Designing Zero-Copy FTP Mechanisms to Achieve High Performance Data-Transfer over InfiniBand WAN

SUNDEEP NARRAVULA, PING LAI, HARI SUBRAMONI,
AMITH MAMIDALA, DHABALESWAR K. PANDA

Technical Report
OSU-CISRC-4/08-TR18.

Designing Zero-Copy FTP Mechanisms to Achieve High Performance Data-Transfer over InfiniBand WAN

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

***Abstract*—FTP has been the most popular method to transfer large files for data-staging, replication, and the like. While existing FTP mechanisms have improved gradually with newer networks, they still inherit the fundamental limitations imposed by the underlying networking protocols (TCP/UDP) they use. These include limited network bandwidth utilization, high memory bandwidth and CPU utilization that TCP/UDP cause on the end-nodes. Thus both the performance and scalability of such systems is limited. The advent of InfiniBand (IB) WAN has enabled the use of high performance transport protocols in the WAN scenarios, which can be leveraged for designing FTP mechanisms. Enabling IB-based FTP capabilities and providing good efficiency for such transfers presents considerable challenge.**

In this paper we present an Advanced Data Transfer Service (ADTS) to enable efficient data transfers over WAN. We leverage the ADTS's capabilities to design high performance file transfer mechanisms (FTP based on ADTS). Our ADTS layer improves data transfer performance by optimizing several aspects including efficient buffer management, memory registration cache, pipelining of data transfers, reducing TCP/IP related data copies, and maintaining persistent FTP data sessions. Further, we reduce the CPU utilization required for the data-transfers (by up to a factor of 6) and demonstrate a significantly higher FTP server scalability. In our experimental results, we observe that our FTP-ADTS design outperforms existing TCP and UDP based approaches by more that 80% in transferring large volumes of data. In addition, we utilize the WAN emulation capabilities of Obsidian InfiniBand WAN routers to study the impact of our designs in a wide range of WAN scenarios, leading to solutions that enable the design of highly capable WAN communication protocols required to power the next-generation high performance parallel and distributed environments.

Keywords: *FTP, RDMA, Zero-Copy, Cluster-of-Clusters, InfiniBand, Obsidian Longbow XR, InfiniBand WAN, iWARP*

I. Introduction

Ever increasing needs in High End Computing (HEC) and cost effectiveness of high performance commodity systems have led to massive deployments of highly capable systems on a global scale. This tremendous increase in compute and storage capabilities has necessitated bulk data transfers among various storage systems and/or compute systems that are often physically separated. Typical uses for such large scale data transfers include distribution of scientific data-sets, content replication, remote site backup, etc. Traditionally, File Transfer Protocol (FTP) [19] has been used for handling such bulk data transfers.

While FTP provides the basic data transfer functionality, it also suffers from several critical performance overheads. In particular, since FTP is based on TCP/IP, it inherits the fundamental limitations imposed by TCP/IP itself, such as high CPU utilization, multiple memory copies and limited peak bandwidths achieved over high delay links. Researchers in [3], [4] have looked at several improvements to FTP for optimizing performance. Techniques such as TCP tuning, striped access, persistent data connections, etc. have been proposed for providing improved performance. However, these approaches still bank on TCP/IP (along with its limitations) for data-transfers and fail to achieve good performance for high-delay, high-bandwidth links.

In order to drive higher bandwidth performance, approaches utilizing *"TCP unfriendly"* mechanisms such as multi-streaming, FTP over UDP or SCTP, etc. [6], [20], [13], [3] have recently gained prominence. Even though these approaches achieve better bandwidth as compared to TCP based approaches, they still suffer from multiple memory copies and high CPU overheads.

On the other hand, a rapid growth of modern high performance networking interconnects, such as InfiniBand (IB) and 10 Gigabit Ethernet/iWARP, have revolutionized the communication capabilities of modern systems. In addition to providing high bandwidth and low latency, these systems also provide advanced features such as zero-copy data transfers and Remote Direct Memory Access (RDMA) that enable the design of novel communication protocols that can avoid the main limitations in using TCP/IP (and UDP or SCTP). Industry vendors [8], [17] have recently introduced IB WAN routers that enable IB over WAN. With the advent of InfiniBand WAN and iWARP, communication libraries are now capable of zero-copy communication over WAN. This presents the scope for designing high performance communication protocols in wide area networks as well. However, in order to maximize the possible benefits of the IB WAN capabilities, the data transfer mechanisms required for FTP need to be designed carefully, which presents a considerable challenge.

In this paper, we present a novel zero-copy approach for bulk data transfers across WAN. In our approach, we design a high performance Advanced Data Transfer Service (ADTS) that utilizes zero-copy capabilities of modern networks to

achieve scalable and efficient data transfers. We leverage the performance of the ADTS layer to design a high performance FTP client/server library. We further investigate the issues involved and study the benefits of our approach in various IB WAN scenarios.

The following are the primary contributions of this paper:

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

Our experimental results show an improvement of up to 80% in latency achieved for large files as compared to TCP based approaches using GridFTP. Further, we demonstrate that for WAN scenarios with high network delays, our approaches work significantly better than the TCP and UDP based approaches. We also observe that our approach achieves peak transfer rates at significantly lower (up to 6 times) CPU utilization.

The remainder of this paper is organized as follows: Section II gives a brief overview of the basic FTP protocol and modern interconnects. We briefly discuss the communication performance limitations in Section III. In Section IV we present our RDMA based design of FTP. We evaluate and analyze the performance of our design in various scenarios in Section V. Section VI describes the related work. Finally, we summarize our conclusions and possible future work in Section VII.

II. Background

In this section we briefly present the required background on File Transfer Protocol (FTP), InfiniBand and InfiniBand WAN.

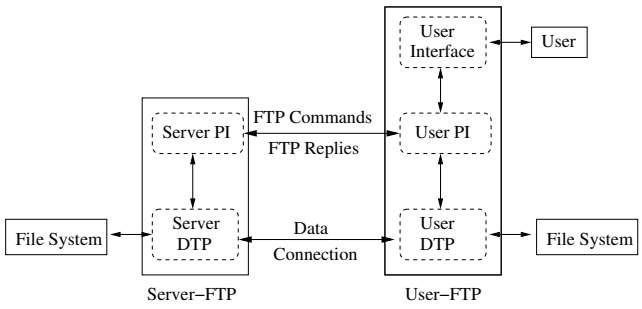

Fig. 1. Basic FTP Model (Courtesy: RFC 959 [19])

A. File Transfer Protocol

File Transfer Protocol (FTP) [19] was initially proposed to promote file sharing and the implicit use of remote computers. It is an application level protocol used to copy files between the local machine (*user*) and the remote machine (*server*) over the network, and fundamentally relies on the TCP/IP protocol. The typical FTP model, as shown in Figure 1, includes control connection and data connection between the user and the server FTP processes for communicating the control commands and the data (file) respectively. The FTP server plays a passive role of listening on the specific port when started, and the user initiates a connection which may involve negotiation of authentication and certification with the server. After that, control commands can be transferred over the connection. The data connections are established as needed for the data transfers.

Through the years since the release of the protocol, FTP has seen a variety of extensions and improvements [1], [16], [15], [3], [20], [6], [13], largely due to the explosive development of Internet and data-intensive applications. This trend is expected to grow with the availability of newer networking technologies and other related approaches.

B. InfiniBand

InfiniBand Architecture (IBA) [5] defines a switched network fabric for interconnecting processing nodes and I/O nodes. It uses a queue-based model. A Queue Pair (QP) consists of a send queue and a receive queue. The send queue contains instructions for transmitting data and the receive queue contains the instructions describing where the receive buffer is. At the low level, InfiniBand supports different transport services including Reliable Connection (**RC**) and Unreliable Datagram (**UD**).

IBA supports two types of communication semantics: Channel Semantics (Send-Receive communication model) for RC and UD, and Memory Semantics (RDMA communication model) for RC. Remote Direct Memory Access (RDMA) [7] operations allow processes to access the memory of a remote process without the intervention of the remote node's CPU. Both the approaches 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. IBA also supports Shared Receive Queue (**SRQ**) mechanism that enables high server scalability while providing efficient flow control. With SRQ receive buffers can be posted on a common receive queue for multiple connections. Further, it also provides a feature which notifies the application when the number of such receive buffers fall below a threshold. The receiver can post additional buffers upon getting the notification making sure that sufficient buffers are always available for incoming data. Thus flow control designs are significantly simplified.

InfiniBand Range Extension with Obsidian Longbows

Obsidian Longbows [8] primarily provide range extension for InfiniBand fabrics over modern 10 Gigabit/s Wide Area Networks (WAN). They communicate using IPv6 Packets over

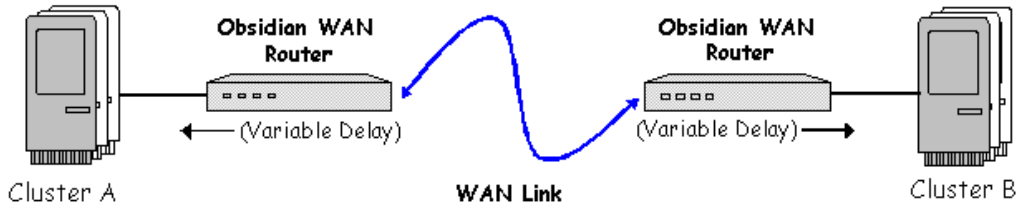

Fig. 2. Cluster-of-Clusters Connected with Obsidian Longbow XRs

SONET, ATM, and 10 Gigabit Ethernet. 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 2. 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.

Distance Emulation using Obsidian Longbows: Typically, a delay of 5 us is expected per each km 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.

III. Performance Limitations of Existing FTP Mechanisms

In this section we briefly discuss the main limitations[1] in the communication performance of popular FTP designs: (i) the standard FTP over TCP/IP, (ii) FTP over UDP and (iii) FTP over SCTP.

FTP over TCP/IP: TCP/IP based data transport imposes certain well known limitations on raw communication performance. The fundamental limitations in the current context are as follows: (i) TCP's inability to achieve good bandwidth for high-delay, high-bandwidth links, (ii) high CPU utilization required for TCP processing and (iii) multiple memory copies causing high memory bandwidth utilization and increased end to end latency. These limitations are inherited by all FTP transfers based on TCP as well.

In order to address the first limitation, researchers have proposed a multi-stream approach. Such an approach requires multiple TCP connections, requiring additional resources at the server, thereby limiting the scalability of the server. Further, multi-stream approaches (along with UDP based data transfers) are considered *"TCP Unfriendly"* [13]. The use of TCP Offload Engines (TOE) have been proposed to alleviate the second limitation. However, this approach still incurs overheads including multiple memory copies and hence higher end-to-end latencies.

[1]It is to be noted that researchers have identified certain limitations such as having a separate control and data connection, sequential nature of command exchanges, etc. in the FTP standard. These issues are orthogonal to our approach and will not be addressed in this paper. The main focus of our approach is to alleviate communication performance overheads.

FTP over UDP/IP: UDP based data transfers suffer from both (i) high CPU utilization and (ii) high memory bandwidth utilization due to multiple memory copies. However, UDP based transfers are capable of achieving higher bandwidths as compared to TCP due to lack of flow control and congestion control for UDP transfers. FTP mechanisms based on UDP inherit all its limitations.

FTP over SCTP/IP: Data transfers using SCTP (Stream Control Transmission Protocol)[2] incur limitations similar to the those mentioned above for TCP/IP. The primary benefit of utilizing SCTP is the capability of managing multiple streams without much of the overhead as seen for TCP. In particular, high bandwidth through multiple streams can be achieved with fewer connections. However, FTP based on SCTP also suffers from the above mentioned CPU utilization overhead and memory copy overheads.

In this work, we address these limitations by leveraging the zero-copy capabilities of modern interconnects. Our design alternatives and optimizations are presented in the following section.

IV. High Performance Zero-Copy FTP

In this section we describe the details of our zero-copy design. In particular, we present our high performance data transfer service layer that is used to provide FTP services to the applications. In the following subsections we present the main design alternatives followed by the design details of the overall FTP library.

A. Design Alternatives

In order to enable zero-copy data-transfers, we consider the following two alternatives: (i) Memory semantics using RDMA and (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 data-transfer is initiated from the sender side by specifying the precise destination target buffer address. It is known that RDMA based approaches [14] achieve better latencies. However, RDMA based approaches have two significant drawbacks. Firstly, the target RDMA buffers need to pre-allocated, registered and the information (including addresses and memory registration keys) need to be communicated to the source process before each buffer can be used. Further, the flow control for the data-transfer using RDMA is very explicit. i.e. the sender can initiate data-transfers only after

explicit notification of buffer availability on the remote target. Secondly, since RDMA does not involve remote node CPU in data-transfer, notifying the completion of each data-transfer requires additional mechanisms. In particular, additional messages based on send-recv (or RDMA-write with immediate) are needed for handling the remote notification, which adds a further overhead. And finally, in the case of WAN links, the latency benefits of RDMA seen for small messages are dominated by the actual network delay. Hence RDMA does not see any specific benefits over send-recv in WAN scenarios. These issues present critical constraints for data-transfers over WAN links.

On the other hand, send-recv based mechanisms show 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 in the FTP buffers, which can then be written to the destination file. Secondly, send-recv mechanisms present opportunities for enabling flow control mechanisms. For example, with the use of SRQ [5], the receiver can post buffers when needed automatically. Such a capability eliminates the need for strict (or explicit) flow control for the data-transfers. This benefit can be quite significant on WAN links because the sender is not throttled due to lack of buffers on the remote node. In addition, as mentioned in Section II-B the InfiniBand's send/recv communications can be used over both the RC and UD transports. Due to these benefits, we utilize channel semantics (send/recv) for all WAN communication protocols[2].

B. Advanced Data Transfer Service (ADTS)

In order to provide robust and efficient data transfers over modern WAN interconnects, in this paper we propose the Advanced Data Transfer Service (ADTS). Figure 3 shows the main components of the ADTS layer.

As mentioned earlier, several high performance clients can utilize zero-copy mechanisms for data transfers. Once the type of channel to be used is negotiated by the client and the server, the *Data Connection Management* component shown in Figure 3, initiates a connection to the remote peer (based on the negotiated PORT/PASV/TPRT commands) on either the zero-copy channel, the TCP/IP channel or the UDP/IP. Transport channels can dynamically be selected by the ftp server processes on a per 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 Advanced Transport Interface provides scope for enabling support for other emerging transport protocols for next generation architectures.

We present our zero-copy channel design and the possible optimizations in the following sections.

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

1) Zero-Copy Channel: Once the client and server negotiate the use of zero-copy (detailed in Section IV-C1) and the appropriate connections are setup by the *Data Connection Management* component, the channel is marked for zero-copy data transfers. However, in order to utilize zero-copy operations (send/recv), the client and the server 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. This requirement adds a significant overhead to the actual data transfer. In order to alleviate this problem, the *Buffer and File Management* component keeps a small set of pre-allocated buffers. The data that needs to be transmitted is first read into these buffers while additional buffers are being allocated and registered as needed. Once the data is ready and the buffers are registered, the data transfers are initiated from these buffers. The buffers that are allocated on-demand are unregistered and released on completion of the complete data transfer.

Unlike the sockets interface where the underlying kernel TCP/IP stack performs the flow control, the ADTS needs to perform flow control for the buffers being sent out. 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 (using SRQ) as mentioned earlier in Sections II-B and IV-A. This enables the ADTS to push the data out of the source node at a high rate.

In certain scenarios, the end nodes involved in the data transfer might not be capable of processing/storing the incoming data at a good rate[3]. In these scenarios, it is necessary to throttle the sender. In order to deal with these cases, we include a flow control fall back mechanism to explicitly throttle the sender as needed.

2) Performance Optimizations: In order to optimize the performance of data transfers over ADTS, we present the following optimizations: Persistent Sessions and Memory Registration Cache, Pipelined Data Transfers.

Persistent Sessions and Memory Registration Cache: While we utilize a set of pre-allocated buffers to speed up processing, these buffers need to be registered for each use. This in turn impacts the performance. Existing RDMA based libraries such as MVAPICH [18], amortize the registration cost by avoiding multiple registration/deregistration calls for multiple transfers by using the same buffers. This technique of maintaining registered buffers is popularly known as registration cache.

In typical FTP data transfers, each transferred file is transmitted on a different data-connection. Due to this, memory registration caching would not help significantly in our case. Further, such an approach would also incur multiple data connection setup costs as well.

In order to alleviate these multiple data-connection costs, in our design we enable persistent data-sessions that keep data connection and the related buffer associations alive during the transfer of multiple files. The maximum number of files to

[3]Please 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.

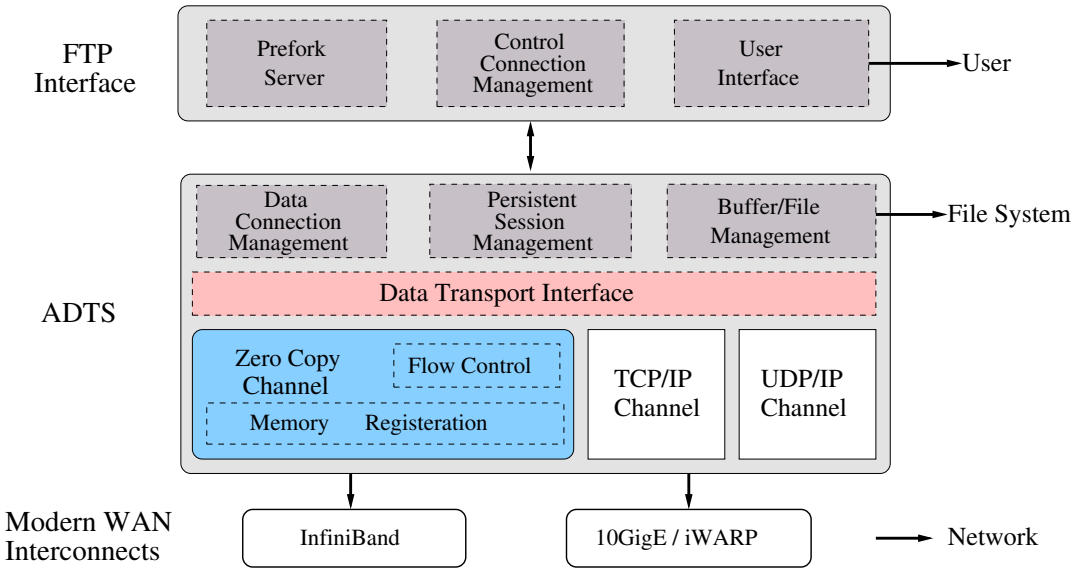

Fig. 3. Overview of the Proposed ADTS Architecture

be transmitted on a given connection is negotiated in advance and the connection is not closed until the specified number of files are transferred or the connection becomes idle. This approach also allows for an efficient use of buffers and memory registrations which boosts the performance significantly.

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

C. Design of FTP-ADTS

Figure 3 shows the basic architecture of our FTP client/server library (*FTP-ADTS*). In our approach we utilize the low-overhead zero-copy ADTS layer to provide high performance FTP transfers. The FTP Interface deals with the rest of the features needed for the FTP library. This interface provides a basic client user interface to enable all client interactions. The other main components are described in the following sections.

1) FTP Control Connection Management: Based on the user information, the client FTP engine initiates a sockets based control connection to the remote FTP server. This connection is used to relay all control information such as FTP commands and error messages. In addition, it is also used to negotiate the transport protocols and modes to be used for the data-transfers. In particular, the client negotiates with the server on Active/Passive (PORT/PASV commands in the FTP specifications [19]) mode connections.

Further, in our FTP client/server library we negotiate the use of zero-copy channel (or TCP/UDP channels) as well. Hence, clients that are capable of zero-copy transfers with the servers can benefit from higher performance. To enable this negotiation, we require the zero-copy enabled 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) Parallel Prefork Server: Multiple parallel accesses to the FTP server is a common scenario in most large datacenters. 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 that handle the client requests. In order to handle bursts of requests, our server maintains a small pool of pre-forked processes that handle burst of requests efficiently.

V. Experimental Results

In this section, we present the experimental results demonstrating the capabilities of our design. We evaluate the performance of our FTP designs in both LAN and WAN scenarios. We further measure the overheads of our approach and compare them with existing popular approaches to analyze the performance and scalability aspects of our design.

In a typical scenario, the primary bottleneck for large file transfer is disk I/O. In order to demonstrate the performance benefits of our design, we use RAM disks for all data storage purposes. Please note that modern HEC systems usually employ the use of high performance parallel or distributed file systems and advanced data storage technologies such as RAID, to obtain improved I/O performance.

Experimental Setup: In our experiments we use a cluster consisting of 64 Intel Xeon Quad dual-core processor nodes with 6GB RAM. The nodes are equipped with IB DDR ConnectX HCAs with OFED 1.3 [10] drivers. The OS used was RHEL4U4. These nodes are also equipped with Chelsio T3b 10 Gigabit Ethernet/iWARP adapters. We use the GridFTP and

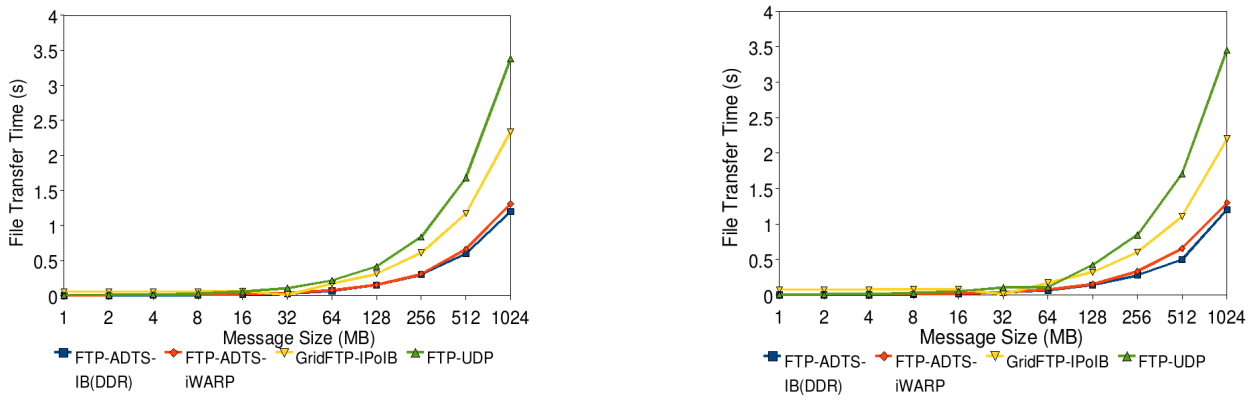

Fig. 4. File Transfer Time in LAN: (a) FTP (get) and (b) FTP (put)

FTP-UDP (FTP using the UDP/IP channel in ADTS) as the base case for our performance comparisons. The connectivity for TCP and UDP are provided by IPoIB-RC and IPoIB-UD, respectively. The nodes in the cluster are divided into *Cluster A* and *Cluster B* and are connected with Obsidian InfiniBand WAN routers as shown in Figure 2.

A. Performance of FTP in 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 scenarios. We evaluate the file transfer time of both file *put* and file *get* operations.

GridFTP and FTP-UDP are used for the TCP/IP and UDP/IP cases, respectively, and FTP-ADTS is used for the zero-copy cases (IB-DDR[4] and iWARP).

Figure 4(a) compares the client-perceived file transfer time of *get* operation with varying file sizes. We can see that our design (FTP-ADTS) achieves significantly better performance for larger file sizes. The FTP-ADTS over IB presents an improvement of by up to 95% and 181% as compared to GridFTP and FTP-UDP, respectively. Similarly FTP-ADTS over iWARP also outperforms these two cases by huge margins. This is expected since the zero-copy operations that FTP-ADTS employs has much lower latency than the IPoIB operations. We also observe that GridFTP 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 [22]. We can observe the similar trends for *put* operations as shown in Figure 4(b). Also, note that the performance of IPoIB itself limits the performance of the FTP operations using it. GridFTP and FTP-UDP are capable of saturating the available bandwidths in LAN scenarios, however, at the cost of high CPU utilization and memory bandwidth usage. We study this aspect further in the following sections.

B. Performance of FTP in WAN Scenarios

In order to study the impact of our design in WAN scenarios, we evaluate the performance of basic 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 (5*us* corresponds to one *km* of wire length).

1) 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, 1000 us and 10000us (corresponding to the distance of 0 km, 2 km, 20 km, 200 km and 2000 km). Figure 5 presents the time for transferring a 32 MByte file and a 256 MByte file. It is obvious that our FTP outperforms the GridFTP and FTP-UDP especially for data transfers over high delay networks. We observe that the *FTP-ADTS* sustains performance for larger WAN delays quite well, while the GridFTP shows a steep latency increase when the WAN delay is 10000 us. In the high delay scenario, our FTP delivers six times better performance, which shows significant promise for our FTP-ADTS design. The improvement is not only due to the underlying zero-copy operations being faster than the TCP/UDP, but also because the network throughput is the bottleneck for IPoIB over WAN, where issues such as RTT time, MTU size and buffer size can severely impact the performance. Another interesting observation is that here the FTP-UDP performs better than GridFTP, which is contrary to the results in the LAN scenario as shown earlier. This is due to the well-known trait that UDP can achieve good performance on high-bandwidth, high-latency networks in which TCP has fundamental limitations [11]. Due to the space constraints in the paper, we only show the performance of *get* operation. FTP *put* operations also show similar performance.

2) Detailed Analysis: In order to demonstrate the fundamental reasons that explain the above observations, we further carried out the following two experiments.

The first experiment is to measure the peak bandwidth of the WAN link with different transmission protocols. Figure 6 shows the results of IPoIB (including TCP/IP and UDP/IP) and verbs (including verbs-RC and verbs-UD) with increasing network delays. We can see that the IB verbs achieves the highest bandwidth and it sustains very well through the whole range of delays (WAN distance), while the TCP/IP bandwidth drops fast with the increasing delay, which in turn jeopardizes the GridFTP performance. On the other hand, although the UDP/IP bandwidth with smaller delays is lower than the TCP/IP bandwidth, it shows no significant degradation as the

[4]It should be noted that currently the Obsidian Longbows can only support the packets at SDR rate, However, in LAN scenarios our cluster is capable of DDR speeds.

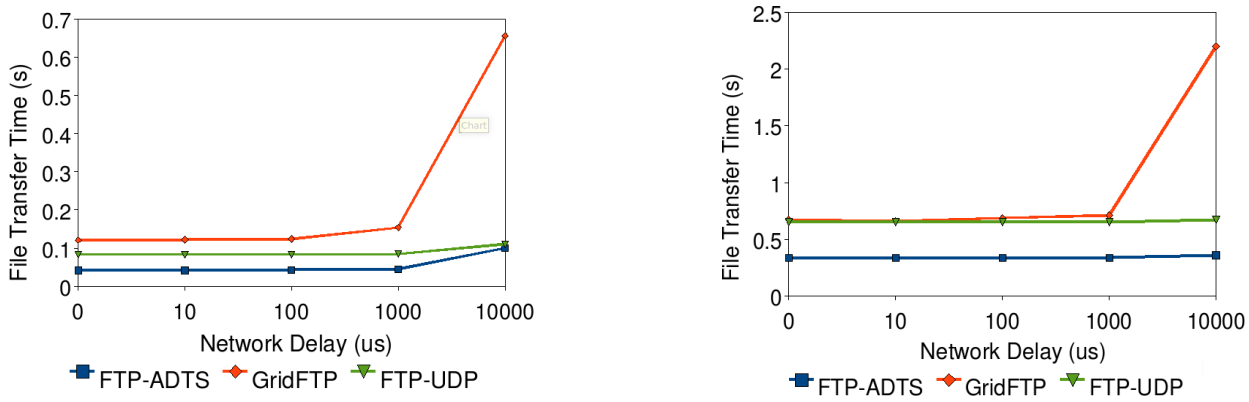

Fig. 5. File Transfer Time in WAN (get): (a) 32 MBytes and (b) 256 MBytes

delay increases, which even makes it better than TCP/IP when the delay is high (> 10000 us). This is because that UDP can swamp the network by firing many packets, but TCP is limited by the congestion control and flow control that constrains it from using the available bandwidth in high bandwidth, high latency scenarios. (Please note that researchers have shown that TCP bandwidth over longer pipes can be improved by using techniques such as multiple parallel streams. While this improves the bandwidth performance of the application, 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 transmitted packet size on bandwidth performance. Usually, larger packet sizes are preferred as they can make more use of the available link bandwidth. We claim that one of the reasons for the benefits of our design is that very large packet size (i.e. 1 MByte) can be used in the zero-copy send-recv (or RDMA) based approaches. Comparatively, the largest packet size that IP can use is 64 KByte (IPv4). In this experiment, we vary the packet size of IB RC Verbs bandwidth test (which is used in our design) with different WAN delays. From the results shown in Figure 7, we observe that the bandwidth for small and medium messages is progressively worse with increasing network delays. i.e. in order to leverage the high bandwidth capability of the IB WAN connectivity under higher network delays, larger messages need to be used. This demonstrates that some of the benefits of our design can be attributed to the use of large packet sizes. Further, we also show that IB WAN is currently not ideally suited for IP traffic (IPoIB), especially over high delay links.

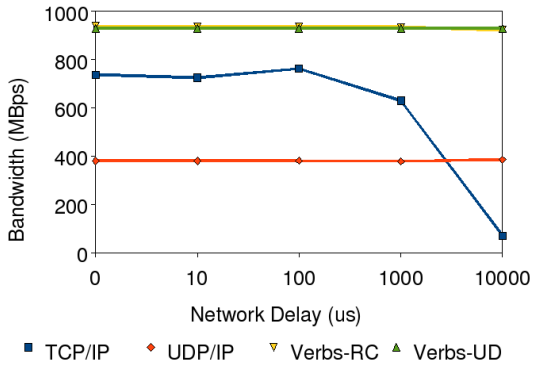

Fig. 6. Peak Network Bandwidth with Increasing WAN Delay

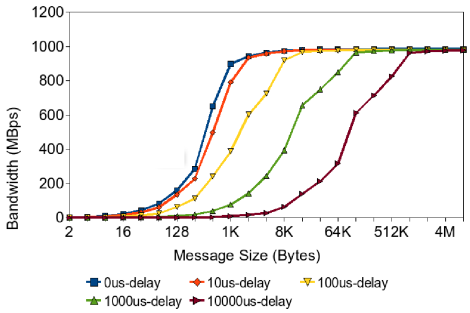

Fig. 7. InfiniBand RC Bandwidth with Increasing WAN Delay

C. CPU Utilization

TCP and UDP based communication libraries often suffer from the added CPU utilization for TCP(UDP)/IP stack processing. In basic implementations the FTP sender reads the data into its buffers and then sends this using the sockets interface. This would then be copied into the kernel's socket buffer before being sent out to the network. Similarly, the receiver would have to get this data into its kernel's internal socket buffers, then into its own buffers before it can be written to the destination disk. While TCP-based FTP implementations optimize the sender side overhead by utilizing the *sendfile* system call, the receiver side overhead is usually unavoidable.

Figures 8 (a) and (b) show the server and client CPU utilization, respectively, of the FTP-ADTS, GridFTP and FTP-UDP while performing multiple back-to-back large file *put* operations. The y-axis shows a normalized CPU utilization metric that represents the total percentage of CPU time being used on our 8-core system.

As expected, the GridFTP and FTP-UDP utilizes a significant amount of CPU on both the server and client for performing the two copies. On the other hand, our ADTS based approach has much lower CPU utilization, due to the use of zero-copy protocol which eliminates the need for additional copies on both the sender and the receiver. Further, we observe that the CPU utilization of GridFTP client is significantly

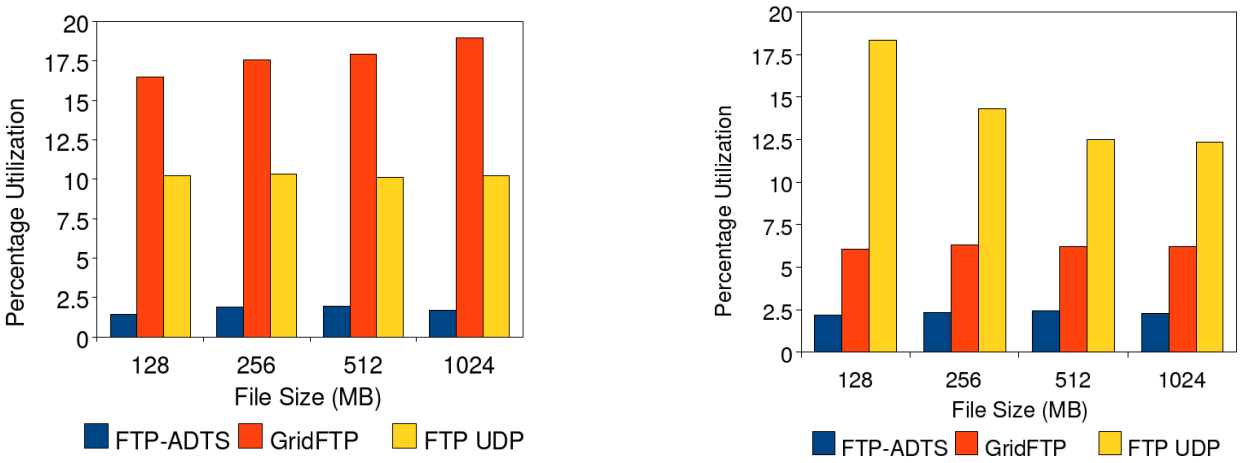

Fig. 8. End Node CPU Utilization (a) Server; (b) Client

lower. This demonstrates the benefits of using *sendfile* to reduce one memory copy on the client side. FTP-UDP does not show such trends as it can not use this optimization in UDP. Overall, our approach requires fairly smaller amounts of CPU time for all file sizes on both the server and client. This shows that our zero-copy ADTS design requires less CPU per request even at very high data rates and hence is more scalable than the corresponding IPoIB based designs.

D. Performance of Multiple File Transfers

In several scenarios such as site replication, mirroring, etc. FTP is used to transfer multiple individual files. Such scenarios present opportunities for several performance optimizations (including ones presented in Section IV-B2). In this section, we present the following two experiments: (i) FTP performance for content replication and (ii) Analysis of performance benefits due to ADTS optimizations.

FTP Performance for Content Replication: In order to demonstrate the benefits of our design, we measure the performance of FTP-ADTS and FTP-UDP using a zipf [23] trace. This trace has a high α value with an average file size of about 66 MB. The average amount of time taken to replicate these traces over WAN is shown in Figure 9. We can see that the FTP-ADTS speeds up the data transfer by up to 65%. This demonstrates that the FTP-ADTS is a promising candidate for FTP design for the zero-copy-enabled networks especially in the long-distance WAN scenarios. These benefits are in addition to the CPU benefits mentioned earlier. We observe that the total transfer time in both cases increases for very large network delays. This is due to the fact that the zipf trace used consists of a large number of requests for smaller sized files.

Benefits of the Proposed Optimizations: In this experiment 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 10 shows this breakup of the per transfer performance for FTP-ADTS with the following two cases: (i) *Basic*: with all optimizations disabled and (ii) *Opt*: with all optimizations enabled.

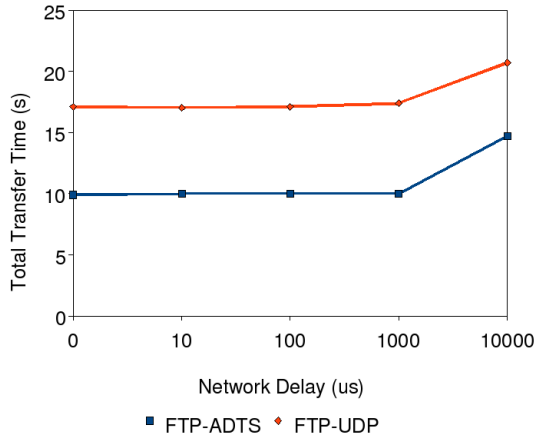

Fig. 9. Site Content Replication using FTP

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, designing distributed anonymous FTP and extendibility [12], [16]. The extension of supporting IPv6 address and transfer files traversing Network Address Translators (NATs) is introduced in [15]. In [3], the authors have proposed GridFTP which performs efficient TCP based transfers including the use of multiple streams for each transfer. Also, scientists aimed to improve the multiple file transfers using SCTP multistreaming, parallel transfers, [13] etc. The use of UDP based transfers has been explored in [20], [6], in order to overcome some of the limitations in TCP.

On the other hand, researchers have explored the use of the advanced features of modern interconnects. In [9], the authors

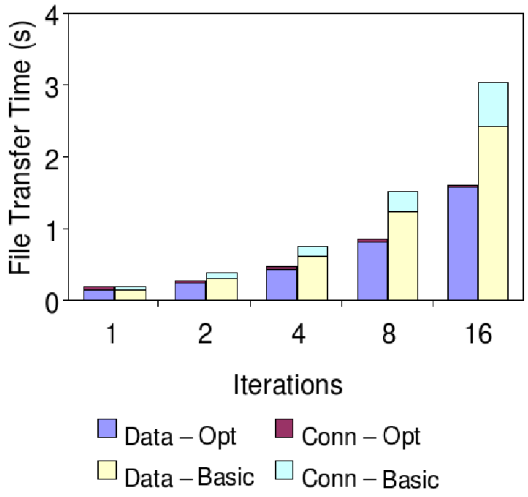

Fig. 10. Benefits of ADTS Optimizations

have designed a dynamic Apache module enabling RDMA based transfers to accelerate the web protocols and improve the throughput and CPU utilization of the server. Recent work on NFS over RDMA demonstrates better performance [21] due to the benefits of RDMA in several scenarios.

Encouraged by these successful trials, we investigate the current FTP design and propose the use the new features such as zero-copy data transfer operations for improving FTP capabilities as well. In this paper, we address the performance limitations of existing TCP/UDP/SCTP-based communication protocols in high-end WAN scenarios, by leveraging these features of modern WAN interconnects to enable efficient FTP operations.

VII. Conclusions

FTP has been the most popular method to transfer large files for data-staging, replication, and the like. While existing FTP mechanisms have improved gradually with newer networks, they still inherit the fundamental limitations imposed by the underlying networking protocol TCP/IP. This includes limitations on the achievable bandwidth and the amount of CPU utilization that TCP/IP causes on the end-nodes, thereby limiting both the performance and scalability of such systems. On the other hand modern WAN capable interconnects such as InfiniBand WAN and 10 Gigabit Ethernet/iWARP have enabled the use of high performance RDMA capabilities in these scenarios. In this paper we have presented an efficient zero-copy Advanced Data Transfer Service (ADTS) that has enabled a high performance FTP design (FTP-ADTS) capable of efficiently transferring data across WANs. Further, we reduced the CPU utilization required for the data-transfers and demonstrated significantly higher FTP server scalability.

In our experimental results, we have observed that our *FTP-ADTS* design outperforms existing approaches by more that 80% in transferring large amounts of data. In addition, we have utilized the WAN emulation capabilities of Obsidian InfiniBand WAN routers to study the impact of our designs in a wide range of WAN scenarios. We also observed that our approach achieves peak transfer rates at significantly lower (up to 6 times) CPU utilization.

Our studies have demonstrated that the IB WAN-based solutions are highly beneficial in WAN scenarios as well and can enable designing of next generation high performance parallel and distributed environments in a radically different manner. As future work we intend to explore network related performance challenges in other parallel and distributed computing middleware and study the impact of modern WAN interconnects in those scenarios.

VIII. Acknowledgments

This research is supported in part by DOE grants #DE-FC02-06ER25749 and #DE-FC02-06ER25755; NSF grants #CNS-0403342 and #CCF-0702675; and equipment donations from Intel, Mellanox and Obsidian.

References

[1] Secure File Transfer Protocol (SFTP). `www.openssh.com`.
[2] Stream Control Transmission Protocol (SCTP). `www.sctp.com`.
[3] W Allcock. GridFTP: Protocol Extensions to FTP for the Grid. Global Grid ForumGFD-R-P.020,2003.
[4] M. Allman and S. Ostermann. Multiple Data Connection FTP Extensions. Technical report, Ohio University, 1996.
[5] Infiniband Trade Association. http://www.infinibandta.org.
[6] Dennis Bush. UFTP. http://www.tcnj.edu/ bush/uftp.html.
[7] RDMA Consortium. http://www.rdmaconsortium.org/home/draft-recio-iwarp-rdmap-v1.0.pdf.
[8] Obsidian Research Corp. http://www.obsidianresearch.com/.
[9] D. Dalessandro and P. Wyckoff. Accelerating web protocols using rdma. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA)*. IEEE Press, 2007.
[10] Open Fabrics Enterprise Distribution. http://www.openfabrics.org/.
[11] Phil Dykstra. Gigabit Ethernet Jumbo Frames. http://sd.wareonearth.com/ phil/jumbo.html.
[12] F. Anklesaria et.al. The Internet Gopher Protocol (a document search and retrieval protocolf). RFC 1436. Network Working Group, Mar. 1993.
[13] Sourabh Ladha and Paul D. Amer. Improving Multiple File Transfers Using SCTP Multistreaming. In *IEEE International on Performance, Computing, and Communications Conference*, April 2004.
[14] Jiuxing Liu, Jiesheng Wu, Sushmitha P. Kini, Pete Wyckoff, and Dhabaleswar K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. In *17th Annual ACM International Conference on Supercomputing*, June 2003.
[15] S. Ostermann M. Allman and C. Metz. FTP Extensions for IPv6 and NATs. RFC 2428. Network Working Group, Sep. 1998.
[16] C. Solutions M. Horowitz and S. Lunt. FTP Security Extensions. RFC 228. Network Working Grou, Oct., 1997.
[17] Net NX 5010 High Speed Exchange. http://www.net.com/products/products-nx.shtml.
[18] MVAPICH2: High Performance MPI over InfiniBand and iWARP. http://mvapich.cse.ohio-state.edu/.
[19] J. Postel and J Reynolds. File Transfer Protocol. RFC 959. Internet Engineering Task Force. 1985.
[20] Karen R. Sollins. The Trivial File Transfer Protocol – TFTP 2 RFC 1350. July, 1992.
[21] Inc. Sun Microsystems and The Ohio State University. NFS over RDMA Design, Version 1.1. Aug, 2007.
[22] W. Allock, J. Bresnahan, R. Kettimuthu, and M. Link. The Globus Striped GridFTP Framework and Server. In *Super Computing, ACM Press*, 2005.
[23] George Kingsley Zipf. Human Behavior and the Principle of Least Effort. Addison-Wesley Press, 1949.