NETWORK-BASED COMPUTING LABORATORY

Designing Multi-Leader based Allgather Algorithms for Multi-core Clusters

Krishna Kandalla, Hari Subramoni, Gopal Santhanaraman, Matthew Koop and Dhabaleswar. K. Panda

Computer Science & Engineering Department
The Ohio State University

OHIO STATE

Outline

- Introduction and Background
- Motivation
- Related Work
- Multi-Leader based Algorithms
- Experimental evaluation
- Conclusions and Future Work

Introduction and Background

- MPI is the de-facto programming model for HPC
- Multi-core clusters are becoming increasingly common
- Modern interconnects like InfiniBand offer high-bandwidth and low-latency
- The collective communication primitives consume a significant amount of time
- Necessary to have multi-core aware collective designs

Allgather Communication

- Each process broadcasts a vector data to every other process in the group
- Commonly used algorithms :
  - Recursive Doubling (RD) Algorithm for small messages

    $$tcomm = ts * log(p) + tw * (p - 1) * m$$

  - Ring Algorithm for large messages

    $$tcomm = (ts + tw * m) * (p - 1)$$

tcomm - *Total Communication cost*

ts - *Communication start-up cost*

tw - *Cost of sending a byte of data*

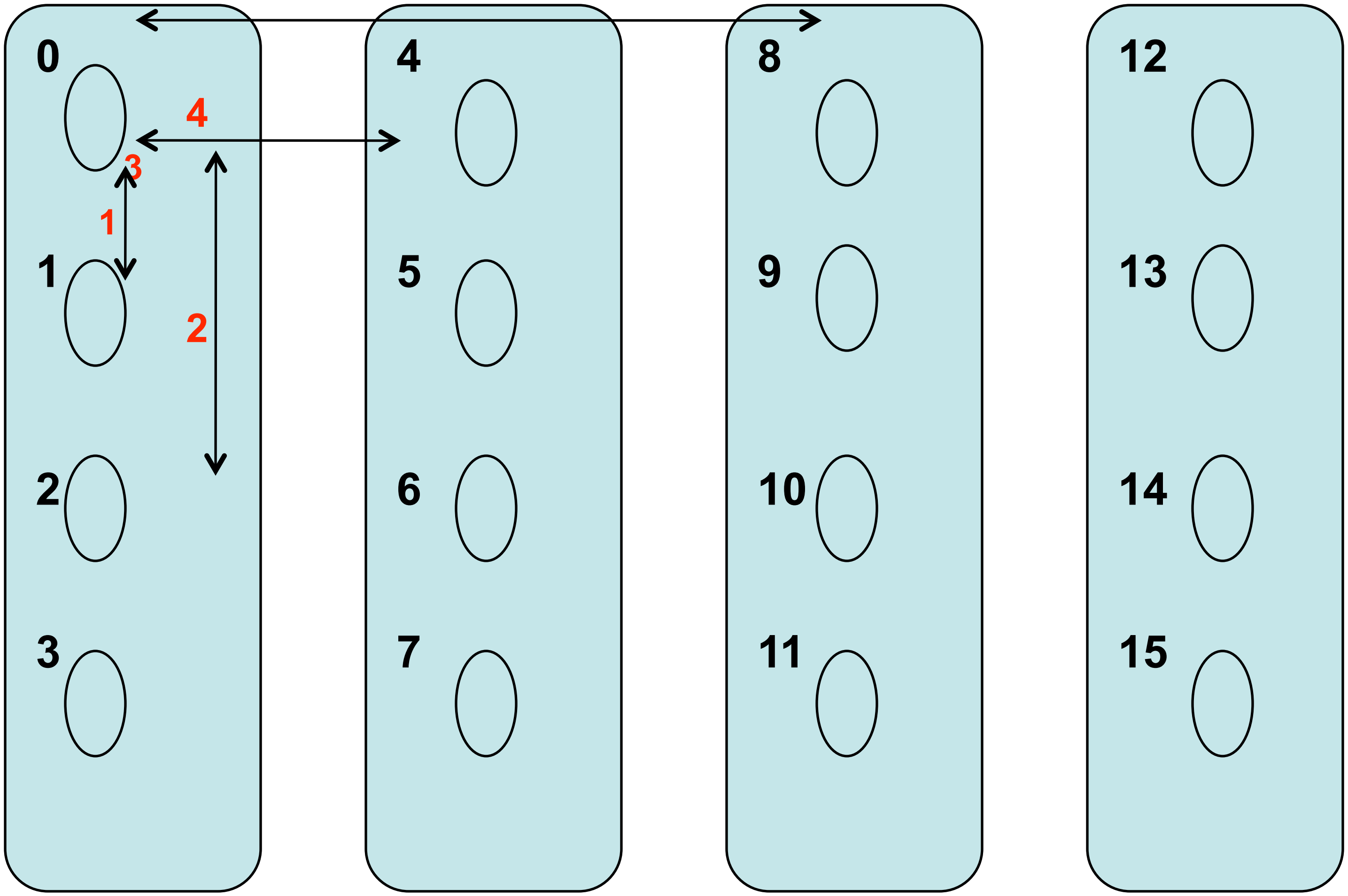
p - *Number of processes*

m - *Message Size.*

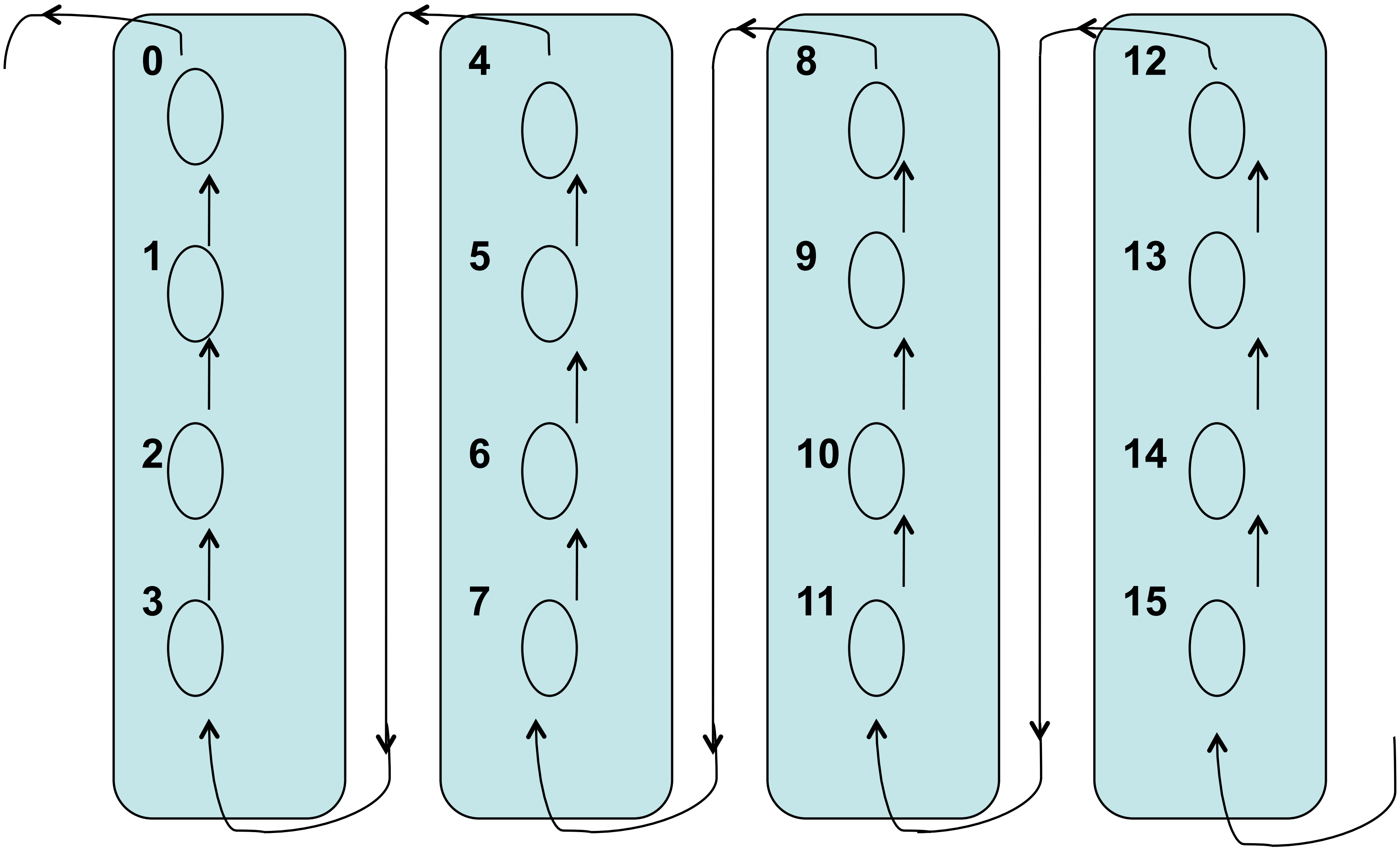
Outline

- Introduction and Background
- Motivation
- Related Work
- Multi-Leader based Algorithms
- Experimental evaluation
- Conclusions and Future Work

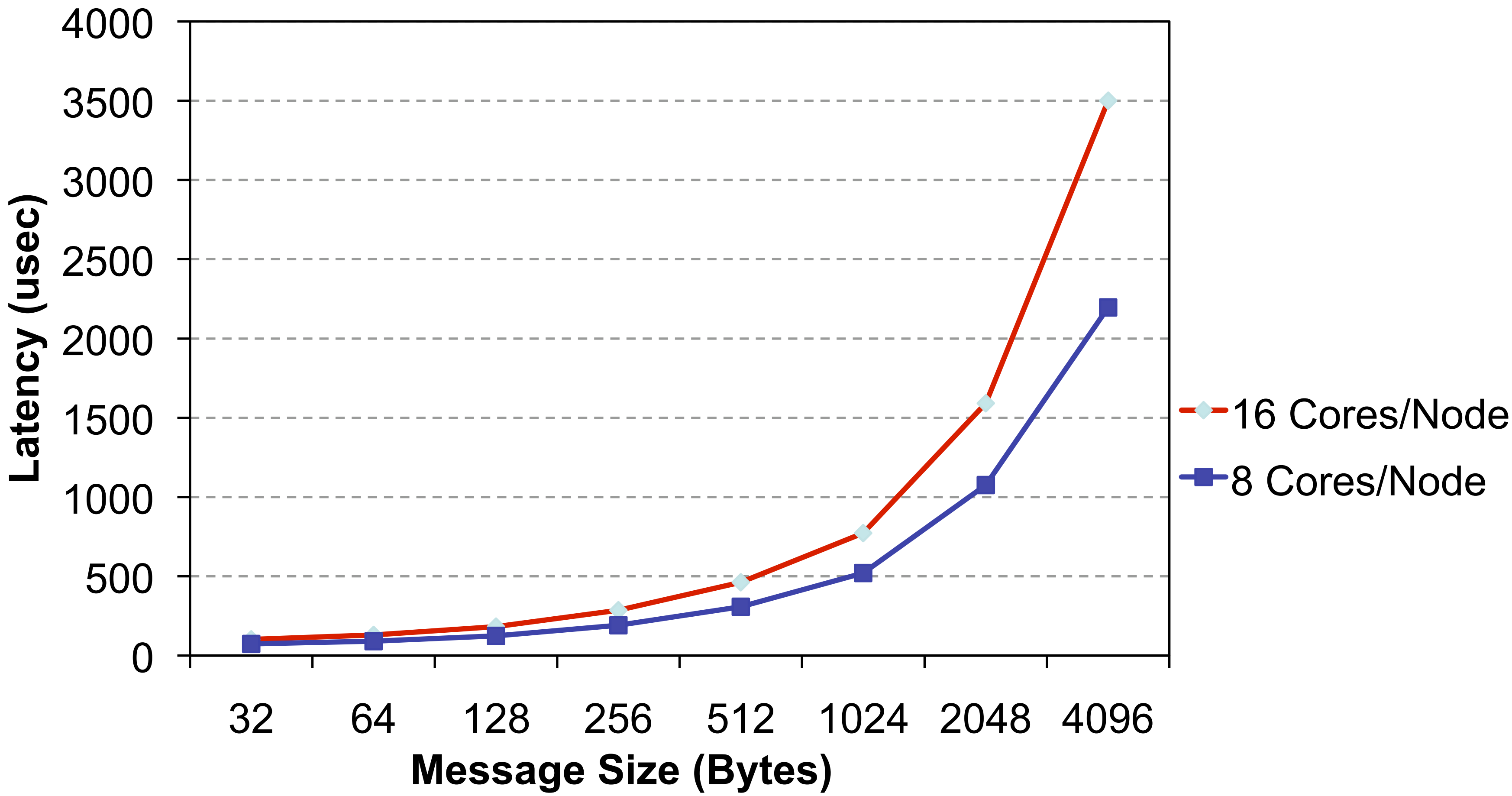
Recursive Doubling (RD) Algorithm on Multi-cores



Ring Algorithm on Multi-cores

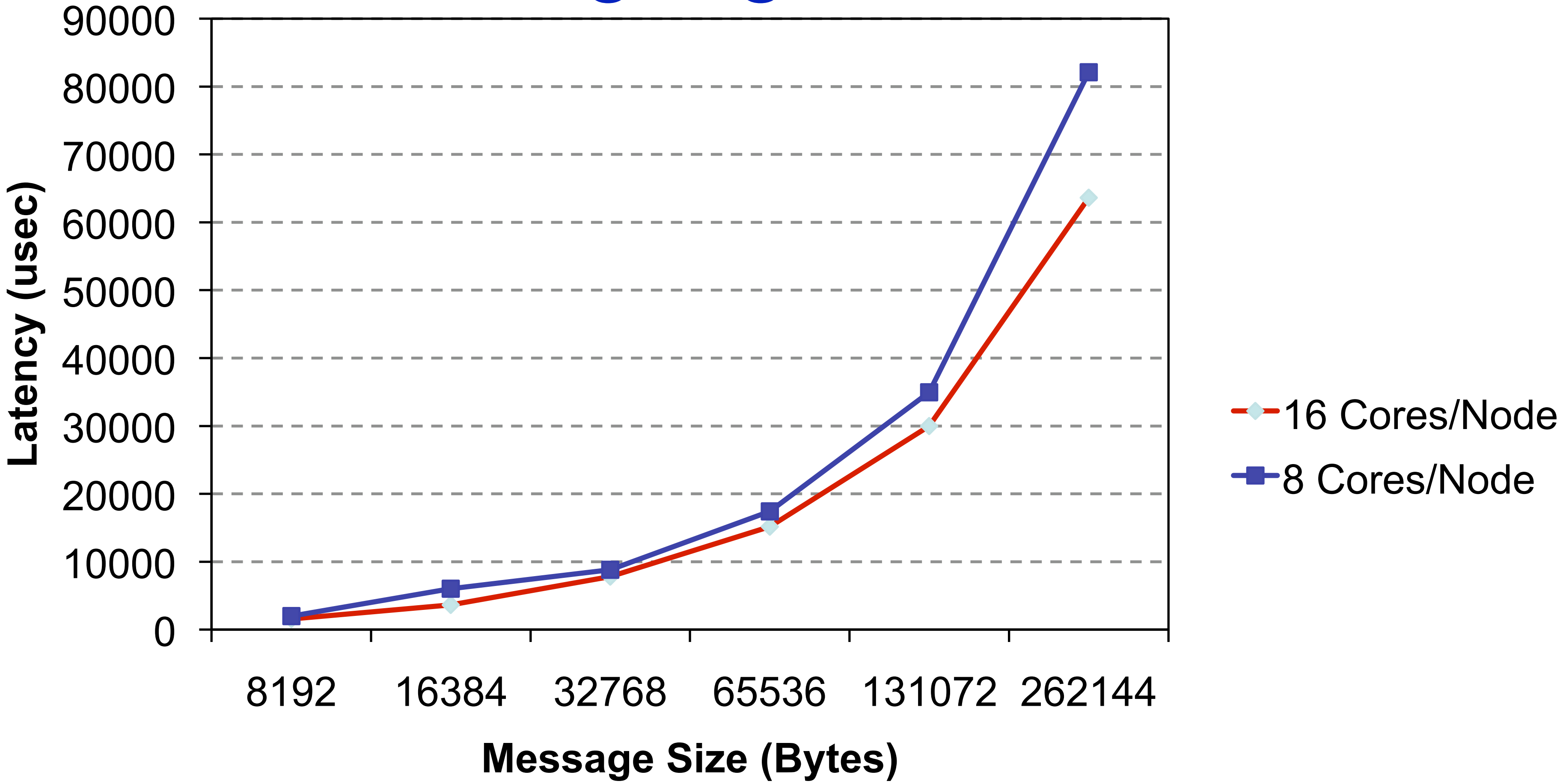


Scaling on Multi-cores : Recursive Doubling Algorithm



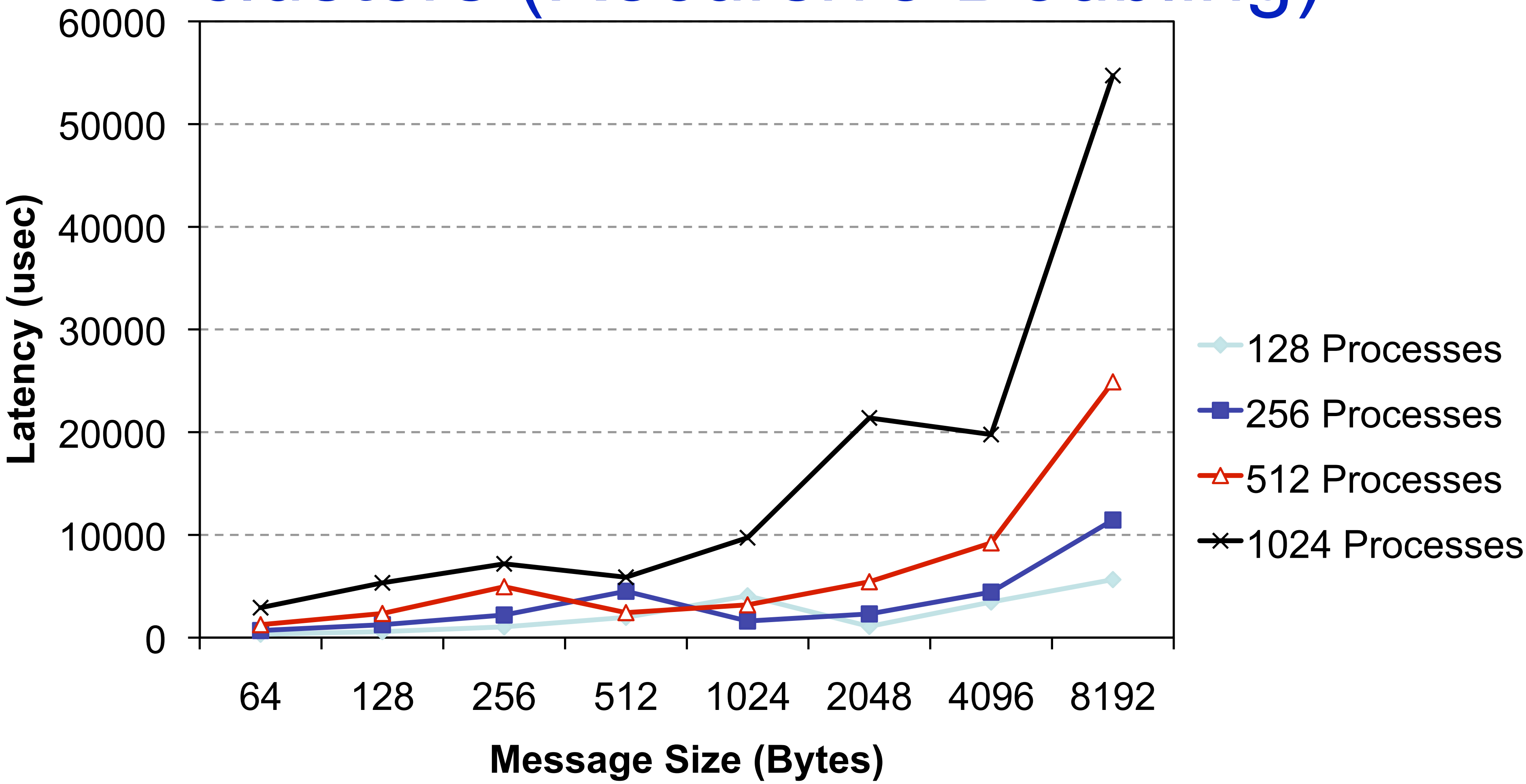
Recursive Doubling (RD) scales poorly with increasing core counts

Scaling on Multi-cores : Ring Algorithm



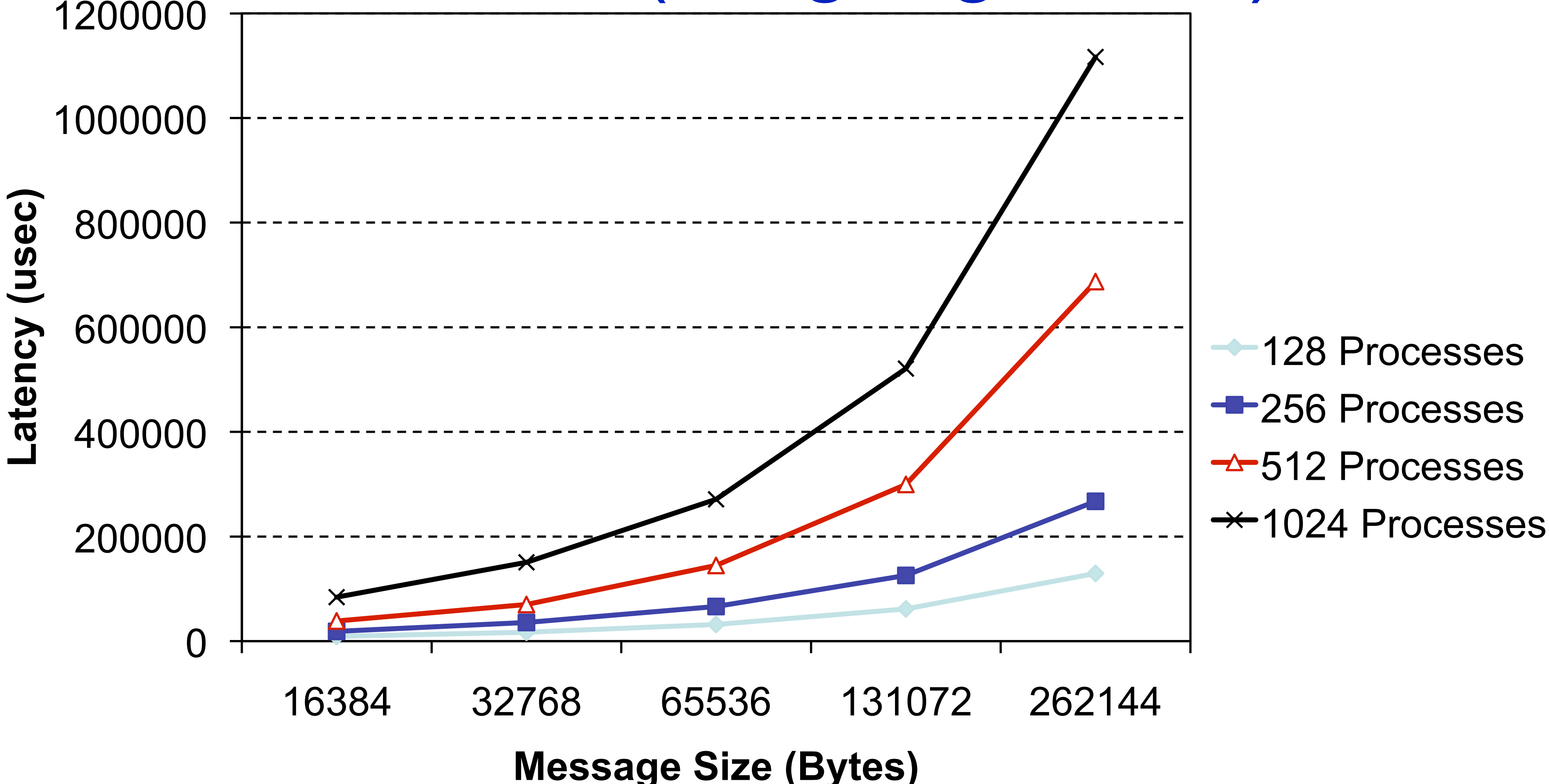
Ring Algorithm scales as expected with increasing core counts

Scaling on Large Scale Multi-core clusters (Recursive Doubling)



Recursive Doubling (RD) scales poorly for large system size

Scaling on Large Scale Multi-core clusters (Ring Algorithm)



Ring Algorithm scales as expected for large system sizes

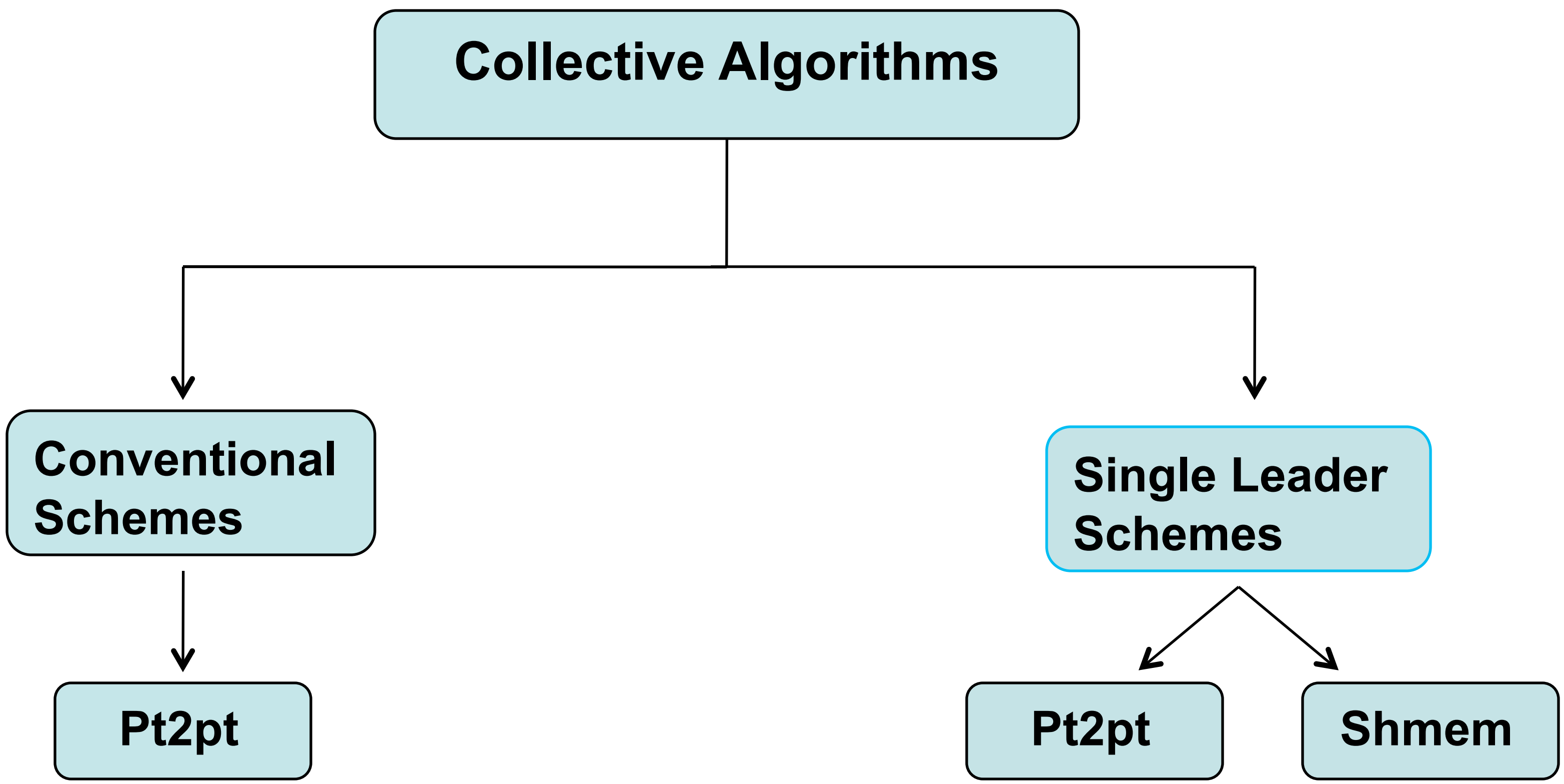
Problem Statement

- Is it possible to design an algorithm to :
- be Multi-core and NUMA aware to achieve better performance and scalability as core-counts and system sizes increase?
- fully exploit the differential memory access costs in NUMA based Multi-core systems?

Outline

- Introduction and Background
- Motivation
- Related Work
- Multi-Leader based Algorithms
- Experimental evaluation
- Conclusions and Future Work

Collective Design Framework



Existing Multi-core aware Algorithms

- Single Leader approaches :

  Aggregation – Distribution .

  Step 1 : Data aggregation at the leader on each node

  Step 2 : Inter leader exchanges

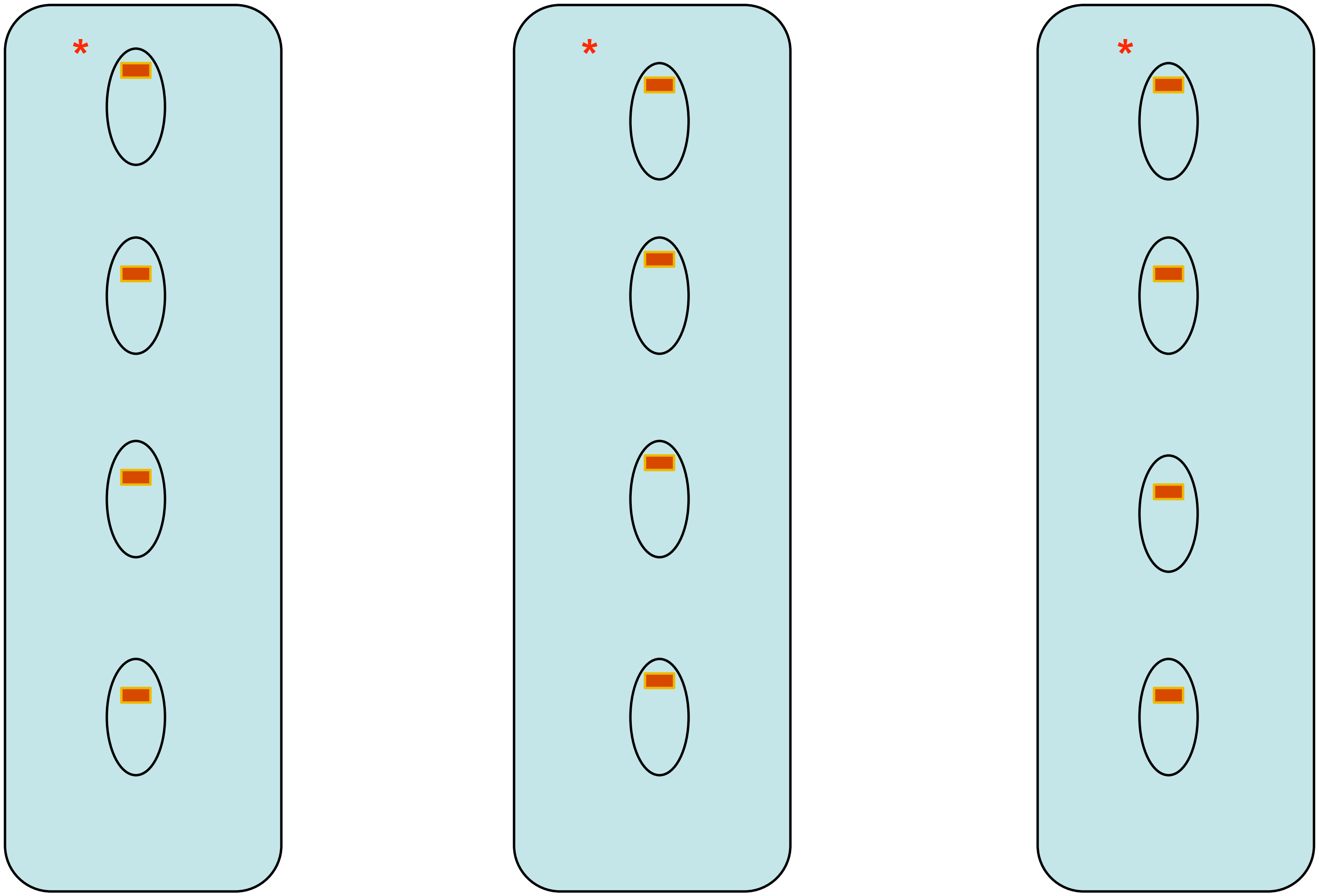
  Step 3 : Data distribution within each node

  Steps 1 and 3 are intra-node operations.

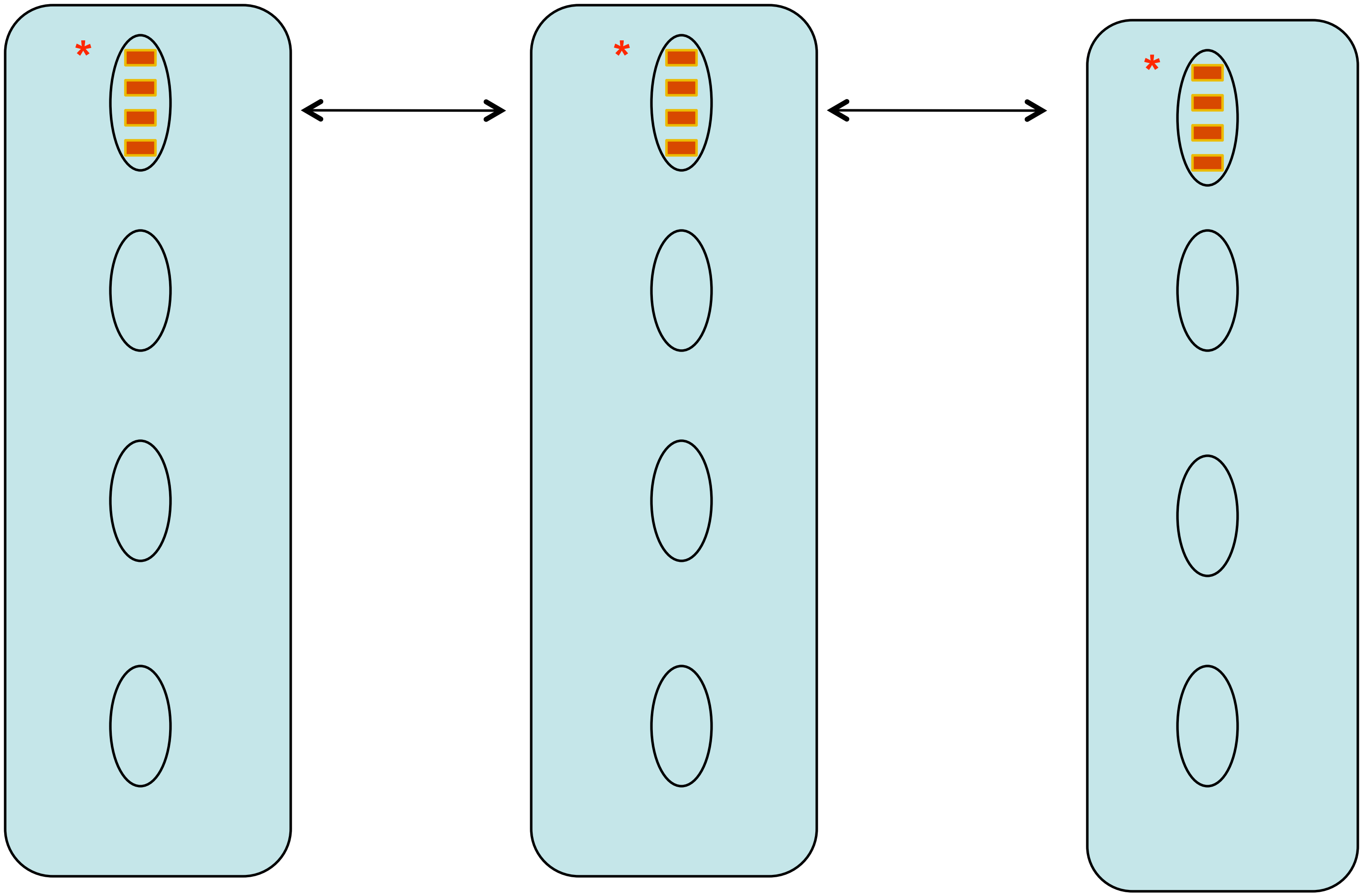
  → Point-to-point MPI calls

  → Shared memory buffer visible to all the processes within a node

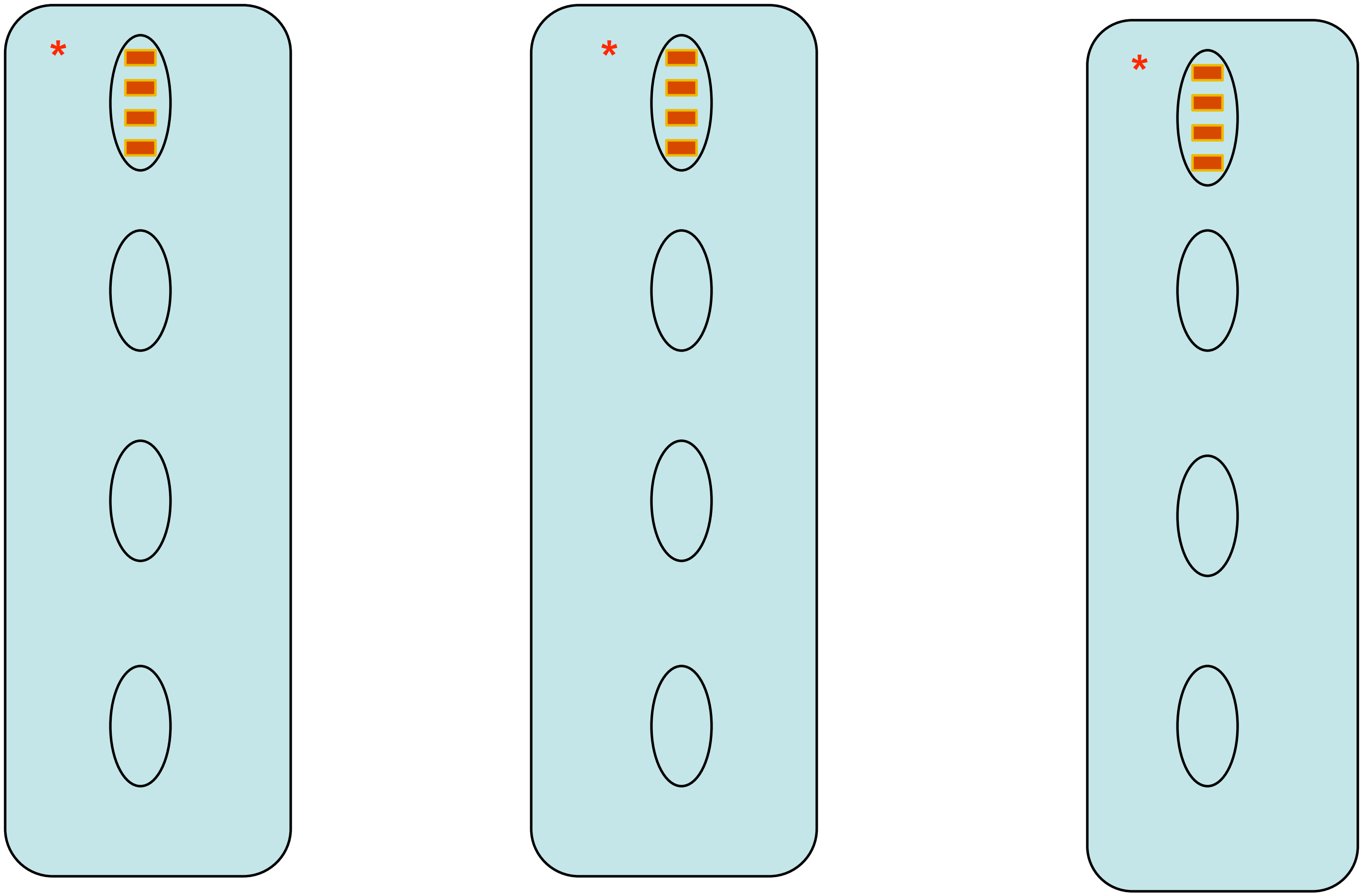
Single Leader Algorithms : Step1 intra-node (pt2pt)



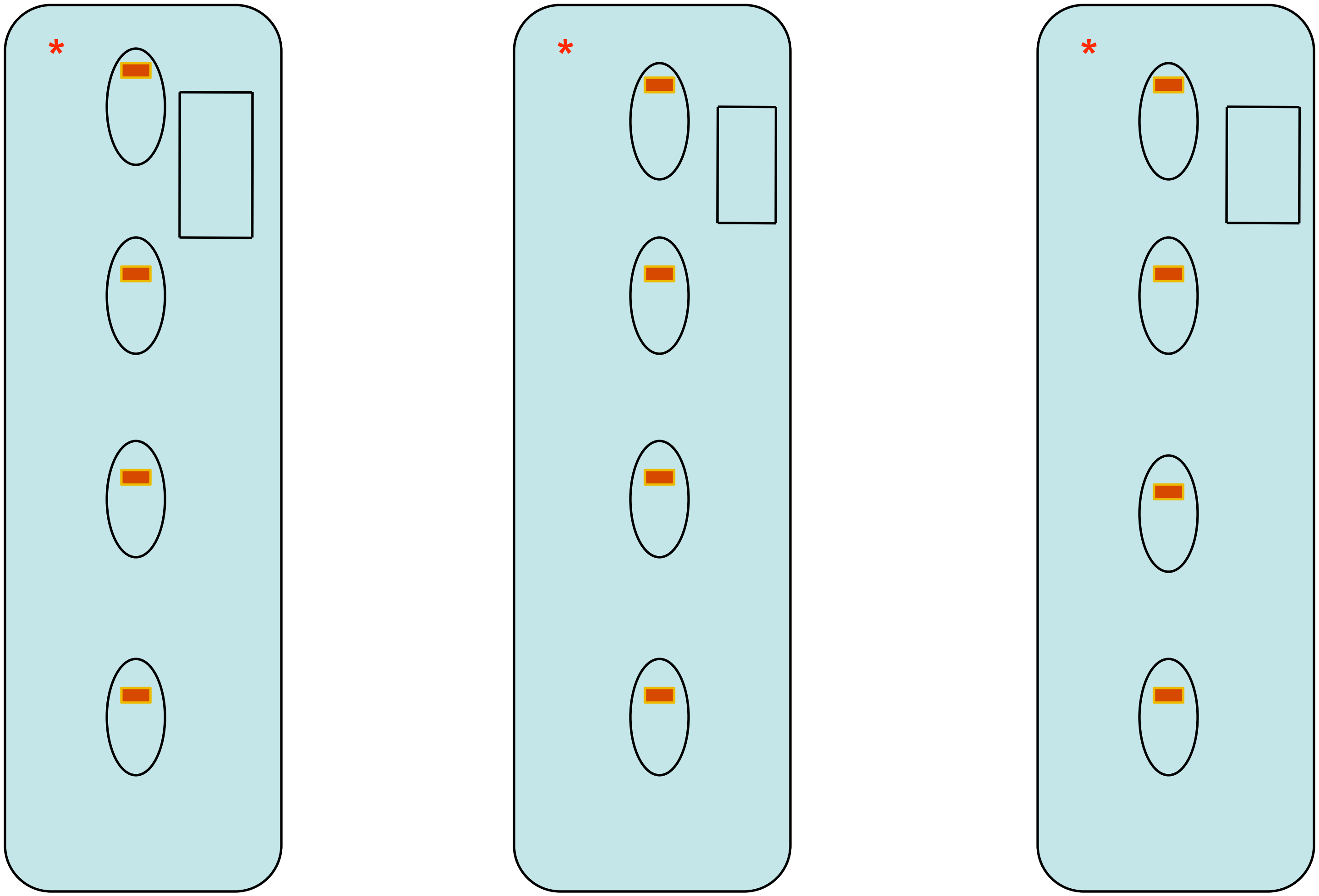
Single Leader Algorithms : Step2 inter-node (pt2pt)



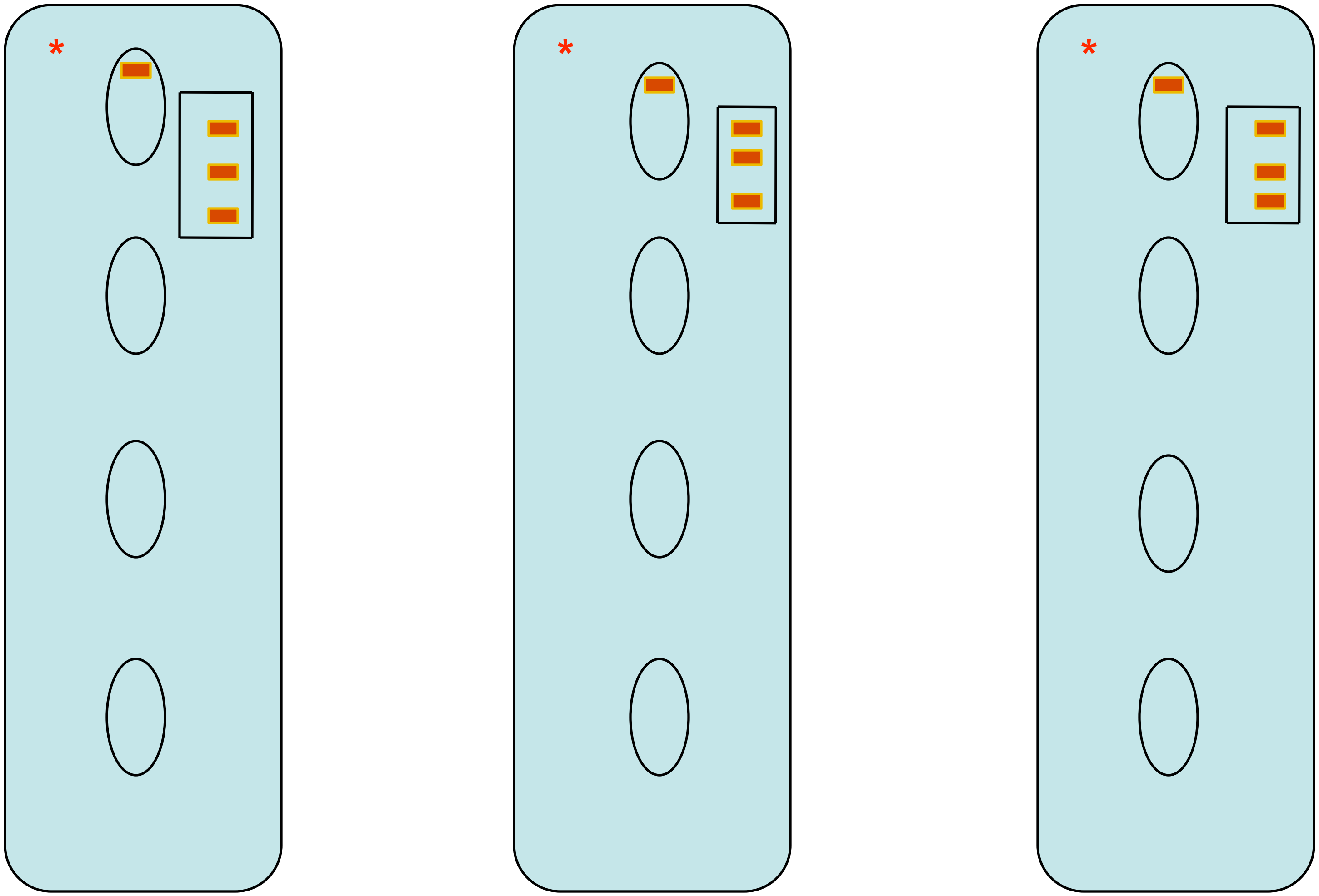
Single Leader Algorithms : Step 3 intra-node (pt2pt)



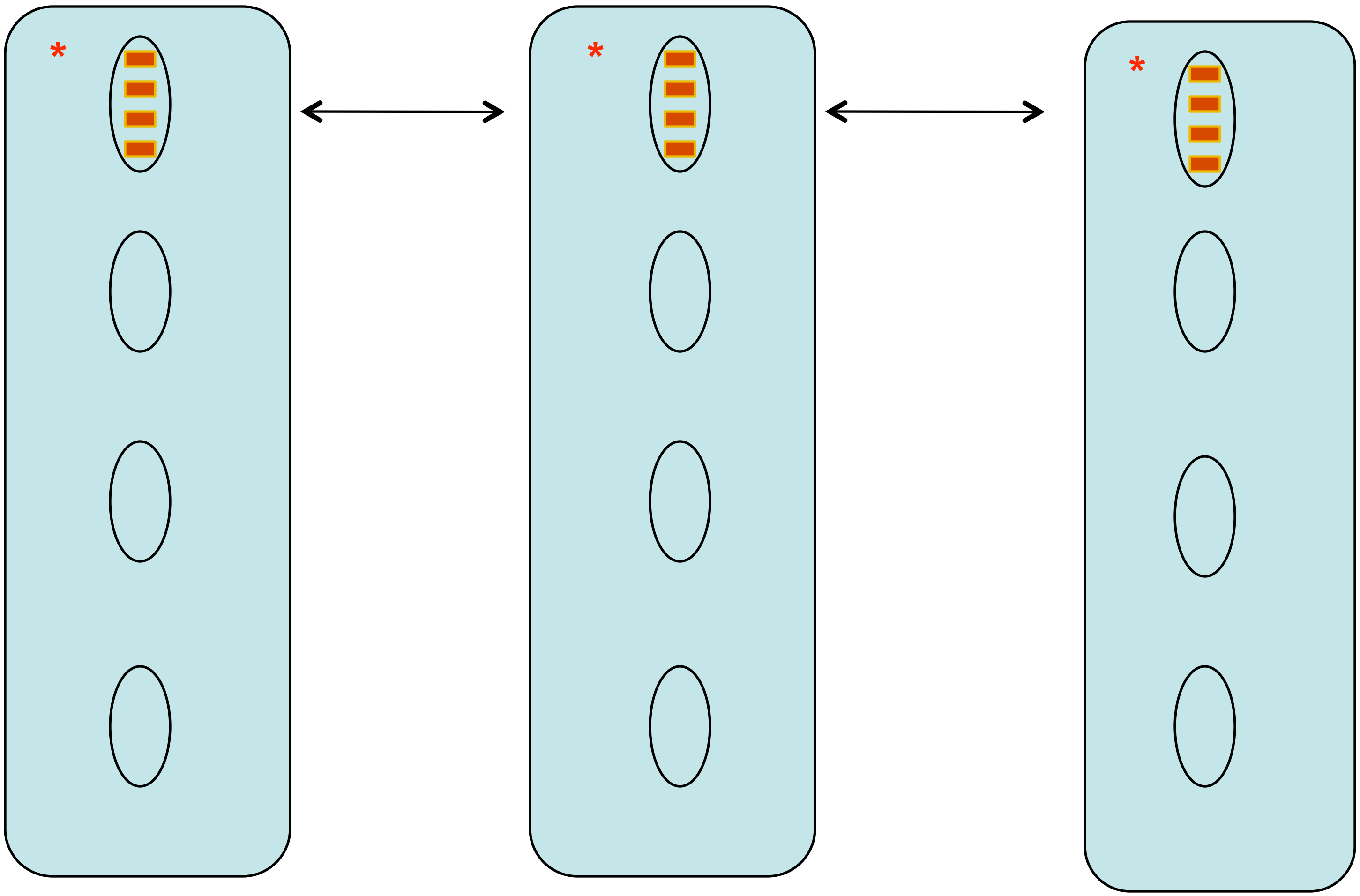
Single Leader Algorithms : Step1 intra-node (shmem)



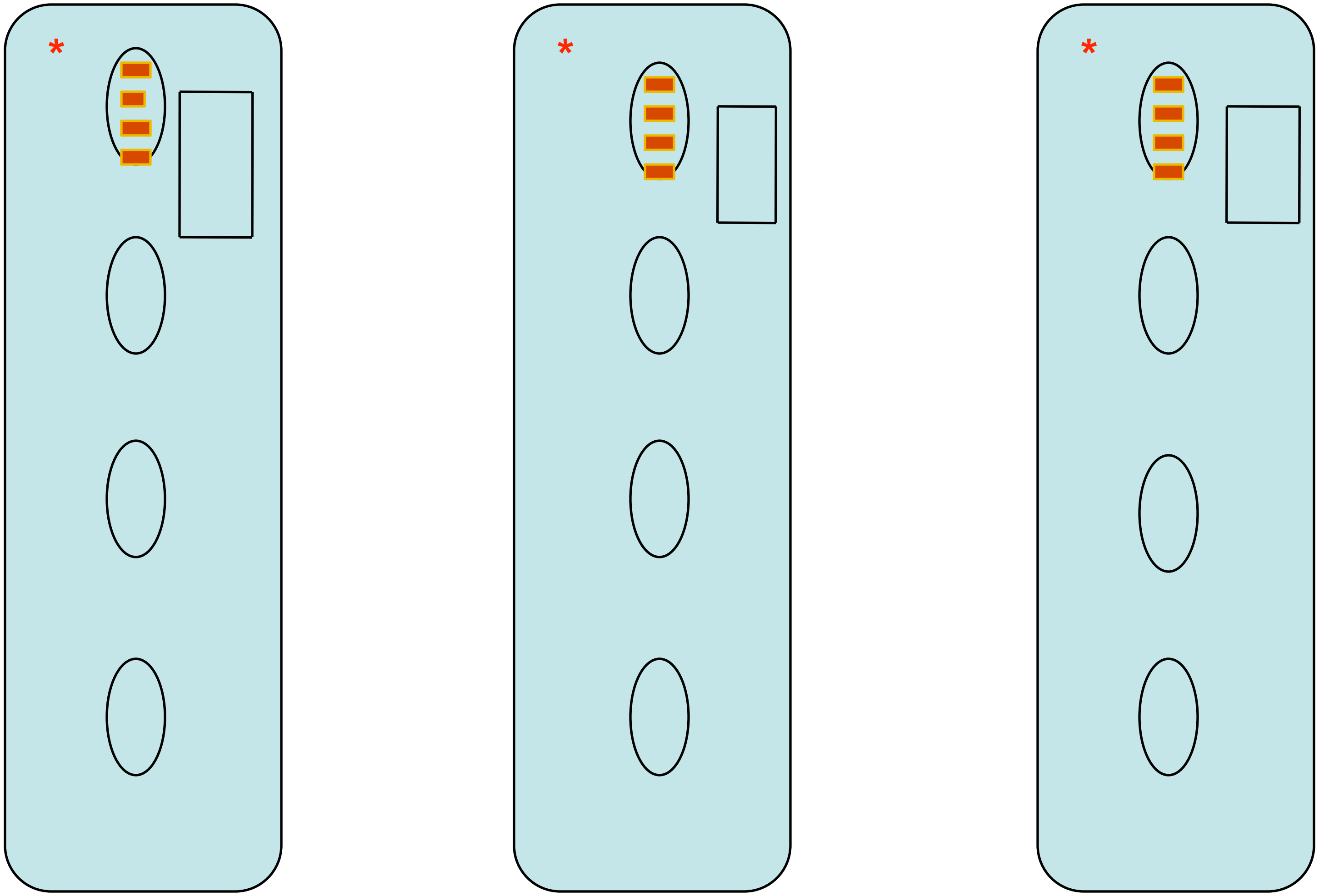
Single Leader Algorithms : Step1 intra-node (shmem)



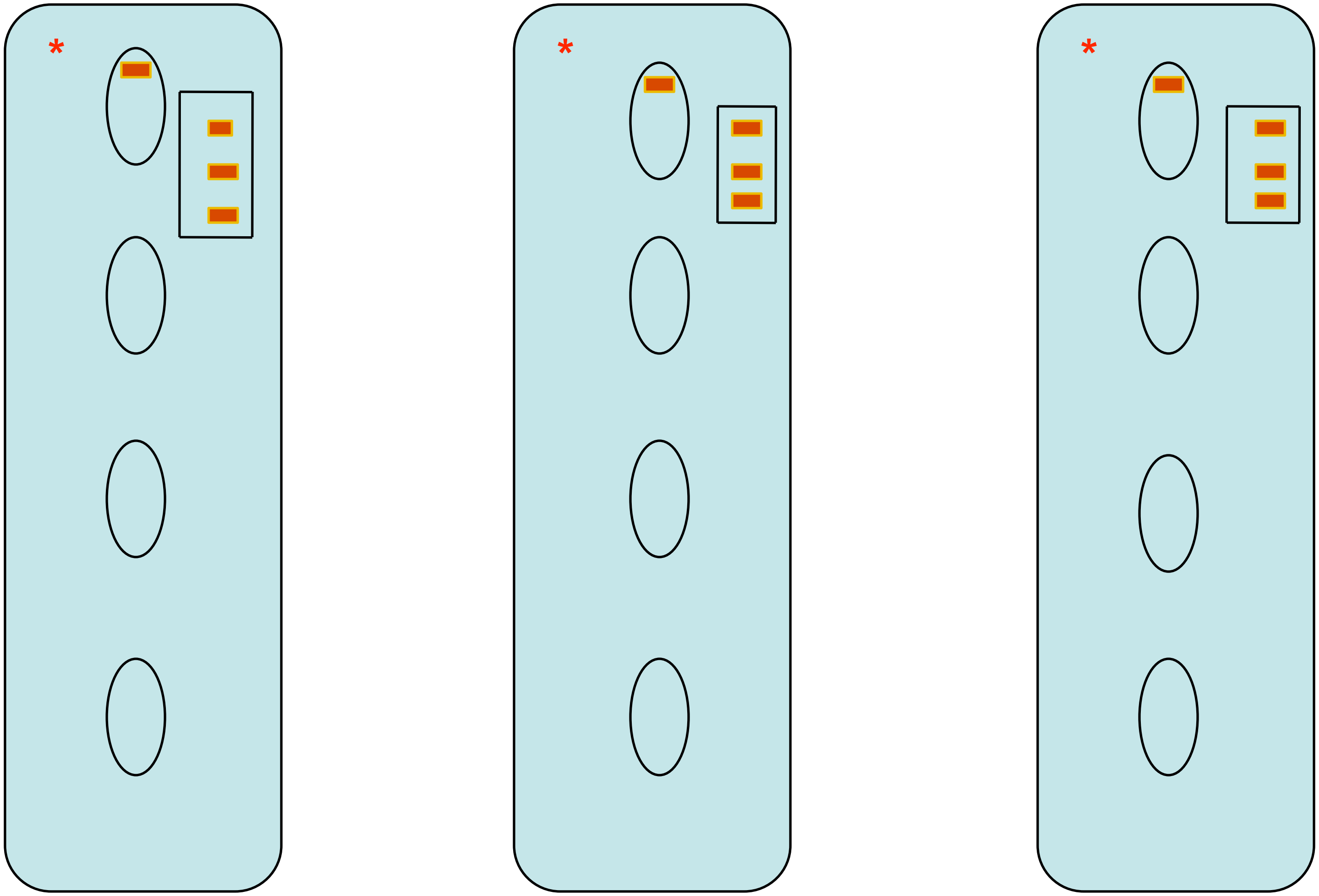
Single Leader Algorithms : Step2 inter-node (pt2pt)



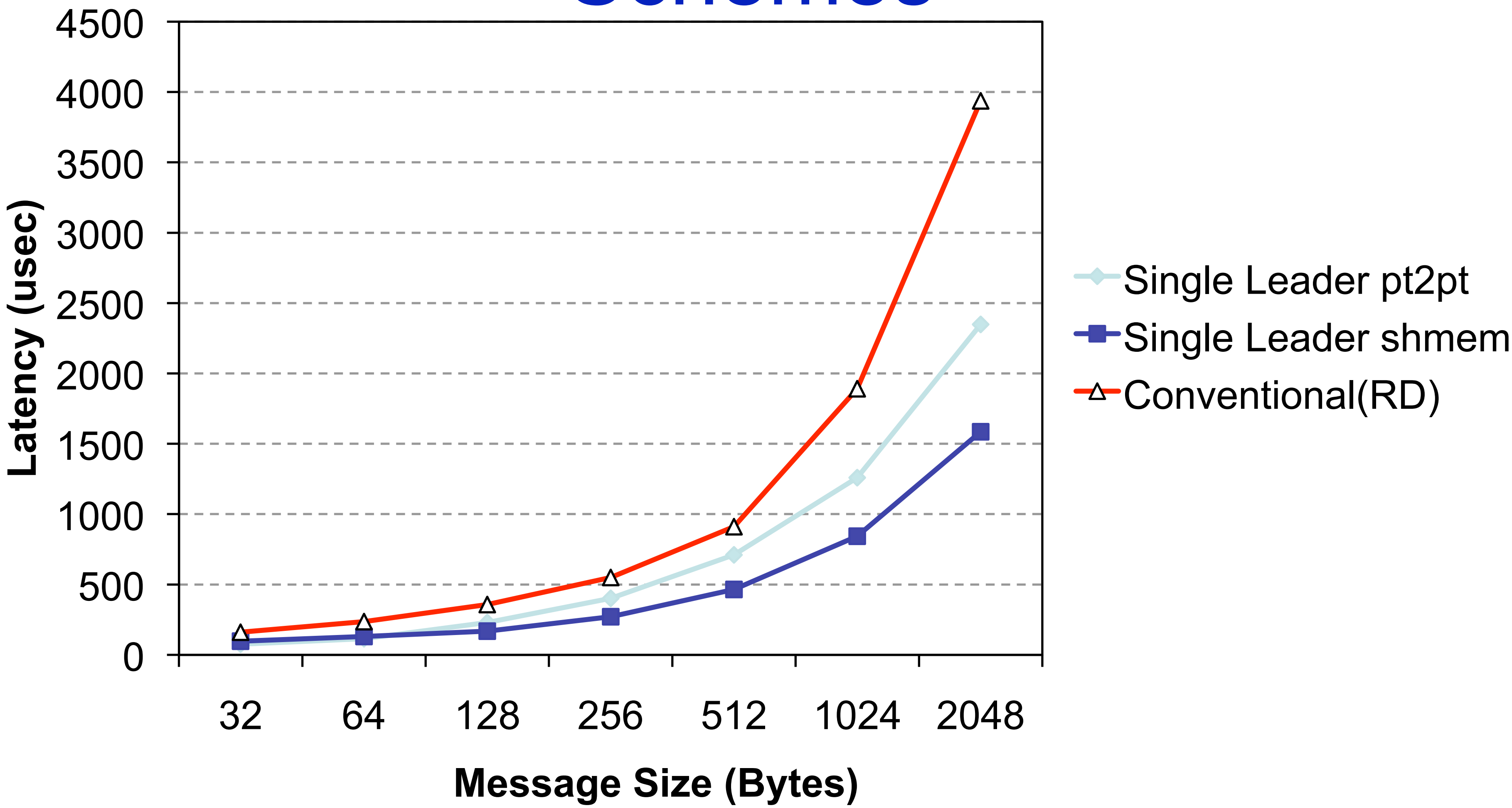
Single Leader Algorithms : Step 3 intra-node (shmem)



Single Leader Algorithms : Step3 intra-node (shmem)

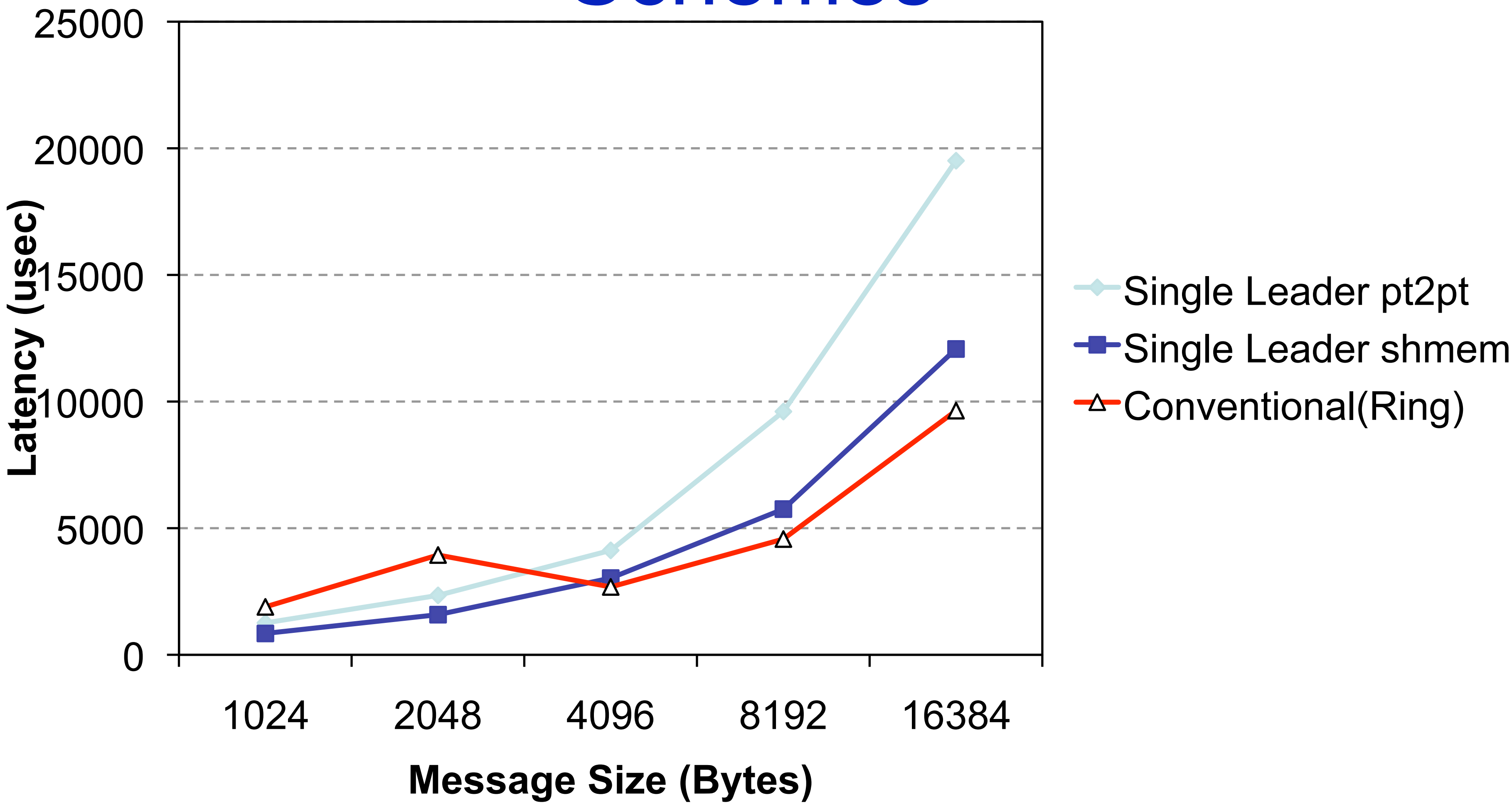


Performance of Single Leader Schemes



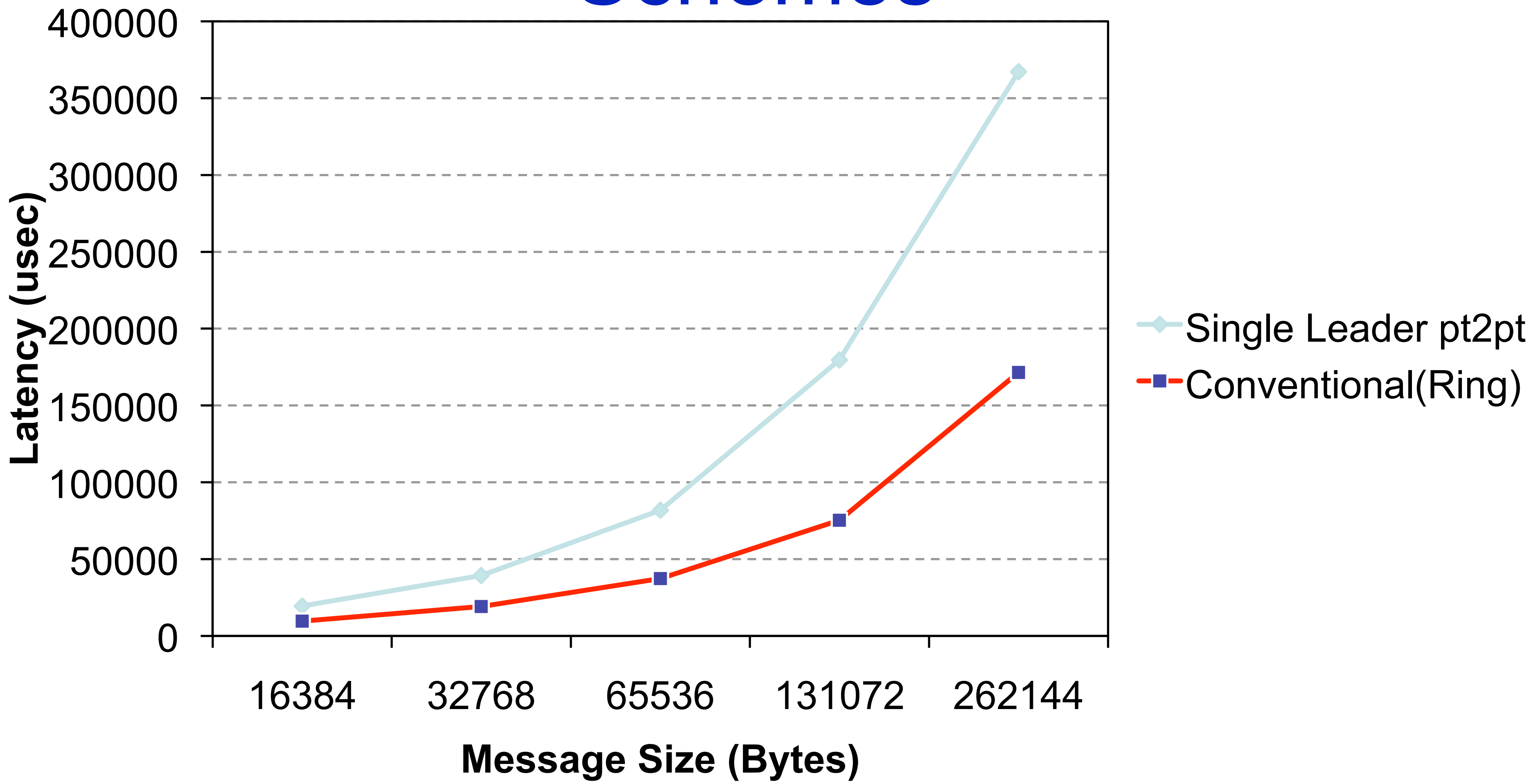
Single Leader schemes show potential for improvement

Performance of Single Leader Schemes



Conventional Ring Algorithm performs better for larger messages

Performance of Single Leader Schemes

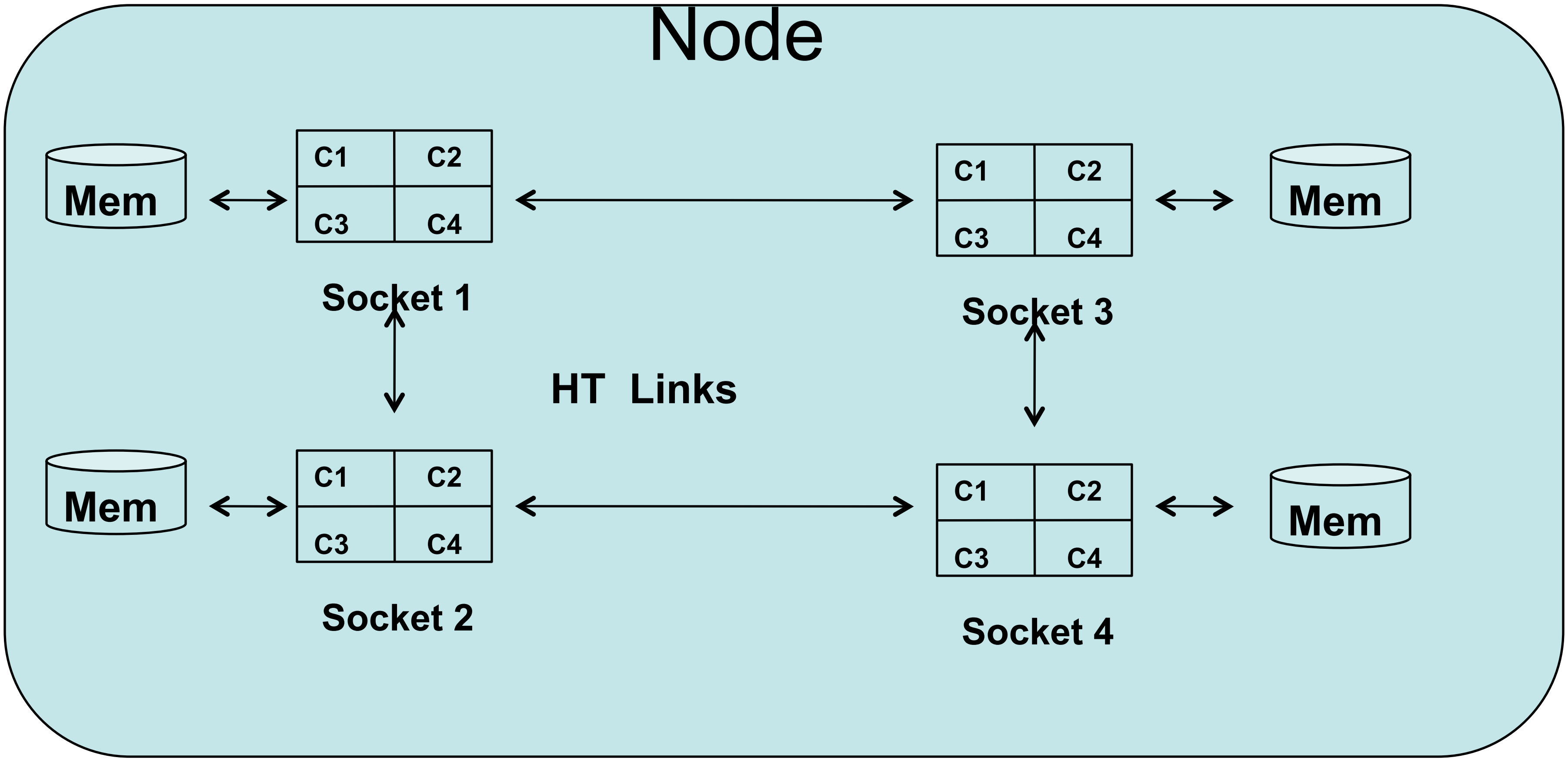


Conventional Ring Algorithm performs better for larger messages

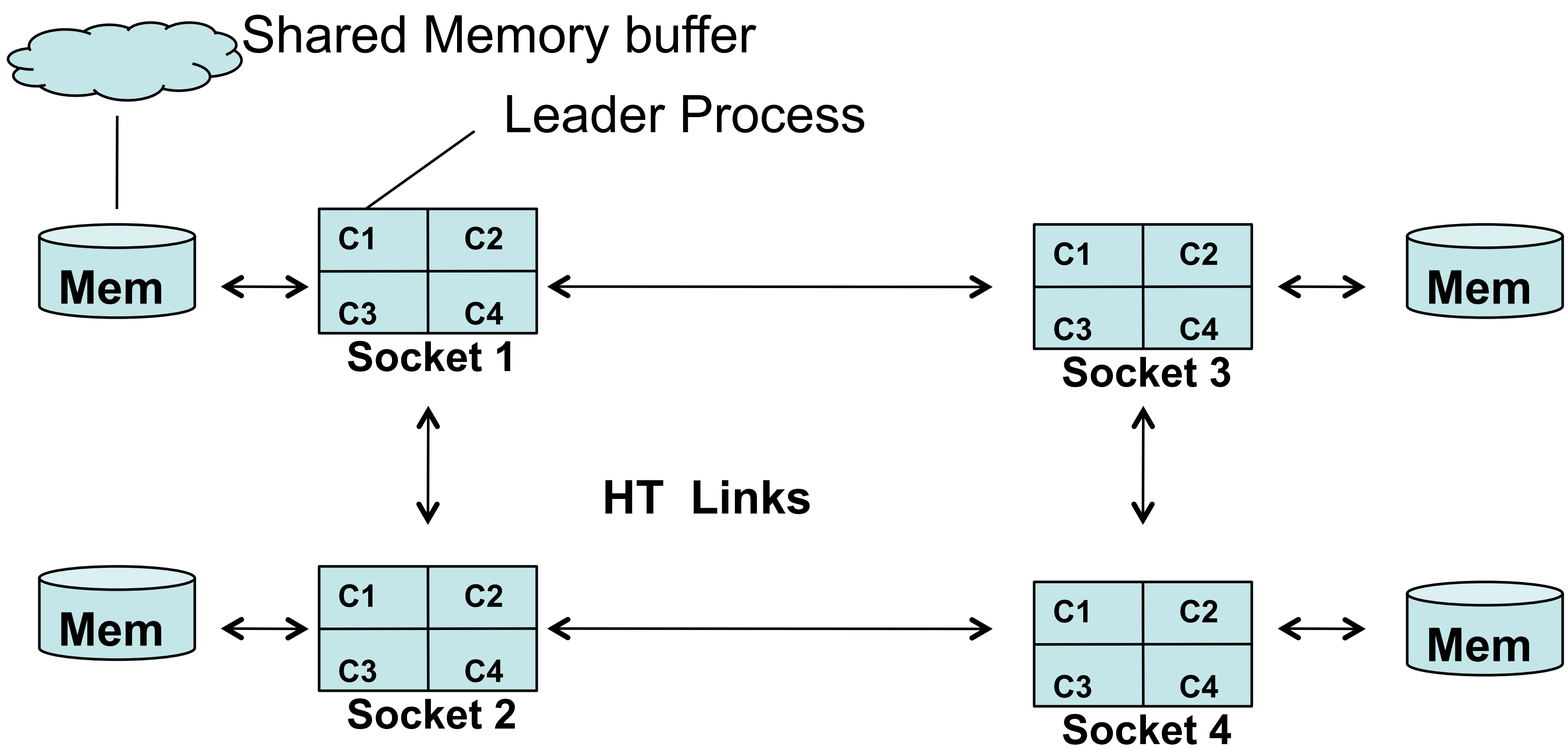
Outline

- Introduction and Background
- Motivation
- Related Work
- Multi-Leader based Algorithms
- Experimental evaluation
- Conclusions and Future Work

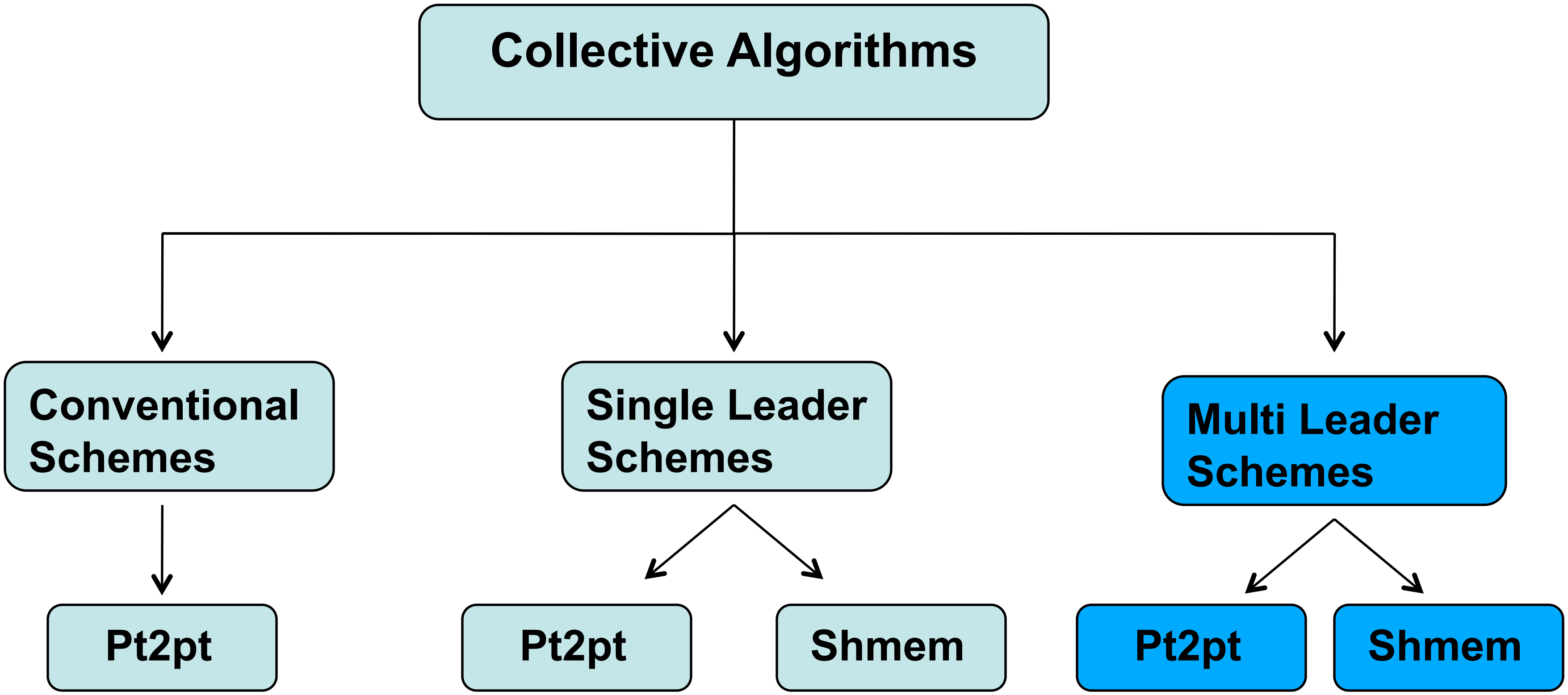
AMD Barcelona Architecture



Single Leader algorithms on the AMD Barcelona Architecture



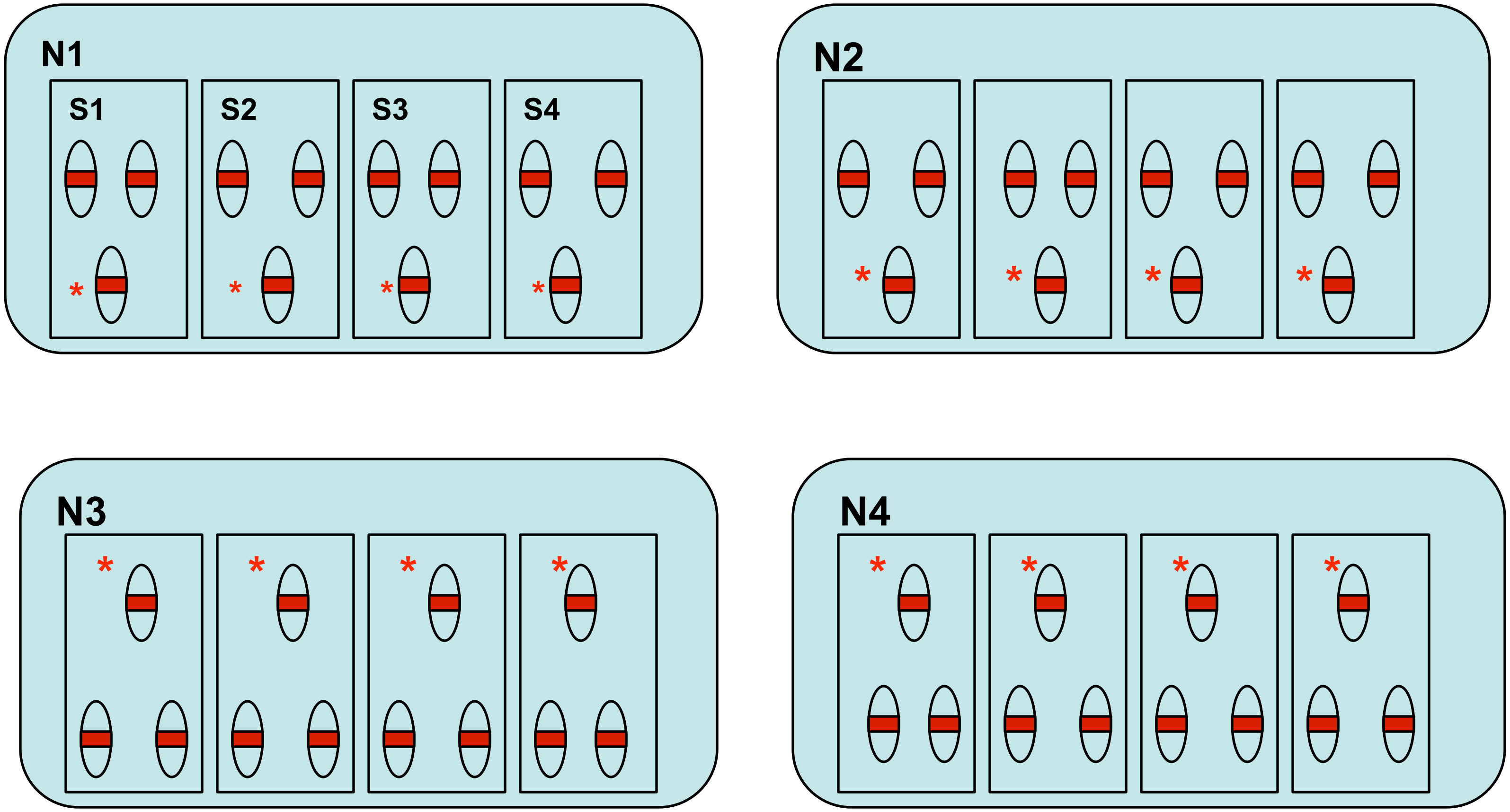
Proposed Collective Design Framework



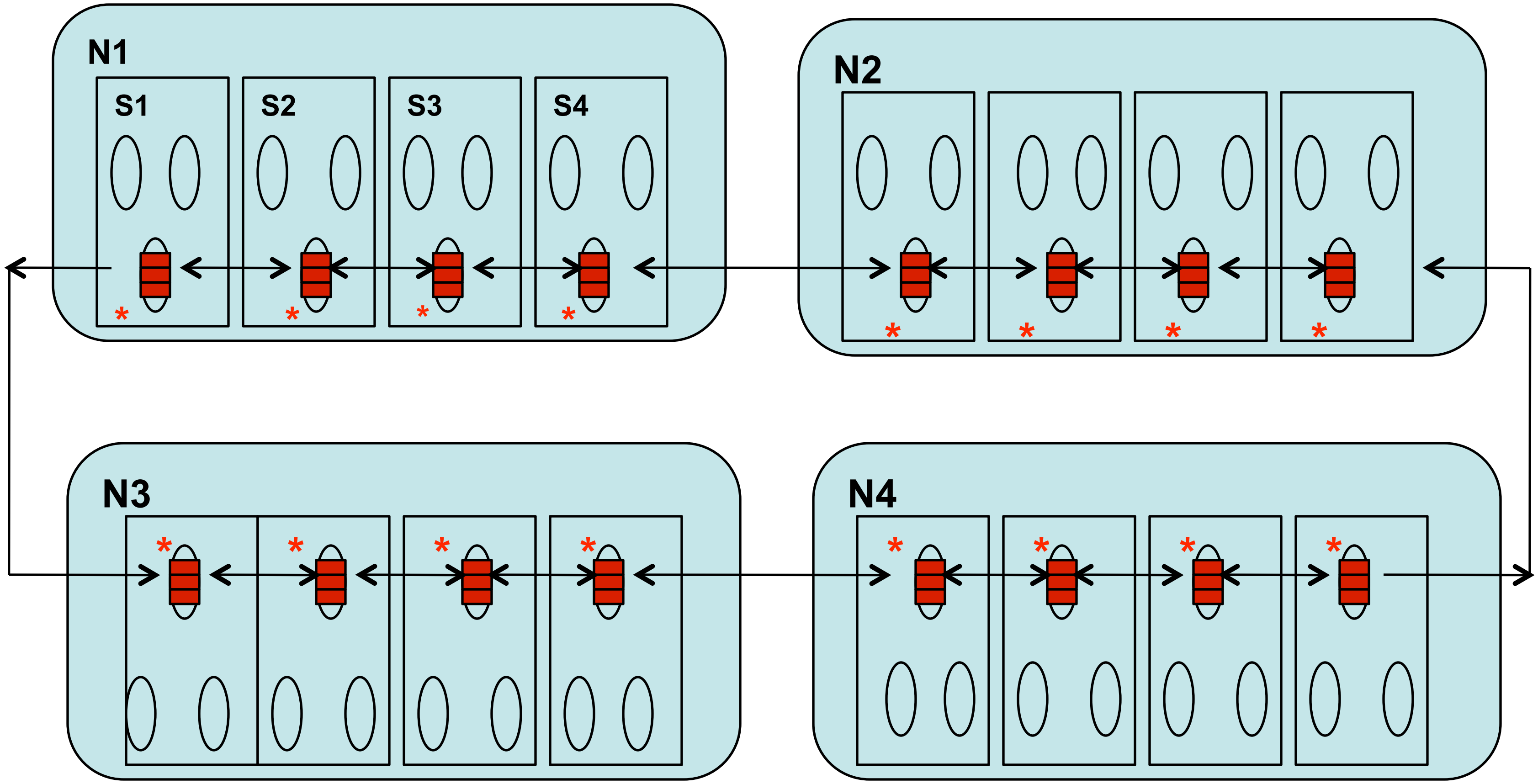
Multi-Leader based Algorithms

- Number of leader processes per node
- Intra-socket and Inter-leader exchange algorithms.

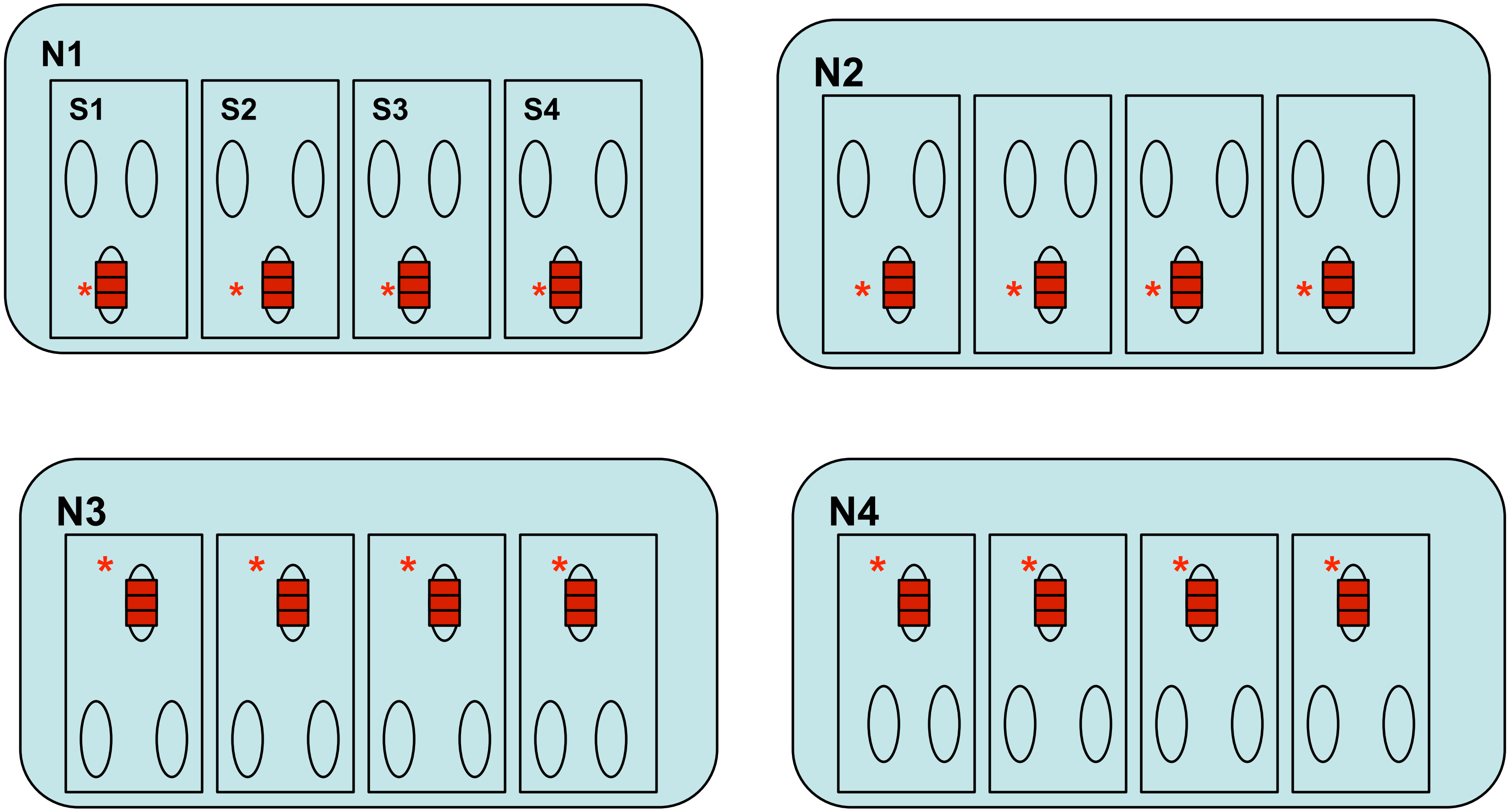
Multi-Leader based Algorithms(Step 1)



Multi-Leader based Algorithms(Step 2)



Multi-Leader based Algorithms(Step 3)



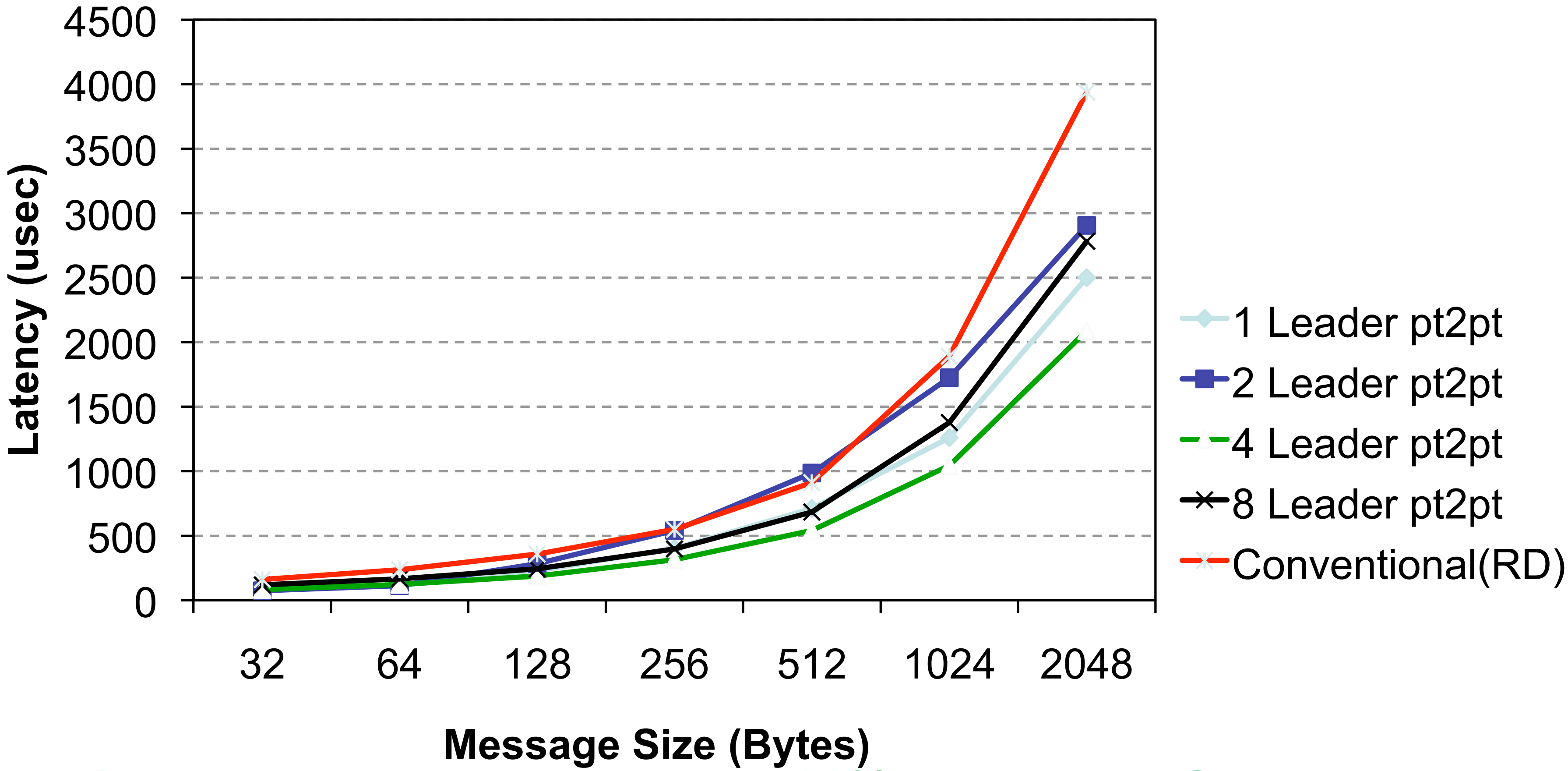
Outline

- Introduction and Background
- Motivation
- Related Work
- Multi-Leader based Algorithms
- Experimental evaluation
- Conclusions and Future Work

Experimental Test-bed

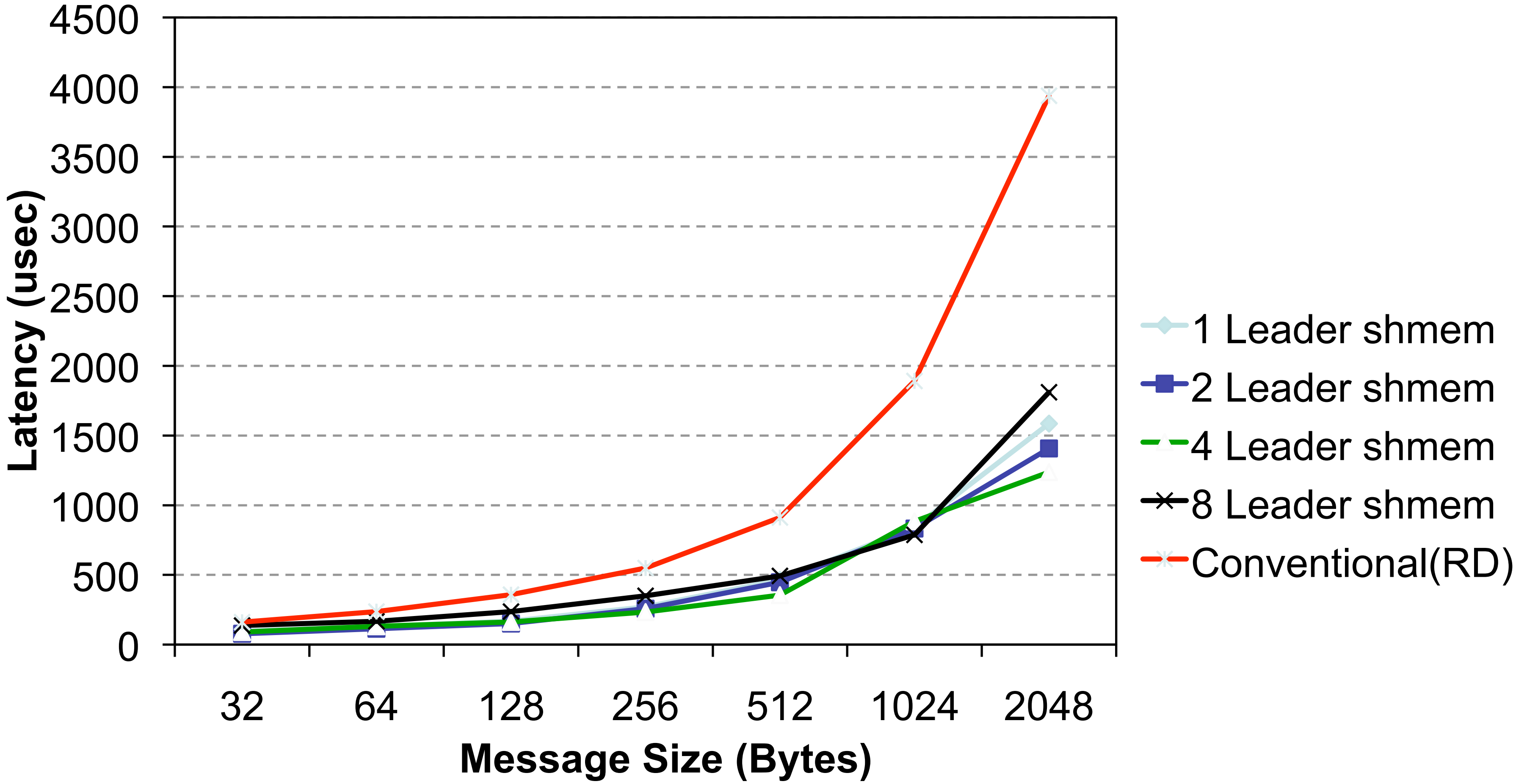
- Each node of our testbed has 16 AMD Opteron 1.95 Ghz processors with 512 KB L2 cache. We used 8 nodes.
- Each node has 16 GB memory and PCI-Express bus, 2 MT25418 DDR HCAs with PCI-Ex interfaces.
- 24-port Mellanox switch is used to connect all the nodes.
- RedHat Enterprise Linux Server 5.

Performance of Multi-Leader pt2pt



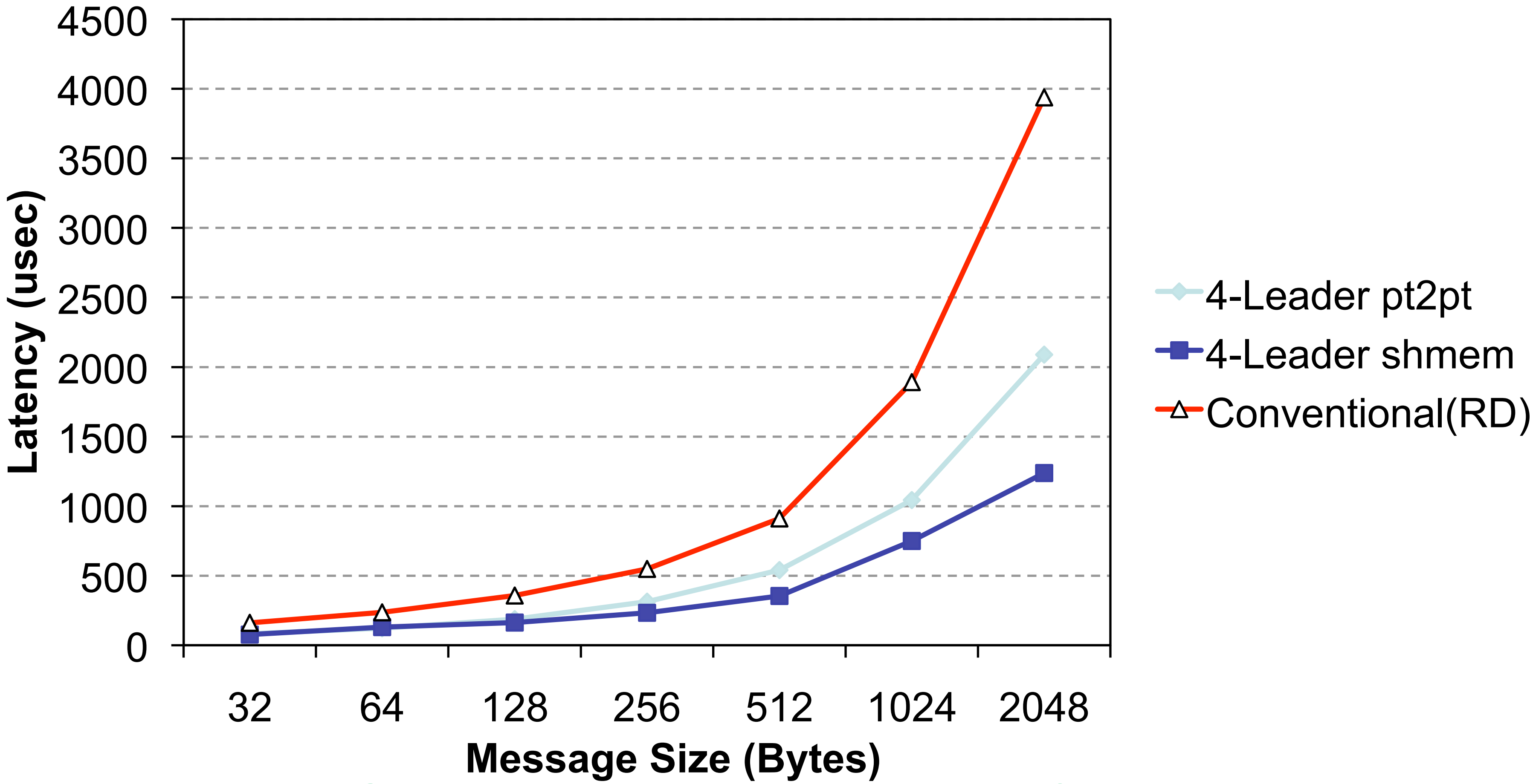
4-Leader scheme does about 20% better than Single Leader scheme and 50% better than RD

Performance of Multi-leader : Shared Memory



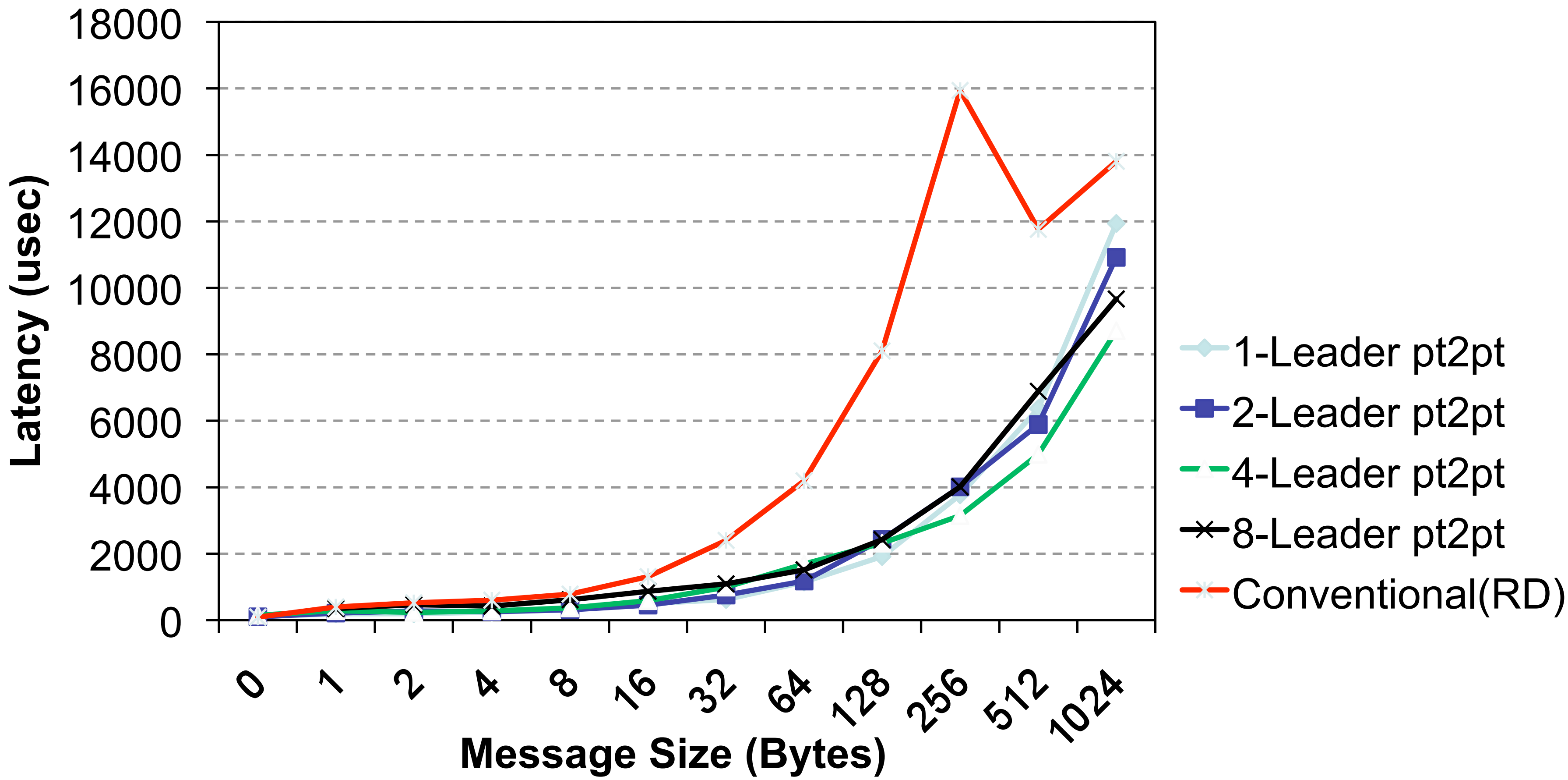
4-Leader scheme performs better than 1-Leader scheme by about 25% and 70% better than RD

Performance of Multi-Leader Schemes (pt2pt Vs Shared Memory)



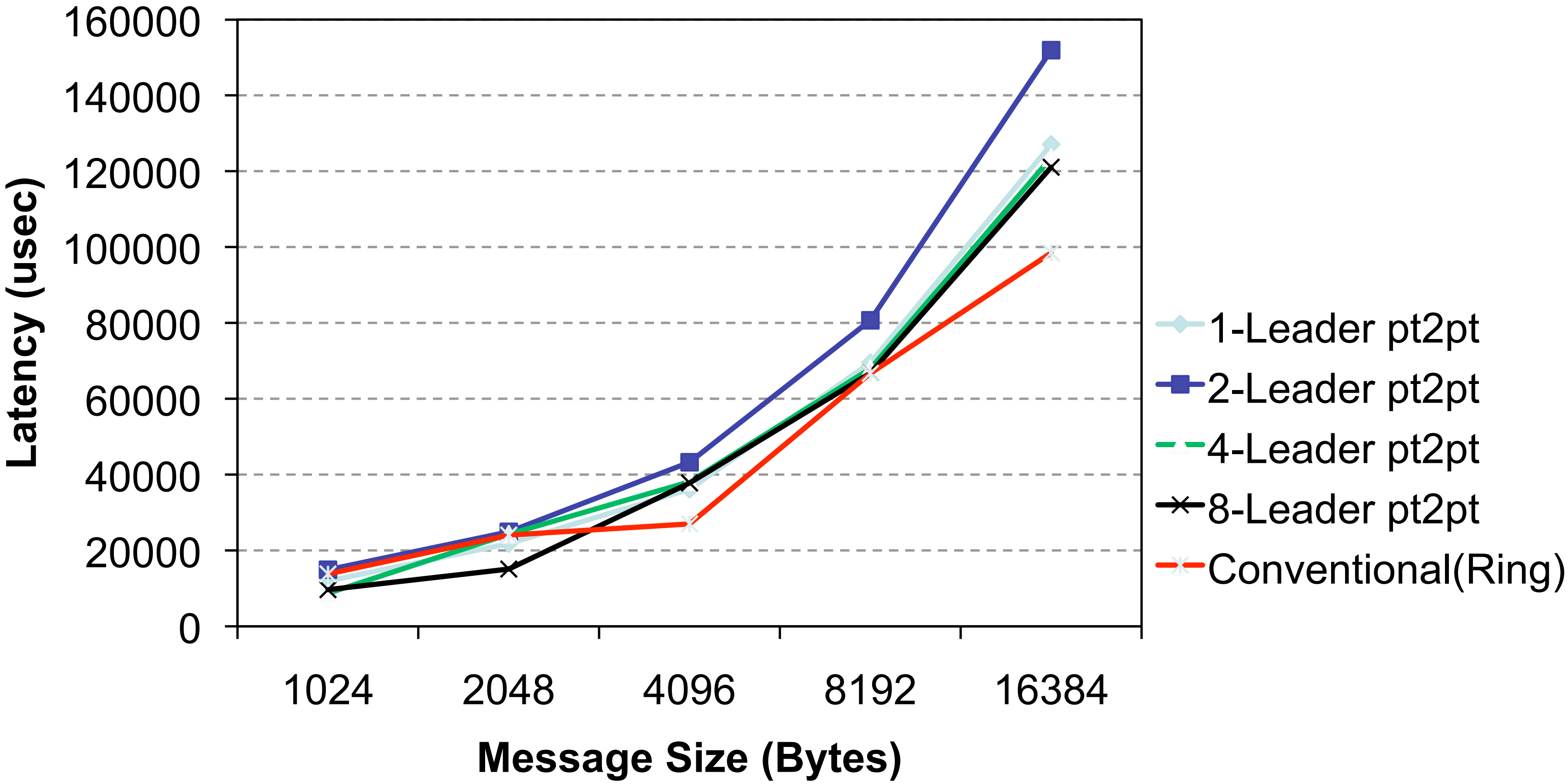
4-Leader Shared Memory approach performs better than 4-Leader Point-to-point scheme by about 40%

Performance of Multi-Leader schemes on large scale Multi-cores



4-Leader Point-to-point scheme outperforms the recursive doubling method on 1024 processes on the TACC Ranger

Performance of Multi-Leader schemes on large scale Multi-cores



Conventional Ring Algorithm performs better for larger messages

Proposed Unified Scheme

	Intra-Node Mechanism	Inter-Leader Algorithm	Design
Small Messages	Point-to-Point	Recursive Doubling	Hierarchical
Medium Messages	Shared Memory	Recursive / Ring	Hierarchical
Large Messages	Point-to-Point	Ring	Conventional

Outline

- Introduction and Background
- Motivation
- Related Work
- Multi-Leader based Algorithms
- Experimental evaluation
- Conclusions and Future Work

Conclusions & Future Work

- Single Leader schemes are limited by scalability and memory contention. Proposed Multi-Leader schemes perform show significant performance benefits.
- *Future work:*

  - Examine the benefits of using kernel based zero-copy intra-node exchanges for large messages.

  - A frame-work that can choose leaders in an optimal manner for emerging multi-core systems.

  - Evaluate the impact of such designs on real-world applications.

MVAPICH

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

Thank you !

{kandalla, subramon, santhana, koop, panda}@cse.ohio-state.edu