Leo Luan, IBM Almaden Research Center
Ted Anderson, IBM Almaden Research Center

June 2003

Grid Namespace for Files

Status of This Memo

This memo provides information to the Grid community regarding the value of a Grid namespace for files. It does not define any standards or technical recommendations. Distribution is unlimited.

Abstract

We propose a directory service, called Virtual Filesystem Directory Service (VFDS), which enables the realization of a Grid File System (GFS) with a uniform, global, hierarchical namespace. This namespace, combined with existing Grid replication and location lookup mechanisms, can support independence of position for user or application and transparency of data location in a scalable and secure fashion. The VFDS pathname model encourages addressing file data at the directory subtree or file system granularity. This enables a transparency of protocol that allows inclusion of data from a variety of distributed file systems. With appropriate protocol conversion mechanisms a federation of file systems is possible. Further, the aggregation of files into subtrees provides natural collections that can improve the scalability and manageability of replication and management mechanisms. A uniform namespace with global scope and hierarchical ownership provides a way to share file data within and across organizations without compromising security or autonomy.

Contents

Leo Luan, IBM Almaden Research Center
Ted Anderson, IBM Almaden Research Center

1. Introduction

One important goal of Grid computing is to enable distributed resource sharing while remaining consistent with requirements of strong security and flexible access control. Distribution brings with it more than large distances, increased latency and reduced bandwidth. It also leads to larger populations of machines, services, users and groups, which naturally result in more diversity of administrative domains, host platforms, network protocols, and other challenges. Dealing with this diversity is a large job that can be mitigated by embracing standards whenever possible. Security, including authentication, authorization, as well as privacy and integrity, is especially crucial in large-scale systems where the informal rules of small communities break down. It assures users that their data is protected and provides participants control over their resources. As with any system that strives for relevance and utility, ease of use and performance are important properties as well.

File data, an important resource for almost any computation, is strongly affected by distributed environments and therefore has been an area of significant focus for Grid development and research, in particular, the Globus project. These efforts have targeted security, data transport, and replication, among other areas. The primary vehicle for delivering improvements has been an enhanced version of the FTP protocol with server and client libraries collectively called GridFTP. FTP, being venerable, popular and relatively simple, was a good candidate for enhancement, preferred over working with existing distributed file systems. Further, the approach of designing a new protocol was prudently avoided. The result is a useful tool for accessing and transporting files within the Grid environment. Another important development of the Globus project is Replica Location Service (RLS). RLS aims at providing a scalable solution to maintain the location information about physical replicas of file data. Using GridFTP and RLS, replica management tools can be developed to maintain file replicas at chosen locations for high-performance file access.

However, Grid technologies to date have not addressed some fundamental requirements needed to facilitate scalable file sharing. In particular, we believe a directory service is needed to provide files with a uniform name across the Grid environment. Such a directory service can not only leverage GridFTP and Globus RLS for secure and high-performance access to files but also allow the inclusion of important data available via standard network file system protocols, such as NFS and CIFS (commonly referred to as NAS protocols). This would allow Grid applications to access NAS file data through a unified namespace thus bringing conventional distributed file system into the Grid.

The directory service we propose here, named Virtual Filesystem Directory Service (VFDS), enables the realization of a Grid File System (GFS) with a uniform, global, hierarchical namespace. This namespace, combined with existing replication and location independence mechanisms, can provide transparency of position and location in a scalable and secure fashion. Uniformity, or transparency of position, means that the file system looks the same to all applications, users, and clients even if they move around. This requires that the mappings from pathnames to the files they represent are global and do not depend upon local configuration.[1] Location transparency means that the data can move from place to place without requiring the accessing application to be aware of the move. Providing both these properties requires a level of indirection between pathnames and physical locations. VFDS maps global pathnames to Logical File Names (LFN) that can be used to resolve their locations through a replica location service provided by their associated virtual organization (VO) [Grid]. Decoupling the visible

[1] These pathnames used on different client platforms may differ syntactically in well-known ways, e.g., Windows clients may have a drive letter prefix and use backslashes to delimit directory and file names in these pathnames, while Unix clients will not use a drive letter and will use forward slashes.

pathnames from their physical addresses allows the logical and physical aspects of the file storage system to be managed independently, which promotes scalability from an administrative perspective. In addition, the users and creators of the data control its logical arrangement, while the physical layout of the data depends upon storage, server and network resources that are often managed by different people. For this reason, VFDS access control of the namespace should be independent of permissions on the replica location service describing the physical infrastructure. The global scope of the namespace provides a way to link file data across organizations without compromising security or autonomy. This smoothly supports cross-domain virtual organizations as well as looser collaborations that span organizations. The use of a hierarchy to arrange the namespace recognizes that both security and classification often rely upon hierarchical structures. Therefore, assigning unique names to files and controlling access to them are most easily accomplished by using pathnames.

The VFDS pathname model encourages addressing file data at the directory subtree or filesystem granularity. This enables a transparency of protocol that allows inclusion of data from a variety of distributed file systems. With appropriate protocol conversion mechanisms a global federation of file systems is possible. Furthermore, the aggregation of files into subtrees provides natural collections that would allow scalable management of namespace and replication.

The namespace is composed of virtual directories maintained in VFDS and links to files and subtrees provided by physical file systems. VFDS is described in more detail in section 3. Clients translate pathnames by contacting VFDS and interpreting and traversing links to reach a file or file system. The linking step may involve a logical to physical mapping using a replica location service to determine the protocol and server address for accessing the file system. The client code can be in a proxy, a native file system driver, or application libraries. Section 4 explains these components and operations in more depth. In this paper we focus on a description of VFDS in terms of its proposed functions and how they can be used. At this time, we are not proposing an implementation design. With this paper, we intend to facilitate discussions on refining the VFDS model and identifying the best implementation path. Alternative implementation approaches to be considered include LDAP, Globus MDS, DNS, and existing Grid services.

2. Related Work and Technologies

2.1 CIFS and NFS

Existing distributed file systems, such as CIFS and current versions of NFS (V2 and V3), have some shortcomings in their application to Grid and wide-area file sharing. Although each has its own naming architecture that allows large file spaces to be assembled, they take a provincial view of uniformity. They are single protocol approaches and often provide only naming at the client, server or local domain level. Typical deployments of these file systems allow a surprisingly weak approach to security, which is counter to the Grid strategy. Likewise the approach taken to location independence and replication is far from ideal.

The effort to extend the popular and successful NFS protocol to wide-area networks has resulted in a fourth version developed under the auspices of the IETF. It is currently a proposed standard described by RFC3530 [NFSv4]. While this protocol has many enhancements compared to earlier versions, it still does not provide for a uniform namespace that would truly unite NFSv4 clients everywhere into a global file system. However, it is complicated, and many features crucial to use in a wide-area environment are optional. Early implementations are not yet ready for wide use, even by the development community, so its stability, availability and performance remain unknown. Because NFSv4 and the Grid share some design goals, it promises to be an important component of a Grid File System.

2.2 AFS/DFS

In many ways experience with AFS and DFS has shown the value of a global, uniform namespace for collaborative activities. Both have a strong commitment to security, compatibility with wide-area networks, and scalability. Scalability has been demonstrated over geography, users, servers, clients, and data. A Grid File System could advantageously copy some of these scalability features, for example groups that users can administer themselves. DFS works as a general exporter and is able to export many local file systems to remote clients, while AFS file storage uses a proprietary organization that makes sharing existing file systems impossible.

Neither, however, is very suitable as a Grid File System, though each has different strengths. Integration with Grid security and existing transport protocols would be a significant task. DFS is a complex system that is tightly integrated with DCE. DCE provides some of the same features as the Grid but using incompatible mechanisms. AFS has a simpler security architecture that is better isolated and so would probably be more amenable to Grid integration. Both require client software installation that does not come with common operating systems. AFS has a flourishing worldwide open-source development community [AFS]; the ownership of DFS is complicated.

2.3 Avaki

Avaki Data Grid provides application and user clients a uniform namespace through an NFS gateway (Data Access Server in Avaki terminology). However, it does not address replication and it is a proprietary technology that does not leverage or interoperate with Grid protocols such as GridFTP and RLS. Several researchers at Avaki proposed a generic namespace specification [SGNP] which, at an abstract level, has similarities with some aspects of our VFDS proposal. SGNP does not specify, however, how real object or resource types, such as file system or database objects, are represented.

2.4 Globus

Globus technology provides much of the crucial infrastructure needed to implement a Grid File System. The high standards for security provided by Grid Security Infrastructure (GSI) provide a good basis for all components of the Grid File System. A Grid File System of Internet scale may be a challenge to the adequacy of the centralized certificate authority model, but Grid's public key based authentication model is more scalable and global than alternatives. Beyond authentication, efforts such as the Grid Community Authorization Service (CAS) provide a delegation framework and may be useful for file system level access control. On the other hand, the ACL and group server model of AFS and DFS may be a better model for file system authorization. RLS can be easily adapted beyond its core task of locating replicas to provide location independence of unreplicated data. The scalability of RLS is not crucial since multiple independent RLS instances can be utilized for distinct parts of the namespace. While the lack of structure in logical file names (LFN) makes them unsuitable as pathnames, it does make RLS very adaptable.

GridFTP provides a secure and efficient method for file transport. In situations where the strongest security and highest performance are required, it is the protocol of choice. Enhancements to the protocol, such as MLST [FTPext] promise to make it suitable in a wider range of circumstances. Adding capabilities for setting attributes and providing synchronization primitives such as file locking is also possible. It is an open question, however, whether GridFTP should evolve very far in this direction, or if other approaches are more suitable.

2.5 Other Related Technologies

Good design principles suggest that every new proposed service will be scrutinized to determine whether its functionalities already exist or can be subsumed by existing services. It is possible that VFDS can be implemented using some existing technologies or infrastructures. Such

candidates include Lightweight Directory Access Protocol (LDAP), Globus Monitoring and Discovery Service (MDS) and Domain Name Service (DNS) and other existing Gird service technologies. A full evaluation of these possibilities is outside the scope of this paper.

2.5.1 LDAP

The Lightweight Directory Access Protocol provides an interface for accessing and searching a database of entries annotated with attributes. The LDAP hierarchy is tightly tied to entry attributes and is more restrictive than file system directories. With the definition of an appropriate schema, LDAP may be a useful component of a VFDS implementation.

2.5.2 Globus MDS (Monitoring and Discovery Service)

Globus MDS [MDS] provides an information services architecture that aims to address performance, security, scalability, and robustness requirements. The architecture comprises two fundamental entities: highly distributed information providers and specialized aggregate directory services. Information providers support access to detailed, dynamic information about Grid entities and aggregate directories provide specialized views of federated resources or services. The LDAP data model is used to represent information as a set of objects organized in a hierarchical namespace. With appropriate schema, an LDAP-based MDS may provide a useful way of aggregating file metadata information to allow fast and large-scale search on files.

2.6 Domain Name Service

DNS provides a hierarchical database of entries that could represent VFDS objects. DNS has demonstrated scalability, flexibility and robustness in same environment that VFDS aspires to. The architecture of DNS may be suitable for VFDS implementation, though, the existing DNS infrastructure probably is not. It would be problematic to require DNS administrators and file system namespace administrators to coordinate on joint management. The constraints and requirements on the two systems are too different for this to be a tenable approach.

3. VFDS Features

This section describes the characteristics of VFDS and defers a discussion of how it is used to the following section on Operational Aspects.

3.1 Directory Structure and Junctions

A VFDS instance consists of a collection of directories organized into a singly rooted tree. Each directory entry has a name and is either a subsidiary directory or a junction. Pathnames are evaluated starting at the instance's root and continuing within the VFDS instance by traversing directories until the pathname is exhausted or a junction is reached. A junction consists of a target that can point to another VFDS instance or to a file system object. Each junction represents a delegation for handling subsequent portions of a pathname to another service, which can be either a VFDS or a file server.

A junction target pointing to another VFDS instance allows the namespace to be assembled from separate VFDS instances. A file system target that is a file or directory can be represented in the form of a URL. When the target is a directory, the subtree rooted in that directory is attached. Most conventional file systems do not contain junctions[2], so file system targets are terminals of

[2] Junctions could be inserted into existing file systems using several methods. A file system object could represent the junction, either a new usage of an existing object (e.g. a symbolic link) or definition of a new object. Alternatively, the VFDS could define additional virtual directories that "overlay" the existing file system's physical directories leading to the junction. These overlaying VFDS directories and junction may override existing file system object or represent new ones.

the namespace in the sense that subsequent names are confined to the same file system. File system targets can be logical or physical; in the former case, a replica location service is used to map the logical name to a physical address containing a specific server name and protocol (i.e. a URL). The target URL describes the protocol or interface, the service or host address and a pathname prefix. In the case of logical targets, the prefix is a logical file name to be presented to the replica location service and the result (after replica selection, if several are returned) indicates the target file or directory. Then the unevaluated part of the pathname is appended to the URL's pathname prefix to produce the input to the next service. Thus, there are five types of junctions depending on the target summarized in the following table.

Junction type	**Target example**	**Description**
VFDS	vfds://vfds.ibm.com/	another VFDS instance
logical subtree	rls://rls.ibm.com/arc_cs_storage	a directory subtree located via a replica location service
logical file	rls://rls.ibm.com/8493802	a file located via a replica location service
physical subtree	gsiftp://shark.tucson.ibm.com	a specific server's directory subtree
physical file	gsiftp://homepages.aol.com/~ota/house.gif	a file on a specific server

An example use of VFDS is depicted in Figure 1. Under the global root `/grid` directory, two VFDS junction points are shown, which provide the junctions to the namespaces maintained separately by two different organizations (`ibm.com` for IBM and `globus.org` for Globus). Within `ibm.com`, two virtual directories are shown (`SG` and `ARC`). These virtual directories are only represented in VFDS and they do not exist in any physical file systems. Virtual directories allow physical file systems or files to be hierarchically organized in the virtual namespace. Under `SG`, `shark` is a junction to the physical directory subtree available at the GridFTP server `shark.tucson.ibm.com`. Under `ARC`, `csstorage` is a junction to a logical directory subtree with the name `arc_cs_storage`. This logical directory subtree name is resolved through a lookup to a replica location service. This logical directory subtree is shown served out of a GridFTP server `csstorage.almaden.ibm.com` and an NFS file system `csstor` on `nas1.almaden.ibm.com`. A junction to a logical file, `fileX`, (in `ais`, a virtual directory) is also shown under the `ARC` virtual directory. VFDS maps `fileX` to a logical file name `8493802`, which is used for a lookup to a replication location service and mapped to two GridFTP server based files.

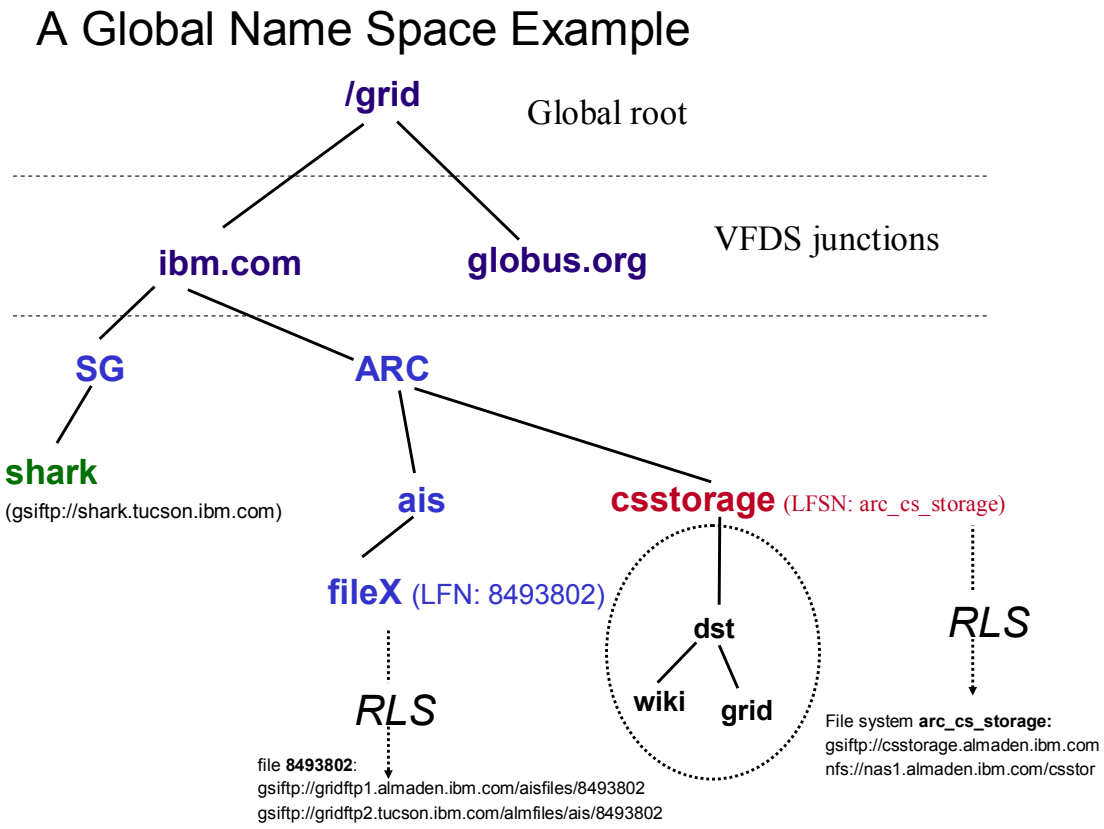

Figure 1: A Global Namespace Example

3.2 Metadata and Access Control

The namespace plays a crucial role in a file system because it provides the mapping between pathnames that are visible to the user and physical addresses for the files and ultimately the content itself. In this respect, the namespace gives meaning to the pathnames. The security of this mapping is paramount, so both updates and lookups to the namespace must be authorized by an access control mechanism. For this reason access control lists (ACL) can be associated with each VFDS entry. This allows creation and deletion of junctions and VFDS directories to be restricted. In addition, since names and their mappings can be sensitive, the ability to read a directory and lookup a name can also be controlled.

Naming and directory information is usually considered part of file system metadata. But other metadata can also be associated with each VFDS entry. As we have seen, access control information, such as an ACL, is a prime example of this. Metadata that help in auditing, such as auditing flags, owner, modification times, etc, can also be associated with entries. Other metadata include: object type, data version number, data (MIME) type, hidden, etc. Some of these attributes can be stored by VFDS with the junction, while others are maintained by the physical file system with the object that is the target of the junction. Good candidates for the latter type include size and space used. Other kinds of metadata are more like configuration or option information associated with aggregates such as directories, filesystems or file servers such as max file size, space available, and the like.

A directory junction (i.e. one whose target is a directory) delegates metadata handling to the target file system, but a file junction allows VFDS to provide metadata and exercise access

control. In some configurations, VFDS directories could replace those in the physical file system by being populated with junctions to individual files. In this case, the metadata of a physical file system is supplemented by utilizing VFDS to overlay its directory structure. The VFDS metadata can include attributes useful to applications and access control permissions affecting the file contents that could enhance the security provided by the file service alone. The authorization for some operations is enforced by VFDS (e.g. lookup) and by the file system for other operations (e.g. read and write). Some operations such as create and delete are controlled jointly.[3] This split responsibility for security can cause consistency problems if some access paths utilize VFDS and others reach the file service directly. One remedy for this would be to prohibit independent access by having the file service accept only access tokens, or capabilities, generated by VFDS. These capabilities would allow enforcement of the division of labor between the services. Without the need to maintain its own directory structure consistent with that of VFDS, the file server can dispense with directories altogether and become a pure object store while VFDS adopts the role of an object file system.

3.3 Interface

The VFDS interface provides a way to consult and modify the namespace. Input paths are relative to VFDS, which means that a VFDS instance does not know where in the global namespace its subtree resides. In fact, a VFDS instance may appear in multiple places if several junctions refer to it. This arrangement cannot avoid cycles, so clients must detect loops. The lookup operation can accept multi-component pathname fragments and traverse all the directories within the same VFDS instance. This presents a trade-off, however, between privacy and efficiency; large lookup request result in fewer requests, but extraneous components may provide VFDS with more information about client access patterns than desirable.

The basic interface is similar to that used by file systems, except that only metadata and directory operations are needed (e.g. lookup, create, setattr, etc.). The junction information returned by lookup and provided to create can be in the form of a URL and so these operations are similar to those for symbolic links. Attributes include access control information as well as some of those common in traditional file systems. Directory names are case insensitive since that doesn't drastically reduce functionality and improves convenience and compatibility with DNS and Windows.

Synchronization mechanisms provide atomicity for updates as well as notification when updates occur. The requirements will vary greatly depending upon the quantity of read and write references VFDS experiences. For high levels of the namespace, where writes are infrequent, a time-to-live strategy for discovering changes will be adequate. At lower levels, to better support sharing between readers and writers, more sophisticated techniques will be necessary. Detailed requirements and mechanism need to be determined.

4. Operational Aspects

4.1 Namespace Organization

The namespace is divided into three sections: root, VFDS, and physical file systems. The root is the directory that defines the top-level names. The VFDS level defines an arrangement of virtual directories that lead to junction points for files and directory subtrees, while the filesystem portion is specified by the terminal file systems. Conventions can be used to layout the root of the namespace to avoid the thorny scalability and ownership issues that arise when providing global uniformity for pathnames. These problems can be avoided by allowing local autonomy over the

[3] In the case of virtual directories containing junctions to files, creation of new files may require additional information about where they should be stored. This is one way that VFDS becomes even more like an object-oriented file server.

appearance of the namespace root while discouraging overuse of that independence. The root directory is basically a superposition of the names, and their mappings, from several ordered sources. Highest priority is a local configuration file, giving a few special names with local or private use such as a temporary storage area. Second could be another file providing mappings for resources within the organization, such as corporate intranet data. Lastly, would be a catchall based on the domain name system (DNS) to locate VFDS servers in the Internet, perhaps using AFSDB or TXT records. Essentially, the root directory is a separate VFDS synthesizing a single directory whose contents are composed from multiple configuration specified sources.

After the first VFDS instance is reached, subsequent names are referred to a series of VFDS instances each of which resolves one or more path components and produces a reference to another target. As the pathname is traversed, one VFDS links to another until a target indicates a terminal file system. Within a file system, name resolution proceeds in the usual way. The resulting files and directories are accessed as appropriate for the protocol used to contact the file service.

Each VFDS or file system operates as proprietor of its portion of the namespace, which allows control to be delegated from one entity to another. The ownership, and hence the authorizing principal, of every name is well defined.

4.2 Security

VFDS uses the Grid Security Infrastructure (GSI) to authenticate both services and service requesters. A pathname traversal, however, may involve several VFDS instances and a file system. Each step requires contact with a different server. In addition, mapping from a logical to physical file system involves one or more implicit steps. While each server must make an authorization decision based on the requesting principal, the requester must also authenticate each server to assure itself that the resulting information is reliable. This is particularly important to ensure that the data associated with a pathname is the data intended by the pathname's creators. There are two approaches to authenticating servers. The most common approach is to derive the service principal name (X.509 subject) in some well-defined way from the name of the service or the server hosting it, and requiring the server to present a verifiable certificate for the same. Another approach is to include the principal name or public key in the reference to the service itself, and authenticating the server against that key. In the case of VFDS, the junction or its associated metadata could include this principal information. This latter strategy may be particularly useful in the case of implicit replica locating steps that link the logical to the physical file system.

The security of the process depends on the security of each step. When multiple protocols are involved, the weakest link governs the overall strength. Because VFDS provides strong security via GSI for the initial steps of the lookup process, the resulting security will depend on that utilized by the terminal file system. When the terminal file system is GridFTP, then GSI is used throughout, but, for example, when the terminal file system is NFSv2 or NFSv3, all but the last stage may be secure. This arrangement may provide a useful incentive to improve the security of terminal file systems: for example, by upgrading to NFSv4 the entire path can be secured. Thus, the use of GSI in VFDS allows, but does not require, strong security to be used to access files throughout the namespace.

The ACL mechanism is much improved by the use of groups of users. Groups allow more compact and comprehensible ACLs, which make the security they provide more reliable and accessible. Groups also allow the definition of meaningful collections of users to be delegated. A scalable implementation tracking group membership cannot restrict updates to administrators but needs to allow individual users to create and maintain their own groups. For example, members of a team can limit the access to their shared work areas to the team using ACLs, but without groups each team member needs to keep track of team membership changes and update all appropriate ACLs. Clearly, serious use of ACLs requires a group mechanism. Equally obvious is

that a global file system with a global authentication system needs a way to manage groups that also has global extent. Such groups would need to support global membership and should also have a global naming scheme. This problem needs to be addressed.

4.3 OGSI Service

VFDS can be accessed as a Grid Service by defining its interface in terms of WSDL. This will be especially useful for administrative operations that can take advantage of generic tools.

4.4 Interoperability Interfaces

Accessing VFDS data using conventional file system interfaces will be a convenient way to provide wide availability of this service and foster interoperability. VFDS is a directory service, so its interface can be easily mapped to that of a conventional distributed file system. A wrapper module would provide translation between conventional protocols and the interface to each VFDS instance. It may also be desirable to add support for some file system protocols directly into the VFDS service.

Besides the protocol mapping, there are some issues in matching semantic assumptions, for example, the difference in the meaning of attributes as defined by a specific protocol and those intended by VFDS. The complications involve simulating unsupported attributes, representing junctions and handling synchronization requests. Fabricating missing attributes should be effective and generally easy, though in some cases, for example, NFS permission bits, there may be significant issues to address. Synchronization refers generically to communication paths between users accessing the same data, which includes byte-range and session locking and cache consistency mechanisms such as OpLocks [CIFS], time-to-live and callbacks. For many uses, synchronization requests can be matched well enough to equivalent features of distributed file system protocols.[4]

Creating a comprehensible representation of junctions is critical to the utility of VFDS. The NFSv4, CIFS and HTTP protocols all support some kind of referrals that can be adapted to return junction target information. These referrals transfer responsibility for pathname traversal from one VFDS to another and then finally to a server that natively supports the protocol. Supporting older versions of NFS or FTP would require a proxy to perform both namespace traversals and file data transport. Possible enhancements to the GridFTP support for third-party transfers would allow a proxy to perform namespace and replica location service lookups, then setup the data transfer directly between the GridFTP client and the physical site of the file.

4.5 Directory Service Clients

Translating a pathname into a physical address requires client code that can contact VFDS, traverse the hierarchy and do logical to physical mappings. There are a number of ways to do this: VFDS wrappers, proxies, libraries and native file system clients. These have a range utilities and performance characteristics in different environments.

When the clients and servers have a suitable distributed file system protocol in common, VFDS can be accessed via a wrapper for the protocol, as described above. Connecting incompatible clients and servers is more problematical. However, the availability of Samba and multi-protocol NAS appliances makes this less troublesome than it once was. A simple approach is to use a proxy, such as a gateway or a caching appliance, which handles protocol translation. A VFDS proxy provides a convenient spot for applying location and replication transparency. After traversing the namespace, the resulting logical file or file system name is translated using a

[4] Defining an appropriate range of synchronization mechanisms that balance performance and functional requirements across the whole spectrum of VFDS environments is challenging and requires additional effort.

replica location service into physical names that explicitly specify protocols and servers. The proxy is also a place where remote content can be cached. In addition, to the usual advantages of caching, all users of a proxy can benefit from sharing the locally cached content. In this capacity, a shared proxy provides a vehicle for hiding the protocol and cache management complexities from both applications and native file system clients.

Of course, the functions of VFDS lookup, location resolution and protocol translation can take place in application libraries or native clients instead. This approach may have higher performance if the advantages of tighter integration outweigh the benefits of the proxy approach, but the development costs are certainly higher.

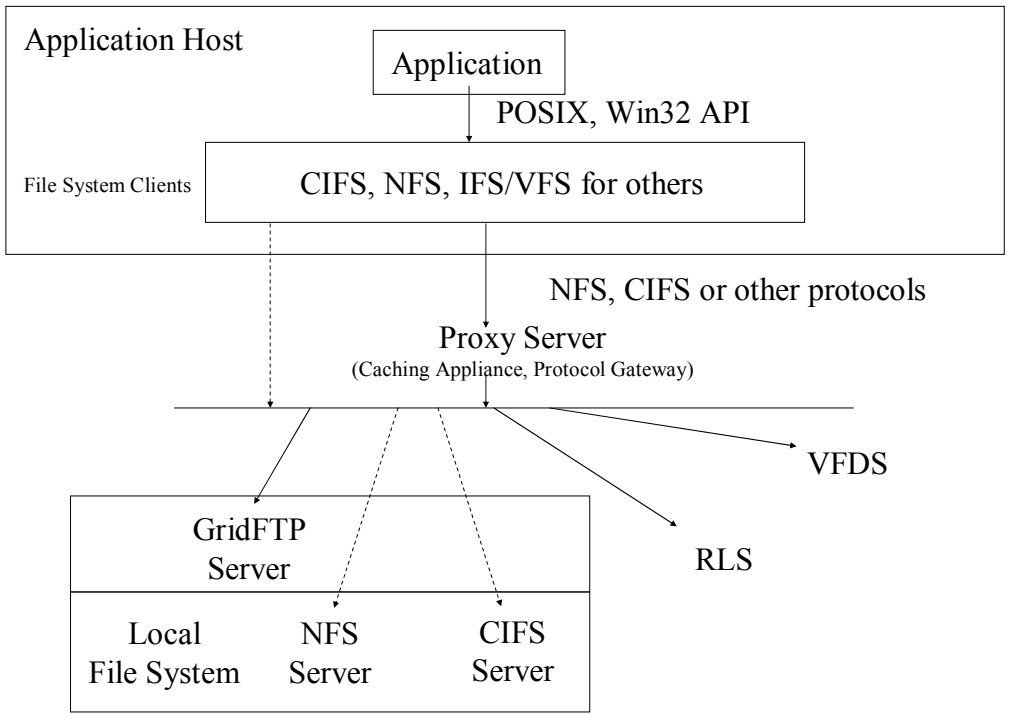

Figure 2: Architecture Possibilities for VFDS Clients

Figure 2 depicts the architecture possibilities for VFDS clients described above.

5. Security Considerations

VFDS uses the Grid Security Infrastructure (GSI) to authenticate both services and service requesters. Various security issues are discussed in section 3.2 Metadata and Access Control and section 4.2 Security.

6. Summary

In this paper we propose to design a Virtual Filesystem Directory Service (VFDS) to enable the realization of a Grid File System by providing a uniform, global, hierarchical namespace. Such a directory service can leverage GridFTP and Globus RLS for secure and high-performance access to files and replica locations as well as allow the inclusion of file data available in conventional distributed file systems. At this time, we are not proposing an implementation design. We

propose to discuss and refine the VFDS values as well as its functionalities and identify the best way to implement them. Alternative implementation approaches to be considered include LDAP, Globus MDS, DNS, and existing Grid services.

7. Acknowledgements

Our colleagues in the Distributed Storage Tank team at IBM were invaluable in helping refine the ideas presented in this paper. Early discussions with Grid Datafarm researchers Osamu Tatebe of AIST and Satoshi Matsuoka at Titech were helpful in defining the scope of VFDS. A prototype was created as part of an IBM Extreme Blue project at Almaden in Spring 2003. Recent discussions with Karl Czajkowski and Ann Chervenak of USC ISI, and Jay Unger and Jane Xu of IBM helped expand our considerations of future implementation approaches. Binny Gill and Manuel Pereira provided helpful comments on the paper.

Author Information

Leo Luan, Ted Anderson
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120, USA

Intellectual Property Statement

Full Copyright Notice

References

[AFS] http://www.openafs.org/

[Avaki] Andrew Grimshaw, “Enterprise Data Grids”, Tutorial Presentation, GGF7 (Global Grid Forum), Tokyo, Japan, March 2003.

[CAS] http://www.globus.org/security/CAS/

[CIFS] http://www.snia.org/tech_activities/CIFS/

[DFS] http://arjun.in.ibm.com/dfs310/AdminGd/duagd002.htm

[FTPext] http://www.ietf.org/internet-drafts/draft-ietf-ftpext-mlst-16.txt

[Grid] Ian Foster, Carl Kesselman, and Steven Tuecke, "The Anatomy of the Grid Enabling Scalable Virtual Organizations". http://www.globus.org/research/papers/anatomy.pdf

[GridFTP] http://www-isd.fnal.gov/gridftp-wg/draft/GridFTPRev2.pdf

[GSI] http://www.globus.org/security/

[LDAP] Heinz Johner, Larry Brown, Franz-Stefan Hinner, Wolfgang Reis, and Johan Westman, "Understanding LDAP", June 1998. http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg244986.html

[MDS] Karl Czajkowski, Steven Fitzgerald, Ian Foster, Carl Kesselman, “Grid Information Services for Distributed Resource Sharing”, Proc. 10th IEEE International Symposium on High Performance Distributed Computing (HDPC-10), IEEE Press, 2001.

[NFSv4] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, D. Noveck, "Network File System (NFS) Version 4 Protocol", RFC3530, April 2003. http://www.ietf.org/rfc/rfc3530.txt

[RLS] Ann Chervenak, Ewa Deelman, Ian Foster, Leanne Guy, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripenu, Bob Schwartzkopf, Heinz Stocking, Kurt Stockinger, Brian Tierney, “Giggle: A Framework for Constructing Scalable Replica Location Services”, to appear in Proceedings of SC2002 Conference, November 2002. http://www.isi.edu/~annc/rls/chervenakFinalSC2002.pdf

[SGNP] Joshua Apgar, Andrew Grimshaw, Steven Harris, Marty Humphrey, Anh Nguyen-Tuong, “Secure Grid Naming Protocol (SGNP): Draft Specification for Review and Comment”, http://www.gridforum.org/Meetings/ggf4/bofs/SGNP%20-%20GWD-R%202002.02.05.pdf.