Collective Algorithms for Sub-communicators

Anshul Mittal Thomas George
Yogish Sabharwal
IBM Research India, New Delhi, India
{mittal.anshul,thomasgeorge,ysabharwal}@in.ibm.com

Nikhil Jain
University of Illinois,
Urbana Champaign, USA
nikhil@illinois.edu

Sameer Kumar
IBM T.J.Watson Research Center,
Yorktown Heights, USA
sameerk@us.ibm.com

Abstract

Collective communication over a group of processors is an integral and time consuming component in many HPC applications. Many modern day supercomputers are based on torus interconnects. On such systems, for an irregular communicator comprising of a subset of processors, the algorithms developed so far are not contention free in general and hence non-optimal.

In this paper, we present a novel contention-free algorithm to perform collective operations over a subset of processors in a torus network. We also extend previous work on regular communicators to handle special cases of irregular communicators that occur frequently in parallel scientific applications. For the generic case where multiple node disjoint sub-communicators communicate simultaneously in a loosely synchronous fashion, we propose a novel cooperative approach to route the data for individual sub-communicators without contention. Empirical results demonstrate that our algorithms outperform the optimized MPI collective implementation on IBM's Blue Gene/P supercomputer for large data sizes and random node distributions.

Categories and Subject Descriptors D.m [*Software*]: Miscellaneous

General Terms Performance, Algorithms

Keywords Collectives, Torus, Sub-communicators

1. Introduction

MPI collective routines that perform one-to-many, many-to-one, and many-to-many communications among the processors are amongst the most important and time consuming components in most high performance computing applications. The performance of these MPI collectives is critical for improved scalability and efficiency of parallel scientific applications. With limited bandwith on most systems, for collectives on large messages, avoiding network contention is critical.

Many modern day supercomputers are based on interconnection networks with a k-ary n-cube topology, for example, the Blue Gene/P and Cray XT machines have a 3D torus topology. Applications on such architectures involve collective operations, not only on the entire processor partition (MPI_COMM_WORLD or the *full-communicator* in MPI) but also on sub partitions (i.e., sub-communicators in MPI). For example, in molecular dynamics, each processor communicates with other processors that contain atoms that interact with the atoms in the original processor, resulting in an arbitrary subset of processors forming a sub-communicator.

Optimizing MPI collectives has been a key topic of interest in high performance computing due to its importance in scaling parallel applications. Most of the existing algorithms are either generic algorithms which work well for most network topologies and a wide range of message sizes, or are specifically tailored for a particular class of collectives on a given network topology and typically large message sizes. The later outperform the generic algorithms for the specific target topologies and message sizes. For example, Van de Geijn et.al [3] proposed an algorithm for large message broadcast that has been shown to outperform the binomial algorithm. Optimization for 3D torus networks has been presented in [1] and [2]. However, most of these algorihms have focused on the cases involving *full-communicator*.

Our Contributions. In this paper, we propose algorithms for Broadcast, Reduce, and Allreduce based on the construction of novel edge disjoint spanning trees for a random sub-communicator. We make the following key contributions.

- We propose algorithms to construct contention free spanning trees for performing collective operations over a random sub-communicator on a 3D torus network. An SPI implementation of our approach shows an improvement of $2-4\times$ for Broadcast and Allreduce operations over an optimized MPI implementation available on Blue Gene/P.
- We extend earlier work by in [1] to handle special communicators such as the master/slave scenario where a single processor is excluded from the communicator, and the case where the communicator comprises of a random set of complete parallel 2D planes. An empirical evaluation of our approach demonstrates a speedup of $1.7-5.8\times$ using an SPI implementation.
- We also explore scenarios where multiple node disjoint irregular sub-communicators exist and communicate simultaneously. For such scenarios, with a moderate number of random sub-communicators we obtain a speedup in the range $1.6-2.6\times$.

2. Algorithm

The key new idea in our approach is to decouple the construction of edge-disjoint spanning trees from the collective operation which allows us to handle any arbitrary communicator and/or root. Our approach thus essentially consists of two steps. The first step is the construction of the edge-disjoint spanning trees. This is a one time operation that is extremely fast and is performed during the call to create the sub-communicator. The construction is based on two algorithms - the first one is a novel algorithm that we propose for constructing edge-disjoint spanning trees for arbitrary sub-communicators while the other one is a multi-color algorithm proposed by Faraj et at. [1], which we extend to handle special cases of irregular communicators. The second step is the actual collective operation performed over the spanning tree.

Construction of edge-disjoint spanning tree for an arbitrary sub-communicator in 2-D torus is done by carefully choosing a lin-

PPoPP'12, February 25–29, 2012, New Orleans, Louisiana, USA.
ACM 978-1-4503-1160-1/12/02.

earized (snake like) ordering of nodes which are part of the sub-communicator. The choice of communicating neighbors in this snake pattern ensures that no two communicating nodes use any common link for data transfer. The spanning tree for 3-D torus is obtained by a generalization of the 2-D case.

3. Common Sub-communicators

In this section, we demonstrate the use of the algorithms mentioned in Section 2, on a few commonly occurring scenarios that involve collective communication on sub-communicators. The different scenarios are classified on the basis of the number and kind of simultaneous sub-communicators.

Random communicator: In such cases, our algorithm can be directly used to achieve a single link throughput performance for the collective.

Single communicator with one node missing: Our solution for this scenario is based on extending the algorithm by Faraj et.al. [1]. In the first step, we follow the multi-color rectangular approach to create three edge disjoint trees for the collective operation using links only in the X+, Y+ and Z+ direction. However, we start the tree construction using a *pseudo root* that is at least one hop away from the missing node in every dimension. The missing node is bypassed during the tree construction. In the second step, the directions of the links is adjusted according to the user specified root.

Random subset of parallel planes: Without loss of generality, let us assume that the missing planes are along the Z- axis. Since each of the 2D planes do not have any missing nodes, we first perform phases 1 and 2 of the collective operation where the data is transferred along the X and Y dimensions respectively, as described in [1]. In phase 3, the data is transferred along the Z dimension, with the missing planes in Z dimension being bypassed to transfer the data to the nodes in next 2D plane in the sub-communicator. Phase 4 remains the same as before. In this case, we can obtain maximum throughput for both Broadcast and Allreduce.

Contiguous sub-communicator groups: In this case, all the sub-communicators can be readily mapped onto contiguous nodes on the processor space such that there are no overlapping paths.

Overlapping sub-communicator groups: For overlapping sub-communicators, we propose a novel approach where multiple sub-communicators co-operate to achieve an optimal performance for the collective operations.

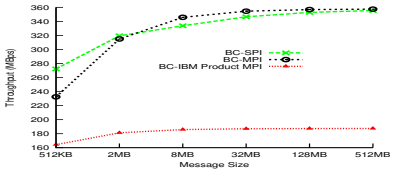

Figure 1. Broadcast on sparse random sub-communicator

4. Results

Hardware: All the experiments were performed on Blue Gene/P which is IBM's massively parallel supercomputer. Each node in BGP supports 850 MBps bidirectional links to each of its nearest neighbors for a total of 5.1GB/s bidirectional bandwidth per node.

Implementation Specifics: We present two versions of our algorithms; (i) an optimized version which uses a lower level API (SPI) and (ii) an MPI version by modifying the MPI stack. We refer to our Broadcast implementation using SPI as *BC-SPI* and the MPI stack implementation as *BC-MPI*. Similarly, we refer to our Allreduce implementations as *AR-SPI* and *AR-MPI* respectively. We compare our results against the IBM's product MPI referred to as *BC-IBM Product MPI* and *AR-IBM Product MPI* for Broadcast and Allreduce, respectively.

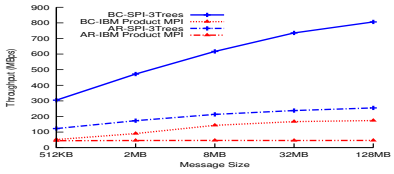

Figure 2. One node missing sub-communicator

We present the performance results of our Broadcast implementation for increasing message sizes in the range 512KB-512MB for a sparse (31/1024 nodes) random sub-communicator in Figure 1. Similar results were obtained for Allreduce. Figure 2 shows the plot of throughput vs. message size for Broadcast and Allreduce on sub-communictor with one missing node.

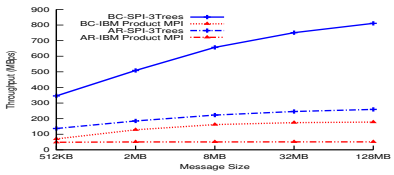

Figure 3. Missing plane sub-communicator.

In Figure 3, we show the plot of throughput vs. message size for Broadcast and Allreduce operations for sub-communicator with missing planes.

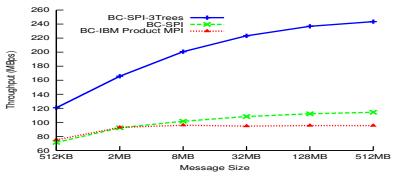

Figure 4. Broacast with mutiple random sub-communicators.

Figures 4 and 5 show the performance of our algorithms for the case when multiple random sub-communicators perform Broadcast and Allreduce respectively.

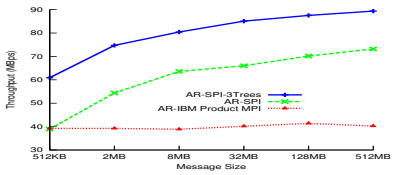

Figure 5. Allreduce with multiple random sub-communicators.

References

[1] A. Faraj, S. Kumar, B. Smith, A. Mamidala, J. Gunnels, and P. Heidelberger. MPI collective communications on the Blue Gene/P supercomputer: algorithms and optimizations". In *ICS*, pages 489–490, 2009.

[2] N. Jain and Y. Sabharwal. Optimal bucket algorithms for large MPI collectives on torus interconnects. In *ICS*, pages 27–36, 2010.

[3] M. Shroff and R. A. V. D. Geijn. Collmark: Mpi collective communication benchmark. Technical report, 2000.