MULTI-CONNECTION AND MULTI-CORE AWARE ALL-GATHER ON INFINIBAND CLUSTERS

Ying Qian Mohammad J. Rashti Ahmad Afsahi
Department of Electrical and Computer Engineering, Queen's University
Kingston, ON, CANADA K7L 3N6
{ying.qian, mohammad.rashti}@ece.queensu.ca ahmad.afsahi@queensu.ca

ABSTRACT
MPI_Allgather is a collective communication operation that is intensively used in many scientific applications. Due to high data exchange volume in MPI_Allgather, efficient and scalable implementation of this operation is critical to the performance of scientific applications running on emerging multi-core clusters.
Mellanox ConnectX is a modern InfiniBand host channel adapter that is able to serve data communication over multiple simultaneously active connections in a scalable manner. In this work, we utilize this feature of the ConnectX architecture and propose novel multi-connection aware RDMA-based MPI_Allgather algorithms for different message sizes over multi-core clusters. Our algorithms also use shared-memory for intra-node communications. The performance results indicate that the proposed four-group multi-connection aware gather-based algorithm is the best algorithm up to 32B messages and gains up to 3.1 times improvement for 4B messages over the native MVAPICH implementation. The proposed single-group multi-connection aware algorithm is the algorithm of choice for 64B to 2KB messages, and outperforms the MVAPICH by a factor of 2.3 for 64B messages. The best algorithm for 4-64KB messages is the two-group multi-connection aware algorithm that achieves an improvement of up to 1.8 for 4KB messages over MVAPICH.

KEY WORDS
Collective Communications, MPI, All-gather, InfiniBand, Multi-core Clusters

1. INTRODUCTION

Multi-core SMP clusters have become the predominant computing platform for high-performance computing and emerging commercial and networking applications. SMP nodes are traditionally equipped with multiple single-core processors. With the emergence of multi-core processors, each core will run at least one process with multiple intra-node and inter-node connections to several other processors. This will put immense pressure on the interconnect and its communication system software.
InfiniBand [1] has been introduced to support the ever-increasing demand for efficient communication, scalability, and higher performance in clusters. It supports *Remote Direct Memory Access* (RDMA) operations that allow a process to directly access the exposed memory areas of a remote process.
Most scientific applications are developed on top of the *Message Passing Interface* (MPI) [2]. MPI provides both point-to-point and collective communication operations. Most MPI implementations over RDMA-enabled networks such as InfiniBand have now integrated MPI point-to-point communications with RDMA operations through user-level libraries to effectively bypass the operating system and lower the CPU utilization. In addition, some MPI implementations such as MVAPICH [3] currently use RDMA for some collective operations.
In this paper, we are interested in the design and efficient implementation of the all-gather collective in MPI. *MPI_Allgather* is an intensive operation, typically used in linear algebra operations, matrix multiplication, matrix transpose, as well as multi-dimensional FFT. Therefore, it is very important to optimize its performance on emerging multi-core clusters.
ConnectX [4] is the latest generation of InfiniBand host channel adapters (HCA) from Mellanox Technologies. Compared to the previous generation of InfiniBand HCAs, ConnectX provides much better performance and salability for simultaneous communication over multiple connections [5]. In this work, we take on the challenge in designing efficient all-gather algorithms by utilizing the multi-connection scalability feature of ConnectX for inter-node communications using RDMA Write, shared memory operations for intra-node communications in multi-core SMP nodes, as well as multiple cores for better system and network utilization. Specifically, we propose and evaluate three multi-connection and multi-core aware all-gather algorithms. Our performance results indicate that the proposed four-group multi-connection gather-based algorithm gains an improvement of up to 3.1 for 4B messages over the native RDMA-based MVAPICH implementation. Moreover, the proposed single-group multi-connection algorithm performs better than the MVAPICH by a factor of 2.3 for 64B messages. The two-group multi-connection algorithm achieves an improvement up to 1.8 for 4KB messages.
The rest of the paper is organized as follows. Section 2 provides an overview of the ConnectX architecture and the implementation of all-gather in MVAPICH. Related work is discussed in Section 3. Section 4 explains the

motivation behind this work by showing that collective operations such as all-gather could benefit from NICs having scalable multi-connection performance over multi-core clusters. Section 5 presents our proposed all-gather algorithms with an analysis of their resource utilization. Section 6 describes our experimental platform. Section 7 presents the performance of the proposed algorithms on a four-node two-way dual-core SMP cluster. Section 8 concludes the paper and discusses plans for future work.

2. BACKGROUND

2.1 ConnectX HCA

ConnectX [4] is the latest generation of InfiniBand HCAs from Mellanox Technologies. It is a two-port HCA that can operate as 4X InfiniBand or 10-Gigabit Ethernet. In this work, we are only concerned with the InfiniBand mode of the ConnectX.

The ConnectX architecture includes a stateless offload engine for NIC-based protocol processing. Compared to the previous generation of InfiniBand cards from Mellanox (InfiniHost III), ConnectX improves the processing rate of incoming packets by having hardware schedule the packet processing directly. This technique allows the ConnectX to have a better performance for processing simultaneous network transactions.

In addition, ConnectX supports a number of enhanced InfiniBand features [4] including hardware-based reliable multicast, enhanced atomic operations, fine-grain end-to-end QoS, and extended reliable connection. Such features will enhance the performance and scalability of the communication subsystem. However, to our knowledge, not all these features have been enabled by the ConnectX drivers and firmware.

2.2 All-gather in MVAPICH

MVAPICH implements the recursive-doubling algorithm for MPI_Allgather for power of two number of processes directly using RDMA operations [3]. No shared memory operation is used in this approach. An MPI send-recv approach is used for any other number of processes.

Based on the message size, the RDMA-based approach uses two different schemes: (1) copy-based approach for small messages into a pre-registered buffer to avoid buffer registration cost, and (2) zero-copy method for large messages, where the cost of data copy is prohibitive [6].

3. RELATED WORK

Thakur and his colleagues discussed recent collective algorithms used in MPICH [7]. They have shown some algorithms perform better than the others depending on the message size and the number of processes involved. Sur and others proposed efficient RDMA-based all-to-all broadcast and personalized exchange algorithms for InfiniBand-based clusters [6, 8].

Some recent work has been devoted to improve the performance of intra-node communications on SMPs [9, 10]. Buntinas and others [9] have used shared buffers, message queues, Ptrace system call, kernel copy, and NIC loopback mechanisms to improve large data transfers in SMP systems. In [10], Chai and others improved the intra-node communication by using the system cache efficiently, and requiring no locking mechanisms.

On collectives for SMP clusters and Large SMP nodes, Sistare and his colleagues presented new algorithms taking advantage of high backplane bandwidth of shared-memory systems [11]. Roweth and his associates studied how collectives have been devised and implemented on $QsNet^{II}$ [12]. In [13], Tipparaju and others overlapped the shared memory and remote memory access communications in devising collectives for IBM SP. Mamidala et al. [14] designed all-gather over InfiniBand using shared memory for intra-node and single-port recursive doubling algorithm for inter-node communication via RDMA. Qian and Afsahi [15] proposed efficient RDMA-based and shared memory aware all-gather at the Elan-level over multi-rail [16] $QsNet^{II}$ clusters.

In [17], Tipparaju and Nieplocha used the concurrency available in modern networks to optimize MPI_Allgather on InfiniBand and $QsNet^{II}$. This paper, similar to [15], uses multiple outstanding RDMA operations, however, they do not study the network systematically as we have done in this paper, nor do they use shared memory communication and multi-core systems.

On multi-connection capability of modern interconnectss, Rashti and Afsahi [18] established a number of connections at the verbs level between two processes running on two nodes (each node having an iWARP or InfiniBand NIC), and then point-to-point communications were performed over those connections. It was observed that the normalized multiple-connection latency of small messages is decreased and throughput is increased up to a certain number of connections.

In another work [5], Sur and others measured the multi-pair RDMA-Write latency and aggregate bandwidth at the InfiniBand verbs level over ConnectX IB HCAs between multi-core platforms. They established a connection between each pair of processes on different nodes. With increasing number of pairs, the results showed that the ConnectX architecture is able to provide almost the same latency for small messages for up to 8 process pairs.

Kumar and his associates recently proposed an algorithm for *MPI_Alltoall* collective on multi-core systems [19]. Their algorithm is perhaps the closest in principle to our Multi-group Multi-connection Aware algorithm presented in Section 5.3.2, however they first do an intra-node exchange among the local processes and then exchange messages across the nodes. The other main difference is that we use RDMA for inter-node communications. Moreover, we study the impact of different number of groups in the inter-node communications on the performance of MPI-Allgather.

4. MOTIVATION

In multi-core clusters, each core will run at least one process with possible connections to other processes. It is very important for the NIC hardware and its communication software to provide scalable performance with the increasing number of connections. Recent research [18, 5] has shown that modern high-performance networks, such as ConnectX, can provide multi-connection scalability for point-to-point communications. In this work, we are interested in utilizing the multi-connection capability of the ConnectX InfiniBand networks in the design and implementation of efficient all-gather. We will complement our design with shared-memory operations for intra-node communications, and multiple cores for better network and system utilization.

To understand whether the multi-connection ability of ConnectX can help all-gather (and collectives in general), we have devised a micro-benchmark in which MPI processes on a set of multi-core nodes are mapped into a number of groups where each group concurrently performs an independent MPI_Allgather. The experiment was conducted on a 4-node, 16-process ConnectX multi-core cluster described in Section 6. Processes of each group are mapped on four different nodes. Therefore, all communications of an all-gather operation within a group are across the network. We have designed the individual all-gather algorithm in such a way that each process transfers its own data to the other three members of the group using three back-to-back RDMA operations in a ring-based fashion, leading to three simultaneous active connections per process. With four groups, we will have up to 12 bi-directional active connections per card.

We consider a single-group single-connection ring-based all-gather as the baseline operation. We then compare our single-group and multiple-group multi-connection all-gather with the baseline all-gather, as shown in Figure 1. The aggregate bandwidth plot in Figure 1 shows the total volume of data per second that passes through the ConnectX HCA. To have a better understanding of how multi-connection could enhance the performance, the bandwidth ratio plot in Figure 1 shows the bandwidth ratio of the single- and multiple-group multi-connection all-gather over the baseline collective operation.

It is evident that the multi-connection all-gather operations achieve higher aggregate bandwidth, and can saturate the network card more than a single-connection all-gather up to 64KB. The single-group multi-connection case improves the throughput up to 1.8 times the baseline, while the two, three and four groups can achieve up to 2.6, 2.7 and 2.7 times, respectively. The results for the 3-group and 4-group all-gather are fairly close to each other, implying that the network is almost saturated by nine simultaneous connections. For all multi-connection tests, the ratio drops below one for 64KB messages and above, which shows performance degradation for large messages when multiple connections are simultaneously active.

The results clearly show that the network is capable of providing scalable performance when multiple connections are concurrently active, at least for small to medium size messages and up to a certain number of connections. The message is that there is now an opportunity to improve the performance of collectives by devising efficient algorithms that use multiple connections concurrently on multi-core systems.

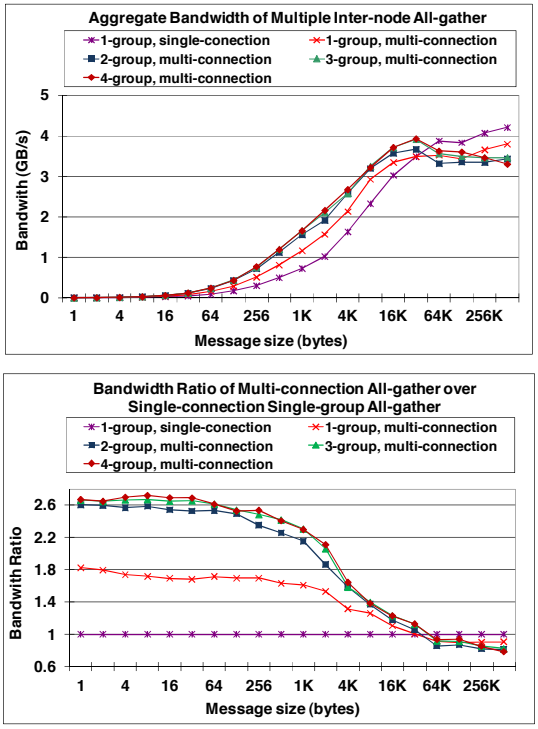

Figure 1. Aggregate bandwidth of multiple independent all-gather operations.

5. THE PROPOSED ALL-GATHER ALGORITHMS

This section presents a number of algorithms for the MPI_Allgather. We first propose the *Single-group Multi-connection Aware* all-gather algorithm, which is a multi-connection extension of the SMP-aware algorithm proposed in [15], targeted at small to medium messages.

We then propose and study two different classes of algorithms to enhance the all-gather performance for different message sizes. We propose the *Multi-group Gather-based Multi-connection Aware* all-gather algorithm to achieve efficient performance for very small messages. This algorithm takes advantage of multiple available cores on the node to distribute the CPU processing load. Finally, to further utilize the multi-connection capability of our InfiniBand network, we study the *Multi-group Multi-connection Aware* all-gather algorithm for medium to large message sizes. This algorithm has a lighter shared memory volume, but uses more number of connections per node.

5.1 Single-group Multi-connection Aware Algorithm

Single-connection SMP-aware collective communication algorithms can greatly improve the performance for small to medium message sizes [15, 14]. With the availability

of scalable multi-connection performance in modern networks, there is now an opportunity to improve the performance further. For this, we propose the inter-node communication phase of the SMP-aware all-gather algorithm [15] to use multiple simultaneous connections. The algorithm has three phases as follows:

Phase 1: Per-node shared memory gather
Phase 2: Inter-node multi-connection aware all-gather among the Master processes
Phase 3: Per-node shared memory broadcast

Each node has a Master process. In Phase 1, each Master process gathers the data from the processes on its node using a shared memory gather operation. In Phase 2, the Master processes participate in an inter-node all-gather operation. Each Master process sends the gathered data in Phase 1, in a multi-connection aware fashion, concurrently over all connections to the other Master processes using RDMA Write operations. In Phase 3, each Master process performs a shared memory broadcast to all local processes. In essence, each Master process copies out the received data in Phase 2 to a shared buffer, where each local process copies the final data then to its own destination buffer.

5.2 Multi-group Gather-based Multi-connection Aware Algorithm

In Phase 2 of the Single-group Multi-connection Aware algorithm, each Master process is responsible for communication with the Master processes on the other nodes. However, the local processes on each node are idle, waiting for the combined data to be shared by their respective Master process. For small messages, the ConnectX has the ability to carry the message data on the work request using programmed input/output (PIO) instead of DMA [5]. This reduces the communication latency however the CPU/core is more involved with the communication process, especially when we have multiple simultaneously active connections. Therefore, to take advantage of multi-core systems and to evenly distribute the communication workload on the available cores, we design a multi-group all-gather algorithm in which the outbound connections of each node are now distributed among the cores.
As shown in Figure 2, in this algorithm we group the processes across the nodes whose intra-node rank is equal. Each group has a Master process that gathers the data from the processes of the group. The algorithm is performed in three phases:

Phase 1: Per-node shared memory all-gather
Phase 2: Per-group inter-node multi-connection aware gather
Phase 3: Per-node shared memory broadcast

In Phase 1, an intra-node shared memory all-gather is performed among processes on the same node. In Phase 2, each group Master process gathers the data from the processes of its group. This means that at the end of Phase 2, each Master process will have the entire data from all processes. In Phase 3, each Master process will then broadcast the data to all processes on its own node. The only limitation with this algorithm is that each node should have at least a group Master process. If the number of nodes is more than the number of groups, some nodes will remain without a Master. To cover this, some groups may need to have more than one Master processes with the same duties.

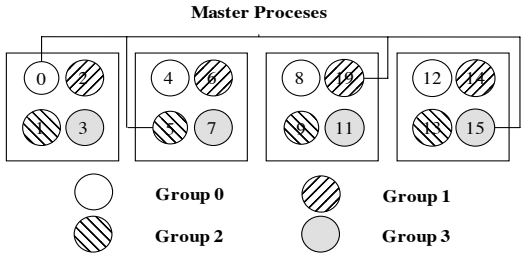

Figure 2. Group structure for gather-based all-gather on a 4-node, 16-core cluster.

5.3 Multi-group Multi-connection Aware Algorithm

Up to this point, we have utilized both multi-connection and multi-core features of a modern InfiniBand cluster in an effort to improve the performance of the all-gather collective operation. However, with our 4-node cluster, both the proposed algorithms in Section 5.1 and Section 5.2 use only a maximum of three simultaneously active connections per card. To examine the multi-connection ability of the NIC more aggressively, we propose the Multi-group Multi-connection Aware all-gather algorithm, in which more number of concurrent active connections is used during the all-gather operation. Basically, with increasing number of cores per node, more groups can be formed in this algorithm, which will even put more pressure on the NIC. To understand the impact of the number of groups on performance, we first study an algorithm with only two independent groups.

5.3.1 Two-group all-gather: As discussed earlier in Section 4, two independent multi-connection all-gather operations achieve a much higher bandwidth than a single all-gather case, at least for small to medium size messages. Therefore, we expect this algorithm to perform well for a range of message sizes. Figure 3 shows the group structure for a two-group multi-connection all-gather on our cluster. Each group has a Master process on each node, and includes half of each node's processes. The algorithm is performed in three phases as follows:

Phase 1: Per-node/per-group shared memory gather by each group Master process
Phase 2: Per-group inter-node multi-connection aware all-gather among group Master processes
Phase 3: Per-node shared memory broadcast from each group Master process

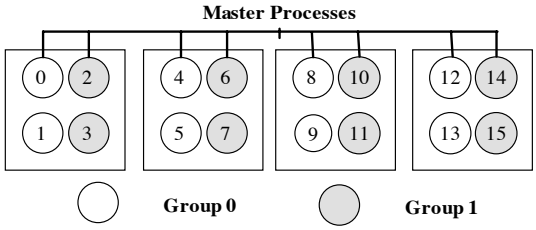

Figure 3. Group structure for 2-group all-gather on a 4-node, 16-core cluster.

In Phase 1, each Master process gathers the data of all intra-node processes that belong to its group. For example in Figure 3, process 0 and 2 gather the data from process 1 and 3, respectively. In Phase 2, an inter-node all-gather is done in each group among Master processes to transfer the data gathered in Phase 1. Each Master process in Phase 2 concurrently performs RDMA Write operations on all its connections. For example, Master process 2 concurrently sends data to Master processes 6, 10, and 14. There will be six concurrently active connections for this case. In Phase 3, all Masters broadcast their received data to their intra-node processes.

5.3.2 Four-group all-gather: A more connection-intensive algorithm that can be implemented on our platform is a 4-group multi-connection aware algorithm. Figure 4 shows the group structure for such an algorithm. The algorithm is done in two phases as follows:

Phase 1: Per-group inter-node multi-connection aware all-gather
Phase 2: Per-node shared memory all-gather

In Phase 1, each process performs an inter-node all-gather within its group. This operation is done in a way similar to Phase 2 of the 2-group algorithm. However, this time 12 concurrent active connections exist per NIC. In Phase 2, each process shares the combined data received from Phase 1 with all other processes on its own node. Note that based on our preliminary results in Figure 1, we do not expect the cards to scale well with this number of connections, at least on our platform.

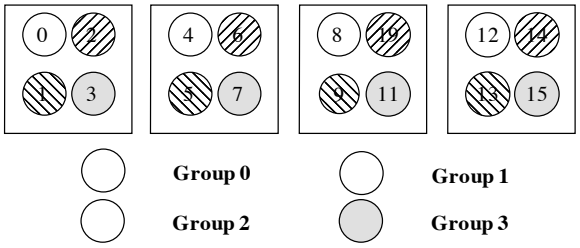

Figure 4. Group structure for 4-group all-gather on a 4-node, 16-core cluster.

5.4 Analysis of Resources Utilization of the Algorithms

In order to estimate the performance, in this section we compare the per-node resource utilization of the proposed algorithms. The analysis is done based on a 4-node cluster, each node having 4 cores. We compare the number of active connections, the amount of shared memory Read/Write operations, and the volume of messages communicated through the network in the algorithms. We assume the message size for the all-gather operation is M bytes. The following notation is used to represent the complexity of the algorithms in Table 1:

αM_βS: $\alpha \times M$ bytes of data are communicated in each of β consecutive steps.
αM_βC: There are β active connections, and $\alpha \times M$ bytes of data are concurrently communicated over each connection.
αM_βW / αM_βR: There are β concurrent shared memory Writes/Reads of $\alpha \times M$ bytes of data each.

As shown in Table 1, the inter-node communication volume is fixed and equal to 12M for all cases. Moving from the 1-group single-connection algorithm to the 1-group multi-connection algorithm, we just change the way the inter-node communication is done. In essence, instead of sending 4M data in three consecutive steps, we send it over three concurrent active connections.
The other notable difference among the algorithms is in their shared memory transactions. Consider the 1-group algorithms on a 4-node (16-core) cluster. In Phase 1, all processes first write their data to the shared memory (1M_4W), and then the Master process reads the data into its send buffer (4M_1R). In Phase 3, the Master process will share with its local processes (16M_1W) all the data it has received from all other Master processes in Phase 2. The local processes will then read the data into their own destination buffers (16M_4R).
The gather-based algorithm has a slightly higher shared memory volume than the 1-group and 2-group algorithms. This means that for larger messages, we do not expect the gather-based algorithm to perform better. With almost the same level of shared memory volume in the 1-group and 2-group algorithms, the 2-group algorithm has an edge over the 1-group and gather-based algorithms, mostly due to the distribution of its inter-node communication among more cores, effectively utilizing both multi-connection and multi-core ability of the InfiniBand cluster. The 4-group algorithm has the lowest shared memory volume. However, we expect to see some overhead due to its aggressive use of simultaneous network connections (over 12 connections).

6. EXPERIMENTAL PLATFORM

The experiments were conducted on a 4-node dedicated multi-core SMP cluster. Each node is a Dell PowerEdge 2850 server having two dual-core 2.8GHz Intel Xeon EM64T processors and 4GB of DDR-2 SDRAM. Each node has a two-port Mellanox ConnectX InfiniBand HCA installed on an x8 PCI-Express slot. Our experiments were done under only one port of the ConnectX HCAs. The machines are interconnected through a Mellanox 24-port MT47396 Infiniscale-III switch.

Table 1. Per-node resource utilization of all-gather algorithms on a 4-node, 16-core cluster.

		1-group single-connection	1-group multi-connection	4-group gather-based multi-connection	2-group multi-connection	4-group multi-connection
Concurrent active connections		1 In , 1 Out	3 In , 3 Out	3 In , 3 Out	6 In , 6 Out	12 In , 12 Out
Network – outbound or inbound data		4M_3S	4M_3C	4M_3C	2M_6C	1M_12C
Shared memory Read/Write operations	**Phase 1**	1M_4W + 4M_1R	1M_4W + 4M_1R	1M_4W + 4M_4R	1M_4W + 2M_2R	-
	Phase 2	-	-	-	-	4M_4W + 16M_4R
	Phase 3	16M_1W + 16M_4R	16M_1W + 16M_4R	16M_1W+ 16M_4R	8M_2W + 16M_4R	-

In terms of software, we used the OpenFabrics Enterprise Distribution, OFED1.2.5, installed over Linux Fedora Core 5, kernel 2.6.20. For MPI, we used MVAPICH-1.0.0-1625.

7. PERFORMANCE RESULTS

In this section, we present the performance results of the proposed algorithms and the native MVAPICH on our cluster, along with a brief description of the implementation. We use cycle-accurate timer to record the time spent in an all-gather operation (1000 iterations) for each process, and then calculate the average all-gather latency among all processes.

For the implementation, we are directly using RDMA Write operations. To avoid the high cost of buffer registration for small messages, we use a copy-based scheme. To avoid the overhead of extra memory copies for large messages, we use a zero-copy scheme. To find the switching point, we have implemented the proposed algorithms using both copy-based and zero-copy techniques, and evaluated them for 1B to 512KB messages. The results shown in Figure 5, for 16 processes on our cluster, are the best results of the two schemes for each algorithm.

In general, our multi-connection aware algorithms perform much better than the native MVAPICH implementation except for 128KB messages and above, due to shared memory bottleneck and poor multi-connection performance for very large messages. As expected, the gather-based algorithm has the best performance for small messages up to 32 bytes, mostly because this algorithm is using multiple cores on the node and lightly utilizes the multi-connection ability of the cards for network communication. The 1-group multi-connection-aware algorithm outperforms other algorithms from 64 bytes to 2KB, since it has a lighter shared memory volume. From 4KB to 64KB, the 2-group multi-connection-aware algorithm performs the best due to a lighter shared memory volume compared to 1-group algorithms and use of multi-cores and multiple concurrent connections.

8. CONCLUSIONS AND FUTURE WORK

Collective operations are the most data intensive communication primitives in MPI. In this paper, we proposed three multi-core and/or multi-connection aware all-gather algorithms over ConnectX InfiniBand networks. Our performance results confirm that utilizing the advanced features of modern network cards to process the communication over multiple simultaneously active connections can greatly improve the performance of collective operations. Our 1-group multi-connection aware algorithm performs better than the 1-group single-connection method for most of the message sizes.

Using the gather-based algorithm, we could also improve the performance by distributing the communication load over multiple cores for small message sizes that involve CPU in communication processing. Another factor that has affected the achieved performance compared to the pure RDMA-based implementation is the use of shared memory. In general, our multi-group multi-connection aware algorithms perform better than the RDMA-only all-gather, except for very large message sizes.

As for the future work, we plan to test our algorithms on larger multi-core clusters. We intend to extend our studies by devising other multi-core and multi-connection aware collective communication primitives. In conclusion, it is our belief that the capabilities of modern networks, such as scalable multi-connection performance, along with the emerging multi-core systems will provide opportunities as well as new challenges for the design of high-performance communication subsystems.

ACKNOWLEDGEMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), Canada Foundation for Innovation (CFI), Ontario Innovation Trust (OIT), Queen's University, and Mellanox Technologies. The authors would like to thank Mellanox for the resources and their technical support.

REFERENCES

[1] InfiniBand Architecture. http://www.infinibandta.org/.
[2] MPI: A Message Passing Interface standard, 1997.
[3] MVAPICH. http://mvapich.cse.ohio-state.edu/.
[4] Mellanox Technologies: ConnectX Architecture, http://www.mellanox.com/products/connectx_architecture.php

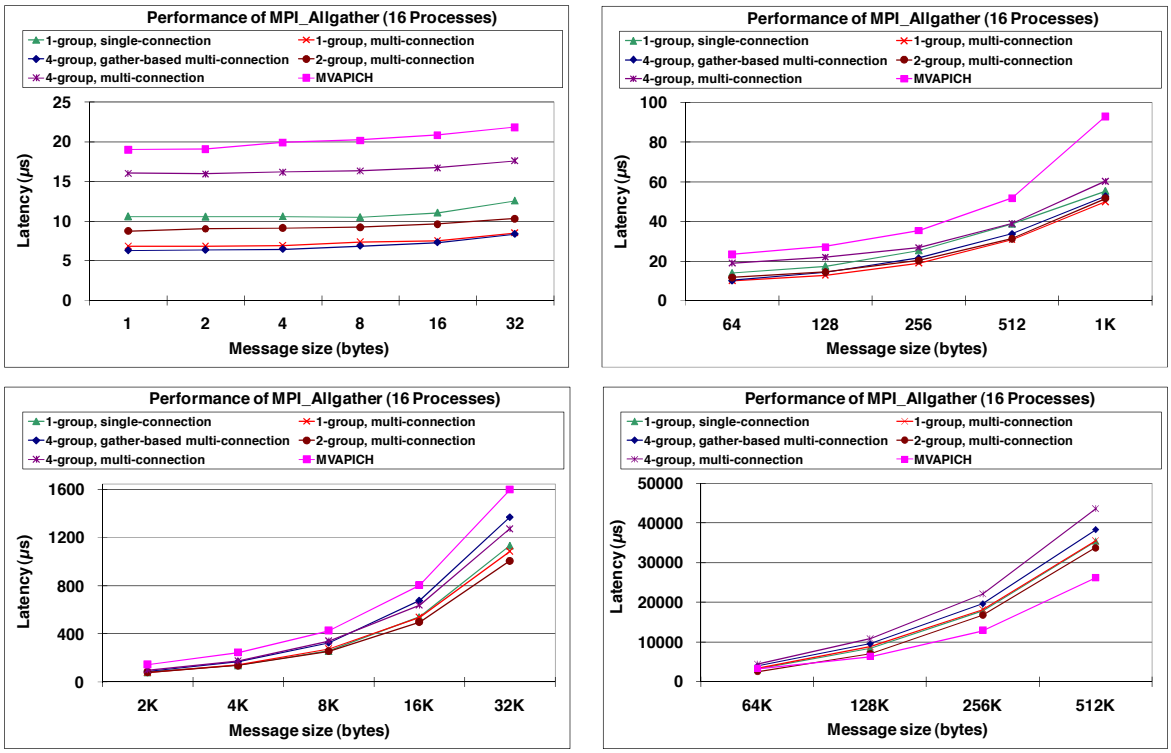

Figure 5. Performance of the proposed MPI_Allgather algorithms on a 4-node, 16-core cluster.

[5] S. Sur, M. Koop, L. Chai, and D.K. Panda, Performance analysis and evaluation of Mellanox ConnectX InfiniBand architecture with multi-core platforms. *Proc. 15th IEEE International Symposium on Hot Interconnects (HotI-15),* Palo Alto, CA, 2007.

[6] S. Sur, U. K.R. Bondhugula, A. Mamidala, H.-W. Jin, and D. K. Panda, High performance RDMA based all-to-all broadcast for InfiniBand clusters. *Proc. International Conference on High Performance Computing (HiPC 2005)*, Goa, India, *2005*.

[7] R. Thakur, R. Rabenseifner, and W. Gropp, Optimization of collective communication operations in MPICH. *International Journal of High Performance Computing Applications, 19*(1), 2005, pp. 49-66.

[8] S. Sur, H.-W. Jin, and D.K. Panda, Efficient and scalable all-to-all personalized exchange for InfiniBand clusters. *Proc. 33rd International Conference on Parallel Processing (ICCP 2004),* Montreal, QC, 2004, 275-282.

[9] D. Buntinas, G. Mercier and W. Gropp, Data transfers between processes in an SMP system: performance study and application to MPI. *Proc. 35th International Conference on Parallel Processing (ICPP 2006),* Columbus, OH, 2006, 487-496.

[10] L. Chai, A. Hartono and D.K. Panda, Designing high performance and scalable MPI intra-node communication support for clusters. *Proc. 8th IEEE International Conference on Cluster Computing (Cluster 2006),* Barcelona, Spain, 2006.

[11] S. Sistare, R. vandeVaart and E. Loh, Optimization of MPI collectives on clusters of large-scale SMPs. *Proc. 1999 Conference on Supercomputing (SC'99)*, 1999.

[12] D. Roweth, A. Pittman and J. Beecroft, Optimized collectives on QsNet^{II}. http://www.quadrics.com/.

[13] V. Tipparaju, J. Nieplocha and D.K. Panda, Fast collective operations using shared and remote memory access protocols on clusters. *Proc. 17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, Nice, France, 2003.

[14] A.R. Mamidala, A. Vishnu, D.K. Panda, Efficient Shared Memory and RDMA based design for MPI_Allgather over InfiniBand, *Proc. EuroPVM/MPI*, Bonn, Germany, 2006.

[15] Y. Qian and A. Afsahi. RDMA-based and SMP-aware multi-port all-gather on multi-rail QsNet^{II} SMP clusters, *Proc. 36th International Conference on Parallel Processing (ICPP 2007)*, Xian, China, *2007*.

[16] S. Coll, E. Frachtenberg, F. Petrini, A. Hoisie and L. Gurvits, Using multirail networks in high performance clusters. *Proc. 3rd IEEE International Conference on Cluster Computing (Cluster 2001),* Newport Beach, CA, 15-24.

[17] V. Tipparaju and J. Nieplocha, Optimizing all-to-all collective communication by exploiting concurrency in modern networks. *Proc. 2005 Conference on Supercomputing* (SC'05), Seattle, WA, 2005.

[18] M.J. Rashti and A. Afsahi, 10-Gigabit iWARP Ethernet: comparative performance analysis with InfiniBand and Myrinet-10G, *7th Workshop on Communication Architecture for Clusters (CAC'07)*, *Proc. 21st International Parallel and Distributed Processing Symposium (IPDPS 2007),* Long Beach, CA, 2007.

[19] R. Kumar, A. Mamidala and D. K, Panda, Scaling Alltoall Collective on Multi-core Systems, *8th Workshop on Communication Architecture for Clusters (CAC'08)*, *Proc. 22nd International Parallel and Distributed Processing Symposium (IPDPS 2008),* Miami, FL, *2008*.