# Scaling Alltoall Collective on Multi-core Systems *

Rahul Kumar Amith Mamidala D. K. Panda

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

## Abstract

*MPI_Alltoall is one of the most communication intense collective operation used in many parallel applications. Recently, the supercomputing arena has witnessed phenomenal growth of commodity clusters built using InfiniBand and multi-core systems. In this context, it is important to optimize this operation for these emerging clusters to allow for good application scaling. However, optimizing MPI_Alltoall on these emerging systems is not a trivial task.*

*InfiniBand architecture allows for varying implementations of the network protocol stack. For example, the protocol can be totally on-loaded to a host processing core or it can be off-loaded onto the NIC or can use any combination of the two. Understanding the characteristics of these different implementations is critical in optimizing a communication intense operation such as MPI_Alltoall. In this paper, we systematically study these different architectures and propose new schemes for MPI_Alltoall tailored to these architectures. Specifically, we demonstrate that we cannot use one common scheme which performs optimally on each of these varying architectures. For example, on-loaded implementations can exploit multiple cores to achieve better network utilization, and in offload interfaces aggregation can be used to avoid congestion on multi-core systems. We employ shared memory aggregation techniques in these schemes and elucidate the impact of these schemes on multi-core systems. The proposed design achieves a reduction in MPI_Alltoall time by 55% for 512Byte messages and speeds up the CPMD application by 33%.*

*This research is supported in part by DOE grants DE-FC02-06ER25755 and DE-FC02-06ER25749, NSF Grants CNS-0403342 and CCF-0702675; grants from Qlogic, Intel, Sun MicroSystems, Cisco Systems, and Linux Networks; and equipment donations from Intel, AMD, Apple, IBM, Microway, PathScale, SilverStorm and Sun MicroSystems.

## 1 Introduction

The complete data exchange collective, MPI_Alltoall, is one of the most intensive communication patterns used in various applications including molecular dynamics applications like CPMD [1], NAMD [14], LU-factorization, FFT and matrix transpose. MPI_Alltoall is known to suffer from performance scaling problems. With the increase in the number of processing elements, owing to multi-core systems, it is highly desirable to optimize this data intensive communication primitive.

Several algorithms have been proposed to optimize this collective in the past. With the introduction of new architectural concepts such as multi-core processors, these algorithms need renewed attention. In multi-core systems, the processes within a node have a very low latency communication between them compared to inter-node latency. This gives scope for improved algorithms. Also, over the past few years there has been a surge in network interface based processing (offload) networks such as Infinihost III [2] from Mellanox. But with change in processor architectures to multi-core processor, host based NICs have re-emerged to take advantage of faster processing power of on-board cores. One example of this is Qlogic's [15] implementation of InfiniBand called InfiniPath. Mellanox has also introduced its latest ConnectX adapter [3] which uses host processing for small messages and network interface processing for medium and large messages.

In this paper, we study the characteristics of the above three NICs and demonstrate how algorithms for MPI_Alltoall is affected by these characteristics. Specifically, we propose different algorithms for multi-core systems connected with varying implementations of InfiniBand network interfaces. In particular, we aim to provide answers to the following questions:

- How do the current multi-core aware algorithms for other collectives suit for alltoall?
- What are the advantages and disadvantages of those

schemes?

- What are the features and characteristics of modern NICs?
- Are there better collective algorithms for modern NICs?

We have implemented our designs and integrated them into MVAPICH [12]. The proposed design has been evaluated on a 512 core cluster. The optimizations proposed in this paper reduces the latency of MPI_Alltoall by 55% for 512Byte messages and speeds-up the CPMD application benchmark by 33%. The rest of the paper is organized in the following way. In Section 2, we provide the background of our work. In Section 3, the motivation for our scheme has been explained. In Section 4, we discuss related work. In Section 5, we discuss detailed design issues and evaluate our designs in Section 6 . Conclusion and future work are presented in Section 7.

# 2 Background

In this section, we briefly describe the required background on current InfiniBand network interfaces, widely employed alltoall algorithms and CPMD application [1]. The CPMD application extensively uses alltoall collective and is used in the evaluation of our proposed scheme.

## 2.1 InfiniPath

InfiniPath [15] is a network interface for InfiniBand interconnect provided by Qlogic. It's main difference from other NICs is that it is host based and does not offload network processing to the NIC. The network interface does not have a send-side DMA so processor cycles are used to copy data to the NIC. The NIC is only responsible for streaming the data out to the network. The rationale is that the host processor is generally much faster than the NIC. Another reason is that data to be sent out is usually in the cache of the host so unnecessary bus transactions are avoided if the buffers are re-used. However, on receive side, it has a DMA engine that writes the incoming data directly to main memory.

## 2.2 InfiniHost III and ConnectX

InfiniHost III is the third generation of InfiniBand Host Channel Adapter (HCA) from Mellanox. It features a full hardware implementation of the InfiniBand architecture with Hardware Transport Engine that drastically reduces the host CPU overhead on communication. ConnectX is the fourth generation InfiniBand HCA from Mellanox. The ConnectX architecture is designed to improve the processing rate of incoming packets. Compared to the previous InfiniHost III architecture, it has more advanced packet processing capabilities. These enhancements to the ConnectX architecture are expected to improve its performance on multi-core nodes when multiple processes are communicating at the same time, generating many simultaneous network messages. For very small message sizes (less than around 512Bytes), PIO is used to send data to the network interface. This is different from the InfiniHost III architecture which uses DMA for all message sizes.

## 2.3 CPMD Application

The Car-Parrinello Molecular Dynamics (CPMD) is designed for ab-initio molecular dynamics. It is widely used for research in computational chemistry, materials science and biology. CPMD makes extensive use of three-dimensional FFT, which requires efficient all-to-all communication [8].

## 2.4 AlltoAll Algorithms

The most popular algorithms for alltoall currently used are: 1. Bruck's algorithm [4], 2. Irecv-Isend algorithm [19] and 3. Pairwise Exchange [19].

Because none of the above algorithms gives the best performance for all message sizes, we choose different algorithms according to the message size. Bruck's algorithm completes in minimum number of steps, log P (P is the number of processes). Hence, it is used for small messages where start-up latencies are a dominant part of the collective time. However, because it sends the same message over the network more than once, it is not suitable for medium or large messages. In the Irecv-Isend algorithm, each process sends the data directly to the destined process, hence it requires P-1 steps to complete. The amount of data going out of each node is equal to the total amount of data that each node must send. The amount of data going into each node is equal to the total amount of the data that each node must receive. Therefore, the algorithm is optimal in terms of amount of data sent on the network and should be suitable for medium and large messages. However, we found the algorithm is not suitable for large messages. At large message sizes, contention on the links comes into play. The algorithm uses a cyclic pattern of communication which is not congestion free on fat-tree networks [9]. The pairwise exchange algorithm gives better results for large messages. At each stage of the pair-wise exchange algorithm, the communication pattern is congestion free on fat-tree networks. Moreover, 'irecv-isend' algorithm makes loose coupling among the sending and receiving processes. It has been found that if processes are tightly coupled, the latencies are lower for large messages [16]. Pair-wise exchange uses send-recv, utilizing rendezvous protocol for large messages and hence are tightly coupled.

We have found that network characteristics play a role in tuning the alltoall collective for different network interfaces. For example, the 'irecv-isend' algorithm performs poorly on InfiniHost III and ConnectX. However, it performs well for medium-sized messages on the InfiniPath network interface. All of the above tuning are available in the open source MVAPICH [12] software.

## 3 Motivation

Communication time of MPI_Alltoall is dependent on two factors: start-up costs and network bandwidth. For small messages, MPI_Alltoall time is dominated by start-up costs. For large messages, network bandwidth determines the time of the operation. To illustrate the impact of start-up costs on latency of MPI_Alltoall, we conducted a simple test to measure the time of 'irecv-isend' alltoall algorithm on a fixed set of nodes. However, we increase the number of cores involved in the collective keeping the size of total data involved in the operation the same. The experiment was conducted on our InfiniPath cluster mentioned in section 6. Figure 1 shows the results. As shown in the figure, we observe that there is a significant increase in MPI_Alltoall time with the increase in the number of cores per node although the amount of data exchanged between the nodes is the same. This is primarily due to an increasing number of sends issued which increases start-up costs. Thus, reducing start-up costs is necessary to obtain good performance.

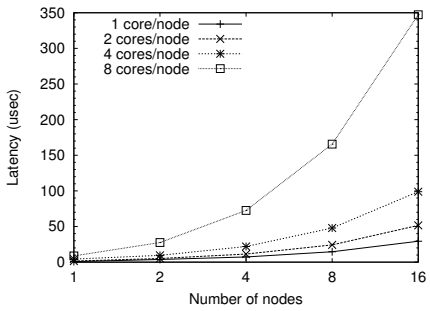



**Figure 1. Increase in Alltoall time with increase in cores/node**

On the other hand, different implementations of InfiniBand network interfaces exist. These different interfaces exhibit varying communication characteristics. We demonstrate this using a simple bi-directional bandwidth test between two nodes. The number of concurrent pairs involved in the test is increased from one to four. Figures 2 and 3 show the multi-pair bi-directional bandwidth performance on InfiniPath and ConnectX adapters, respectively.

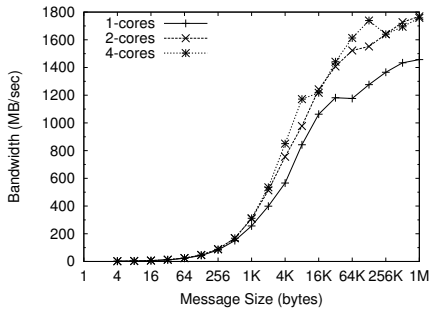



**Figure 2. InfiniPath SDR: Multi-pair Bidirectional Bandwidth.**

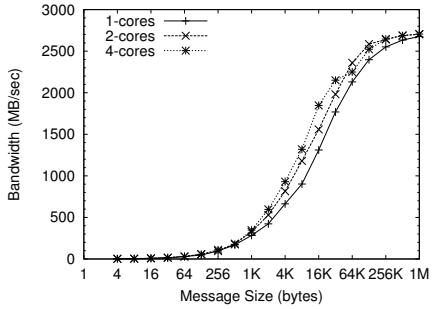



**Figure 3. ConnectX DDR: Multi-pair Bidirectional Bandwidth.**

We observe that using more than one core to send the data out of the node is advantageous as it provides better network utilization, as can be seen by the increase in bandwidth on using more cores. This is not the case for earlier generation InfiniHost III architecture, as can be seen in Fig.4. For MPI_Alltoall operations, this observed behavior is significant because, like the bi-directional test, multiple cores are involved in data exchange across the nodes.

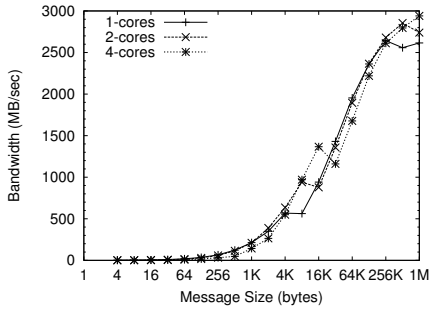



**Figure 4. InfiniHost III DDR: Multi-pair Bidirectional Bandwidth.**

Also on multi-core systems, aggregation and distribution schemes can be used to decrease the network steps. This is discussed further in the design section. In the aggregation scheme, all cores of a node copy data from

the send buffers to a shared buffer, where data can be sent out on the network. In the distribution scheme, after all cores of a node receive data from the network in a shared memory location, the data is copied into the respective receive buffers. We performed tests for both schemes in which we measure the amount of time it takes to perform the data copy operations. The results of the experiment are plotted in Fig.5 for InfiniPath interface. Although both schemes perform the same number of memory copies with the same amount of data sizes, the time taken to complete the copies are different. This characteristic is seen only on host-based network interfaces where data is sent via PIO and not when the DMA engine is used to copy the data to NIC. We believe this performance gap is due to a memory cache transaction in the distribution scheme; when receiving data from network, the receive side DMA directly writes it to the main memory. Whereas in the aggregation scheme, data copy is from cache to cache because data to be sent is generally in the cache of the host. Therefore, we observe that aggregation is better compared to distribution when PIO is used to copy the data instead of DMA engine.

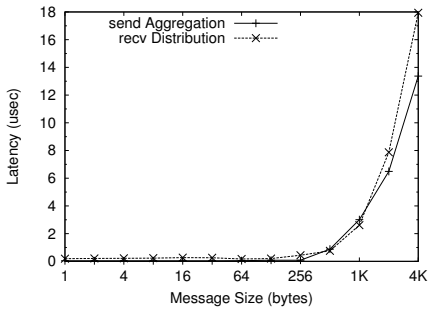



**Figure 5. Receive-side distribution more costly than send-side aggregation.**

Thus, as described above, different network interfaces exhibit varying communication characteristics. The alltoall schemes need to take into account these factors to obtain good performance.

## 4 Related Work

Several optimizations have been proposed in the past to take advantage of shared memory to design collectives. Husbands et al. [7] first proposed hierarchical tree based MPI_Bcast algorithm to minimize the use of network on the Sun SMP system. Sistare et al. [17] propose a hierarchical scheme but do not use a tree-based point to point communication within the SMP node. They develop shared memory based schemes to optimize broadcast, reduce, allreduce and barrier within the SMP node. Tipparaju et al. [20] also propose hierarchical tree based collective operations using shared and remote memory access protocols. In our earlier work [11], we proposed a hierarchical multicast based design for broadcast. Most of the work to optimize collectives for shared memory based systems have proposed hierarchical leader based schemes. In this paper we propose a non-leader based scheme to optimize alltoall collective for multi-core systems.

## 5 The Proposed Design

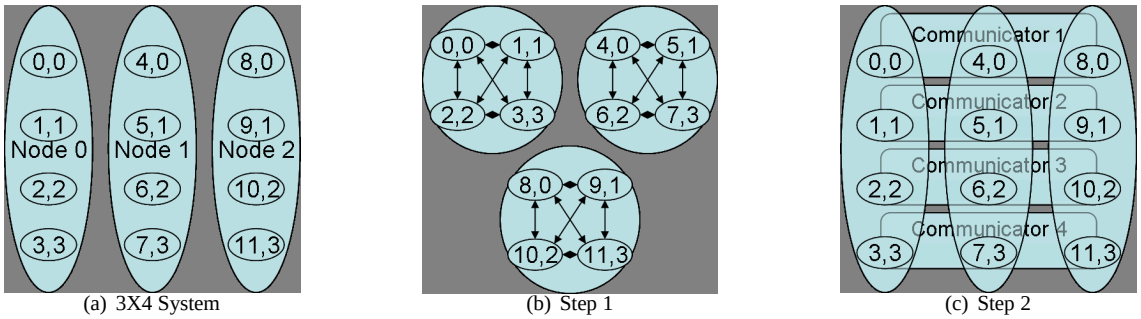
In the leader based scheme, at the sender, all data of a node is aggregated to the leader of the node followed by inter-node communication and then distribution to all the processes of the node at the receiver. The leader based scheme proposed for SMP based clusters cannot be naively used for all multi-core systems. It has the following disadvantages which are addressed by the proposed design:

1. Significant shared memory overhead (assuming all the intra-node communication is done using shared memory communication). This is because the inter-node alltoall can begin only after all processes of the node have written data to the shared memory location.

2. Utilizes a single core to perform the inter-node communication and therefore does not take advantage of the increase in bi-directional bandwidth available with the increase in number of cores used to send the data.

3. Does not distinguish between shared memory aggregation of data at the sender and shared memory distribution at the receiver.

One can also design a leader-based scheme with two leaders per node. This scheme would achieve better bandwidth as it utilizes more cores to send the data to other nodes. However, this also increases the number of network sends by two times and hence increases start-up costs. Instead, a scheme in which more cores of the node participate in inter-node communication without an increase in number of sends issued by each core will benefit significantly. This can be achieved if each core of a node communicates with one and only one core of all other nodes. This does not increase the number of network sends issued by each process and network start-up costs are almost the same with better network utilization. In our implementation of the above scheme, we opted to use all cores of a node to participate in inter-node communication. However, a subset of the cores can also be used. We explain the design keeping in mind that all of the cores are being used. However, it can be extended to address the situation wherein only a subset of the cores are utilized.




(a) 3X4 System (b) Step 1 (c) Step 2

**Figure 6. Communication steps of the proposed design**

Since a core/process communicates with a single core of the other nodes, intra-node communication must be used to send the respective data to the other cores of the node. There are two ways in which this can be performed: 1. Before sending the data to other nodes, perform an intra-node communication. In this, a core receives all data that has to be received by cores with which it will communicate. We call this send-side aggregation. OR 2. Perform the inter-node communication first. In the inter-node communication, a core sends all of its data destined for a node to the core with which it communicates. It then performs an intra-node communication in which a core sends the data to the respective cores for which it was destined. We call this receive-side distribution.

We chose the first option for our implementation because we observed that send-side aggregation is less costly for network interfaces which use PIO to perform the data transfer to NIC. This is shown in Fig. 5.

The proposed algorithm for alltoall completes in two steps:

1. Step 1: Intra-node exchange - This step takes place simultaneously within all nodes. Each core/process sends all the data that has to go to shared memory rank x (rank of the process in its node) of all nodes to process with shared memory rank x on its node.

2. Step 2: Inter-node exchange - Each core/process performs an inter-node alltoall communication with processes having the same shared memory rank. The message size of this alltoall is more than that of the intended alltoall communication.

Figure 6 shows the communication that takes place in each of the above steps for a system of size 3X4 i.e. three nodes and each node has four cores as shown in Fig.6(a). Each small ellipse represents a core and the surrounding bigger ellipse represents a node. The numbers 'a' and 'b' on the core represent 'a' as MPI_Rank and 'b' as the shared memory rank. Figure 6(b) shows the communication of step 1. In this step, only intra-node communication takes place. After step 1, all processes having the same shared memory rank are part of one communicator as shown in figure 6(c). Each of the core/process participates in alltoall communication within the new communicator.

The above scheme has the following advantages:

1. Lower shared memory overhead. Each process waits for other processes to write only a subset of their data not their whole data.

2. Uses more than one core to send out the data, allowing for better bandwidth.

3. Takes advantage of low aggregation at the sender and eliminates the need for more costly distribution at the receiver. This is only applicable for some NICs.

For systems which do not use PIO, send-side aggregation and receive-side distribution have similar performance.

## 6 Experimental Results

The following two test-beds were used to conduct the experiments:

First is a 512-core InfiniBand Linux cluster. Each of the 64 nodes have dual 2.33 GHz Intel Xeon "Clovertown" quad-core processors for a total of 8 cores per node. Each node has two network interface cards. First is a host-based SDR network interface QLE7140 by Qlogic and the second is offload DDR network interface card MT25208 dual-port Memfree HCA by Mellanox. InfiniBand software support is provided through InfiniPath software stack 2.1 on Qlogic HCA and OpenFabrics/Gen2 stack [13], OFED 1.2 release for Mellanox HCA. The Mellanox HCA is built using the Infinihost III architecture.

Second is a 4 node dual 2.33 GHz Intel Xeon "Clovertown" quad-core processors for a total of 8 cores per node. Each node is connected with Mellanox ConnectX cards which operate at DDR speed (20Gbps). The ConnectX card (MT25408) has firmware version

2.0.139 and operates with new Open-Fabrics drivers which are based on OFED 1.2 distribution.

We have used MVAPICH-PSM and MVAPICH-Gen2 [12] to test our collective schemes on the two devices. MVAPICH is a popular open-source MPI implementation over InfiniBand. It is based on MPICH [6] and MVICH [10] and is used by over 610 organizations worldwide.

## 6.1 Alltoall Performance

In this section, we evaluate and compare the performance of the proposed scheme using OSU Alltoall Benchmark. The benchmark calls MPI_Alltoall back-to-back and reports an average over a large number of iterations (typically 1000).

### 6.1.1 Performance on InfiniPath

Figure 7 shows the MPI_Alltoall collective performance on 64X8 configuration where AXB implies 'A' nodes and 'B' cores per node. The legend 'orig' refers to the current algorithms employed in MVAPICH tuned for the testbed for appropriate message sizes.

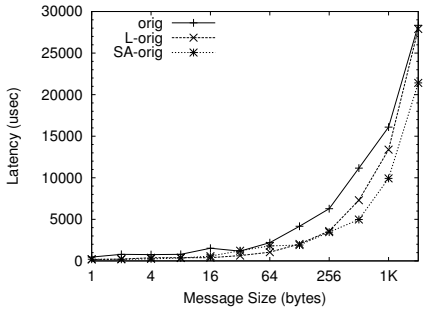



**Figure 7. InfiniPath: Alltoall time on 64X8 system**

MVAPICH-PSM currently uses Bruck's algorithm for up to 256 Bytes, direct 'irecv-isend' from 256Bytes to 32KB and pairwise exchange for messages larger than 32KB for the Qlogic HCA. We have found that the direct 'irecv-isend' algorithm performs poorly on Mellanox HCA; therefore MVAPICH-GEN2 uses Bruck's algorithm for up to 8KB and pairwise exchange for messages larger than 8KB. The 'L-orig' scheme refers to leader based scheme and uses the original tuned alltoall to perform the inter-node alltoall communication among the leaders. The 'SA-orig' scheme refers to the new proposed scheme explained in section 5. It uses the tuned alltoall explained above to perform step 2 of the proposed scheme.

The leader based scheme performs well for very small messages. However, due to high shared memory overhead, the benefits fade with increasing message size. The proposed scheme outperforms the current algorithm and leader-based algorithm up to 2KB message size. This is primarily due to better utilization of network bandwidth by using multiple cores. As the system size increases, higher performance gains are obtained and up to a greater message size.

Figure 8 compares the MPI_Alltoall time for send-aggregation and receive-distribution scheme. The 'orig-RD' scheme refers to inter-node alltoall followed by distribution at the receive-side. InfiniPath HCA uses PIO to copy data from the host to the NIC. Therefore, we see the performance difference between the two schemes. For a detailed explanation, refer to section 3. At each point on the graph beyond 256Bytes, the send aggregation scheme is better by at least 10 percent. The benefits are seen only after 256 Bytes. For messages less than 256 Bytes, Bruck's algorithm is used which touches (copies) the data after receiving it from the network [19].

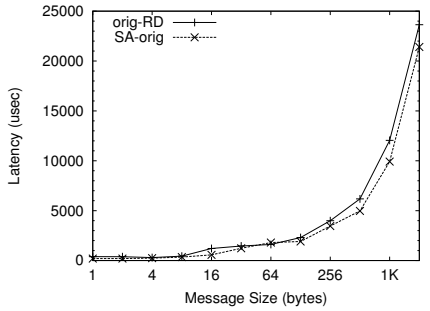



**Figure 8. InfiniPath: Send Aggregation vs Recv Distribution alltoall time on 64X8 system**

Figure 9 shows the MPI_Alltoall time for 512Byte message on varying system sizes. The results show that the performance gains in alltoall time increase with increasing system sizes.

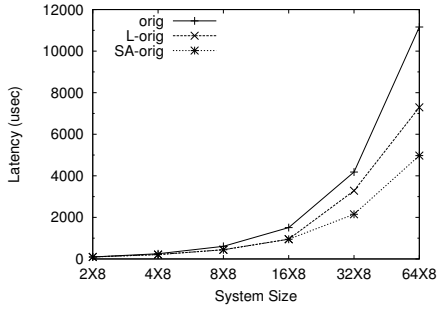



**Figure 9. InfiniPath: Alltoall time of 512Byte message**

#### 6.1.2 Performance on Infinihost III

On Infinihost NIC, simultaneously using multiple cores deteriorates the performance of communication latency [18]; therefore, multi-pair bi-directional bandwidth shows deterioration in performance with increasing number of cores as can be seen in Fig.4. Therefore, leader-based scheme performs best here because it is able to eliminate the effects of congestion. This can be seen from the results in Fig. 10.

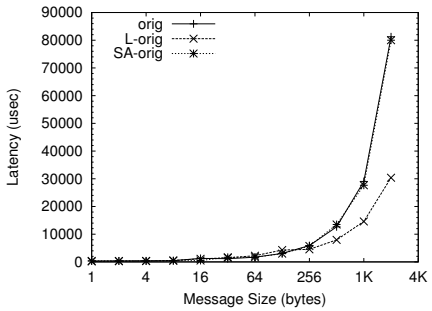



**Figure 10. Infinihost III: Performance of different schemes on 64X8 system**

On offload NIC, send-side aggregation scheme and receive-side distribution scheme show similar performance as seen in Fig. 11.

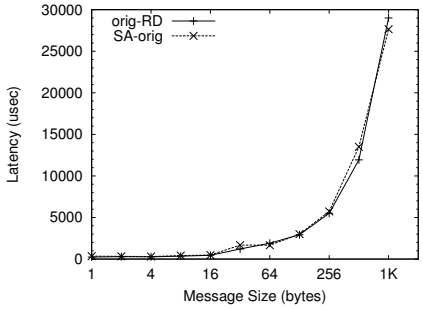



**Figure 11. Infinihost III: Send Aggregation vs Recv Distribution alltoall time on 64X8 system**

#### 6.1.3 Performance on ConnectX

From Fig.3 we see that on ConnectX architecture, multi-pair bi-directional bandwidth increases with more cores. Therefore, single leader-based scheme does not perform as well as send-aggregation and recv-distribution schemes. The alltoall time for different schemes can be seen in Fig.12.

### 6.2 Application Performance

The CPMD application was used to evaluate the performance impact of the proposed scheme on applications. The InfiniPath network interface testbed was used

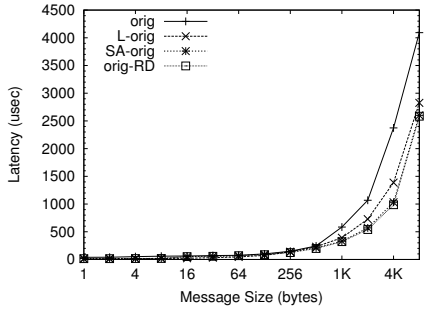



**Figure 12. ConnectX: Performance of different schemes on 4X8 system**

for the evaluation. Figure 13(a) shows the performance improvement over the current algorithms for different input files on 16X8 system. Figure 13(b) shows the performance improvement for si63 atoms with 120ryd cutoff for different system sizes. As we saw earlier, the performance improvement for MPI_Alltoall increases with increasing system sizes, this is also reflected in the application performance improvement. Fig.13(b) shows that the 'L-orig' scheme begins to perform well at 64X8 system size. At 64X8 system size, the message size of alltoall collective decreases as the problem size remains the same. Therefore, leader-based collective performs comparable to the proposed scheme.

## 7 Conclusion and Future Work

The results of this paper show that various network interfaces implemented for the same interconnect, exhibit different network characteristics. A single collective algorithm does not perform optimally for all network interfaces due to differing network characteristics. The paper proposes an optimized alltoall collective algorithm for multi-core systems connected using modern InfiniBand network interfaces. However, we believe that the work can be applied to onload implementation of other networks as well, like the ethernet-based JNIC architecture [5]. We plan to evaluate our designs on such systems in the future, as well as , extend the proposed framework to all other collectives.

## References

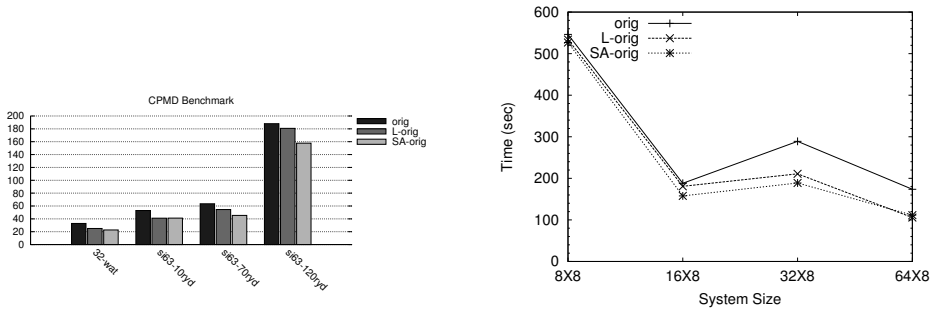

[1] http://www.cpmd.org/.

[2] http://www.mellanox.com/products/infinihost_iii_ex_cards.php.

[3] http://www.mellanox.com/products/connectx_architecture.php.

[4] J. Bruck, C.-T. Ho, S. Kipnis, E. Upfal, and D. Weathersby. Efficient Algorithms for All-to-All Communications in Multiport Message-Passing Systems. *IEEE Transactions in Parallel and Distributed Systems*, 8(11):1143–1156, November 1997.

[5] Mike Schlansker et. al. High-performance Ethernet-based Communications for future Multi-core Processors. In *SC '07: Pro-*




(a) Different Input Files

(b) Varying System Sizes

**Figure 13. CPMD Application Benchmark Performance on InfiniPath**

*ceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 49–59, 2007.

[6] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard. Technical report, Argonne National Laboratory and Mississippi State University.

[7] P. Husbands and J.C. Hoe. MPI-StarT: Delivering Network Performance to Numerical Applications. *Supercomputing, 1998. SC98. IEEE/ACM Conference on*, pages 17–17, 07-13 Nov. 1998.

[8] J. Hutter and A. Curioni. Dual-level Parallelism for Ab Initio Molecular Dynamics: Reaching Teraflop Performance with the CPMD Code. In *Parallel Computing*, pages 1–17, 2005.

[9] Sameer Kumar and Laxmikant V. Kale. Scaling All-to-All Multicast on Fat-tree Networks. *icpads*, 00:205, 2004.

[10] Lawrence Berkeley National Laboratory. MVICH: MPI for Virtual Interface Architecture. http://www.nersc.gov/research/FTG/mvich/ index.html, August 2001.

[11] A.R. Mamidala, Lei Chai, Hyun-Wook Jin, and D.K. Panda. Efficient SMP-aware MPI-level Broadcast over InfiniBand's Hardware Multicast. *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 8 pp.–, 25-29 April 2006.

[12] Network-Based Computing Laboratory. MVAPICH: MPI over InfiniBand and iWARP. http://mvapich.cse.ohio-state.edu.

[13] OpenFabrics Alliance. OpenFabrics. http://www.openfabrics.org/.

[14] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kale. NAMD: Biomolecular Simulation on Thousands of Processors. In *Supercomputing*, 2002.

[15] Qlogic. InfiniPath. http://www.pathscale.com/infinipath.php.

[16] D. Roweth and A. Moody. Performance of All-to-All on QsNetII, quadrics white paper, available at http://www.quadrics.com/. 2005.

[17] S. Sistare, R.v. Vaart, and E. Loh. Optimization of MPI Collectives on Clusters of Large-Scale SMPs. *Supercomputing, ACM/IEEE 1999 Conference*, pages 23–23, 13-18 Nov. 1999.

[18] S. Sur, M. Koop, L. Chai, and D. K. Panda. Performance Analysis and Evaluation of Mellanox ConnectX InfiniBand Architecture with Multi-Core Platforms. In *15th IEEE Int'l Symposium on Hot Interconnects (HotI15)*, August 2007.

[19] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of Collective communication operations in MPICH. *Int'l Journal of High Performance Computing Applications*, 19(1):49–66, Spring 2005.

[20] V. Tipparaju, J. Nieplocha, and D.K. Panda. Fast Collective Operations Using Shared and Remote Memory Access Protocols on Clusters. *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pages 10 pp.–, 22-26 April 2003.