

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

Rajkumar Kettimuthu¹. Steven Link². John Bresnahan¹. Michael Link¹. Ian Foster^{1,3}

¹Computation Institute

²Department of Computer Science

³Department of Computer Science

Argonne National Lab & U.Chicago

Northern Illinois University

University of Chicago

Chicago, IL 60637

DeKalb, IL 60115

Chicago, IL 60637

ABSTRACT

Scientific research of all disciplines unavoidably creates substantially large volumes of data throughout the process of discovery, analysis and conclusion. Given the necessity for data sharing and data relocation, members of the scientific community are often faced with a productivity loss which correlates with the time cost incurred during the data transfer process. 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, it suffers from lower transfer rates. Though the pipelining and concurrency solution in GridFTP provides improved transfer rates for lots of small files datasets, these solutions cannot be applied in environments that have strict firewall rules. In such scenarios, tarring up the files in a dataset on the fly will help. In certain scenarios, compression is desired, in other cases, a checksum of the files after they are written to disk, is desired. There are robust system tools in Unix that perform these tasks (tar, compress, checksum, etc.). In this paper, we present the Globus XIO Pipe Open Driver (Popen) that enables GridFTP to leverage the standard Unix tools to perform certain tasks. We show how this driver is used in GridFTP to provide a number of useful features. We demonstrate the effectiveness of this functionality through an experimental study.

CATEGORIES

H.3.4 Systems and Software

GENERAL TERMS

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

1. INTRODUCTION

Global scale science that can meet today's global challenges 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, 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 those of the 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 achieved 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 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 up the files in a dataset on the fly will help improve the performance. There are scenarios where it makes sense to compress a file before transfer. In many cases, users want to verify the integrity of the data by doing a checksum after the data has been written to disk. There are robust tools available to perform these tasks and it makes sense for GridFTP to utilize these tools. In this paper, we present the Globus XIO [10] Pipe Open Driver (Popen) that enables GridFTP to leverage the standard Unix tools to perform certain tasks. We show how this driver is used in GridFTP to provide a number of useful features, such as SSH-based security for GridFTP, on-the-fly tarring and untarring of files. We demonstrate the effectiveness of this functionality through an experimental study comparing the performance of on-the-fly tarring and untarring of files, alongside 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. In section 4, we describe the use cases of

Popen driver including SSH GridFTP, on-the-fly tar and checksum. Section 5 provides the experimental results and we summarize in Section 6.

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 of 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 on 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 (DSI) [11] allows for easier integration with various storage systems. It supports non-TCP [12] based 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 the 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.

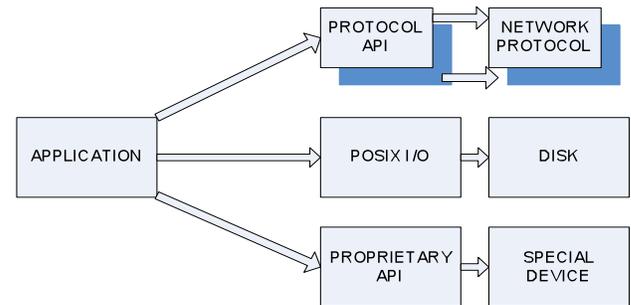


Figure 1. Typical Application Interaction with Various Devices

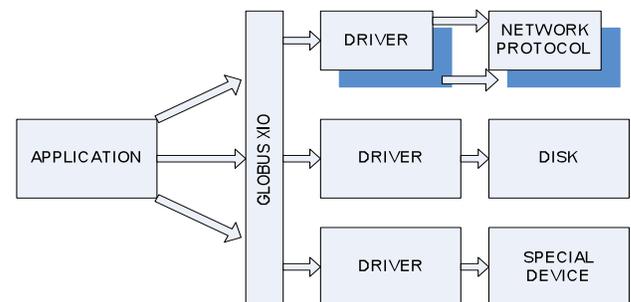


Figure 2. Application Interaction with Various Devices via Globus XIO

3. GLOBUS XIO POPEN DRIVER

Combining multiple tools to accomplish complex tasks with ease is not a new concept. In a typewritten piece of paper, Doug McIlroy described pipes in 1964, long before the advent of Unix. Unix pipes are commonly used to construct powerful Unix command lines by combining Unix commands. Pipes are a Unix feature which allows

you to connect several commands together 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. 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 2 shows five boxes with different names. 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. “Popen” driver is the new XIO driver created to provide piping capability. The piping functionality is achieved by allowing the GridFTP client to replace the “File” driver that handles the file system interactions with the new XIO driver, the pipe open (Popen) driver, on the GridFTP server’s disk I/O stack. As the data blocks pass through the Popen driver, it gets piped to the Unix command that the client provides and 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 really provides the same functionality as you can expect with UNIX pipes, yet the user doesn’t need to worry too much about what is exactly 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.

We note that 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 only permits execution of programs on its Popen whitelist. If a client requests a program to be run that is not on the whitelist, the transfer fails. Below we demonstrate 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
```

Breaking the above command down, we see the standard command for running a GridFTP server, followed by the first whitelist command with 3 arguments, popen, file and ordering. The fs whitelist is a comma-separated list of drivers allowed on the disk stack. We load the Popen module which gives us access to the Popen driver functionality, we load the File driver, this allows us to also conduct non-Popen operations (regular file system interactions), and finally we load the ordering driver. This

is there because when sending data to a pipe it needs to be in order. Often, GridFTP data streams are not in order. The Ordering driver will re-order data and make sure the data that is being directed to the pipe in order. Next we see the popen whitelist, this is a comma-separated list of programs that the Popen driver is allowed to execute. In our case above we see tar:/bin/tar, effectively allowing the Popen driver to use the tar program.

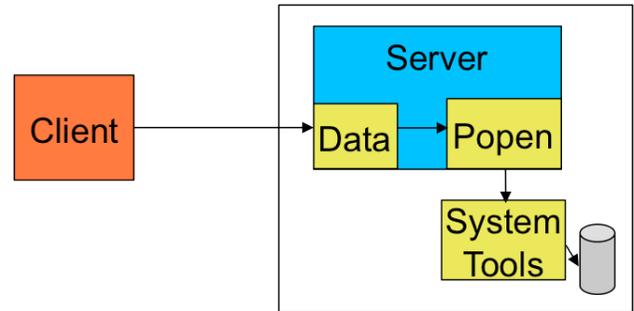


Figure 3: Pipe Open XIO Driver

4. POPEN DRIVER USE CASES

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

4.1 SSH GridFTP

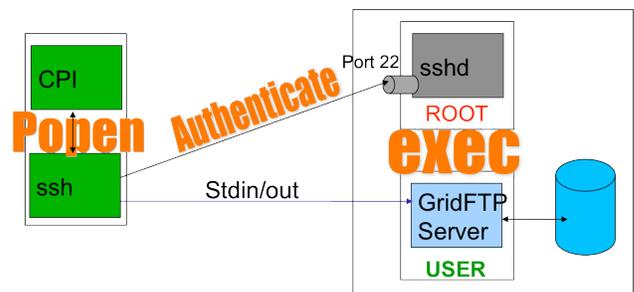


Figure 4. SSH GridFTP

One of the key advantages of the Popen driver is that it allows us to add SSH [18] as an alternate security for authenticating with GridFTP, with relative ease. 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 whom often report that they struggle due to challenges inherent in configuring GSI. For this user community, GridFTP's reliance on GSI can represent a time investment that is not justified, due to the associated need to establish, configure, and manage an

appropriate PKI. Thus, these communities have expressed a strong interest in seeing extensions to GridFTP that would allow for alternative security solutions. 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. Now the standard input and standard output (both protected by SSH) becomes the control channel. This process is illustrated in Figure 4.

4.2. On-the-fly tar

As described in Section 2.1, 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, biomedical and health care environments for example, 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 using the standard FTP mode, where a separate TCP data connection has to be formed for each file to be sent. The result is greatly reduced performance. Faced with large datasets composed of many very small files, it's noted that the FTP protocol becomes quite inefficient because with each file, a data channel needs to be opened and then closed. This problem develops further when considering the TCP TIME_WAIT that takes place before completely closing the data channel. This wait time can exist in some cases for up to 4 minutes and usually not less than 1 minute, however, the actual time is dependent on your 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 any additional data channel requests for the transfer will hang. Taking into consideration the numerous complexities of the many small files problem, we leveraged the Popen driver to tar up 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 un-tar 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 provides several tangible benefits, among them; entire transfers are completed with one command similar to the standard `globus-url-copy` command, with some additional arguments relating to the program that we are piping data through. Another major benefit here is that this requires only a single data channel, which is quite necessary in situations where a limit is placed on concurrent channel connections. This situation can arise with firewalls that simply won't allow a user to have concurrent connections.

We find it necessary to compare the performance of the Popen driver with the performance of several of our other options for transferring files. The reader should take into account the overall benefits of using only one data channel in a transfer when examining the results, as 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, using the CKSM command is pretty 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 illustrates the process of doing integrity checks via popen for a directory upload

1. Upload the directory, e.g. 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 checksum's of the two datasets to verify integrity of the uploaded directory.

5. EXPERIMENTAL RESULTS

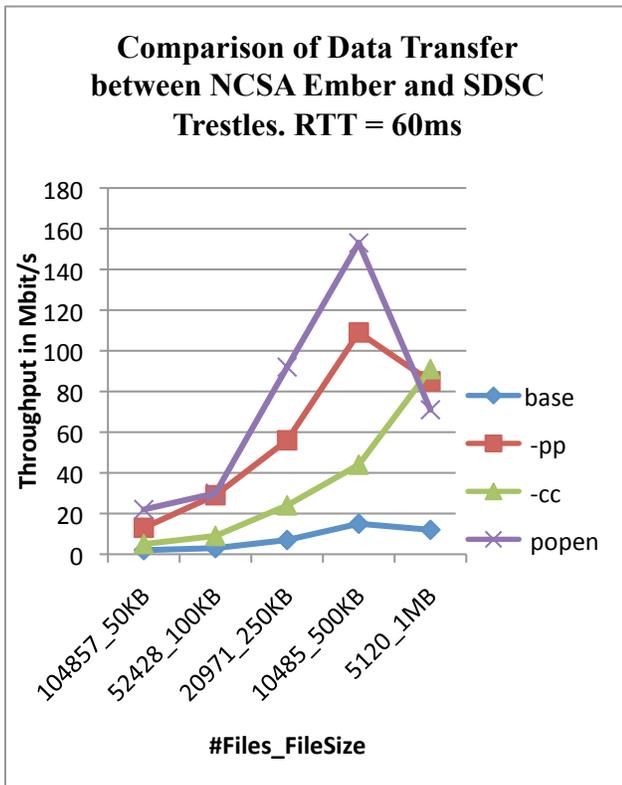


Figure 5. Throughput comparison on 60ms WAN

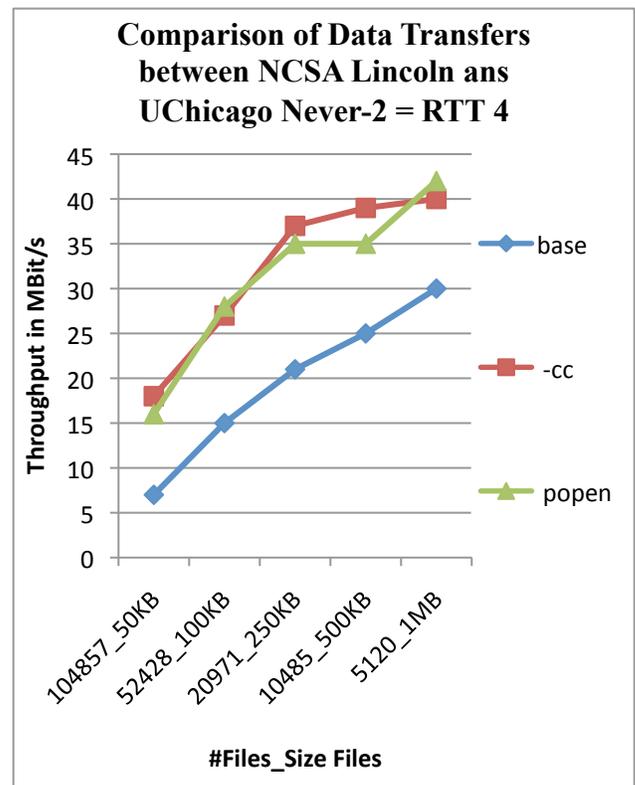


Figure 7. Throughput comparison on 4ms WAN

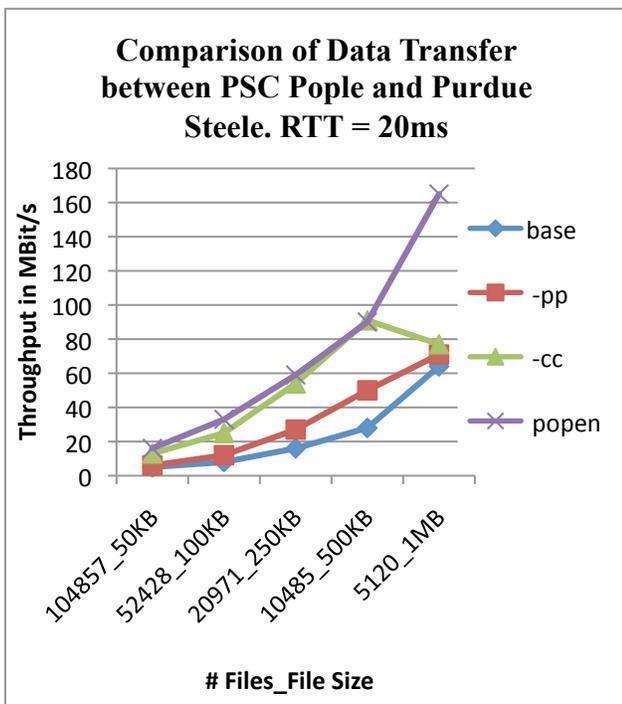


Figure 6. Throughput comparison on 20ms WAN

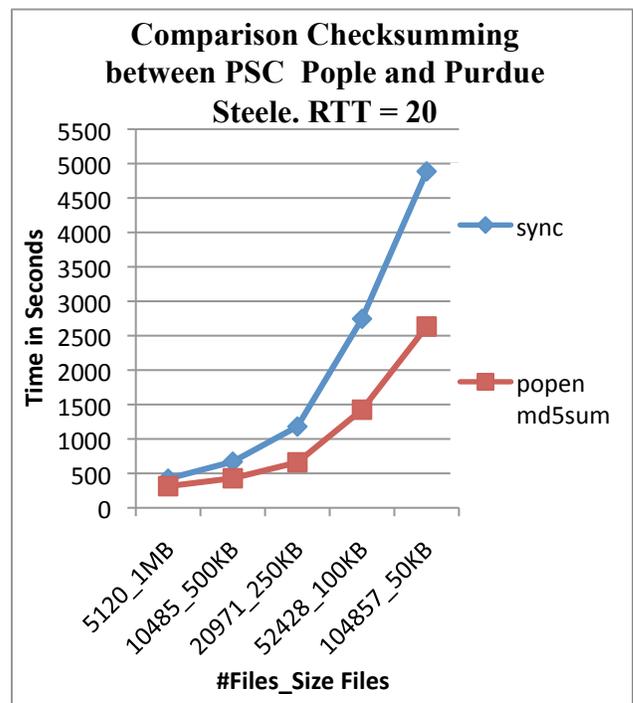


Figure 8. Checksum performance comparison

5.1 Experimental Setup

Our testing will reflect a total of three separate testing relationships between a total of 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 (RTT) where the goal was to test with resource pairs having RTT's of 4ms, 20ms, and 60ms. 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, all tests were conducted 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: Pittsburgh Supercomputing Center's (PSC) resource Pople, and Purdue University's resource Steele with an RTT of 20ms, National Center for Supercomputing Application's (NCSA) resource Ember and SanDiego Supercomputing Center's (SDSC) resource Trestles with an RTT of 60ms, NCSA's resource Lincoln and University of Chicago's resource Never-2 with an RTT of 4ms. All datasets used for testing were 5GB in total size. The range of files in the datasets, which consisted of 5 separate denominations of files and file sizes, were as follows— 104,857 50KB, 52,428 100KB, 20,971 250KB, 10,485 500KB and 5,120 1MB. 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 finally, 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 our RTT increases, so does the difference in transfer throughput between the Popen driver tarstream and others with the exception of -pp. For -pp the transfer throughput increases as RTT increases as it is 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 good as the tar option. We are not sure why there is a drop in performance for the 1MB files in Figure 5. We suspect that it is attributed to external factors such as increase in system or network load.

We identify that in several cases using -cc, that the results begin to close in, but generally not overcome, our popen tarstream. It should be recalled that, our -cc was executed with a value of 10 for all tests, a value which 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). That's why -pp numbers are not shown on that graph. We will include those numbers in the final version of the paper.

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. Those resources are PSC Pople and Purdue Steele. However, we will be using the all of 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), it results in significant timesaving as shown in Figure 8. Please note that the x-axis in Figure 8 is different from that of the x-axes in Figures 5.7. The file size goes from 1MB to 50KB. Percentage improvement in performance increases as the file size decreases. For the dataset with largest number of files – 104,857 50KB files, it took the traditional checksum method nearly 2500 seconds longer (almost twice as long) to complete than it did our md5sum method. Data integrity check using the popen md5sum option takes a certain amount of initial preparation. It should be noted 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 different use cases of this driver in the context of GridFTP. We showed how its been 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.

Acknowledgment

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. Raffanetti, D. Ressler, 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 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>
- [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.