Design and Implementation of MPICH2 over InfiniBand with RDMA Support

J. Liu, W. Jiang, P. Wyckoff, D. K. Panda,
D. Ashton, D. Buntinas, W. Gropp, B. Toonen

Computer Science and Engineering, The Ohio State University

Ohio Supercomputer Center

Mathematics and Computer Science Division, Argonne National Laboratory

Presentation Outline

- Introduction and Motivation
- Background
  - InfiniBand
  - MPICH2
  - MPICH2 RDMA Channel Interface
- Design and Optimization
  - Basic Design and Optimization
  - Zero Copy Design
  - Performance Comparison
- Conclusion

Introduction

- InfiniBand is becoming popular for parallel computing
  - High performance
  - Many novel feature such as RDMA
- MPI is the *de facto* standard of writing parallel applications
  - MPICH from Argonne is one of the most popular MPI implementation
  - MPICH2 is the next generation of MPICH

Motivation

- Optimizing MPICH2 using InfiniBand RDMA operations
  - Focus on MPI-1 functions
- Taking advantage of the new RDMA channel interface in MPICH2
  - RDMA channel is a very simple interface
  - But, can it achieve high performance?

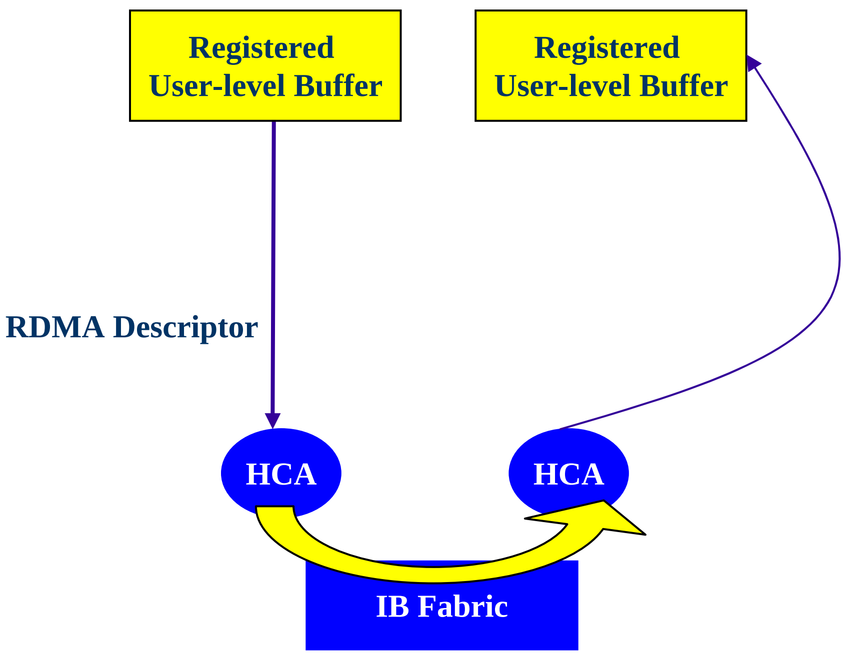
Presentation Outline

- Introduction and Motivation
- Background
  - InfiniBand
  - MPICH2
  - MPICH2 RDMA Channel Interface
- Design and Optimization
  - Basic Design and Optimization
  - Zero Copy Design
  - Performance Comparison
- Conclusion

InfiniBand Overview

- Industry standard interconnect
- High performance
  - Low latency
  - High bandwidth
- Many novel feature
  - RDMA
  - Multicast, atomic operation, QoS, etc

InfiniBand RDMA



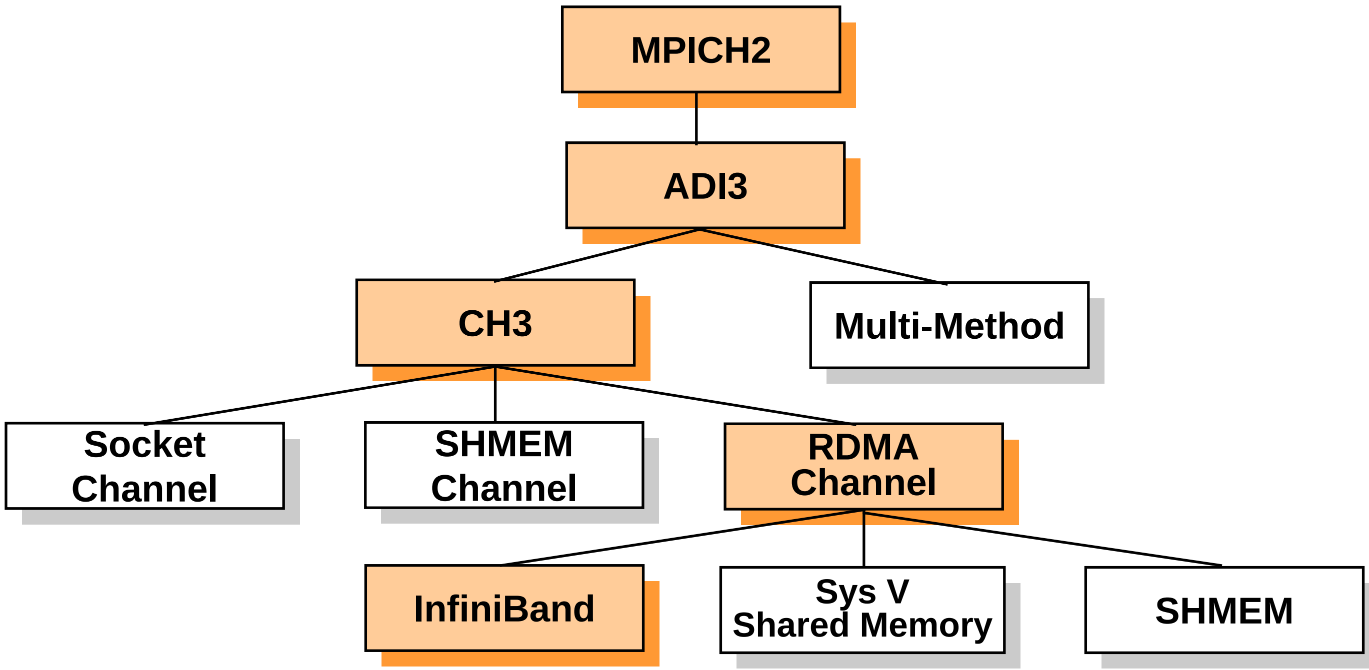
RDMA Model

- Sender directly accesses receiver's memory
- Transparent to receiver side software
- Better performance than send/receive in current InfiniBand hardware

MPICH2 Overview

- Successor of MPICH
- Supports both MPI-1 and MPI-2
  - We focus on MPI-1 functions in this paper
- Completely new design
  - Performance
  - Flexibility
  - Portability
- Porting can be done at different levels
  - ADI3
  - CH3
  - RDMA Channel Interface

MPICH2 Implementation Structure

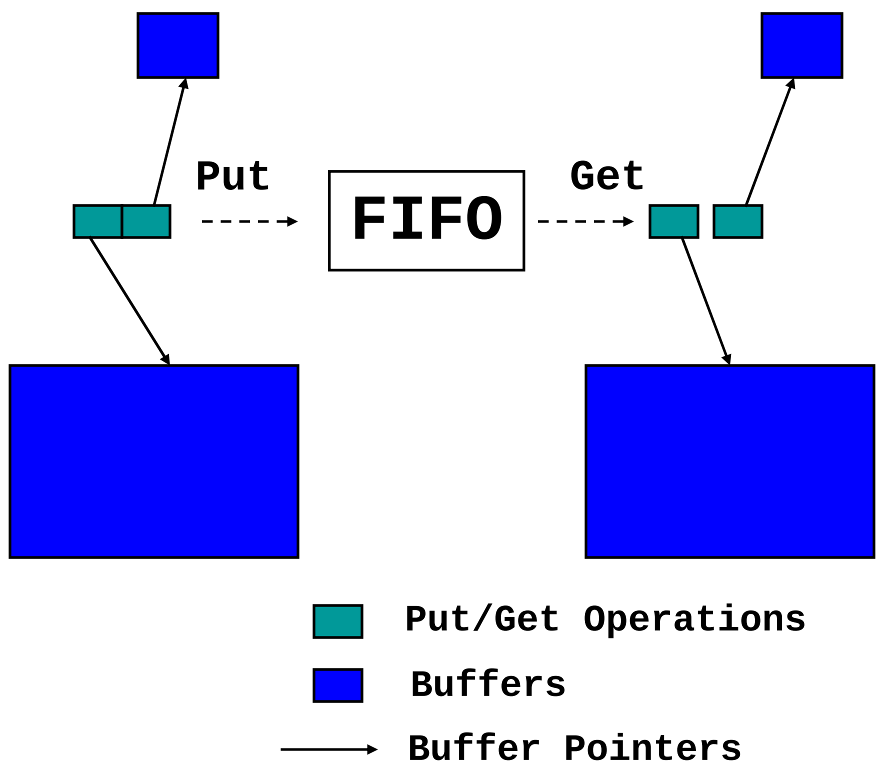


- We focus on implementing RDMA Channel Interface over InfiniBand

RDMA Channel Interface

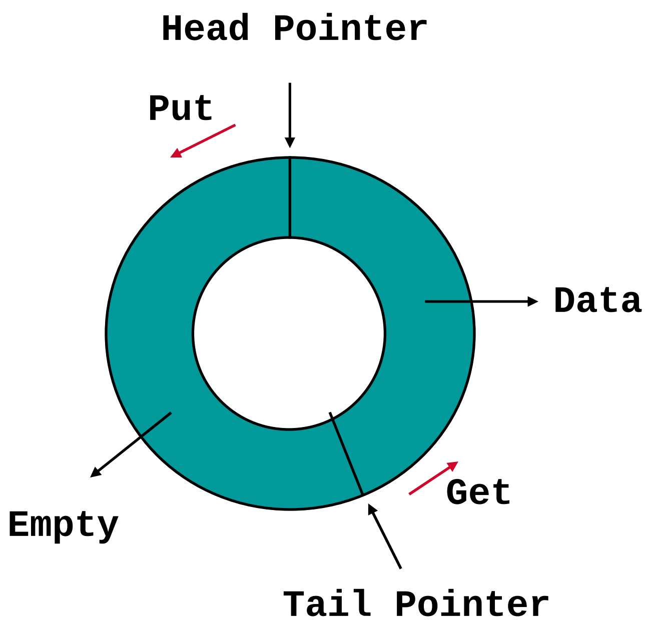
- Very simple interface
  - Three functions for process management, initialization and finalization
  - Two functions for communication
    - Put
    - Get

Put and Get Functions



- A logically shared FIFO channel between sender and receiver
- Put writes into the channel
- Get reads from the channel
- Both functions accept a list of buffers
- Building blocks for all other communication

Example Implementation of Put and Get with Globally Shared Memory



- Buffer pool, head and tail pointers in shared memory
- Put: Write data and advance head pointer
- Get: Read data and advance tail pointer

Presentation Outline

- Introduction and Motivation
- Background
  - InfiniBand
  - MPICH2
  - MPICH2 RDMA Channel Interface
- Design and Optimization
  - Basic Design and Optimization
  - Zero Copy Design
  - Performance Comparison
- Conclusion

Basic Design

- Based on the design for globally shared memory
- Buffer pool at the receiver
  - Sender uses RDMA write
  - Receiver uses local memory read
- Keep two copies of head and tail pointers
  - Use RDMA write to make them consistent

Put in Basic Design

- Put:
  - Determine available buffer space
  - Copy data to pre-registered buffer
  - Write data using RDMA write
  - Adjust local head pointer
  - Adjust remote head pointer using RDMA write

Get in Basic Design

- Get:
  - Determine available new data
  - Copy data to user buffer
  - Adjust local tail pointer
  - Adjust remote tail pointer using RDMA write

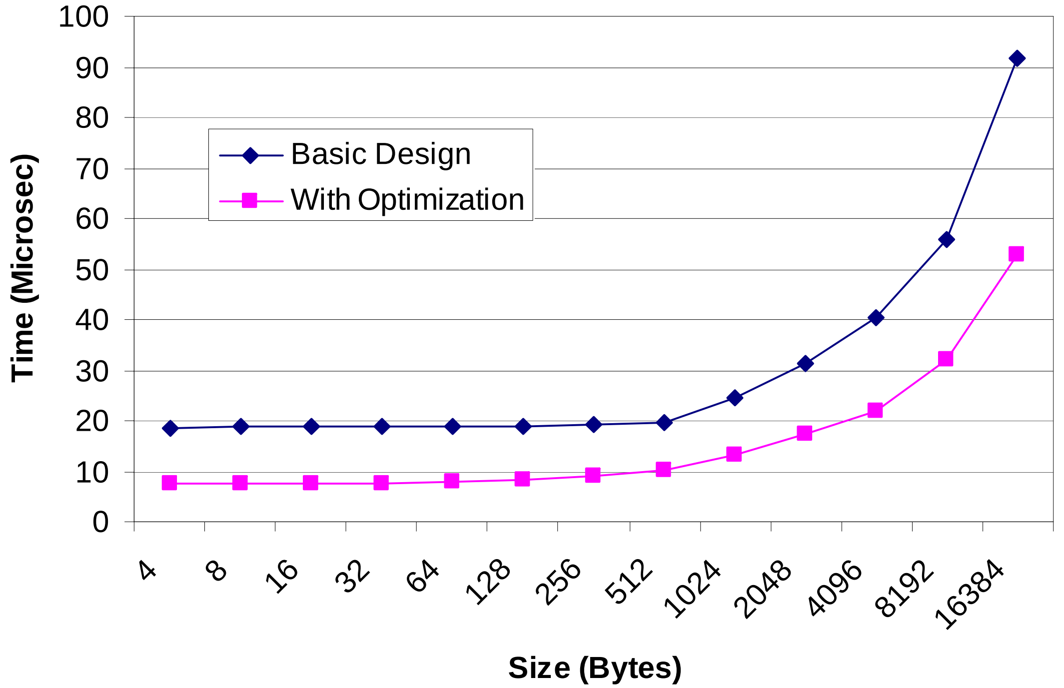
Optimizing the Basic Design

- Piggybacking Pointer Updates
  - Combine data and remote head pointer update at the sender side
  - Update remote tail pointer lazily at the receiver side
- Pipelining large messages
  - Divide large message into chunks
  - Overlap copy with RDMA operation

Experimental Testbed

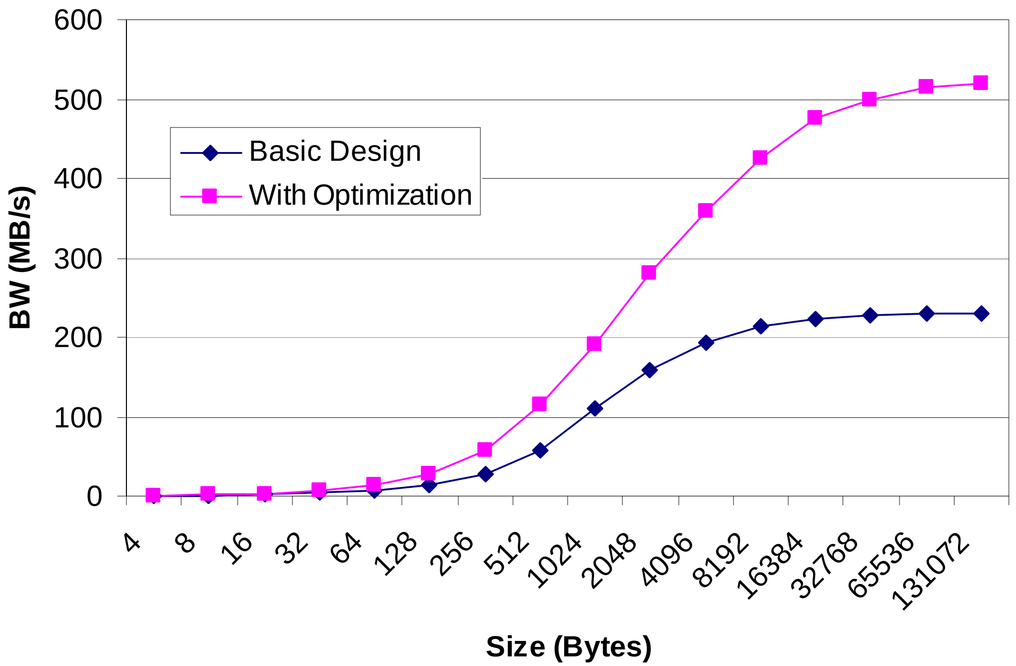
- 8 SuperMicro SUPER P4DL6 nodes (2.4 GHz Xeon, 400MHz FSB, 512K L2 cache)
- Mellanox InfiniHost MT23108 4X HCAs (A1 silicon), PCI-X 64bit 133MHz
- Mellanox InfiniScale MT43132 switch

Latency of Basic Design with Optimization



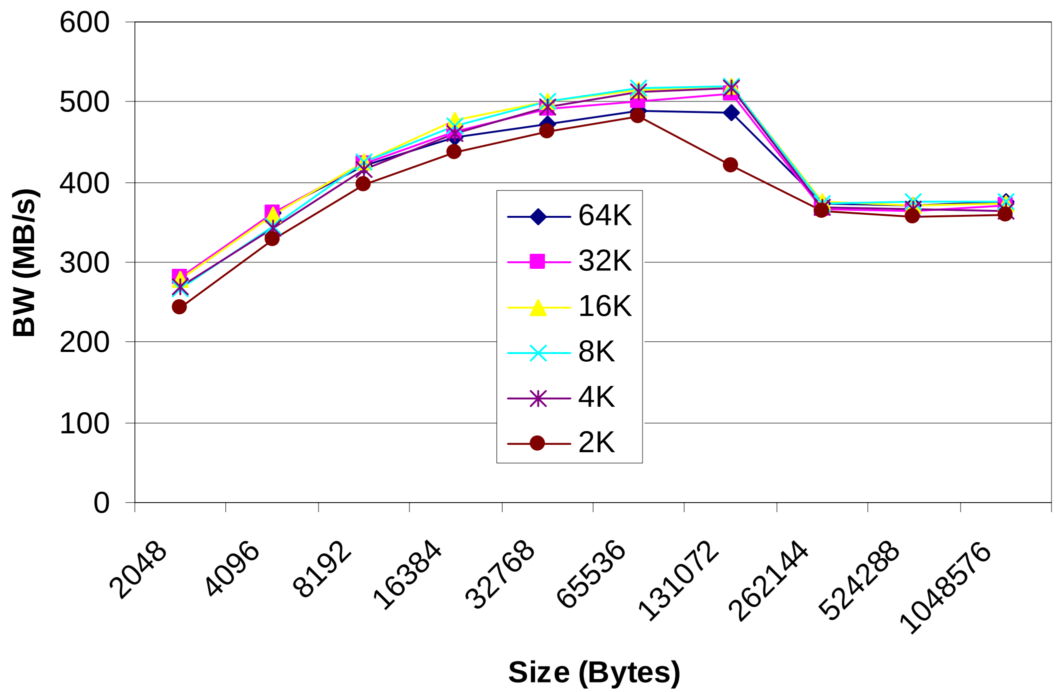
- Latency for Basic Design: 18.6 us
- With optimization: 7.4 us

Bandwidth of Basic Design with Optimization



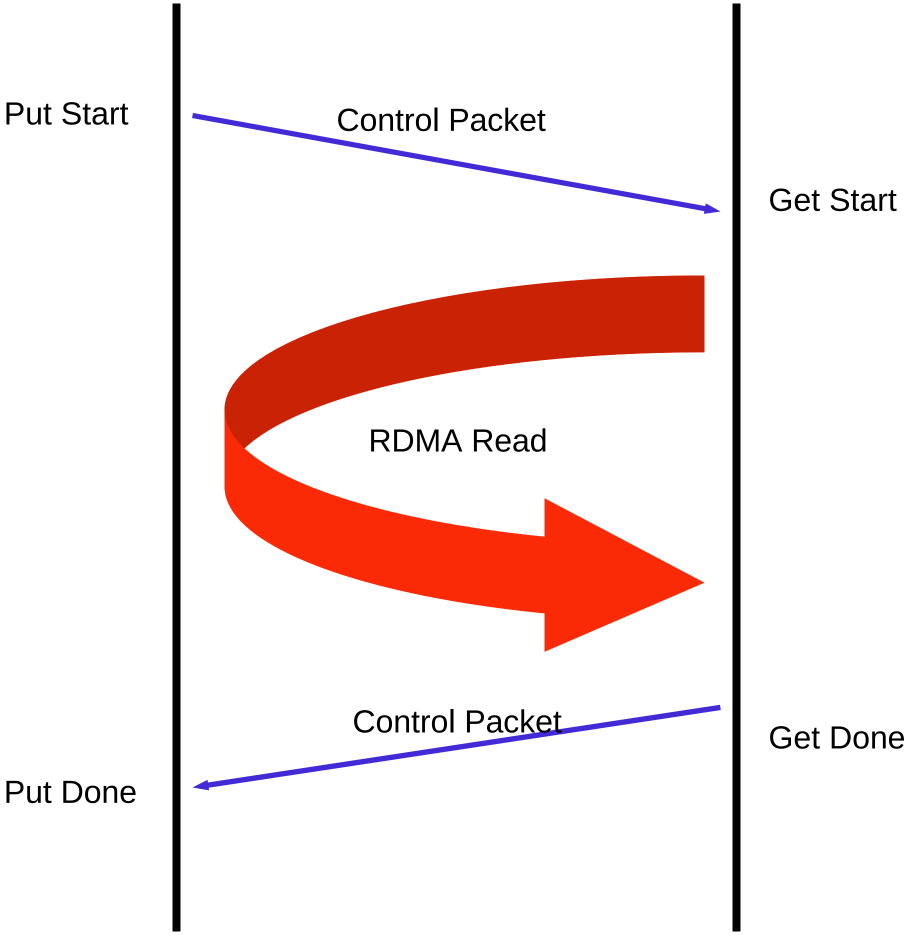
- Bandwidth for Basic Design: 230 MB/s
- With optimization: 520 MB/s

Impact of Pipelining Chunk Size on Bandwidth



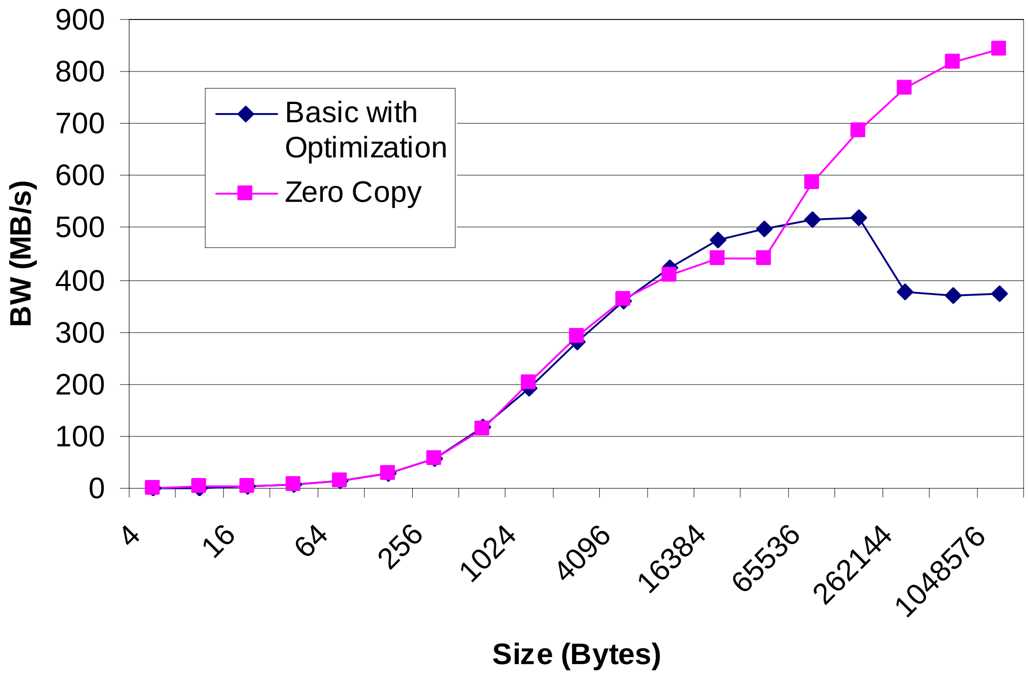
- Chunk sizes around 16K give best performance

Zero Copy Design



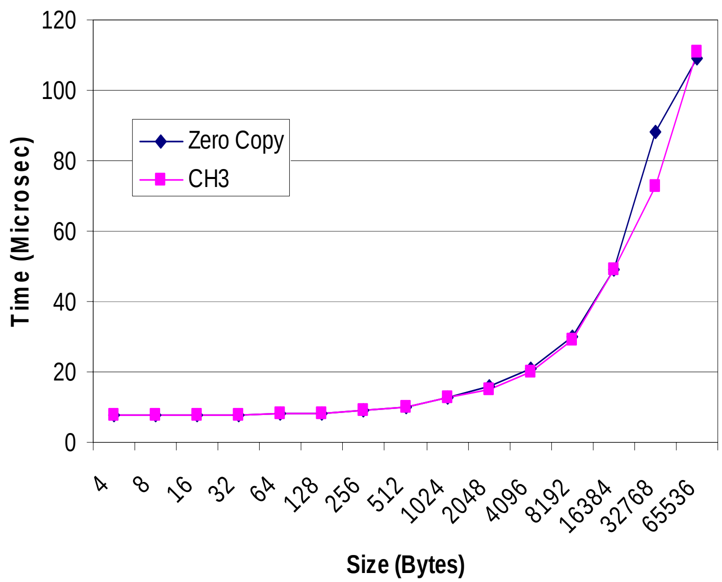
- Small messages are handling similar to basic design with optimization
- Large messages
  - Shared buffer pool used only for control messages
  - Data transfer using RDMA Read
  - No extra copies

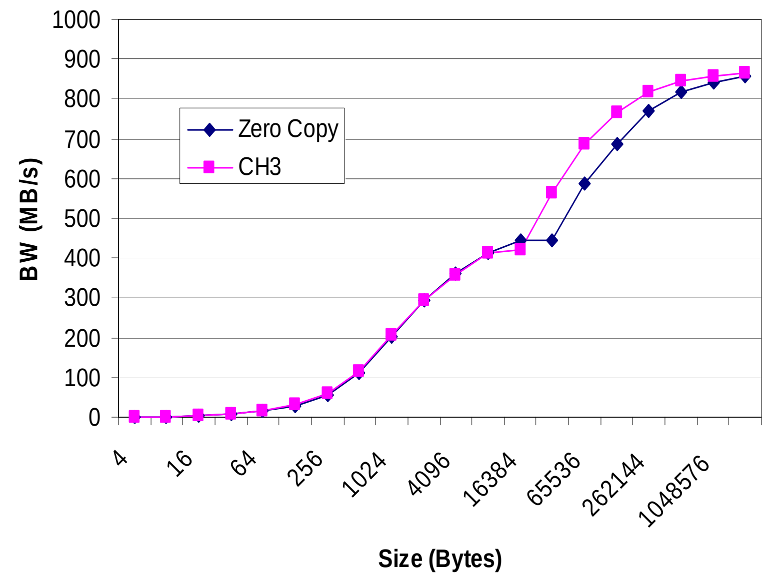
Bandwidth of Zero Copy Design



- Bandwidth for Basic Design with optimization: 520 MB/s
- Zero Copy: 857 MB/s

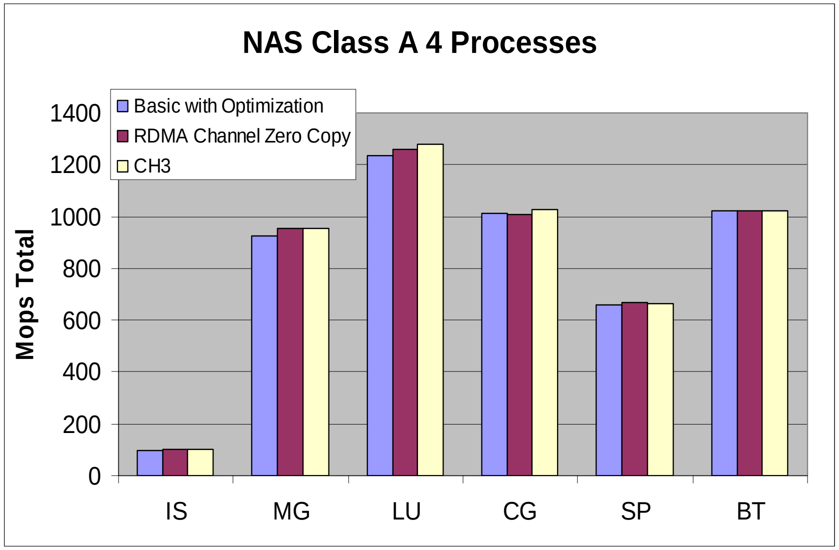
Comparing RDMA Channel with CH3 (Latency and Bandwidth)

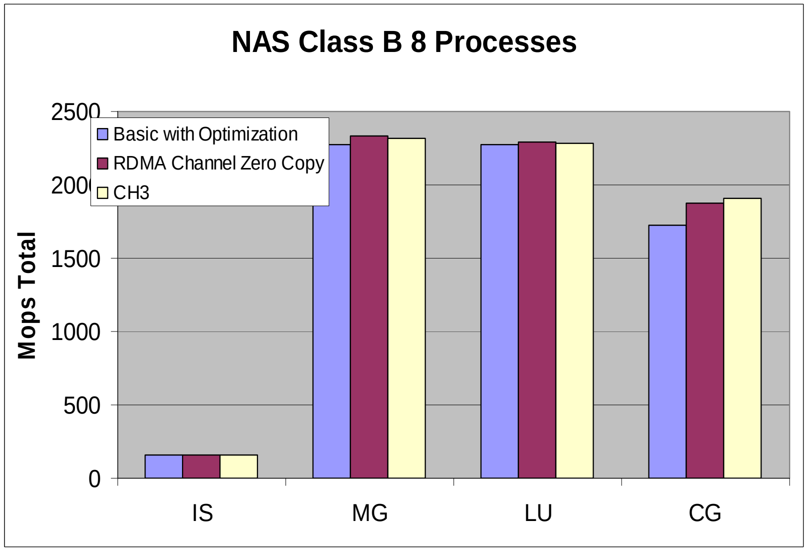




- Comparing with another implementation done at CH3 level
  - Also does zero copy for large messages using RDMA Write
- RDMA Channel design with Zero Copy does very close to the CH3 level design
- Difference in bandwidth is due to the performance difference of RDMA write and RDMA read in InfiniBand

Comparing RDMA Channel with CH3 (NAS Benchmarks)





- RDMA Channel zero copy design perform comparably to CH3 design

Conclusions

- We presented a study of using RDMA to implement MPICH2 over InfiniBand
- We focus on RDMA Channel Interface in MPICH2
  - Design, optimization and evaluation
- We show that RDMA Channel provides a simple yet powerful interface
  - We achieved 7.6 microsec latency and 857 MB/s bandwidth

MVAPICH2 Software Release

- Based on MPICH2 RDMA Channel Interface
  - Zero copy design
- Open source software
- Currently used by many organizations

Web Pointers

NBC home page

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

MVAPICH2 home page

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