Efficient Shared Memory and RDMA based design for MPI_Allgather over InfiniBand

Amith R. Mamidala, Abhinav Vishnu
Dhabaleswar K. Panda

Department of Computer Science and Engineering
Ohio State University

{mamidala, vishnu, panda}@cse.ohio-state.edu

Presentation Outline

- Introduction
- MPI_Allgather
- Motivation & Problem Statement
- Design
- Performance Evaluation
- Conclusions & Future Work

Introduction

- Recent Advances in cluster computing
  - Sizes of cluster reaching tens of thousands of nodes
  - Multi-core Architecture enabling higher process density/node (upto 16 to 32)
- MPI : de-facto programming model for parallel applications
  - Many communication primitives
  - MPI_Allgather
    - Recursive-doubling algorithm
- InfiniBand
  - Widely being deployed to build large-scale clusters
  - Offers many features for efficient and scalable performance
    - RDMA, SRQ with Send/Recv (Reliable Connection)
    - Send/Rev (Unreliable Datagram)
- MVAPICH
  - Scalable and Efficient Point-to-Point protocols over IBA
  - Efficient Shared Memory Design for intra-node
  - Independent Channels of communication

Overview of MVAPICH and MVAPICH2 Projects (OSU MPI for InfiniBand)

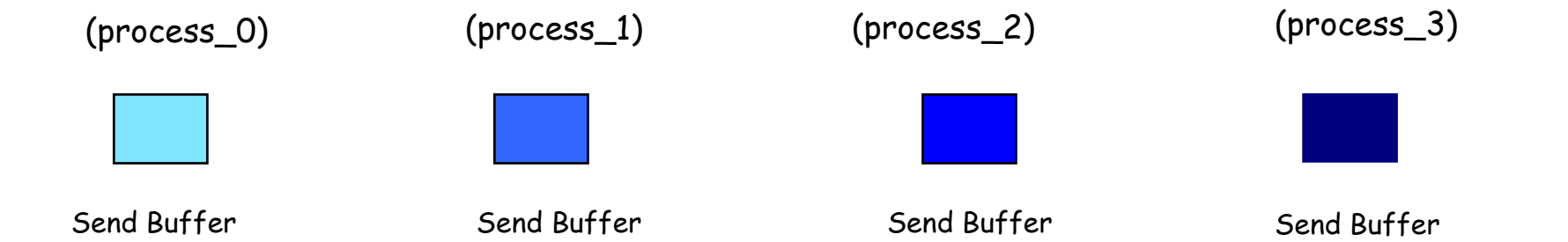
- Focusing on
  - MPI-1 (MVAPICH)
  - MPI-2 (MVAPICH2)
- Open Source (BSD licensing) with anonymous SVN access
- Directly downloaded and being used by more than 405 organizations worldwide (in 30 countries)
- Available in the software stack of
  - Many IBA and server vendors
  - OFED (OpenFabrics Enterprise Distribution)
- Empowers multiple InfiniBand clusters in the TOP 500 list including the 9K+ processor Sandia ThunderBird Cluster (6th rank)
- URL: http://nowlab.cse.ohio-state.edu/projects/mpi-iba/

Presentation Outline

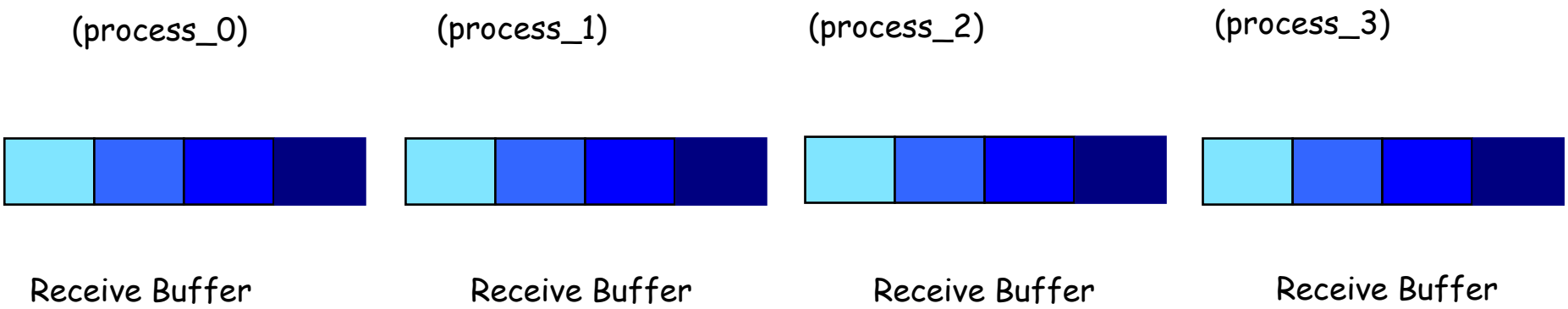
- Introduction
- MPI_Allgather
  - Brief summary of Recursive-Doubling
- Motivation & Problem Statement
- Design
- Performance Evaluation
- Conclusions & Future Work

MPI_Allgather

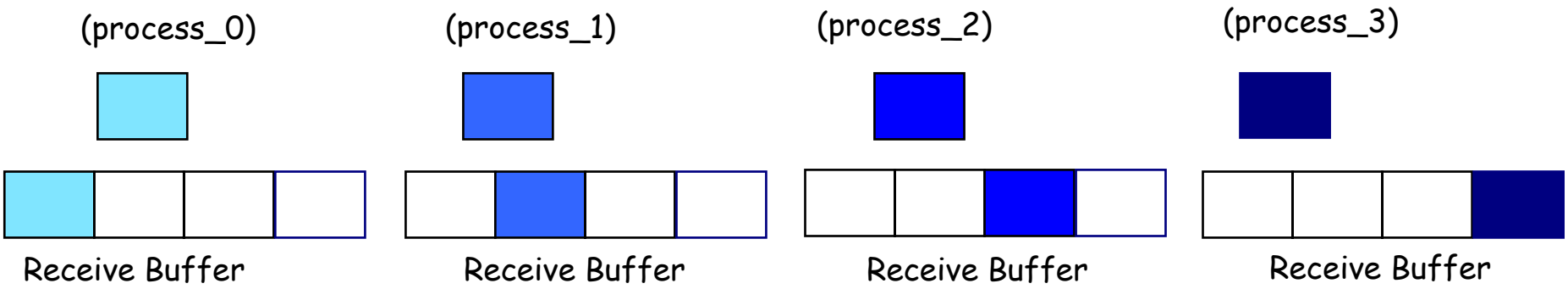
Before MPI_Allgather:



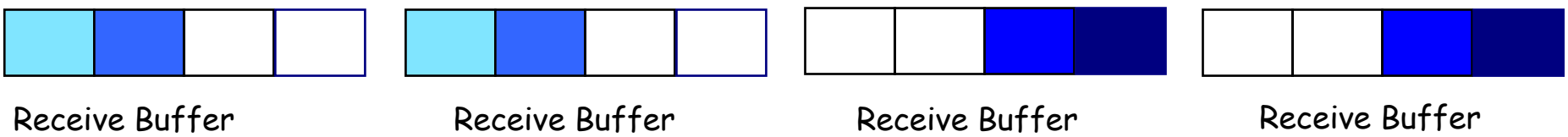
After MPI_Allgather:



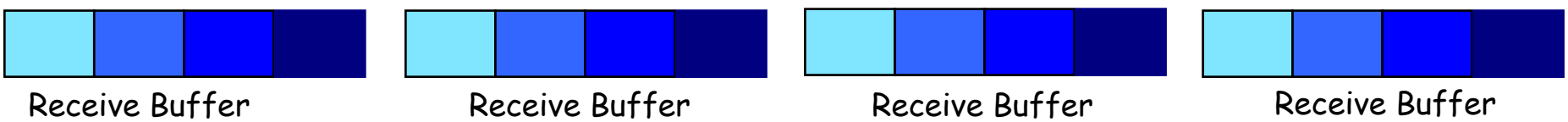
Recursive Doubling Algorithm



After step 1:

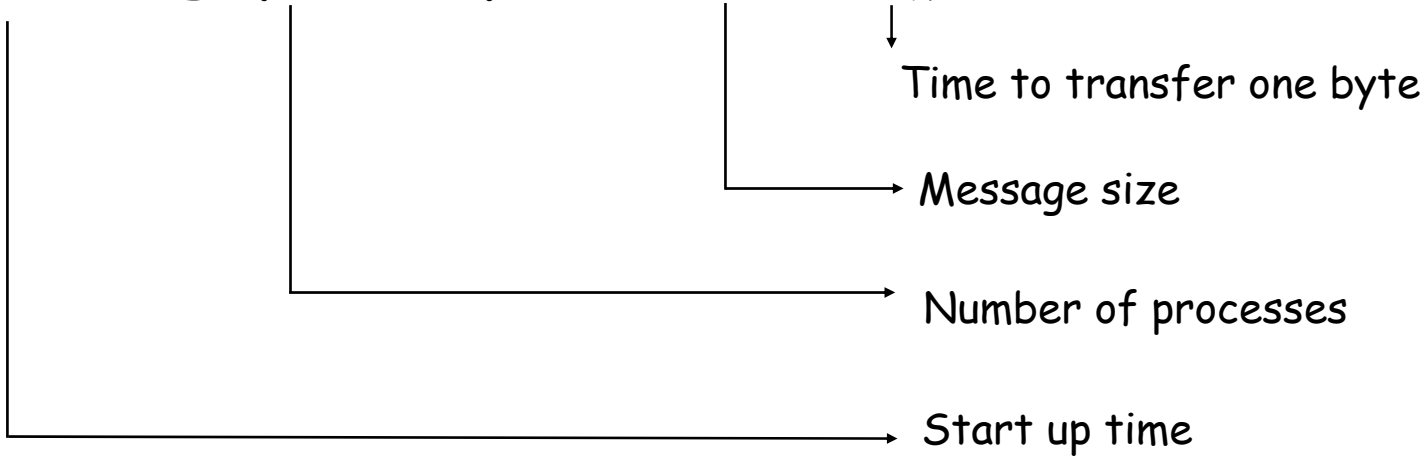


After step 2:



Recursive Doubling contd..

- $T_{rd} = t_s * \log(p) + (p-1) * m * t_w$

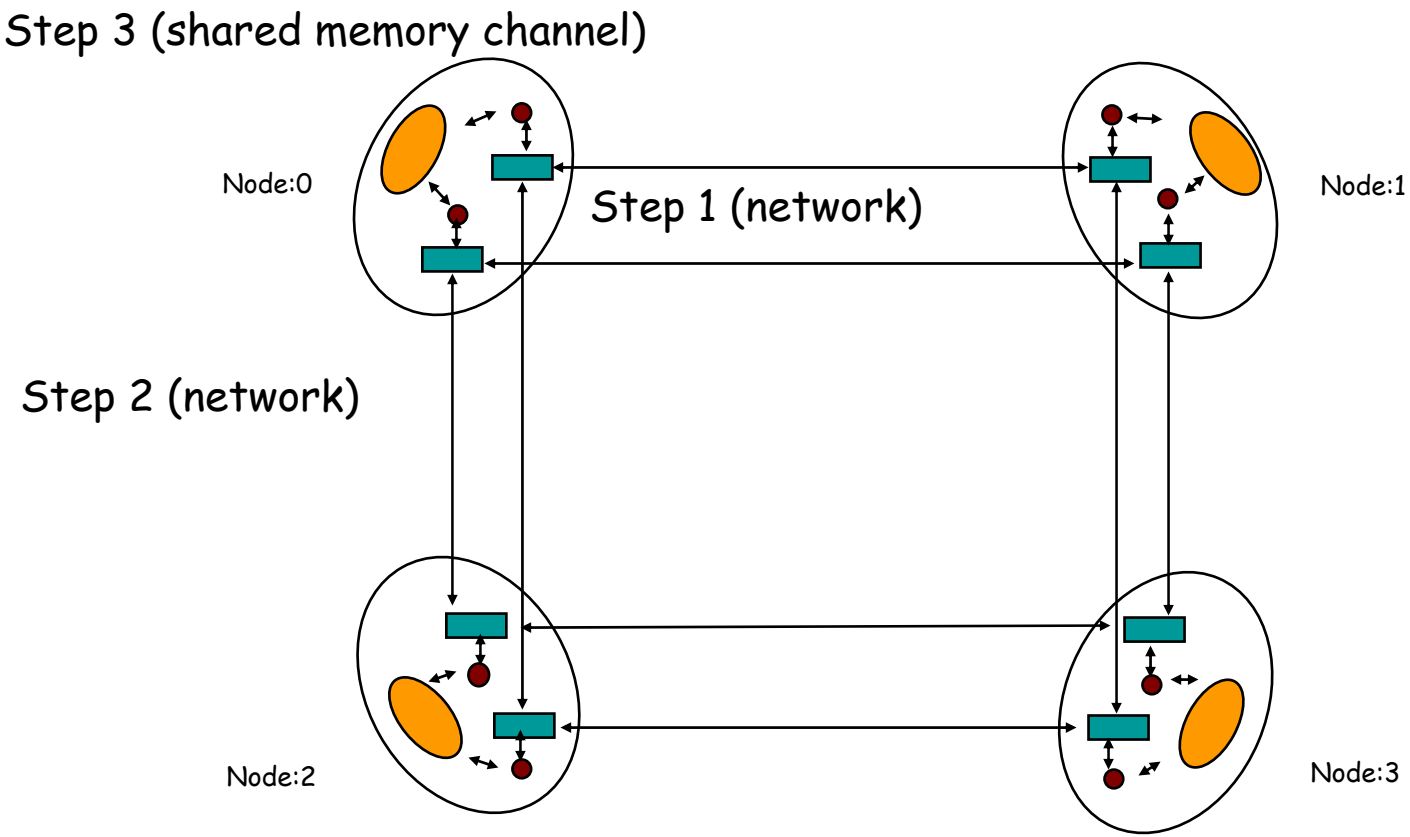


- Total message volume = (p-1)*m

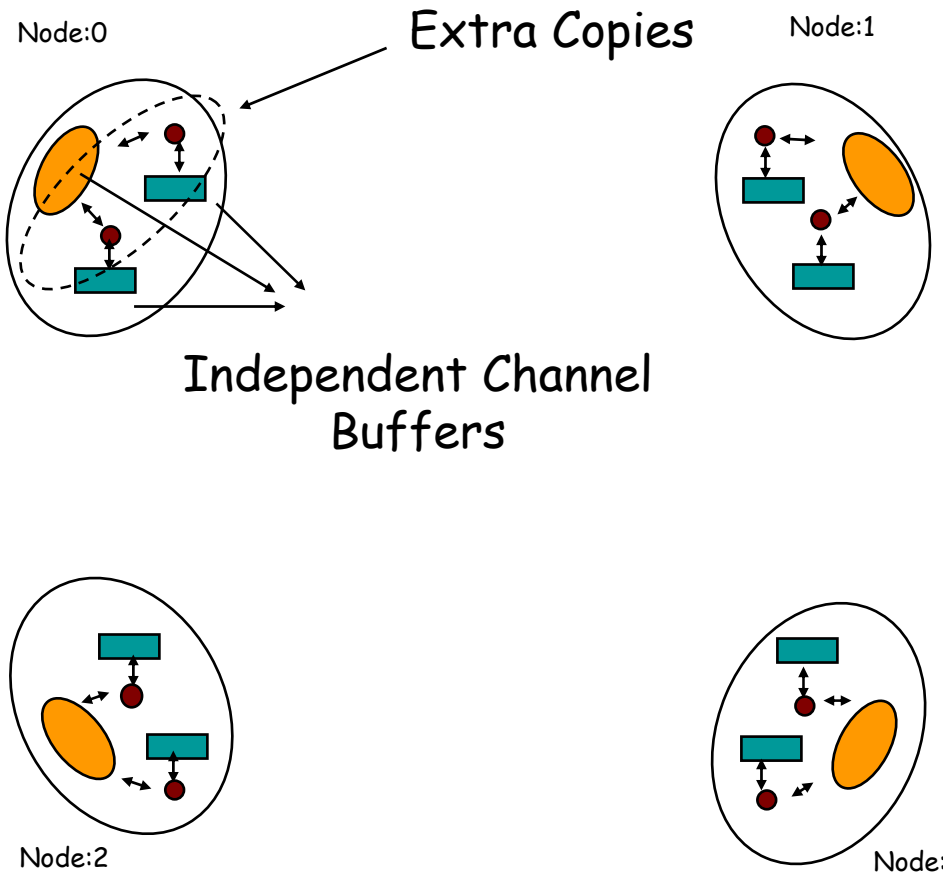
Presentation Outline

- Introduction
- MPI_Allgather
- Motivation & Problem Statement
- Design
- Performance Evaluation
- Conclusions & Future Work

Recursive Doubling with multiple process/node



Problems with the approach



Problems:

1. No Buffer Sharing
2. No Control over Scheduling
   - No overlap
   - Total volume over n/w (n-1)*(nm) [$(p-1)*m*t_w$]
   - Same volume within a node

Problem Statement

- What mechanisms are required to eliminate extraneous copy costs?
- How can the scheduling of the operations be done for efficient overlap of intra- and inter-node communication?

Presentation Outline

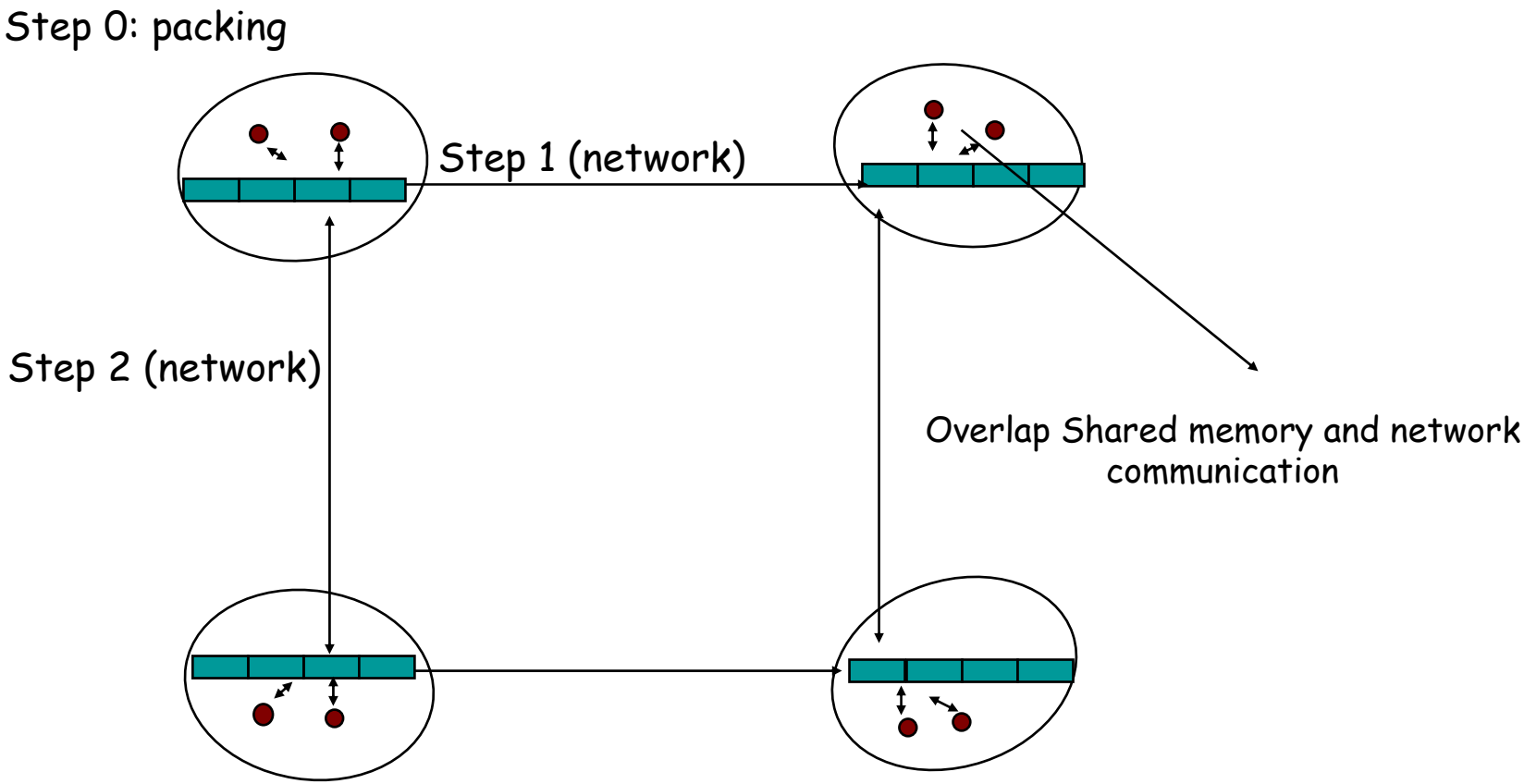
- Introduction
- MPI_Allgather
- Motivation & Problem Statement
- Design
- Performance Evaluation
- Conclusions & Future Work

Shared Memory & RDMA based MPI_Allgather

- Require Shared Communication Buffers
- Why RDMA and not Send/Recv?
  - No preposting buffers required
  - Easily pluggable
  - Simpler design
- Existing RDMA-based design for MPI_Allgather exists
  - Applicable to one process per node
  - Extend to multiple processes per node

  ("High Performance All-to-All Broadcast for InfiniBand Clusters", S.Sur, U.K.R. Bondhugula, A.R. Mamidala, H.-W. Jin, D.K. Panda in HiPC 2005)

Illustration



Shared Memory & RDMA design

- Memory shared using mmap of shared file
- Data of each process contains a small header containing the rank of the process
- Addresses exchanged prior to the operation during creation of communicator
- RDMA write completion by separate shared flags
- Local rank zero acts as the "leader"
- Synchronization done through another set of shared flags

Presentation Outline

- Introduction
- MPI_Allgather
- Motivation & Problem Statement
- Design
- Performance Evaluation
  - Evaluation Testbed
  - Results
- Conclusions & Future Work

Evaluation Testbed

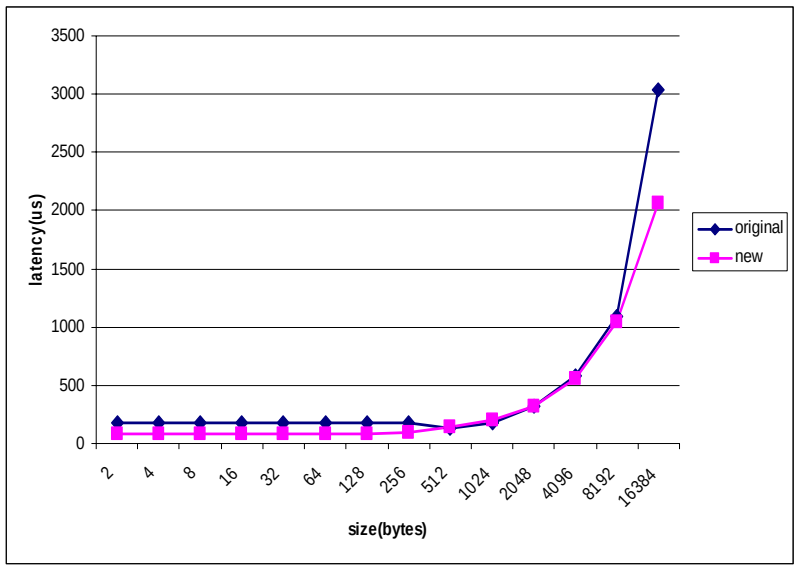
- Cluster A:
  - Dual Opteron 2.4 GHz
  - 1024 KB L2 cache
  - MT25204 Mellanox IBA HCAs
- Cluster B:
  - Dual Intel Xeon 2.66 GHz processors
  - PCI-X Interface
  - MT23108 Mellanox HCAs

MPI_Allgather Latency

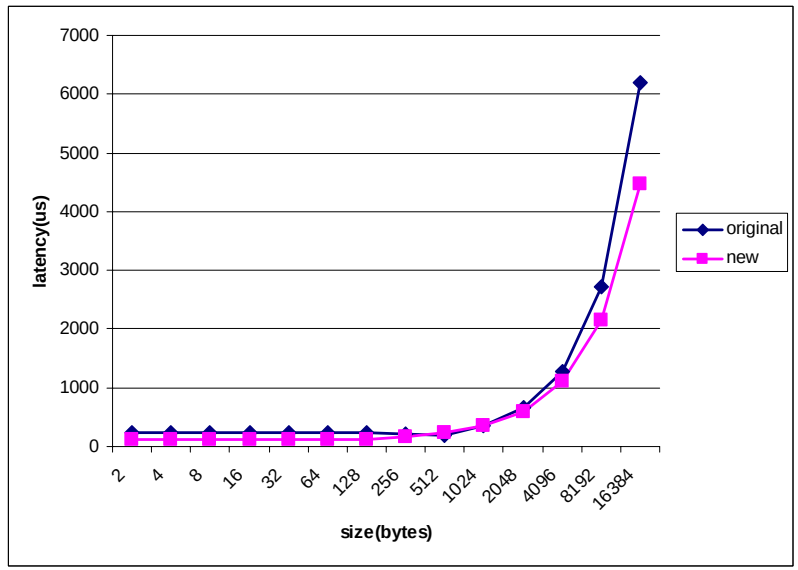
- Comparison with three schemes:
- Two sets of graphs:
  - Set 1:
    - Original: based on separate point-to-point and shared memory channels
    - New: design proposed with overlapping
  - Set 2:
    - Overlap: design proposed with overlapping
    - No-Overlap: design proposed with no overlapping

Cluster A

Allgather-16x2



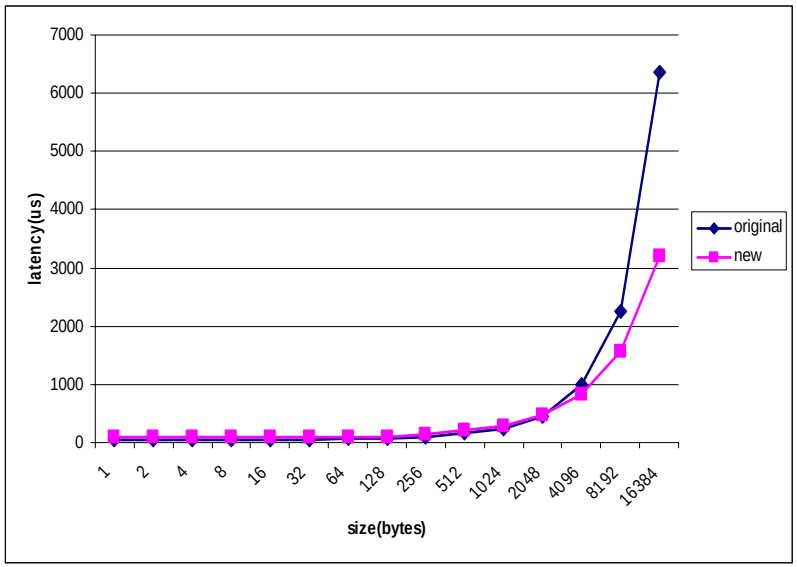
Allgather-32x2



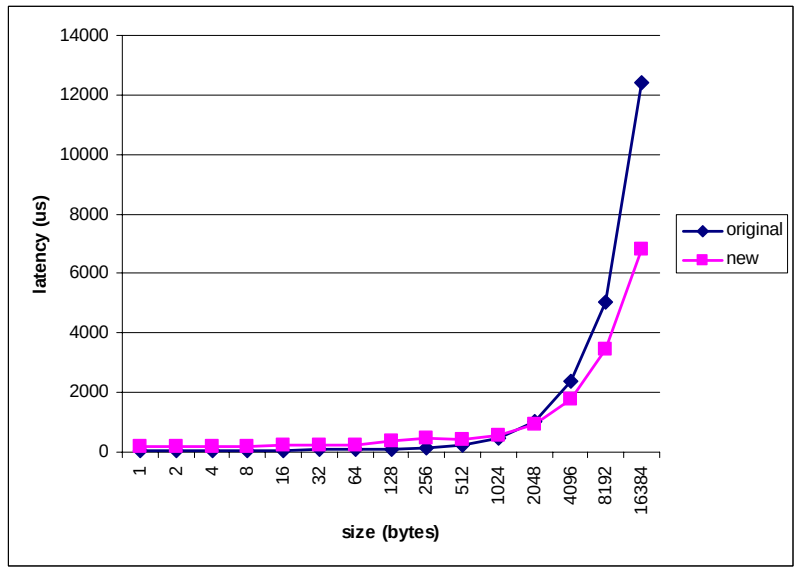
- Improvement upto 1.47 and 1.39 for 32, 64 processes respectively

Cluster B

Allgather-16x2



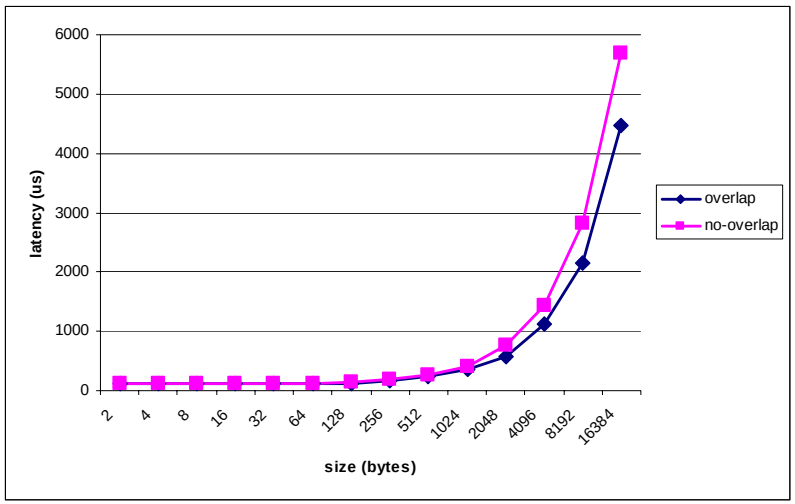
Allgather-32x2



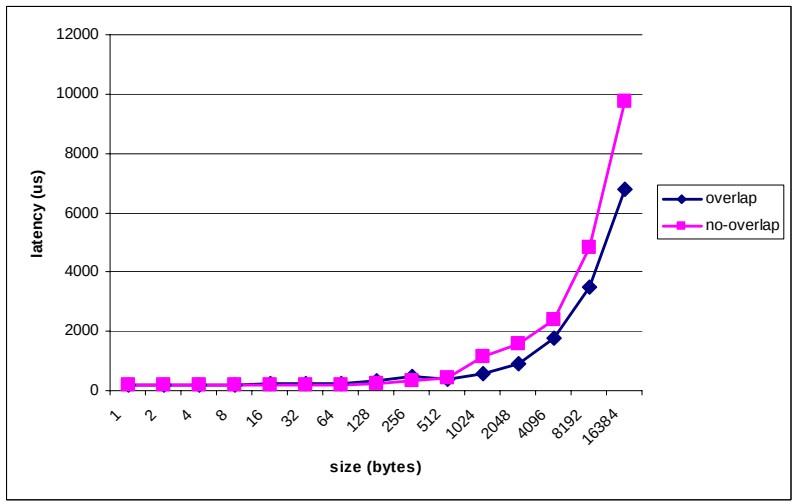
- Performance gains upto 1.97 and 1.82 for 32 and 64 processes respectively

Overlap Benefits

Allgather-32x2(Cluster A)



Allgather-32x2(Cluster B)



- Overlap benefits upto 30% and 43% for Cluster A and Cluster B respectively

Presentation Outline

- Introduction
- MPI_Allgather
- Motivation & Problem Statement
- Design
- Performance Evaluation
- Conclusions & Future Work

Conclusions & Future Work

- Implemented new RDMA and Shared Memory based design
- Common buffer for intra- and inter-node communication – pinning shared memory region
- Showed the benefits of reduced copies and overlap
- Incorporated into MVAPICH
- Apply to other MPI_Allgather algorithms especially for odd-number of processes
- Application Level Study
- Running on quad/higher core systems

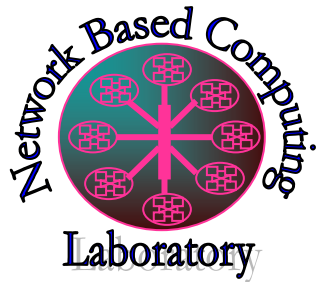
Acknowledgements

Our research is supported by the following organizations

- Current Funding support by

- Current Equipment support by


Web Pointers



http://www.cse.ohio-state.edu/~panda/

http://nowlab.cse.ohio-state.edu/

MVAPICH Web Page

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

THANK YOU!

&

Questions ?