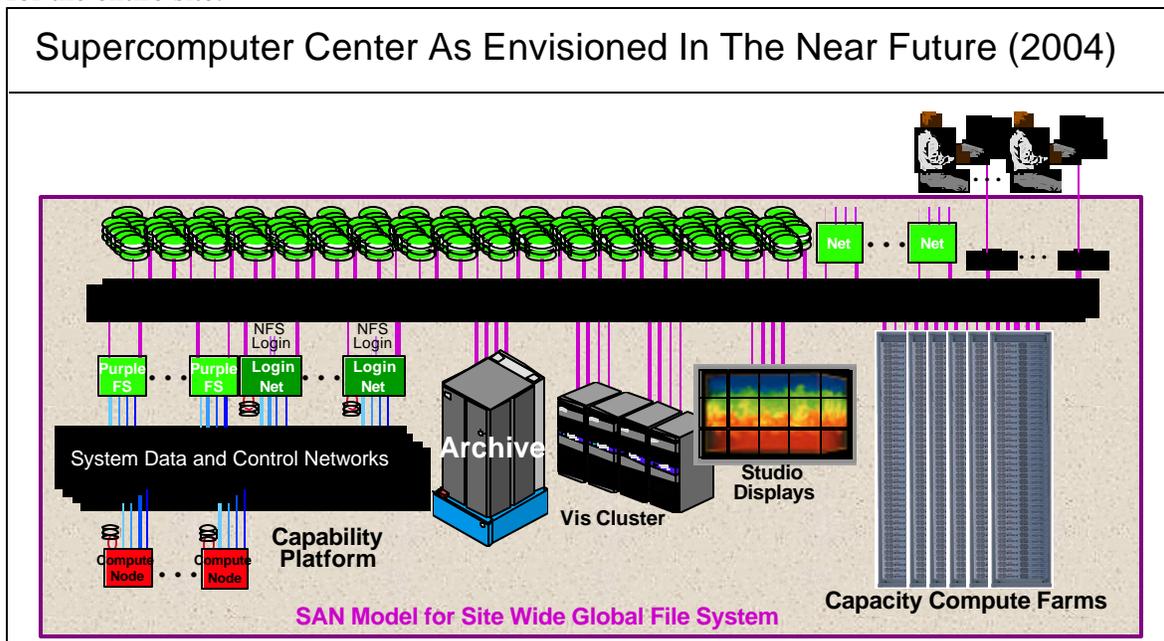


Figure 4: Proposed LLNL Computation Environment for 2004

The above figures are site centric views, but it is important to note that the proposed SGS should logically span multiple sites.

2.4 Summary of Five Technology Categories

The core areas that an ASCII SGS File System should address are: global access, scalable infrastructure, integrated wide area network (WAN) support, scalable administration capabilities, and security. Since this effort is targeted at development or enhancement of a commercial product, ASCII has an interest in how a file system design impacts the range of platforms supported.

Separately and alone, these component technologies are of little use to us. The target file system should *fully integrate the functionality of each of these five categories*.

At present, commercially available file systems present many challenges to the ASCII mission. The following itemizes file systems focus areas into five categories.

- **Global Access**

Ideally we would be able to provide a single uniform name space across all machines which wish to mount the file system. Additionally, the global name space should allow for local site autonomy while still providing global consistency.

A second key aspect of global access is “how pervasive is the technology” (i.e., which platforms are supported). Our heterogeneous environment establishes a need for a widely available solution. One hurdle to *wide-scale heterogeneous support* is the intrinsic complexity of today’s file systems; if the solution is too complex or monolithic, some machines will likely not be supported. A second hurdle to *wide-scale heterogeneous support* involves the kernel intensive nature of file systems; this fact makes it extremely difficult for parties that don’t own the operating systems of clients or peers to keep up with new kernel releases. Third party file systems from vendors that don’t own the operating systems usually suffer from two problems: (1) they, typically, can only support a subset of the popular operating systems; and (2) within the subset of operating systems supported, these third party file system solutions typically lag in support of new operating system releases by one or multiple versions. NFS and CIFS were able to overcome these obstacles primarily because the vendors of client or peering operating systems provide the file system software themselves. These support issues are a very important part of our strategy for choosing file system technologies.

- **Scalable Infrastructure for Clusters and the Enterprise**

We need our compute engines to be able to share one common, high-performance file system with each other and with associated machines, such as visualization engines. Parallel applications on our simulation platforms simply cannot write and read from simple NFS servers, as the performance provided by these systems or other network based file systems is several orders of magnitude lower than we require.

Additionally, these systems do not scale to ASCI-sized machines, with their thousands of nodes. Furthermore, we do not have the desire, or disk capacity, to duplicate our simulation platform file space on our visualization engines. Current solutions require us to invent customized high performance parallel copy software to move data between these systems and to spend considerable money on enormous, overly complicated, external networks and infrastructure between the various platforms in order to enable performance. Moreover, the storage networks are duplicated in order to provide local high-speed performance and capacities for the large data sets.

- **Integrated Infrastructure for WAN**

Many aspects of ASCI depend on the successful deployment of one capability machine, which can be accessed from all three labs. This requires significant WAN user management infrastructure, which simply does not exist today. For example, we require a mechanism that provides a global mapping of the various distributed entities to local entities. That is, a Sandia person may have different usernames, uids, gids, and privileges among machines that they access at Sandia, LANL and LLNL. Alternatively, file system technologies can be designed to operate within a global context where resources and entities are globally unique and each user has a single identity. That is, a Sandia person has a unique identity, group memberships and privileges that are recognized among the different machines and services at Sandia, LANL and LLNL. Currently, DFS is utilized to provide a WAN based file system capability. However, the support for DFS clients and DFS performance, both local and remote, are not sufficient to solve our data sharing needs.

- **Scalable Administration**

System administration for I/O subsystems should remain straightforward and require few new skills from release to release. Storage management should not need to scale linearly with the number of I/O devices, and attached hosts. Furthermore, the daunting learning curve of today's file systems is problematic--we need to be able to use junior system administrators for everyday file system maintenance. The specialized knowledge to manage systems like the current parallel file systems used in our simulation platforms is so involved that we have very few people with enough knowledge to do

most troubleshooting and adjustments. Current parallel file systems simply require too much time and effort to keep up and provide performance only to directly attached and highly integrated hosts. Additionally, scalable management should provide for site-specific actions as well as global operations, much like the DFS system provides today. Finally, file systems could provide useful tools that can be used to dynamically determine the bottlenecks throttling an application's I/O.

- **Security**

We need adequate mechanisms to enforce need-to-know directives. Future file systems should provide a minimal set of security capabilities. Since a shared network storage device grants access to file system blocks (or objects) via a remote request over some network, it is imperative that adequate measures ensure that all unauthorized accesses (write, read, create, delete) are disallowed. This is especially true when the desire is to share the file system across multiple administrative domains such as LLNL, LANL, Sandia, and DOE headquarters. Access Control Lists (ACLs) are a start, but they do not provide the whole solution and there are different, incompatible versions of ACLs. The tension between providing convenient file sharing and proper security makes for quite a challenging problem. Additionally, new scalable global file systems technologies will typically involve more machines being attached to disk storage directly or via a storage area network than ever before. This means that machines with general users will be directly logged into far more systems that have a direct channel to the disk. That increases the security risk by requiring that a far larger number of hosts be installed, maintained, and managed so that their ability to protect data on the media is adequate. There is also a need for advanced security capabilities in future file systems; these capabilities include file-level auditing, data encryption while in transit and content-based authorization. Global file system technologies with support for multiple administrative domains make individual security management that much more complex. While it is technically feasible to extend file sharing using ACLs and groups to multiple domains, there is a greater administrative burden when dealing with multiple sets of data, sharing a subset of the data or if the group role is not aligned across domains. Content-based authorization utilizes attribute metadata associated with the file, the requester's credentials and a site-defined rules-based policy engine to make authorization decisions. The policy engine can also be used to implement Extremely Desired access controls.

3.0 Minimum Requirements (Go/No-go Criteria)

The following items are Mandatory. That is, they must be present in any proposal for that proposal to be considered responsive (and, therefore, eligible for an award).

3.1 POSIX-like Interface [MANDATORY]

We require that any file system solution should provide a POSIX-like interface. That is, it should comply with IEEE/ANSI 1003.x standards in terms of the functional interface bindings between languages and the operating system (e.g., the C language bindings). This is a lesser requirement than full POSIX functional compliance. For example, we do not require that atime/mtime/ctime information be globally accurate for calls to stat() or fstat(). The XPG4-Unix Standard versions of mmap() and munmap() are required, but may be omitted from initial demonstrations. (For info on XPG4-Unix, see “X/Open CAE Specification, Issue 4, 1994.”)

3.2 Integration Compatibility [MANDATORY]

We require that the SGS File System is able to address all five technology targets described in Section 4.0. For any proposal that does not address all five technology targets (which is allowed), the proposed technology must not prohibit the integration with other technology to achieve the remaining technology targets.

3.3 No Single Point Of Failure [MANDATORY]

We require that the SGS File System be designed in such a way that it is always possible to remove a single point of failure by adding the appropriate hardware. That is, the software architecture must be designed to recover and continue operation after the failure of any single hardware component (given a corresponding hardware backup component).

4.0 Desired Performance Features (The 5 Technology Targets)

There are a number of file system technologies that need to be provided and possibly significantly improved if the high performance computing sector of U.S. Government agencies are to carry out their defined mission for the next few years. We have divided these critical desired technologies into five areas: Global Access, Scalable Infrastructures for Clusters and the Enterprise, Integrated Infrastructures for WAN Access, Scalable Management & Operational Facilities, and Security. Additionally, issues like RAS (reliability, availability, and serviceability), standards based API, and archive integration along with development philosophy and documentation are all vital to the success of this endeavor. In the following sections of this document, desired features of the needed file system will be addressed. It is expected that a proposal would attempt to address one, many, or all of these issues.

Separately and alone, these five component technologies (4.1 through 4.5) are of little use to us. The target SGS File System should *fully integrate the functionality of each of these five categories*.

Each item is prioritized into one of three categories: *Extremely Desired* (those items that are most desired); *Highly Desired* (those items that, while very desirable, are not quite as important as Extremely Desired); and *Desired*.

4.1 Global Access

We have a need for file systems that are heterogeneous and global. There are several aspects associated with being a Global File System in our view, including global name space, heterogeneous direct access, and ability to export via a common network file system protocol like NFS or CIFS. The following subsections of this document describe the required and desired features in the Global File System area.

4.1.1 Global Scalable Name Space [EXTREMELY DESIRED]

The name space of the file system should be global. Global means that it is possible to construct a view of the distributed file system hierarchical name space that is identical simultaneously at multiple participating sites and clients. In other words, subject to a site's administrative constraints, it should be possible to provide seamless name space translation to another participating site's name space. Further, at least one fully qualified path name to any file or directory object should be identical from any client anywhere without requiring the user to know the actual location of the data or metadata.

The requirement for a global name space is not intended to imply any requirement for the localization of file metadata or data. The intention is to allow participating Scalable, Global, Secure (SGS) File System sites to offer a name space that is organizationally consistent, though it may be a construction of many geographically remote instances, each under separate administrative control.

The name space shall be usable by any application conforming to the POSIX standard.

4.1.2 Client software [EXTREMELY DESIRED]

There will be portable, open SGS File System client software available for all major platform operating system environments in the government high performance computing environment for direct access of the file system. Access to the file system should be heterogeneous. The heterogeneous environment is made up of Linux (Intel and Alpha), Tru64, AIX, IRIX, Solaris, HP-UX, and Windows/NT/2000 operating systems. Within each of these operating system environments, many levels of the operating system, including older versions and extremely new (even beta) releases of operating systems co-exist. Unlike many classical business environments, where lagging operating system version support for software is not a penalty due to the need for extreme stability, in the high performance computing environment, frequently very young OS levels are required for scalability features and other extreme environmental support features. Given this large set of OS support required, we expect a palatable story on how OS support for the global file system client will be a factor (e.g., consortia of OS vendors, sample client code with public source, full open client protocols). Attractive strategies such as open source, open protocol specifications, and reference implementations (e.g., an open source client for Linux) are extremely desired.

4.1.3 Exportable interfaces and protocols [EXTREMELY DESIRED]

The file system should be exportable via NFS and CIFS to provide coherent access to non-major supercomputing platforms.

Any proposed SGS File System design should be able to participate as one piece of the supercomputing and site infrastructure. Standard supported protocols should be offered to allow data to be easily shared with other compute resources. After all, the data most likely will need to be accessed by nodes other than those which produced the data during such activities as archival, post-processing and visualization.

Additionally, it is probably not the case that support for the new file system will appear in all machines and operating systems at once. Using legacy protocols allows the risk of delays between different operating system implementations to be somewhat mitigated.

For the reasons stated, the ability to export name space and file objects to NFS V4 is **Extremely Desired**. The ability to export name space and file objects to Microsoft file sharing is **Desired**.

Finally, flexibility in transport communications is **Highly Desired**. That is, we highly desire a proposal which is transport agnostic (able to utilize which ever transport the HPC marketplace accepts).

4.1.4 Coexistence with other file systems [EXTREMELY DESIRED]

It is also a requirement that any SGS File System client be capable of co-existence with other file systems on the client system. In other words, the use of the SGS File System on a particular node should not preclude that node's use of other file systems in a native fashion.

4.1.5 Transparent global capabilities [EXTREMELY DESIRED to DESIRED (see table 3)]

Current and future government supercomputing machines have a distance-computing requirement as well. They are intended to be able to cooperate over large geographical distances. For that reason, the file system architecture should not hinder an implementation that must leverage long-haul network links. Two such geographically separated machines should potentially be able to access storage resources on both, through the same API and namespace, as well as with equivalent mechanisms. From the perspective of the compute process, transparency with scalable, parallel movement of data should be limited, again, only by communication links and other hardware. Any implementation should attempt to support this requirement.

Meta-data and data movement protocols should not be implemented in such a way that distance induced latency make full utilization of high bandwidth interconnects over long distances inefficient. This may be an issue for command/response protocols to remote storage. It also may be an issue for buffer size matching of devices to local networks to wide area networks simultaneously attached to local and wide area networks.

Note that these requirements do not necessarily dictate a “globus-like” approach. Any architecture which can provide parallel access to remote files through standard POSIX interfaces is sufficient.

Files available to one processor should be accessible to all processors on the system without special knowledge by the users and independent of the number or ordering of SMPs or of individual processors. As well as read/write access, all the usual file operations such as open, remove, list, etc. need to be provided. All computing nodes within the same computing complex shall be able to access the same file system as well as all remote compute nodes within the “virtual complex,” subject to local site policies. For the time being, we’ll define remote computing nodes as those nodes outside of the traditional computing complex but now an integral part of a “virtual complex” -- a possibly heterogeneous cluster connected together with ultra-high bandwidth networking technology. At some point we would like to have “computing complex” expanded to include heterogeneous machines with ordinary high-end networking connections. Table 3 indicates the priority of needed local/distributed capabilities.

| Remote Access Capabilities | | | | | | |
|----------------------------|-------------|------------------------------------|---------------------|--|------|------|
| Range | Mixed Arch? | Number of Compute nodes (1000s) | Distance (miles) | Importance (Extremely Desired, Highly desired, Desired) | | |
| | | | | 2003 | 2004 | 2005 |
| Machine-wide | homo | 10 or less | < 1 | ExDe | ExDe | ExDe |
| Site-wide | homo | 10 - 100 | 1 - 10 | ExDe | ExDe | ExDe |
| Site-wide | hetero | 10 - 100 | 1 - 10 | ExDe | ExDe | ExDe |
| DP Complex | homo | 100 - 1000 | 3000 | HiDe | HiDe | HiDe |
| DP Complex | hetero | 100 - 1000 | 3000 | HiDe | HiDe | HiDe |

Table 3

4.1.6 Integration in a SAN environment [EXTREMELY DESIRED]

Any SGS File System effort should be able to exploit, in a flexible and extensible manner, the System Area Networks that will be an integral part of high performance computing sites. For instance, employing things like ST, VIA or other OS bypass mechanisms should be considered. However, the choices should not be limiting in nature. The server software and client SGS File System implementations should be able to make use of transports not yet developed and, potentially, only available at a particular site. This may be done through a middle-ware communications layer, pluggable modules or standard, transport independent interfaces, for instance.

4.2 Scalable Infrastructure for Clusters and the Enterprise

It is expected that many of the global file system projects considered in the PathForward activity will not have the extreme scalability in both data and meta-data operations in mind. As the disparity between processor speeds and I/O interfaces increases, the need to scale activities via parallel access to multiple devices becomes more critical. We require that the SGS File System scale in performance with available local and distributed resources, such as network bandwidth, storage resources etc. Quantities that should scale include name space size (above), storage units, clients, servers, bandwidth, number of files, file metadata activity, total file system capacity, and data set size (see Sections below).

4.2.1 Parallel I/O Bandwidth [EXTREMELY DESIRED]

Probably most important for scalability of I/O bandwidth, data should be able to move between multiple media sources and sinks in parallel. Transfers between multiple clients and multiple independent file objects should be able to proceed in a fashion that minimizes mutual interference [that is high throughput for independent, concurrent file accesses]. As well, transfers to a single file from multiple processes should have minimum interference in a similar fashion (i.e., maximum throughput with concurrent access from multiple processes to the same file). Ideally, benchmarks of the aggregate throughput to multiple files by independent processes should demonstrate linear scalability up to the limit imposed by the underlying system software and hardware. Similarly, coordinated access to a single file by multiple processes should be able to demonstrate linear scalability when access is made to non-overlapping allocation units. Furthermore, support for parallel transfers should support environments where there are one or more file system clients per “SMP compute platform” or on cooperating multiple SMPs.

There should be a facility to allow for N to M mapping of parallel files, so that a file stored with parallel geometry N can be read by a process(es) with parallel geometry M. For instance, a file written in column order should be able to be later read in row order. It is understood that *some* performance penalty may be incurred in a case such as this.

At times, it may be necessary to devote every processor in an ASCI machine to one large application. We require the I/O bandwidth to scale as specified in Figure 5. We assume an appropriate underlying infrastructure and configuration of the contemplated SGS File System. Specifically, we require that: (1) the aggregate I/O rate for large sequential **POSIX** accesses should fall along the violet and green lines of Figure 5 for

any given number of clients in a parallel application; (2) the aggregate I/O rate for **large MPI-IO** accesses (accesses of 256K and over) should fall along the violet and green lines of Figure 5 for any given number of clients in a parallel application; (3) the aggregate I/O rate for **random 1K MPI-IO** accesses should be at least 10% of the large MPI-IO rate for the same number of clients. In all instances, the I/O bandwidth designated by the green line, also called the **File System Maximum Sustained Bandwidth**, is to be obtained by the following formula:

$$B_{FS} = N * B_{drives} * E$$

Where

B_{FS} = File System Max Sustained Bandwidth

N = total number of disk drives

B_{drives} = sustained bandwidth of the slowest disk

E = file system efficiency factor (.85)

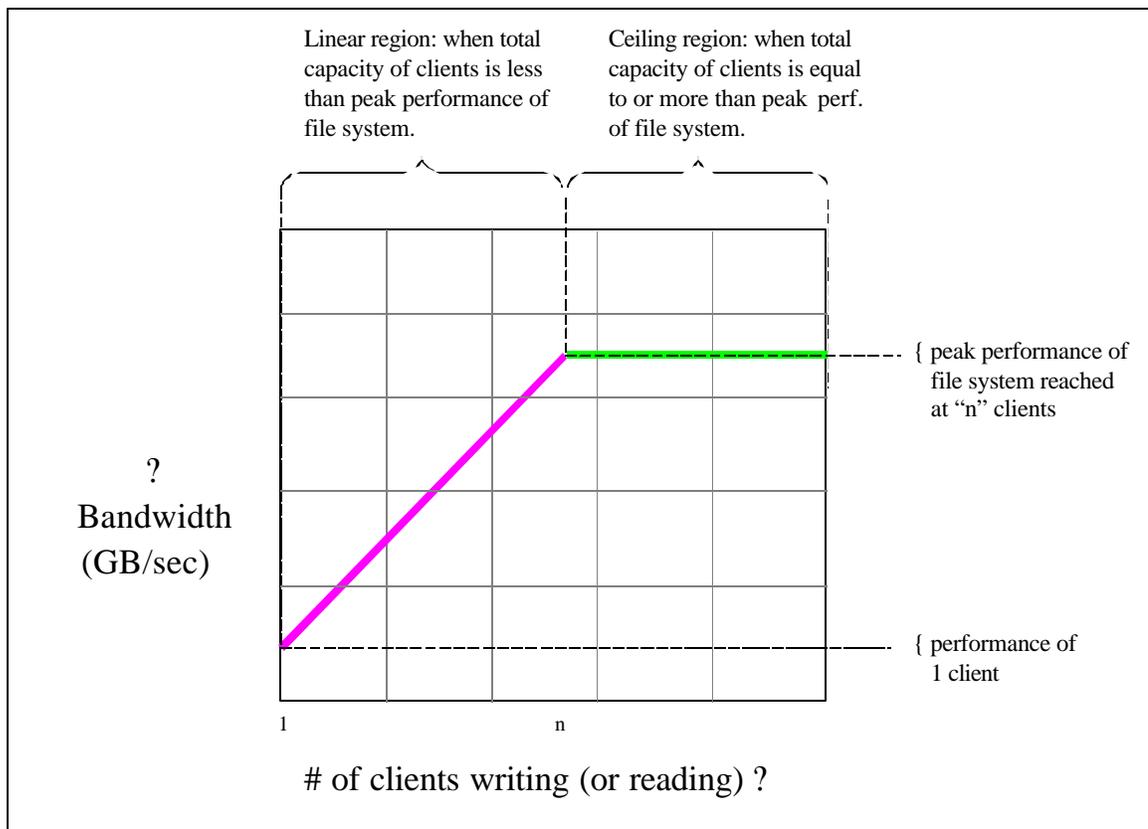


Figure 5

We project a need for I/O rates listed in the bottom row of Table 4. We assume thousands of tasks writing Gigabytes. I/O rates will need to track floating-point performance and aggregate memory. The lower estimates of memory shown in Table 4 assume that an n teraflops machine will require $n^{3/4}$ TB of memory (this relationship has been suggested by recent work on petaflops architectures); the higher sizes assume n Teraflops will require $2/3n$ Terabytes. The lower I/O rate estimates are based on the throughput needed to store one half of the smaller memory in five minutes; the higher I/O rates assume that applications will store one byte for every 500 floating point operations, a common rule of thumb.

| Aggregate Bandwidth Rates for One Parallel Job Simulation & Physics Model Aggregate FS Requirements | | | | |
|---|-------|-------|--------|--------|
| | 1999 | 2002 | 2005 | 2008 |
| Teraflops | 3.9 | 30 | 100 | 400 |
| Memory Size (TB) | 2.6 | 13-20 | 32-67 | 44-167 |
| I/O Rates (GB/s) | 4 – 8 | 20-60 | 50-200 | 80-500 |

Table 4

Two application I/O mode performances are described in the table. It is valid either when one large parallel application is reading or writing to multiple files or when there is only a single file involved. Usually this will be one file per process, but it may also be one file per node. Each file may be 1 GB or more, with each task (or node) serially writing. In short, the number of files involved in any given application's set should not significantly alter the performance curve.

Like parallel performance, single-node performance is very important and should have a minimum of file system overhead. The file system needs to be able to provide 95% of the maximum sustained bandwidth achievable (when appropriately configured to eliminate network contention) by the client OS transferring data through the client's access network to the available storage devices, bypassing the client file cache.

4.2.2 Support for very large file systems [EXTREMELY DESIRED]

As Table 5 indicates, ASCI requires petabyte-range file systems in the near future. It is required that there be no practical limit on individual file size [e.g. 2^{64} byte file sizes]. The name space should be able to manage a million or more directories (possibly containing 10^4 – 10^6 files each).

| File System Capacities | | | | |
|--|--|---|---|---|
| | 1999 | 2002 | 2005 | 2008 |
| Teraflops | 3.9 | 30 | 100 | 400 |
| Memory size (TB) | 2.6 | 13-20 | 32-67 | 44-167 |
| File system size (TB) | 75 | 200 - 600 | 500 -2,000 | 3,000 – 20,000 |
| Number of Client Tasks [see RATIONALE point (1)] | 6144 to 8192 | 8192 to 16384 | 8192 to 32768 | 8192 to 65536 |
| Number of Users | 1,000 | 3,000 | 3,500 | 3,500 |
| Number of Directories [see RATIONALE point (2)] | $5.0 \cdot 10^6$ | $1.5 \cdot 10^7$ | $1.8 \cdot 10^7$ | $1.8 \cdot 10^7$ |
| Number of devices/subsystem [see RATIONALE point (3)] | 5000 (18GB drives) | 3250 – 10000 (72GB drives) | 2084 - 8375 (300GB drives) | 1,350 – 8750 (1200 GB drives) |
| Number of Files [see RATIONALE point (4)] | $7.5 \cdot 10^7$ to $1.0 \cdot 10^9$ | $3.75 \cdot 10^8$ to $4.0 \cdot 10^9$ | $4.5 \cdot 10^8$ to $1.0 \cdot 10^{10}$ | $4.5 \cdot 10^8$ to $1.0 \cdot 10^{10}$ |

Table 5

RATIONALE: (1) These numbers are based on the # of procs; (2) Most files at supercomputer sites are not the large output of parallel simulations – rather, the majority of files are email and other small files found in user’s home directories. At one high performance computing site, today’s statistics show ~1000 dirs per user for home directories and ~8 dirs per user for heavy parallel output. Assume ~5,000 dirs per user will be sufficient for SGS File System; (3) Number of devices should scale with disk density & capacity. Assume disk density increases about doubles every 18 months. All device counts assume the form factor remains 3.5” drives; [Hen94] (4) Today’s statistics show ~21.4 files per directory on home directories and around 44 files per directory for parallel output (max observed is 2,000,000 files per directory). For the SGS File System **low bound**: assume 25 files per directory overall; for the SGS File System **high bound**: assume a few directories will need a 10 million files.

4.2.3 Scalable file creation & Metadata Operations [EXTREMELY DESIRED]

File create performance is another important factor for whole-machine jobs. Concurrent file creation should scale as specified in the performance scaling table.

| File Create Performance –versus- Number of Nodes One parallel program creating multiple files (one per node) into a single directory. N = total number of processors in machine R = File create rate for one processor | | | | |
|--|------------------------------|------------------|------------------------------|-----------------|
| | 1/4 th machine | 1/2th machine | 3/4 th machine | Full machine |
| Aggregate File Create Rate | $.20*N*R$ | $.40*N*R$ | $.60*N*R$ | $.75*N*R$ |

Table 6

Again, we assume an appropriately configured SGS File System and infrastructure as well as proper name space domain decomposition of the benchmark.

Assuming that the machine is a balanced configuration able to achieve the bandwidth numbers specified above, the file system should be able to untar a single source-tree of randomly sized files at the files-per-second rates specified in Table 7.¹

| File Create Rates Untarring random size files from 100 bytes to 16K with an average of 5000 files per directory. (Assume $Q=5000$ and $R=2000$) | | | | |
|---|---------------------|---------------------|---------------------|---------------------|
| Priority | 1999 (files/sec) | 2002 (files/sec) | 2005 (files/sec) | 2008 (files/sec) |
| Highly Desirable | 200 | Q | $3*Q$ | $9*Q$ |
| Extremely Desired | 200 | R | $3*R$ | $9*R$ |

Table 7

Metadata intensive operations should not be significantly slower than local file systems and/or NFS. For example, the following activities should complete in less than one second:

- ‘ls’ on an active directory. That is, calls to `readdir()`, `lstat()` should complete quickly in the presence of directory additions and/or deletions.
- ‘rm’ on very large files (assumes removal from the directory structure, with file space reclamation in the background) That is, calls to `unlink()` and `rmdir()` should complete quickly when performed on directories containing many thousands of entries. We desire removal of files from the directory structure at a rate of at least .75 the creation rate for the given time period.
- ‘mkdir’ on an active file system That is, calls to `mkdir()` should complete quickly in the presence of other, simultaneous, directory additions and/or deletions.

¹ Untarring a file is meant to measure metadata file-create performance.

We recognize the difficulty of accurately maintaining metadata information in a distributed SGS File System. We realize that in a large, highly distributed file system, it may be a very expensive proposition to deliver metadata information to a parallel application that is globally consistent and accurate. For that reason, we can allow the file system to make a best effort attempt to deliver metadata information without guaranteeing complete accuracy or coherency. It would be useful if the file system could guarantee that the inaccuracy be bounded in time. For instance, a tunable parameter could be provided that specifies that the metadata supplied to the application is not more than some number of seconds out of date.

However, there are times when accuracy and coherency are paramount. For instance, an application or utility requesting an access control list may need to be assured that some immediately prior change has been propagated to all hosts with access to the file system. For that reason, the file system should be able to supply a globally consistent and accurate picture of the requested metadata when prompted. This may be through a different call than the default, or with some qualifier. However it is implemented, it should be available.

4.2.4 Archive Driven Performance [EXTREMELY DESIRED]

We envision HSM products that are scalable such as a parallel HSM data mover. While the design and/or interface to such HSM products is beyond the scope of this RFP, we encourage any helpful support along those lines. The SGS File System is expected to provide much higher bandwidth than the archive system. Therefore, the bottleneck for any transfer between the SGS File System and the archive (disk-cache or direct to tape) should be the archive.

4.2.5 Adaptive Prefetching (Desired)

If an implementation utilizes prefetch and write-behind, we desire a method to disable it. For instance, it is normally desirable for a file system's prefetch heuristics to infer access patterns from reading in common 2-dimensional and 3-dimensional array patterns that enable it to optimize media accesses and, significantly, boost performance. This is typically done through making use of the "file view" established within an MPI-IO context. While this is, generally, a highly desired feature of the product, we recognize that possible uses of the file system may cause problems. For these, problematic, cases we would like the application to have the ability to disable prefetching in order to allow tighter control over media accesses and alternative, application implemented, strategies.

4.3 Integrated Infrastructure for WAN Access

4.3.1 WAN Access To Files [HIGHLY DESIRED]

Inter-site cooperation and collaboration are common activities at the national laboratories and other government agencies. We desire that the file system support a common global name space mountable by remote clients. Using tools like ftp, users can access data on

remote resources but this generally results in the creation of a local copy. This becomes a maintenance nightmare. Moreover, it is extremely difficult to maintain the proper authorization controls for multiple copies of the same data or even a single copy of the data if the local site's authorization controls are not able to inter-operate with a remote site's security infrastructure.

In addition, we desire high bandwidth access for large file usage. When a remote site is reading or writing a large file, and multiple connections exist between the two sites, the file system should be able to exploit the multiple paths by providing transparent parallelism.

A common technique for dealing with the large latencies imposed by geographical distances is to replicate the file in different locations. While this technique introduces a host of possible problems including synchronization of changes and reduced utilization of available disk space, it can help solve latency and bandwidth limitations. Our desired requirements in this regard should not be taken to mean a *grid file system*. Rather, we desire a file system which addresses WAN latency and bandwidth limitations without requiring intervention by the user to oversee edit synchronizations.

4.3.2 Global Identities [HIGHLY DESIRED]

We should connect remote super computing sites into one large inter-site collective. Our environment and mission dictates that we cannot exist as a single administrative domain. Each local site has different administrative policies, different people who have privileges to make local modifications, and possibly different local names for the same user or object. Furthermore, using the "inter-site" should be intuitive.

Resources at each super computing site, in addition to operating and being managed as autonomous units, should inter-operate between sites and support remote access within several contexts. These include the need for uniform naming, seamless access with minimal differentiation between local and remote resources, authorization controls to uniquely and properly grant privilege to locally and remotely authenticated entities, wide platform availability and the need for strong and sustained industry support.

4.3.3 WAN Security Integration [HIGHLY DESIRED]

WAN environments place special challenges on security infrastructures. We desire a seamless file system infrastructure that provides the necessary flexibility for individual site administration without precluding inter-site secure operation.

4.4 Scalable Management & Operational Facilities

It is very desirable that the management of a very scalable global file system is scalable as well. In other words, it is important that management overhead of a global file system not increase linearly as the size of the file system grows; this includes meta-data growth, data movement bandwidth growth, and total storage capacity growth. Several studies have concluded that the cost of data ownership is actually more expensive than the initial hardware/software investment. The following subsections address required and desired management related features for the SGS File System.

While file system tools such as fsck, mount, mkfs, etc., are not technically *part* of the file system, these tools are critical components and will heavily affect the success (or failure) of any SGS File System. Rather than specify a reference implementation or design for each critical tool component, the approach we have taken with this RFP is to specify the functionality we desire. In some cases, the file system may have no notion of the desired entity, and POSIX provides no interface to acquire the necessary information (e.g. a batch job script, a parallel application running on multiple nodes, the collection of activities being performed by a given user or group across multiple machines). We recognize that file system tools are in many cases limited by Operating System (OS) support. Certainly new OS standards and/or MIBs are beyond the scope of this RFP. However, we do hope to encourage new modular file system tools that could easily be extended to support the advanced operational capabilities outlined below. Therefore we specify the desired file system tool functionality below with the recognition that new OS interfaces are required, and we will utilize other RFPs / activities to provide the required MIBs and/or interfaces proposed by the Offeror.

4.4.1 Need to minimize human management effort [EXTREMELY DESIRED]

High performance computing environments are large with many diverse storage resources. Operation and management of these resources is a primary concern when integrating any machine into the environment. Personnel considerations are such that it is very important for a file system to take care of itself as much as possible. A system that requires constant attention, reconfiguration or maintenance is going to be problematic at best. As well, in a distributed file system, the number of servers involved may need to change. When such a change is required, the clients, as well as the remaining servers should be able to adapt, minimizing the attention they require. To this end, we desire input into the management tools and philosophies of any SGS File System project.

Currently, file system administration is very tedious and labor intensive. Analyzing behavior by the applications, (evaluating traces) is difficult and should be necessary only on extremely rare occasions. Rather, the file system should make an effort to inform sysadmins when a component is slow or stalled. Further, it should automatically correct the situation if there are no negative consequences, or present the sysadmin with options when a decision with negative implications should be made. Design for “lights out” operation is Extremely Desired.

The file system should support common configuration tasks such as adding or deleting disks, re-balancing data or metadata, or defragmentation online.

Further, there should be diagnostic tools whenever something seems amiss.

4.4.2 Integration with other Management Tools [HIGHLY DESIRED]

While this is not a strict requirement, it would be very useful if control, status and management functions were able to integrate with a storage management suite. It is also desirable that all storage devices, storage fabric, and storage software in the SGS

File System be compliant of standards and emerging standards in this area (e.g., SES – SGSI enclosure services and SNMP).

4.4.3 Dynamic tuning & reconfiguration [EXTREMELY DESIRED]

We anticipate that optimal performance of the SGS File System will require proper settings of a number of site configurable parameters. For example, SGS File System may provide parameters on how much caching is performed, where this caching is performed, and under what conditions the caches are flushed. We require the capability to adjust such parameters “on the fly” without taking the file system down. Tuning a live system is vital; we should not have to unmount and stop the file systems, edit configuration parameters, and then restart the file system.

Furthermore, we frequently need to add or remove devices. The file system should be able to remain up during such activities.

Finally, the file system should be able to dynamically adjust for load balancing once the devices within a file system are increased or decreased.

4.4.4 Diagnostic reporting [EXTREMELY DESIRED]

Usage statistics should be available on: (a) a per-client node basis; and (b) a per-logical disk basis.

Per/client statistics that allow recognizing imbalance are important. For example, if a logical disk becomes unavailable for some time, an imbalance may result when the logical disk comes back online. It should be possible to determine the presence or absence of these imbalances. At the lowest level would be something like nfsstat that shows the number of calls. Better still would be rates, throughput, and outstanding requests. This could be something like IO stat and a performance monitor. This could be used to further fine tune nodes that provide different services. For example, nodes performing a lot of transfers to/from storage may perform better with a different tuning. Also, for dedicated runs and special cases, with the ability to tune on the fly we could optimize the file system for that application.

4.4.5 Support for configuration management [EXTREMELY DESIRED]

The file system should either provide its own configuration management software, or be a “good-citizen,” cooperating with an existing configuration software product. If the product supplies its own, then information about past and present file system versions, applied fixes and/or patches, as well as hardware levels down to the individual disk information should be maintained. If the product does not supply its own configuration management tool, we desire a modular design that could easily support a new configuration management MIB.

4.4.6 Problem determination GUI [DESIRED]

A GUI that displays the major components of the file system as a diagram (including all major components involved in the proper operation of data flow) that dynamically

updates to show status and highlights problem areas should be provided. The diagram would be updated in real time to reflect current usage and/or problems. Use of SNMP traps and MIBS allowing integration with current network management software is desirable.

4.4.7 User statistics reporting [EXTREMELY DESIRED]

Real time usage statistics should be available on: (a) a per-file basis; and (b) a per-parallel job basis (all of the I/O associated with a given job). Additionally, bandwidth and capacity usage statistics should be available on a per user/per project /per site basis such that a charge back accounting system might be able to be implemented as a site decision. Such usage should also be available on a trend analysis basis so that capacity and bandwidth planning can be implemented on a per site basis.

4.4.8 Security management [EXTREMELY DESIRED]

There should be management tools to manage all aspects of security, including ACL editor, logs, etc.

4.4.9 Improved Characterization and Retrieval of Files [DESIRED]

A hierarchical naming convention such as Unix is a weak tool for storing information about files and their relationships. We desire additional characterization of the files such as searchable descriptive information about their contents. Furthermore, we desire additional control over how these files are retrieved; we work with various kinds of workloads including use-once video streams (where if a packet is lost, there is no need to resend it), high performance simulation applications which require sustained data access rates; and lesser priority I/O such as the delivery of email and so forth. [Hen90] [KK90] [McC96].

We are aware of several suitable technologies which involve additions at the VFS layer, adding a file system meta database [Ols93], or adding classes of service (COS). Multiple classes of service need to be supported based on characteristics such as bandwidth (stripe width). COSs TBD.

4.4.10 Full documentation [EXTREMELY DESIRED]

There should be a full set of the usual system, operation, and user documentation.

4.4.11 Fault Tolerance, Reliability, Availability, Serviceability (RAS) [EXTREMELY DESIRED]

Most of the government's super computing program bases its largest compute resources around very large clusters of single or multi-processor-based nodes. The component count in these machines is very, very large. Consequently, failures of individual components are expected, and should be planned for in all designs and operational procedures.

It is unreasonable to expect that this is different with regard to the attached file systems. For that reason, file system architectures should be able to tolerate many kinds of component failure and dynamically adapt. Where architecture cannot automatically

adapt, the design should allow for minimal disruption to the body of currently executing tasks. For instance, a complete reboot of the IO system, or machine, due to a lost message or failed link is unacceptable, when alternatives are available. There should be no disruption of processes that do not use a failed resource.

We also require that any SGS File System effort should include a test or benchmark that will measure fault tolerance and availability.

It is not necessary that the fault tolerance and availability be provided by classical RAID, but the requirement should be met in some affordable manner.

It is expected that the reliability and availability of the SGS File System be directly related to the fault and reliability model options used to configure the SGS File System as well as the underlying hardware infrastructure on which the SGS File System is implemented. By this we mean that various levels of fault tolerant and reliable configurations of an SGS File System are possible, including fully fault tolerant and reliable configurations as well as less than fully fault tolerant and reliable configurations.

| Fully fault tolerant and available system configuration | Less than fully fault tolerant and available system configuration |
|--|--|
| <p>??All data are protected either through parity based techniques, logging, mirroring or other techniques, and</p> <p>??all required communication paths between various parts of the SGS File System are fully redundant with dynamic route adjustment upon failure, and</p> <p>??all required subsystems have redundant fail over infrastructure.</p> <p>??It is not a requirement that the SGS File System take on the entire responsibility of providing this fully fault tolerant and available capability. However, it is required that it be possible to provide the appropriate infrastructure such that the SGS File System can be configured to provide this fully fault tolerant and available system environment.</p> | <p>?? A less than fully fault tolerant and available system configuration should, wherever possible, utilize quorum or other techniques to provide a graceful and consistent degradation of the SGS File System service wherever possible. As well, a coordinated recovery from this degradation after problems have been corrected should be available.</p> <p>?? In a mixed availability configuration, a user I/O job should be able to provide hints as to the fault tolerance requirement for an individual file such that on file create, data blocks are allocated to the configuration with the appropriate levels of fault tolerance.</p> |

Table 8

4.4.12 Integration with Tertiary Storage [EXTREMELY DESIRED]

It is required to provide the standard XDMS DMAPI interface event and library system for any SGS File System project [XDMS].

Integration with Archive means:

- The file system should support an XDMS (X/OPEN's XDMS protocol) DMAPI event and library system.

- The file system should support intra-complex transparent access to the archive.

The SGS File System is not expected to achieve the POSIX and MPI-IO performance metrics itemized in sections 4.2.1-4.2.3 with files managed via DMAPI.

The level of implementation for DMAPI should be such that enough meta-data information is propagated to the HSM agent in order to allow full recovery in the event of a catastrophic disaster. A complete implementation of the DMAPI **dtime** attribute, as well as native support of the DMAPI opaque attributes is attractive.

4.4.13 Standard POSIX and MPHIO [EXTREMELY DESIRED]

The semantics of the file system's application programmer interface (API) needs to be as standard, consistent and orthogonal as is reasonable, with minimal extensions from standards. For example, there seems to be a significant effort on the part of system implementers to add a plethora of options to the open call and control aspects of the API. Then, too, options for one file system are not the same as available on another though their effect may be similar.

The exact API specification should be one of the first deliverables of any SGS File System project we embark on, and should be extensible to be able to take advantage of future concepts if possible. Additionally, standard file system utilities like "ls," "find," redirection, etc., should be supported in some manner that doesn't require re-linking.

The file system in combination with the client software should offer the standard POSIX and MPI-IO APIs.

The common parallel IO API in use within the high performance computing environment today is MPI-IO. We require that the combined SGS File System and client software provide the functionality that enables MPI-IO to take full advantage of the parallel nature of the storage fabric and storage devices in a controlled manner. For good advice about what file systems can do to support high-performance implementations of MPI-IO see [TGL98].

If the file system requires additional information not found in the standard Posix or MPI-IO API, the file system should utilize MPI Hints (as opposed to non-standard API calls).

4.4.14 Special API semantics for increased performance [HIGHLY DESIRED]

If enforcement of POSIX atomicity semantics limits I/O performance, the file system API may offer application programs the option of running in a special mode in which those semantics are not enforced by the file system. In such a mode, as long as distinct processes do not simultaneously write to the same allocation unit in a file, the result of the file operations should be the same as if made standard POSIX semantics. Similarly, either the file system should provide control over the data caches or qualify the mode of operation with a proviso that only data written at the local client is reliable. A method of switching between modes should be provided if control over potentially incoherent

copies of the data is not possible. We are willing to accept that POSIX's atomicity semantics could, and perhaps should, be provided by software outside of the scope of the file system itself, but we desire that this software layer be developed by anyone providing the SGS File System solution. However, the operating system should provide a fully compliant POSIX interface since the SGS File System should be supported as a native file system. Additions to the POSIX interface that significantly enhance performance are allowed (but discouraged, except insofar as they may be hidden from the user by being underneath the MPI-IO interface). The use of "hints" from the user that enhance performance but do not change the semantics of file system operations is allowed.

4.4.15 Time to build a file system [EXTREMELY DESIRED]

Building (formatting to make usable) an ASCII size file system should be as fast as possible. The time to build the file system should be some large percentage of the file system size divided by the maximum sustained I/O bandwidth of the devices..

4.4.16 Backup/Recovery [EXTREMELY DESIRED]

The SGS File System (or portions thereof) should be able to be backed up and recovered in a consistent, expedient manner. The file system should be able to be integrated with third party backup packages including packages that utilize standards based remote control protocols like NDMP. The bandwidth of said backups and recoveries should only be limited by the backup infrastructure. Therefore, the bottleneck for any transfer between the SGS File System and the backup infrastructure should be the backup infrastructure itself.

4.4.17 Snapshot Capability [HIGHLY DESIRED -> EXTREMELY DESIRED]

A software snapshot capability should exist that provides a way to take a "consistent, point in time backup" while minimizing the need for additional disk space. The snapshot feature should take advantage of any hardware assist that a disk subsystem may provide. It is desired that the capability for multiple snapshots to be maintained be limited only by the amount of disk space for snapshots that a site may wish to allocate. Additionally, it is desired that a user have online, read only access to those snapshots where the users had access at the time of the snap. This will allow users to recover from "oops" events (unintentional deletes, changes, etc.) without administrator intervention.

One such snapshot implementation was developed by Network Appliance to allow for minimal overhead such that a copy of the file system metadata (at some sub-tree of the directory hierarchy) is quickly taken pointing to the same data blocks as the active file system's metadata. When the active file system is changed, data blocks that are normally garbage collected remain in use until that copy of the snapshot is eliminated at a later time. Thus, the only overhead is that of the metadata and any changed data blocks.

4.4.18 Flow Control & Quality of I/O Service [HIGHLY DESIRED]

It is desirable that the SGS File System incorporates mechanisms to ensure buffered components are not overrun. Furthermore, since not all jobs have the same priority, it is desirable that the file system provides Quality of I/O Service capabilities. This could be accomplished by a system-admin tool which permits an administrator to prioritize (reserve a guaranteed portion of resources) a given entity. Possible entities include jobs, processes, files, or activities associated with a given user. A modular tool design that easily interfaces to OS and/or batch facilities to designate the entities is desired.

4.4.19 Benchmarks [EXTREMELY DESIRED]

It is expected that the Offeror will develop tests to ensure that performance and RAS requirements are met. Such tests are valuable and give significant additional capability to test and evaluate file systems. Any SGS File System should have a diverse set of scalability benchmarks as part of the deliverables, mutually agreed between the Offeror and the U.S. Government agencies. These scalability benchmarks should cover the above areas. In particular, they should cover parallel access patterns: from one process to many storage devices, from many processes to one file or storage device, and from many processes to many storage devices. These benchmarks should test a variety of diverse concurrent access patterns. Additionally, the benchmarks should be able to test and demonstrate file system metadata performance and scalability.

4.5 Security

We need adequate measures to enforce our need-to-know orders, and adequate logging to assess external and internal attempts to thwart such measures. We require fine-grain access control mechanisms and auditing mechanisms to support a need-to-know sharing and protection model, authorization mechanisms that will be integrated with our current authentication and security infrastructure, and data protection and integrity for information in transit. Existing security-related technologies (e.g., Access Control Lists, shared secret key systems ala Kerberos, public key systems ala Entrust, transport-based data protection and security ala Ipsec) provide some aspects of a fine-grain need-to-know file system. However, a number of issues still remain before an SGS File System can be trusted to provide fine grain inter-site security with a possible "insider threat." Additionally, the security should support the idea of domains of control, so that the individual sites within a multi-site SGS File System complex may have site specific security. The following subsections address security related required and desired features of the needed SGS File System.

4.5.1 Authentication [EXTREMELY DESIRED]

We require standard authentication techniques to minimize inappropriate access or administrative actions. Since authentication mechanisms are likely to change over time, access to authentication functions should be through an industry standard authentication API such as Generic Security Services (GSS).

The authentication scheme employed should not conflict with other schemes that may be present. For example, there should be no reliance on an obsolete version of Kerberos that prohibits the usage of the up to date version of Kerberos for other packages.

We desire a strong authentication scheme that in the best case extends all the way to the storage device. ASCI currently employs Kerberos for authentication of host and user credentials. In addition, the ASCI sites have recently deployed an Entrust Public Key Infrastructure (PKI) that could provide the basis for authentication. Support for an authentication mechanism based upon Kerberos or PKI should be present at initial delivery. If a PKI is used for authentication, the authentication software should interface to the PKI through the Lightweight Directory Access Protocol to a directory containing X.509v3 certificates, so that any compliant PKI (including Entrust) can be used.

4.5.2 Authorization - [EXTREMELY DESIRED]

The file system should provide support for Access Control Lists (ACLs) as the file and directory access protection mechanism and the authorization mechanism for file system management. The ACL mechanism should include support for multiple administrative domains, file and directory inheritance, explicit allow controls for I/O operations and management and the necessary tools for the management of ACLs. The management of ACLs should be consistent for any object in the global name space regardless of client location. The ACL entries should be able to specify permissions for the following-owner of object, group owner of the object, user(s), group(s) and others within the administrative domain, user(s), group(s) and others in external trusted administrative domains and unknown entities. If PKI is the authentication mechanism, then the file system should support an extensible authorization mechanism to enable the use of digitally signed authorization certificates for authorization. Since no standards exist, the ASCI sites will work with the Offeror to develop specifications for such certificates

4.5.3 Content-based Authorization - [HIGHLY DESIRED]

It is highly desirable for the file system to provide support for an extensible authorization mechanism that would enable support for content-based authorization and a site-defined policy engine. This would be an additional authorization check to that specified in 4.5.2. The site-defined policy engine would be invoked by the file server and would be provided with the requested operation, the requester's identity and credentials, the object and metadata, some of which would be site or object specific, associated with the object. The capability to perform a content-based authorization check should be selectable on a per-object basis, possibly via an ACL control.

As a site-defined policy engine would need to store data out of band but related to normal file system attributes, some mechanism that is capable of extending the normal system maintained attributes should be included. Such a system would, necessarily, not normally be able to make use of these attributes, so they would be maintained as opaque, probably keyed, data. In addition to protected, system-level, opaque attributes, it could be useful to maintain a similar mechanism for the user. This would allow the file owner to supply hints and option selection to the site-defined policy mechanism. As with other file object attributes (e.g., ACLs, group ownership), the file system would support an inheritance mechanism to associate user-level metadata or attributes with the newly created objects. The file system would provide a simple opaque object storage, retrieval and deletion capability to support management of the user-level metadata.

4.5.4 Logging and auditing [EXTREMELY DESIRED]

The file system should provide the necessary tools to support vulnerability assessments and be able to reconstruct past events. For example, we require the ability to log and audit granted access, access denial and other security-related events. The logging and auditing capability should be tunable so that only the requested events are logged. It is understood that auditing/logging activities can impact performance. We desire minimal performance impact; a scalable logging design is desired.

4.5.5 Encryption [DESIRED]

Data encryption can provide protection against snooping for data while in transit or at rest. One mechanism for controlling access to data is to encrypt it in a key that is only accessible by the users who are authorized for that data. If encryption is implemented for data privacy, it should meet, at a minimum, the FIPS 140-1 Level 1 requirements and be submitted to NIST for FIPS certification. The FIPS-approved encryption algorithms are Triple-DES and AES (Rijndael). If encryption is implemented for authorization, the file encryption capability should be integrated with a PKI-based authentication mechanism (as described in Section 4.5.1). Each file should be encrypted with a unique symmetric (Triple-DES or AES) key, and the symmetric key should be encrypted using users' public keys, so that only users who are authorized to read a file are able to decrypt the corresponding file encryption key (and hence the file) using their private key. File encryption should take place in the client, so that the data is protected in transmission, storage, and at rest. Also, there should be a way for individual sites to plug in the encryption scheme of their choice.

If encryption is fundamental to the authorization scheme, there should be no means of disabling the use of encryption. If encryption is implemented for data privacy, the decision to encrypt can be mandatory or discretionary. The file system administrator can implement a mandatory decision to encrypt all data that can not be overridden by the user. However, if the administrator chooses to leave the decision to the user, the default mode of transfers shall be without encryption. Product performance is critical and the performance acceptance phase should be run with and without encryption enabled, unless it is required to meet the authorization requirement.

4.5.6 Deciding what can be trusted [EXTREMELY DESIRED]

Shared file systems that are accessed by many clients present special security concerns. In particular, rogue nodes on a common SAN could violate some security schemes. We are willing to consider the kernel and persistent daemons running as root as trusted for their resident *machine*. A *machine* is a self-contained unit in which internal message traffic cannot *easily* be snooped. The file system should support a mechanism for limiting access to only the set of trusted machines and provide an administrative mechanism for immediately disabling access to any trusted machine as needed. Additionally, the file system should support security mechanisms that extend down to the storage device to mitigate risk associated with a trusted machine.

References

- [**Hen90**] D. Hendricks, “A Filesystem for Software Development”, in Proc. USENIX Summer Conference, pp. 333-40, Anaheim, CA, June 1990.
- [**Hen94**] John L. Hennessy and David A. Patterson, “Computer Architecture: A Quantitative Approach”, second ed., San Francisco : Morgan Kaufmann Publishers, 1994.
- [**KK90**] D. Korn and E. Krell, “A New Dimension for the Unix® File System.” Software Practice and Experience 20(S1), pp. 19-34, June 1990.
- [**Kle86**] S. Kleiman, “Vnodes: An Architecture for Multiple File System Types in Sun UNIX”, in Proc. USENIX Summer Conference, pp. 238-24, Atlanta GA, June 1986.
- [**McC96**] Michael McClennen and Stuart Sechrest, “Getting More Information into File Names”, Technical report CSE-TR-279-96, University of Michigan, January, 1996.
- [**Mog86**] J. C. Mogul, “Representing Information about Files.” Technical report STAN-CS-86-1103, Stanford University, March, 1986.
- [**MPI97**] The Message Passing Interface Forum, “MPI-2: Extensions to the Message-Passing Interface”, July, 1997.
- [**Ols93**] M. Olson, “The Design and Implementation of the Inversion File System”, in Proc. USENIX Winter Conference, pp. 1-14, San Diego CA, January 1993.
- [**Pik90**] R. Pike, D. Presotto, K. Thompson, and H. Trickey, “Plan 9 from Bell Labs.” In Proc. UK UUG, 1990.
- [**TGL98**] R. Thakur, W. Gropp, and E. Lusk, “On Implementing MPI-IO Portably and with High Performance.” Proc. Sixth Workshop in Input/Output in Parallel and Distributed Systems, May 1999, <<http://www.mcs.anl.gov/~thankur/papers/mpio-impl.ps.gz>>.
- [**XDMS**] DMAPI Specification, <<http://www.opengroup.org/pubs/catalog/c429.htm>>

Glossary

Some of these terms utilize info from the Free Online Dictionary of Computing <http://www.fodoc.org>, the GNU Project <http://www.gnu.org/philosophy/free-sw.html>, and other DOE webpages.

| | |
|---------------------|--|
| API | Application Programming Interface – The functions and prototypes that a given software layer can program to. |
| Archive | See “Tertiary Storage.” |
| ASCI | Accelerated Strategic Computing Initiative – A U.S. Government funded program which aims to make predictive simulation possible; stimulate the U.S. computer manufacturing industry to create more powerful, high-end supercomputing capability required by these applications; create a computational infrastructure and operating environment that makes these capabilities accessible and usable. |
| Backup | Duplicating a file for safekeeping. Many times, the backup is kept at a remote location to provide catastrophe (fire, earthquake, ...) protection. |
| Blue-Mountain | The initial three machines purchased under the ASCI program were Red (a large Intel Teraflops at Sandia National Labs), Blue-Mountain (a large cluster of SGI Origin systems at Los Alamos National Lab), and Blue-Pacific (a large IBM SP2 at Lawrence Livermore National Lab). |
| Blue-Pacific | The initial three machines purchased under the ASCI program were Red (a large Intel Teraflops at Sandia National Labs), Blue-Mountain (a large cluster of SGI Origin systems at Los Alamos National Lab), and Blue-Pacific (a large IBM SP2 at Lawrence Livermore National Lab). |
| Copy Left | The rule that when redistributing the program, you cannot add restrictions to deny other people certain freedoms (see Free Software). This rule does not conflict with the central freedoms; rather it protects them. |
| CRADA | Cooperative Research and Development Agreements -- A formal agreement involving Lawrence Livermore and an industrial partner(s), in which both parties provide personnel, services, facilities, or equipment for the conduct of specified research and development. A basic purpose of CRADAs is to provide U.S. economic benefit. CRADAs differ from PathForwards in that they are typically more risky and have a further time horizon for product. For additional info click on http://www.llnl.gov/IPandC/Industry/crada.html also PathForward. |
| DAFS | A file system being developed by a consortia lead by Network Appliances and characterized by RDMA data movement. |
| DFS | Distributed File System: A file system which may be mounted by multiple clients distributed over a computer network. An example of a DFS is NFS. Note: In this document, we use DFS generically for any distributed file system in this document – any reference to the TransArc product with the same name will be clearly specified. |
| DisCom ² | Distance and Distributed Computing and Communication: DisCom ² is an ASCI project intended to deliver key computing and communications technologies to efficiently integrate distributed resources with high-end computing resources at a distance. |
| DLM | Distributed Lock Manager: A federated locking mechanism used to synchronize disjoint items or events. |

| | |
|---------------|--|
| DMF | DMF is a project initiated at the three labs to address shareable files between sites. It is intended to be a comprehensive solution for fast, portable, serial and parallel I/O providing data share-ability and application & tool interoperability for scientific data. |
| DOE | The United States Department of Energy (http://www.doe.gov). |
| DOD | The United States Department of Defense (http://www.defenselink.mil). |
| DP | Defense Programs – Activities in support of the nations national security and national defense. |
| DP-10 | Deputy Assistant Secretary for Research, Development, and Simulation – LANL, LLNL, and Sandia are under the umbrella of the Department of Energy. The work proposed in this paper would be funded by the Defense Program (DP) of the DOE. The Defense Program is divided up into several subprograms (e.g., DP-10, DP-20, DP-30, DP-40, and DP-50). This work falls into DP-10 which is “Strategic Computing and Simulation.” |
| DP-14 | Office for Advanced Simulation & Computing – The subsection of DP-10 for which file system R&D falls into. |
| Free Software | “Free software,” is <i>free</i> in the sense of liberty, not price. To understand the concept, you should think of “free speech,” not “free beer.” “Free software” refers to the users’ freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software: (1) The freedom to run the program, for any purpose; (2) The freedom to study how the program works, and adapt it to your needs; (3) The freedom to redistribute copies so you can help your neighbor ; and (4) The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this. Also see the less strict term “open source.” |
| FY | Fiscal Year – The US government uses the fiscal year October 1 through September 30 th . For example, FY01 is October 1, 2000 through September 30, 2001. |
| GFS | Global File System: (Not to be confused with Univ. of Minnesota’s GFS): A file system that provides a single unified name space across multiple (possibly heterogeneous) platforms. |
| GPL | General Public License -- A legal software license arrangement developed by GNU to promote open software. The licenses for most software are designed to prevent users from sharing or changing it. By contrast, the GNU General Public License is intended to guarantee the freedom to share and change free software - to make sure the software is free for all its users. The GPL is designed to make sure that anyone can distribute copies of free software (and charge for this service if they wish); that they receive source code or can get it if they want; that they can change the software or use pieces of it in new free programs; and that they know they can do these things. The GPL forbids anyone to deny others these rights or to ask them to surrender the rights. These restrictions translate to certain responsibilities for those who distribute copies of the software or modify it. |
| HDF5 | A low-level I/O API and file format. Provides a full-featured I/O system enabling data subsetting, portability, etc. (See http://hdf.ncsa.uiuc.edu) |
| HPC | High Performance Computing – The computing market segment concerned with the most demanding computational tasks and very high-end equipment. |

| | |
|-------------|---|
| HSM | Hierarchical Storage Manager -- Software which automatically migrates files between secondary storage and tertiary storage as needed such that the user is unaware of the division between secondary storage and tertiary storage. |
| IP | Intellectual Property -- The ownership of ideas and control over the tangible or virtual representation of those ideas. Use of another person's intellectual property may or may not involve royalty payments or permission, but should always include proper credit to the source. |
| LANL | Los Alamos National Laboratory – A large DOE R&D facility in Los Alamos, New Mexico (http://www.lanl.gov). |
| LLNL | Lawrence Livermore National Laboratory – A large DOE R&D facility in Livermore, California (http://www.llnl.gov). |
| LVM | Logical Volume Manager – Software which has access to a number of individual devices and exports a collective view of the devices. For example, an LVM could make ten 1GB disks appear as one 10GB disk to upper software layers. |
| MIB | Management Information Base – A database of objects that can be monitored by a network management system. Both SNMP and RMON use standardized MIB formats that allows any SNMP and RMON tools to monitor any device defined by a MIB. |
| MPI | Message Passing Interface – One of the two most popular ways to write cooperative parallel applications at the ASCI labs (the other being OpenMP), the Message Passing Interface specifies an interface standard for the message passing paradigm of parallel applications. (See http://www.mpi-forum.org). |
| MPI-IO | The most recent version of the MPI standard (MPI 2.0) includes interfaces for performing parallel I/O. These interfaces are sometimes referred to as MPI-IO. |
| NAP | Network Attached Peripheral: Individual NAS components. |
| NAS | Network Attached Storage: Devices that provide storage services on an internet or intranet. Today, NAS devices typically provide NFS or CIFS services. |
| NASD | Network Attached Secure Disks: A NAP with added security features. |
| NNSA | National Nuclear Security Administration – A branch of the Department of Energy tasked to enhance United States national security through the military application of nuclear energy, to provide the United States Navy with safe, militarily effective nuclear propulsion plants and to ensure the safe and reliable operation of those plants, and to promote international nuclear safety and nonproliferation. |
| NTK | Need to Know -- In the context of security, information that has been deemed to be of a sensitive nature should be secured such that only those people/applications with a “need to know” may access them. For instance, it may be proper for an employee and his physician to have access to medical records, but improper for others. Need to know implies three components: access control which is mechanisms to tag the data as restricted access; authentication which is mechanisms to confirm who is requesting the information, authorization which is mechanisms to enforce authentication and identification. |
| Open Source | Software products provided in such a way that “you can look at the source code.” See also the more strict term “Free Software.” |
| OpenMP | A compiler-based standard for coordinated parallel applications. OpenMP consists of a number of compiler directives and pragmas to perform threads-based parallelization. See also MPI. |

| | |
|-------------|---|
| OS | Operating System |
| PathForward | Funding delivered to industry to accelerate possible commercial solutions for ASCI needs. There are software PathForwards and hardware PathForwards. |
| PKI | Public Key Infrastructure – a system of digital certificates, Certificate Authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction. PKIs are currently evolving and there is no single PKI nor even a single agreed-upon standard for setting up a PKI. However, nearly everyone agrees that reliable PKIs are necessary before electronic commerce can become widespread. A PKI is also called a trust hierarchy. |
| POSIX | Portable Operating Systems Interface – An international standard developed by the IEEE and adopted by the ISO. Provides UNIX users with an international harmonized standard for operating system interfaces. |
| PSE | Problem Solving Environment -- An ASCI Level 3 programmatic effort to Support the rapid development of predictive simulation codes adapted for the efficient use of very-large-scale parallel computer; and to ensure that the power of the application/platform combination can be readily applied by scientists to the challenges of stockpile stewardship and management. See webpage at http://www.llnl.gov/asci/pse/ |
| Purple | The ‘working name’ of the next ASCI machine to be sited at LLNL. |
| Q | A large Compaq cluster being delivered to Los Alamos National Lab. |
| R&D | Research and development |
| RAID | Redundant Array of Independent Disks – striping of a stream of data onto multiple disks usually with some kind of hardware generated parity stripe. |
| RAIT | Redundant Array of Independent Tapes – striping of a stream of data onto multiple tape drives usually with some kind of hardware generated parity stripe. |
| RAS | Reliability, Availability, Serviceability – The three components that define how robust a file system is. Reliability is concerned with how often failures occur. Availability is concerned with how often the file system is unavailable for use. Serviceability is concerned with how easily/quickly problems are resolved. |
| RDMA | Remote Direct Memory Access – Anything that provides a the ability to copy from one user space to a different user space (usually on a different machine node) directly without the usual software fragmentation , software reassembly, and kernel-to-userspace copies. |
| Red | The initial three machines purchased under the ASCI program were Red (a large Intel Teraflops at Sandia National Labs), Blue-Mountain (a large cluster of SGI Origin systems at Los Alamos National Lab), and Blue-Pacific (a large IBM SP2 at Lawrence Livermore National Lab). |
| RedStorm | A large Compaq cluster being collaboratively designed and managed by Sandia National Labs and Celera Genomics Inc. See http://www.energy.gov/HQPress/releases01/janpr/pr01022.htm |
| RFI | Request For Information: A call for information. In contrast to an RFP, an RFI does not require a detailed proposal. That is, it does not generally require a thorough itemized project plan, a thorough itemized funding plan, nor any documentation on anticipated contractual terms and conditions. |

| | |
|------------------|--|
| RFP | Request For Proposal: A formal request to potential suppliers for a proposed solution, including a technical scope and pricing. RFPs are used for both purchasing a sophisticated item and for funding directed R&D. See also RFI. |
| SAN | Storage Area Networks. A dedicated network wherein general host(s) access either NAS or NAPS. <i>Also System Area Network: Network-based I/O architectures that provide many of the capabilities of a computer-bus including processor, memory, and I/O device access. An example of a System Area Network is InfiniBand. (See http://www.infinibandta.org)</i> |
| SCCD | Scientific Computing and Communications Department – The supercomputer center at LLNL. |
| SGS | Scalable , Global, Secure – the defining characteristics of the file system we desire. |
| SDM | Scientific Data Management. SDM is a subproject with the VIEWS project to develop an environment that allows scientists to store, retrieve, search and reduce data within the natural context of their work. This framework integrates scientific data models, commercial databases, mass storage systems, networking and computing infrastructure, and intelligent post-processing to provide assistance in managing the complexity and scale of ASCI data. |
| Shim | A small piece of data inserted in order to achieve a desired memory alignment or other addressing property. For example, the PDP-11 Unix linker, in split I&D (instructions and data) mode, inserts a two-byte shim at location 0 in data space so that no data object will have an address of 0 (and be confused with the C null pointer). |
| SNL | Sandia National Laboratory – A large DOE R&D facility split between two physical sites: Albuquerque New Mexico, and Livermore, California (http://www.sandia.gov). |
| SOW | Statement of Work -- A formal description of an agreed upon task. |
| Tertiary Storage | The lowest strata in the Von Neumann memory hierarchy model characterized by the largest (most cost effective) capacities and the slowest access. Traditionally, tertiary storage has been comprised of tapes and tape robotics, but future systems may employ disk drives as disks become larger and cheaper. |
| TRC | Technical Review Committee – The team of computer scientists who will be reviewing the proposals. |
| Tri-lab | Refers to the three U.S. national security laboratories: Lawrence Livermore National Laboratory, Los Alamos National Laboratory, and Sandia National Laboratories. |
| VIEWS | Visual Interactive Environment for Weapons Simulation: An ASCI project responsible for the development of a software infrastructure which enables the interaction and visualization of ASCI scale datasets. VIEWS software will permit seeing and understanding the results of ASCI codes. |
| WAN | Wide Area Network: Any network technology that spans large geographic distances. ASCI WANs should be able to span Northern California and New Mexico with high-speed links, and possibly other sites with lower speed links. (Contrast with Local Area Network and Metropolitan Area Network.) |
| White | A 12 Tflop IBM SP at LLNL. See http://www.llnl.gov/asci/news/white_news.html |

