Zero-copy MPI Derived Datatype Communication over InfiniBand [*]

Gopalakrishnan Santhanaraman, Jiesheng Wu, and Dhabaleswar K. Panda

Department of Computer Science and Engineering
The Ohio State University
{santhana, wuj, panda}@cse.ohio-state.edu

Abstract. This paper presents a new scheme, Send Gather Receive Scatter (SGRS), to perform zero-copy datatype communication over InfiniBand. This scheme leverages the gather/scatter feature provided by InfiniBand channel semantics. It takes advantage of the capability of processing non-contiguity on both send and receive sides in the Send Gather and Receive Scatter operations. In this paper, we describe the design, implementation and evaluation of this new scheme. Compared to the existing Multi-W zero-copy datatype scheme, the SGRS scheme can overcome the drawbacks of low network utilization and high startup costs. Our experimental results show significant improvement in both point-to-point and collective datatype communication. The latency of a vector datatype can be reduced by up to 62% and the bandwidth can be increased by up to 400%. The Alltoall collective benchmark shows a performance benefit of up to 23% reduction in latency.

1 Introduction

The MPI (Message Passing Interface) Standard [3] has evolved as a *de facto* parallel programming model for distributed memory systems. As one of its most important features, MPI provides a powerful and general way of describing arbitrary collections of data in memory in a compact fashion. The MPI standard also provides run time support to create and manage such MPI derived datatypes. MPI derived datatypes are expected to become a key aid in application development.

In principle, there are two main goals in providing derived datatypes in MPI. First, several MPI applications such as (de)composition of multi-dimensional data volumes [1, 4] and finite-element codes [2] often need to exchange data with algorithm-related layouts between two processes. In the NAS benchmarks such as MG, LU, BT, and SP, non-contiguous data communication has been found to be dominant [10]. Second, MPI derived datatypes provide opportunities for MPI implementations to optimize datatype communication. Therefore, applications developed with datatype can achieve portable performance over different MPI applications with optimized datatype communication.

In practice, however, the poor performance of many MPI implementations with derived datatypes [2, 5] becomes a barrier to using derived datatypes. A programmer often prefers packing and unpacking noncontiguous data manually even with considerable effort. Recently, a significant amount of research work have concentrated on improving datatype communication in MPI implementations, including 1) *Improved datatype processing system* [5, 13], 2) *Optimized packing and unpacking procedures* [2, 5], and 3) *Taking advantage of network features to improve noncontiguous data communication* [17].

[*] This research is supported in part by Department of Energy's Grant #DE-FC02-01ER25506, and National Science Foundation's grants #CNS-0204429, and #CCR-0311542.

In this paper, we focus on improving non-contiguous data communication by taking advantage of InfiniBand features. We focus on zero-copy datatype communication over InfiniBand. Zero copy communication protocols are of increased importance because they improve memory performance and also have reduced host cpu involvement in moving data. Our previous work [17] used multiple RDMA writes, *Multi-W*, as an effective solution to achieve zero-copy datatype communication. In this paper we look at an alternate way of achieving zero-copy datatype communication using the *send/receive* semantics with the *gather/scatter* feature provided by InfiniBand. We call this scheme *SGRS* (Send Gather Receive Scatter) in the rest of this paper. This scheme can overcome two main drawbacks in the Multi-W scheme: *low network utilization* and *high startup cost*. We have implemented and evaluated our proposed SGRS scheme in MVAPICH, an MPI implementation over InfiniBand [12, 9].

The rest of the paper is organized as follows. We first give a brief overview of InfiniBand and MVAPICH in Section 2. Section 3 provides the motivation for the SGRS scheme. Section 4 describes the basic approach, the design issues involved and the implementation details. The performance results are presented in Section 5. Section 6 presents related work. We draw our conclusions and possible future work in Section 7.

2 Background

In this section we provide an overview of the Send Gather/Recv Scatter feature in InfiniBand Architecture and MVAPICH.

2.1 Send Gather/Recv Scatter in InfiniBand

The InfiniBand Architecture (IBA) [6] defines a System Area Network (SAN) for interconnecting processing nodes and I/O nodes. It supports both channel and memory semantics. In channel semantics, send/receive operations are used for communication. In memory semantics, RDMA write and RDMA read operations are used instead. In channel semantics, the sender can gather data from multiple locations in one operation. Similarly, the receiver can receive data into multiple locations. In memory semantics, non-contiguity is allowed only in one side. RDMA write can gather multiple data segments together and write all data into a contiguous buffer on the remote node in one single operation. RDMA read can scatter data into multiple local buffers from a contiguous buffer on the remote node.

2.2 Overview of MVAPICH

MVAPICH is a high performance implementation of MPI over InfiniBand. Its design is based on MPICH [15] and MVICH [8]. The Eager protocol is used to transfer small and control messages. The Rendezvous protocol is used to transfer large messages. Datatype communication in the current MVAPICH is directly derived from MPICH and MVICH without any change. Basically, the generic packing and unpacking scheme is used inside the MPI implementation. When sending a datatype message, the sender first packs the data into a contiguous buffer and follows the contiguous path. On the receiver side, it first receives data into a contiguous buffer and then unpacks data into the user buffers. In the rest of this paper, we refer to this scheme as *Generic scheme.*

3 Motivating Case Study for the Proposed SGRS Scheme

Consider a case study involving the transfer of multiple columns in a two dimensional $M \times N$ integer array from one process to another. There are two possible zero-copy schemes. The first one uses multiple RDMA writes, one per row. The second one uses Send Gather/Receive Scatter. We compare these two schemes over the VAPI layer, which is an InfiniBand API provided by Mellanox [11]. The first scheme posts a list of

RDMA write descriptors. Each descriptor writes one contiguous block in each row. The second scheme posts multiple Send Gather descriptors and Receiver Scatter descriptors. Each descriptor has 50 blocks from 50 different rows (50 is the maximum number of segments supported in one descriptor in the current version of Mellnox SDK). We will henceforth refer to these two schemes as "Multi-W" and "SGRS" in the plots. In the first test, we consider a 64×4096 integer array. The number of columns varies from 8 to 2048. The total message size varies from 2 KBytes to 512 KBytes accordingly. The bandwidth test is used for evaluation. As shown in Figure 1, the SGRS scheme consistently outperforms the Multi-W scheme.

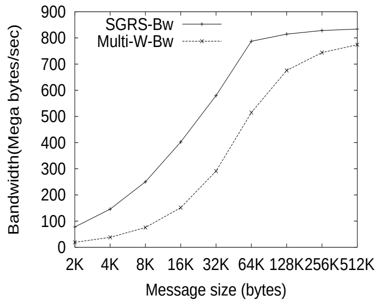

Fig. 1. Bandwidth Comparison over VAPI with 64 Blocks.

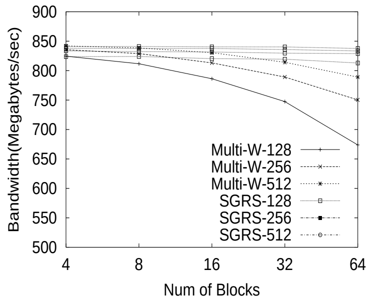

Fig. 2. Bandwidth Comparison over VAPI with Different Number of Blocks

In the second test, the number of blocks varies from 4 to 64. The total message size we studied is 128 KBytes, 256 KBytes, and 512 KBytes. Figure 2 shows the bandwidth results with different number of blocks and different message sizes. When the number of blocks is small, both Multi-W and SGRS schemes perform comparably. This is because the block size is relatively large. The network utilization in the Multi-W is still high. As the number of segments increase we observe a significant fall in bandwidth for the Multi-W scheme whereas the fall in bandwidth is negligible for the SGRS scheme. There are two reasons. First, the network utilization becomes lower when the block size decreases (i.e. the number of blocks increases) in the Multi-W scheme. However, in the SGRS scheme, the multiple blocks in one send or receive descriptor are considered as one message. Second, the total startup costs in the Multi-W scheme increases with the increase of the number of blocks because each block is treated as an individual message in the Multi-W scheme and hence the startup cost is associated with each block. From these two examples, it can be observed that the SGRS scheme can overcome the two drawbacks in the Multi-W by increasing network utilization and reducing startup costs. These potential benefits motivate us to design MPI datatype communication using the SGRS scheme described in detail in Section 4.

4 Proposed SGRS (Send Gather/Recv Scatter) Approach

In this section we first describe the SGRS scheme. Then we discuss the design and implementation issues and finally look at some optimizations to this scheme.

4.1 Basic Idea

The basic idea behind the SGRS scheme is to use the scatter/gather feature associated with the send receive mechanism to achieve zero copy communication. Using this feature we can send/receive multiple data blocks as a single message by posting a send

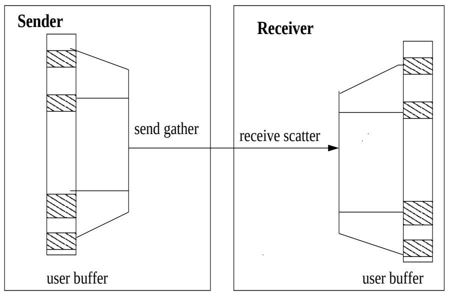

Fig. 3. Basic Idea of the SGRS Scheme.

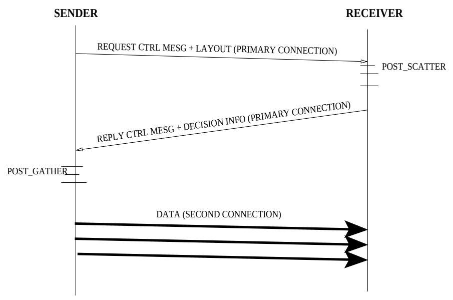
SENDER
RECEIVER
REQUEST CTRL MESG + LAYOUT (PRIMARY CONNECTION)
POST_SCATTER
REPLY CTRL MESG + DECISION INFO (PRIMARY CONNECTION)
POST_GATHER
DATA (SECOND CONNECTION)

Fig. 4. SGRS Protocol.

gather descriptor at source and a receive scatter descriptor at destination. Figure 3 illustrates this approach. The SGRS scheme can handle non-contiguity on both sides. As mentioned in Section 2, RDMA Write Gather or RDMA Read Scatter handles non-contiguity only on one side. Hence, to achieve zero-copy datatype communication based on RDMA operations, the Multi-W scheme is needed [17].

4.2 Design and Implementation Issues

Communication Protocol: The SGRS scheme is deployed in Rendezvous protocol to transfer large datatype messages. For small datatype messages, the Generic scheme is used. As shown in Figure 4, the sender first sends the Rendezvous start message with the data layout information out. Second, the receiver receives the above message and figures out how to match the sender's layout with its own layout. Then, the receiver sends the layout matching decision to the sender. After receiving the reply message, the sender posts send gather descriptors. It is possible that the sender may break one block into multiple blocks to meet the layout matching decision.There are four main design issues: Secondary connection, Layout exchange, Posting descriptors and Registration.

Secondary connection: The SGRS scheme needs a second connection to transmit the non-contiguous data. This need arises because it is possible in the existing MVAPICH design to prepost some receive descriptors on the main connection as a part of its flow control mechanism. These descriptors could unwittingly match with the gather-scatter descriptors associated with the non-contiguous transfer. One possible issue with the extra connection is scalability. In our design, there are no buffers/resources for the second connection. The HCA usually can support a large number of connections. Hence the extra connection does not hurt the scalability.

Layout exchange: The MPI datatype has only local semantics. To enable zero-copy communication, both sides should have an agreement on how to send and receive data. In our design, the sender first sends its layout information to the receiver in the Rendezvous start message as shown in Figure 4. Then the receiver finds a solution to match these layouts. This decision information is also sent back to the sender for posting send gather descriptors. To reduce the overhead for transferring datatype layout information, a layout caching mechanism is desirable [7]. Implementation details of this cache mechanism in MVAPICH can be found in [17]. In Section 5, we evaluate the effectiveness of this cache mechanism.

Posting Descriptors: There are three issues in posting descriptors. First, if the number of blocks in the datatype message is larger than the maximum allowable gather/scatter limit, the message has to be chopped into multiple gather/scatter descriptors. Second, the number of posted send descriptors and the number of posted receive descriptors must be equal. Third, for each pair of matched send and receive descriptors, the data length must be same. This basically needs a negotiation phase. Both these issues can be handled by taking advantage of the Rendezvous start and reply message in the Rendezvous protocol. In our design, the receiver makes the matching decision taking into account the layouts as well as scatter-gather limit. Both the sender and the receiver post their descriptors with the guidance of the matching decision.

User Buffer Registration: To send data from and receive data into user buffer directly, the user buffers need to be registered. Given a non-contiguous datatype we can register each contiguous block one by one. We could also register the whole region which covers all blocks and gaps between blocks. Both attempts have their drawbacks [16]. In [16],*Optimistic Group Registration(OGR)* has been proposed to make a tradeoff between the number of registration and deregistration operations and the total size of registered space to achieve efficient memory registration on datatype message buffers.

5 Performance Evaluation

In this section we evaluate the performance of our SGRS scheme with the Multi-W zero copy scheme as well as the generic scheme in MVAPICH. We do latency, bandwidth and CPU overhead tests using a vector datatype to demonstrate the effectiveness of our scheme. Then we show the potential benefits that can be observed for collective communication such as MPI_Alltoall that are built on top of point to point communication. Further we investigate the impact of layout caching for our design.

5.1 Experimental Testbed

A cluster of 8 SuperMicro SUPER X5DL8-GG nodes, each with dual Intel Xeon 3.0 GHz processors, 512 KB L2 cache, PCI-X 64-bit 133 MHz bus, and connected to Mellanox InfiniHost MT23108 DualPort 4x HCAs. The nodes are connected using the Mellanox InfiniScale 24 port switch MTS 2400. The kernel version used is Linux 2.4.22smp. The InfiniHost SDK version is 3.0.1 and HCA firmware version is 3.0.1. The Front Side Bus (FSB) runs at 533MHz. The physical memory is 1 GB of PC2100 DDR-SDRAM memory.

5.2 Vector Latency and Bandwidth Tests

In this benchmark, increasing number of columns in a two dimensional M*4096 integer array are transferred between two processes. These columns can be represented by a vector datatype. Figure 5 compares the ping-pong latency in the MPI implementation using the two zero-copy schemes. We set up two cases for the number of rows (M) in this array: one is 64 and one is 128. The number of columns varies from 4 to 2048, the corresponding message size varies from 2 KBytes to 512 KBytes. We also compare it with the latency of the contiguous transfer which serves as the lower bound. We observe that the SGRS scheme reduces the latency by up to 61% compared to that of the Multi-W scheme. Figure 6 shows the bandwidth results. The improvement factor over the Multi-W scheme varies from 1.12 to 4.0.

In both latency and bandwidth tests, it can also be observed that when the block size is smaller, the improvement of the SGRS scheme over the Multi-W scheme is higher. This is because the improved network utilization in the SGRS scheme is more significant when the block size is small. When the block size is large enough, RDMA operations on each block can achieve good network utilization as well. Both schemes perform comparably.Compared to the Generic scheme, the latency results of the SGRS scheme are better in cases when the block size is larger than 512 bytes. When the message size is small and the block size is small, the Generic scheme performs the best. This is because the memory copy cost is not substantial and the Generic scheme can achieve better network utilization. The bandwidth results of the SGRS scheme are always better than the Generic scheme.

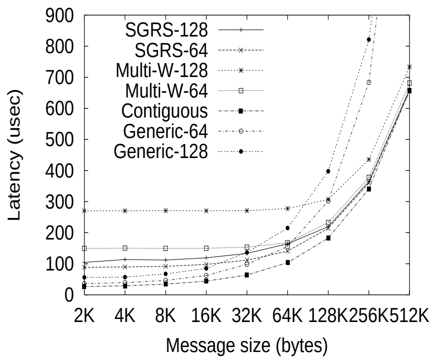

Fig. 5. MPI Level Vector Latency

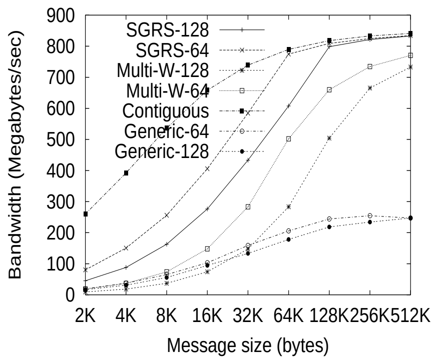

Fig. 6. MPI Level Vector Bandwidth

5.3 CPU overhead tests

In this section we measure the CPU overhead involved for the two schemes. Figures 7 and 8 compare the CPU overheads associated at the sender side and receiver side, respectively. The SGRS scheme has lower CPU involvement on the sender side as compared to Multi-W scheme. However on the receiver side the SGRS scheme has an additional overhead as compared to practically close to zero overhead incase of Multi-W scheme.

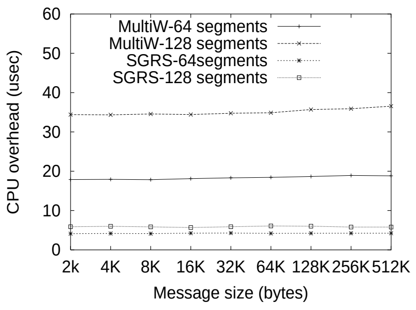

Fig. 7. Sender side CPU overhead

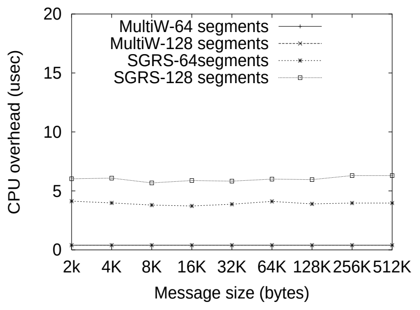

Fig. 8. Receiver side CPU overhead

5.4 Performance of MPI_Alltoall

Collective datatype communication can benefit from high performance point-to-point datatype communication provided in our implementation. We designed a test to evaluate MPI_Alltoall performance with derived datatypes. We use the same vector datatype we had used for our earlier evaluation.

Figure 9 shows the MPI_Alltoall latency performance of the various schemes on 8 nodes. We study the Alltoall latency over the message range 4K-512K. We ran these experiments for two different numbers of blocks: 64 and 128. We observe that the SGRS scheme outperforms the Multi-W scheme consistently. The gap widens as the number of blocks increases. This is because the startup costs in the Multi-W scheme increase with the increase of the number of blocks. In addition, given a message size, the network utilization decreases with the increase of the number of blocks in the Multi-W scheme.

5.5 Impact of Layout Caching

In both the Multi-W and SGRS schemes, the layout has to be exchanged between the sender and receiver before data communication. In this test, we studied the overhead

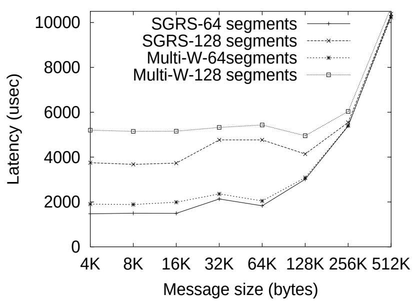

Fig. 9. MPI_Alltoall Latency

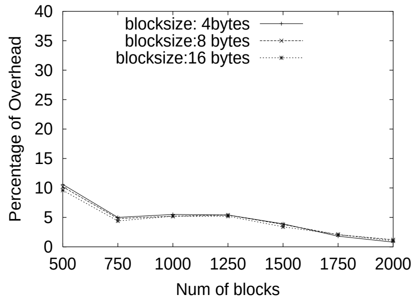

Fig. 10. Overhead of Transferring Layout Information

of transferring the layout information. We consider a synthetic benchmark where this effect might be prominent. In our benchmark, we need to transfer the two leading diagonals of a square matrix between two processes. These diagonal elements are actually small blocks rather than single elements. Hence, the layout information is complex and we need considerable layout size to describe it. As the size of the matrix increases, the number of non-contiguous blocks correspondingly increases as well as the layout description. Figure 10 shows the percentage of overhead that is incurred in transferring this layout information when there is no layout cache as compared with the case that has a layout cache. For smaller message sizes, we can see a benefit of 10 percent and this keeps diminishing as the message size increases. Another aspect here is that even though for small messages the layout size is comparable with message size, since the layout is transferred in a contiguous manner, it takes a lesser fraction of time to transfer this as compared to the non-contiguous message of comparable size. Since the cost associated in maintaining this cache is virtually zero, for message sizes in this range we can benefit from layout caching.

6 Related Work

Many researchers have been working on improving MPI datatype communication. Research in datatype processing system includes [5, 13]. Research in optimizing packing and unpacking procedures includes [2, 5]. The closest work to ours is the work [17, 14] to take advantage of network features to improve noncontiguous data communication. In [14], the use of InfiniBand features to transfer non-contiguous data is discussed in the context of ARMCI which is a one sided communication library. In [17], Wu *et al.* have systematically studied two main types of approach for MPI datatype communication (*Pack/Unpack-based approaches* and *Copy-Reduced approaches*) over InfiniBand. The Multi-W scheme has been proposed to achieve zero-copy datatype communication.

7 Conclusions and Future Work

In this paper we presented a new zero-copy scheme to efficiently implement datatype communication over InfiniBand. The proposed scheme, *SGRS*, leverages the Send Gather/Recv Scatter feature of InfiniBand to improve the datatype communication performance. The experimental results we achieved show that this scheme outperforms the existing Multi-W zero-copy scheme in all cases for both point to point as well as collective operations. Compared to the Generic scheme, for many cases, the SGRS reduces the latency by 62%, and increases the bandwidth by 400%. In the cases where the total datatype message size is small and the contiguous block sizes are relatively small, packing/unpacking based schemes [17] perform better. But beyond a particular "cutoff" point, the zero-copy scheme performs better. The SGRS scheme pushes this cutoff point to a relatively smaller value compared to the Multi-W scheme. As part of future work, we would like to compare this scheme with other schemes and evaluate this scheme at the application level.A combination of this scheme with other schemes can be incorporated to choose the best scheme for a given datatype message adaptively.

References

1. Mike Ashworth. A Report on Further Progress in the Development of Codes for the CS2. In *Deliverable D.4.1.b F. Carbonnell (Eds), GPMIMD2 ESPRIT Project, EU DGIII, Brussels*, 1996.
2. Surendra Byna, Xian-He Sun, William Gropp, and Rajeev Thakur. Improving the Performance of MPI Derived Datatypes by Optimizing Memory-Access Cost. In *Proceedings of the IEEE International Conference on Cluster Computing*, 2003.
3. Message Passing Interface Forum. MPI: A message-passing interface standard. *The International Journal of Supercomputer Applications and High Performance Computing*, 8(3–4), 1994.
4. B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, and H. Tufo. FLASH: An Adaptive Mesh Hydrodynamics Code for Modelling Astro physical Thermonuclear Flashes. *Astrophysical Journal Suppliment*, 131:273, 2000.
5. William Gropp, Ewing Lusk, and Deborah Swider. Improving the Performance of MPI Derived Datatypes. In *MPIDC*, 1999.
6. InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.0, October 24, 2000.
7. J. L. Träff, H. Ritzdorf and R. Hempel. The Implementation of MPI–2 One-sided Communication for the NEC SX. In *Proceedings of Supercomputing*, 2000.
8. Lawrence Berkeley National Laboratory. MVICH: MPI for Virtual Interface Architecture, August 2001.
9. 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.
10. Qingda Lu, Jiesheng Wu, Dhabaleswar K. Panda, and P. Sadayappan. Employing MPI Derived Datatypes to the NAS Benchmarks: A Case Study . Technical Report OSU-CISRC-02/04-TR10, Dept. of Computer and Information Science, The Ohio State University, Feb. 2004.
11. Mellanox Technologies. Mellanox InifniBand Technologies. http://www.mellanox.com.
12. Network-Based Computing Laboratory. MVAPICH: MPI for InfiniBand on VAPI Layer. `http://nowlab.cis.ohio-state.edu/projects/mpi-iba/index.html`.
13. Robert Ross, Neill Miller, and William Gropp. Implementing Fast and Reusable Datatype Processing. In *EuroPVM/MPI*, Oct. 2003.
14. V. Tipparaju, G. Santhanaraman, J. Nieplocha, and D. K. Panda. Host-Assisted Zero-Copy Remote Memory Access Communication on InfiniBand. In *IPDPS '04*, April 2004.
15. W. Gropp and E. Lusk and N. Doss and A. Skjellum. A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard.
16. Jiesheng Wu, Pete Wyckoff, and Dhabaleswar K. Panda. Supporting Efficient Noncontiguous Access in PVFS over InfiniB and. In *Proceedings of the IEEE International Conference on Cluster Computing*, 2003.
17. Jiesheng Wu, Pete Wyckoff, and Dhabaleswar K. Panda. High Performance Implementation of MPI Datatype Communication over InfiniBand. In *International Parallel and Distributed Processing Symposium (IPDPS '04)*, April 2004.