Globus XIO Pipe Open Driver: Enabling GridFTP to Leverage Standard Unix Tools

Rajkumar Kettimuthu,[1] Steven Link,[2] John Bresnahan,[1] Michael Link,[1] Ian Foster[1,3]

[1]Computation Institute
Argonne National Lab. & U.Chicago
Argonne, IL 60637

[2]Department of Computer Science
Northern Illinois University
DeKalb, IL 60115

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

ABSTRACT

Scientific research creates substantially large volumes of data throughout the processes of discovery and analysis. Given the necessity for data sharing and data relocation, members of the scientific community are often faced with a productivity loss that correlates with the time cost incurred during the data transfer process. The GridFTP protocol was developed to improve this situation by addressing the performance, reliability, and security limitations of standard FTP and other commonly used data movement tools such as SCP. The Globus implementation of GridFTP is widely used to rapidly and reliably move data between geographically distributed systems. Traditionally, GridFTP performs well for datasets containing large files. When the data is partitioned into many small files, however, it suffers from lower transfer rates. Although the pipelining and concurrency solution in GridFTP provides improved transfer rates for datasets using lots-of-small-files, these solutions cannot be applied in environments that have strict firewall rules. In some cases, tarring the files in a dataset on the fly will help; in other cases, a checksum of the files after they are written to disk is desired. In this paper, we present the Globus XIO Pipe Open Driver which enables GridFTP to leverage the standard Unix tools to perform these tasks. We demonstrate the effectiveness of this functionality through several experiments.

CATEGORIES

H.3.4 [**Systems and Software**].

GENERAL TERMS

Pipe, Checksum, Bulk Data Movement, Data Transfer, Tar Stream

TeraGrid '11, July 18-21, 2011, Salt Lake City, Utah, USA.

1. INTRODUCTION

Global-scale science requires the ability to share and use an ever-increasing range and volume of data from geographically distributed sources. Rapid increases in the raw capacity of the science networks makes it feasible to move large volumes of data across wide area networks. In practice, however, rapid, efficient, and robust wide-area, end-to-end transport is technically challenging. Globus GridFTP [1] implements the GridFTP extensions [2] to the File Transfer Protocol (FTP) [3], which provide support for parallel data movement, failure detection, and other features. Globus GridFTP is widely deployed and used on well-connected Grid [4,5] environments such as TeraGrid [6] because of its ability to scale to network speeds. However, when the data is partitioned into many small files instead of fewer large files, it suffers from lower transfer rates. The latency between the serialized transfer requests of each file directly lowers the achievable throughput. Pipelining [7] allows many transfer requests to be sent to the server before any one completes. It hides the latency of each transfer request by sending the requests while a data transfer is in progress. The concurrency [8,9] solution addresses this same throughput issue by opening up multiple transfer sessions and transferring multiple files concurrently. However, both pipelining and concurrency cannot be applied in environments that have strict firewall rules. In such scenarios, tarring the files in a dataset on the fly will help overcome such restraints. In many cases, users want to verify the integrity of the data by doing a checksum after the data has been written to disk. In this paper, we present the Globus XIO [10] Pipe Open Driver (Popen), which enables GridFTP to leverage standard Unix tools to perform such tasks. Specifically, we compare the performance of on-the fly tarring and untarring of files with that of pipelining and concurrency. Additionally, we compare the performance of checksum via Popen with that of the legacy checksum feature in GridFTP. The rest of the paper is organized as follows. Section 2 provides background on GridFTP and Globus XIO. Section 3 describes the Globus XIO Pipe Open Driver. Section 4 describes the use cases of Popen including SSH GridFTP, on-the-fly tar, and checksum. Section 5 presents the experimental results. Section 6 briefly discusses some observations about our approach.

2. BACKGROUND

In this section we provide details on GridFTP and the Globus eXtensible Input/Output (XIO) framework.

2.1 GridFTP

The GridFTP protocol is a backward-compatible extension of the legacy RFC959 FTP protocol. It maintains the same command/response semantics introduced by RFC959. It also maintains the two-channel protocol semantics. One channel is for control messaging (the control channel), such as requesting what files to transfer and the other is for streaming the data payload (the data channel). Once a client successfully forms a control channel with a server, it can begin sending commands to the server. In order to transfer a file, the client must first establish a data channel. This task involves sending the server a series of commands on the control channel describing attributes of the desired data channel. Once these commands are successfully sent, a client can request a file transfer. At this point a separate data channel connection is formed using all the agreed-upon attributes, and the requested file is sent across it.

In standard FTP, the data channel can be used to transfer only a single file. Subsequent transfers must repeat the data channel setup process. GridFTP modifies this part of the protocol to allow many files to be transferred across a single data channel. This enhancement is known as data channel caching. GridFTP also introduces other enhancements to improve performance over the standard FTP mode. For example, parallelism and striping allow data to be sent over several independent data connections and reassembled at the destination. These enhancements require the use of the extended block mode (MODE E) of GridFTP. In this mode, data channels must go from sender to receiver. GridFTP servers are typically configured to listen on one port for the control channel, and to use a configurable port range for data channel connections. Firewalls have to be configured accordingly.

Globus GridFTP is widely used to move large volumes of data over the wide area network. The XIO-based Globus GridFTP framework makes it easy to plug in other transport protocols. The Data Storage Interface [11] allows for easier integration with various storage systems. It supports non-TCP-based [12] protocols such as UDT [13,14] and RDMA [15]. It also provides advanced capabilities such as multilinking [16] and transfer resource management [17].

2.2 Globus XIO

XIO is an extensible and flexible I/O library written for use with the Globus Toolkit. XIO is written in C programming language and provides us with one API that currently supports many different wire protocols. All implementations of these protocols are encapsulated as drivers that are modular.

GridFTP uses the XIO interface for network and disk I/O operations. The XIO framework presents a single, standard open/close, read/write interface to many different protocol implementations. The protocol implementations, called drivers, are responsible for manipulating and transporting the user's data. Drivers are grouped into a stack. When an I/O operation is requested, the XIO framework passes the operation request down the driver stack. An XIO driver can be thought of as a modular protocol interpreter that can be plugged into an I/O stack without concern about the application using it. This modular abstraction is what allowed us to achieve our success here without disturbing the application's tested code base and without forcing endpoints to run new and unfamiliar code.

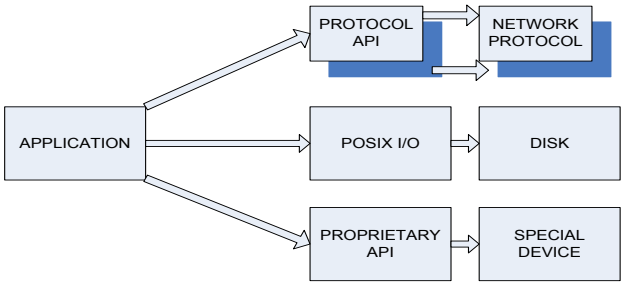

Figure 1. Typical application interaction with various devices.

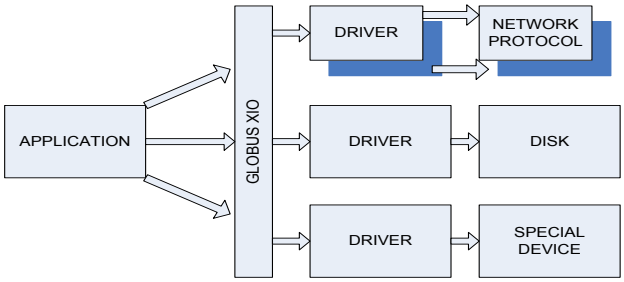

Figure 2. Application interaction with various devices via Globus XIO.

3. GLOBUS XIO POPEN DRIVER

Combining multiple tools to accomplish complex tasks with ease is not a new concept. In 1964—long before the advent of Unix, Doug McIlroy described pipes. Unix pipes are commonly used to construct powerful Unix command lines by combining Unix commands in one line and pass data from one to the next. The data is processed by each command and then passed on to the next command. The Globus XIO Pipe Open Driver is designed

to allow GridFTP clients to use pipe to combine GridFTP with other Unix tools even on the remote GridFTP server.

Figure 3 shows the configuration. The "Client" box represents client logic. The "Server" box represents the GridFTP server logic. The "Data" and "Popen" boxes represent Globus XIO drivers. The Data driver handles the network interactions for the GridFTP server. The Popen driver provides piping capability by allowing the GridFTP client to replace the "File" driver that handles the file system interactions with the Popen driver, on the GridFTP server's disk I/O stack. As the data passes through the Popen driver, it gets piped to the Unix command that the client provides; the output of that command gets written to the disk. When the data is being read from the server, the data read from the disk is passed through the Unix command before it gets written to the network. This approach is minimally invasive to the tested and robust GridFTP server.

The Popen driver allows users to access the standard I/O of existing programs by opening pipes to standard I/O. This approach provides the same functionality expected with Unix pipes, yet the user doesn't need to worry about what exactly is happening with the pipes. Essentially, the user can execute commands to programs allowing for instance, a directory of files on a remote server to have a checksum computed using /bin/md5sum and the result sent to the standard input of the initiating host.

Arguably, execution of arbitrary programs as part of a data transfer opens a potential security risk. For this behavior to be allowed on the server, all programs that the user might execute using the Popen driver, as well as the Popen driver itself must be explicitly added to the whitelist at the time the GridFTP server is run. A server permits execution of programs only on its Popen whitelist. If a client requests a program to be run that is not on the whitelist, the transfer fails. Following is the command necessary to enable Popen and tar when running a GridFTP server:

```
globus-gridftp-server
-fs-whitelist popen,file,ordering
-popen-whitelist tar:/bin/tar
```

Here we see the standard command for running a GridFTP server, followed by the first whitelist command with three arguments, popen, file and ordering indicating the drivers allowed on the disk stack. We load the Popen module which gives us access to the Popen driver functionality; the file driver, this allows us to conduct non-Popen operations (regular file system interactions); and the ordering driver ensures that the data being sent to the pipe is in order (often, GridFTP data streams are not in order). Next we see the popen whitelist: a comma-separated list of programs that the Popen driver is allowed to execute—in our example above we see tar:/bin/tar, effectively allowing the Popen driver to use the tar program.

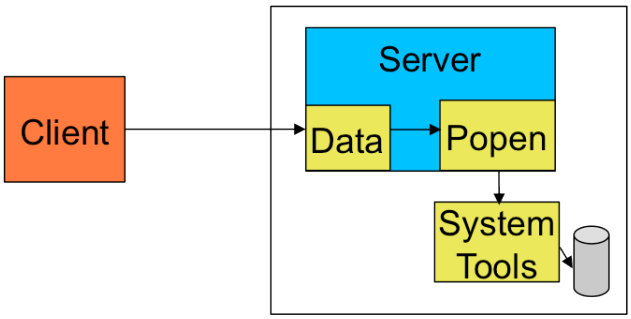

Figure 3. XIO Pipe Open Driver.

4. POPEN DRIVER USE CASES

In this section we describe several use cases of the Globus XIO Popen driver.

4.1 SSH GridFTP

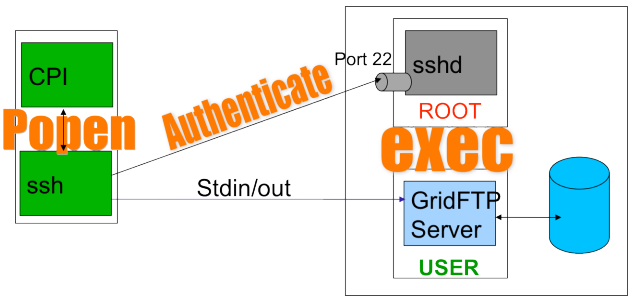

Figure 4. SSH GridFTP.

One of the key advantages of the Popen driver is that it allows us to add SSH [18] as an alternative security mechanism for authenticating with GridFTP. The Globus Toolkit's GridFTP code base has become the de facto standard for data movement within Grid projects and is in use in the vast majority of such projects in the U.S. and abroad. These projects appreciate GridFTP's integration with the public key infrastructure (PKI)-based Grid Security Infrastructure (GSI) [19], as well as its implementation of the fast, efficient, and robust GridFTP data transport protocol. Another important user community for GridFTP comprises small application groups and researchers who often struggle with configuring GSI. For this user community, GridFTP's reliance on GSI can represent a time investment that is not justified because of the associated need to establish, configure, and manage an appropriate PKI. To meet the expressed needs of these communities, we have added SSH as an alternative security mechanism to authenticate GridFTP clients and servers using the Popen driver. The Popen driver allows us to route the control channel over

SSH easily. While globus-url-copy (commonly used GridFTP client) popens the SSH client, the SSH client authenticates with the SSH daemon running on the server machine and remotely starts the GridFTP server as a user process on the server machine. The standard input and standard output (both protected by SSH) become the control channel. This process is illustrated in Figure 4.

4.2. On-The-Fly Tar

GridFTP enhancements such as data channel caching, parallelism, and striping require the use of the extended block mode (MODE E) [2] of GridFTP. In this mode, data channels must go from sender to receiver. Also, MODE E uses a configurable port range for data channel connections. Firewalls have to be configured accordingly.

Some environments (e.g., ―biomedical and health care) impose specialized requirements on a computing infrastructure [20]. In particular, participating institutions have differing firewall requirements, ranging from no firewall to one or more institutional firewalls.

Some sites do not allow any inbound connections to client machines. Thus, MODE E, which enables advanced GridFTP performance features such as pipelining, parallelism, and striping, cannot be leveraged in transfers on these clients for downloads, because inbound connections are blocked by firewalls. Data has to be downloaded by using the standard FTP mode, where a separate TCP data connection me be formed for each file to be sent. The result is greatly reduced performance.

Faced with large datasets composed of many very small files, the FTP protocol becomes inefficient because with each file a data channel needs to be opened and then closed. This problem is exacerbated when considering the wait time before the data channel is completely closed. This wait time can be up to 4 minutes and usually not less than 1 minute (the actual time depends on the operating system). Additionally, this small-files case can be worsened if the executing transfer reaches the maximum number of TCP connections (by doing concurrent transfers), in which case any additional data channel requests for the transfer will hang. Taking into consideration the numerous complexities of the many-small-files problem, we used the Popen driver to tar the files on the fly. This powerful feature allows us to archive a directory at the source, transfer the file as a single archive, and untar the file as it arrives at its destination directory.

The following steps illustrate a scenario of a directory download:

1. The client creates a control channel connection to the server and tells the server that the requested data must be archived prior to the transfer.
2. If the server has popen support enabled, the server archives the data with the specified command and sends the resulting data as a stream over a single data channel, as generated by the archive program (e.g., tar).
3. The client receives the archive file over the data channel and unpacks it as it is received (again using tar), recreating the directory structure in the client file system.

This approach provides several benefits. One is that, entire transfers are completed with a single command similar to the standard globus-url-copy command, with some additional arguments relating to the program that we are piping data through. Another benefit is that only a single data channel is required―a necessary feature with firewalls that do not allow users to have concurrent connections.

We note, however, that users should take into account the overall benefits of using only one data channel in a transfer, since in some cases transfers using concurrent connections will close in on or slightly exceed the performance of the Popen driver.

4.3 Checksum

GridFTP protocol has a CKSM command to checksum a file―in order to checksum a directory containing large number of files, traversing the remote directory and performing a checksum on files, one at a time, with the CKSM command is time-consuming. The Popen driver with md5sum whitelisted allows us to pipe commands through md5sum and complete a checksum of a file or directory on a remote host. The following steps illustrate the process of doing integrity checks via popen for a directory upload:

1. Upload the directory, for example by using the tar-stream method.
2. Invoke the proper globus-url-copy command to use Popen driver-enabled GridFTP to run md5sum to compute checksums of all files in the uploaded directory and to transfer the result file back to the local machine.
3. Create a local checksum file of the local directory
4. Compare the results of the checksums of the two datasets to verify integrity of the uploaded directory.

5. EXPERIMENTAL RESULTS

In this section we describe the experimental setup and the various tests we conducted comparing file transfer performance and the effect of using checksums.

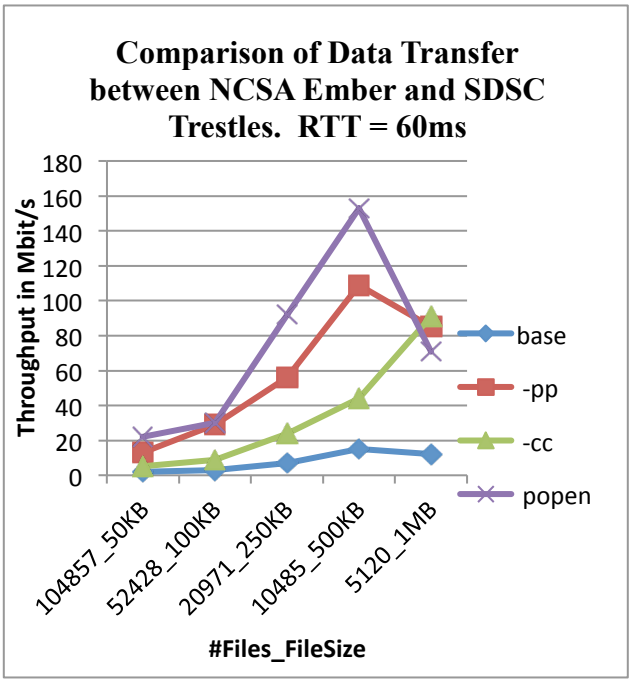

Figure 5. Throughput comparison on 60 ms WAN.

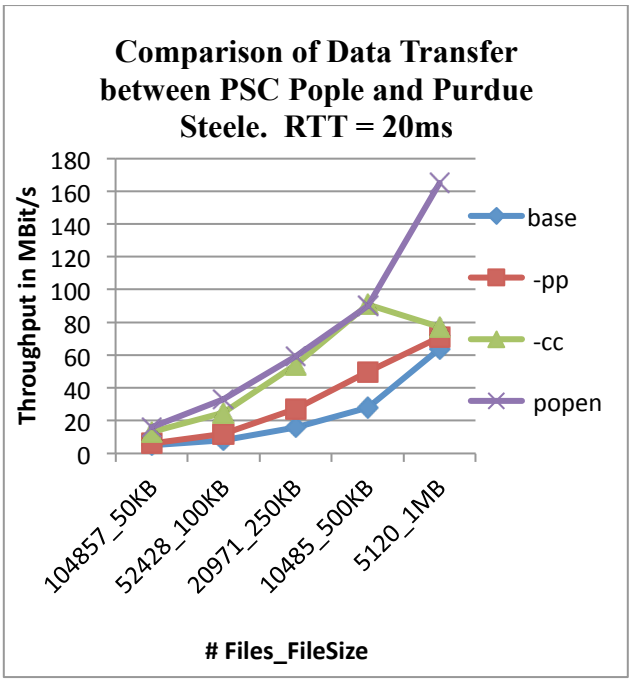

Figure 6. Throughput comparison on 20 ms WAN.

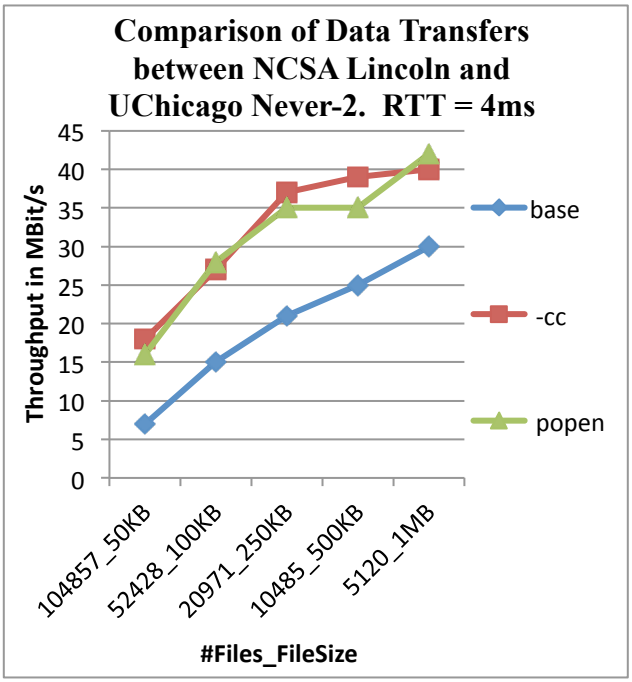

Figure 7. Throughput comparison on 4 ms WAN.

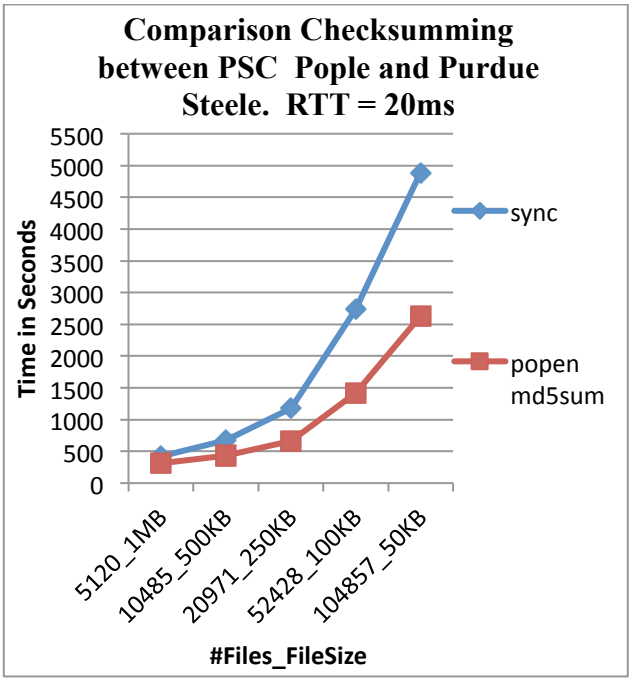

Figure 8. Checksum performance comparison.

5.1 Experimental Setup

Our testing involved six different hosts, although the methods and transfer datasets should be considered identical. These hosts, five of which belong to TeraGrid, were selected based on their round trip times (RTTs) where the goal was to test with resource pairs having RTTs of 4 ms, 20 ms, and 60 ms. The tests were invoked via pre-scripted commands and results redirected into appropriate files. To avoid any potential setbacks with system administrators and current firewall restrictions, we conducted all tests on Globus GridFTP servers built and installed on the user account of the individual conducting the testing. The servers were run with the appropriate whitelist arguments on both the source and destination hosts of the transfer. Our chosen resources were as follows: the Pittsburgh Supercomputing Center (PSC) resource Pople, and the Purdue University resource Steele with an RTT of 20 ms, the National Center for Supercomputing Application (NCSA) resource Ember and the San Diego Supercomputing Center (SDSC) resource Trestles with an RTT of 60 ms, and the NCSA resource Lincoln and the University of Chicago resource Never-2 with an RTT of 4 ms. All datasets used for testing were 5 GB. The number of files and file sizes were as follows: 104,857 50 KB files; 52,428 100 KB files; 20,971 250 KB files; 10,485 500 KB files; and 5,120 1 MB files. Various "blips" in testing occurred and are attributable to system load.

5.2 On-the-Fly Tar

Figures 5–7 compares the performance of transfers using the tar stream functionality in GridFTP with that of the baseline GridFTP and GridFTP with other lots-of-small-files optimizations. The chosen transfer utilities and their arguments were as follows: globus-url-copy with no lots of small files optimization (base), globus-url-copy with only (-pp), globus-url-copy with only (-cc) (in which case a value of 10 was found to be the most acceptable while balancing performance and efficiency), and globus-url-copy with the necessary commands to utilize the Popen driver to tar and untar the dataset (popen).

Our testing reveals that in most cases, as the RTT increases, so does the difference in transfer throughput between the Popen driver tar stream and others—with the exception of –pp. For –pp the transfer throughput increases as RTT increases, as expected. The –pp option eliminates the inter-file latency on the control channel, and thus the effect of –pp is more pronounced as the RTT increases. However, –pp is not as efficient as the tar option. We are not sure why performance drops for the 1 MB files in Figure 5. We suspect that it is caused by external factors such as increase in system or network load.

In several cases using –cc, the results begin to approach, but generally not overcome, our popen tar stream. We note that our –cc was executed with a value of 10 for all tests, a value that is already a relatively resource-demanding 10 concurrent connections, and yet we still see superior performance from the data channel in use by our tar stream.

We ran into some issues running –pp tests between NCSA and UChicago (Figure 7). Hence, –pp numbers are not shown on that graph. We are still examining these issues.

5.3 Checksum

We compared the performance of computing checksums using md5sum via the Popen driver in the GridFTP server with that of computing checksums using the CKSM command in GridFTP. The CKSM command in the Globus implementation of GridFTP uses the OpenSSL libraries to compute checksums. These tests were conducted on only one pair of the resources listed in Section 5.1, namely, the PSC Pople and Purdue Steele. However, we use the same datasets for these tests.

While a few extra steps were involved in computing the checksums using the Popen driver (as described in Section 4.3), significant time savings results, as shown in Figure 8. Note that the x-axis in Figure 8 is different from that in Figures 5–7. The file size goes from 1 MB to 50 KB. The percentage improvement in performance increases as the file size decreases. For the dataset with largest number of files (104,857 50KB files), the traditional checksum method took nearly 2,500 seconds longer (almost twice as long) to complete than it did our md5sum method. Data integrity checking using the popen md5sum option takes a certain amount of initial preparation. We note that the times represented in our results for the popen md5sum method are a combination of the resource time taken to complete all the steps involved.

6. SUMMARY

In this paper, we described the design of Globus XIO Pipe Open Driver that enables an application to leverage existing tools much in the same way as standard Unix pipes. We showed a few use cases of this driver in the context of GridFTP. The driver was used to provide functionalities such as SSH-based security for GridFTP, on-the-fly tar to improve the performance of lots-of-small-files data sets, and faster checksum calculation for directories containing many files using the Unix checksum utilities. We also evaluated the performance of some of these capabilities and showed that they can bring significant performance improvements.

ACKNOWLEDGMENTS

This work was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Dept. of Energy, under Contract DE-AC02-06CH11357.

REFERENCES

[1] 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.

[2] W. Allcock, “GridFTP: Protocol Extensions to FTP for the Grid,” Global Grid Forum GFD-R-P.020, 2003.

[3] J. Postel and J. Reynolds, “File Transfer Protocol,” IETF, RFC 959, 1985.

[4] I. Foster and C. Kesselman, The Grid: Blueprint for a new Computing Infrastructure. Morgan Kaufmann Publishers Inc., 1999.

[5] I. Foster, C. Kesselman, and S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organization,” The International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 200–222, Fall 2001.

[6] TeraGrid. http://www.teragrid.org.

[7] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser, and I. Foster, "GridFTP Pipelining," in *Teragrid 2007 Conference* Madison, WI, 2007.

[8] R. Kettimuthu, A. Sim, D. Gunter, B. Allcock, P. Bremer, J. Bresnahan, A. Cherry, L. Childers, E. Dart, I. Foster, K. Harms, J. Hick, J. Lee, M. Link, J. Long, K. Miller, V. Natarajan, V. Pascucci, K. Raffenetti, D. Ressman, D. Williams, L. Wilson, L. Winkler, “Lessons Learned from Moving Earth System Grid Data Sets over a 20 Gbps Wide-Area Network”, 19th ACM International Symposium on High Performance Distributed Computing (HPDC), 2010.

[9] W. Liu, B. Tieman, R. Kettimuthu, I. Foster, “A Data Transfer Framework for Large-Scale Science Experiments,” 3rd Intl. Wksp. on Data Intensive Distributed Computing (DIDC 2010) in conjunction with 19th Intl. Symposium on High Performance Distributed Computing (HPDC 2010), June 2010.

[10] W. Allcock, J. Bresnahan, R. Kettimuthu, and J. Link, “The Globus eXtensible Input/Output System (XIO): A Protocol Independent I/O 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.

[11] 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.

[12] J. Postel, “RFC 793: Transmission Control Protocol,” September 1981

[13] 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.

[14] J. Bresnahan, M. Link, R. Kettimuthu, I. Foster, “UDT as an Alternative Transport Protocol for GridFTP,” 7th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT 2009), Tokyo, Japan, May 2009.

[15] H. Subramoni, P. Lai, R. Kettimuthu, D.K. Panda, “High Performance Data Transfer in Grid Environment Using GridFTP over InfiniBand,” 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010), May 2010.

[16] J. Bresnahan, M. Link, R. Kettimuthu, I. Foster, “GridFTP Multilinking,” 2009 TeraGrid Conference, Arlington, VA, June 2009.

[17] J. Bresnahan, M. Link, R. Kettimuthu, and I. Foster, “Managed GridFTP,” 8th Workshop on High Performance Grid and Cloud Computing, May 2011

[18] T. Ylonen and C. Lonvick, eds., “The Secure Shell (SSH) Authentication Protocol,” IETF, RFC 4252, 2006

[19] www.globus.org/security/overview.html

[20] R. Kettimuthu, R. Schuler, D. Keator, M. Feller, D. Wei, M. Link, J. Bresnahan, L. Liming, J. Ames, A. Chervenak, I. Foster, C. Kesselman, “Data Management Framework for Distributed Biomedical Research Environments,” IEEE eScience Workshop on High-Performance Computing in the Life Sciences, Dec 2010.