The Globus Striped GridFTP Framework and Server

Executive Summary

The GridFTP extensions to the File Transfer Protocol define a general-purpose mechanism for secure, reliable, high-performance data movement. These GridFTP extensions have been standardized in the Open Grid Forum (formerly the Global Grid Forum). Multiple implementations exist and interoperate in production on a daily basis.

The most commonly used of these implementations is the Globus GridFTP suite of tools, which we describe here. The Globus implementation provides a scriptable command line client, a highly extensible server, and a flexible set of development libraries for custom solutions. There is also a service similar to a job scheduler, called the Reliable File Transfer (RFT) service, to which you can submit your data movement "job" for it to manage on your behalf. So, what sets the Globus implementation apart?

The Globus implementation is secure: The Grid Security Infrastructure (GSI) is based on PKI/X.509 certificates and is supported out of the box. Kerberos support is also possible. There is a dynamically loadable authorization module that allows for site specific authorization of every operation. The data is not encrypted or integrity checked by default, due to the performance penalty, but simple options can enable both

The Globus implementation is fast: It employs multiple TCP streams to work around the problems common to trying to get good bandwidth in the wide area and it can use an entire cluster with a fast parallel file system as a single server. For example, a Globus striped server achieved speeds of 27.3 Gbit/s memory-to-memory and 17 Gbit/s disk-to-disk over a 60 millisecond round trip time, 30 Gbit/s network.

The Globus implementation is robust: Restart markers allow us to restart with the loss of only a few seconds of data. From those sites that report usage stats, we have 7000+ installations and average over 100,000 transfers per day, and have peaked at over 1 million. In one experiment, we showed that the server can support 1800 concurrent clients without excessive load.

The Globus implementation is extensible: Our Data Storage Interface (DSI) completely abstracts away the underlying storage. If you can implement the DSI, then you can put a GridFTP compliant server in front of your source of data. We currently have DSIs for POSIX, HPSS (tape archive), the Storage Resource Broker (SRB) and a prototype of one for doing space reservation via the Condor NeST storage utility. We also utilize a read, write, open, close abstraction called the Globus eXtensible IO (XIO) system that allows us to be transport protocol agnostic. This means that in environments where it makes sense protocols much more aggressive than TCP can be utilized. Finally, if what we provide does not meet your needs, we provide easy to use development libraries to provide you with the ultimate in extensibility.

Data sets are growing at exponential rates, network speeds are growing faster than Moore's law, multi-gigabit network connections are becoming ubiquitous, but most applications can not take advantage of the network to move their data. The combination of security, performance, and extensibility found in the Globus GridFTP suite of tools make it a good foundation on which to build tools and applications that can make taking advantage of fast networks to move big data sets a reality.

1 Features of the Globus GridFTP implementation

Third-party control of data transfer. To manage large datasets for distributed communities, authenticated third-party control of data transfers between storage servers is critical. A third-party operation allows a user or application at one site to initiate, monitor and control a data transfer operation between two other sites: the source and destination for the data transfer.

Authentication, data integrity, data confidentiality. GridFTP supports Generic Security Services (GSS)-API authentication of the control channel (RFC 2228) and data channel (GridFTP extensions), and supports user-controlled levels of data integrity and/or confidentiality. Data channel authentication is of particular importance in third party transfers since the IP address of the host connecting for the data channel will be different than that of the host connected on the control channel, and there must be some way to verify

that it is the intended party.

Striped data transfer. Data may be striped or interleaved across multiple servers, as in a parallel file system. Thus, GridFTP defines protocol extensions that support the transfer of data partitioned among multiple servers.

Parallel data transfer. On wide-area links, using multiple TCP streams in parallel between a single source and destination can improve aggregate bandwidth relative to that achieved by a single stream. Note that striping and parallelism may be used in tandem, i.e., you may have multiple TCP streams open between each of the multiple servers participating in a striped transfer.

Partial file transfer. Some applications can benefit from transferring portions of files rather than complete files: for example, analyses that require access to subsets of massive object-oriented database files. GridFTP supports requests for arbitrary file regions.

Support for reliable and restartable data transfer. Reliable transfer is important for many applications that manage data. Fault recovery methods are needed to handle failures such as transient network and server outages.

Usage Statistics. The Globus GridFTP server reports usage statistics back to the Globus Alliance. No identifying data is sent. A single UDP packet is sent at the end of each transfer listing operational data such as number of bytes, how long the transfer ran, how many streams and stripes, buffer size, etc.. This is on by default, but can be switched off via a number of mechanisms. Further information can be found here: http://www.globus.org/toolkit/docs/4.0/Usage_Stats.html

2 Globus Striped GridFTP Design

The Globus striped GridFTP system aims for (a) modularity, to facilitate the substitution of alternative mechanisms and use in different environments and configurations, and (b) efficiency, in particular the avoidance of data copies. We achieve these goals via an architecture that allows a protocol processing pipeline to be constructed by composing independent modules responsible for different functions.

The implementation (Figure 1) comprises three logically distinct components: *client* and *server protocol interpreters* (PIs), which handle the control channel protocol (these two functions are distinct because the protocol exchange is asymmetric), and the *data transfer process* (DTP), which handles the accessing of the actual data and its movement via the data channel protocol. These components can be combined in various ways to create servers with different capabilities. For example, combining the server PI and DTP components in one process creates a conventional FTP server, while a striped server might use one server PI on the head node of a cluster and a DTP on all other nodes.

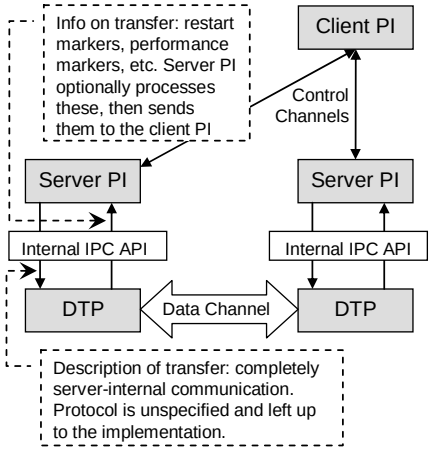

Figure 1: Globus GridFTP architecture

The DTP itself is further decomposed into a three-module pipeline (Figure 2). The *data access module* provides an interface to data source(s) and/or sink(s). The *data processing module* performs server-side data processing, if requested by an extended store/retrieve (ESTO/ERET) command. Finally, the *data channel protocol module* reads from, and/or writes to, the data channel. This basic structure allows for a wide variety of systems, from simple file server logic (data access module reads/writes files, data processing module does nothing, data channel protocol module writers/reads the data channel) to more complex and specialized behaviors (e.g., data module generates data dynamically in response to user requests).

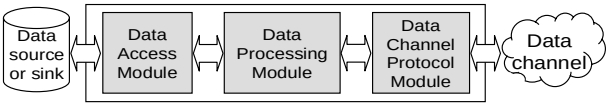

Figure 2: Globus GridFTP data transfer pipeline

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, by Los Alamos National Laboratory, by the National Science Foundation's Middleware Initiative, and by IBM. We gratefully acknowledge access to facilities provided by NCSA, SDSC, DOT, and ISI.