COLLECTIVE ALGORITHMS FOR SUB-COMMUNICATORS

Anshul Mittal
Stanford University

Nikhil Jain
University of Illinois
Urbana Champaign

Thomas George
IBM Research India
New Delhi

Sameer Kumar
IBM T.J.W. Research Center
Yorktown Heights, USA

Yogish Sabharwal
IBM Research India
New Delhi

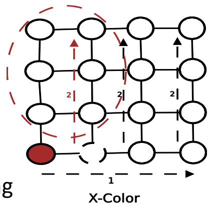
Collective communication over a group of processors is an integral and time consuming component in many high performance computing applications.
We present novel techniques to perform collective operations over a subset of processors in a torus network obtaining over 5x speedup over current best.

Motivation

- **Why Collectives?**
  Performance of MPI collectives is critical for improved scalability and efficiency, for large messages network contention need to be avoided
- **Why Sub-communicators?**
  Many applications involve collective operations on sub partitions (i.e., sub-communicators in MPI).
- **Why a new approach?**
  The algorithms developed so far are not contention free for a random sub-communicator

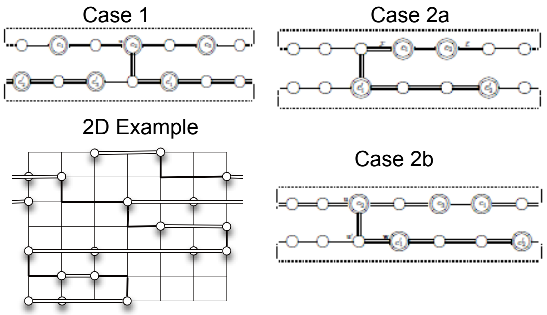
Common Sub-communicators

- **Random Communicator**
  Our algorithm achieves a single link throughput performance
- **Single Missing Node**
  The tree is constructed using a pseudo root that is at least one hop away from the missing node in every dimension. The missing node is by-passed during the tree construction and the directions of the links are adjusted according to the user specified root.



- **Random subset of parallel planes**
  Phase 1 and 2 proceeds as described in [1] assuming that the missing planes are in the Z direction. For 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.

[1] Faraj et.al., MPI collective communications on the Blue Gene/P supercomputer: algorithms and optimizations". In ICS 2009



Algorithm

```
1. Let $y_1, y_2, \ldots, y_r$ be a sequence of $y$-coordinates that
   contain c-nodes such that the $y$-distance between two
   consecutive $y$-coordinates in this sequence $\leq d_y/2$,
   i.e., $(y_{i+1} - y_i) \mod d_y \leq d_y/2$ for all $1 \leq i < r$.
   Let $\ell(i)$ denote the line corresp. to $y_i$ for $1 \leq i \leq r$.

2. Let $< c_1, .., c_m >$ be the chain of c-nodes in line $\ell(1)$.
   Connect the c-nodes in the chain in a straight line.

3. For $i = 1$ to $r - 1$ do
   Connect $\ell(i)$ to $\ell(i+1)$ based on the following cases:

   Case 1: $\ell(i+1)$ does not have a gap of $d_x/2+1$
   a. Select any c-node in $\ell(i)$, say $u$, as the connector.
   b. Progress down from $u$ to $\ell(i+1)$ and then move
      towards right in $\ell(i+1)$, say $c_1'$
   c. Form a chain in $\ell(i+1)$ starting with $c_1'$ and cover
      all the c-nodes in $\ell(i+1)$ proceeding in the same
      direction.

   Case 2: $\ell(i+1)$ has a gap of $d_x/2+1$
      Let $< c_1, c_2, \ldots, c_m >$ be the chain of $\ell(i)$
      where $c_1$ is the head node and $c_m$ is the tail node

      Case 2(a): there exists a c-node, $u$, in $\ell(i)$
        that is directly above a node not contained in
        the chain of $\ell(i+1)$
      a. Progress down from $u$ to $\ell(i+1)$ and then
      move towards the closest node in $\ell(i+1)$
      c. Form a chain starting with this node and cover
         all the c-nodes in $\ell(i+1)$ proceeding in the
         same direction.

      Case 2(b): All c-nodes in $\ell(i)$ are above nodes
        contained in the chain of $\ell(i+1)$
        Let $u \in \{c_1, c_m\}$ be a node that has an edge
        adjacent to it in $\ell(i)$ that is not used in the tree
      a. Progress from $u$ in the direction of the unused
         edge till you reach above the first or last
         c-node of the chain of $\ell(i+1)$
      b. Progress down to the c-node of $\ell(i+1)$
      c. Form a chain starting with this node and cover
         all the c-nodes in $\ell(i+1)$
```

Results

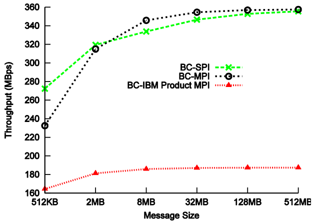
All the experiments were performed on Blue Gene/P.

Implemented versions

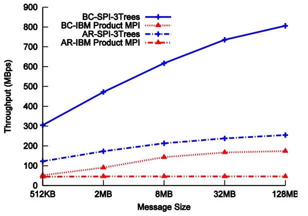
1. Optimized version which uses lower level API (SPI)
2. MPI version by modifying the MPI stack

Results

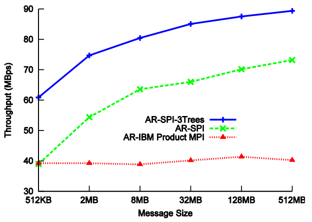
Our MPI implementation obtains a speedup of 1.5-3x over IBM's product MPI for a single random communicator. Our SPI implementation achieves a speedup of 1.7-5.8x for special sub-communicators. For multiple sub-comms., we observe a speedup of 1.6-2.6x.



Broadcast on sparse random sub-communicator



Broadcast and Allreduce on missing planes



Simultaneous Allreduce on multiple sub-communicators