A GridFTP Transport Driver for Globus XIO

Rajkumar Kettimuthu[1,2], Liu Wantao[3,4], Joseph Link[5], and John Bresnahan[1,2,3]

[1]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL USA
[2]Computation Institute, The University of Chicago, Chicago, IL USA
[3]Department of Computer Science, The University of Chicago, Chicago, IL USA
[4]Beihang University, Beijing, China
[5]Globus Contributor, Addison, IL USA

Abstract

GridFTP is a high-performance, reliable data transfer protocol optimized for high-bandwidth wide-area networks. Based on the Internet FTP protocol, it defines extensions for high-performance operation and security. The Globus implementation of GridFTP provides a modular and extensible data transfer system architecture suitable for wide area and high-performance environments. GridFTP is the de facto standard in projects requiring secure, robust, high-speed bulk data transport. For example, the high energy physics community is basing its entire tiered data movement infrastructure for the Large Hadron Collider computing Grid on GridFTP; the Laser Interferometer Gravitational Wave Observatory routinely uses GridFTP to move 1 TB a day during production runs; and GridFTP is the recommended data transfer mechanism to maximize data transfer rates on the TeraGrid. Commonly used GridFTP clients include globus-url-copy, uberftp, and the Globus Reliable File Transfer service. In this paper, we present a Globus XIO based client to GridFTP that provides a posix-like (open/close/read/write) interface to the users. Such a client greatly eases the addition of GridFTP support to third-party programs, such as SRB and MPICH-G2. Further, this client provides an easier and familiar interface for applications to efficiently access remote files. We compare the performance of this client with that of globus-url-copy on multiple endpoints in the TeraGrid infrastructure. We perform both memory-to-memory and disk-to-disk transfers and show that the performance of this posix-like client is comparable to that of globus-url-copy. We also show that our GridFTP client significantly outperforms the GPFS WAN on the TeraGrid.

1. Introduction

Large-scale collaborative science applications necessitate efficient and secure transport of data across geographically dispersed locations. The GridFTP [1] extensions to the File Transfer Protocol [2] define a general-purpose mechanism for secure, reliable, high-performance data movement. The Globus implementation of GridFTP [3] provides a modular and extensible data transfer system architecture suitable for wide-area and high-performance environments. Key features of GridFTP include the following:

1. Performance: Typical GridFTP provides order of magnitude performance improvements compared to standard FTP. GridFTP achieves good performance by using non-TCP protocols such as UDT [4] and parallel streams to minimize bottlenecks inherent in TCP/IP.
2. Cluster-to-cluster data movement: GridFTP can do coordinated data transfer by using multiple computer nodes at the source and destination. This approach can increase performance by another order of magnitude.
3. Reliability: GridFTP provides support for reliable and restartable data transfers.
4. Multicasting: Globus GridFTP is capable of doing one-source-to-many-destination transfers.
5. Multiple Security options: The Globus GridFTP framework supports various security options, including Grid Security Infrastructure

(GSI) [5], anonymous access, username- and password-based security such as regular FTP servers, SSH-based [6] security, and Kerberos [7].
6. Modularity: The XIO-based [8] Globus GridFTP framework makes it easy to plug in other transport protocols. The Data Storage Interface (DSI) [9] allows for easier integration with various storage systems.
7. Third-party control: GridFTP also allows secure third-party clients to initiate transfers between remote sites.
8. Partial file transfer: Scientists often find it expedient to download only portions of a large file, instead of the entire file. GridFTP supports this capability by specifying the byte position in the file to begin the transfer.
9. Negotiation of TCP buffer/window sizes: GridFTP employs FTP command and data channel extensions to support both automatic and manual negotiation of TCP to get optimal performance.

In this paper, we present a Globus extensible input/output (XIO) [8] client to GridFTP that provides users with a posix-like (open/close/read/write) interface. Such a client provides an easier and familiar interface for applications to efficiently access remote files. Also, it eases the addition of GridFTP support to third-party programs, such as SRB [10] and MPICH-G2 [11]. In addition to the open/close/read/write interface, this client provides interfaces to use the features of GridFTP. It supports GridFTP features such as partial file transfer, parallel TCP streams, and connection caching.

The paper is organized as follows. Section 2 provides an overview of Globus XIO. Section 3 discusses the GridFTP driver for Globus XIO and explains in detail how to use the driver. Section 4 contains the experimental results. Section 5 summarizes the benefits of GridFTP XIO client.

2 Globus XIO

Globus XIO is the extensible input/0utput component of the Globus Toolkit® [12]. It is a framework that presents a single standard open/close/read/write interface to many protocol implementations. Globus XIO comprises two main components: framework and drivers. Figure 1 illustrates the architecture.

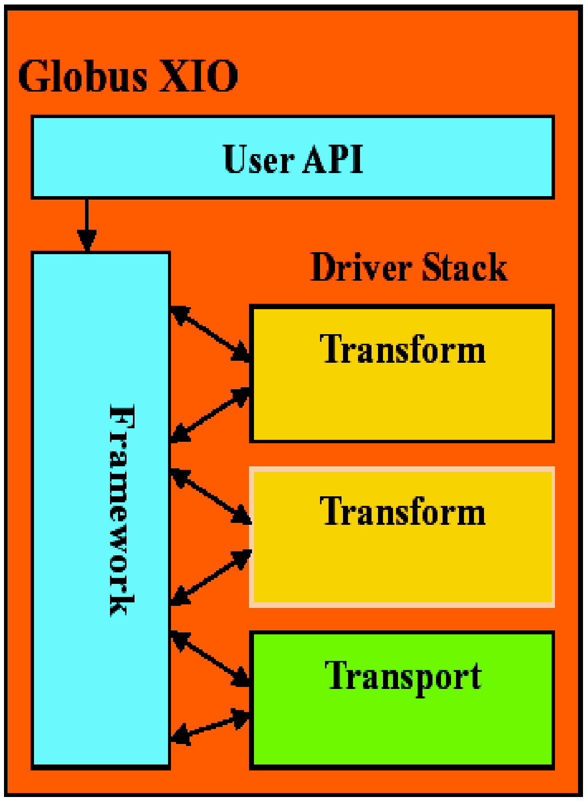

Figure 1: Globus XIO Architecture

2.1 Globus XIO Framework

The Globus XIO framework manages I/O operation requests that an application makes via the user API. The framework does not manipulate or deliver the data in an I/O operation; the drivers do all of that work. The framework's job is to manage requests and map them to the drivers' interface.

2.2 Drivers

A driver in Globus XIO is responsible for manipulating and transporting the user's data. There are two types of drivers: transform and transport. Transform drivers are those that manipulate the data buffers passed to it via the

user API and the XIO framework. Transport drivers are those that are capable of sending the data over a wire.
Drivers can be grouped into a stack. When an I/O operation is requested, the Globus XIO framework passes the operation request to every driver in the order the drivers are in the stack. When the bottom-level driver (the transport driver) finishes shipping the data, it passes the request completion notification back to the XIO framework. Globus XIO then delivers the notification back up the stack in this manner until it reaches the top, at which point the application is notified that its request is completed.
In a driver stack there must only be one transport driver, and it must be at the bottom of the stack. The reason is that the transport driver is what actually moves the bits on a wire. The protocols that use multiple transport drivers have to construct multiple stacks. For example, a protocol that uses TCP for exchanging control information and UDP for transferring the actual data needs two different stacks: one with TCP as a transport and the other with UDP as the transport driver. Any number of transform drivers can be in a stack.
Good examples of transform drivers are security wrappers and compression. Driver stacks can be mixed and matched. An HTTP driver has been created as a transform driver, and typically it sits in a stack directly above TCP. For example, if a driver is created by implementing UDT (a reliability layer over UDP), a stack can be formed of HTTP on top of UDT without a single code change to any of the drivers involved. A protocol may provide some special features that other protocols may not support. Globus XIO provides a way for the protocol developers to expose the functionalities that are specific to their protocols.
Each driver may have its own attribute structure. Globus XIO provides user API to control the attributes of a driver. It gives the user an opportunity to tweak parameters that are specific to a driver. The attribute support is optional; a driver may choose to have no attribute support.
Globus XIO has no extra memory copies. The user passes data in a buffer to the framework. The framework then passes a pointer to that buffer down the driver stack. If any driver along the way needs to insert a header or append a footer, it can use the readv/writev functions that XIO provides. If a driver needs to alter the data (as a compression driver would), then a copy is required. However, this is a mandatory copy to achieve the desired functionality; it is not an extra copy.

3 GridFTP Driver

A GridFTP driver for Globus XIO was created to provide a posix-like client interface to GridFTP servers. It can work with any FTP server. The GridFTP driver uses the Globus FTP client library to communicate with GridFTP or any other FTP server.
The Globus FTP client library provides high-level commands that implement the protocol without requiring the developer to have an in-depth knowledge of the protocol.
In the following section, we provide step-by-step instructions and code snippets to demonstrate the use of the user API. Globus XIO is a C library. Since the core API is simply Open/Close/Read/Write, the user API is fairly straightforward.
We provide synchronous calls (globus_xio_read()) and asynchronous calls (globus_xio_register_read()) and have vector variants of each (globus_xio_[register] _readv()). Two important XIO data structures must be considered when using a transport driver:
1. Handle – this is returned to the user once the XIO framework has all the information needed to open a new connection. It is then used to reference the connection on all future I/O calls.
2. Attribute – in order to set driver-specific parameters, a custom attribute structure can be used. For the GridFTP driver, number of TCP

streams, TCP buffer size etc., are supplied in this way.

Step 1: Activate Globus
The first step is to activate the Globus module. Until activation is complete, no XIO function calls can be successfully executed. The module is activated with the following line:
globus_module_activate(GLOBUS_XIO_MODULE);

Step 2: Load Driver
The next step is to load all the GridFTP driver. The function globus_xio_load_driver() is used to load a driver.
globus_result_t res;
globus_xio_driver t driver;
res = globus_xio_load_driver(&driver, "gridftp");
If, upon completion of the above function call, the variable "res" is equal to GLOBUS_SUCCESS, then the driver was successfully loaded and can be referenced with the variable "driver."

Step 3: Create Stack
Once globus xio is activated and a driver loaded, a driver stack must be built. In this case, the stack consists of only one driver, the GridFTP driver. The stack is established with the following code (building from the above code snips):
globus_xio_stack_t stack;
globus_xio_stack_init(&stack);
globus_xio_stack_push_driver(stack, driver);

Step 4: Opening the Handle
Once the stack is created, a handle to the GridFTP server can be opened. The following code illustrates this step:
globus_xio_handle_t handle;
globus_xio_handle_create(&handle, stack);
res = globus_xio_open(handle, contact_string, attribute);
This establishes connection with the gridftp server. The contact string must contain the scheme, host name, and the resource (path to the file). Optionally, it may also contain the port and subject. The format of the contact string is as follows:
<scheme> "://" location ["/" [<path to resource>]]
scheme:
gsiftp | ftp
location:
[auth-part] host-part
auth-part:
<user> [":" <password>] "@"
host-part:
["<" <subject> ">"] host-name [":" <port or service>]
host-name:
<hostname> | <dotted quad> | "[" <ipv6 address> "]"
The attribute is used to set driver-specific features such as the number of TCP streams, TCP buffer size, or partial transfer.

Step 5: Reading/Writing on the Handle
With an open handle to a GridFTP server, one can read or write data to it with either globus_xio_read() or globus_xio_write().

Step 6: Closing the Handle
After all I/O operations on the handle have been performed, the final step is to call globus_xio _close(handle).

In addition, the GridFTP driver provides a control interface to perform seek operation.
res = globus_xio_handle_cntl(xio_handle, GLOBUS_ XIO_GRIDFTP_SEEK, offset);
Seek is always started from the beginning of the file.

4 Experimental Results

We compared the performance of the GridFTP XIO client with that of globus-url-copy, a commonly used GridFTP client. We transferred files of various sizes ranging from 1 MB to 1 GB between two pairs of sites on the TeraGrid. Figure 2 shows the result of the transfers between Argonne National Laboratory and the National Center for Supercomputing Applications, a network with a round-trip time of 4 ms. Figure 3 shows the result of the transfers between Oak Ridge National Laboratory and San Diego Supercomputing Center. Each experiment was run 10 times, and the average value was used to compare the performance. The results show that GridFTP XIO client provides almost the same throughput as globus-url-copy; no additional overhead is introduced.

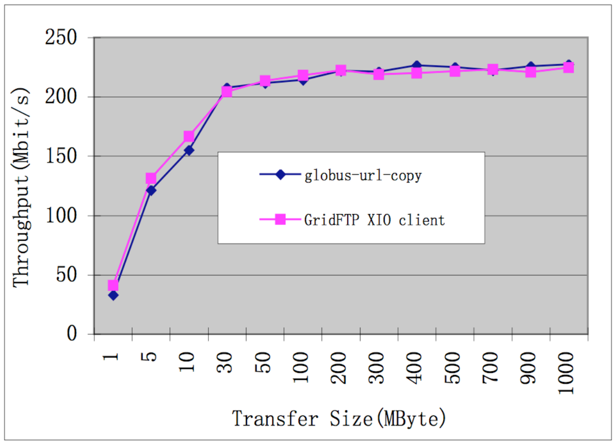

Figure 2: Comparison of performance of the GridFTP XIO client with globus-url-copy over a TeraGrid network between Argonne and NCSA (4 ms round-trip time)

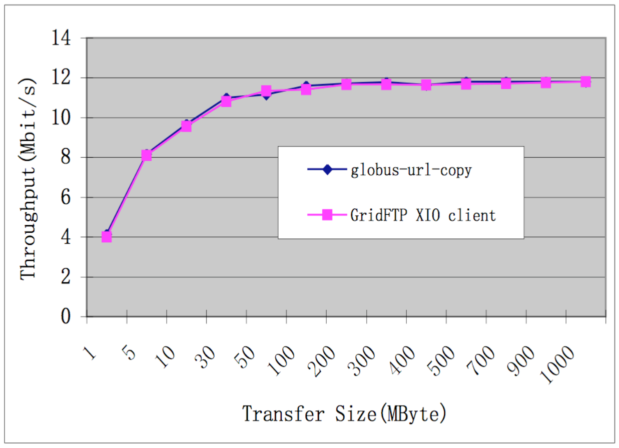

Figure 3: Comparison of performance of the XIO client with globus-url-copy over a TeraGrid network between ORNL and SDSC (72 ms round-trip time)

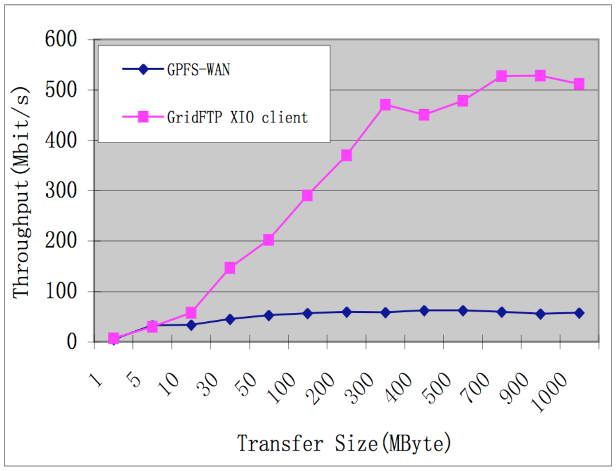

Figure 4: Comparison of performance of the XIO client with GPFS WAN mounted at Argonne

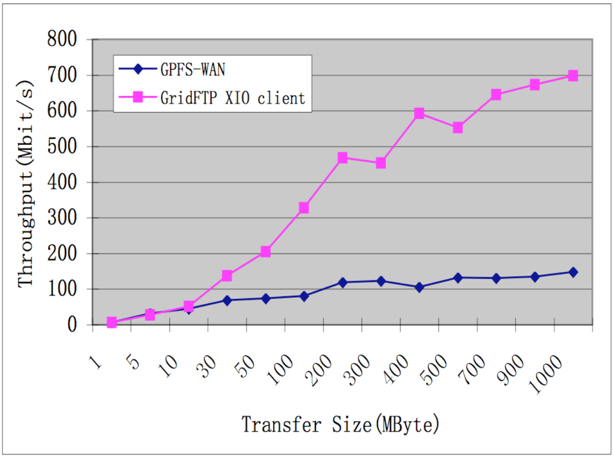

Figure 5: Comparison of performance of the GridFTP XIO client with the GPFS WAN mounted at Indiana University

In the next set of experiments, we compared the performance of the XIO client with that of GPFS WAN. On the TeraGrid infrastructure, the GPFS WAN is physically located at SDSC and is mounted at Argonne, NCSA, and Indiana University. To measure the performance of the GPFS WAN, we used the “cp” command to copy files of various sizes (ranging from 1 MB to 1 GB) from a local disk at Argonne to the GPFS WAN mounted at Argonne. We did a similar measurement at Indiana University by copying from a local disk at Indiana to the GPFS WAN mounted at Indiana. To make a fair comparison, we performed wide-area transfers of the (same) files on the local disks at Argonne and Indiana to the GPFS (not the GPFS WAN) file system at SDSC, using the GridFTP XIO client. We ran the GridFTP server at SDSC and the XIO client at Argonne and Indiana. We can see that for small files, the GPFS-WAN and GridFTP XIO client have similar performance, while for larger files, the GridFTP XIO client performs dramatically better than GPFS-WAN. Figure 4 compares the performance of file copies from Argonne to SDSC through GridFTP XIO client with that of the GPFS WAN mounted at Argonne. Figure 5 compares the performance of file copies from Indiana to SDSC through GridFTP XIO client with that of the GPFS WAN mounted at Indiana.

5 Summary

The GridFTP XIO client provides a simple and familiar open/close/read/write interface for applications to access files from a remote GridFTP server. The performance of this client is comparable to globus-url-copy, a commonly used command line client for GridFTP. Thus, remote files can be accessed efficiently using this tool. The access time is significantly faster than that of the GPFS WAN. This client also makes it easier to add GridFTP support to third-party programs. Overall, this tool provides value to TeraGrid and other Grid users by making remote file access efficient and easy.

Acknowledgments

This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357, and in part by National Science Foundation's CDIGS.

References

[1] W. Allcock, "GridFTP: Protocol Extensions to FTP for the Grid," Global Grid Forum GFD-R-P.020, 2003.
[2] J. Postel and J. Reynolds, "File Transfer Protocol," IETF, RFC 959, 1985.
[3] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus Striped GridFTP Framework and Server, SC'05," ACM Press, 2005.
[4] Y. Gu and R. L. Grossman, "UDT: UDP-based Data Transfer for High-Speed Wide Area Networks," Comput. Networks 51, no. 7 (May 2007), 1777–1799.
[5] www.globus.org/security/overview.html
[6] T. Ylonen and C. Lonvick, eds., "The Secure Shell (SSH) Authentication Protocol," IETF, RFC 4252, 2006
[7] http://web.mit.edu/Kerberos/
[8] W. Allcock, J. Bresnahan, R. Kettimuthu, and J. Link, "The Globus eXtensible Input/Output System (XIO): A Protocol Independent IO System for the Grid," in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium - Workshop 4, Vol. 5, IEEE Computer Society, Washington, DC, 2005. 179.1. DOI= http://dx.doi.org/10.1109/IPDPS.2005.429
[9] R. Kettimuthu, M. Link, J. Bresnahan, and W. Allcock, "Globus Data Storage Interface (DSI) – Enabling Easy Access to Grid Datasets," First DIALOGUE Workshop: Applications-Driven Issues in Data Grids, Aug. 2005.
[10] A. Rajasekar, M. Wan, R. Moore, W. Schroeder, G. Kremenek, A. Jagatheesan, C. Cowart, B. Zhu, S. Chen, and R. Olschanowsky, "Storage Resource Broker - Managing Distributed Data in a Grid," Computer Society of India Journal, Special Issue on SAN, 33, no. 4 (2003), 42–54.
[11] N. T. Karonis, B. Toonen, and I. Foster, "MPICH-G2: A Grid-enabled Implementation of the Message Passing Interface," Journal of Parallel and Distributed Comput. 63, no. 5 (May 2003), 551–563. DOI= http://dx.doi.org/10.1016/S0743-7315(03)00002-9.
[12] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," International Journal of Supercomputer Applications 11, no. 2 (1997), 115–128. 1997.