A practically constant-time MPI Broadcast Algorithm for large-scale InfiniBand Clusters with Multicast

T. Hoefler, C. Siebert, W. Rehm

Open Systems Lab
Indiana University
Bloomington, USA

Computer Architecture Group
Chemnitz University of Technology
Chemnitz, Germany

IPDPS’07 - CAC’07 Workshop
Long Beach, CA, USA
26th March 2007
Introduction

- MPI is (still) the de-facto standard in parallel programming
- systems are going to extreme scale
- applications start to use high scalability
- collective operations are an important tool
- scalable collective operations are very important

Our approach

Use special hardware features to improve scalability of collective operations.
**Introduction**

- MPI is (still) the de-facto standard in parallel programming
- systems are going to extreme scale
- applications start to use high scalability
- collective operations are an important tool
- scalable collective operations are very important

**Our approach**

Use special hardware features to improve scalability of collective operations.
Traditional Approach

- ensure scalability with $O(\log_2 P)$ algorithms
- optimized implementations available for different collectives
- looks promising, but:
  - grows fast for small process-counts (e.g., 256 processes need $t = 8 \cdot t_{send}$)
  - processes are skewed by the algorithm (e.g., node 1 leaves the tree faster than node 7)
Multicast Support

Multicast characteristics
- unreliable
- no guaranteed in-order delivery
- datagrams limited in size (MTU)
- MC groups must be network-wide unique

MPI Interface
- reliable transmission
- virtually unlimited message size
- multiple independent MPI jobs on a single network
## Multicast Support

### Multicast characteristics
- unreliable
- no guaranteed in-order delivery
- datagrams limited in size (MTU)
- MC groups must be network-wide unique

### MPI Interface
- reliable transmission
- virtually unlimited message size
- multiple independent MPI jobs on a single network
Traditional Approaches to Ensure Reliability

**ACK Schemes**
- linear ACK - hot-spot problems
- tree-based ACK - high latency
- co-root scheme - combination of both, similar problems
- every (co-)root waits for last process in its group
- retransmission timeout

**NACK Schemes**
- topologies similar to ACK
- root has to wait for some time (or save the message buffer)
- timeout very hard to determine and not reliable
- synchronization problems (delayed processes?)

T. Hoefler, C. Siebert, W. Rehm

Constant-time Multicast
Traditional Approaches to Ensure Reliability

**ACK Schemes**
- linear ACK - hot-spot problems
- tree-based ACK - high latency
- co-root scheme - combination of both, similar problems
- every (co-)root waits for last process in its group
- retransmission timeout

**NACK Schemes**
- topologies similar to ACK
- root has to wait for some time (or save the message buffer)
- timeout very hard to determine and not reliable
- synchronization problems (delayed processes?)
A new Approach

The new algorithm

- two-stage approach
- packets are fragmented to the MTU
- first stage sends fragmented message via Multicast
- processes that received the fragment correctly become new root
- second stage performs a reliable ring-broadcast
- ⇒ highest possible parallelism
The algorithm

T. Hoefler, C. Siebert, W. Rehm
The algorithm
problematic if multiple MPI jobs run in a subnet

ideal solution: MADCAP for InfiniBand™
does not exist (subnet-manager?)
select MCGID randomly
carefully seeded cryptographically secure pseudorandom number generator (Blum-Blum-Shub)

112 bit address space
collision probability for 1000 groups: $10^{-18}$
### Packet Format

<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>BID</th>
<th>Data (Payload)</th>
<th>CRC−32</th>
</tr>
</thead>
</table>

**Fields**

- **Sequence Number**: number of fragment
- **BID**: Broadcast Identifier
- **CRC**: (optional) checksum
- **packet error rate**: 0.287%
- implemented as collv1 component
- MCGID is selected per communicator
- one UD QP per communicator (scalable)
- \( n \) pre-posted RR\( s \) on this QP (selectable, default 5)
- use to “tuned” for small communicators/large messages
- API independent macro layer for OFED/MVAPI
Performance Results

Benchmark Environment

- odin cluster at Indiana University
- 128 InfiniBand™ nodes
- 2Ghz dual core AMD Opteron(tm) processor 270
- → 1-byte IMB latency
Performance Results

- 1-byte latency for each rank

![Graph showing performance results](image-url)
Performance Results

1-byte latency or rank 1

Time in microseconds vs. Communicator Size

IB
TUNED

T. Hoefler, C. Siebert, W. Rehm
Performance Results

- 1-byte latency or rank $N - 1$
Conclusions

- a new algorithm to use Multicast for MPI_BCAST
- massively parallel scheme to deal with reliability issues
- (average) constant-time \((2 \cdot t_{\text{send}})\) bcast implementation
- tree-based algorithms cause process skew
- the newly proposed algorithm does not skew processes

Future Work

- investigate other collective operations
- investigate the influence of process skew on applications
- investigate large message support
Conclusions

- a new algorithm to use Multicast for MPI_BCAST
- massively parallel scheme to deal with reliability issues
- (average) constant-time \((2 \cdot t_{\text{send}})\) bcast implementation
- tree-based algorithms cause process skew
- the newly proposed algorithm does not skew processes

Future Work

- investigate other collective operations
- investigate the influence of process skew on applications
- investigate large message support