High Performance Data Transfer in Grid Environment Using GridFTP over InfiniBand*

Hari Subramoni [1], Ping Lai [1], Raj Kettimuthu [2], and Dhabaleswar K. (DK) Panda [1]

[1] *Department of Computer Science and Engineering,*
The Ohio State University
{subramon, laipi, panda}@cse.ohio-state.edu

[2] *Mathematics and Computer Science Division,*
Argonne National Laboratory
{kettimut}@mcs.anl.gov

Abstract

GridFTP, designed by using the Globus XIO framework, is one of the most popular methods for performing data transfers in the Grid environment. But the performance of GridFTP in WAN is limited by the relatively low communication bandwidth offered by existing network protocols. On the other hand, modern interconnects such as InfiniBand, with many advanced communication features such as zero-copy protocol and RDMA operations, can greatly improve communication efficiency. In this paper, we take on the challenge of combining the ease of use of the Globus XIO framework and the high performance achieved through InfiniBand communication, thereby natively supporting GridFTP over InfiniBand-based networks. The Advanced Data Transfer Service (ADTS), designed in our previous work, provides the low-level InfiniBand support to the Globus XIO layer. We introduce the concepts of I/O staging in the Globus XIO ADTS driver to achieve efficient disk-based data transfers. We evaluate our designs in both LAN and WAN environments using microbenchmarks as well as communication traces from several real-world applications. We also provide insights into the communication performance with some in-depth analysis. Our experimental evaluation shows a performance improvement of up to 100% for ADTS-based data transfers as opposed to TCP- or UDP-based ones in LAN and high-delay WAN scenarios.

Keywords: *GridFTP, RDMA, Zero-Copy, Globus-XIO, Cluster-of-Clusters, InfiniBand, Obsidian Longbow XR, InfiniBand WAN and iWARP*

I. Introduction

Ever-increasing demands in high-end computing and intensive data exchange, together with the cost effectiveness of high-performance commodity systems, have led to massive deployments of compute and storage systems on a global scale. In such Grid-based scenarios, as shown in Figure 1, bulk data transfer within and across these physically separated clusters or data centers has become an inescapable requirement for scientific data-set distribution, content replication, remote data backup, and so forth. Generally, File Transfer Protocol (FTP) [1] is used for handling bulk data transfers. Since the earliest FTP implementation based on TCP/IP, many efforts have been undertaken to improve and extend it [2], [3], [4], [5], [6], [7]. Among these, GridFTP, designed specifically for high-bandwidth wide area networks, has emerged as the most popular FTP implementation in the Grid environment.

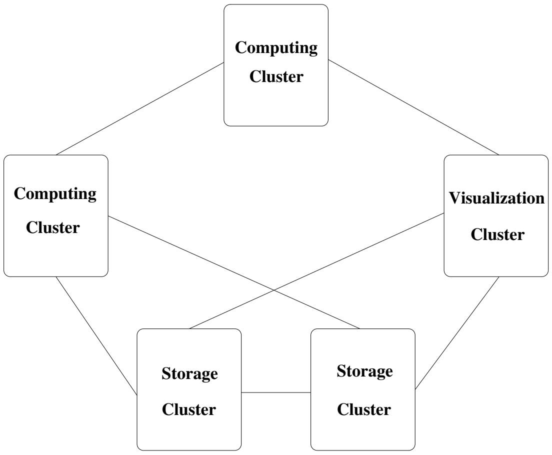

Figure 1. Typical Scenario in Modern Grid Environment

On the other hand, high-performance interconnects such as InfiniBand (IB) [8] and 10 Gigabit Ethernet/iWARP [9] are rapidly gaining momentum for designing high-end clusters and data centers. In addition to providing high bandwidth and low latency, they provide advanced features, such as zero-copy communication and Remote Direct Memory Access (RDMA), that enable the design of novel communication protocols and libraries. The recent introduction of IB WAN routers [10], [11] allows us to extend these capabilities across multiple campuses or even across WAN distances. The Advanced Data Transfer Services (ADTS) [12] is one such high-performance FTP

*This research is supported in part by U.S. Department of Energy grants #DE-FC02-06ER25749, #DE-FC02-06ER25755 and contract #DE-AC02-06CH11357 ; National Science Foundation grants #CNS-0403342, #CCF-0702675, #CCF-0833169, #CCF-0916302, #CCF-0621484 and #OCI-0926691; grant from Wright Center for Innovation #WCI04-010-OSU-0; grants from Intel, Mellanox, Cisco, QLogic, and Sun Microsystems; Equipment donations from Intel, Mellanox, AMD, Obsidian, Advanced Clustering, Appro, QLogic, and Sun Microsystems.

library designed to run natively over IB-based LANs and WANs.

Although IB-based communication offers excellent performance benefits, it is not easy to port an existing application to use IB, because of the difference in communication semantics. The Globus XIO framework [13], used to design GridFTP, offers an easy-to-use interface to end users as well as applications, to perform file transfers over the network without having to worry about the communication semantics of the underlying devices (network or disk). In this paper, we take on the challenge of combining the ease of use of the Globus XIO framework and the high performance achieved through InfiniBand communication by natively supporting GridFTP over InfiniBand based networks. In particular, we port the ADTS library to the Globus XIO framework so that it can be used by GridFTP to transfer files. We also enhance the disk I/O performance of the existing ADTS library by adding support to decouple the network processing from the disk I/O operations.

We evaluate our design at both the microbenchmark and application levels. The results of the microbenchmark-level experiments show that the implementation of the Globus XIO framework over the ADTS library offers up 100% improvement in performance over the TCP or UDP based ones (IPoIB) in both LAN as well as WAN scenarios. We also see that the separation of network processing from the disk I/O processing has an impact on the performance, especially when we go to WAN scenarios where delays can be of the order of tens or even hundreds of milliseconds. Similar trends are seen with the application-level evaluation as well. We simulate the communication patterns of various high-end computing applications such as the Community Climate System Model [14] and ultra-scale visualization [15], which transfer large quantities of data over high-delay networks.

The remainder of this paper is organized as follows. Section II describes the motivation of this work. Section III gives a brief overview of GridFTP and InfiniBand. In Section IV-B we present our design of Globus ADTS XIO driver. We evaluate and analyze the performance in various scenarios in Section V, describe the related work in Section VI, and summarize the conclusions and possible future work in Section VII.

II. Motivation

Modeling of complex systems, such as climate [16] or supernova [17], at higher fidelity generates proportionately larger volumes of data that must be visualized, examined, and studied by widely dispersed scientific teams for gaining insights and making new discovery. The amount of data being created by major computational codes exceeds the capability of current network-based data distribution methods. This is true despite the availability of 10 Gbps backbones like ESnet [18] or Science Data Network (SDN), which is symptomatic of much deeper challenges which will get exacerbated with the next generation of Petascale and Exascale projects.

Historically, wide-area data transport is handled mostly by Transmission Control Protocol (TCP), which has been the basis for File Transfer Protocol (FTP), GridFTP [2], [19], and HyperText Transfer Protocol (HTTP). The unprecedented demands that Petascale applications place on data transport will push TCP well beyond its useful envelope. The fundamental problem with TCP ultimately reduces to its treatment of bandwidth as a shared resource, and there have been a number of efforts to develop high-performance versions of TCP [20], [21], [22], [23] and reliable layers on top of UDP [24], [25] but only with limited success. Existing research shows that a reliable UDP (UDT) is able to achieve a much higher throughput than a TCP connection. However, the user level processing incurred by UDT requires GridFTP to consume a lot more resources then even a regular TCP connection would incur[26].

On the other hand, InfiniBand [8], originally designed for short range data transfers between high-performance storage and computing systems now have WAN capabilities thanks to devices such as the Longbow ER/XR routers [10] from Obsidian. Such capabilities and the introduction of newer communication technologies such as RDMA over Ethernet (RDMAoE) [27] is also accelerating the acceptance of high performance IB as a suitable candidate for WAN communication. Given all these, there now exists multiple ways for the end user to perform a long distance data transfer, which we summarize in Figure 2.

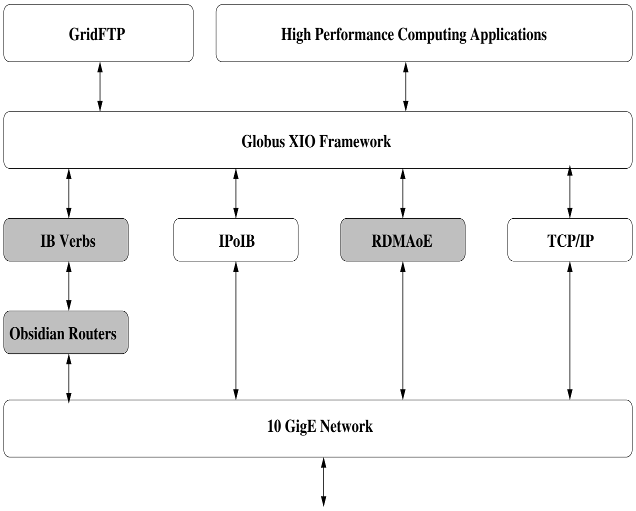

Figure 2. Communication Options Available in Modern Grid Environment

Of the various communication options shown in Figure 2, we have already seen that TCP, UDP and protocols adapted from them (e.g., IPoIB or SDP) cannot achieve good performance in IB based systems. Our earlier work on ADTS [12] shows, taking GridFTP as an example, the various application level limitations imposed by these adaptations. Given this scenario, we are left with two choices to design our high performance FTP library - IB Verbs and RDMA over Ethernet. These options are highlighted in gray color in Figure 2. We focus on the first (IB Verbs) of the two design choices in this paper.

III. Background

In this section we give some brief background material on the topics covered in the paper.

A. GridFTP

GridFTP [2] is a high-performance, secure, reliable extension of the standard FTP data transfer protocol, optimized for high-bandwidth, wide area networks. The Globus implementation of GridFTP [19] provides a software suite optimized for a broad range of data access applications, including bulk file transfer and data extraction from complex storage systems. Thousands of installations around the world rely on GridFTP for over 6 million transfers per day [16], [28] GridFTP has been used in record-setting DOE/HEP Large Hadron Collider Computing Grid data service challenges [29] and is a fundamental building block for a wide range of distributed data management and computing systems.

The following are some of the key advantages of GridFTP:

- Performance: Enhanced performance through use of parallel streams and coordinated data transfers
- Security: PKI/X.509 and SSH-based Grid security
- Robustness: Restart markers allowing interrupted transfers to restart with minimal delay overhead
- Extensibility: Easy-to-use OCRW interface to users and applications

B. InfiniBand and InfiniBand WAN

At the low level, InfiniBand supports different transport services including Reliable Connection (**RC**) and Unreliable Datagram (**UD**).

InfiniBand Architecture (IBA) [8] defines a switched network fabric for interconnecting processing nodes and I/O nodes, using a queue-based model. It has multiple transport services including Reliable Connection (**RC**) and Unreliable Datagram (**UD**), and supports two types of communication semantics: Channel Semantics (Send-Receive communication) over RC and UD, and Memory Semantics (Remote Direct Memory Access - RDMA communication) over RC. Both semantics can perform zero-copy data transfers, i.e. the data can directly be transferred from the application source buffers to the destination buffers without additional host level memory copies.

TCP/IP network protocol stack can be adapted for use on InfiniBand through the IPoIB driver [30]. When it is applied, an InfiniBand device is assigned an IP address and can be accessed just like any regular TCP/IP device.

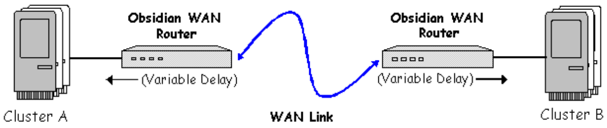

Figure 3. Cluster of Clusters Connected with Obsidian Longbow XRs

InfiniBand Range Extension with Obsidian Longbows: Obsidian Longbows [10] primarily provide range extension for InfiniBand fabrics over modern 10 Gigabit/s Wide Area Networks (WAN), supporting IB traffic at SDR rates (8 Gbps). The Longbows work in pairs, establishing point-to-point links between two clusters, with one Longbow at each end of the link as shown in Figure 3. The Longbows unify both the networks into one InfiniBand subnet, which is transparent to the InfiniBand applications and libraries except for the increased latency added by the wire delays.

Typically, a delay of 5 μs is expected per kilometer of wire length in WAN. The Obsidian Longbow routers provides a web interface for each to specify the delay. We leverage this feature to emulate cluster of clusters with varying degrees of separation in our experiment.

IV. Design and Methodology

In this section we describe our approach to transferring large-volume data over the Grid.

A. Design of the Globus ADTS XIO Driver

As seen in Section II, most of the high-performance computing applications require movement of terabytes to petabytes of data across large distances. Such large volumes of data necessitate the involvement of high-capacity, unfortunately slower, storage devices such as hard disks or RAID's. Because of the relatively low bandwidth offered by the existing networking stacks (TCP or UDT), the disk I/O performance of the FTP library was not a serious performance bottleneck. With the introduction of the Globus ADTS XIO driver, however, the LAN as well as WAN performance will take a big leap as seen in our previous work on the ADTS library [12]. In this scenario, it becomes imperative for us to effectively decouple the network from the disk I/O in order to achieve faster file transfers.

The ADTS library was initially designed for memory-based file transfers, and the performance bottlenecks introduced by devices with a slower data rate such as a hard disk or RAID's [31] were not addressed. For this purpose, we redesign the ADTS library and introduce multiple threads (read, write, and network thread) as well as a set of buffers to stage the data and thus decouple the disk from the network. Such decoupling of network from the slower mass storage devices will be useful especially in high-delay WAN scenarios where consistently keeping the network pipe full is a challenge. The new architecture of the ADTS library is shown in Figure 4.

The read thread prefetches a set of locations from the disk and keeps it ready for the network thread to send over the physical link. To prevent frequent context switches between the threads and ensure that the network thread gets a large amount of the CPU time, we define low- and high-water marks for the read and write threads. We begin reading only when the number of available buffers goes below the low-water mark. The high-water mark is set to the maximum size of the circular buffer. We have similar low- and high-water marks for the write thread as well. The values need to be carefully tuned to achieve best performance for a given network delay. The functionality of the write thread is similar to that of the read thread.

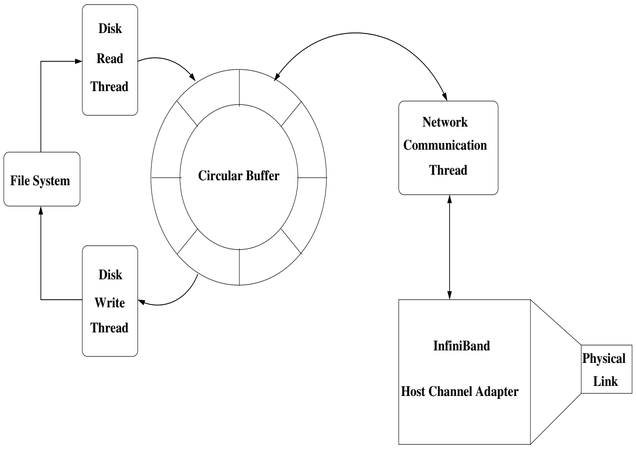

Figure 4. I/O Staging Design for ADTS XIO Driver

The ADTS XIO library has multiple buffers with configurable sizes.

- Network Buffer: Used by the ADTS driver to inject and extract data from the network
- Staging Buffer: Used by the ADTS driver to
  - buffer the data from disk before injecting into the network and
  - buffer the data from network before writing it to the disk

From our earlier work on the ADTS library [12], we found out that we are able to saturate the link bandwidth in LAN as well as high delay WAN scenarios at a packet size of 1 MB. Hence, for all our experiments we set the network buffer size to be 1 MB. The staging buffer is a configurable parameter and will be critical in determining the level of decoupling obtained between the network and the disk I/O. The exact size of the staging buffer to be used varies depending on the file size as well as the delay involved in the data transfer. A smaller buffer size would mean less memory consumed as well as more frequent writes or reads from the disk, resulting in more time being spent on performing disk I/O operations than network file transfers. The impact of such frequent disk I/O operations may not be readily visible in low latency networks where it is easier to keep the pipeline full. But in WAN scenarios with high delays, we will see the true impact of buffer size on the performance of applications.

B. Methodology

Grid computing by definition [32], [33] can have many applications running on computational systems in geographically distributed locations, attempting to work on pieces of a huge dataset. Such an approach involves sending and retrieving data and results from other applications or a central location. For such an architecture, it is more natural for the application to fetch the data in-band, while the application is in operation), rather than out-of-band and run the risk of the data being stale.

Hence, apart from the standard set of comparisons using GridFTP, we also use this approach to evaluate our design. We integrate the Globus XIO based file transfer mechanisms into sample applications based on the client-server model similar to the one shown in Figure 5. As we can see, with the Globus XIO framework, little effort is needed for an application to effectively use a high-performance file transfer library such as ADTS.

V. Experimental Results

In this section we describe the experimental setup, provide the results of our experiments, and give an in-depth analysis of the results.

A. Experimental Setup

The experimental setup consists of two hosts connected through the Obsidian WAN routers as shown in Figure 6. The hosts are equipped with the Intel Nehalem series of processors with Dual quad-core processor nodes operating at 2.40 GHz with 12 GB RAM and a PCIe 2.0 interface. Although we have DDR/QDR rate cards available, the total available bandwidth is limited by the SDR (8 Gbps) transmission rates offered by the Obsidian routers. Red Hat Enterprise Linux Server release 5.3 with Linux Kernel version 2.6.18-128 was used on both the hosts. For all IPoIB (TCP/IP) based tests, auto-tuning of the socket buffers was enabled. Due to the bug [34] with the *CUBIC* congestion control protocol [23] on the 2.6.18 kernels, the *HTCP* congestion control mechanism was used. We believe that this will not adversely impact the performance

```
int main(int argc, char *argv[])
{
    globus_result_t                 res;
    globus_xio_driver_t             adts_driver;
    globus_xio_stack_t              adts_stack;
    globus_xio_handle_t             handle;
    globus_size_t                   nbytes;
    char *                          contact_string = NULL;
    char                            buf[256];
    int                             server = 0;

    contact_string = argv[1];
    adts = argv[2];
    server = argv[3];

    globus_module_activate(GLOBUS_XIO_MODULE);
    res = globus_xio_driver_load(
            "adts",
            &adts_driver);
    assert(res == GLOBUS_SUCCESS);

    res = globus_xio_stack_init(&adts_stack, NULL);
    assert(res == GLOBUS_SUCCESS);
    res = globus_xio_stack_push_driver(adts_stack,
                                        adts_driver);
    assert(res == GLOBUS_SUCCESS);

    res = globus_xio_handle_create(&handle,
                                    adts_stack);
    assert(res == GLOBUS_SUCCESS);

    res = globus_xio_open(handle, contact_string, NULL);
    assert(res == GLOBUS_SUCCESS);

    if (server) {
        res = globus_xio_write(handle, buf,
                sizeof(buf) - 1, 1, &nbytes, NULL);
        assert(res == GLOBUS_SUCCESS);
    } else {
        res = globus_xio_read(handle, buf,
                sizeof(buf) - 1, 1, &nbytes, NULL);
        assert(res == GLOBUS_SUCCESS);
    }
    globus_xio_close(handle, NULL);

    globus_module_deactivate(GLOBUS_XIO_MODULE);

    return 0;
}
```

Figure 5. Example Globus XIO ADTS Program

of TCP/IP based results as *HTCP* is an aggressive version of TCP/IP which give good performance in LAN and WAN scenarios [35]. OFED version 1.4.2 was installed on both the hosts.

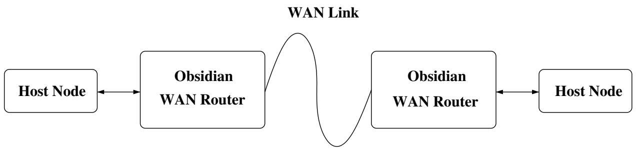

Figure 6. Experimental Setup

B. Microbenchmark-Level Evaluation

In this section we present the results of the microbenchmark-level experiments carried out on the experimental systems mentioned above.

1) Memory-to-Memory Data Transfer: In this experiment, the data transfers were performed from host memory directly to target memory. This approach was used in order to measure the raw performance of the underlying network protocol with varying network delay. As we can see from Figures 7 (a) and 7 (b), while the ADTS-based implementation is able to either saturate the link bandwidth or consume a substantial portion of it successfully, the IPoIB-based implementation is able to use only 10% - 15% of the total link bandwidth. Each point in the graph represents the throughput number obtained while transmitting 128 GB of aggregate data as multiples of 256 MB files. An interesting observation is that, for low-delay scenarios, the best performance is obtained when we use smaller staging buffers. Buffering packets help improve file transfer performance only when I/O performance is worse than the network performance or when there is high delay in the network. Since we are doing memory transfers and the performance of memory operations is much better than that of network based operations, we will in fact see a reduction in performance in low-delay scenarios as the size of the staging buffer size. But when there is considerable delay in the network, as is the case with 1 ms network delay, buffering the packet actually helps because we are able to dedicate more time to polling the network. Overall, we see that a staging buffer size of 32 MB gives the best performance for varying network delays.

2) Disk-to-Disk Data Transfer: The performance comparison of ADTS- and IPoIB-based GridFTP *Get* operation for disk to disk data transfer is shown in Figure 8. Though the performance is considerably less as opposed to the memory based file transfers, the trends seen are similar to those observed for the memory to memory transfers. From the figure, we can see that the performance of ADTS-based GridFTP is almost twice as better compared to that of IPoIB-based GridFTP in almost all scenarios. As the IPoIB numbers do not show much difference in performance with varying delays for any of the staging buffer sizes used, we only show the case which gives the best performance (64MB staging buffer size). From this graph, we can conclude that there is no one size of the staging buffer that suits all possible network delays. The staging buffer size will have to chosen carefully and dynamically based the network characteristics prevalent at that time.

3) Disk-to-Disk Data Transfer with Data Staging and Asynchronous I/O: The performance of staged asynchronous disk to disk I/O based GridFTP operations is shown in Figure 9. We can clearly see that we are able to get better performance by using larger staging buffer size. As expected, the staged asynchronous I/O based approach does better than the synchronous I/O based approach seen in the previous section for medium to large delays as it helps in keeping the pipeline full by allowing the network thread to poll the network more frequently. The IPoIB-based approach gets saturated at a much lower level as usual. As the IPoIB numbers do not show much difference

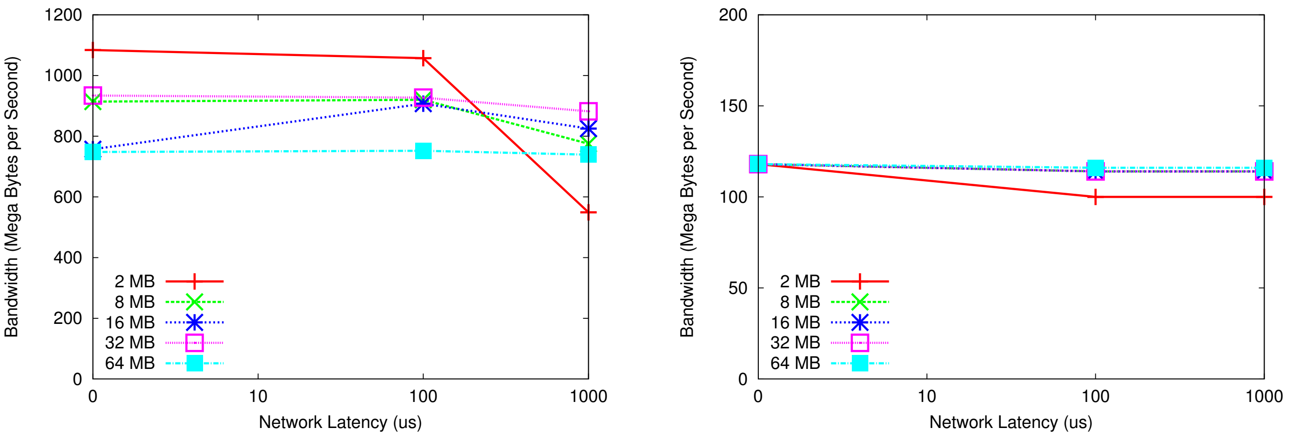

Figure 7. Performance Memory-Based FTP *Get* Operation for (a) ADTS Driver and, (b) TCP Driver

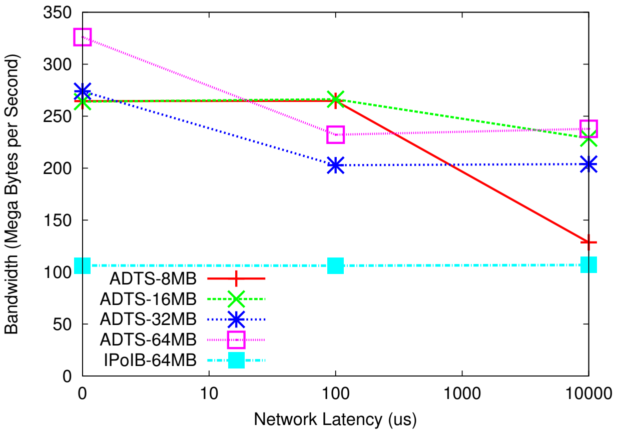

Figure 8. Performance of FTP *Get* Operation for Disk-Based Transfers with Staging and Synchronous I/O

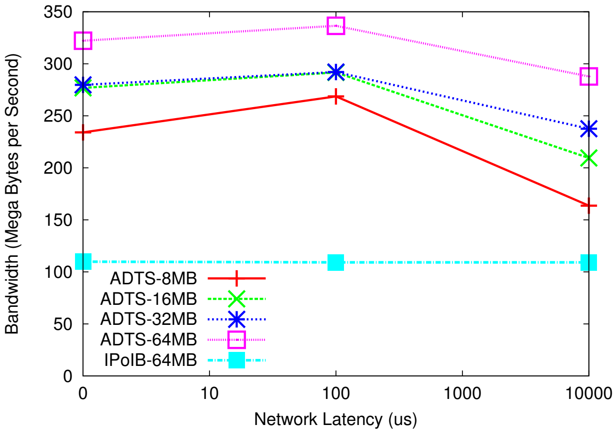

Figure 9. Performance of FTP *Get* Operation for Disk-Based Transfers with Staging and Asynchronous I/O

in performance with varying delays for any of the staging buffer sizes used, we only show the case which gives the best performance (64MB staging buffer size).

C. Application-Level Evaluation

In this section, we use communication traces from two applications to evaluate the performance of our design. The first application is the replication of Community Climate System Model (CCSM) [14] hosted at National Center for Atmospheric Research (NCAR) as part of the Earth System Grid [16] project. The intention is to replicate large volumes of data (in the order of 160 TBytes) sourced at NCAR to other participating sites such as Lawrence Livermore National Laboratory (LLNL). The files are on the order of 256 MB. We use our design to emulate this data replication between two clusters connected with Obsidian Longbows. The latency between NCAR and LLNL is around 30 ms, which we use to configure the Longbow latency. The other application is the Ultra-Scale

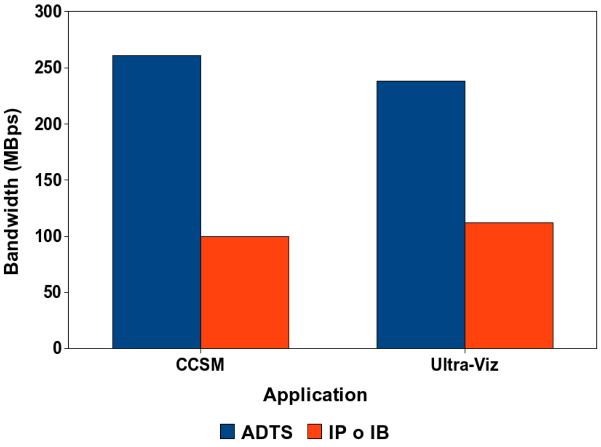

Figure 10. Performance of FTP *Get* Operation for CCSM Application and Ultra-Scale Visualization Traces

Visualization project that transfers large numbers of 2.6 GB files between ORNL and UC Davis. As before, we use Obsidian Longbows to simulate the distance between these two sites by setting the latency to 80 ms.

Figure 10 shows a comparison of the bandwidths achieved by our design and the IPoIB-based design during the data replication. The new design clearly outperforms the IPoIB-based design by more than a factor of two. Even with such large delay, our design is able to sustain good performance. This supports the prospective use of the new design in real Grid scenarios.

VI. Related Work

Researchers have investigated FTP from multiple angles, including security, performance, distributed anonymous FTP, and extensibility [3], [36]. The extension to support IPv6 and transfer files over Network Address Translators is introduced in [7]. In [2], the authors proposed GridFTP, which performs efficient, TCP-based transfers through the use of multiple streams for each transfer. Also, scientists investigated ways to improve multiple file transfers using SCTP multistreaming, parallel transfers [6]. The use of UDP-based transfers was explored in [5], [4], in order to overcome some of the limitations in TCP. Our group has investigated the use of high-performance transport protocols supported by advanced interconnects in FTP design [12]. In this work, we designed and implemented the ADTS layer and evaluated its impact on the FTP performance.

Several solutions have been developed to address limitations of TCP's AIMD-based congestion control mechanism [37]. These solutions include improvements to TCP, such as Binary Increase Congestion Control [38], CUBIC [23], High-Speed TCP (HSTCP) [39], Hamilton TCP (HTCP) [22], Scalable TCP [21]; new transport protocols such as XCP [40], XTP [41]; and reliable layers on top of UDP [24], [25], [42], [43] that target high data rates over wide-area connections. But the task of sustaining end-to-end throughput at rates of 10 Gbps over thousands of miles remains complex. Recently, UDT has been integrated into GridFTP as an alternative transport protocol for TCP [26], However, the user-level processing in UDT requires more CPU and memory resources than does TCP.

Researchers have explored the advanced features of InfiniBand for designing better middleware and applications. The work on NFS over RDMA demonstrates better performance [44] because of the benefits of RDMA in several scenarios. Several researchers [45], [46], [47], [35] have investigated various aspects of the performance characteristics over IB WAN.

VII. Conclusions and Future Work

In this paper we take on the challenge of designing an easy-to-use, high-performance FTP library, drawing on the advantages of ease of use given by the Globus XIO framework used by GridFTP and the high-performance nature of the ADTS FTP library designed by us. We enhance the existing ADTS design by adding new elements to efficiently decouple the network and disk I/O processing. Such decoupling allows us to get better performance for real-world applications in WAN scenarios with large delays as evidenced by the results of our microbenchmark as well as application level evaluations. Our experimental evaluation clearly indicates the benefits of using a high-performance transport driver like ADTS for GridFTP. We see that the ADTS-based design outperforms the existing designs based on IPoIB (TCP/IP, UDP) by a factor of two in LAN as well as WAN scenarios.

As part of future work we plan to evaluate the performance of GridFTP running over ADTS on InfiniBand interfaces that have support for the RDMA over Ethernet protocol. This approach will allow us to run directly on real 10GigE networks and perform more in-depth analysis of the effects of TCP/IP cross traffic on IB flows in Ethernet networks. Such an experimental setup will also allow us to compare IB directly with TCP/IP, something that we have not been able to do well until now. We also plan to run I/O benchmarks such as IOZone [48], Bonnie [49] and FIO [50] to evaluate how effective our disk I/O performance is as opposed to the best possible value that can be obtained as shown by these benchmarks.

References

[1] J. Postel and J. Reynolds, "File Transfer Protocol. RFC 959."

[2] W. Allcock, "GridFTP: Protocol Extensions to FTP for the Grid. Global Grid ForumGFD-R-P.020,2003."

[3] M. Allman and S. Ostermann, "Multiple Data Connection FTP Extensions," Ohio University, Tech. Rep., 1996.

[4] D. Bush, "UFTP," http://www.tcnj.edu/ bush/uftp.html.

[5] K. R. Sollins, "The Trivial File Transfer Protocol – TFTP 2 RFC 1350. July, 1992."

[6] S. Ladha and P. D. Amer, "Improving Multiple File Transfers Using SCTP Multistreaming," in *Int'l on Performance, Computing, and Communications Conference*, April 2004.

[7] M. Allman, S. Ostermann, and C. Metz, "FTP Extensions for IPv6 and NATs. RFC 2428. Sep. 1998."

[8] "InfiniBand Trade Association," http://www.infinibandta.org.

[9] "Architectural Specifications for RDMA over TCP/IP," http://www.rdmaconsortium.org/.

[10] "Obsidian Research Corp." http://www.obsidianresearch.com/.

[11] Net NX 5010 High Speed Exchange, http://www.net.com/products/products-nx.shtml.

[12] P. Lai, H. Subramoni, S. Narravula, A. Mamidala, and D. K. Panda, "Designing Efficient FTP Mechanisms for

High Performance Data-Transfer over InfiniBand," in *(ICPP 09)*, Sep. 2009.

[13] W. Allcock, J. Bresnahan, R. Kettimuthu, and J. Link, "The Globus eXtensible Input/Output System (XIO): A Protocol Independent IO System for the Grid," *Parallel and Distributed Processing Symposium, International*, vol. 5, p. 179a, 2005.

[14] Community Climate System Model, http://www.ccsm.ncar.edu.

[15] Institute for Ultra Scale Visualization, "Ultravis: Ultra Scale Visualization," http://ultravis.ucdavis.edu/.

[16] Earth System Grid, http://www.earthsystemgrid.org.

[17] SciDAC Computational Astrophysics Consortium, http://www.supersci.org.

[18] "Energy Sciences Network (ESnet)," http://www.es.net.

[19] W. Allcock, J. Bresnahan, R. Kettimuthu, and M. Link, "The Globus Striped GridFTP Framework and Server," in *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. Washington, D.C.: IEEE Computer Society, 2005, p. 54.

[20] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1246–1259, 2006.

[21] T. Kelly, "Scalable TCP: Improving Performance in Highspeed Wide Area Networks," *SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83–91, 2003.

[22] D. J. Leith and R. Shorten, "H-TCP protocol for high-speed long distance networks," in *Second International Workshop on Protocols for Fast Long-Distace Networks*, Argonne, IL,, 16-17 February 2004.

[23] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.

[24] Y. Gu and R. L. Grossman, "UDT: UDP-based Data Transfer for High-Speed Wide Area Networks," *Comput. Netw.*, vol. 51, no. 7, pp. 1777–1799, 2007.

[25] E. He, J. Leigh, O. Yu, and T. A. DeFanti, "Reliable Blast UDP : Predictable High Performance Bulk Data Transfer," *Cluster Computing, IEEE International Conference on*, vol. 0, p. 317, 2002.

[26] J. Bresnahan, M. Link, R. Kettimuthu, and I. Foster, "UDT as an Alternative Transport Protocol for GridFTP," in *International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, 21-22 May 2009. [Online]. Available: http://www.mcs.anl.gov/ kettimut/PFLDNeT09.pdf

[27] "Open Fabrics Enterprise Distribution," http://www.openfabrics.org/.

[28] Argonne National Laboratory, "Advanced Photon Source," http://www.aps.anl.gov/.

[29] "LHC: Large Hadron Collider," http://lhc.web.cern.ch/lhc/.

[30] Mellanox OFED Stack for Linux Users Manual, http://www.mellanox.com/pdf/products/software/ Mellanox_OFED_Linux_User_Manual_1_20.pdf.

[31] D. A. Patterson, G. Gibson, and R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," in *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 1988, pp. 109–116.

[32] "Grid Computing," http://en.wikipedia.org/wiki/Grid_computing.

[33] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.

[34] Office of Science, DOE, http://fasterdata.es.net/TCP-tuning/linux.html.

[35] N. S. V. Rao, W. Yu, W. R. Wing, S. W. Poole, and J. S. Vetter, "Wide-Area Performance Profiling of 10GigE and InfiniBand Technologies," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008.

[36] F. Anklesaria et.al, "The Internet Gopher Protocol. RFC 1436. Network Working Group," March 1993.

[37] M. Allman, V. Paxson, and W. Stevens, "TCP Congestion Control - RFC 2581," United States, 1999.

[38] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks," in *IEEE Infocom*. IEEE, 2004. [Online]. Available: http://www.ieee-infocom.org/2004/Papers/52_4.PDF

[39] "HighSpeed TCP," http://www.icir.org/floyd/hstcp.html.

[40] D. Katabi, M. Handley, and C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," in *SIGCOMM '02: Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. New York, NY, USA: ACM, 2002, pp. 89–102.

[41] W. T. Strayer, M. J. Lewis, and R. E. Cline, Jr., "XTP as a Transport Protocol for Distributed Parallel Processing," in *HSNS'94: Proceedings of the High-Speed Networking Symposium on USENIX 1994 High-Speed Networking Symposium*. Berkeley, CA, USA: USENIX Association, 1994, pp. 6–6.

[42] Tsunami Network Protocol Implementation, http://www.indiana.edu/ uits/cpo/tsunami, 2002.

[43] R. L. Grossman, M. Mazzucco, H. Sivakumar, Y. Pan, and Q. Zhang, "Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks," *J. Supercomput.*, vol. 34, no. 3, pp. 231–242, 2005.

[44] R. Noronha, L. Chai, T. Talpey, and D. K. Panda, "Designing NFS with RDMA for Security, Performance and Scalability," in *In The 2007 International Conference on Parallel Processing (ICPP-07)*, Aug 2007.

[45] S. Carter, M. Minich, and N. S. V. Rao, "Experimental evaluation of infiniband transport over local- and wide-area networks," in *High Performance Computing Symposium*, 2007.

[46] W. Yu, N. S. Rao, P. Wyckoff, and J. S. Vetter, "Performance of RDMA-capable Storage Protocols on Wide-Area Network," in *Peta-byte Storage Workshop, in conjunction with SC08*, 2008.

[47] S. Narravula, H. Subramoni, P. Lai, R. Noronha, and D. K. Panda, "Performance of HPC Middleware over InfiniBand WAN," in *ICPP'08*, Sep. 2008.

[48] "IOzone Filesystem Benchmark," http://www.iozone.org/.

[49] "Bonnie - IO Throughput Benchmark," http://www.garloff.de/kurt/linux/bonnie/.

[50] "FIO - Linux IO Benchmarking Tool," http://freshmeat.net/projects/fio/.