Globus GridFTP: What's New in 2007 (Invited Paper)

John Bresnahan[1,2,3], Michael Link[1,2], Gaurav Khanna[4], Zulfikar Imani[3],
Rajkumar Kettimuthu[1,2] and Ian Foster[1,2,3]

[1]Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

[2]Computation Institute, University of Chicago, Chicago, IL 60637

[3]Department of Computer Science, University of Chicago, Chicago, IL 60637

[4]Department of Computer Science & Engineering, Ohio State University, Columbus, OH 43210

ABSTRACT

GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks. It is based on the Internet FTP protocol, and it defines extensions for high-performance operation and security. The Globus implementation of GridFTP provides a software suite optimized for the gamut of data access issues—from bulk file transfer to the details of getting data out of complex storage systems within sites. We summarize some recent developments in Globus GridFTP.

Keywords

GridFTP, LOSF, Split DSI, GridFTP over UDT, GWFTP

1. INTRODUCTION

Science is increasingly data-driven. Geographically distributed communities of scientists need to access and analyze large amounts of data, both in simulation science applications such as climate modeling and in experimental science applications such as high-energy physics.

Transferring terabytes or even petabytes of data over wide-area networks at speeds of multigigabit per second necessitates the use of advanced network transport and provisioning technologies to dynamically provision, configure and control, and monitor the wide-area networking infrastructure. In this short paper, we discuss new functionalities and recent developments that we have incorporated into the Globus GridFTP framework [1] to take advantage of the latest network technologies and transport protocols. Specifically, we describe the following six initiatives.

1. *GridFTP Pipelining,* which improves the performance of lots of small files transfers
2. *GridFTP over UDT*, which can provide significantly higher end-to-end performance than GridFTP over TCP, particularly on wide area networks
3. *Split DSI for GridFTP,* which can help overcome some of the TCP limitations and can also help to avoid some bottleneck links.
4. *GridFTP Where there is FTP* (GWFTP), which allows the use of ordinary FTP clients to invoke operations on GridFTP servers
5. *Network provisioning*, which allows for binding of GridFTP transfers to optical paths
6. GridFTP over Infiniband, which enables GridFTP to take advantage of the recent developments in Infiniband over SONET to achieve high performance on wide area networks.

Because of lack of space, we do not discuss three other initiatives that should also be of interest to many: the ability to dynamically add or remove data nodes from a striped GridFTP server, which helps to improve resource utilization and to be robust in the presence of data node failures; SSH authentication, for environments where Grid Security Infrastructure is not supported or required; and space and bandwidth management, to increase the reliability of transfers.

2. GridFTP Pipelining:

Given resources, the GridFTP implementation provided by the Globus Toolkit can scale to network speeds and has been shown to deliver 27 Gb/s on 30 Gb/s links. The protocol is optimized to transfer large volumes of data commonly found in Grid applications. Datasets of sizes from hundreds of megabytes to terabytes and beyond can be transferred at close to network speeds by using GridFTP.

Unfortunately, conventional implementations of GridFTP have a limitation as to how the data must be partitioned to reach these high-throughput levels. Not only must the amount of data to transfer be large enough to allow TCP to reach full throttle, but the data must also be in large files, ideally in one single file. If the dataset is large but partitioned into many small files (on gigabit networks we consider any file smaller than 100 MB as a small file), the performance of GridFTP servers suffers drastically

This problem is known as the "lots of small files" (LOSF) problem. In this paper we study the LOSF problem and present a solution known as *pipelining*.

GridNets 2007, October 17–19, 2007, Lyon, France.

The GridFTP protocol is a backward-compatible extension of the legacy RFC959 FTP protocol. It maintains the same command/response semantics introduced by RFC959. File transfer requests are done with the RETR (send) or STOR (receive) command. A client sends one of these commands to the server across the control channel. Data then begins to flow between the client and server over the data channel. Once all of the data has been transferred, a "226 Transfer Complete" acknowledgment message is sent from the server to the client on the control channel. Only when this acknowledgment is received can the client request another transfer. During this time the data channel is idle. The latency between transfers adds to the overall transfer time and thus detracts from the overall throughput.

Pipelining approaches the LOSF problem by trying to minimize the amount of time between transfers. Pipelining allows the client to have many outstanding, unacknowledged transfer commands at once. Instead of being forced to wait for the "226 Transfer Successful" message; the client is free to send transfer commands at any time. The server processes these requests in the order they are sent. Acknowledgments are returned to the client in the same order. As can be seen from Figure 1, pipelining improves the throughput of lots of small files transfers significantly.

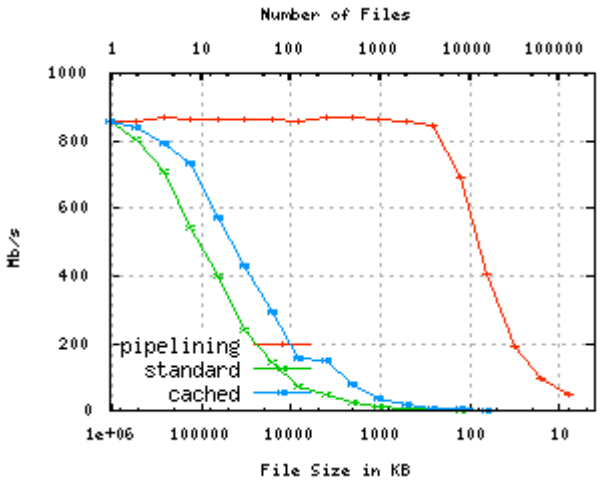

Figure 1: Comparison of the performance of pipelined GridFTP transfers with standard (nonpipelined) GridFTP transfers in a WAN

3. GridFTP over UDT

UDT [2] is an application-level data transport protocol that uses UDP to transfer bulk data, while implementing its own reliability and congestion control mechanisms. UDT achieves good performance on high-bandwidth, high-delay networks in which TCP has significant limitations.

GridFTP uses the Globus Extensible Input Output (XIO) [3] interface to invoke network I/O operations. The Globus XIO framework presents a single, standard open/close read/write interface to many different protocol implementations, including TCP, UDP, HTTP, and file—and now UDT. The protocol implementations are called drivers. Once created, a driver can be dynamically loaded and stacked by any Globus XIO application.

In creating a Globus XIO driver for UDT, we took advantage of a recent addition to the Globus XIO system, the *wrapblock* feature. The standard Globus XIO driver interface supports an asynchronous interaction model. While this is more scalable and efficient than a synchronous model, however, it is also the most difficult to code against. Further, many protocol implementations do not have asynchronous APIs, and transforming them into an asynchronous model can be time consuming. The wrapblock functionality uses thread pooling and event callback techniques to transform the asynchronous interface into a blocking interface. This approach makes the task of creating a driver from an existing library a trivial task. For example, the Globus XIO driver for UDT is less than 700 lines of code.

Table 1: Throughput achieved when transferring 1 GB of data over two wide-area networks, using various mechanisms

Testbed / Transport Mechanism	Argonne to New Zealand Throughput in Mbit/s	Argonne to Los Angeles Throughput in Mbit/s
Iperf – 1 stream	19.7	74.5
Iperf – 8 streams	40.3	117.0
GridFTP mem TCP – 1 stream	16.4	63.8
GridFTP mem TCP – 8 streams	40.2	112.6
GridFTP disk TCP – 1 stream	16.3	59.6
GridFTP disk TCP – 8 streams	37.4	102.4
GridFTP mem UDT	179.3	396.6
GridFTP disk UDT	178.6	428.3
UDT mem	201.6	432.5
UDT disk	162.5	230.0

Preliminary results show that GridFTP over UDT significantly outperforms GridFTP over TCP on networks where throughput is the bottleneck. Table 1 compares the performance of Iperf, GridFTP over TCP, GridFTP over UDT, and raw UDT on two different networks—a wide-area network between Argonne and the University of Auckland, New Zealand, with a round-trip time of 204 ms; and a second wide-area network between Argonne and Los Angeles, with a round-trip time of 60 ms.

3. Split DSI

By default, GridFTP employ TCP as the underlying transport protocol. The observed throughput of GridFTP transfers in a wide-area environment is often lower than the achievable throughput, owing to the slow-start and congestion control mechanisms of TCP. The technique of dividing a TCP connection into a set of shorter, better performing connections by splitting it at multiple intermediate points with the goal of improving the overall throughput has been widely studied [4], [5], [6], [7]. A split-TCP connection is expected to perform better than a single end-to-end TCP connection owing to several reasons. Firstly, the round-trip time on each intermediate hop is shorter as compared to the direct end-to-end path. Therefore, the congestion control mechanism of TCP would sense the maximum throughput quickly thereby attaining steady state where in it will give maximal possible throughput until a congestion event occurs. Secondly, any packet loss is not propagated all the way back to the source but only to the previous intermediate hop. In addition, if the bandwidth on each of the intermediate hops is higher as compared to the direct path, the overall throughput is expected to improve.

GridFTP has a Data Storage Interface (DSI), which interacts with the storage system. The DSI layer accepts requests like the get, put and stat and performs the necessary functions based on the underlying storage system it interfaces with. Rizk et. al. [4] have implemented a DSI interface to achieve the split –TCP functionality. Their implementation lets a GridFTP client specify a multi-hop transfer between a source and a destination URL through a series of intermediate hosts by employing split URLs. A split URL is essentially a concatenation of multiple normal URLs. For example, a globus-url-copy command issued with a source URL A/B and the destination URL C/D means that the file will be transferred from A to D via B and C. We have made a number of improvements to the aforesaid DSI implementation. In the previous implementation, a GridFTP server could either act as an end server or as an intermediate server, which is very restrictive from the point of view of production-use GridFTP servers. We have generalized it by allowing the DSI to perform different functions based on the input it receives. For example, the DSI *get* could either directly contact the underlying storage or forward the data to another GridFTP server. In addition, we have also implemented FTP operations like MLSD, MKDIR, RMDIR etc... for the intermediate servers. Furthermore, the previous implementation worked only in the non-secure mode. We have incorporated the necessary changes to make the split-TCP work with GSI security mechanisms. We are working on the algorithm to determine the optimal path.

4. GWFTP

GridFTP has existed for some time and has proven quite robust and usable. However, there are few available GridFTP clients. FTP-959, on the other hand, has innumerable clients. To leverage these clients, we have created an intermediate program, called GWFTP, to act as a proxy between existing FTP clients and GridFTP servers. Users can connect to GWFTP with their favorite standard FTP client, and GWFTP will then connect to a GridFTP server on the client's behalf. To clients, GWFTP looks much like an FTP proxy server. When wishing to contact a GridFTP server, FTP clients instead contact GWTFTP. Clients tell GWFTP their ultimate destination via the FTP USER <username> command. Instead of entering their username, client users send the following.

USER <GWTFTP username>::<GridFTP server URL>

This command tells GWTFTP the GridFTP endpoint with which the client wants to communicate. For example:

USER bresnaha::gsiftp://wiggum.mcs.anl.gov:2811/

An active GSI credential is needed before running GWFTP. GWFTP uses the GSI credential to authenticate with the GridFTP server. Two security options are provided with GWFTP to authenticate its clients. By using the -pw or --pwfile option one can have standard password-based authentication with GWFTP. By using -ah or --authorized-hosts option with a comma-separated list of authorized IP addresses, one can limit what hosts can freely connect to one's GWFTP proxy.

5. Network Provisioning for Data-Intensive Scientific Computing

Significant advances have been made in the area of network provisioning. Optical communication links that can provide extremely high bandwidths over thousands of miles are becoming increasingly common. Such network provisioning technologies need to be integrated into a scalable architecture that can provide on-demand setup of channels at varying bandwidth resolutions. A number of projects have been involved in the deployment of optical networking technology over a wide area. Examples include National Lambda Rail [8], UKLight [9], and the

DOE Science Ultranet [10]. However, all of these projects focus exclusively on the wide-area networking infrastructure. The issue of integrating production-use computing and storage facilities and advanced wide-area networks has not been adequately addressed. In other words, networking services are needed that can mediate the flow of traffic within a site's local infrastructure and selectively forward it onto high-bandwidth advanced networks. This requirement is termed the "last mile" problem in the wide-area networking context. Projects that have focused on this problem include LambdaStation [11] and Terapaths [12].

To be able to leverage the functionality of a service such as Terapaths in conjunction with GridFTP, we need to invoke the Terapaths API from Globus and set up a high-bandwidth path before invoking GridFTP to perform the transfer. Since the Terapaths API is accessible via a Web service over SOAP, we are developing a high-level service that leverages the Globus C-Web Services core software to invoke the Terapaths Web Services to reserve a quality-of-service path before initiating a GridFTP transfer. Since the end-to-end data movement involves multiple resources including non-network resources such as compute and storage resources, GARA [13] architecture that allows co-reservation of multiple heterogeneous resources would be of great help. We are exploring ways to use such an architecture to provide better end-to-end service.

6. GridFTP over Infiniband

InfiniBand (IB) defines a point-to-point interconnect architecture that leverages networking principles—switching and routing—to provide a scalable, high-performance server I/O fabric. InfiniBand provides transport services for upper-layer protocols and supports flow control and quality of service to provide ordered, guaranteed packet delivery across the fabric. The InfiniBand Host Channel Adapter (HCA, which provides an interface to a host device and supports all software verbs defined by InfiniBand) provides RDMA support, which offloads data movement, as well as the multiplexing and demultiplexing of different stream from the CPU into the HCA. Infiniband has been shown to be effective for data transport in storage networks (over a few miles). Recent experiments conducted at Oak Ridge National Laboratory [14] shows that Infiniband can be effective over a wide area with specialized hardware. With the Obsidian longbow Infiniband switch, 7.2 Gbit/s throughput was achieved on an 8600-mile, 8 Gbit/s loop with IB over SONET.

GridFTP can use Infiniband through Sockets Direct Protocol (SDP) [15] with little or no code change to GridFTP. SDP provides a standard sockets interface for the Open Fabrics Enterprise Distribution [16], a unified, open-source software stack for the two major RDMA fabric technologies: InfiniBand and iWARP (also known as RDMA over Ethernet). The disadvantage of this method is that since the entire SDP implementation is in the kernel, it does not leverage the kernel bypass capability of the InfiniBand architecture. User-level verbs allow applications to interface directly with the InfiniBand hardware. By developing a Globus XIO driver for the verbs interface, the RDMA capabilities can be used efficiently. We are working to develop such a driver.

7. Conclusion

We have summarized recent developments in the Globus GridFTP framework that take advantage of the latest developments in networking infrastructure and transport protocols to better serve scientific communities. We have also introduced functionality that allows clients to use any FTP client to invoke data transfers with a GridFTP server.

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 SciDAC-2 CEDPS.

References

[1] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, "The Globus striped GridFTP framework and server," in SC'05, ACM Press, 2005.

[2] Gu, Y. and Grossman, R. L. 2007. UDT: UDP-based data transfer for high-speed wide area networks. Comput. Networks 51, 7 (May. 2007), 1777–1799. DOI= http://dx.doi.org/10.1016/j.comnet.2006.11.009

[3] Allcock, W., Bresnahan, J., Kettimuthu, R., and Link, J. 2005. 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 (Ipdps'05) - Workshop 4 - Volume 05 (April 04 - 08, 2005). IPDPS. IEEE Computer Society, Washington, DC, 179.1. DOI= http://dx.doi.org/10.1109/IPDPS.2005.429

[4] A. Bakre and B. R. Badrinath. I-tcp: indirect tcp for mobile hosts. In ICDCS '95: Proceedings of the 15th International Conference on Distributed Computing Systems, page 136, Washington, DC, USA, 1995. IEEE Computer Society.

[5] M. Beck, T. Moore, J. S. Plank, and M. Swany. Logistical networking: Sharing more than the wires. In C.

A. Lee S. Hariri and C. S. Raghavendra, editors, Active Middleware Services, Norwell, MA, 2000. Kluwer Academic.

[6] Kevin Brown and Suresh Singh. M-tcp: Tcp for mobile cellular networks. SIGCOMM Comput. Commun. Rev., 27(5):19–43, 1997.

[7] C. Kiddle P. Rizk and R. Simmonds. A GridFTP overlay network service. In In Proceedings of the 7th IEEE/ACM International Conference on Grid Computing, Barcelona, Spain, 2007.

[8] http://www.nlr.net/

[9] http://www.uklight.ac.uk/

[10] http://www.csm.ornl.gov/ultranet/topology.html

[11] http://www.lambdastation.org/

[12] http://www.atlasgrid.bnl.gov/terapaths/

[13] I. Foster, M. Fidler, A. Roy, V, Sander, L. Winkler, "End-to-end quality of service for high-end applications," Computer Communications, 27(14):1375–1388, 2004

[14] www.es.net/hypertext/RD-Workshop-April2007/Rao.ppt

[15] Sockets Direct Protocol. http://www.infinibandta.com

[16] http://www.openib.org/

\