Scaling Alltoall Collective on Multi-core Systems

Rahul Kumar, Amith R Mamidala,
Dhabaleswar K Panda

Department of Computer Science & Engineering
The Ohio State University

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

Presentation Outline

- Introduction
- Motivation & Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future Work

Introduction

- Multi-core architectures being widely used for high performance computing
  - Ranger cluster at TACC has 16 core/node and in total more than 60,000 cores
- Message Passing is the default programming model for distributed memory systems
- MPI provides many communication primitives
- MPI Collective operations are widely used in applications

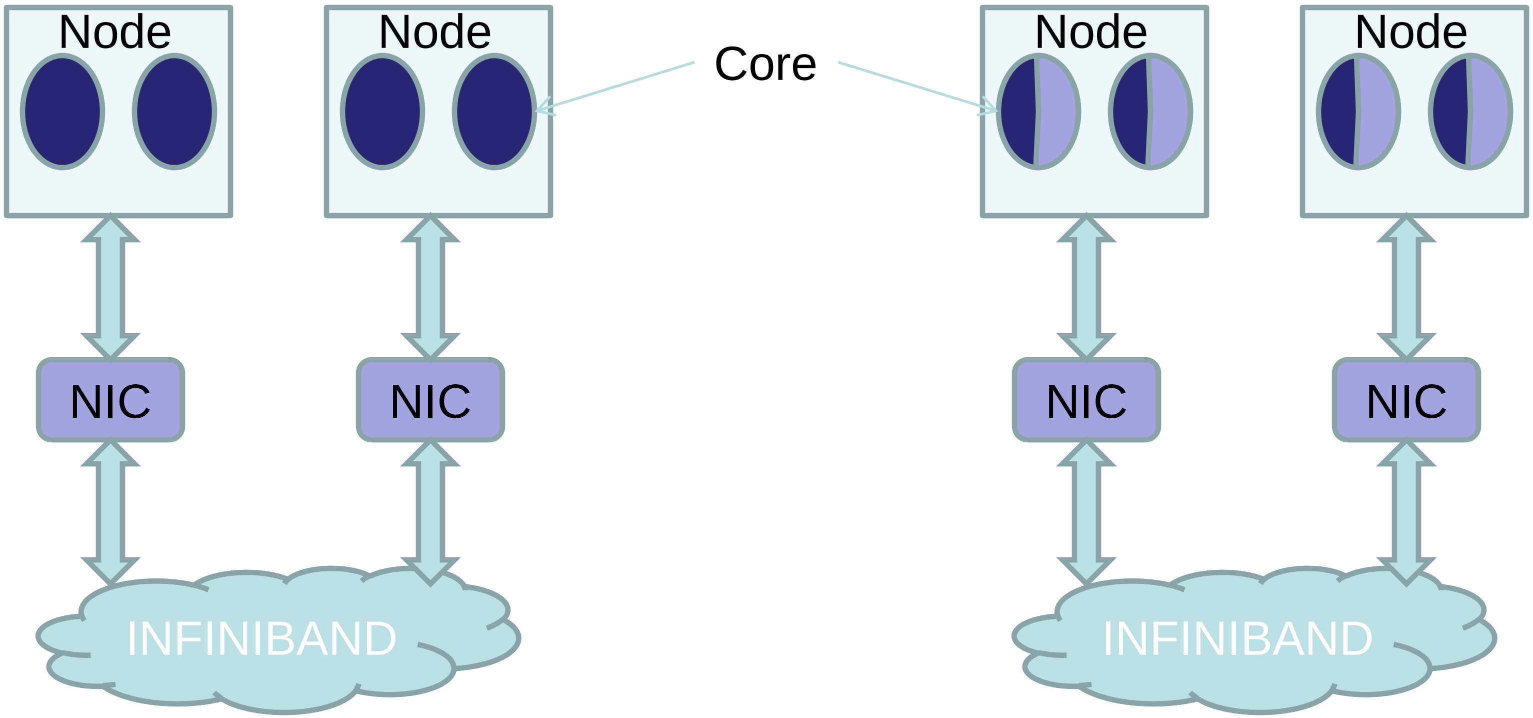
Introduction

- MPI_alltoall is the most intensive collective and is widely used in many applications such as CPMD, NAMD, FFT, Matrix transpose.
- In MPI_Alltoall every process has a different data to be sent to every other process.
- An efficient alltoall is highly desirable for multi-core systems as the number of processes have increased dramatically due to cheap cost ratio of multi-core architecture

Introduction

- 24% of the top 500 supercomputers use InfiniBand as their interconnect (based on Nov '07 rankings).
- Several different implementations of InfiniBand Network Interfaces
  - Offload implementation e.g. InfiniHost III(3rd generation cards from Mellanox)
  - Onload implementation e.g. Qlogic InfiniPath
  - Combination of both onload and offload e.g. ConnectX from Mellanox.

Offload & Onload Architecture



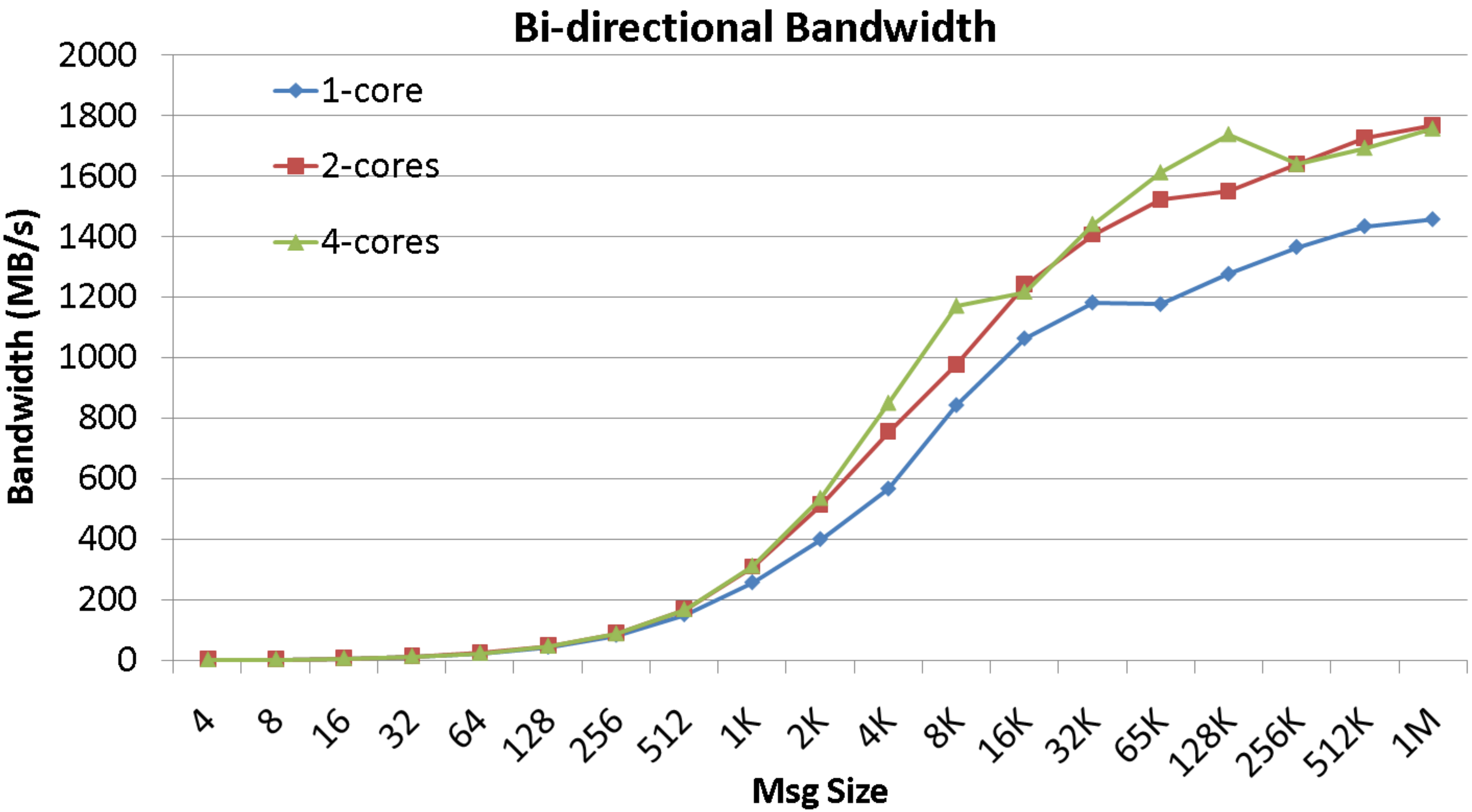
Offload architecture **Onload architecture**

- In an offload architecture, the network processing is offloaded to network interface. The NIC is able to send message relieving the CPU of communication
- In an onload architecture, the CPU is involved in communication in addition to performing the computation
- In onload architecture, the faster CPU is able to speed up the communication. However, ability to overlap communication with computation is not possible

Characteristics of various Network Interfaces

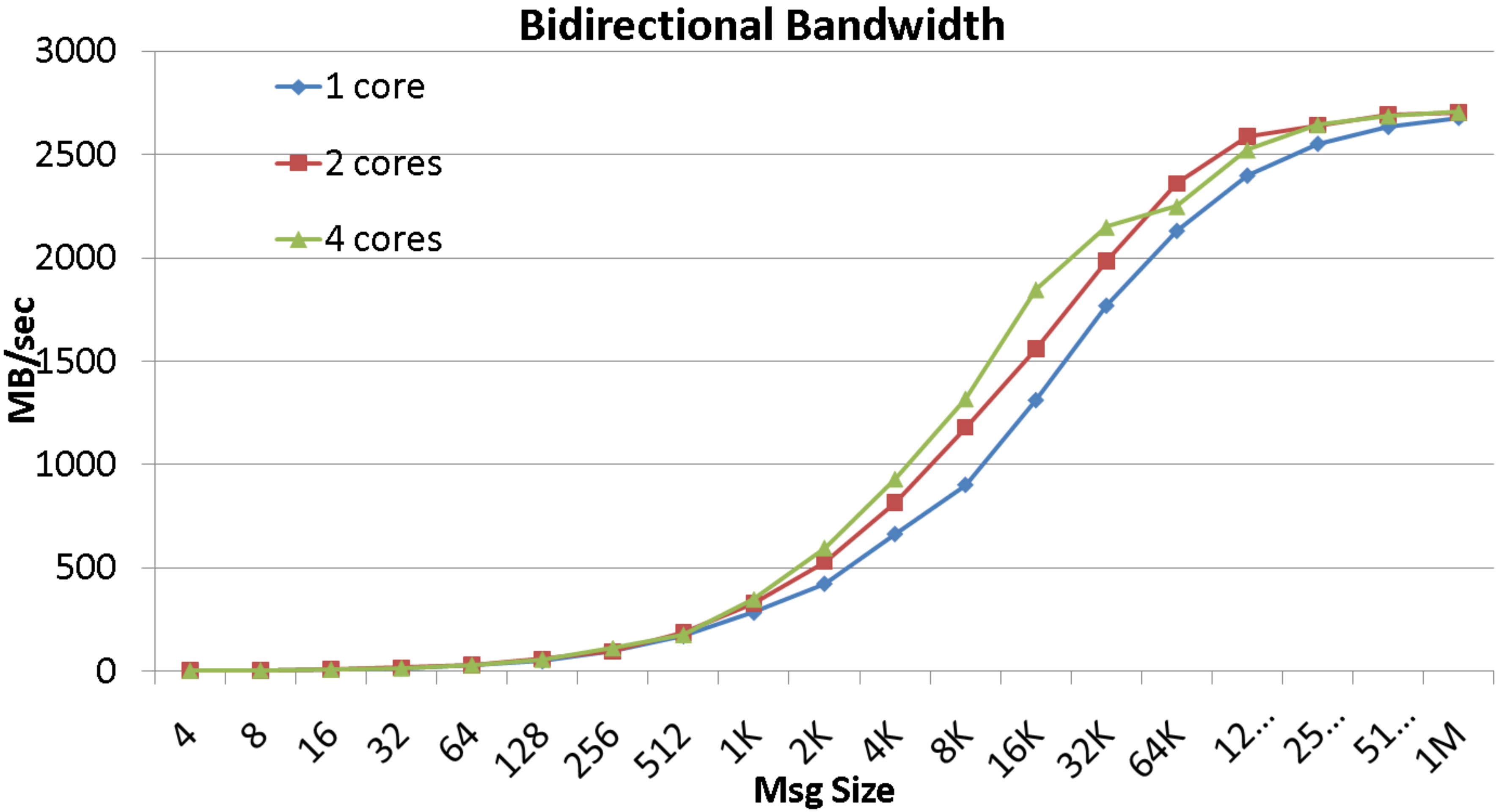
- Some basic experiments were performed on various network architectures and the following observations were made
- The bi-directional bandwidth of onload network interfaces increases with more number of cores used to push the data on the network
- This is shown in the following slides

Bi-directional Bandwidth: InfiniPath (onload)



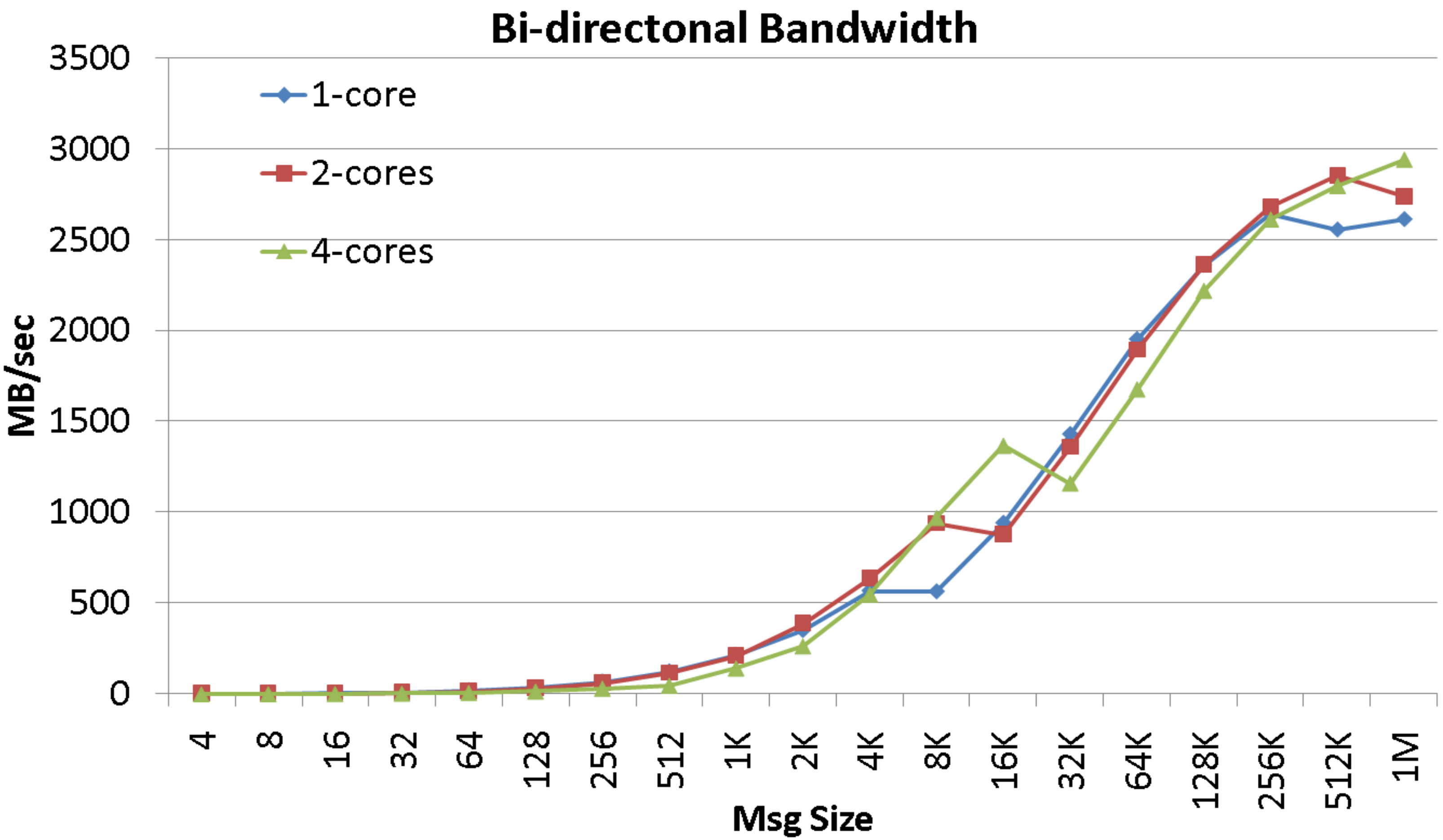
- **Bidirectional Bandwidth increases with more cores used to push data**
- In onload interface, more cores help achieve better network utilization

Bi-directional Bandwidth: ConnectX



- A similar trend is also observed for connectX network interfaces

Bi-directional Bandwidth: InfiniHost III (offload)



- However, **in Offload network interfaces the bandwidth drops on using more cores**
- We feel this to be due to congestion at the network interface on using many cores simultaneously

Results from the Experiments

- **Depending on the interface implementation, their characteristics differ**
  - **Qlogic onload implementations**: Using more cores simultaneously for inter-node communication is beneficial
  - **Mellanox offload implementations:** Using less cores at the same time for inter-node communication is beneficial
  - **Mellanox ConnectX architecture:** Using more cores simultaneously is beneficial

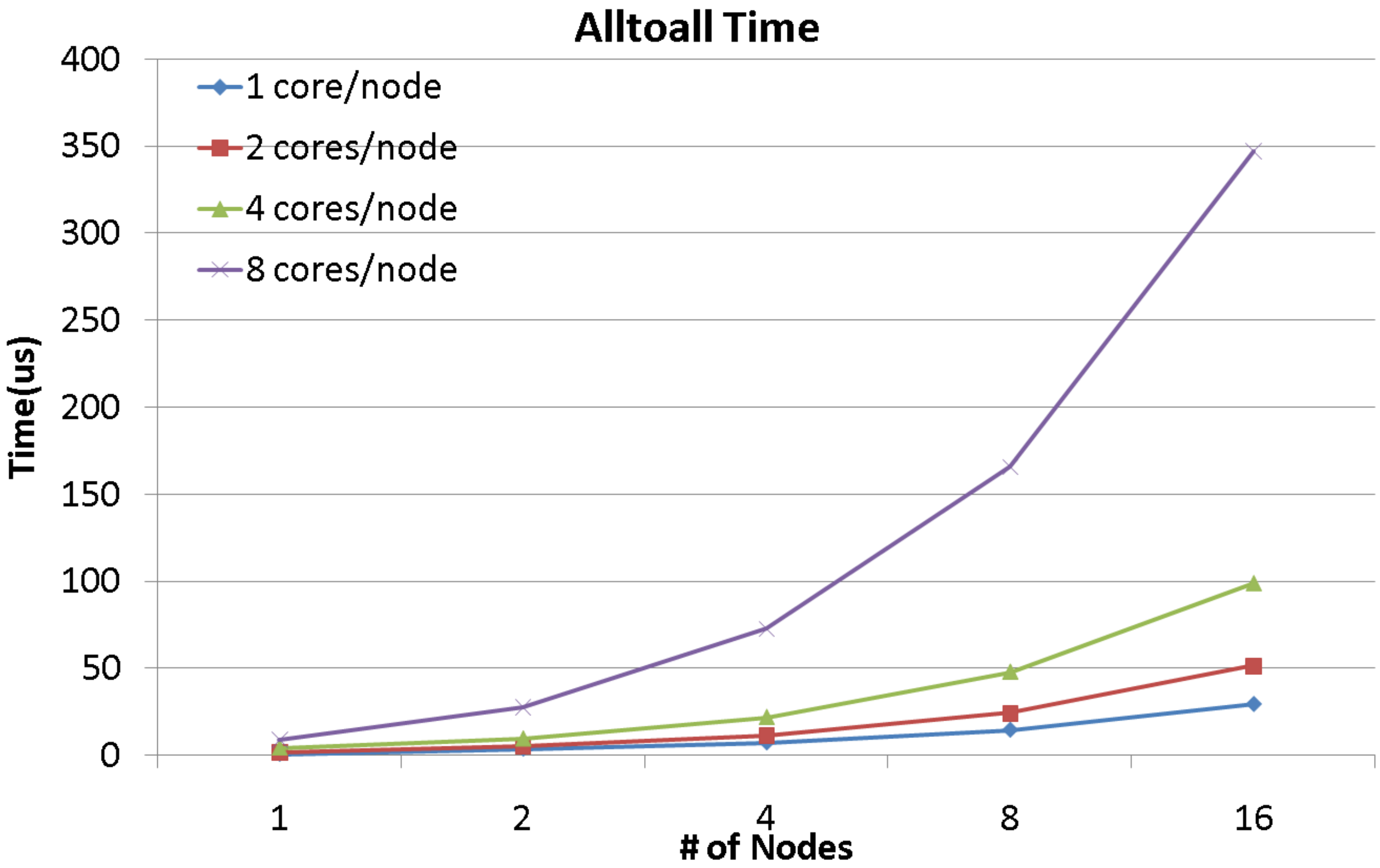
Presentation Outline

- Introduction
- Motivation & Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future Work

Motivation

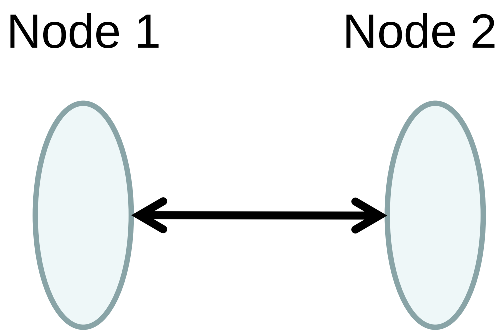
- To evaluate the performance of existing alltoall algorithm we conduct the following experiment
- In the experiment alltoall time is measured on a set of nodes.
- The number of cores per node participating in alltoall are increased gradually.

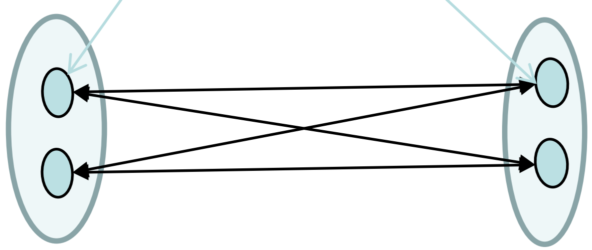
Motivation



•The alltoall time doubles on doubling the number of cores in the nodes

What is the problem with the Algorithm?





- Alltoall between two nodes involves one communication step

- With two cores per node, the number of inter-node communication by each core increases to two

- So on doubling the core alltoall time is almost doubled.
- This is exactly what we obtained from the previous experiment.

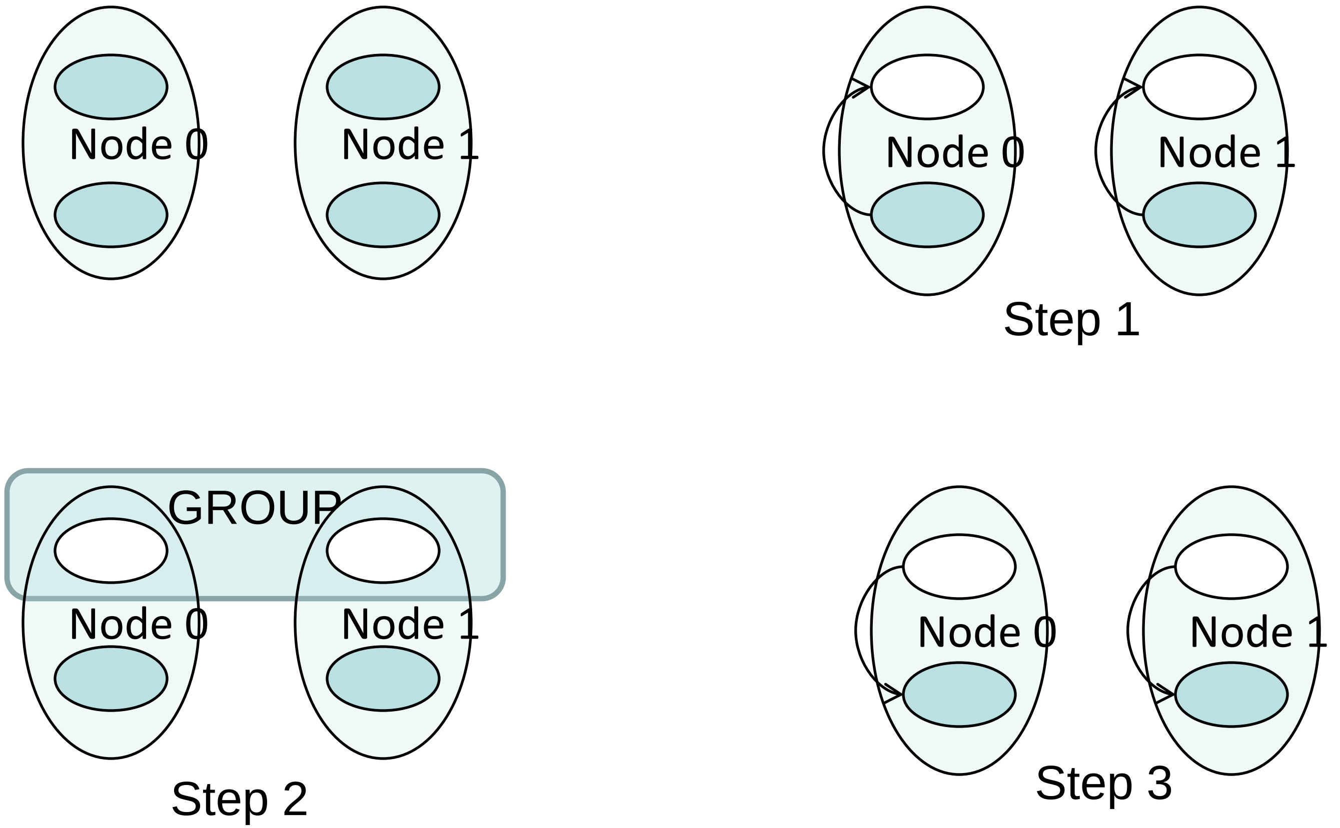
Problem Statement

- Can low cost shared memory help to avoid network transactions?
- Can the performance of alltoall be improved especially for multi-core systems?
- What algorithms to choose for different infiniband implementations?

Related Work

- There have been studies that propose a leader-based hierarchical scheme for other collectives
  - A leader is chosen on each node
  - Only the leader is involved in inter-node communication
  - The communication takes place in three stages
    - The cores aggregate data at the leader of the node
    - The leader perform inter-node communication
    - The leader distributes the data to the cores
- We implemented the above scheme for Alltoall as illistrated in the diagram in next slide

Leader-based Scheme for Alltoall



- **step 1**: all cores send data to the leader
- **step 2**: the leader performs alltoall with other leader
- **step 3**: the leader distributes the respective data to other cores

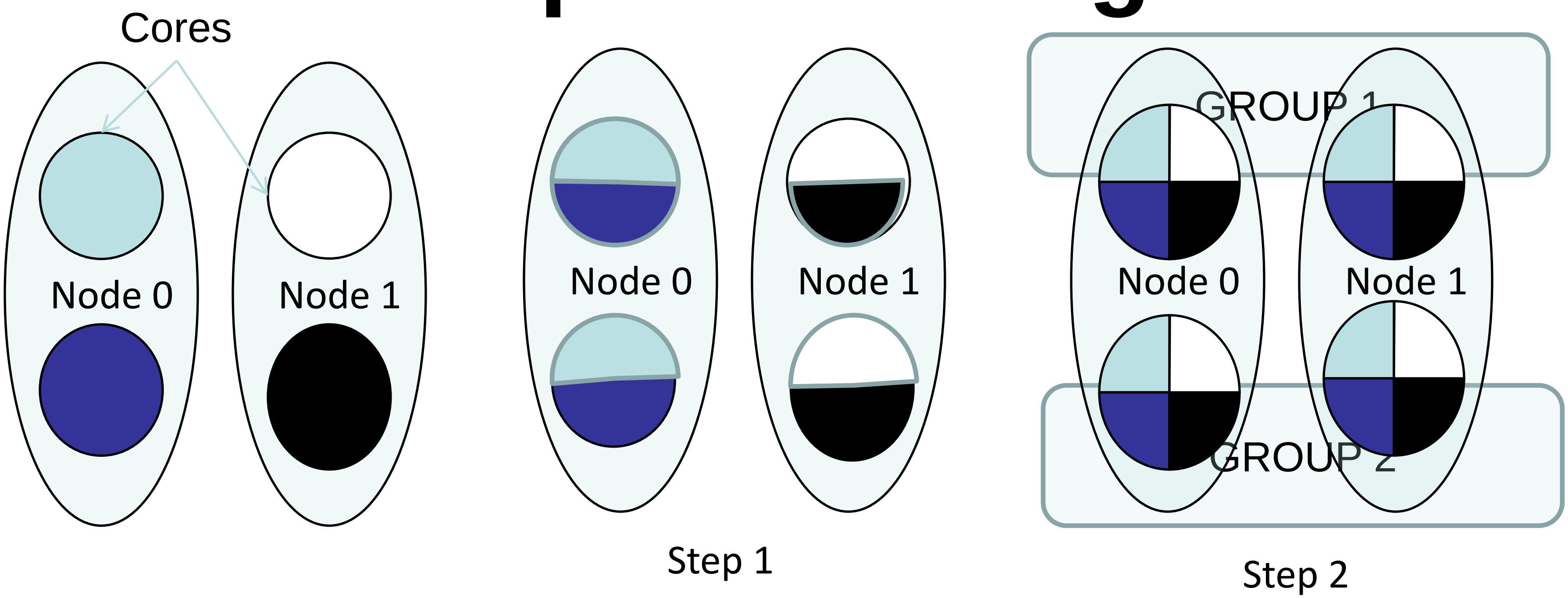
Issues with Leader-based Scheme

- It uses only one core to send the data out on the network
- Does not take advantage of increase in bandwidth with the use of more cores to send the data out of the node

Presentation Outline

- Introduction
- Motivation & Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future Work

Proposed Design



- All the cores take part in the communication
- Each core communicates with one and only one core from other nodes
- Step 1: Intra-node Communication
  - The data destined for other nodes is exchanged among the cores
  - The core which communicates with the respective core of the other node receives the data
- Step 2: Inter-node Communication
  - Alltoall is called among each group

Advantages of the Proposed Scheme

- The scheme takes advantage of low cost shared memory
- It uses multiple cores to send the data out on the network, thus achieving better network utilization
- Each core issues same number of sends as the leader-based scheme, hence start-up costs are lower

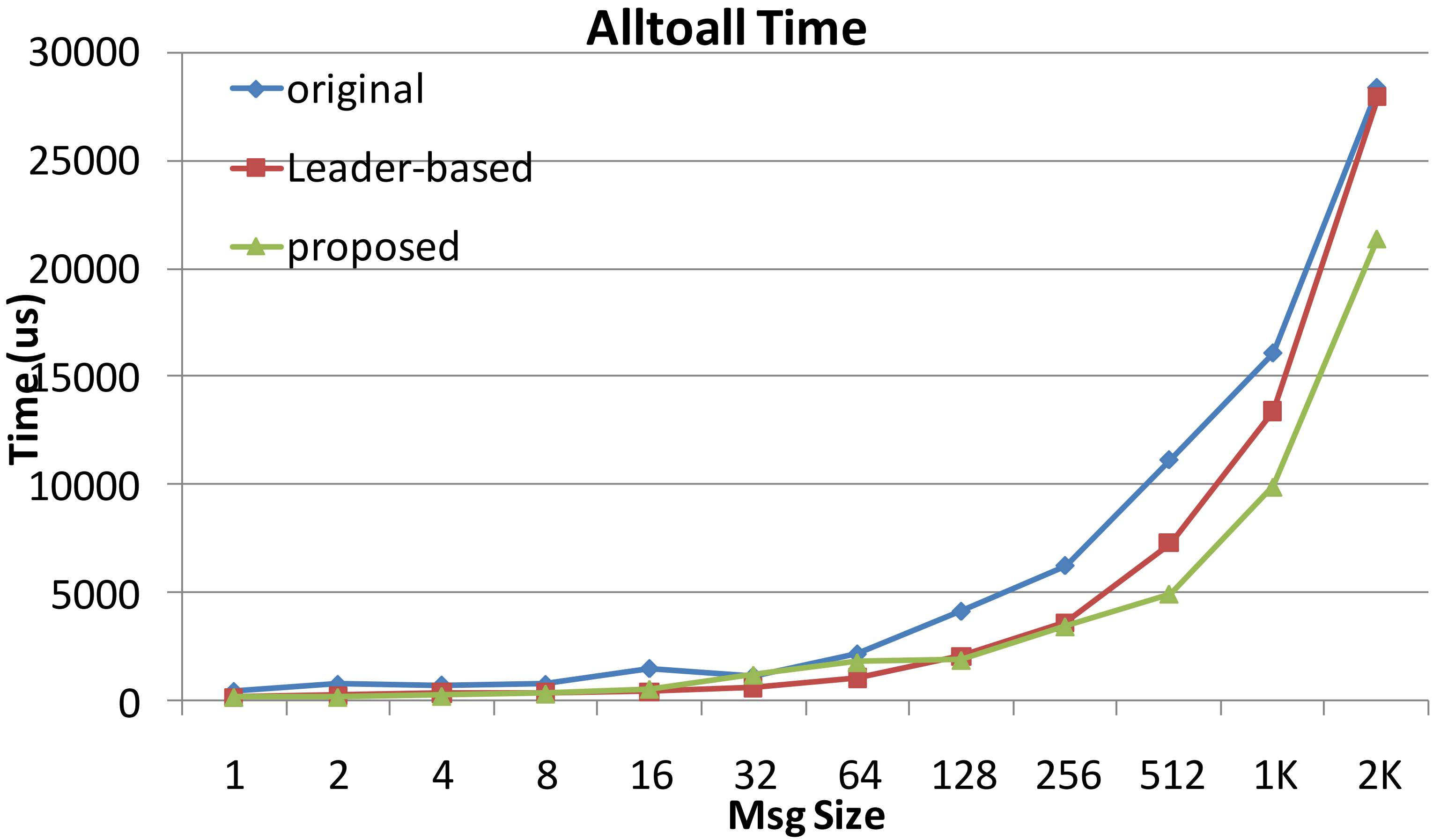
Presentation Outline

- Introduction
- Motivation & Problem Statement
- Proposed Design
- Performance Evaluation
- **Conclusion & Future Work**

Evaluation Framework

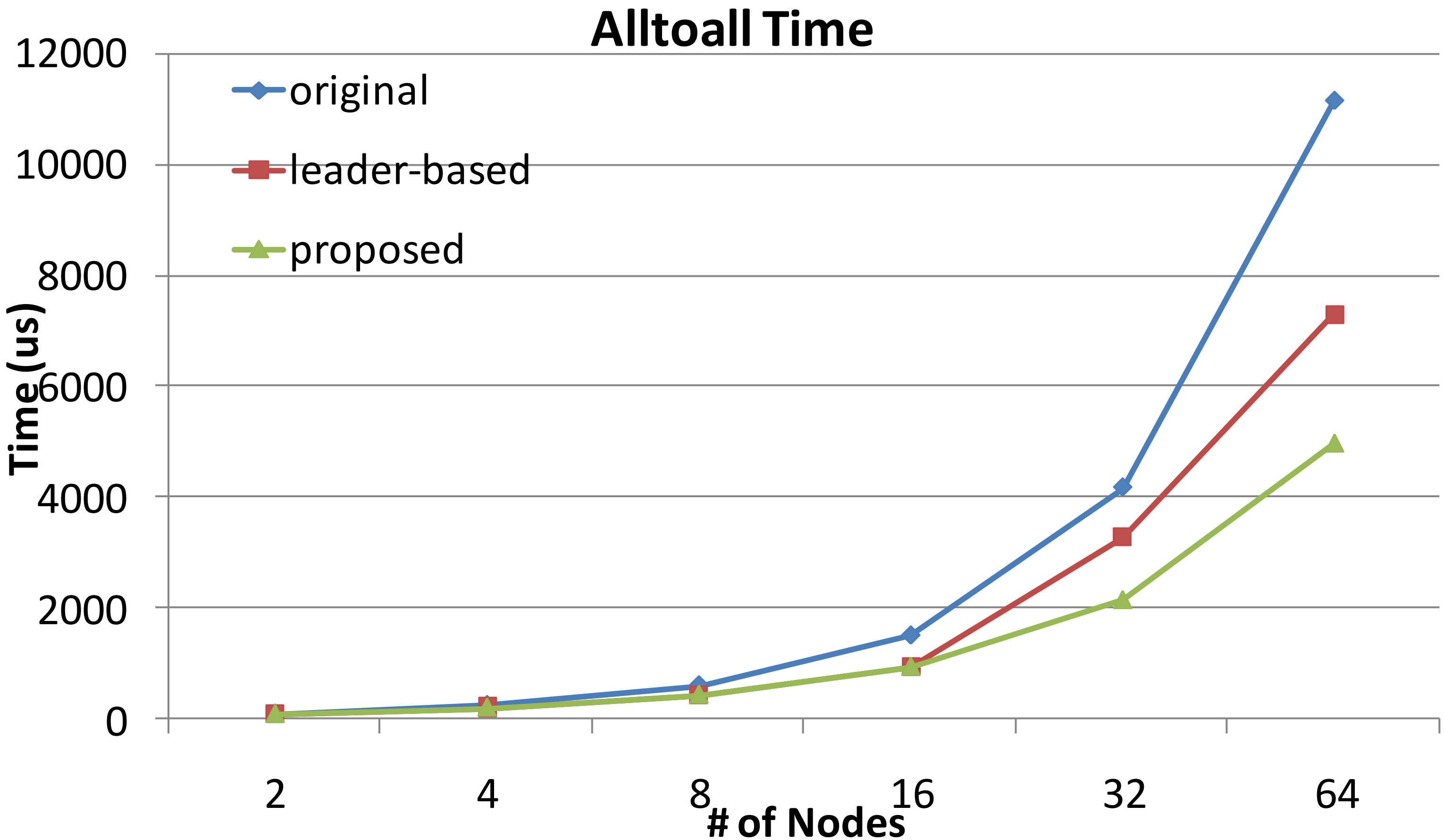
- Testbed
  - Cluster A: 64 node (512 cores)
    - dual 2.33 GHz Intel Xeon “Clovertown” quad-core
    - InfiniPath SDR network interface QLE7140
    - InfiniHost III DDR network interface card MT25208
  - Cluster B: 4 node (32 cores)
    - dual 2.33 GHz Intel Xeon “Clovertown” quad-core
    - Mellanox DDR ConnectX network interface
- Experiments
  - Alltoall collective time
    - Onload InfiniPath network interface
    - Offload InfiniHost III network interface
    - ConnectX network interface
  - CPMD Application performance

Alltoall: InfiniPath



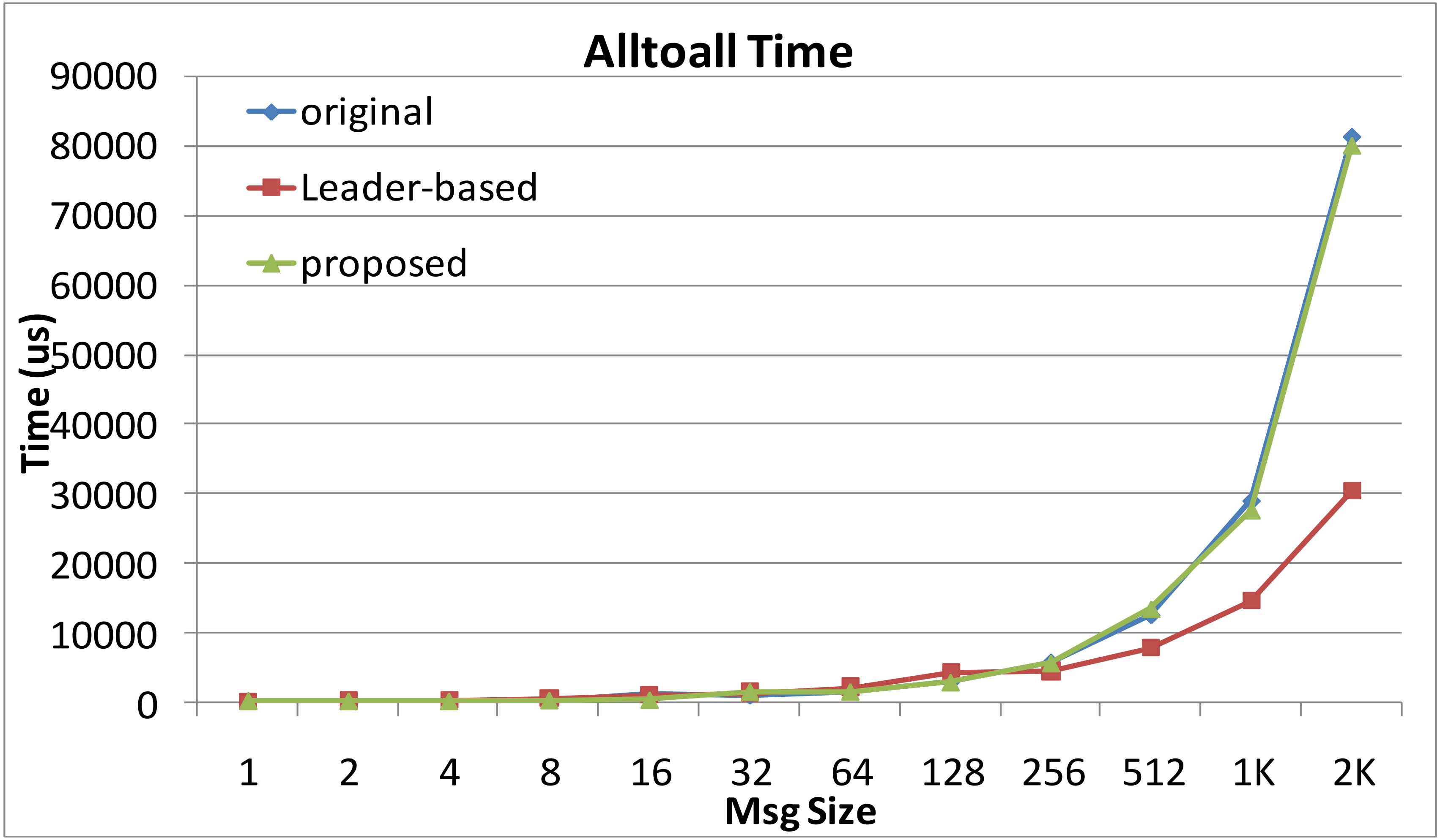
- The figure shows the alltoall time for different message size on 512 core system
- Leader-based reduces the alltoall time
- **Proposed design gives the best performance on onload network interfaces**

Alltoall-InfiniPath: 512 Bytes Message



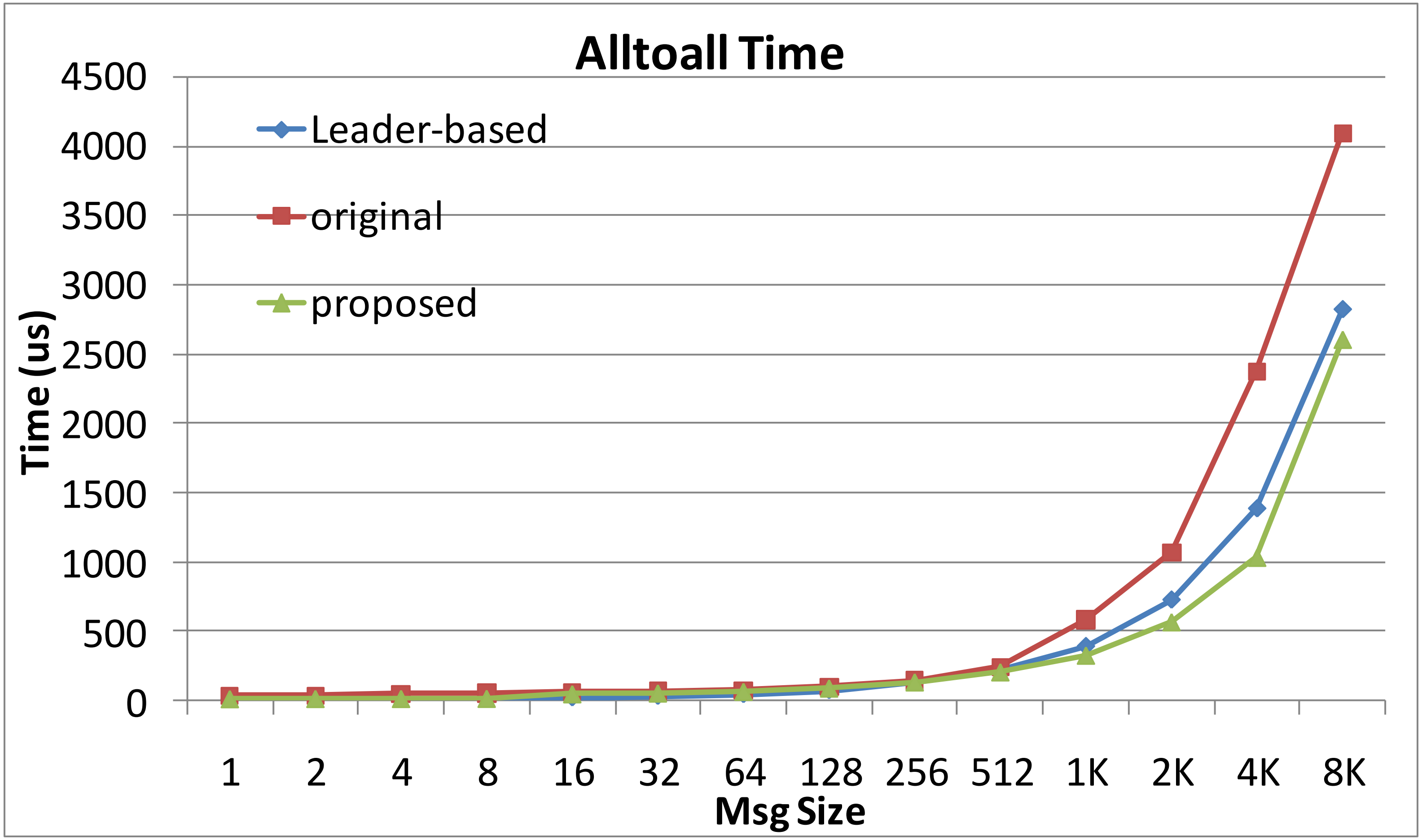
- The figure shows the alltoall time for 512 Bytes message on varying system size
- The **proposed scheme scales** much better than other schemes on increase in system size

Alltoall: InfiniHost III



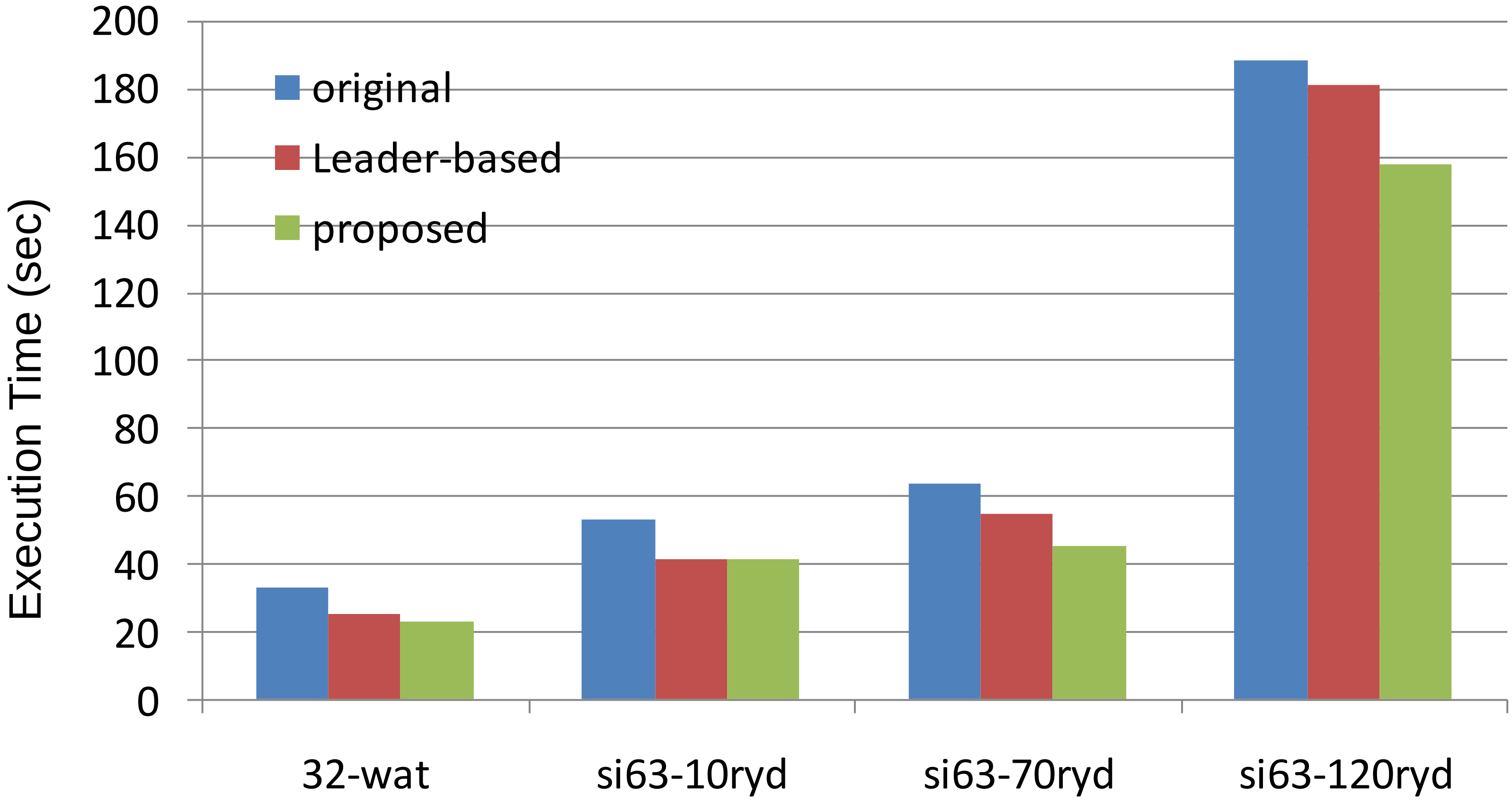
- The figure shows the performance of the schemes on offload network interfaces
- Leader-based scheme performs best on offload NIC as it avoids congestion.
- **This matches our expectations**

Alltoall: ConnectX



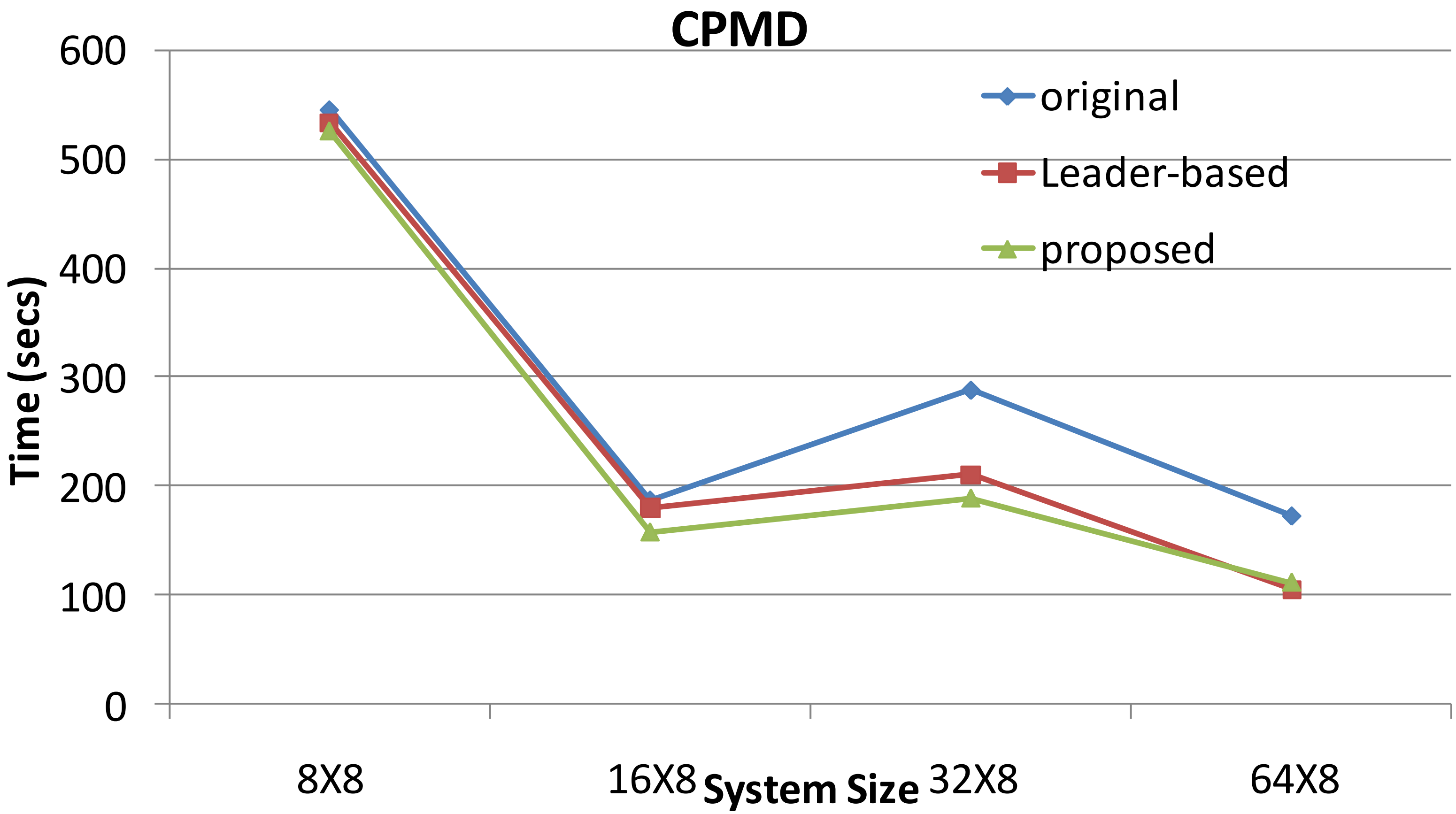
- As seen earlier, bi-directional bandwidth increases with the use of more cores on ConnectX architecture
- Therfore, the **proposed scheme attains the best performance**

CPMD Application



- CPMD is designed for ab-initio molecular dynamics. CPMD makes extensive use of alltoall communication.
- Figure shows the performance improvement of CPMD Application on 128 core system
- The **proposed design delivers the best execution time**

CPMD Application Performance on Varying System Size



- This figure shows the application execution time on different system sizes.
- The reduction in application execution time increases with increasing system sizes. **Proposed design scales very well.**

Presentation Outline

- Introduction
- Motivation & Problem Statement
- Proposed Design
- Performance Evaluation
- Conclusion & Future Work

Conclusion & Future Work

- Interfaces implemented for the same interconnect, exhibit different network characteristics.
- A single collective algorithm does not perform optimally for all network interfaces.
- We have proposed an optimized alltoall collective algorithm for multi-core systems connected using modern InfiniBand network interfaces.
- The proposed design achieves a reduction in MPI_Alltoall time by 55% and speeds up the CPMD application by 33%.
- We plan to evaluate our designs on 10GigE-based systems.
- And also extend the study to other collectives like broadcast and allgather.

Web Pointers

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

MVAPICH Web Page

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

Acknowledgements

Our research is supported by the following organizations

- Current Funding support by
- Current Equipment support by