Efficient Zero-Copy Noncontiguous I/O for Globus on InfiniBand

Weikuan Yu *† Yuan Tian* Jeffrey S. Vetter†

Dept. of Computer Science*
Auburn University
{wkyu,tianyua}@auburn.edu

Computer Sci. & Mathematics†
Oak Ridge National Lab
vetter@ornl.gov

Abstract

Noncontiguous I/O access is one of the main access patterns in parallel and distributed applications. An I/O architecture EXIO enables Globus, a popular run-time environment for distributed computing, on RDMA networks such as InfiniBand. In this paper, we investigate the benefits of InfiniBand zero-copy RDMA to noncontiguous I/O on Globus. Our experimental results demonstrate that, by enabling zero-copy RDMA on InfiniBand, EXIO significantly improves the performance of Globus noncontiguous I/O. Compared to the packing and unpacking, zero-copy RDMA improve the bandwidth by up to 2.7 times. Compared to both IPoIB and 10GigE, it increases the bandwidth by more than three times. While achieving efficient noncontiguous I/O, RDMA-based noncontiguous I/O on InfiniBand also leads to dramatical reduction of CPU utilization on Globus clients and servers.

1 Introduction

Many geographically distributed high-performance computing systems and data centers are producing and/or supplying large volumes of data sets. Such data often must be transported to storage, visualization and analysis systems that are remotely located. The traditional approach of utilizing TCP/IP tools such as GridFTP [22] and bbcp [2] for data movement on the wide-area network (WAN) requires sophisticated per-connection optimizations. Globus [9], as a popular grid-based run-time environment, uses GridFTP that extends FTP protocols for data movement on high-bandwidth wide area networks. With more communication protocols being integrated, an extensible Input/Output (XIO) system has evolved as the I/O abstraction layer of Globus. Recently, we have enabled an RDMA-based data movement protocols EXIO [26] for the Globus run-time system.

Non-contiguous data communication is a typical pattern in scientific applications. However, the bandwidth performance on non-contiguous data can be as low as 10% of the achievable bandwidth with contiguous data. Thus efficient handling of these data types is desirable to the performance of these scientific applications. Traditionally, non-contiguous data communication is handled using an array of I/O vectors. For example, TCP/IP provides a pair of vector send/receive interfaces, i.e., readv/writev. To put the actual data on the wire, the underlying communication protocol either packs the data into a contiguous memory region or repetitively sends the entire data as separated contiguous fragments. Because of its importance, some cluster interconnects provide native support for non-contiguous scatter/gather data communication. For example, Myrinet provides scatter/gather support for its IP emulation stack in the firmware that runs on the Myrinet Network Interface Cards (NICs); InfiniBand supports scatter/gather in its NIC hardware. The scatter/gather support provides multiple advantages: (a) they provide a convenient non-contiguous data communication interface to the higher level programming libraries, (b) they pass down more structural information of the data to the lower-level communication systems, which allows the low level to take advantage of the best mechanisms available.

EXIO [26] is an I/O architecture that extends Globus on RDMA networks such as InfiniBand. In this paper, we investigate the benefits of InfiniBand zero-copy RDMA to noncontiguous I/O for Globus. Our experimental results demonstrate that, by enabling zero-copy RDMA on InfiniBand, EXIO significantly improves the performance of Globus noncontiguous I/O. Compared to IPoIB and 10GigE, it increases the bandwidth by more

than three times. While achieving efficient noncontiguous I/O, RDMA-based noncontiguous I/O on InfiniBand also leads to dramatical reduction of CPU utilization on Globus clients and servers.

The rest of the paper is organized as follows. We describe the design of EXIO in Section 2. Then we describe the implementation of zero-copy noncontiguous I/O in Section 3. Experimental results are provided in Section 4. Section 5 discusses related work. Finally, we conclude the paper in Section 6.

2 Extended XIO (EXIO) and Contiguous Data Movement

EXIO is designed as an extended XIO framework that can enable Globus on RDMA networks. Originally, Globus [9] Extensible Input/Output (XIO) [1] maps application I/O requests to individual drivers without introducing extra memory copies. However, XIO drivers, such as TCP [1] and UDT [10, 3], make use of legacy TCP/IP-based communication protocols, which internally employ kernel-level memory copies. To integrate RDMA for high-performance zero-copy I/O in the Globus Toolkit, EXIO has implemented the following capabilities: (a) extending XIO to match the communication characteristics of RDMA networks; (b) establishing RDMA connections inside EXIO; and (c) managing the progress of RDMA communication protocols. In addition to supporting RDMA networks, EXIO retains the other drivers from its precursor XIO. Therefore, it is capable of supporting data movement on top of the legacy TCP/IP communication protocols.

2.1 EXIO Software Architecture

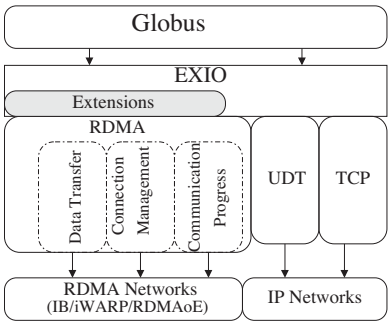


Figure 1. Software Architecture of EXIO

Figure 1 shows the software architecture of EXIO with these extensions. EXIO is designed to work with all RDMA networks including InfiniBand [11], iWARP [19] and RDMAoE (RDMA over Ethernet) [21]. The RDMA driver groups functionalities of an RDMA network into three categories: connection management, communication progress, and data transfer.

Instead of using system file descriptors to represent and manage network connections, such as TCP/IP sockets, EXIO provides extensions to support RDMA-based connections. It removes the existing dependence of XIO on system file descriptors, and introduces new handles to identify network connections based on the underlying networks. In addition, EXIO is expanded with functionalities to invoke different callback routines for these networks. For example, for events from a RDMA network, a callback routine from the RDMA driver will be called by EXIO to process RDMA events without EXIO knowing the internal details of an RDMA network. Connection management and communication progress are more generically designed, intended to work with RDMA-compliant [18] networks.

2.2 Contiguous Data Movement in EXIO

Based on the aforementioned design of EXIO, we have implemented Globus on InfiniBand. This implementation is based on the Globus Toolkit version 4.1.3. It is developed on top of the latest versions (1.3 and 1.4) of OFED (OpenFabrics Enterprise Distribution). We first describe the protocols for contiguous data movement in EXIO.

A number of fixed-size, registered memory buffers are allocated for data communication. InfiniBand requires data communication to fall in registered memory regions. To save the registration cost, we enable a registration cache for memory regions. The left diagram of Figure 2 shows the transfer of a short message. From the source memory of the sender, a short message is first copied into a buffer and then sent across the network. At the receiver side, it is received into a pre-posted receive buffer. The incoming message remains in the receive buffer, until the receiver has posted a matching receive operation. Then the message is copied into the destination memory at the receiver. This is also called the eager protocol because the sender does not wait for the receiver to become ready on the destination memory.

A *rendezvous* protocol is designed to transfer long messages in a zero-copy manner using RDMA write operations. As shown by the right diagram of Figure 2, for a long message (a message that is longer than a single memory buffer), a *rendezvous* request (*req*) is sent to the receiver. The receiver returns an acknowledgment (*ack*) to the sender when it matches a receive operation with this request. The sender then sends the long message

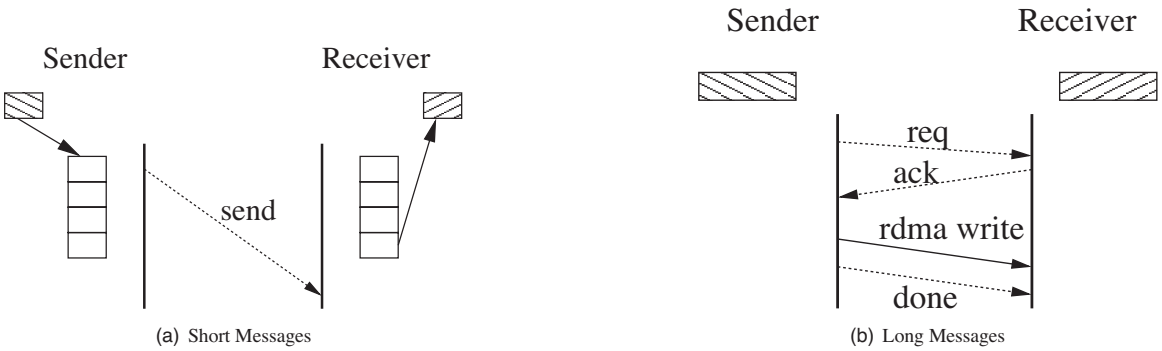


Figure 2. Contiguous Data Movement of EXIO on InfiniBand

with a RDMA write operation. At the completion of RDMA write, a *done* message is sent to the receiver for the completion notification of the entire message. Except for three control messages (req, ack, and done), the actual data for a long message is sent without any intermediate buffering through one zero-copy RDMA write operation.

3 RDMA-based Noncontiguous I/O for EXIO

InfiniBand supports scatter/gather in its NIC hardware. It is provided in both InfiniBand channel semantics and memory semantics. In the channel semantics, at the send side, InfiniBand NIC hardware packs the segments specified in a send gather descriptor on the fly and inject the data into the network; at the receiver side, NIC drains the data from the network, and correspondingly unpacks, into a list of memory segments as specified in a receive scatter descriptor. In contrast, the scatter/gather support available in InfiniBand memory semantics only operates on local non-contiguous memory. As shown in Figure 3(a), RDMA write can only gather data from non-contiguous local memory segments and write into a remote contiguous memory segment; On the other hand, RDMA read, as shown in Figure 3(b), can only pull data from a remote contiguous memory region and scatter into non-contiguous local memory segments.

EXIO was designed with two different methods for noncontiguous I/O. In the first method, noncontiguous I/O was implemented through data packing and unpacking, in which data segments are packed into a contiguous memory region on the sender side. The packed data are then sent over to another contiguous memory region at the receiver side, wherein they are scattered to the list of targeted memory segments. In the second method, EXIO takes advantage of the NIC-level support to pack/unpack non-contiguous segments on the fly. As an initial study, our implementation makes use of zero-copy RDMA write for Globus noncontiguous I/O. Figure 4 shows the diagram of Globus noncontiguous I/O using zero-copy RDMA write on InfiniBand. Since RDMA write only gathers local segments and deliver data into a single segment in the remote memory, we invoke an RDMA write operation for each destination segment. Data segments in the source memory are sequentially distributed into these destination segments. These RDMA write operations are posted together and followed by an RDMA send operation to notify the remote side about the completion of data transfer.

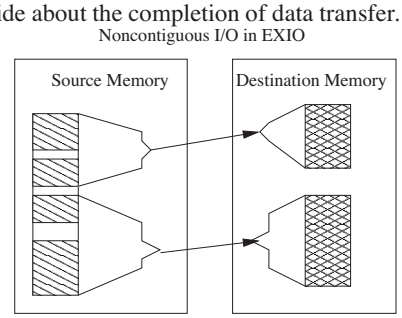
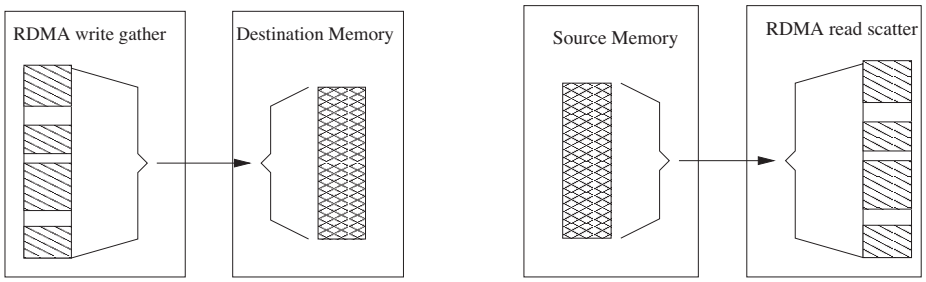


Figure 4. Globus Noncontiguous I/O using Zero-copy RDMA Write

4 Performance Evaluation

Our experiments were conducted on a cluster of nodes with 2.1 GHz 64-bit quad-core Intel Harpertown processors. Each node is equipped with 8x PCI-Express Gen 2.0 bus. These nodes run Linux 2.6.18 kernels. They are equipped with Myricom 10-Gigabit Ethernet (10GigE) cards. For InfiniBand experiments, we used a pair of nodes that are equipped with InfiniBand QDR (quad data rate) cards, which are connected to a Mel-



(a) RDMA Write with Noncontiguous Local Segments

(b) RDMA Read with Noncontiguous Local Segments

Figure 3. InfiniBand Scatter/Gather Support in its Memory Semantics

lanox 36-port QDR switch. IB QDR has a peak bandwidth of 32Gbps. The InfiniBand software stack called OpenFabrics Enterprise Distribution (OFED), versions 1.4, was used in our experiments.

4.1 Performance Benefi ts of Zero-copy RDMA Scatter/Gather to Noncontiguous I/O

We developed a bandwidth benchmark to measure the performance of Globus noncontiguous I/O operations. The data-transfer bandwidth was measured when a client (sender) sends 2000 messages to the server (receiver). Each message was divided into 16 segments, separated by 1MB apart. In all tests, 50 initial iterations were executed before measurements were taken.

Figure 5 shows the bandwidth of noncontiguous I/O in EXIO on the InfiniBand network, using two different methods as described in Section 3. Compared to the packing/unpacking method, zero-copy RDMA improves the bandwidth for noncontiguous I/O by up to three times. Due to the high demand on memory bandwidth, the packing/unpacking method was peaked at 1500 MB/sec, and then tailed off for messages bigger than 2MB. To show the effectiveness of noncontiguous I/O, we also measured the bandwidth of transferring contiguous messages of the same size. As shown in the figure, our zero-copy noncontiguous I/O is able to achieve the performance close to that of contiguous I/O. Among all the message sizes, the efficiency stays above 70%, and more than 95% for large messages.

4.2 Comparisons to IPoIB and 10-GigaBit Ethernet

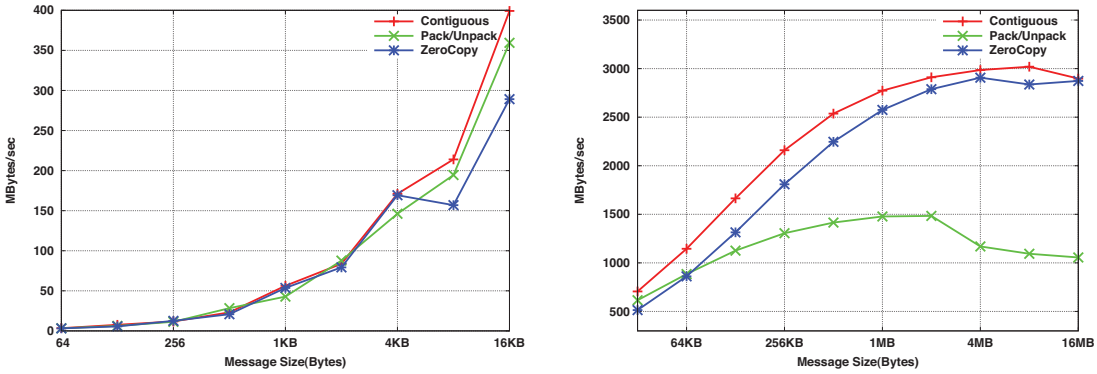
Using the same bandwidth program, we evaluated the performance of Globus on different networks. The Myricom 10GigE cards were used in this experiment. We also compared the performance of RDMA-based noncontiguous I/O to the emulated IP implementation IPoIB, using the default connected mode.

Figure 6 shows the bandwidth performance of noncontiguous I/O on different networks. 10GigE cards were enabled with interrupt coalescing for best bandwidth results. IPoIB can reach a peak bandwidth of 971 MB/sec; 10GigE of 1181 MB/sec; while RDMA improves the peak bandwidth 2906 MB/sec. For small- and mid-range messages, RDMA-based noncontiguous I/O performs worse than both 10GigE and IPoIB. This is because the costs of additional processing in RDMA.

4.3 CPU Utilization

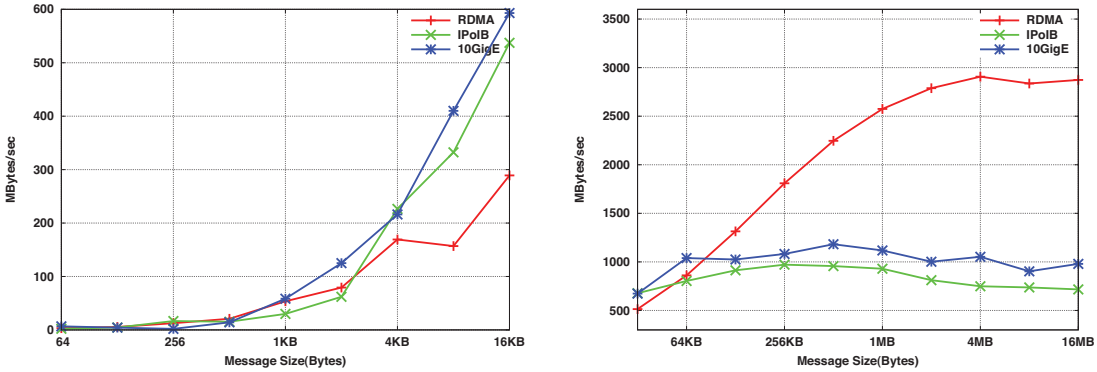
Another significant strength of RDMA is its benefit in reducing the CPU involvement in network communication. We extended the bandwidth test with the capability to report CPU utilization. To this purpose, we increased the number of messages from 2000 to 100,000. This prolonged bandwidth test does not improve the bandwidth numbers much, but it allows us to measure the CPU utilization for every message size. We recorded the TSC (time stamp counter) before and after the measurement of each message size. Then we recorded the increment of CPU utilization during the same period. The ratio of these two was taken as the percentage of CPU utilization. This measurement was conducted on both the sender side and the receiver side.

Figure 7 shows the comparison of Globus CPU utilization on top of 10GigE, IPoIB and RDMA. Figures 6 can be referred to for the corresponding bandwidth numbers. For small messages, the receiver on 10GigE and InfiniBand RDMA utilized 100% CPU or even higher (due to the use of another CPU), while the sender generally used less than 100% CPU to keep the receiver busy. In the case of IPoIB, both the sender and receiver CPUs are not fully utilized when communicating small segments for noncontiguous I/O. As the message size in-



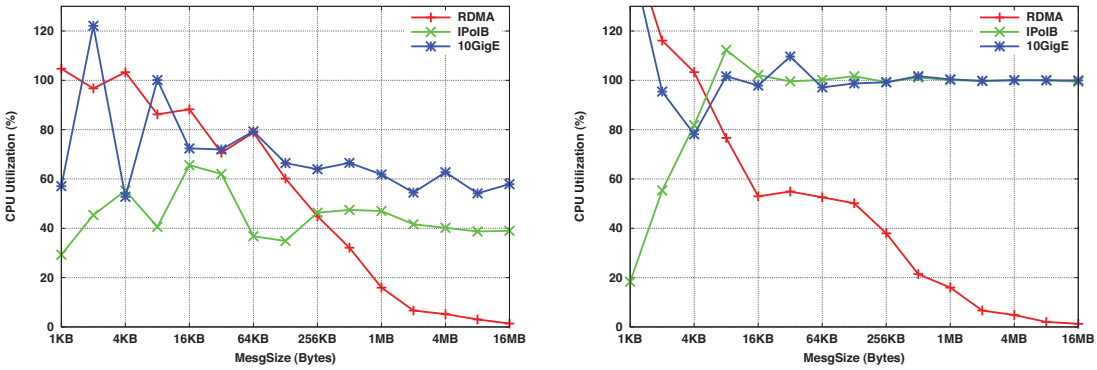
(a) Small Messages (b) Large Messages

Figure 5. Noncontiguous I/O with Zero-copy RDMA Scatter/Gather



(a) Small Messages (b) Large Messages

Figure 6. Comparison of Different Networks



(a) Sender (b) Receiver

Figure 7. Comparison of CPU Utilization

creases, the bandwidth test becomes mostly communication bound. In the cases of IPoIB and 10GigE, the network is fast enough to keep the CPU busy. The receiver side CPU is particularly saturated because of heavier processing overhead. The sender side needs only 40% and 60% CPU, respectively for IPoIB and 10GigE, to keep the receiver busy. In contrast, Globus with RDMA uses much less CPU on both the sender and the receiver sides, while at the same time it is able to transmit noncontiguous segments to achieve effective utilization of the network bandwidth (c.f. 4.1).

5 Related Work

Leveraging RDMA from high speed networks for high-performance data movement has been very popular in various programming models and storage paradigms. Liu *et al*. [14] designed RDMA-based MPI over InfiniBand. Zhou *et al*. [27] studied the benefits of VIA networks in database storage. DeBergalis *et al*. [8] implemented a file system DAFS on top of VIA. Implementations of PVFS [16] on top of RDMA networks such as InfiniBand and Quadrics were described in [23] and [24], respectively. Callaghan *et al*. [4] provided an initial implementation NFS over RDMA (NFSoRDMA) on Solaris. An implementation of NFSoRDMA was made available for Linux [20] systems. RDMA has also been exploited for data movement in SCSI-based storage protocols including efforts from academia and industry [12, 6, 7]. Our recent work [26] complements these efforts to enable RDMA for grid-oriented applications. This work continues our earlier effort to investigate the benefits of RDMA to noncontiguous I/O in Globus [9].

Enabling high-performance and grid-computing applications using RDMA recently attracted a lot of interests, both in terms of infrastructure deployment and research investigation. A group of researchers from Oak Ridge National Laboratory extensively studied the performance of InfiniBand-based communication protocols, programming models as well as storage protocols across long-distance OC192 connections on UltraScience Net [5, 25, 17]. Their work revealed the strength of InfiniBand across long distance. Narravula *et al*. [15] studied the performance of different HPC middleware across simulated long-distance InfiniBand connections. Together, these efforts revealed the performance impact of different network parameters and reliability configurations for InfiniBand on WAN. Recently, Lai *et al*. [13] designed an efficient FTP protocol for high performance data-transfer by exploiting a communication library Advanced Data Transfer Service (ADTS) over InfiniBand. We designed an extended I/O (EXIO) system [26] to integrate RDMA into the Globus environment [9]. In this work, we focus on the performance of an important I/O pattern, noncontiguous I/O.

6 Conclusions

In this paper, we examined the scatter/gather networking mechanisms of InfiniBand. Then we developed an initial RDMA write-based zero-copy support for Globus noncontiguous I/O. We evaluated the benefits of InfiniBand zero-copy RDMA compared to the packing and unpacking based method, We also compared the performance of noncontiguous I/O on InfiniBand RDMA compared to the same on 10 Gigabit Ethernet and IPoIB. Our experimental results demonstrate that, by enabling zero-copy RDMA on InfiniBand, EXIO significantly improves the performance of Globus noncontiguous I/O. It improves the bandwidth by up to 2.7 times, compared to the packing/unpacking method. The bandwidth improvement is more than three times, compared to both IPoIB and 10GigE. While achieving effective noncontiguous I/O, using InfiniBand RDMA also dramatically reduces the CPU utilization on Globus clients and servers.

In future, we plan to support Globus noncontiguous I/O through other InfiniBand network mechanisms, such as RDMA read/scatter and scatter/gather in the channel semantics. In addition, when system access becomes available, we plan to leverage long-distance RDMA networks, such as InfiniBand on UltraScience Net, to carry out a comprehensive evaluation of Globus on different networking technologies across long distances. We also plan to study adaptive mechanisms in selecting the best networking drivers for optimal performance on different networks or different combinations of them.

7 Acknowledgments

This work was funded in part by a UT-Battelle grant (UT-B-4000087150) to Auburn University, and in part by a National Science Foundation award CNS-0917137. This research was also sponsored in part by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. We are grateful for an InfiniBand equipment donation from HPC Advisor Council to Auburn University.

References

[1] W. E. Allcock, J. Bresnahan, R. Kettimuthu, and J. M. Link. The globus extensible input/output system (xio): A protocol independent io system for the grid. In *IPDPS*, 2005.

[2] bbcp. http://www.slac.stanford.edu/∼abh/bbcp/.

[3] J. Bresnahan, M. Link, R. Kettimuthu, and I. Foster. Udt as an alternative transport protocol for gridftp. In *Proceedings of the Second International Workshop on Protocols for Fast Long-Distance Networks*, 2009.

[4] B. Callaghan, T. Lingutla-Raj, A. Chiu, P. Staubach, and O. Asad. NFS over RDMA. In *Proceedings of the ACM SIGCOMM workshop on Network-I/O convergence*, pages 196–208. ACM Press, 2003.

[5] S. Cater, M. Minich, and N. Rao. Experimental evaluation of infiniband transport over local and wide-area networks. In *High Performance Computing Symposium (HPC'07)*, September 2008.

[6] M. Chadalapaka, H. Shah, U. Elzur, P. Thaler, and M. Ko. A study of iscsi extensions for rdma (iser). In *Proceedings of the ACM SIGCOMM workshop on Network-I/O convergence (NICELI)*, pages 209–219, 2003.

[7] D. Dalessandro, A. Devulapalli, and P. Wyckoff. iSER Storage Target for Object-Based Storage Devices. In *Proceedings of SNAPI workshop*, volume 0, pages 107–113, 2007.

[8] M. DeBergalis, P. Corbett, S. Kleiman, A. Lent, D. Noveck, T. Talpey, and M. Wittle. The Direct Access File System. In *Proceedings of Second USENIX Conference on File and Storage Technologies (FAST '03)*, March 2003.

[9] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, (2):115–128.

[10] Y. Gu. *UDT: a high performance data transport protocol*. PhD thesis, Chicago, IL, USA, 2005. Chairperson-Grossman, Robert L.

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

[12] M. Ko, M. Chadalapaka, , U. Elzur, H. Shah, P. Thaler, and J. Hufferd. iSCSI Extensions for RDMA Specification. http://www.ietf.org/internet-drafts/draft-ietf-ips-iser-05.txt.

[13] P. Lai, H. Subramoni, S. Narravula, A. Mamidala, and D. Panda. Designing efficient ftp mechanisms for high performance data-transfer over infiniband. In *Proceedings of Int'l Conference on Parallel Processing (ICPP' 09)*, September 2009.

[14] J. Liu, J. Wu, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. *Int'l Journal of Parallel Programming*, 32(3), June 2004.

[15] S. Narravula, H. Subramoni, P. Lai, R. Noronha, and D. K. Panda. Performance of hpc middleware over infiniband wan. In *Proceedings of Int'l Conference on Parallel Processing (ICPP' 08)*, September 2008.

[16] P. H. Carns and W. B. Ligon III and R. B. Ross and R. Thakur. PVFS: A Parallel File System For Linux Clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, October 2000.

[17] N. S. V. Rao, W. Yu, W. R. Wing, S. W. Poole, and J. Vetter. Wide-area performance profiling of 10gige and infiniband technologies. November 2008.

[18] R. Recio, P. Culley, D. Garcia, and J. Hilland. An rdma protocol specification (version 1.0), October 2002.

[19] A. Romanow and S. Bailey. An Overview of RDMA over IP. In *Proceedings of International Workshop on Protocols for Long-Distance Networks (PFLDnet2003)*, 2003.

[20] T. Talpey et. al. NFS/RDMA ONC Transport. http://sourceforge.net/projects/nfs-rdma.

[21] M. Technologies. RDMA over Ethernet (RDMAoDCB) A Technology Brief.

[22] GT 4.0 GridFTP. http://www.globus.org.

[23] J. Wu, P. Wychoff, and D. K. Panda. PVFS over InfiniBand: Design and Performance Evaluation. In *Proceedings of the International Conference on Parallel Processing '03*, Kaohsiung, Taiwan, October 2003.

[24] W. Yu, S. Liang, and D. K. Panda. High Performance Support of Parallel Virtual File System (PVFS2) over Quadrics. In *Proceedings of The 19th ACM International Conference on Supercomputing*, Boston, Massachusetts, June 2005.

[25] W. Yu, N. Rao, and J. Vetter. Experimental analysis of infiniband transport services on WAN. In *International Conference on Networking, Architecture and Storage*. 2008.

[26] W. Yu, Y. Tian, J. Vetter, N. S. Rao, T. Liu, and G. Shainer. Exio: Enabling globus on rdma networks - a case study with infiniband. Technical Report AU-CSSE-PASL/10-TR01, PASL, Dept. of CSSE, Auburn University, Auburn, AL 36849, June 2010.

[27] Y. Zhou, A. Bilas, S. Jagannathan, C. Dubnicki, J. F. Philbin, and K. Li. Experiences with VI Communication for Database Storage. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pages 257–268. IEEE Computer Society, 2002.