GWD-I
Category: Informational
GFS-WG (Grid File Systems Working Group)

Edited by
Pradeep Padala, University of Florida
September 19, 2003

GGF DOCUMENT SUBMISSION CHECKLIST	
	COMPLETED (X) - Date
1. Author name(s), institution(s), and contact information	X - Sept 19, 2003
2. Date (original and, where applicable, latest revision date)	X - Sept 19, 2003
3. Title, table of contents, clearly numbered sections	X - Sept 19, 2003
4. Security Considerations section	X - Sept 19, 2003
5. GGF Copyright statement inserted	X - Sept 19, 2003
6. GGF Intellectual Property statement inserted.	X - Sept 19, 2003
7. Document format - PDF	X - Sept 19, 2003

ppadala@cise.ufl.edu

GWD-I
Category: Informational
GFS-WG (Grid File Systems Working Group)

Edited by
Pradeep Padala, University of Florida
September 19, 2003

A Survey of Grid File Systems

Status of the Memo

This memo provides information to the Grid community regarding requirements of a Gird file system. It does not define any standards or technical recommendations. Distribution is unlimited.

Abstract

This document provides a survey of grid file systems.

Contents

1 Introduction

Grid[1] computing started as sharing of enormous computational resources distributed all over the world. A computational grid, as it is called, is a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to high-end computational capabilities. As grids evolved, management of peta-scale data became cumbersome with ad-hoc data management mechanisms. As noted by Ann Chervenak[2] et al. combination of large dataset size, geographic distribution of users and resources and computationally intensive scientific analyses prompted the development of data grids. The data grid provides mechanisms for managing the distributed data in a seamless way.

A grid middleware provides facilities to use the grid for applications and users. Middleware like Globus[3], Legion[4] and UNICORE[5] provide software infrastructure to tackle various challenges of computational and data grids. A *data middleware*, which usually is part of general purpose grid middleware, provides facilities for data management. Various research communities have developed successful data middleware like Storage Resource Broker(SRB)[6], Grid Data Farm [7] and European Data Grid Middleware.

These middleware have been very successful in providing framework for managing high volumes of data but they are often incompatible. There is a growing need for a standard to describe and organize the data. The Grid File System Working Group (GFS-WG) is developing standards to manage data in a file system style semantics. As a step towards this goal, we are exploring common mechanisms and functionality provided by data middleware. In this paper, we survey existing major data management mechanisms and identify common mechanisms. We also try to provide a sketch of the requirements for a grid file system.

1.1 Grid File Systems

Currently, various middleware provide file system style functionality for accessing data on the grid. We divide the existing frameworks into two categories: Distributed and Parallel file systems and Grid data middleware with or with out file system like interface.

1.1.1 Distributed and Parallel File Systems

Traditionally, data is shared among machines in a network using distributed and parallel file systems. File systems like AFS(Andrew File System)[8], NFS(Network File System)[9] and DFS(Distributed File System)[**?**] provides mechanisms to access remote data through POSIX interfaces. These file systems are usually not scalable over a wide area network. They also do not have the concept of virtual organizations with heterogeneous policies.

1.1.2 Grid Data Middleware

There is a rich set of tools available in this category and are closest to providing file system services. Middleware like Globus data middleware, SRB and Grid Data Farm provide various POSIX I/O primitives for accessing files in a data grid.

In the following sections we survey major research efforts in the above categories. The emphasis is on investigating the file system like interface support for grids.

2 Distributed and Parallel File Systems

3 Grid Data Middleware

3.1 SRB - Storage Resource Broker

```
Project Name:  Storage Resource Broker
Project details (introduction, architecture ...):
```

The SRB is used to implement data grids (data sharing), digital libraries (data publication), and persistent archives (data preservation). We are now working on integration with knowledge generation systems. The goal is to provide infrastructure that makes it possible to automate all interactions with data, including discovery, access, manipulation, publication, sharing, and preservation.

The approach is based upon the organization of the digital entities into a collection, and the management of the collection. The approach is greatly simplified by having the collection own the data, using a logical name space to manage state information about each digital entity, and using a storage repository abstraction for the operations performed upon storage systems, an information repository abstraction for the operations used to manage a collection in a database, and an access abstraction to make it easy to support additional APIs.

Logical name spaces are used for digital entities (data virtualization), users (distinguished names managed by the collection), and resources (logical resource names to support operations on sets of resources). A object storage environment is provided, in which the access to files is based on Unix file operations, rather than disk/block operations.

The SRB is in production use at SDSC, managing 66 TBs of data and 10 million files. Larger systems are run at NASA sites, in the UK data grid, and in the DOE. Other agencies using the software include NSF, NIH, NARA.

Project features:
For each item below, please provide as much detail as possible. For some item, you can just answer with a yes (feature exists), no (feature doesn't exist) or planned (for development)
*) Logical Name Space
Yes - Logical name space independent from physical space
Yes - Hierarchical name space with directories and files (if your project uses a flat logical name space, mention so)
This view can be imposed on the logical name space, either through a Windows interface, or through a Java API that is under development
- structure of logical name space.
A logical collection hierarchy is imposed on the registered digital entities, by associating each digital entity with a sub-collection. Each sub-collection can have a different set of metadata attributes,, including the ability to associate unique metadata attributes with a single file. The logical organization is used to support queries. A query on a sub-collection is supported using the attributes present within that sub-collection and all subordinate sub-collections. The logical name space can be interpreted as a directory structure. It is possible to register a directory hierarchy into the logical name space, replicating the directory path name hierarchy.

Yes - POSIX operations on logical files/directories (provide API details in the API section)
Yes o create, open, close, read, write, delete files
Yes o unlink, seek, sync, stat, fstat, chmod files

```
Yes        o create, open, delete directories
Yes        o read and update contents of directories
Yes    - Soft links between objects/files in logical folders so that a single file
      can be listed in multiple directories
Yes    - Shadow links, physical file name in a remote system from which
the file is registered
In development    - Publication links, logical name in another collection
into which the file is registered
Yes, as directories    - Aggregation of physical files in a single
logical name, providing the capability to access aggregation of files
with a single name.
Yes    - Registration of existing files into logical name space
Yes    - Support for logical collections (association of metadata with logical names)
Yes    - Support for digital entities (URLs, SQL commands, files,
blobs, directories, tables)

*) Uniform storage interface
Yes    - Access to UNIX file systems
Yes    - Access to Distributed and parallel file system objects/files
Yes    - Access to database objects
Yes    - Access to tapes in robots
       - Interfaces to archives
Yes        o   Storage Resource Manager (SRM)
Yes        o   HPSS
Yes        o   DMF
Yes        o   ADSM
No        o   Enstore
Yes       o   UniTree
No        o   JASMine
No        o   Castor
Yes        o   Atlas Data Store
Yes        o   DCache
Yes        o   Interfaces are being added regularly for additional legacy
systems.  At the moment, support for mySQL/BerkeleyDB is being added.

*) Replica Management
Partial    - Distributed/Hierarchical replica catalog
The BIRN project is using the ability of Oracle to replicate metadata
to build a distributed catalog

Yes    - Synchronous creation of replicas with associated metadata creation
Yes    - Asynchronous creation of replicas, register a file as a replica of an existing logical name
Yes    - Fault tolerance, writing to k of n physical resources in a replica
Yes    - Augmenting and Removing replicas
    - Replica consistency
Could be used this way
    o   Write-once or read-only replicas - no consistency mgmt provided
Only for data in containers
    o   Consistent read/write replicas: modifications propagated
        after a write is done and the write lock is released so that a
        read from any replica should return same data.
```

We rely upon the write-lock semantics of the underlying storage systems for
accessing files. We use a dirty flag to mark which replica has been modified.
All operations on that logical name are then directed to that replica. We
provide a synchronization mechanism (either user-initiated or automated) to
propagate changes to the rest of the replicas.

For files in containers, we manage write-locks using standard Posix semantics.
All writes are done as appends to the end of the logical container. Dirty
flags are used to ensure that all modifications are done to the same file.
Synchronization is done across replicas of containers.

Yes o Synchronization of replicas, based upon dirty flag for the modified replica
o Other consistency protocols/mechanisms
Yes - Load balancing among replicas
partial - Replication of fragments of a file/object
We support fragmentation of a file across multiple tapes. Replication is still done on the entire file

*) Data Access/Transfer
Yes - POSIX semantics to logical files
- Parallel I/O support
Yes o Parallel I/O on get/put commands
Yes o Parallel I/O on partial reads/writes
Yes o Parallel I/O on third party transfers
Yes o Server initiated parallel I/O
Yes o Client initiated parallel I/O
- Reliable file transfer
Yes o Status and monitoring information for data transfers
Yes o Automatic restart if failed
Client level o Restart after interruption from application
Yes o Storage completion at the end of single write
In progress - Striping across disks/nodes/sites
- Network tuning
Yes o Static tuning of network/data buffers
We use a default buffer size of 800 MBs

No o Dynamic tuning of network/data buffers
Instead, we use parallel I/O streams to fill the network pipe. This gives better performance

In Progress - GridFTP support
Partial - User selectable transfer mechanisms.
We support multiple APIs, each of which has their own transfer mechanism (http, Java, GridFTP)

Yes - Custom control protocols
We optimized interactions with remote storage systems to minimize the number of messages, support bulk

*) Latency Management
Yes - Streaming

```
Yes    - Disk caching
Yes    - Pre-fetching of buffers
Yes    - Remote I/O proxies for aggregating I/O commands, remote data
      filtering, metadata extraction
Yes        o   Remote proxies through DataCutter
Expected when the next version of GridFTP comes out        o   Remote proxies through GridFTP
Yes    - Staging
Yes    - Replication as a method of latency management
Yes    - Bulk metadata operations
yes    - Bulk file registration
Yes    - Bulk data load
Yes    - Bulk data unload

*) Metadata Management
Yes    - Methods for creating, updating and publishing metadata
    - File level metadata
Yes        o   size of the file
Yes        o   creation/modification/access time
Yes        o   creator
We have roles for owner, curator of a collection, annotation permission


Yes        o   replica number
Yes        o   write locks on containers
Yes        o   dirty flags for changed replicas
 Yes       o   Audit trails on data usage
Yes        o   version number
Yes        o   retention period
We provide two retention periods, one for data on a disk cache, and one for data in an archive

        o   other
    - Storage metadata
Yes        o   storage type
Yes        o   size of the storage
No        o   duration available
Yes        o   permanency of storage
        o   ...
    - Access control metadata
Yes        o   access control lists by user, group per file
Yes        o   hierarchical access control mechanisms per directory
    - Descriptive and Provenance metadata
Yes        o   metadata describing derived data
Yes        o   provenance info
We support Dublin core attributes, and plan to support METS schema

Yes        o   user-defined metadata per file
    - Metadata catalog architecture
Planned        o   Hierarchical metadata catalog
The peer-to-peer federation of catalogs can be used to implement a master catalog into which local catal

Yes        o   Distributed metadata catalog
```

```
Implemented by using the capabilities of databases such as Oracle

In progress        o   Peer-to-peer federation of metadata catalogs
This will support replication of metadata into an independent database


Yes    - Metadata based query support (finding data based on the attributes)
Yes    - Metadata consistency controls on update
*) APIs
Yes    - File API
Yes    - Object level API
Yes    - Web service API - WSDL
Yes    - Language dependent APIs - C, C++, Java, Python ...
Yes    - Library API - Open Archives Initiative
Yes    - Web browser API - http
*) Authentication
    - Collection or community-owned data
Yes      o  GSI authentication
Yes      o  PKI authentication
Yes      o  challenge-response authentication
Yes      o  ticket-based authentication (date range, or number of accesses)
Yes      o  ACLs on data based on logical name, implying access restrictions follow file)
Yes      o  ACLs on metadata by table or record
Yes      o  Single sign-on, distinguished names for users that are site independent
    - Files owned by individuals
Partial      o  GSI authentication
Through shadow links, a collection can access data owned by an individual.
Permission must be given to the collection for the read access.  The user
retains all of their original control


Partial      o  PKI authentication
As above

Partial      o  ticket-based authentication (date range, or number of accesses)
As above.  A typical use is to provide a URL in which the access ticket is embedded.  The user then cli

Yes      o  ACLs based on logical name, implying access restrictions follow file)
Partial      o  Single sign-on, distinguished names for users that are site independent
As above.

*) Optimization or Performance Improvements
Partial    - Automatic optimal replica selection
Since access latencies increase dramatically from file systems, to databases,
to archives, we pick a replica based upon the type of system.  Any disk is
preferred over any database.  Any database is preferred over any archive.

Yes    - Bulk data transfer operations
Partial   - Optimized for management of large files (size greater
            than bandwidth-delay product)
```

We support arbitrary sized files, but do not implement markers in the data
transmission path to support partial retransmission

Yes - Optimized for management of small files (size less than bandwidth-delay product)
Yes - Pre-spawned service instances
- Other
*) Robustness, Fault Tolerance and Error Handling
Yes - Automatic fail over to alternate replicas when the first copy is
unavailable
Yes - Automatic re-trials to access temporarily un-available data/meta-data
Yes - Exponential backoff between re-trials
Partial - Data transfer resumption after system restart
Done at client level

No - Configurable time-outs
Yes - File checksum
- Other
Implementation notes:
Yes - client server architecture
Yes - RPC based service invocation to minimize number of control messages
- Supported architectures
Yes o 32-bit Linux
Yes o 64-bit Linux
yes o Sun-OS
Yes o AIX
Yes o IRIX
Yes o HP True-64
Yes o Windows NT
Yes o Mac OSX

- What kind of application does the system assume?

The system is used to implement digital libraries for publishing data, data
grids for sharing data, and persistent archives for long-term preservation.
The applications range from management of PB data collections, to repliation of
TB-sized collections across multiple sites, to web-based access to collections,
to support of web services for data subsetting, metadata extraction, image
cutouts.

- What kind of software or what kind of algorithm is used to manage
(filesystem) metadata?

The system uses either commercial database technology (DB2, Oracle, Sybase,
SQLServer, Informix) or public domain databases (Postgres, mySQL) to manage the
metadata. An information repository abstraction is used to make it possible to
manage a catalog in the chosen database technology.l

Why is it chosen?

Relational database technology was chosen to make it possible to index the tables, optimize the performance, manage millions to hundreds of millions of digital entities, and provide a wide variety of access mechanisms (WSDL, C library calls, Java, etc.)

- How is the lock mechanism implemented?

Locking is done by setting an attribute in the database. Since all references to a digital entity are based on the retrieval of metadata about the logical entity, the lock status can be checked on the start of any operation.

- How large environment does the system applied?

The system is used to support a collection of more than 100 million digital entities. Another implementation (NASA) uses the system to manage interactions with a PB disk cache. At SDSC the system manages 75 TBs of data comprising over 11 million files. A high-energy physics data grid based on the SRB has 17 sites distributed across 7 countries.

- How does the system perform?

The performance of the system is tied to the capabilities of the underlying database. Using a highly tuned version of Oracle, bulk registration rates of 400 files/sec have been measured, bulk load rates of 300 files per second, data transfer rates of 250 MB/sec (limited by the performance of the remote file system for receiving the data),. The system can saturate either the source, network, or sink when sending data using parallel I/O streams. In wide area networks, additional tuning is being done on the control protocol to further improve the ability to list massive collections.

- Further improvement?

The current upgrades are peer-to-peer federation, including publication links for controlling metadata consistency within federations, dynamic specification of consistency constraints, and integration with the emerging OGSA technology for access.

3.2 Gfarm - Grid Data Farm

Project Name: Grid Datafarm
Project details (introduction, architecture ...):

The Grid Datafarm architecture is designed for global petascale data-intensive computing. It provides a global parallel file system with online petascale storage, scalable disk I/O bandwidth, and scalable parallel processing performance. The global parallel file system consists of local disks of cluster nodes in a grid of clusters. Fault tolerance and load balancing are automatically managed by file replicas.

```
Project features:

For each item below, please provide as much detail as possible. For
some item, you can just answer with a yes (feature exists), no (feature
doesn't exist) or planned (for development)

*) Logical Name Space
     - Logical name space independent from physical space

Yes

     - Hierarchical name space with directories and files (if your project
       uses a flat logical name space, mention so)

Yes

     - Structure of logical name space (Is it a graph? what does the leaf
       represent?)

Tree.  The leaf represents a replica set of files.

     - POSIX operations on logical files/directories (provide API details
       in the API section)
         o create, open, close, read, write, delete files
         o unlink, seek, sync, stat, fstat, chmod files
         o create, open, delete directories
         o read and update contents of directories

All operations except sync, chmod are supported in the current version
1.0 beta 4, although chmod will be supported in the next release.
Also, several operations are supported such as fileno, feof, ferror,
clearerr, fflush, getc, ungetc, putc, puts, getline, putline, chdir,
utimes, access, closedir, and execve.  Moerover, several oerations for
creating and deliting a file replica are supported.

     - Soft links between objects/files in logical folders so that a
       single file can be listed in multiple directories

Planned.

     - Shadow links, physical file name in a remote system from which
       the file is registered


Planned.

     - Publication links, logical name in another collection into
       which the file is registered

Planned.
```

```
     - Aggregation of physical files in a single logical name, providing
       the capability to access aggregation of files with a single name.

Yes.  We call the aggregation of physical files a Gfarm file, which
can be used not only by the access of aggregation of files but also by
file-affinigy scheduling.  Each physical file in a single Gfarm file
can be accessed in local or index file view that is an original idea
of Grid Datafarm.

     - Registration of existing files into logical name space

Yes.

     - Support for logical collections (association of metadata with
       logical names)

Yes.  Hierarchical logical names associate file system metadata
including mode, user, group, access/modification/change times, size,
checksum type, checksum.

     - Support for digital entities (URLs, SQL commands, files, blobs,
       directories, tables)

Planned.

*) Uniform storage interface

Yes. - Access to UNIX file systems
Yes. - Access to Distributed and parallel file system objects/files
No.  - Access to database objects
     - Access to tapes in robots
Planned.
     - Interfaces to archives
No       o   Storage Resource Manager (SRM)
No       o   HPSS
No       o   DMF
No       o   ADSM
No       o   Enstore
No       o   UniTree
No       o   JASMine
No       o   Castor
No       o   Atlas Data Store
No       o   DCache
No       o   Other

*) Replica Management
     - Distributed/Hierarchical replica catalog

No.  Currently it is implemented by a single openldap server.

     - Synchronous creation of replicas with associated metadata creation
```

```
Yes.

     - Asynchronous creation of replicas, register a file as a replica
       of an existing logical name

No.  We do not allow operations that cause the inconsistency between
file system metadata and physical file.

     - Fault tolerance, writing to k of n physical resources in a replica

Yes.

     - Augmenting and Removing replicas

Yes.

     - Replica consistency

We do not provide enough replica consistency yet, although it is
planned.  In the current implementation, unupdated replicas are
deleted.  In the future release, we will support it by versioning.

     - Load balancing among replicas

Yes.  We select one of replicas based on the runtime load average.

     - Replication of fragments of a file/object

Yes.

*) Data Access/Transfer
     - POSIX semantics to logical files
Yes.  We provide a file system.
     - Parallel I/O support
         o   Parallel I/O on get/put commands
Get/put are not file system operations.  It is for FTP.  This is out
of scope.
         o   Parallel I/O on partial reads/writes
Yes.  We have original file views; local and index, for parallel reads
and writes.
         o   Parallel I/O on third party transfers
Yes.  When replicating a file that is aggregation of files, each file
is directly transferred in parallel.
         o   Server initiated parallel I/O
???  What does it mean?  Is this for FTP?
         o   Client initiated parallel I/O
???  What does it mean?  Is this for FTP?
     - Reliable file transfer
Planned  o   Status and monitoring information for data transfers
Planned  o   Automatic restart if failed
```

```
Planned  o   Restart after interruption from application
No       o   Storage completion at the end of single write
     - Striping across disks/nodes/sites
Yes, one file is stored across disks/nodes/sites but it is not really
striping.
     - Network tuning
Yes.     o   Static tuning of network/data buffers
Planned  o   Dynamic tuning of network/data buffers
     - GridFTP support
Planned
     - User selectable transfer mechanisms.
Yes
     - Custom control protocols
This depends on lower layer for communication.  Grid Datafarm does not
care about it.  On the other hand, we support three
authentication/communication method; sharedsecret/raw, gsi/raw, and
gsi/gsi.

*) Latency Management
     - Streaming
Yes.  It can be specified by a user configuration file or by a
node-wide configuration file.
     - Disk caching
Yes.
     - Pre-fetching of buffers
Planned.
     - Remote I/O proxies for aggregating I/O commands, remote data
       filtering, metadata extraction
         o   Remote proxies through DataCutter
         o   Remote proxies through GridFTP
     - Staging
Planned
     - Replication as a method of latency management
can be
     - Bulk metadata operations
Planned
     - Bulk file registration
Planned
     - Bulk data load
Planned
     - Bulk data unload
Planned

*) Metadata Management
     - Methods for creating, updating and publishing metadata
Only via file operations.  This ensures the consistency.
     - Does the system utilize multiple metadata servers, or multiple
       metadata databases?  If yes, how is the consistency maintained?
Planned.
     - File level metadata
Yes      o   size of the file
```

```
Yes      o   creation/modification/access time
Yes      o   creator
         o   replica number
No.  But it can be counted
         o   write locks on containers
Planned
         o   dirty flags for changed replicas
Planned
         o   Audit trails on data usage
Planned
         o   version number
Planned
         o   retention
What does it mean?
         o   other
mode including types and access permissions, group, checksum and
checksum type, number of aggregated files

no   - Storage metadata
         o   storage type
         o   size of the storage
         o   duration available
         o   permanency of storage
         o   ...
     - Access control metadata
yes      o   access control lists by user, group per file
yes      o   hierarchical access control mechanisms per directory
     - Descriptive and Provenance metadata
planned  o   metadata describing derived data
planned  o   provenance info
planned  o   user-defined metadata per file
     - Metadata catalog architecture
need research and evaluation
         o   Hierarchical metadata catalog
         o   Distributed metadata catalog
         o   Peer-to-peer federation of metadata catalogs
     - Metadata based query support (finding data based on the attributes)
planned
     - Metadata consistency controls on update
planned

*) APIs
     - File API
yes
     - Object level API
no
     - Web service API - WSDL
planned
     - Language dependent APIs - C, C++, Java, Python ...
C (Fortran and C++ can use it)
     - Library API - Open Archives Initiative
```

```
planned
     - Web browser API - http
planned

*) Authentication
     - Collection or community-owned data
yes    o  GSI authentication
yes    o  PKI authentication
GSI is PKI
no     o  challenge-response authentication
no     o  ticket-based authentication (date range, or number of accesses)
       o  ACLs on data based on logical name, implying access
          restrictions follow file)
unix file system standard access permission control in the logical hierarchy
       o  ACLs on metadata by table or record
unix file system standard access permission control in the logical hierarchy
yes    o  Single sign-on, distinguished names for users that are site
          independent

     - Files owned by individuals
yes    o  GSI authentication
yes    o  PKI authentication
no     o  ticket-based authentication (date range, or number of accesses)
       o  ACLs based on logical name, implying access restrictions follow file)
unix file system standard access permission control in the logical hierarchy
yes    o  Single sign-on, distinguished names for users that are site independent

*) Optimization or Performance Improvements
     - Automatic optimal replica selection
yes.  it depends on the runtime load average of servers, or local
files are preferred.
     - Bulk data transfer operations
yes.  The case of transfering files.
     - Optimized for management of large files (size greater than
       bandwidth-delay product)
Yes, but manually.
     - Optimized for management of small files (size less than
       bandwidth-delay product)
Yes, but manually.
     - Pre-spawned service instances
io daemon has beed running.
     - Other

*) Robustness, Fault Tolerance and Error Handling
yes  - Automatic fail over to alternate replicas when the first copy is
       unavailable
     - Automatic re-trials to access temporarily un-available data/meta-data
planned
     - Exponential backoff between re-trials
planned
     - Data transfer resumption after system restart
```

```
planned
     - Configurable time-outs
planned
     - File checksum
yes, md5 checksum is included in the file system metadata.
     - Other
When physical files are unavailable in some reasons, the correnponding
metadata is deleted.
Implementation notes:
     - client server architecture
io daemons run on every file system node.  currently one file system
metadata server.  Files in a virtual file system, or a global parallel
file system (Gfarm file system) can be accessed by commands,
explorer-like GUI, POSIX API, and Gfarm filesystem API
     - RPC based service invocation to minimize number of control messages
yes
     - Supported architectures
yes    o  32-bit Linux
plannedo  64-bit Linux
yes    o  Sun-OS
client o  AIX
client o  IRIX
client o  HP True-64
plannedo  Windows NT
plannedo  Mac OSX
    - What kind of application does the system assume?
Mostly data-intensive application that has data access locality.
However, every application can run anyway.
    - What kind of software or what kind of algorithm is used to manage
      (filesystem) metadata?  Why is it chosen?
currently, openldap server.  it will be replaced due to the
performance reason.
    - How is the lock mechanism implemented?
planned
    - How large environment does the system applied?
We usually have a Trans-Pacific testbed with hundreds of nodes.
    - How does the system perform?
Using 80-node AIST Gfarm cluster, we achieved 7.7 GB/s and 9.8 GB/s
for writing and reading a 1.7-TB file, respectively.  File replication
of a 640-GB file performs 1.7 GB/s (= 14.5 Gbps) using 32 streams.
Using a Trans-Pacific testbed, we achieved 741 Mbps out of 893 Mbps
for file raplication of a 8-GB file.
    - Further improvement?
We need to research more scalable metadata architecture.

Other comments:
```

4 Summary of findings

5 Security Considerations

Not applicable to this document

Author Information

Pradeep Padala, University of Florida, Gainesville, Florida 32611–6120, ppadala@cise.ufl.edu

Intellectual Property Statement

Full Copyright Notice

References

[1] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[2] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, 1999.

[3] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, 1997.

[4] Andrew S. Grimshaw, William A. Wulf, and the Legion team. The legion vision of a worldwide virtual computer. *Communications of the ACM*, 40(1):39–45, January 1997.

[5] V. Huber. UNICORE: A Grid computing environment for distributed and parallel computing. *Lecture Notes in Computer Science*, 2127:258–266, 2001.

[6] Chaitanya Baru, Reagan Moore, Arcot Rajasekar, and Michael Wan. The sdsc storage resource broker. In *Proceedings of IBM Centers for Advanced Studies Conference*. IBM, 1998.

[7] Osamu Tatebe, Youhei Morita, Satoshi Matsuoka, Noriyuki Soda, and Satoshi Sekiguchi. Grid datafarm architecture for petascale data intensive computing. In Henri E. Bal, Klaus-Peter Löhr, and Alexander Reinefeld, editors, *Proceedings of the Second IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid2002)*, pages 102–110, Berlin, Germany, 2002. IEEE, IEEE Computer Society.

[8] John H. Howard. An overview of the andrew file system. In *Proceedings of the USENIX Winter Conference*, pages 23–26, Berkeley, CA, USA, January 1988. USENIX Association.

[9] B. Callaghan, B. Pawlowski, and P. Staubach. RFC 1813: NFS version 3 protocol specification, June 1995.