Fast and Scalable MPI-Level Broadcast Using InfiniBand's Hardware Multicast Support

J. Liu A. Mamidala D. K. Panda

Computer Science and Engineering
The Ohio State University

Presentation Outline

- Introduction and Overview
- Designing MPI_Bcast with InfiniBand Multicast
- Performance Evaluation
- Conclusions

Introduction

- MPI provides both point-to-point and collective communication
- Efficient and scalable collective communication is very important to high performance applications
- Modern interconnects provide certain support in hardware for collective communication
  - Hardware multicast in InfiniBand
- Collective at hardware level usually has different semantics from the MPI level

Motivation

- Can we exploit InfiniBand hardware multicast in MPI collective communication?
  - Focus on MPI_Bcast
- How can we bridge the semantic gap of InfiniBand multicast and MPI_Bcast?
  - Efficiency
  - Scalability

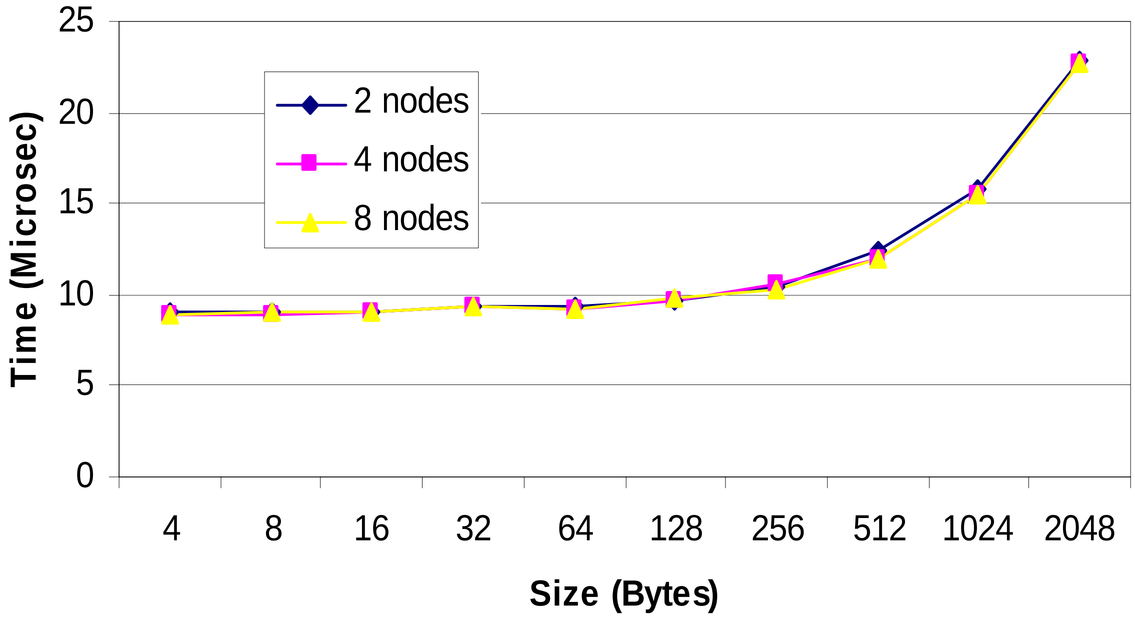
InfiniBand Overview

- Industry standard for high speed interconnect
- High performance
- Many novel features
  - Hardware multicast
  - RDMA, atomic operations, QoS, etc

InfiniBand Multicast

- Only one send operation is needed to initiate the multicast
- Message is delivered to multiple destinations by hardware
- Available in Unreliable Datagram (UD) mode
  - Unreliable
  - Un-ordered
  - Cannot exceed MTU
    - 2 KB in current hardware

Multicast Performance



- Good latency for small messages
- Very scalable wrt the number of destinations
- Less traffic
- Independent progress

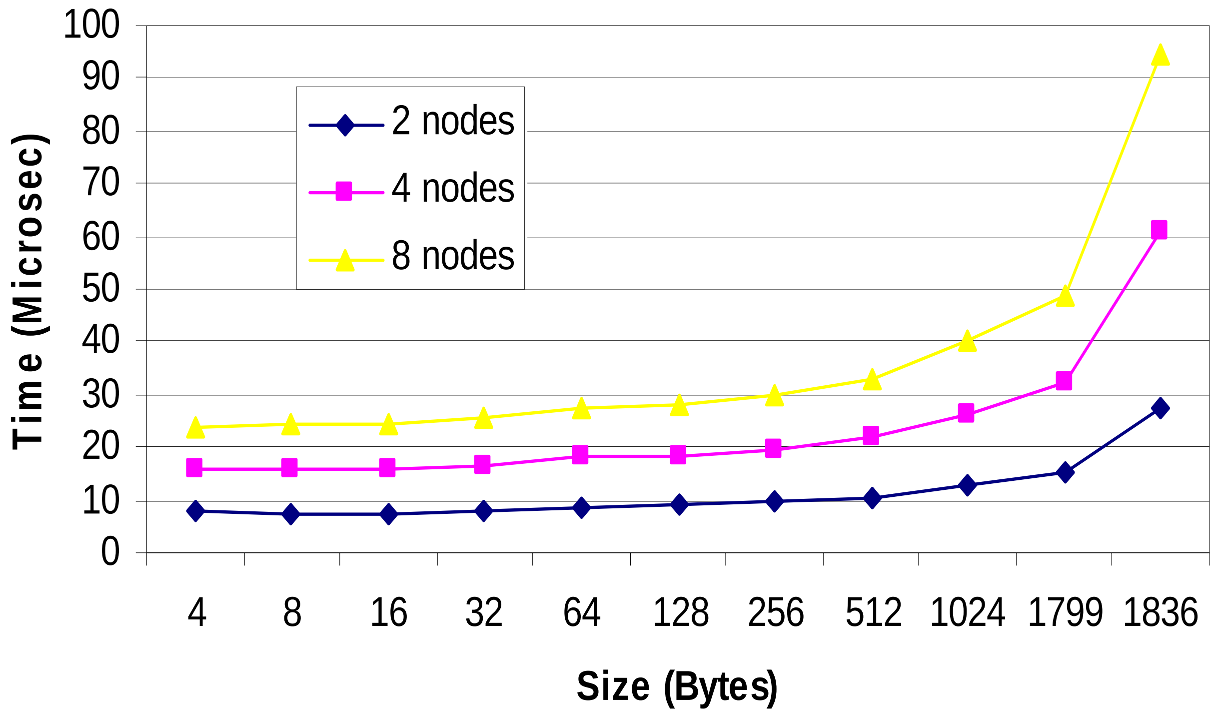
MPI_Bcast Overview

- A very commonly used collective operations
- Delivers a message to all process in a communication group
  - Reliable
  - Ordered
  - Message size can be very large
- Usually implemented on top of MPI point-to-point communication
  - Current approach in MVAPICH

MVAPICH

- MPI implementation of InfiniBand
  - Open source
  - Used by many organizations world-wide
- Powering the 3rd, 111th, 116th most powerful supercomputers in the world
  - Virginia Tech System X (2200 processor G5 cluster)
    - As mentioned in Dr. Varadarajan's talk yesterday
  - Sandia National Lab (256 Processor Xeon cluster)
  - Los Alamos National Lab (512 Processor Opteron cluster)

MPI_Bcast Performance in Current MVAPICH



- Not scalable wrt the number of destinations
- More traffic
- Progress depends on intermediate nodes

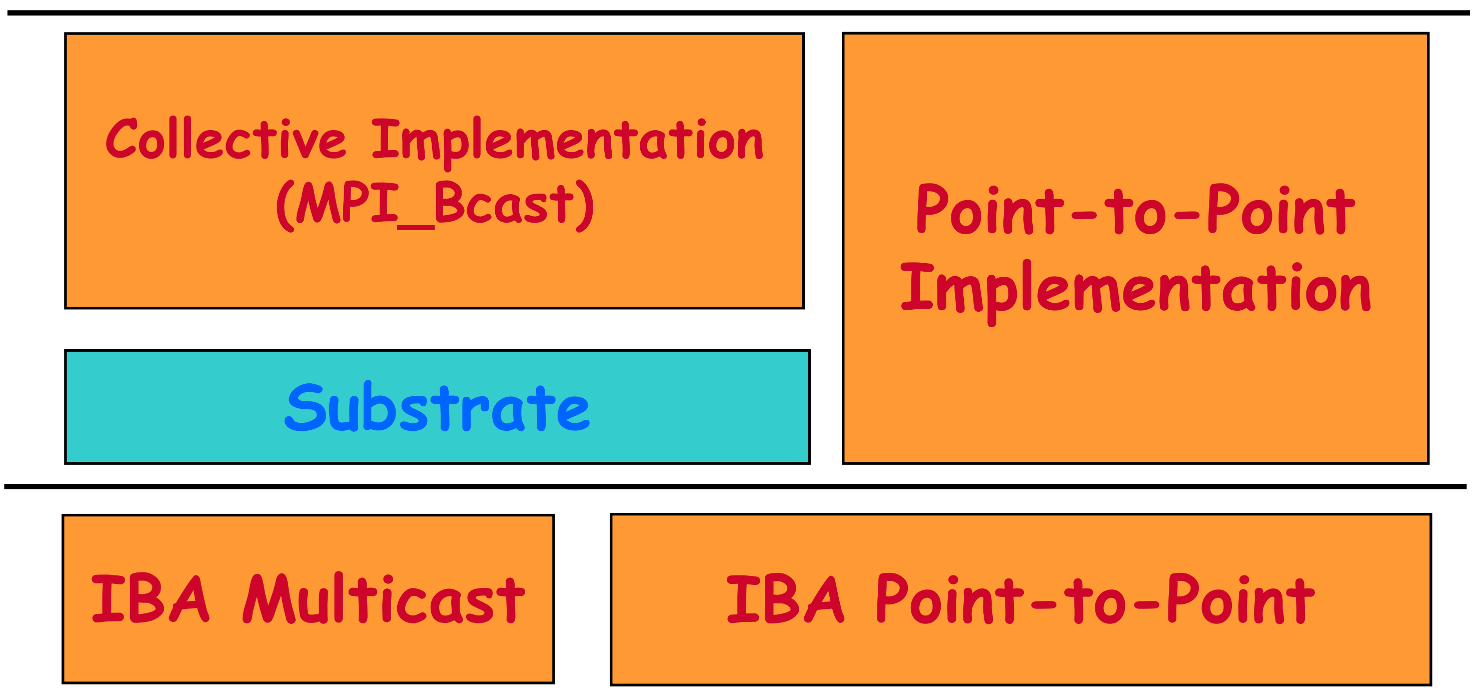
Presentation Outline

- Introduction and Overview
- Designing MPI_Bcast with InfiniBand Multicast
- Performance Evaluation
- Conclusions

Design Challenges

- Semantic gap between InfiniBand multicast and MPI_Bcast
  - Reliability
  - Message ordering
  - Message size
- Need to bridge this gap
  - Performance
  - Scalability

Design Architecture



- A substrate to bridge the semantic gap with low overhead

Outline of Design Issues

- Basic design
- Sliding window based design
- Avoiding ACK implosion
- Reducing ACK traffic
- Dealing with large messages
- Detailed design issues

Basic Design

- Root sends message using multicast
- Receivers send back ACK
- Root blocks until all ACKs come
- Problems
  - High overhead at root because it needs to block
  - ACK implosion
  - ACK traffic

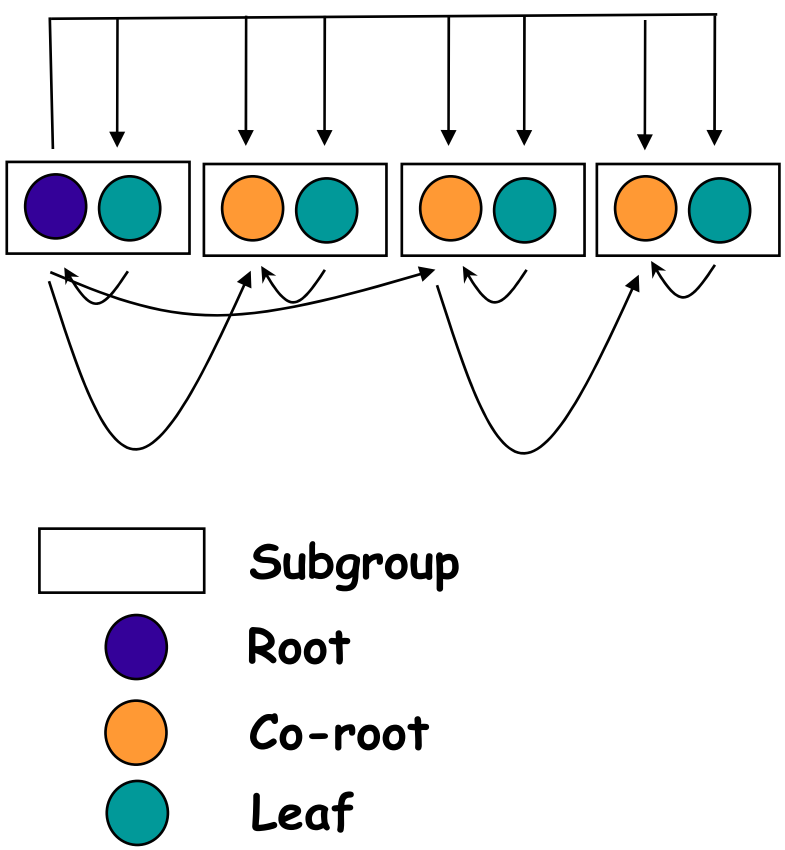
Sliding Window Based Design

- Use a window of buffers
- Root does not block
- ACKs can be collected in the background
- Root needs to block if running out of buffers
- Problems
  - ACK implosion
  - ACK traffic

Avoiding ACK Implosion

- Use a hierarchical structure for ACK collection
- Tree based approach
  - Dependence on intermediate nodes
  - Prone to false retransmission
  - large retransmission traffic

Co-Root Based Approach



- Two level hierarchy
- Root does multicast
- Root does a broadcast to all co-roots
  - Use point-to-point
  - Reliable
- Root and co-roots responsible for ACK collective in its subgroup

Advantages of Co-Root Based Approach

- More even load distribution
  - Co-roots help with *both* ACK collection and retransmission
- Better communication progress
  - No intermediate nodes
- Less retransmission traffic
  - Co-roots keep track of its subgroup

Reducing ACK Traffic

- Delaying ACKs
  - Combining multiple ACKs
    - ACK for every M broadcast messages
  - Piggybacking
    - Attach ACK with other messages

Handling Large Messages

- Divide the message into multiple chunks
- Use multicast to send each chunk
- Problems
  - Copying cost

Detailed Design Issues

- Buffer management
- Handling out-of-order and duplicated messages
- Timeout and retransmission
- Flow control
- RDMA based ACK collection

Presentation Outline

- Introduction and Overview
- Designing MPI_Bcast with InfiniBand Multicast
- Performance Evaluation
- Conclusions

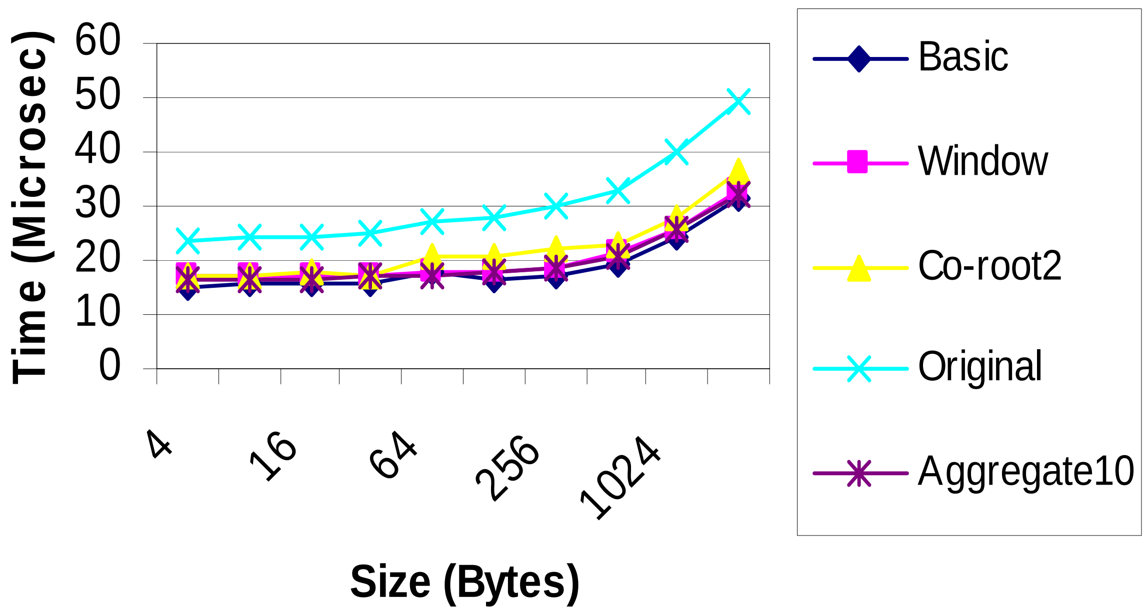
Experimental Testbed

- 8 SuperMicro SUPER P4DL6 nodes (2.4 GHz Xeon, 400MHz FSB, 512K L2 cache)
- Mellanox InfiniHost MT23108 4X HCAs (A1 silicon), PCI-X 66bit 133MHz
- Mellanox InfiniScale MT43132 switch

Schemes Used in the Experiments

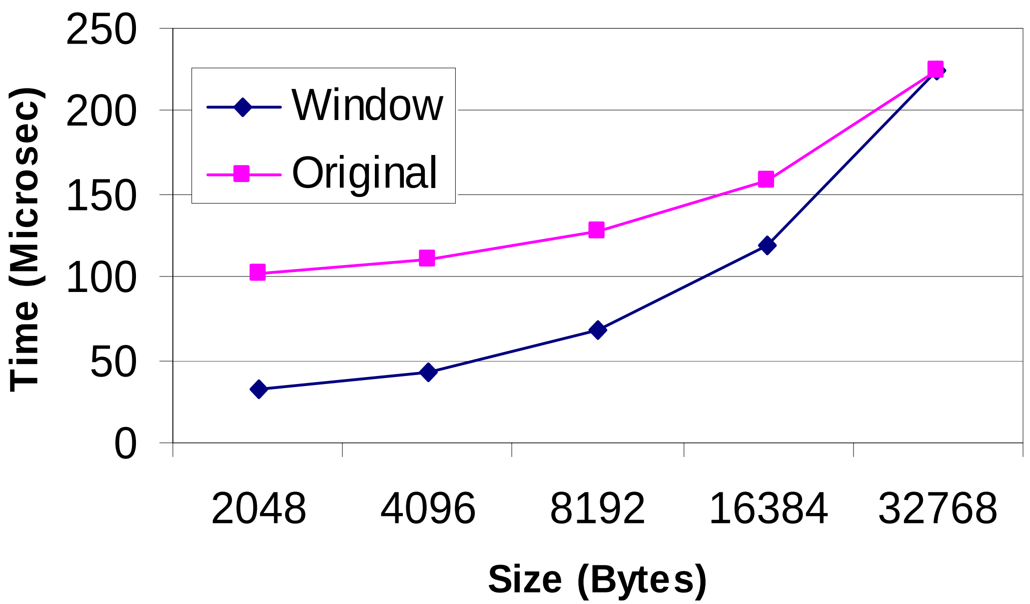
- Original
  - Original MVAPICH implementation based on point-to-point communication
- Basic
  - Basic design
- Window based schemes
  - Window
    - Sliding window based design
  - Co-root2
    - Sliding window + one co-root
  - Aggregate 10
    - Sliding window + ACK for every ten broadcast

MPI_Bcast Latency on 8 nodes (Small Messages)



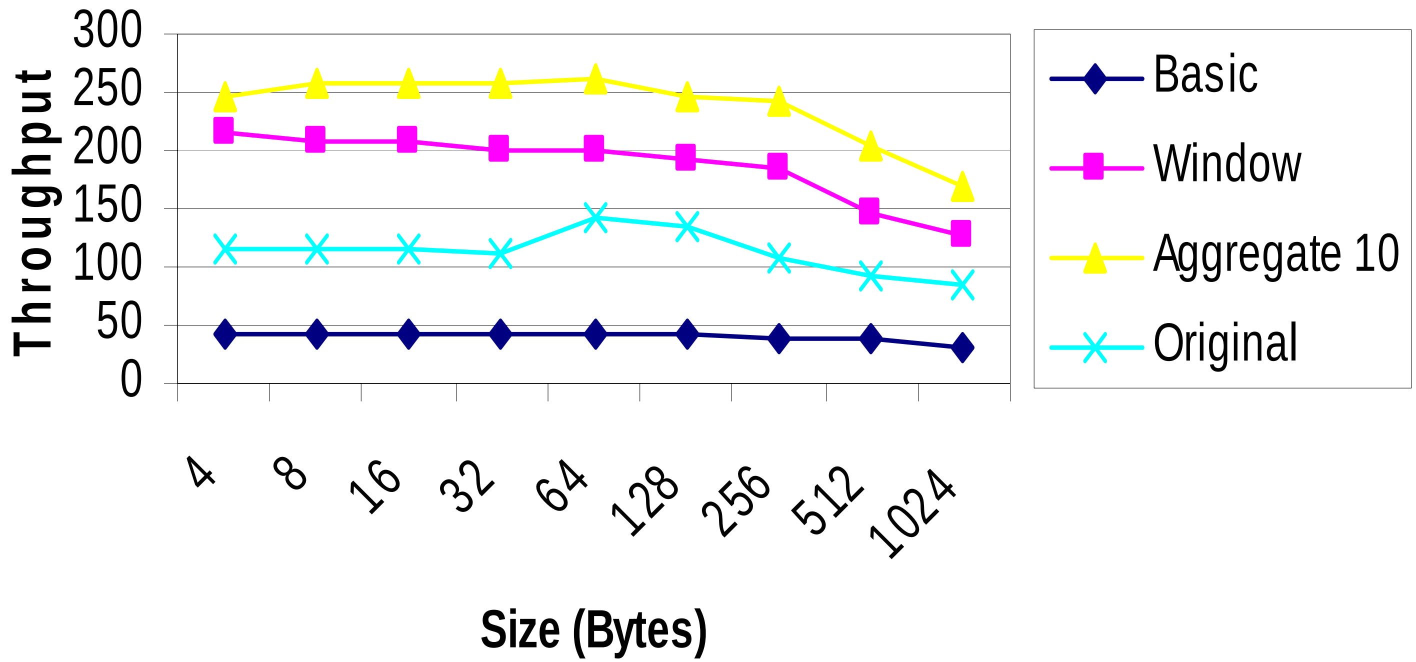
- All new schemes perform comparably
- Up to 58% compared with original implementation

MPI_Bcast Latency on 8 Nodes (Large Messages)



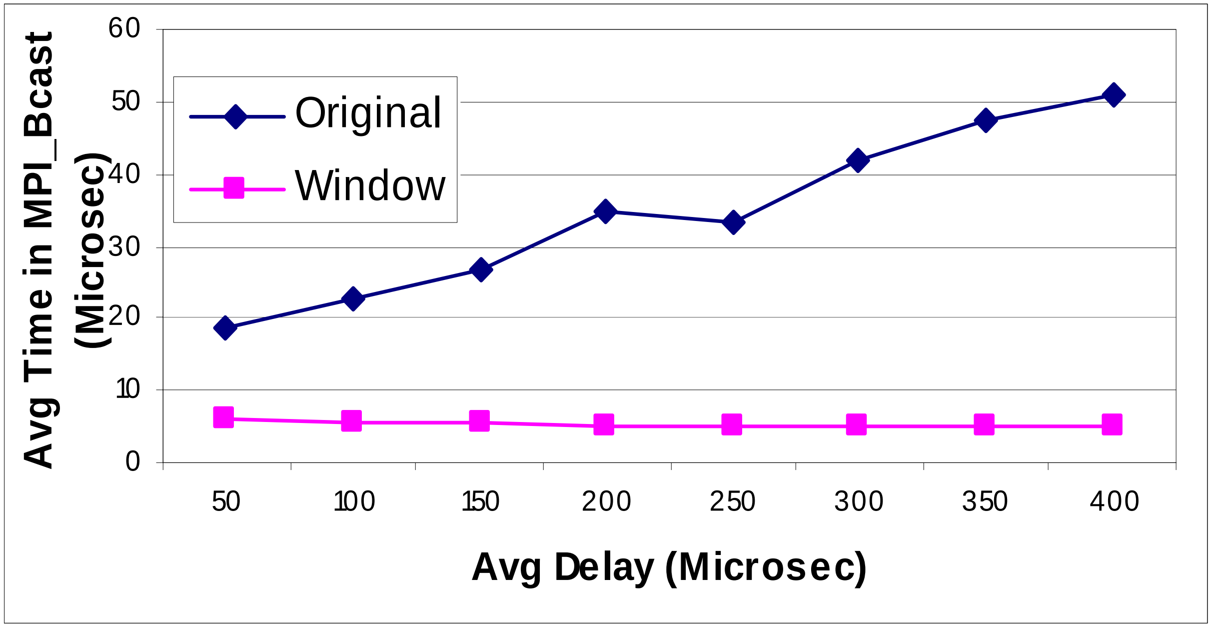
- Up to 210% improvement
- Worse than the original implementation for messages larger than 32 KB due to extra copies

MPI_Bcast Throughput



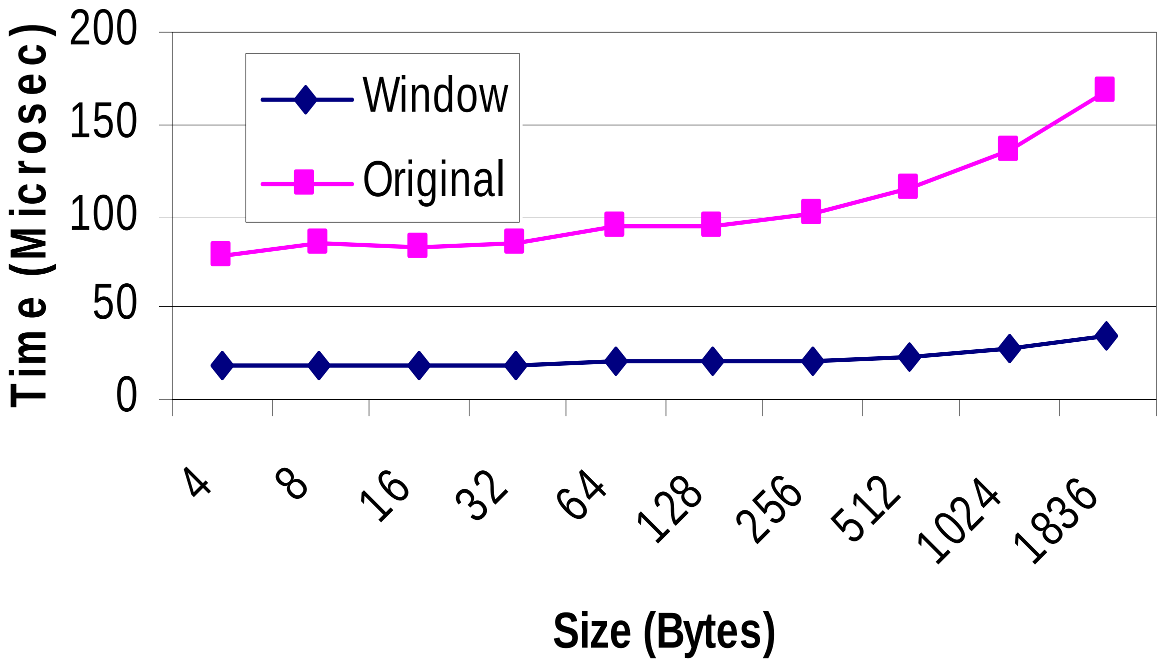
- Measure how fast back-to-back MPI_Bcast can be issued and finished
- Up to 112% improvement for Aggregate10

Impact of Process Skew



- Random skew is added before MPI_Bcast at each receiver
- Measure time spent in MPI_Bcast
- Hardware multicast based scheme performs significantly better

MPI_Bcast Latency on 1024 Nodes (Based on Analytical Model)



- Systems similar to those in our current testbed
- Window based scheme achieves less than 20 us latency for 4 byte messages, less than 40 for 1836 bytes
- Up to 4.86 time improvement

Conclusions

- Designs of MPI_Bcast using InfiniBand multicast
  - A substrate to bridge the semantic gap
  - Techniques to improve performance and scalability
- Performance evaluation on 8 nodes
  - Up to 58% improvement in latency
  - Up to 112% improvement in throughput
  - Better tolerance of skew
- Analytical model
  - Less than 20 us latency on 1024 nodes
  - Up to 4.86 times improvement

Future Work

- Integrate into MVAPICH release
- Explore NACK based schemes
- Evaluate using larger testbeds
- Explore zero copy approaches for large messages using InfiniBand multicast
- Work on other collectives

Web Pointers

NBC home page

http://www.cis.ohio-state.edu/~panda/
http://nowlab.cis.ohio-state.edu/

MVAPICH home page

http://nowlab.cis.ohio-state.edu/projects/mpi-iba/