THE GLOBUS PROJECT

White Paper

Universal Data Transfer for the Grid

WHITE PAPER

Universal Data Transfer for the Grid

April 27, 2000

Introduction

"Access to distributed data is typically as important as access to distributed computational resources."

The Globus Project aims to develop new technologies that enable the creation of stable computational grids. Computational grids provide the computational infrastructure for powerful new tools for scientific investigation, including desktop supercomputing, smart instruments, collaborative environments, and distributed supercomputing.[1] Our strategy is to focus on services at the middleware level, where we generalize the requirements of grid applications and deliver technologies that support entire classes of applications. The middleware layer offers services for managing large numbers of diverse computational resources administered by independent organizations, and provides application developers with a simplified view of the resulting computational environment.

Our group has done considerable work in collaboration with scientific communities, developing applications that take advantage of the middleware capabilities offered by the Globus Toolkit. We have learned through this collaboration that *access to distributed data* is typically as important as *access to distributed computational resources*. Distributed scientific and engineering applications typically require access to large amounts of data (terabytes or petabytes). Future applications envisioned by our team also require widely distributed access to data. (For example, access in many places by many people, virtual collaborative environments, etc.)

In many cases, application requirements call for the ability to read large datasets and to create new datasets. They often do not require the ability to change existing datasets. Consequently, the distributed scientific computing community has envisioned a tiered model for data storage. From the perspective of a specific organization or location, this model would include:

- large locally-stored datasets,
- locally-stored replicas of datasets from remote locations, and
- access to remote datasets that are not replicated locally.

[1] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115-128, 1997.

The terms "local" and "remote" are imprecise. Locality is really a matter of degrees. For example, in some application scenarios, there isn't enough room on "local" disk (/tmp, etc), so it is necessary to access "remote" data residing on a storage system that is in close physical proximity to the computer. In other cases, the issue is not so much available space as it is data management policies; applications need to access data that is "remote" (stored at other organizations) because the management policy does not permit the data to be copied to a more "local" location.

In order to support this model, grid middleware should provide both a means of managing the datasets described in this model, and a number of basic mechanisms that generalize the requirements of most data-intensive applications. It is our intent, therefore, to offer a generic infrastructure in the form of core data transfer services and generic data management libraries. These fundamental building blocks can then be used in a variety of interesting ways to build systems and applications.

The Globus Project is currently engaged in defining and developing the following core capabilities, which we believe will be necessary in order to build a persistent *Data Grid* environment.

- A high-performance, secure, robust data transfer mechanism
- A set of tools for creating and manipulating replicas of large datasets
- A mechanism for maintaining a catalog of dataset replicas.

Relationship to existing distributed data storage systems

There are already a number of distributed storage resource managers in use by the scientific and engineering community. These storage systems have been created in response to specific needs for storing and accessing large datasets. They each focus on a distinct set of requirements and provide distinct services to their clients.

For example, some storage systems (DPSS, HPSS) focus on high-performance access to data and utilize parallel data transfer streams and/or striping across multiple servers to improve performance.[2] [3] Other systems (DFS) focus on supporting high-volume usage and utilize dataset replication and local caching to divide and balance server load.[4] The SRB system connects heterogenous data collections and provides a uniform client interface to these repositories, and

[2] B. Tierney, W. Johnston, J. Lee, G. Hoo, and M. Thompson. End-to-end performance analysis of high speed distributed storage systems in wide area ATM networks. In *NASA/Goddard Conference on Mass Storage Systems and Technologies*, 1996, LBNL-39064.

[3] R.W. Watson and R.A. Coyne. The parallel I/O architecture of the high-performance storage system (HPSS). In *IEEE MSS Symposium*, 1995.

[4] It is interesting to note that DFS also provides for cache coherence in the face of multiple writers. This is one of its drawbacks for data grid applications, since this feature forces unnecessary overhead. As mentioned previously, data grid applications seldom need the ability to change existing datasets.

also provides metadata for use in identifying and locating data within the storage system.[5] Still other services (HDF5) focus on the structure of the data, and provide services for accessing structured data from a variety of underlying storage systems.[6]

Unfortunately, most of these customized storage systems utilize incompatible protocols for accessing data and require the use of their own clients. The use of multiple incompatible protocols for data storage effectively partitions the datasets available on the grid. Applications that require access to data stored in different storage systems must use different methods to retrieve data from each system. It can be challenging to transfer a dataset from one system to another.

We believe that it would be mutually advantageous to have a common level of interoperability between all of these disparate systems: a common—but extensible—underlying data transfer protocol. A common data transfer protocol for all of these customized storage systems would confer benefits to both the keepers of large datasets and the users of these datasets. Dataset providers would gain a broader user base, because their data would be available to any client. Dataset users would gain access to a broader range of storage systems and data.

Furthermore, establishing a common data transfer protocol would eliminate the current duplication of effort in developing unique data transfer capabilities for different storage systems. A pooling of effort in the data transfer protocol area would lead to greater reliability, performance, and overall features that would then be available to all distributed storage systems.

Related efforts

In addition to the work being done on the storage systems mentioned above, we also expect that the results of this work will be of use in the following specific communities.

- The high-energy physics community (e.g., CERN, Fermi, Argonne)
- The climate modeling community
- The earthquake simulation and modeling community
- The NASA data mining project

[5] Information on SRB is available on the World Wide Web at http://www.npaci.edu/DICE/SRB/.

[6] Information about HDF/HDF5 is available on the World Wide Web at http://hdf.ncsa.uiuc.edu/.

Motivation for a common transfer mechanism

"The use of multiple incompatible protocols for data storage effectively partitions the datasets available on the grid."

As described in the introduction to this paper, the use of multiple storage systems has led to a partitioning of the available datasets and storage services. Breaking through the partitions created by these incompatible storage systems can be achieved by creating a layer of interoperability either above or below the storage systems.

Adding a layer of interoperability above the storage system is the most obvious approach. This would involve the creation of a "metastorage system", or an abstraction of existing storage systems that uses a number of different logical storage systems to implement high-level storage features. Advantages of this approach include a common interface for applications that require data storage functionality and the ability to accomplish the objective without any changes to existing storage systems. Disadvantages include the need to develop interfaces to existing storage systems (and new interfaces when new storage systems are invented) and the unfortunate effect that specialized features of particular storage systems would be hidden behind the abstracted service layer.

Adding a layer of interoperability beneath the storage system would involve decoupling the low-level data transfer mechanisms from the distributed storage services that use them. Once this is done, a common data transfer mechanism (using a single, universal data transfer protocol) can be used for all of the storage systems. Advantages of this approach include the ability to develop new specialized storage systems that are automatically compatible with existing systems, and the fact that existing data storage systems could take advantage of a richer set of data transfer functionality. The major disadvantage is that the developers of existing storage systems must retrofit their systems to use the common data transfer mechanism.

Most of the existing distributed storage services already decouple their high-level services from the underlying data transfer mechanism. We have already collaborated with UCSD in implementing GSSAPI security in the SRB system's data transfer services. The HPSS system uses a parallel FTP service as its data transfer mechanism. NCSA's HDF5 implementation utilizes a "virtual file driver" mechanism that provides structured file access via arbitrary storage systems and data transfer mechanisms. (HDF5 has also been demonstrated to work with the GASS transfer mechanism.)

This pre-existing decoupling offers us the opportunity to experiment with a common data transfer protocol for any or all of the above data storage services. The existence of a common wire protocol for data transfer means that there is a fundamental level of interoperability between systems that use the common protocol. Whether storage system developers write their own code for implementing the common wire protocol or use code provided by the Globus Toolkit, the use of a common protocol will ensure that their storage systems have a high-performance data transfer capability that automatically interoperates with other storage systems.

Characteristics of the data transfer mechanism

In order to make this proposition attractive to the users and developers of existing storage systems, we must provide a transfer mechanism that offers a superset of the features offered by any of the mechanisms currently in use. A common data transfer protocol for the Grid would ideally offer all of the features currently available from any of the protocols currently in use. At a minimum, it must offer all of the features that are required for the types of scientific and engineering applications that we intend to support on the Grid.

We have observed that the FTP protocol is the protocol most commonly used for data transfer on the Internet, and the most likely candidate for meeting the grid's needs. It is attractive in particular for the following reasons.

- It is a widely implemented and well-understood IETF standard protocol.
- It provides a well-defined architecture for protocol extensions, and supports dynamic discovery of the extensions supported by a particular implementation.
- Numerous groups have added various extensions through the IETF. Some of these extensions would be particularly useful in the grid.
- It supports transfers between client and server.
- It supports third party transfers between two servers.

We have already implemented an FTP client and server with support for GSSAPI security[7] following protocol extensions defined by the Internet community.[8] The GSSAPI supports either PKI or Kerberos authentication.

Most current FTP implementations support only a subset of the features defined in the FTP protocol (RFC 969) and its accepted extensions. Some of the seldom-implemented features would be useful to data grid applications. The standards also lack several features which data grid applications require. We intend to select a subset of the existing FTP standard and further extend it, adding the following features. We believe that the resulting protocol will be a suitable candidate for the common data transfer protocol for the grid.

Automatic negotiation of TCP buffer/window sizes

Manually setting TCP buffer/window sizes is an error-prone process (particularly for non-experts) and is often simply not done. Our enhanced FTP will support automatic negotiation of TCP buffer sizes both for large files and large sets of small files.

Parallel data transfer

On wide-area links, using multiple TCP streams (even between the same source and destination) can improve aggregate bandwidth over using a single TCP stream. Also, partitioning data across

[7] Information about the Globus gsiftp is available on the World Wide Web at http://www.globus.org/security/v1.1/.

[8]

multiple servers will further improve performance. Our enhanced FTP will support parallel data transfer, both from a single server and from multiple servers. We will additionally propose a protocol mechanism for automatic negotiation of the level of parallelization in a data transfer, and may provide a reference implementation.

Third-party control of data transfer

In order to manage large data sets (including replication), it is necessary to provide third-party control of transfers between storage servers. Our enhanced FTP will provide this capability by adding GSSAPI security to the existing third-party transfer capability defined in the FTP standard.

Partial file transfer

Many applications require transfer of only a portion of a file. Transferring the entire file could be too expensive. Our enhanced FTP will support partial file transfer.

Security

Security is critical when transferring or managing files. Our enhanced FTP will implement anonymous and GSSAPI authentication with optional integrity and/or privacy, as defined by the existing GSSAPI-enabled FTP standard. Though we will not provide higher-level authorization systems at this time, we will provide and demonstrate the necessary hooks for these systems.

Support for reliable data transfer

Reliable transfer is important for many applications that manage data. Fault recovery methods for handling transient network failures, server outages, etc. are needed. The FTP standard includes basic features for restarting failed transfer that are not widely implemented. We will provide these features in our implementation.

Furthermore, users should not be required to code to a different API to get advanced reliability services, and we do not wish to dictate specific fault recovery algorithms. A plug-in interface to the data transfer service will allow us to add a reliability service at run time. Our enhanced FTP will include a plug-in interface for reliability, as well as one or more plug-ins that implement specific reliability services.

Implementation plan

The following sections explain our intended plan for accomplishing what is described in this paper. The Globus team at Argonne National Laboratory and the University of Southern California Information Sciences Institute will be working intensively on this during the remainder of the first half of 2000.

Needless to say, we will be working closely with a variety of groups at other institutions engaged in related work. These are noted in the sections below where applicable.

Documentation

This white paper serves as the first piece of documentation regarding this work. Its purpose is to explain our goals and strategy to various Grid and application communities and to stimulate discussion and feedback.

We also intend to develop and submit a draft RFC to the IETF that outlines the specific protocol extensions that we have selected from existing FTP RFCs and the new extensions that we intend to add to this set of features in support of the Data Grid. We will use the RFC process to standardize our candidate Data Grid transfer protocol.

Production Libraries

Our strategy of providing infrastructure-level building blocks for the Data Grid calls for a series of libraries that are of sufficiently high quality that they can be used in new and existing storage systems and applications. These libraries will provide the basic client and server protocol support for applications and storage systems that wish to participate in the Data Grid.

Specific libraries will include:

- Protocol client and server libraries (client side will support both C and Java)
- A client library to support third-party transfers (for C and Java)
- A protocol driver for the Globus GASS client (for C and Java)
- A library to provide support for the "URL copy" function (for C and Java)
- At least one "reliable transfer" plug-in for the protocol client library (for C and Java)
- A library for basic LDAP manipulation of replica catalogs (for C and Java)
- A replica management library (C and Java)

Production Tools

We will provide a number of tools (executable programs) that provide low-level data grid management features.

- A program that copies the contents of one URL to a new URL.
- A set of programs for manipulating replica catalogs.
- An FTP server that is fully compliant with our draft RFC (see "Documentation" above).

Patches for Third-Party Software

In order to encourage the use of our Data Grid transfer protocol, we will provide patches to the existing NCFTP client and the WUFTPD server. (These FTP implementations are both provided in source form with open source licenses.)

We will also work with IBM (the designers of the HPSS storage system) and the developers of the parallel FTP service that HPSS uses to add any remaining features to their protocol implementation, making HPSS the first fully-compliant storage system for the Data Grid.

Prototypes and Applications

In addition to designing a data transfer protocol for the Data Grid, pursuing the standards process, developing a reference implementation of the protocol and libraries that can be used by storage systems and applications, and providing patches for existing FTP tools, we also intend to build a number of prototype storage systems and applications that demonstrate the value of this common data transfer protocol.

- We will use the production libraries listed above to build several examples of custom storage managers. We will also use the libraries to build one or two storage system clients and use these to demonstrate interoperability.
- We will also use these clients and servers to conduct intensive performance testing experiments, demonstrating the performance benefits of the transfer protocol.

Finally, we will deploy the transfer protocol and replica management technologies into several user communities, including the Parallel Climate Model Data Intercomparison (PCMDI) community, the Next Generation Internet (NGI) Light Source community, the Particle Physics Data Grid community, and the Grid Portals community.

Our experiences with these application communities will validate the usefulness of our work in enabling advanced scientific discovery.