NETWORK-BASED COMPUTING LABORATORY

Designing Optimized MPI Broadcast and Allreduce for Many Integrated Core (MIC) InfiniBand Clusters

K. Kandalla, A. Venkatesh, K. Hamidouche, S. Potluri,
D. Bureddy and D. K. Panda

Presented by Dr. Xiaoyi Lu

Computer Science & Engineering Department,
The Ohio State University

OHIO STATE

Outline

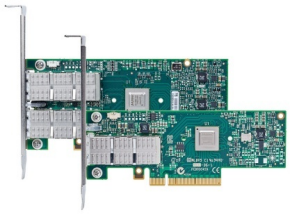
- Introduction
- Motivation and Problem Statement
- Designing Optimized Collectives for MIC Clusters:
  - MPI_Bcast
  - MPI_Reduce, MPI_Allreduce
- Experimental Evaluation
- Conclusions and Future work

Introduction

- HPC applications can now scale beyond 400,000 cores (TACC Stampede)
- Message Passing Interface (MPI) has been the de-facto programming model for parallel applications
- MPI defines collective operations to implement group communication operations
- Parallel applications commonly rely on collective operations owing to their ease-of-use and performance portability
- Critical to design collective operations to improve performance and scalability of HPC applications

Drivers of Modern HPC Cluster Architectures

Multi-core Processors



High Performance Interconnects - InfiniBand
<1usec latency, >100Gbps Bandwidth

Accelerators / Coprocessors
high compute density, high performance/watt
>1 TFlop DP on a chip

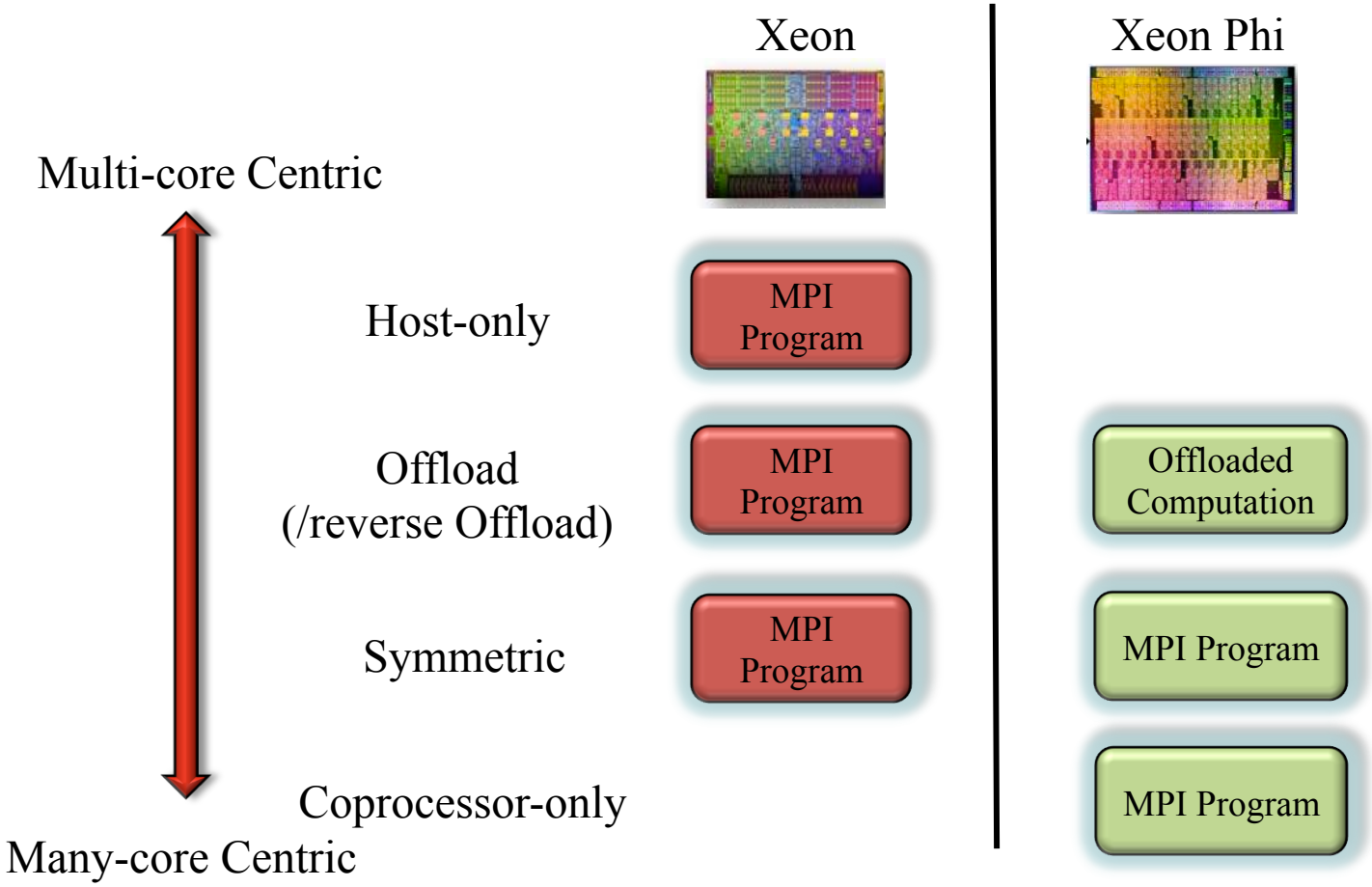
- Multi-core processors are ubiquitous
- InfiniBand very popular in HPC clusters
- Accelerators/Coprocessors becoming common in high-end systems
- Emerging HPC hardware is leading to a generation of heterogeneous systems
- Necessary to optimize MPI communication libraries on emerging heterogeneous systems

Intel Many Integrated Core (MIC) Architecture

- X86 compatibility – reuse the strong Xeon software eco-system
  - Programming models, libraries, tools and applications
- Xeon Phi, first product line based on MIC
  - Already powering several Top500 systems
  - Tianhe-2@NUDT/China (1), Stampede@TACC (6), Conte@Purdue(28), Discover@NASA(65)

MPI Applications on MIC Clusters

- Flexibility in launching MPI jobs on clusters with Xeon Phi



MVAPICH2/MVAPICH2-X Software

- MPI(+X) continues to be the predominant programming model in HPC
- High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP, and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1) ,MVAPICH2 (MPI-2.2 and MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - Used by more than 2,055 organizations (HPC Centers, Industry and Universities) in 70 countries
  - More than 181,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 6th ranked 462,462-core cluster (Stampede) at TACC
    - 19th ranked 125,980-core cluster (Pleiades) at NASA
    - 21st ranked 73,278-core cluster (Tsubame 2.0) at Tokyo Institute of Technology and many others
  - Available with software stacks of many IB, HSE, and server vendors including Linux Distros (RedHat and SuSE)
  - http://mvapich.cse.ohio-state.edu
- Partner in the U.S. NSF-TACC Stampede System

Outline

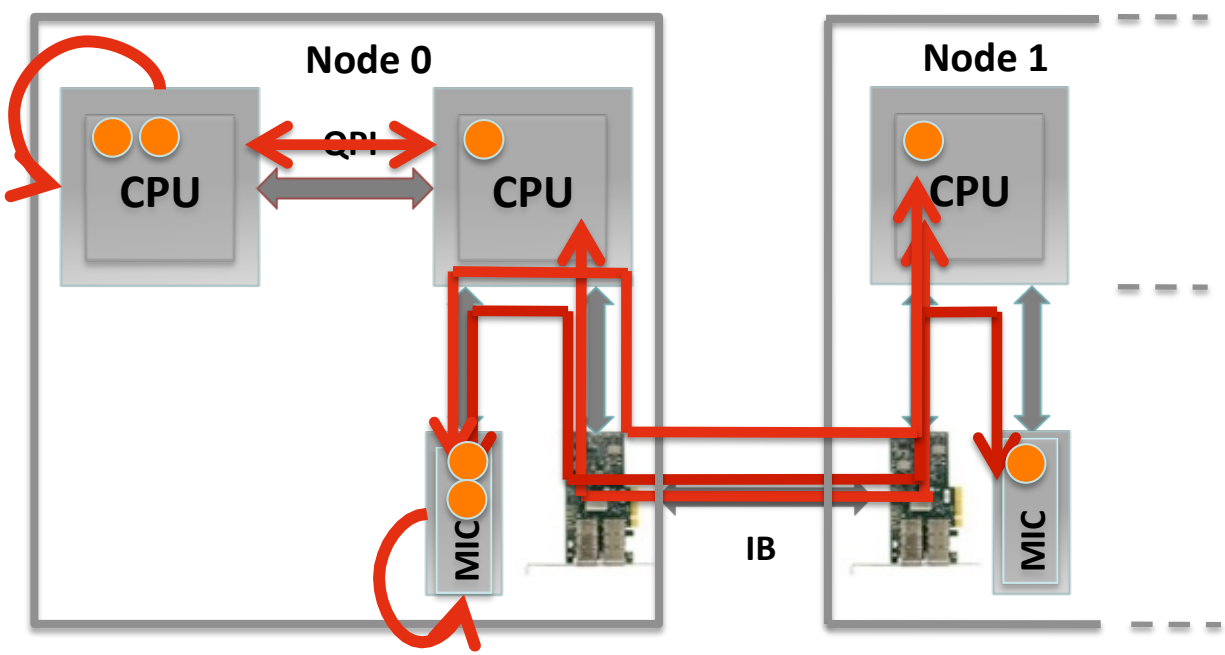
- Introduction
- Motivation and Problem Statement
- Designing Optimized Collectives for MIC Clusters:
  - MPI_Bcast
  - MPI_Reduce, MPI_Allreduce
- Experimental Evaluation
- Conclusions and Future work

Motivation

- MIC architectures are leading to heterogeneous systems
- MIC coprocessors have slower clock rates, smaller physical memory
- Heterogeneous MIC clusters introduce many new communication planes, with varying performance trade-offs
- Critical to design communication libraries to carefully optimize communication primitives on heterogeneous systems

Data Movement on Intel Xeon Phi Clusters

- Connected as PCIe devices – Flexibility, however adds complexity



MPI Process

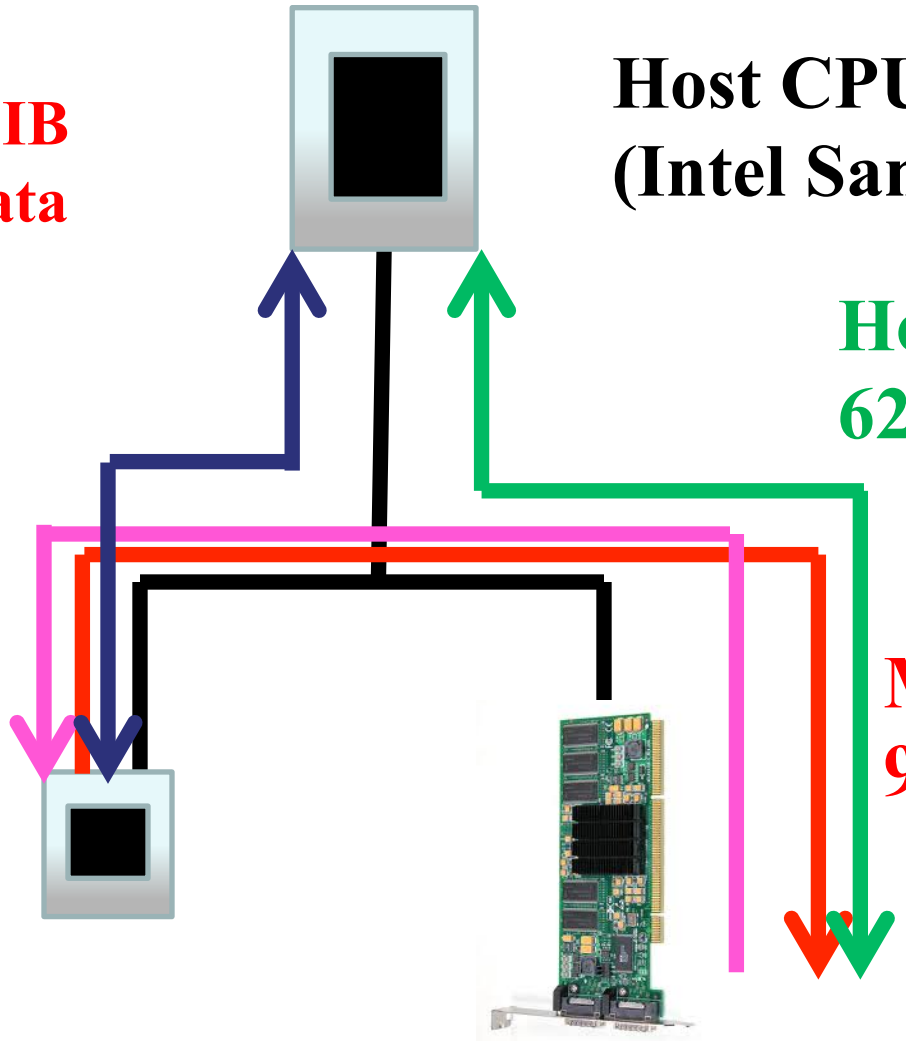
1. Intra-Socket
2. Inter-Socket
3. Inter-Node
4. Intra-MIC
5. Inter-Node MIC-MIC
6. Inter-Node MIC-Host

and more . . .

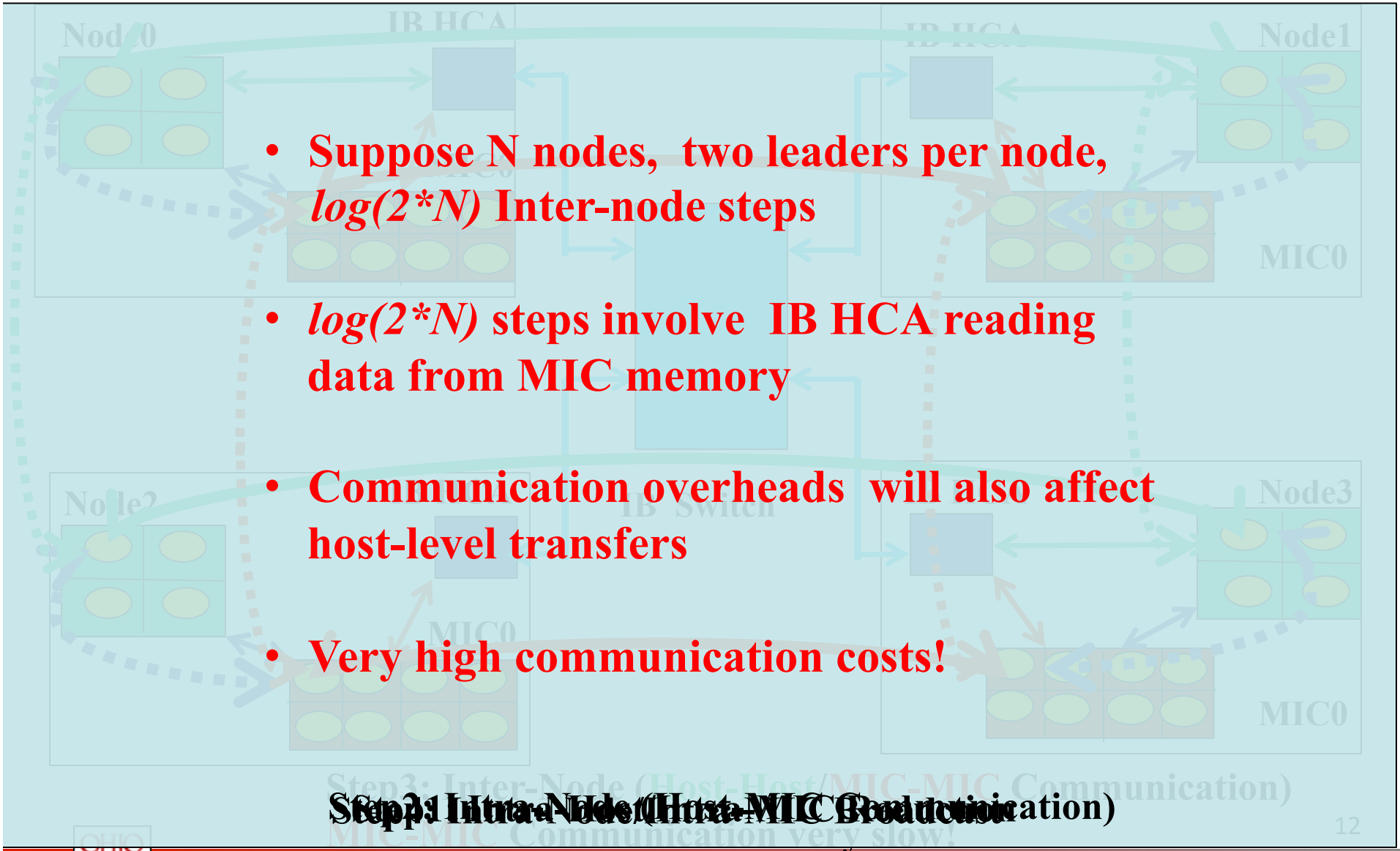
- Critical for runtimes to optimize data movement, hiding the complexity

Communication Paths and Peak Bandwidths (MB/s)

Critical to avoid communication paths involving IB HCA reading data from MIC Memory

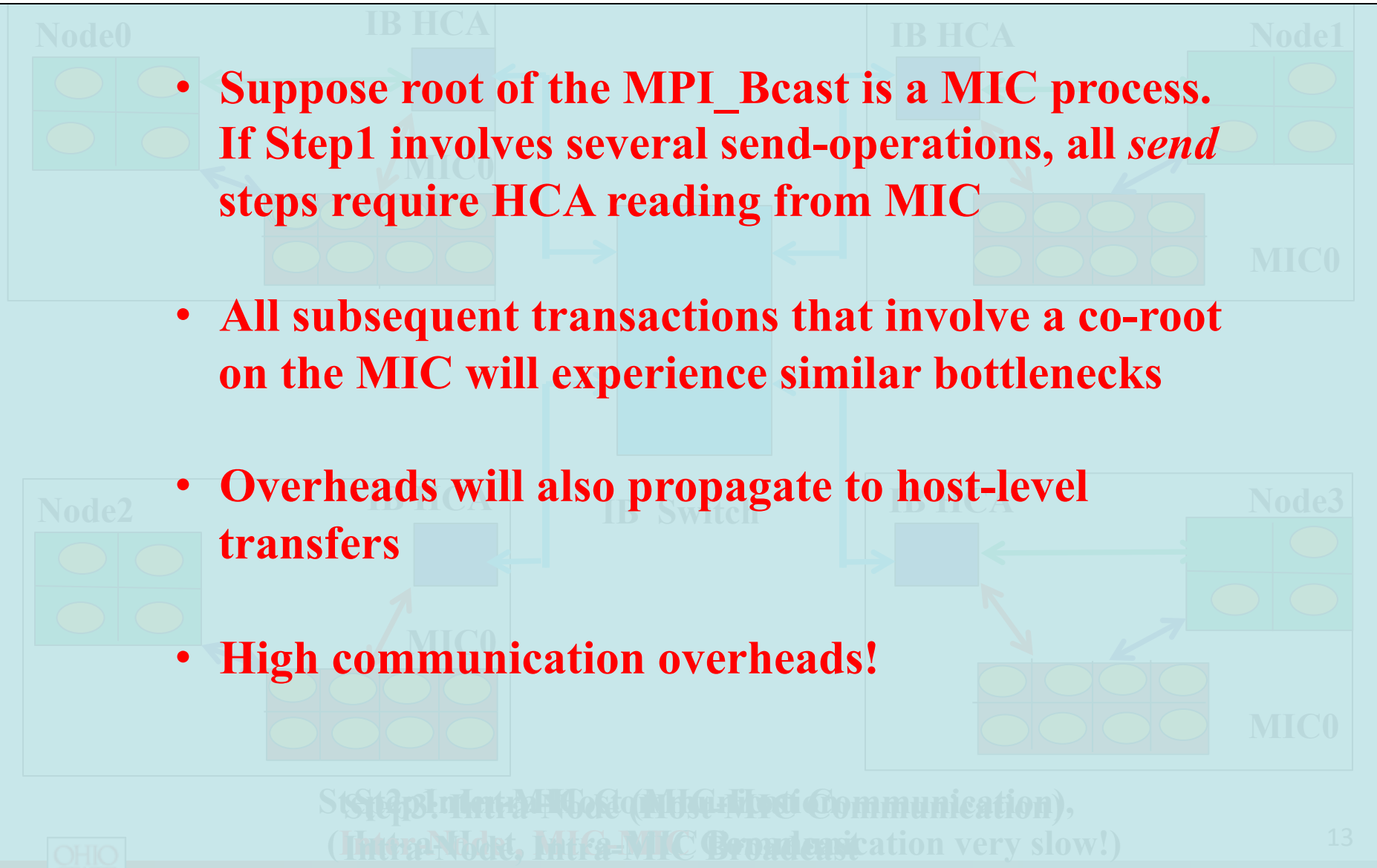


Basic Recursive-Doubling (Allreduce)



- **Suppose N nodes, two leaders per node, *log(2*N)* Inter-node steps**
- ***log(2*N)* steps involve IB HCA reading data from MIC memory**
- **Communication overheads will also affect host-level transfers**
- **Very high communication costs!**

Basic Knomial Tree (Bcast)



- **Suppose root of the MPI_Bcast is a MIC process. If Step1 involves several send-operations, all *send* steps require HCA reading from MIC**
- **All subsequent transactions that involve a co-root on the MIC will experience similar bottlenecks**
- **Overheads will also propagate to host-level transfers**
- **High communication overheads!**

Problem Statement

- Can we design a generic framework to optimize collective operations on emerging heterogeneous clusters?
- How can we improve the performance of common collectives, such as, MPI_Bcast, MPI_Reduce and MPI_Allreduce on MIC clusters?
- What are the challenges in improving the performance of intra-MIC phases of collective operations?
- MIC is a highly threaded environment, can we utilize OpenMP threads to accelerate collectives such as MPI_Reduce and MPI_Allreduce?

Outline

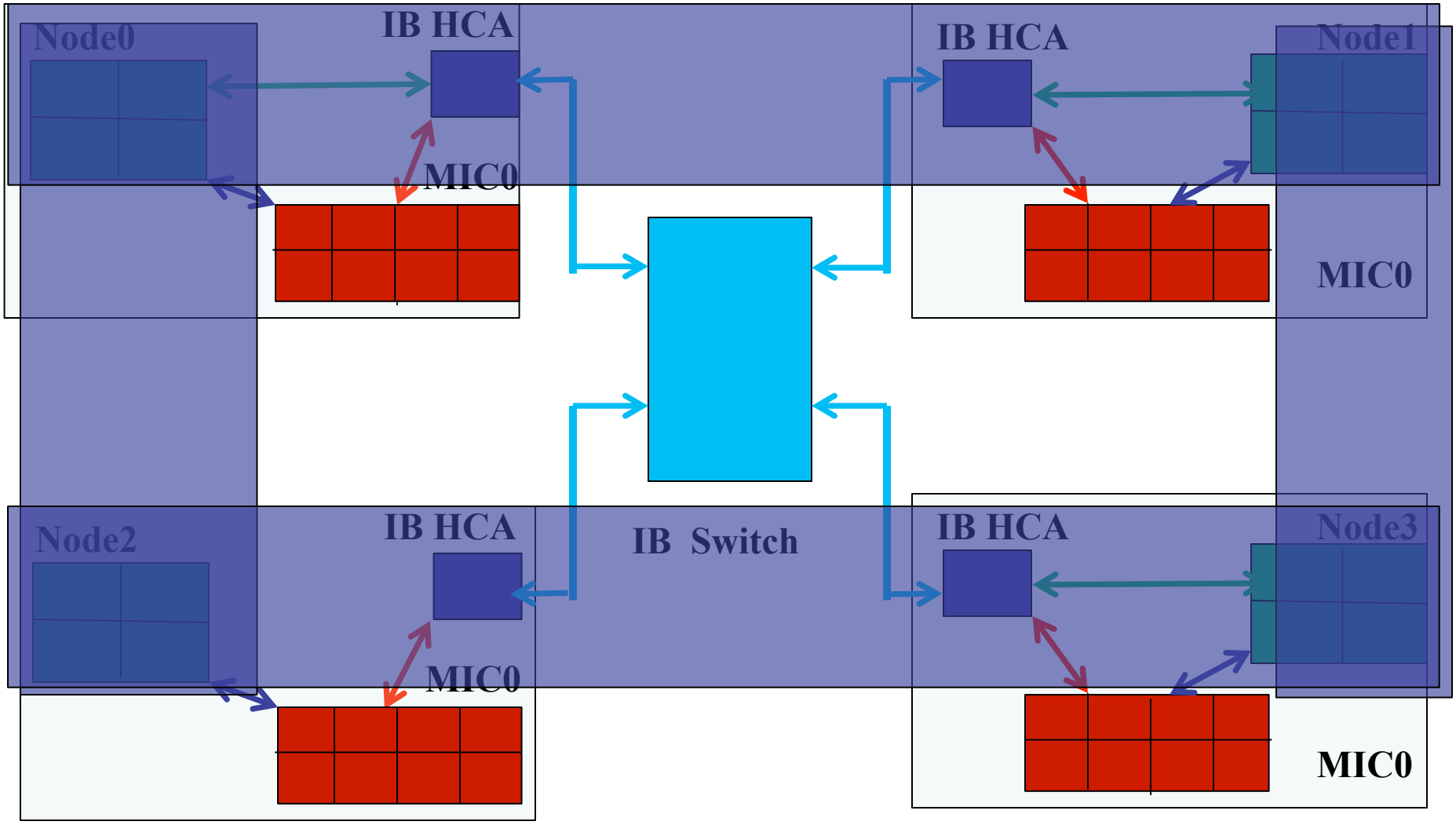
- Introduction
- Motivation and Problem Statement
- Designing Optimized Collectives for MIC Clusters:
  - MPI_Bcast
  - MPI_Reduce, MPI_Allreduce
- Experimental Evaluation
- Conclusions and Future work

Design Objectives

- Primary objectives of the proposed framework:
  - **Minimize involvement of MIC cores while progressing collectives**
  - **Eliminate utilization of high overhead communication paths**
  - **Improve the performance of intra-MIC phases of collectives**
- Proposed approach:

  Split heterogeneous communicator into two new sub-groups.
  - **Host-Comm (HC):** comprising of host processes
  - **MIC-Leader-Comm (MLC):** includes only MIC leader processes

Proposed Hierarchical Communicator Sub-System



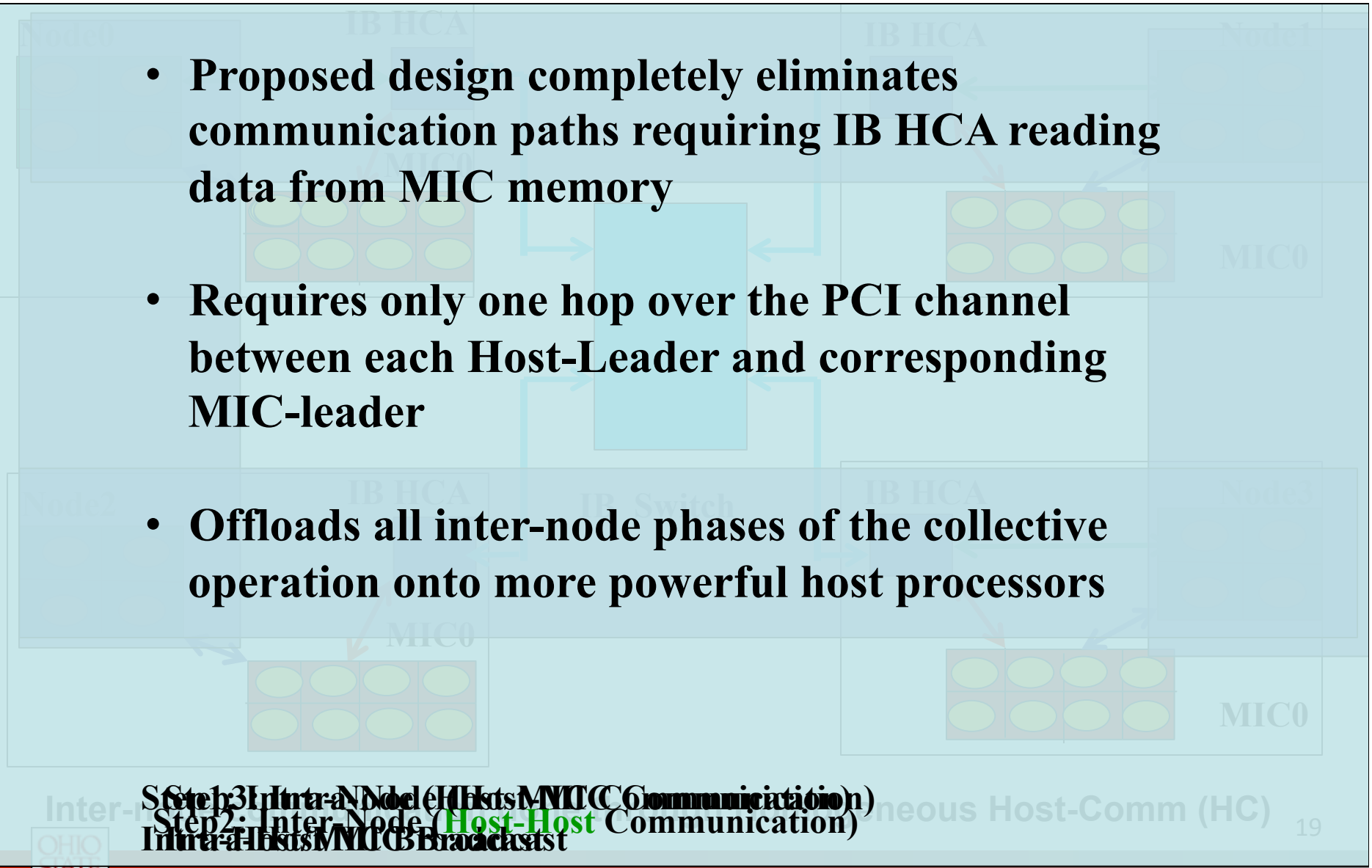
Host-Comm (HC), not including any MIC processes (Homogeneous)

Outline

- Introduction
- Motivation and Problem Statement
- Designing Optimized Collectives for MIC Clusters:
  - MPI_Bcast
  - MPI_Reduce, MPI_Allreduce
- Experimental Evaluation
- Conclusions and Future work

Proposed Knomial Tree (Bcast)

- **Proposed design completely eliminates communication paths requiring IB HCA reading data from MIC memory**
- **Requires only one hop over the PCI channel between each Host-Leader and corresponding MIC-leader**
- **Offloads all inter-node phases of the collective operation onto more powerful host processors**



Outline

- Introduction
- Motivation and Problem Statement
- Designing Optimized Collectives for MIC Clusters:
  - MPI_Bcast
  - MPI_Reduce, MPI_Allreduce
    - Inter-node Design Alternatives
    - Intra-node, Intra-MIC Design Alternatives
- Experimental Evaluation
- Conclusions and Future work

Inter-node MPI_Reduce, MPI_Allreduce Design Alternatives (Basic)

Consider Allreduce with N nodes:

- **Eliminates instances of IB HCA reading from MIC memory**
- **Each MIC leader performs maximum of two transactions over the PCI**
- **Critical communication steps performed by the host**

- **First set of intra-MIC Allreduce is still performed by the slower MIC cores**
- **The inter-Host phase cannot begin until the MIC leader is done with Intra-MIC reduce**

Step1: Intra-Node Reduction (Host-Host)
Step2: Inter-Node Reduction (Host-Host)
Step3: Inter-Node Broadcast (Host-MIC Communication)

Inter-node MPI_Reduce, MPI_Allreduce Design Alternatives (Direct)

Consider Allreduce with N nodes:

- **Eliminates instances of IB HCA reading from MIC memory**
- **Critical communication steps performed by the host**
- **Intra-MIC computation is also offloaded to the host**

- **Large number of transfers between MIC processes and the Host-Leader on each node**
- **Heavier load on the Host-Leader processes**

Step1a: Intra-Host Reduction, Host-Leader-MIC communication

Step2: Inter-Host Allreduction

Step3: Intra-Host/Intra-MIC Broadcast

Inter-node MPI_Reduce, MPI_Allreduce Design Alternatives (Peer)

Consider Allreduce with N nodes:

- **Eliminates instances of IB HCA reading from MIC memory**
- **Each MIC process performs maximum of two transactions over the PCI**
- **Critical communication steps performed by the host**
- **Intra-MIC computation is also offloaded to the host**
- **Load on Host processes is balanced**

- **Large number of transfers between MIC processes and the corresponding Host Processes**

Step2: Inter-Host Reduction/Broadcast

Outline

- Introduction
- Motivation and Problem Statement
- Designing Optimized Collectives for MIC Clusters:
  - MPI_Bcast
  - MPI_Reduce, MPI_Allreduce
    - Inter-node Design Alternatives
    - Intra-node, Intra-MIC Design Alternatives
- Experimental Evaluation
- Conclusions and Future work

Intra-node MPI_Reduce (Basic Slab-Design)

Consider Allreduce with N processes in a node:

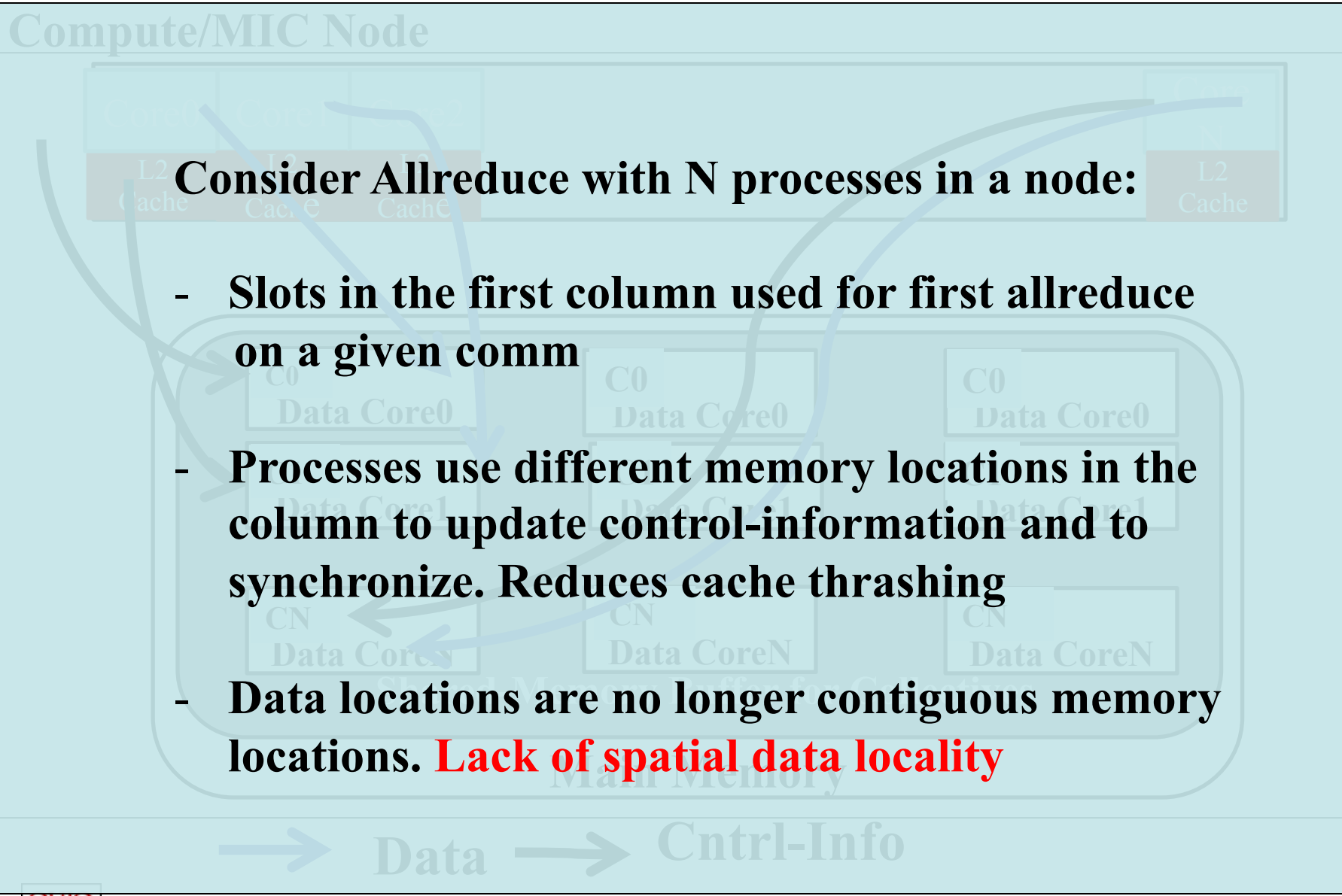
- **Data Slab0 and Control-Array0 used for Comm0**

- **Processes use contiguous memory locations in Data Slab0 to read/write data. Better spatial Locality**

- **Control-Array0 used for updating control-information and synchronization. Same cache line shared between N processes.**

Can lead to cache thrashing

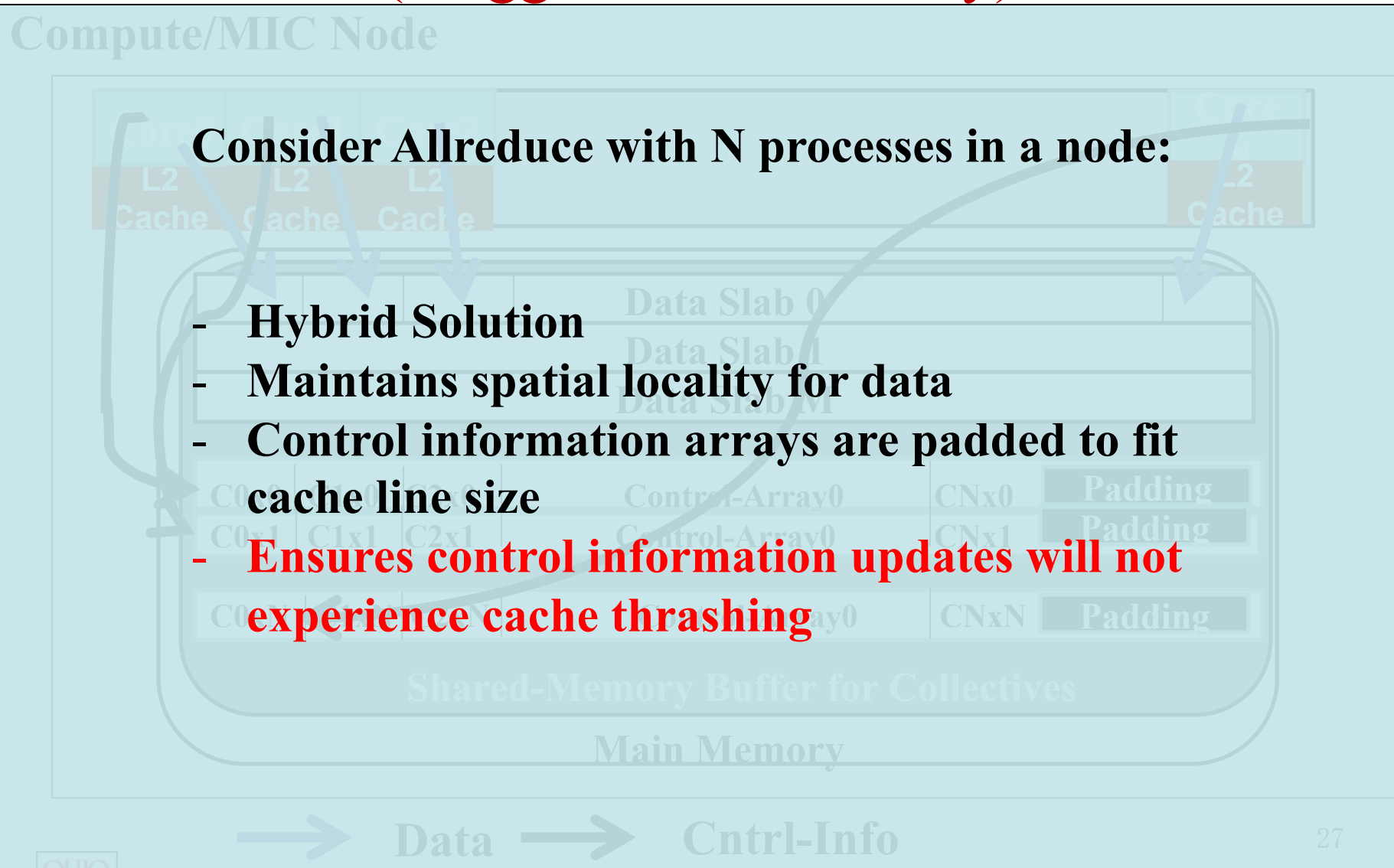
Intra-node MPI_Reduce (Slot-Based Design)



Consider Allreduce with N processes in a node:

- **Slots in the first column used for first allreduce on a given comm**
- **Processes use different memory locations in the column to update control-information and to synchronize. Reduces cache thrashing**
- **Data locations are no longer contiguous memory locations. Lack of spatial data locality**

Intra-node MPI_Reduce (Staggered-Cntrl-Array)



Consider Allreduce with N processes in a node:

- **Hybrid Solution**
- **Maintains spatial locality for data**
- **Control information arrays are padded to fit cache line size**
- **Ensures control information updates will not experience cache thrashing**

Intra-Node/Intra-MIC Related Design Considerations (MPI_Reduce/MPI_Allreduce)

- Issue:

  Designs relying on node leader (lowest rank on a node/mic) to perform the reductions via shared-memory can lead to load imbalance and skewed execution

- Proposed Solution:

  Construct tree-based designs for shared-memory based reductions to balance the compute load across all cores

  Trees with configurable degrees to customize across architectures

- Use OpenMP threads on MIC to accelerate compute operations of intra-node Reduce/Allreduce operations

Outline

- Introduction
- Motivation and Problem Statement
- Designing Optimized Collectives for MIC Clusters:
  - MPI_Bcast
  - MPI_Reduce, MPI_Allreduce

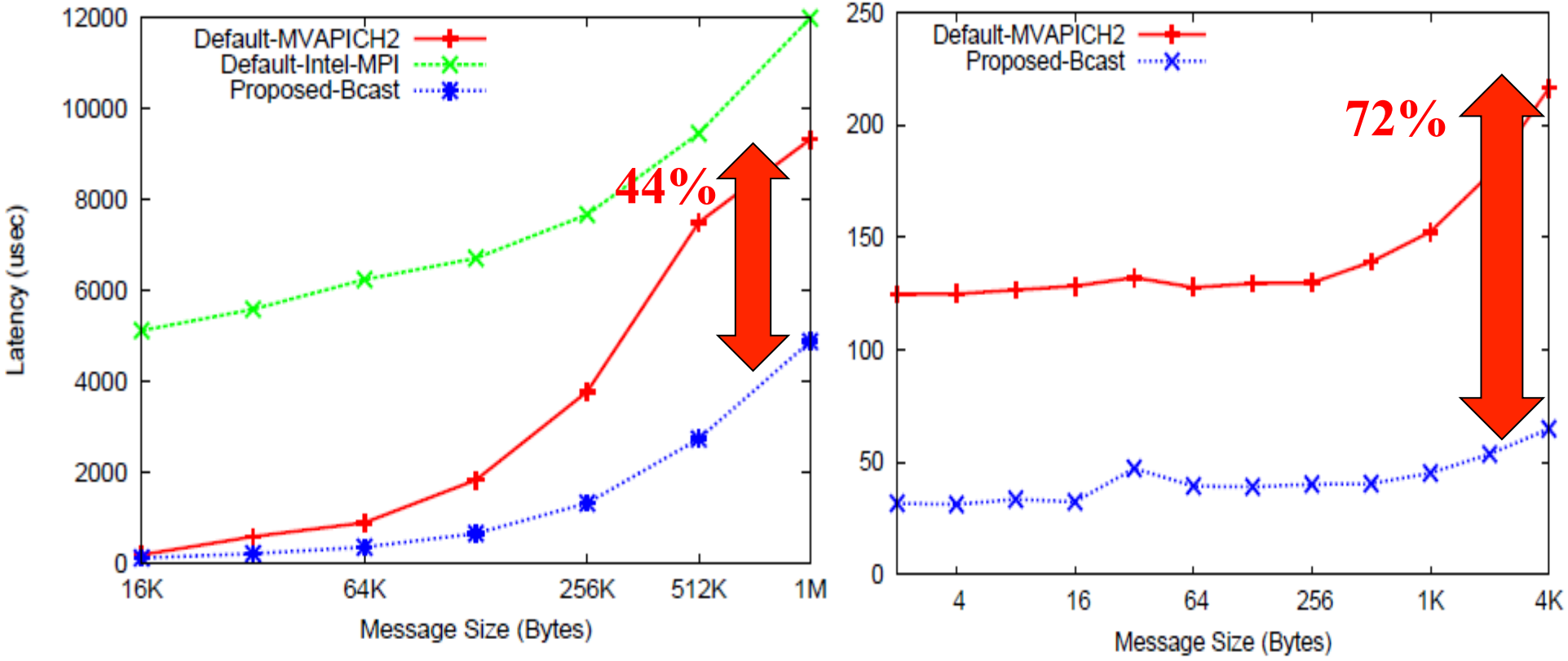
    Inter-node Design Alternatives

    Intra-node, Intra-MIC Design Alternatives
- Experimental Evaluation
- Conclusions and Future work

Experimental Setup

- TACC Stampede Cluster
- Up to 64 nodes with Sandy Bridge-EP (Xeon E5-2680) processors and Xeon Phi Coprocessors (Knight's Corner) used
- Host has 32 GB memory and MIC has 8GB
- Mellanox FDR switches (SX6036) to interconnect
- OSU Micro-benchmark suite (OMB)
- Optimizations added to MVAPICH2-1.9 (MV2-MIC)
  - Version already has intra-node SCIF based optimizations
- IMPI 4.1.1.026

MPI_Bcast Latency Comparison



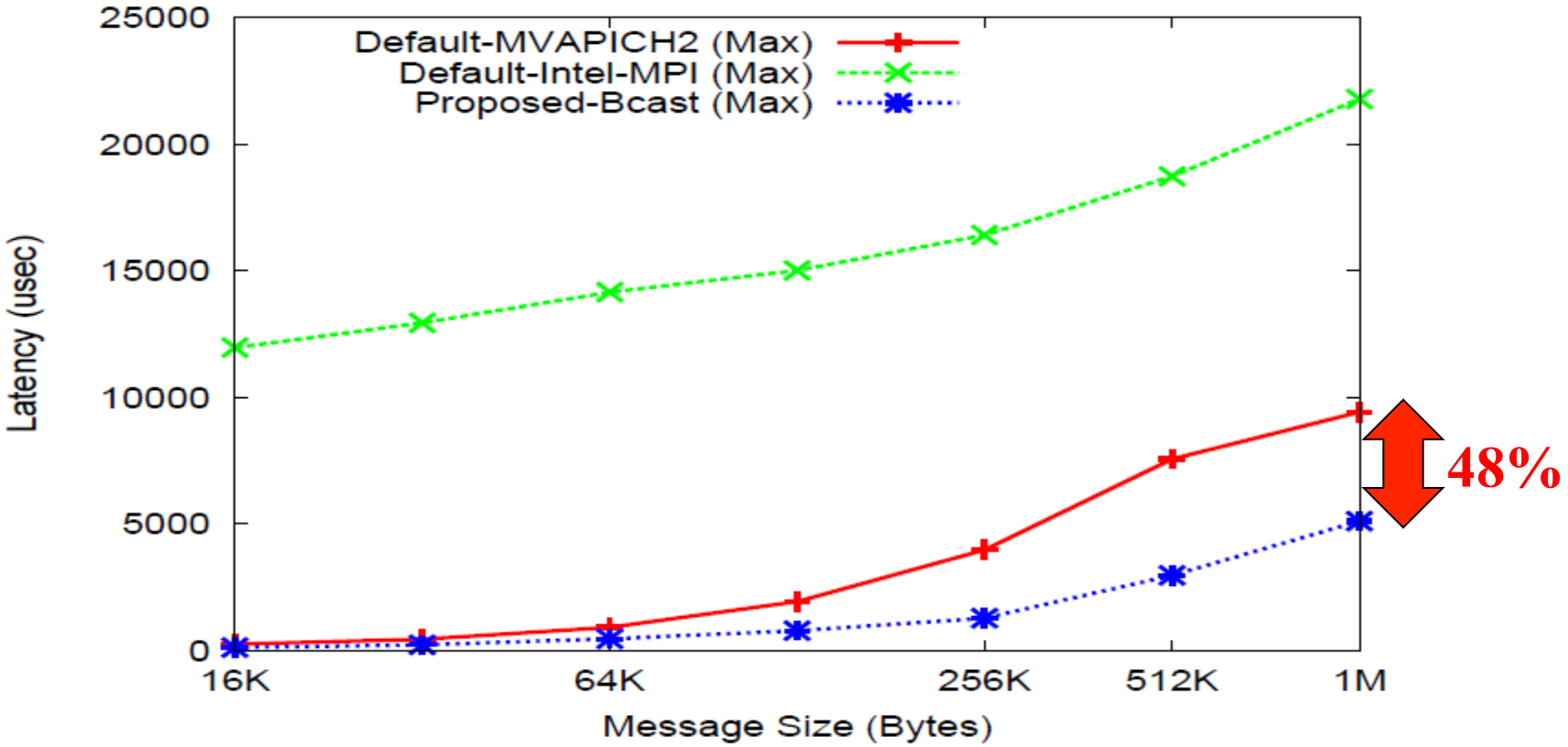
576 MPI Processes, 32 nodes (2H-16M)

4,864 MPI Processes, 64 nodes (16H-60M) (*)

Proposed Bcast designs outperform default MPI_Bcast in MVAPICH2 by up to 72%

***We were unable to scale Intel-MPI for up to 64 nodes**

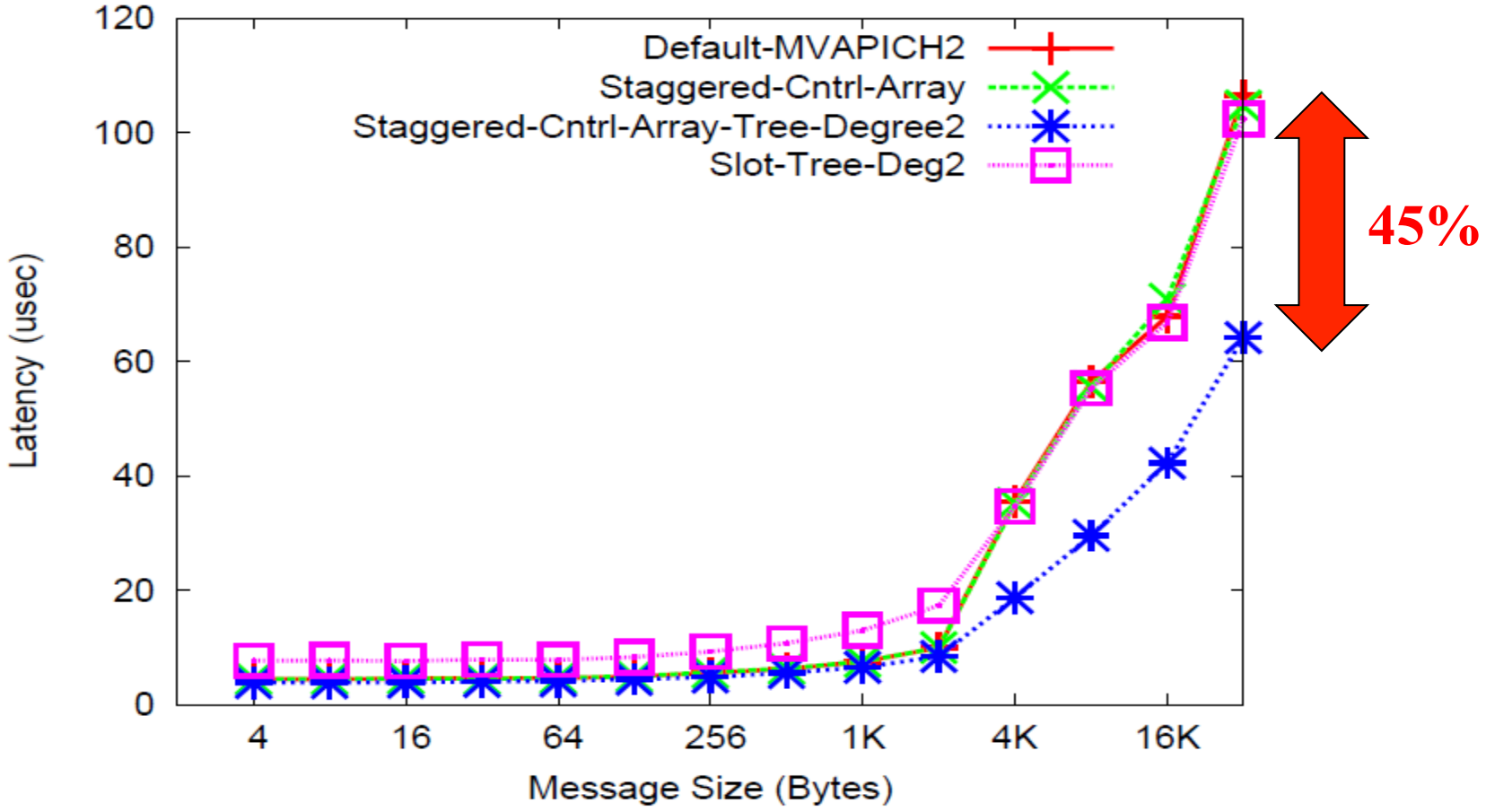
Inter-node MPI_Bcast Max Latency Comparison



1,204 MPI Processes, 32 nodes (16H-16M)

Proposed Bcast designs outperform default MPI_Bcast in MVAPICH2 by up to 48%

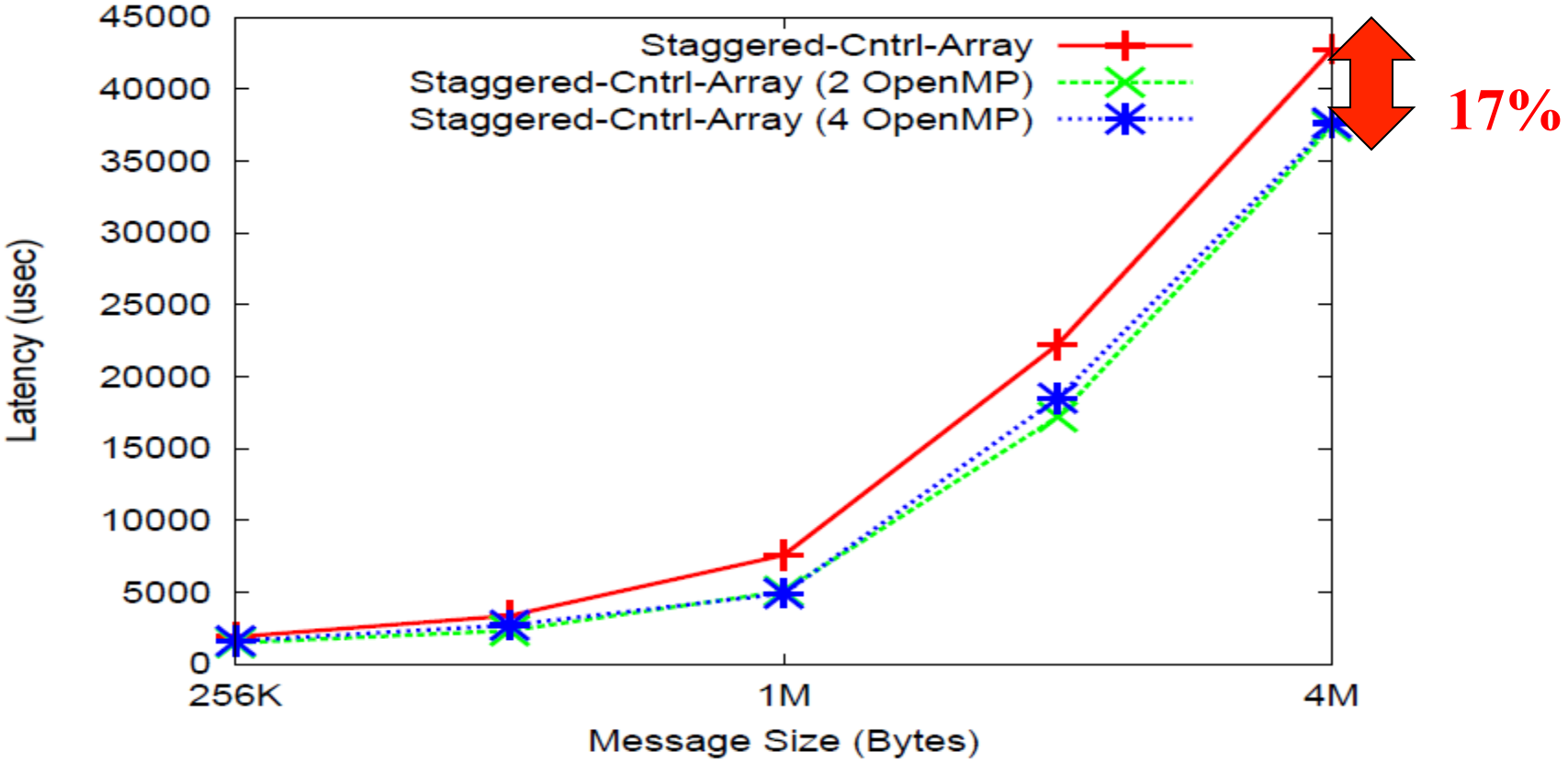
Intra-node MPI_Reduce Latency Comparison



16 MPI Processes in a MIC coprocessor

Staggered-Cntrl-Array with Tree-Degree 2 outperforms default by up to 45%

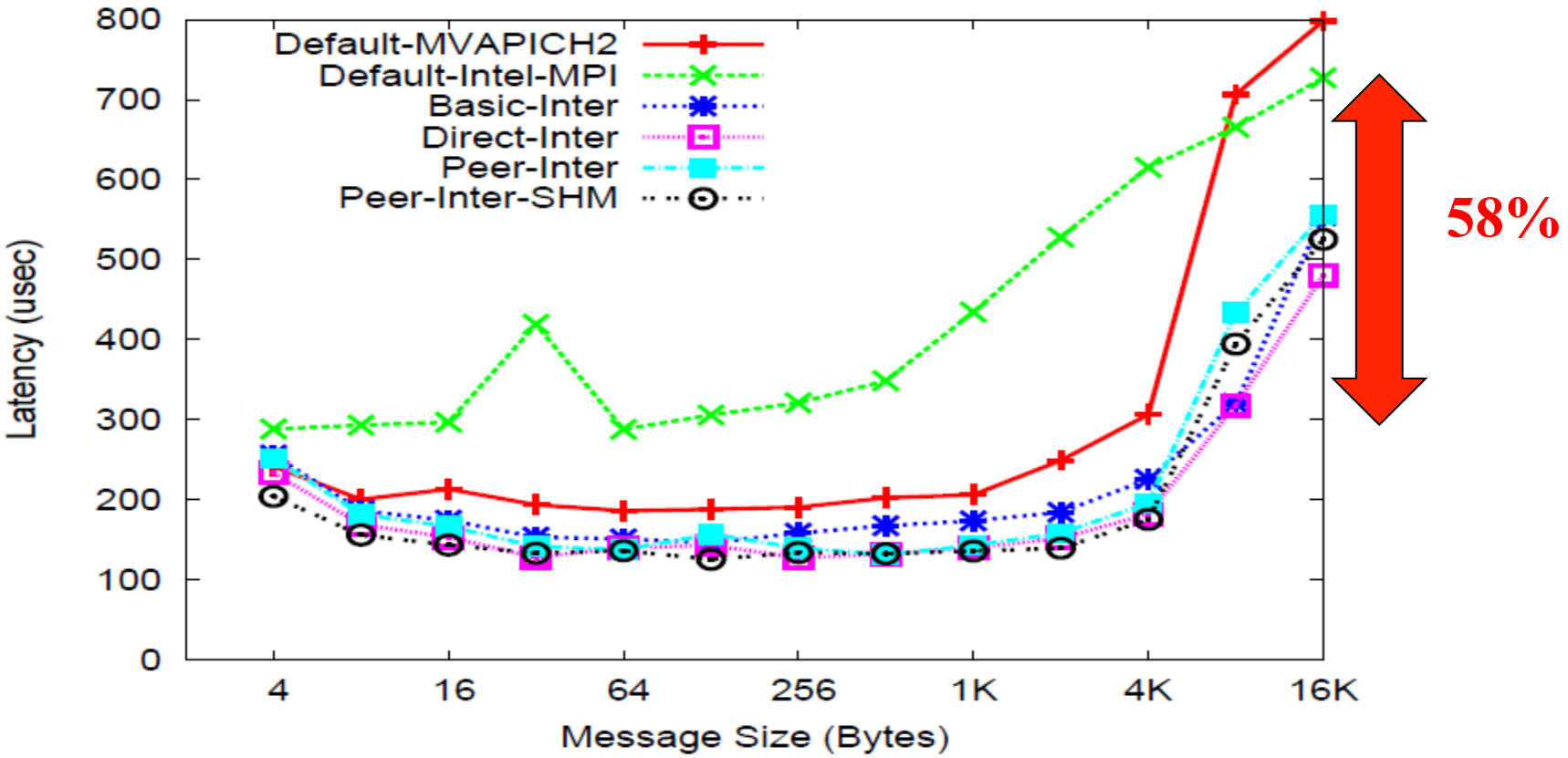
Intra-MIC MPI_Reduce Latency Comparison



16 MPI Processes in a MIC coprocessor

Staggered-Cntrl-Array with 4 OpenMP threads outperforms default by up to 17%, for very large messages

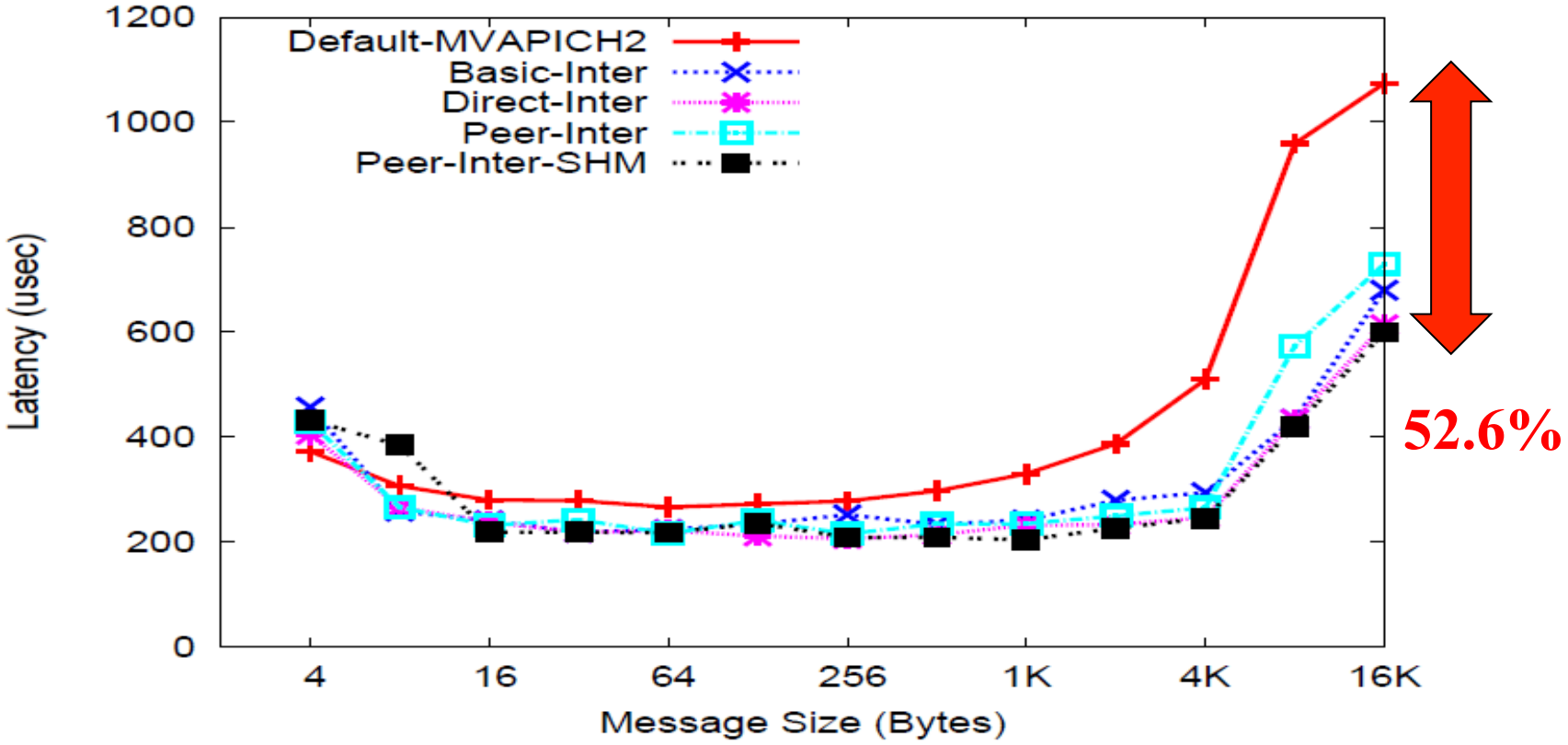
Inter-node MPI_Allreduce Latency Comparison



1,024 MPI Processes, 32 Nodes (16H-16M)

Proposed schemes are 58% better than default MVAPICH2.

Inter-node MPI_Allreduce Max Latency Comparison

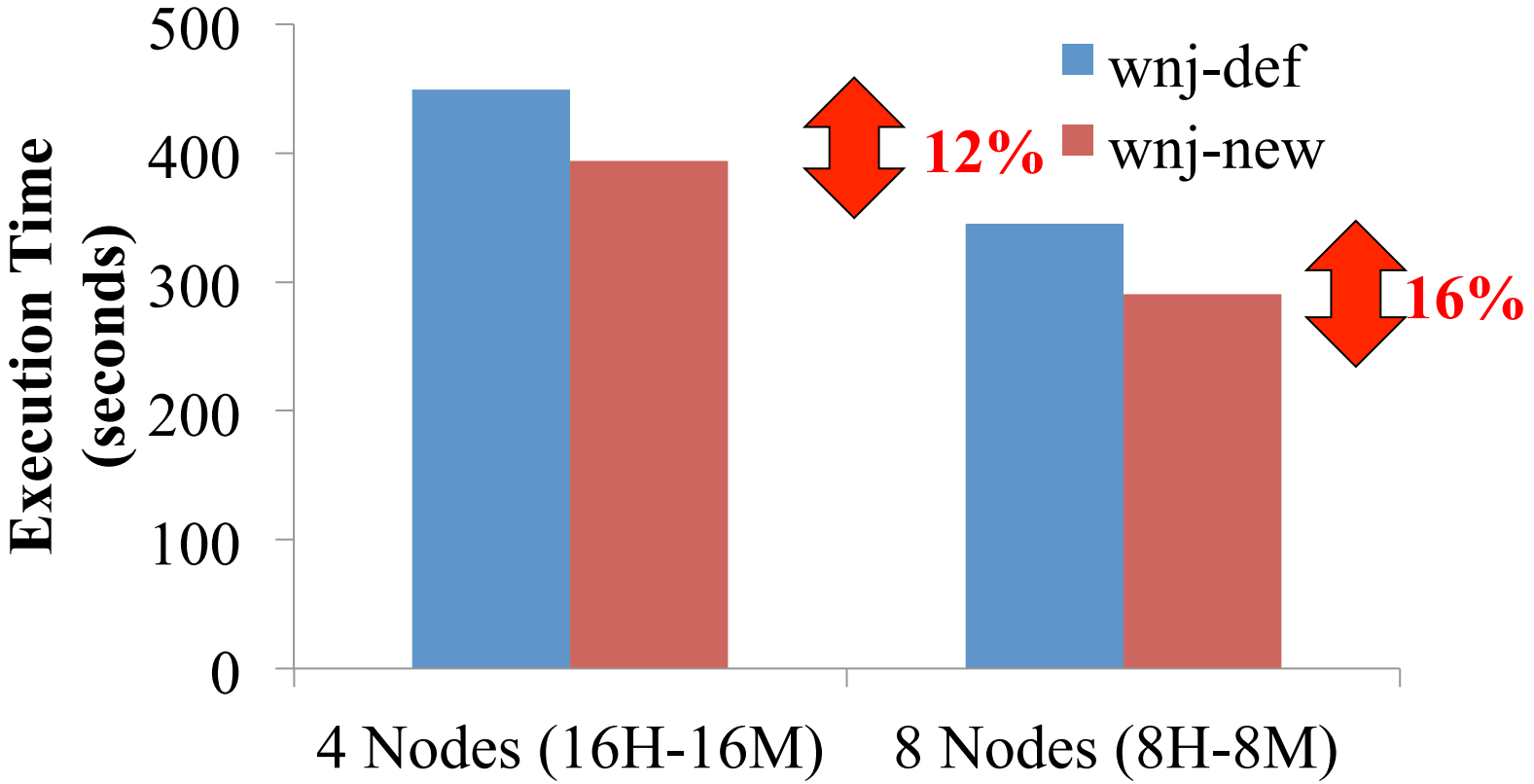


2,048 MPI Processes, 64 nodes (16H-16M) (*)

Proposed Allreduce designs outperform default MPI_Allreduce in MVAPICH2 by up to 52.6%

***We were unable to scale Intel-MPI for up to 64 nodes**

Windjammer Application Performance Comparison



Windjammer Application performance with 128 Processes

Proposed designs improve the performance of Windjammer application, by up to 16% which heavily uses broadcast

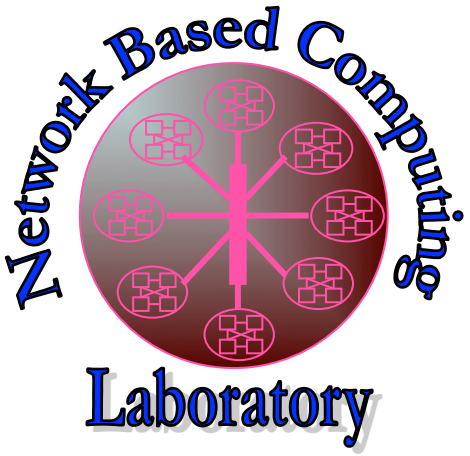
Conclusions and Future Work

- Proposed a generic framework to design collectives in a hierarchical manner on emerging MIC clusters
- Explored various design alternatives to improve intra-MIC and inter-MIC phases of collectives, such as, MPI_Reduce and MPI_Allreduce
- Proposed designs improve performance of MPI_Bcast, by up to **72%**, and MPI_Allreduce, by up to **58%**
- Also improves Windjammer application execution time by up to **16%**

Future work

- Extend designs to improve performance of other important collectives for emerging MIC clusters

Thank you!



MVAPICH

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

{kandalla, akshay, hamidouc, potluri, bureddy, panda}
@cse.ohio-state.edu
Network-Based Computing Laboratory, Ohio State University