Designing Efficient FTP Mechanisms for High Performance Data-Transfer over InfiniBand

Ping Lai, Hari Subramoni, Sundeep Narravula, Amith Mamidala, Dhabaleswar K. Panda
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH, USA
{*laipi, subramon, narravul, mamidala, panda*}*@cse.ohio-state.edu*

***Abstract*—The rapid growth of InfiniBand, 10 Gigabit Ethernet/iWARP and IB WAN extensions is increasingly gaining momentum for designing high end computing clusters and data-centers. For typical applications such as data staging, content replication and remote site backup etc., FTP has been the most popular method to transfer bulk data within and across these clusters or data-centers. Although the existing sockets based FTP approaches can be transparently used in these systems through the protocols like IPoIB or SDP, their performance and scalability are limited due to the additional interaction overhead and unoptimized protocol processing. This leads to a challenge how to design more efficient FTP mechanisms by leveraging the advanced features of modern interconnects.**

In this paper we design a new Advanced Data Transfer Service (ADTS) with the capabilities such as zero-copy data-transfer, memory registration cache, persistent data sessions and pipelined data transfer etc. to enable efficient zero-copy data transfers over IB and iWARP equipped LAN and WAN. We then utilize ADTS to design a high performance FTP library (FTP-ADTS). From our experimental results, we observe that our design outperforms existing sockets based approaches by more that 95% in transferring large volumes of data over LAN. It also provides significantly better performance at much lower (by up to a factor of 6) CPU utilization in various IB WAN scenarios. These results present the promising future for designing high performance communication protocols to power the efficiency and scalability of next-generation parallel and distributed environments.

I. Introduction

Ever increasing demands in high end computing and intensive data exchange in data-centers 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 scenarios, bulk data transfer within and across these physically separated clusters or data-centers has been an inescapable requirement for the uses of scientific data-sets distribution, content replication, remote data backup, etc. Generally, File Transfer Protocol (FTP) [21] is used for handling bulk data transfers. Through the years since the earliest FTP implementation based on TCP/IP, there have been a lot of efforts on its improvement and extensions [10], [12], [15], [23], [16], [13], while greatly boost its performance in both LAN and WAN.

On the other hand, System Area Networks (SAN) such as InfiniBand (IB) [4] and 10 Gigabit Ethernet/iWARP [1] are rapidly gaining momentum for designing the high-end clusters and data-centers. These high performance interconnects have revolutionized the communication capabilities of modern systems. In addition to providing high bandwidth and low latency, they also provide advanced features like zero-copy communication and Remote Direct Memory Access (RDMA) that enable the design of novel communication protocols. Furthermore, industry vendors [6], [20] have recently introduced IB WAN routers to extend these capabilities beyond a single cluster or data-center, e.g., across multiple campuses or even across WAN range. Hence, communication libraries are now capable of zero-copy communications over WAN, which also provides a new scope for designing newer FTP mechanisms.

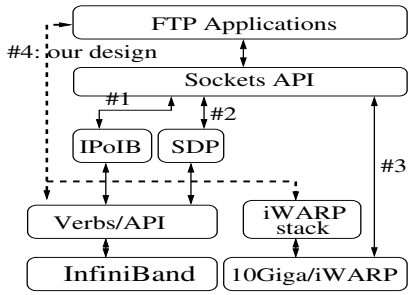
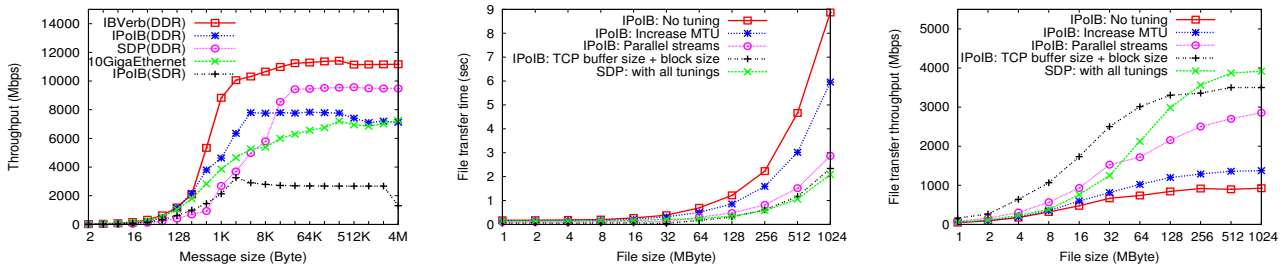


Figure 1. Protocol Stack over IB and 10Giga/iWARP (Existing approaches: #1,#2,#3; Our design: #4)

Multiple approaches can be utilized for implementing FTP in IB based LAN and WAN, as illustrated in Figure 1. We can directly apply the existing sockets based FTP through intermediate drivers (schemes #1, #2 and #3), or we can design new FTP mechanisms using the native IB features (scheme #4). IPoIB (IP over IB) [18] and SDP (Sockets Direct Protocol) [4] are two popular schemes for the first choice. IPoIB encapsulates the standardized IP packets over IB fabrics so that IP based applications can access an IB device as usual. SDP is also a sockets-like implementation which allows sockets applications to be transparently deployed on IB and meanwhile retains



(a) Peak Bandwidth of Communication Protocols (b) Latency of GridFTP with Various Tuning (c) Throughput of GridFTP with Various Tuning

Figure 2. Performance of Current Generation Communication Protocols and GridFTP

most of the native performance. These two schemes require no modifications on existing FTP implementations, but at the cost of losing significant native performance. We can observe this in Figure 2(a) where native IB verbs with DDR (expected peak bandwidth of 16 Gbps) switch achieve much higher bandwidth as compared to other protocols. Moreover, the performance for FTP, e.g., GridFTP, using IPoIB and SDP with various tunings (discussed in detail in Section II) lead to poor performance compared to that achieved at the network level. [1] This leads to a challenge whether FTP mechanism can be designed to maximize the possible benefits of IB and IB WAN for high performance data transfer.

In this paper, we take on this challenge and design a novel zero-copy FTP for bulk data transfers across WAN. We first design a generic high performance Advanced Data Transfer Service (ADTS) layer that support zero-copy communications, and on top of that, we design the FTP client/server library named as FTP-ADTS. The primary contributions of our work are listed as follows:

- Design an Advanced Data Transfer Service (ADTS) that leverages zero-copy capabilities of modern interconnects to perform efficient bulk data transfers, with further optimizations such as memory registration caches, persistent data connections and pipelined data transfers
- Leverage ADTS to design a high performance zero-copy FTP library
- Provide a robust and inter-operable mechanism to support both the zero-copy capable clients and the traditional TCP (or UDP) clients
- Study the performance benefits of our design in the current and emerging high performance environments including both IB LAN and WAN scenarios, and compare it with the current designs available in the literature

Our experimental results show an improvement of up to 95% in latency of transferring large files in LAN as compared to IPoIB (TCP or UDP) based approaches. Further, we demonstrate that for WAN with high network delays, our approach performs significantly better than the existing approaches. We also observe that it achieves peak transfer rates at significantly lower (up to 6 times) CPU utilization.

The remainder of this paper is organized as follows: Section II discusses the motivation in detail, and Section III gives an overview of IB and IB WAN. In Section IV we present our design of ADTS and FTP-ADTS. 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. Detailed Motivation

We have seen in Section I that the TCP or UDP adapted protocols (e.g., IPoIB or SDP) cannot achieve good performance in IB based systems. In this section, we take GridFTP as an example to show the application level limitations by these adaptations.

While running GridFTP through IPoIB and SDP in an IB cluster, we used various well know tunings [11] to improve the performance. Experimental setup are as described in Section V. Figure 2(b) and 2(c) compare the file transfer time and throughput (for *get* operation) with increasing tuning levels (e.g., the legend of "TCP buffer size + block size" means that the tuning on MTU, parallel streams, TCP buffer size and transmission buffer size are all applied). It can be observed that GridFTP over IPoIB performs very badly without any tuning, while it is gradually improved with more and more tuning. For the case of using SDP as the underlying protocol, we only measured the performance with all the tunings enabled. Its performance is not much better than that of using IPoIB, which means that the low-level networking benefits of SDP seen in Figure 2(a) are not fully translated into FTP-level benefits. (Therefore, in Section V, we only compare our design and the existing designs over IPoIB.) Overall, we see that the traditional sockets based FTP is not capable of utilizing the benefits of IB either over IPoIB or SDP. This drives us to leverage the native advanced features, e.g., the zero-copy capabilities of modern interconnects in this work, to design more efficient approaches.

[1] The 10Gigabit Ethernet has the similar peak bandwidth as IPoIB with DDR. We will not discuss this in the following sections, as we focus on the IB based environment in this paper.

III. InfiniBand and InfiniBand WAN

In this section we present the background on InfiniBand and InfiniBand WAN. InfiniBand Architecture (IBA) [4] 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.

Figure 3. Clusters Connected with Obsidian Longbow

InfiniBand Range Extension with Obsidian Longbows: Obsidian Longbows [6] 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 *us* is expected per each km of wire length in WAN. The Obsidian Longbow routers provide 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.

It is to be noted that in this paper we use IB-WAN routers from Obsidian. However, such routers are also available from other vendors such as BayNetworks [2].

IV. Designing Efficient FTP Mechanisms

In this section we describe the details of our zero-copy FTP design. We present the ADTS design followed by the FTP-ADTS library design.

A. Architecture of the Proposed Design

Figure 4 presents the overall architecture of the proposed design. Utilizing the advanced networking capabilities of modern interconnects, we first design a generic Advanced Data Transfer Service (ADTS) layer which has the network and transport functionalities to support high performance zero-copy data transfers as well as the traditional TCP or UDP based data transfer. We then design the FTP-specific functionality and interface on top of ADTS, which is named as FTP-ADTS. (Note that ADTS layer can be reused for other applications by appropriate porting.) These two layers are described in detail in the following.

B. Designing Advanced Data Transfer Service

As shown in Figure 4, ADTS consists of several components that handle different tasks.

Depending on the network equipment, clients may be capable of performing zero-copy data transfer or only support the TCP or UDP based communication. Once the type of transport protocol (channel) is negotiated, the *Data Connection Management* component initiates a connection to the remote peer using the corresponding transport. This channel selection can be adapted dynamically on a per client connection basis to handle different kinds of clients, thus improving robustness and interoperability. In addition to zero-copy, TCP/IP and UDP/IP channels, the DATA Transport Interface provides scope for enabling support for other emerging transports for next generation architectures.

1) Design of Zero-Copy Channel:

Design Alternatives: As mentioned in Section III, we have two possible alternatives for zero-copy design: (i) Memory semantics using RDMA; (ii) Channel semantics using Send and Receive.

RDMA based data-transfer requires allocating and registering buffers on both the source and the destination nodes. The sender initiates the data-transfer by specifying the destination buffer address. It is known that RDMA based approaches [17] achieve better latencies. However, they have three significant drawbacks. Firstly, the target RDMA buffers need to be pre-allocated, registered and the address information need to be communicated to the source process before they can be used. Further, the flow control in RDMA communication is explicit. i.e. the sender can initiate data-transfers only after receiving explicit notification of buffer availability. Secondly, since RDMA operations do not involve remote node CPU, notifying the completion of data-transfers to the remote node requires to send additional messages that based on send-recv, which adds more overhead. And finally, in IB WAN environment, the latency benefits of RDMA seen for small messages are dominated by the actual network delay. Hence it offers no superior benefits over send/recv in these scenarios.

On the other hand, send/recv mechanism shows good benefits. Firstly, zero copy benefits of send-recv mechanism are identical to those seen with RDMA, i.e. the remote data can be directly received to the FTP buffers without kernel-level buffering. Secondly, send-recv mechanisms support easy flow control. For example, the receiver can use SRQ [4] to post buffers when needed automatically. It eliminates the need for explicit flow control. This benefit can be quite significant on IB WAN links because the sender is not throttled due to lack of buffers on the remote node.

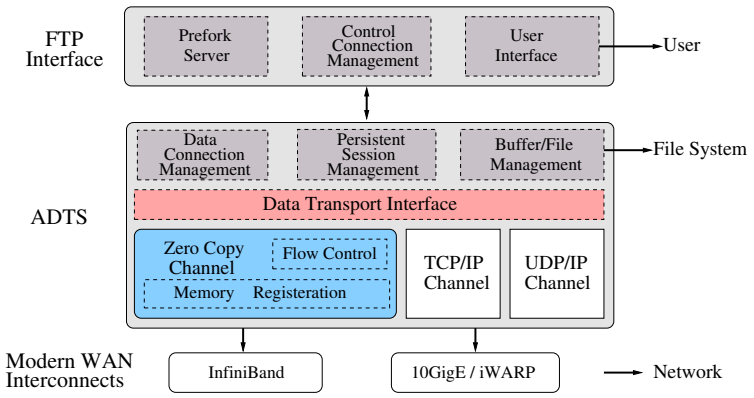


Figure 4. Proposed FTP-ADTS Architecture (Shaded boxes are design components focused in this paper)

In addition, as mentioned in Section III, the InfiniBand's send/recv communications can be used over both the RC and UD transports. Due to these benefits, we utilize send/recv for our design[2]. We will describe the details in the following.

Send/Recv based Design: Once the client and server negotiate the use of zero-copy and the appropriate connections are setup by the *Data Connection Management* component, the channel is marked for zero-copy data transfers. As aforementioned, we need to handle flow control and buffer management.

Each buffer that the underlying layer (IB or iWARP) accesses needs to be registered and pinned in memory. In order to alleviate this overhead, the *Buffer/File Management* component keeps a small set of pre-allocated buffers. Part of the data is first read into these buffers while additional buffers are being allocated and registered as needed. The buffers that are allocated on-demand are unregistered and released on completion of the data transfer.

Unlike the sockets interface where the underlying kernel TCP/IP stack performs the flow control, the ADTS needs to perform explicit flow control to support zero-copy operations. i.e. data cannot be sent out unless the sender is assured of buffer availability on the receiver. In our design, we perform our flow control on the receiver side by using SRQ as discussed earlier. This enables the ADTS to push the data out of the source node at a high rate.

In certain scenarios, the end nodes might not be capable of processing/storing the incoming data at a good rate[3], so it is necessary to throttle the sender. We use a flow control fall back mechanism to throttle the sender as needed.

2) Design Enhancements:

In order to further improve the performance of ADTS, we use the following optimizations: memory registration cache and persistent sessions, and pipelined data transfers.

Memory Registration Cache and Persistent Sessions: While we pre-allocate a set of buffers to speed up processing, the buffers need to be registered for use. This in turn impacts the performance. Existing libraries such as MVAPICH [5] amortize the registration cost by avoiding multiple registration/deregistration calls for multiple transfers on the same buffers, which is popularly known as registration cache. We apply the same technique here.

In typical FTP data transfers, each file is transmitted on a different data-connection, which incurs multiple data connection setup costs. In this situation, memory registration caching would not help significantly to reduce the registration overhead. In order to alleviate costs, we enable persistent data-sessions that keep data connection and the associated buffer alive during the transfer of multiple files. The maximum number of files to be transmitted on a given connection is negotiated in advance and the connection is not closed until all of them are transferred or the connection becomes idle. This approach also allows for efficient use of buffers and memory registrations which boosts the performance significantly.

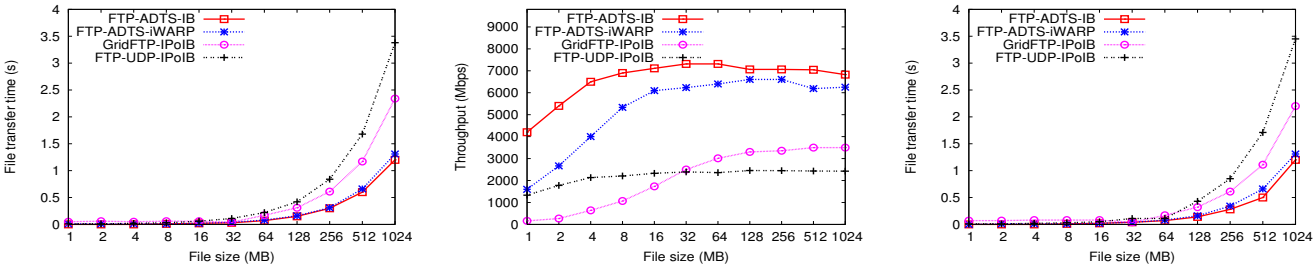
Pipelined Data Transfers: In order to maximize the utilization of both the network bandwidth and the local disk bandwidth (or the local file system being used), the ADTS is designed with two threads. The network thread deals with processing of network related work queues, completion notifications, flow control etc., while the disk thread handles the reads and writes from/to the disk. With this multi-threaded functioning, all data transfers are packetized and pipelined and hence better performance is obtained.

C. Design of FTP-ADTS

Figure 4 shows the basic architecture of our FTP client/server library (*FTP-ADTS*). It utilizes the low-overhead zero-copy ADTS layer to provide high performance FTP transfers. The top level FTP Interface deals with the rest of the features needed for the FTP library. It provides a basic user interface to enable all client interactions. Other

[2]Note that RDMA operations support one-sided data-transfer which can be highly beneficial for certain kinds of applications even in IB WAN scenarios. However, in the context of this paper, zero-copy send/recv based mechanisms are more beneficial.

[3]Note that in the context of high performance FTP transfers, it is reasonable to assume that the end-nodes are capable of sustaining high IO bandwidths.



(a) File Transfer Time for *get* (b) File Transfer Throughput for *get* (c) File Transfer Time for *put*

Figure 5. FTP Performance in IB LAN

main components are described as follows.

1) Control Connection Management:

Based on the user provided information, the client FTP engine initiates a socket based control connection to the remote server. This is used to relay all control information such as FTP commands and error messages. In addition, it is also used to negotiate the Active/Passive modes (PORT/PASV commands in the FTP specification [21]) and transport protocols to be used for data-transfers. Hence, clients that are capable of zero-copy transfers can benefit from higher performance. To enable this, we require the zero-copy capable client to send an additional command TPRT (Transport PRoTocol) advertising its transport preference. Once this negotiation is complete, the ATDS layer initiates the appropriate data connection.

2) Prefork Server:

Multiple parallel connections to the FTP server is a common scenario in most large data-centers. In order to efficiently support such parallelism, we design our FTP server as a multi-process server. The main FTP server daemon forks multiple processes for different clients. Further, the server maintains a small pool of pre-forked processes efficiently to handle bursts of requests.

V. Experimental Results

In this section, we present the experimental results. We evaluate the performance of our FTP design in both LAN and WAN scenarios, and further measure the CPU utilization and the benefits of design enhancement. GridFTP and FTP-UDP (we implement a UDP based FTP which has the consideration of unreliable transfers) over IPoIB are utilized as the base case for performance comparisons. We set the TCP window size and MTU size that yield the best performance for IPoIB based on our previous work [19], and did all the tuning as that in Section II.

We use RAM disks for data storage so that there is no disk I/O bottleneck and we can analyze the communication performance more clearly. This is a reasonable simplification in that modern HEC systems or data-centers usually employ high performance parallel or distributed file systems and advanced data storage technologies such as RAID to obtain improved I/O performance.

Experimental Setup: We use a cluster consisting of 64 Intel Xeon Quad dual-core processor nodes with 6GB RAM on each node. The nodes are equipped with both IB DDR ConnectX HCAs with OFED 1.3 [7] drivers and Chelsio T3b 10 Gigabit Ethernet/iWARP adapters. The OS used was RHEL4U4. The nodes are divided into *Cluster A* and *Cluster B* that are connected with Obsidian InfiniBand WAN routers as shown in Figure 3.

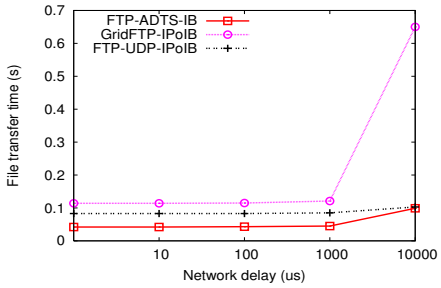
A. Performance in IB LAN Scenarios

This experiment shows the basic performance improvement achieved by our FTP-ADTS as compared to FTP-UDP and GridFTP in low-delay, high-bandwidth LAN. We evaluate the client-perceived file transfer time of both *put* (upload) and *get* (download) operations. Since our FTP can been used either over IB or over 10 Giga/iWARP, we measure the performance in both cases and use FTP-ADTS-IB and FTP-ADTS-iWARP to represent them, respectively.

Figures 5(a) and 5(b) compares the performance of *get* operation with varying file sizes. We see that FTP-ADTS achieves significantly better performance for larger file sizes. FTP-ADTS-IB presents an improvement by up to 95% and 181% as compared to GridFTP and FTP-UDP, respectively, and FTP-ADTS-iWARP also provides significant improvement. This indicates that FTP-ADTS can make much more benefits of IB, which is consistent with what observed in Section I. Since FTP-ADTS-iWARP performs a little worse than FTP-ADTS-IB, we will use FTP-ADTS-IB as the representative of our design in the following analysis. We also observe that GridFTP-IPoIB does not perform well for small file sizes, but does better as the file sizes increase. This confirms what has been indicated about GridFTP in [24]. We see the similar trends for *put* operations as shown in Figure 5(c).

B. Performance in IB WAN Scenarios

In order to study the performance in IB WAN scenarios, we evaluate the FTP operations with server and client running on a pair of nodes connected by a pair of Obsidian routers. The distance between them is emulated by varying the WAN delays as mentioned in Section III.



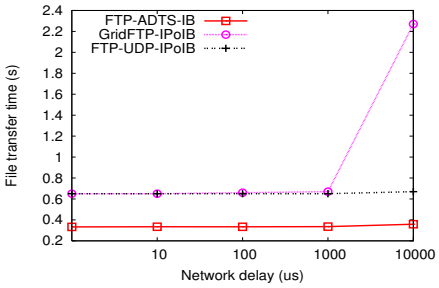


Figure 6. File Transfer Time in IB WAN (get): (a) 32 MBytes and (b) 256 MBytes

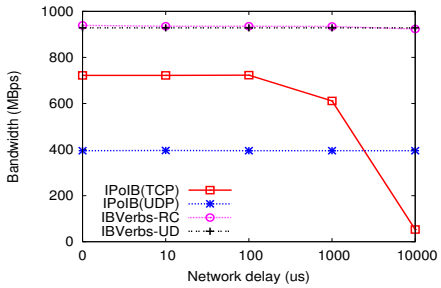


Figure 7. Peak Network Bandwidth with Increasing WAN Delay

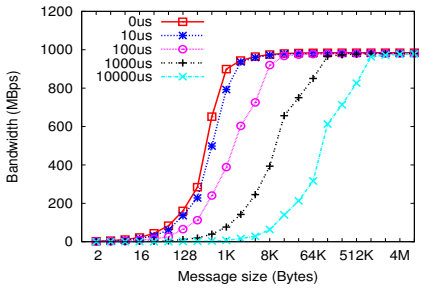


Figure 8. IB RC Bandwidth with Increasing WAN Delay

Basic Performance: We compare the performance of our design, GridFTP and FTP-UDP with varying WAN delays of 0 us (no delay), 10 us, 100 us, 1,000 us and 1,0000us (corresponding to the distance of 0 km, 2 km, 20 km, 200 km and 2,000 km). Figures 6 (a) and (b) present the time for downloading a 32 MByte and a 256 MByte file, respectively. It is obvious that our FTP performs better, especially over high delay network. Particularly, FTP-ADTS sustains performance for larger WAN delays quite well, while GridFTP-IPoIB shows a steep latency increase when the WAN delay arrives 10000 us. The improvement is not only due to the underlying faster zero-copy operations, but also because the network throughput is the bottleneck for IPoIB in WAN where issues such as RTT time and MTU size can severely degrade the performance. Further, although we have done standard TCP tuning and GridFTP parameters tuning, the overhead incurred by the interactions between TCP and IPoIB (especially in WAN) contributes to the performance degradation [22]. Another observation is that here the FTP-UDP performs better than GridFTP. It is well-known that UDP can achieve good performance over high-bandwidth, high-latency networks where TCP has fundamental limitations [3] because of flow control. Due to the space constraints in the paper, we do not show the performance of FTP *put* operations which demonstrates similar trends.

Detailed Analysis: We further carried out the following two experiments to demonstrate the fundamental reasons.

First we measure the WAN peak bandwidth of different transports. Figure 7 shows the results for IPoIB (including TCP/IP and UDP/IP) and IB verbs (including RC and UD) as WAN delays increases. It is to be noted that Obsidian routers currently support SDR (8Gbps) link only. We can see that IB verbs achieves the stable highest bandwidth through the whole range of delays (RC and UD have the same bandwidth because the very large messages used elliminate the limitation of reliable management in RC [19]), while the TCP/IP bandwidth drops fast when the delay is high, which is consistent with the degradation in GridFTP performance. On the other hand, although the UDP/IP bandwidth with smaller delays is lower than the TCP/IP, it shows no significant degradation as the delay increases. It is even better than TCP/IP when the delay is high ($>=$ 10000 us). This is because that UDP can swamp the network by firing many successive packets, but TCP is not capable of saturating available bandwidth due to congestion control and flow control. (Please note that researchers have shown that TCP bandwidth over longer pipes can be improved by techniques such as multiple parallel streams. While this improves the bandwidth performance, this also needs additional resources at the end nodes. We intend to study the impact of parallel zero-copy protocol and parallel TCP/IP streams in future.)

The second experiment is to characterize the impact of packet size. Usually, larger packets are preferred for better use of the link bandwidth. We claim that our design is benefited from the use of very large packet size (i.e. 1 MByte) in IB send-recv operations, while IPoIB is limited by the largest packet size of 64 KByte. In this experiment, we vary the packet size of IB RC verbs (which is used in our design) and measure the bandwidth. From the results shown in Figure 8, we observe that the bandwidth for small and medium messages becomes progressively worse with increasing network delays. This demonstrates that some of the benefits in our design can be attributed to the use of

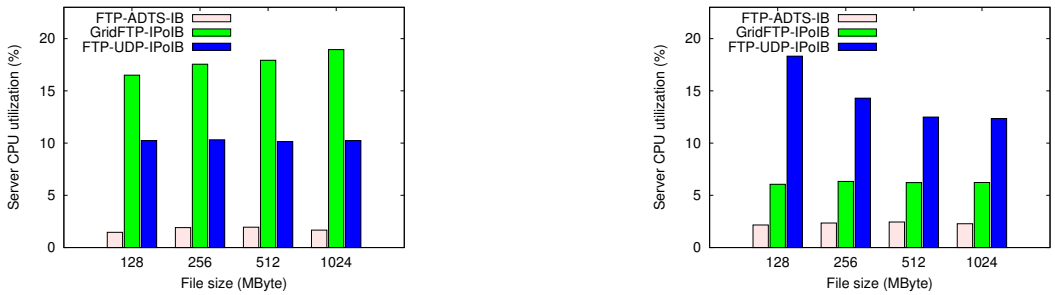


Figure 9. End Node CPU Utilization (a) Server; (b) Client

large packet sizes.

C. Performance of Multiple File Transfers

In some data-center scenarios such as site replication, mirroring, etc., FTP is used to transfer multiple files. We evaluate the performance of site replication in this section.

We measure the performance of FTP-ADTS and FTP-UDP (since we already observed that FTP-UDP is better than GridFTP over the high-delay IB WAN links, we only use FTP-UDP here) using a zipf [26] trace. This trace has a high α value with an average file size of about 66 MB. The average amount of time to replicate this trace over WAN is shown in Figure 10. We see that the FTP-ADTS speeds up the data transfer by up to 65%. This presents that the FTP-ADTS is a promising candidate for some real applications. We also observe that the time in both cases increases for very large network delays. This is due to the fact that here the zipf trace consists of a large number of smaller sized files that could degrade the WAN performance.

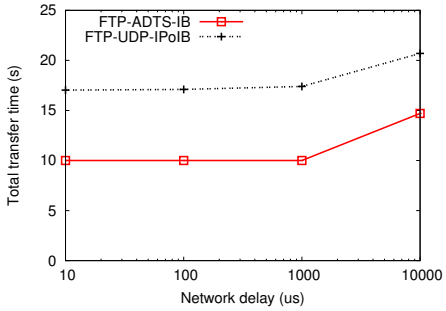


Figure 10. Site Replication over IB WAN using FTP

D. Benefits in CPU Utilization

TCP or UDP based communications often suffer from the added CPU utilization for TCP(UDP)/IP stack processing [9]. Particularly, at sender side, the data is copied into kernel's socket buffer before being sent out, and at receiver side, the data is also copied to kernel's buffer before being written to the destination disk. While the sender side overhead can be alleviated by utilizing the *sendfile* system call, the receiver side overhead is usually unavoidable.

Figures 9 (a) and (b) show the normalized CPU utilization (the total percentage of CPU time being used on our 8-core system) at server and client, respectively, while performing multiple back-to-back large file *put* operations. As expected, the GridFTP and FTP-UDP over IPoIB uses a significant amount of CPU on both the server and the client for additional data copies. On the other hand, FTP-ADTS has much lower CPU utilization due to the use of zero-copy protocol. Further, we observe that the CPU utilization of GridFTP client is very low, which demonstrates the benefits of using *sendfile* to reduce one memory copy on the client side. FTP-UDP does not show such trends since UDP can not use this optimization. Overall, our approach requires fairly smaller amount of CPU time for all file sizes, therefore is more scalable than the IPoIB based designs.

E. Benefits of the Design Enhancements

In this experiment we detail the benefits of design enhancements (seen in Section IV-B2).

We break up the performance of the FTP-ADTS while transferring a set of small files into Connection time (*Conn*) and Data Transfer time (*Data*). Figure 11 shows this breakup with varying number of successive file transfers in two cases: (i) *Basic*: with all optimizations disabled and (ii) *Opt*: with all optimizations enabled. We clearly observe the following two trends: (i) pipelined data transfers, buffer reuse and memory registration caches improve the performance significantly (upto 55% improvement for the transfer of 16 files of size 1MB) and (ii) the use of persistent sessions improves the connection setup time considerably, i.e. the cost of initiating the connections is incurred only once instead of incurring on a per transfer basis.

VI. Related Work

Researchers have investigated FTP from multiple angles including security, performance, distributed anonymous FTP and extensibility [12], [14]. The extension to support IPv6 and transfer files over Network Address Translators (NATs) is introduced in [13]. In [10], the authors have proposed GridFTP which performs efficient TCP based transfers through the use of multiple streams for each transfer. Also, scientists aimed to improve multiple file transfers using SCTP multistreaming, parallel transfers [16]. The use of UDP based transfers has been explored in [15], in order to overcome some of the limitations in TCP.

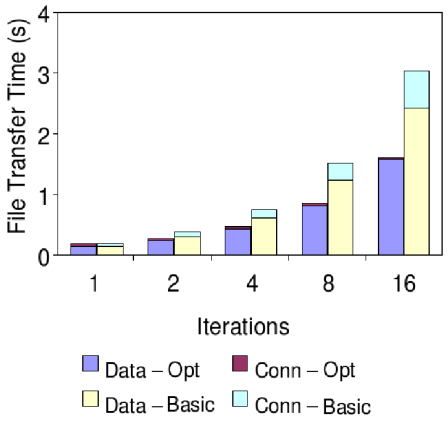


Figure 11. Benefits of Design Enhancements

On the other hand, researchers have explored the advanced features of modern interconnects (e.g. IB) and IB WAN. The work on NFS over RDMA demonstrates better performance [8] due to the benefits of RDMA in several scenarios. Literature [22], [25], [19] investigate various aspects of the performance characteristics over IB WAN.

In this paper, we propose to leverage the advanced features such as zero-copy operations provided by high performance interconnects for improving the FTP capabilities in high-end IB based clusters or data-centers.

VII. Conclusions and Future Work

The emerging WAN capable interconnects such as InfiniBand WAN and 10 Gigabit Ethernet/iWARP are making rapid advances in the high-end computing clusters and data-centers, where FTP is the most popular method to transfer bulk data in many applications. Although the existing solutions, e.g., TCP or UDP or SCTP based FTP implementations, can be directly used in these systems by the intermediate protocols such as IPoIB and SDP, their performance is far lower than the maximum network capabilities. Attempting to improve this, in this paper we propose and design a novel FTP library (FTP-ADTS) that is ported to our Advanced Data Transfer Service (ADTS) and that is capable of efficiently transferring data by leveraging the zero-copy operations of modern interconnects.

From our experimental results, we have observed that our *FTP-ADTS* outperforms existing approaches by upto 95% in transferring large amounts of data in LAN, as well as significant improvements in various IB WAN scenarios. We also observed that our approach achieves peak transfer rates at much lower (up to 6 times) CPU utilization, resulting in much better scalability.

Our studies demonstrates that the novel FTP design using IB advanced features can provide very efficient file transfer, thus offering the insight to design next generation high performance applications in a radically different manner. In the future we intend to explore these challenges in other communication middleware and study the impact of modern WAN interconnects on their designs.

References

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

[2] BayNetworks. www.baynetworks.com/.

[3] Gigabit Ethernet Jumbo Frames. http://sd.wareonearth.com/ phil/jumbo.html.

[4] Infiniband Trade Association. http://www.infinibandta.org.

[5] MVAPICH2. http://mvapich.cse.ohio-state.edu/.

[6] Obsidian research corp. http://www.obsidianresearch.com/.

[7] Open Fabrics Enterprise Distribution. http://www.openfabrics.org/.

[8] Sun Microsystems and The Ohio State University. NFS over RDMA Design, Version 1.1, Aug 2007.

[9] TCP Overhead. http://www.cs.duke.edu/ari/trapeze/freenix/node11.html.

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

[11] W. Allcock and J. Bresnahan. Maximizing Your Globus Toolkit GridFTP Server. In *CLUSTERWORLD*, 2004.

[12] M. Allman and S. Ostermann. Multiple Data Connection FTP Extensions. Technical report, Ohio University, 1996.

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

[14] F. Anklesaria and et.al. The Internet Gopher Protocol. RFC 1436. Network Working Group, Mar. 1993.

[15] D. Bush. UFTP. http://www.tcnj.edu/ bush/uftp.html.

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

[17] J. Liu, J. Wu, S. P. Kini, P. Wyckoff, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. In *ACM Int'l Conference on Supercomputing*, 2003.

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

[19] S. Narravula, H. Subramoni, P. Lai, R. Noronha, and D. K. Panda. Performance of HPC Middleware over InfiniBand WAN. In *Int'l Conference on Parallel Processing*, Sept. 2008.

[20] Net NX 5010 High Speed Exchange. http://www.net.com.

[21] J. Postel and J. Reynolds. File Transfer Protocol. RFC 959.

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

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

[24] W. Allock, J. Bresnahan, R. Kettimuthu, and M. Link. The Globus Striped GridFTP Framework and Server. In *Super Computing, ACM Press*, 2005.

[25] 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.

[26] G. K. Zipf. Human Behavior and the Principle of Least Effort. Addison-Wesley Press, 1949.