Accelerating MPI Message Matching and Reduction Collectives For Multi-/Many-core Architectures

M. Bayatpour, S. Chakraborty , H. Subramoni, X. Lu, and D. K. Panda

Department of Computer Science and Engineering
The Ohio State University

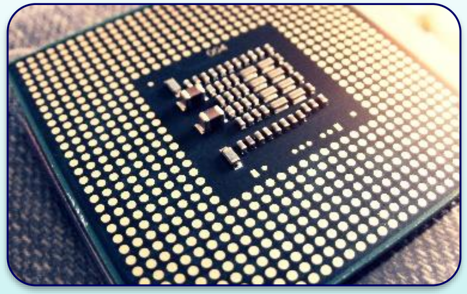
MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Adaptive and Dynamic Design for MPI Tag Matching

M. Bayatpour, H. Subramoni, S. Chakraborty and D. K. Panda

Department of Computer Science and Engineering
The Ohio State University

Current Trends in HPC



Supercomputing systems scaling rapidly

- Multi- and Many-core architectures
- High-performance Interconnects

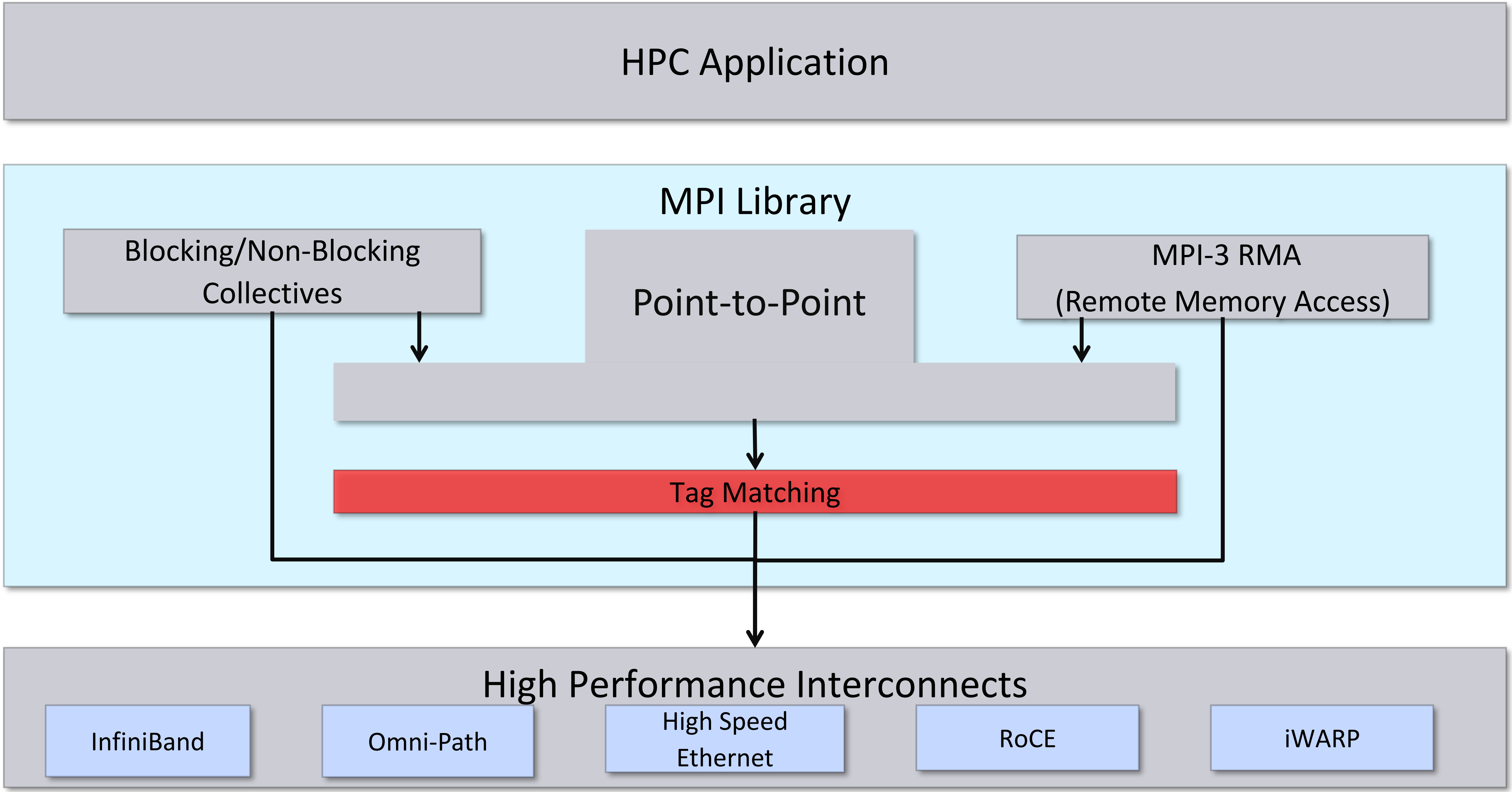
InfiniBand and Omni-Path are popular HPC Interconnects

- Low-latency and High-bandwidth
- 192 systems (39%) in Jun'17 Top500 use IB

MPI used by vast majority of HPC applications

- Helping applications scale to thousands of cores
- Large systems exposing new scalability issues

Components of an MPI Library



MPI Tag Matching 101

- On the receiver side, one needs to match the incoming message with the message that was posted by receiver
- Three parameters should match
  - Context id, Source Rank, Tag
  - Wildcards (MPI_ANY_SRC, MPI_ANY_TAG) introduce additional complexity
- Two kinds of the queues are involved in the receiver side
  - Posted queue
  - Unexpected queue

Search Time Analysis of the Default Double Linked List Design

- Most MPI libraries use double linked list for unexpected and posted queues
- Message to be removed could be in any position of the queue
  - Removal time in the best case is O(1) and in the average case is linear O(N)
- Tag matching is in the critical path for point-to-point based operations
- Number of the processes in a job is increasing
  - Future extreme-scale systems are expected to have millions of cores*
  - Multithreaded programming models
- All can push the search functions to go deeper in the lists
  - Impose significant overhead on the performance

* Thakur R, Balaji P, Buntinas D, Goodell D, Gropp W, Hoefler T, Kumar S, Lusk E, Träff JL. MPI at Exascale. Proceedings of SciDAC. 2010 Jul;2:14-35.

Proposed Adaptive Design

- Based on the Bin-based and default simple double linked list scheme
- Three phases
  - Starts with the default design
  - Observes the communication pattern for each process during the runtime
  - If all the conditions are held, it begins to convert the default scheme to the Bin-based scheme
- Each process can have its own scheme
  - Some may stay at the default scheme, some may need to convert to bin-based scheme

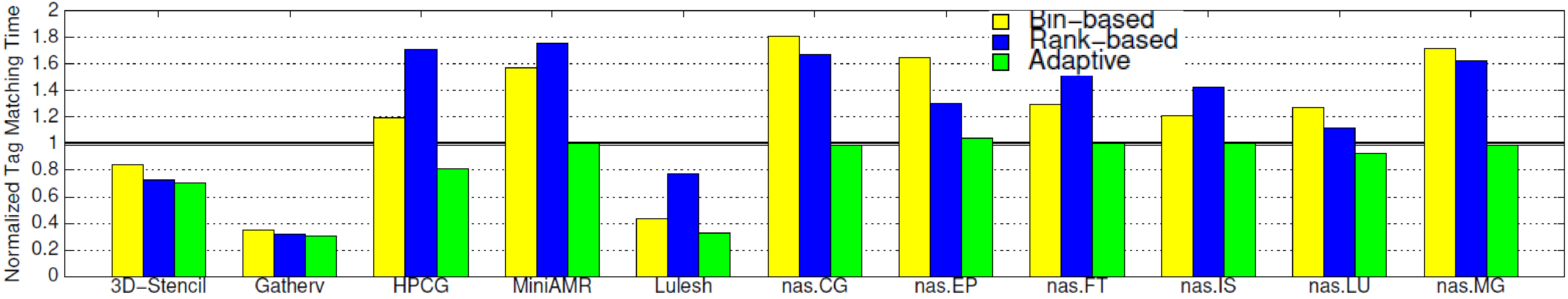
Proposed Adaptive Design (Cont'd)

- For each of the posted and unexpected queues, we consider the following thresholds
  - Number of the calls to the tag matching functions in the library (CALLS_NUM)
  - The average number of queue look-up attempts per CALLS_NUM (MACTCH_ATTMPS)
- Each process maintains both during the runtime
- If both thresholds are crossed
  - Adaptive design changes from the double linked list scheme to the bin-based scheme

Proposed Adaptive Design (Cont'd)

- Currently, conversion is one way from default to bin-based scheme and may occur only one time through the entire runtime
- These thresholds are fixed through entire runtime and they are configurable
  - We have tuned them based on empirical analysis using OSU micro benchmarks
- We consider two possible sizes for NUM_BINS
  - ¼ JOB_SIZE and ½ JOB_SIZE
  - Based on MATCH_ATTMPS, we decide which one to choose

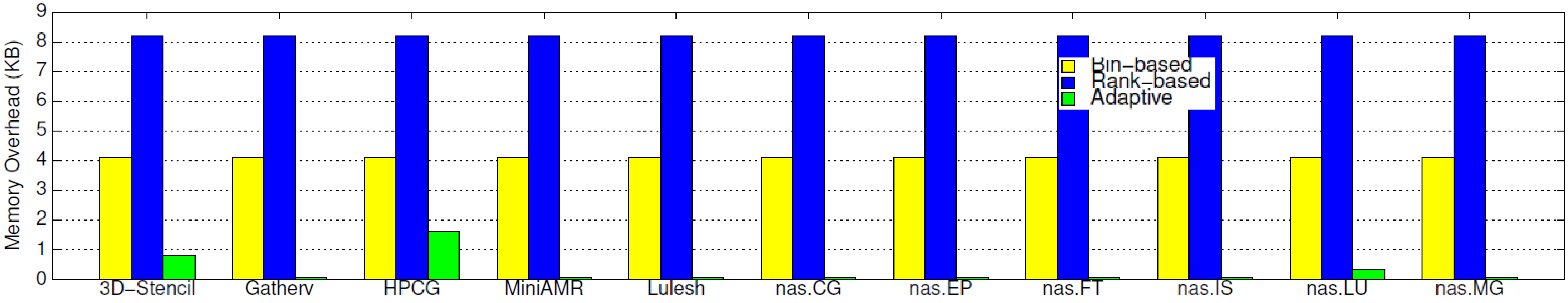
Summary of Tag Matching Performance



(b) Total Tag Matching Time, Normalized to Default (Lower is Better)

- Comparison of different designs/benchmarks at 512 processes on RI
- Adaptive design shows the best performance

Summary of Memory Consumed for Tag Matching



(a) Memory Overhead per Process, Compared to Default (Lower is Better)

- Comparison of different designs/ benchmarks at 512 processes on RI with default design
- Adaptive design shows minimal memory overhead

Scalable Reduction Collectives with Data Partitioning-based Multi-Leader Design

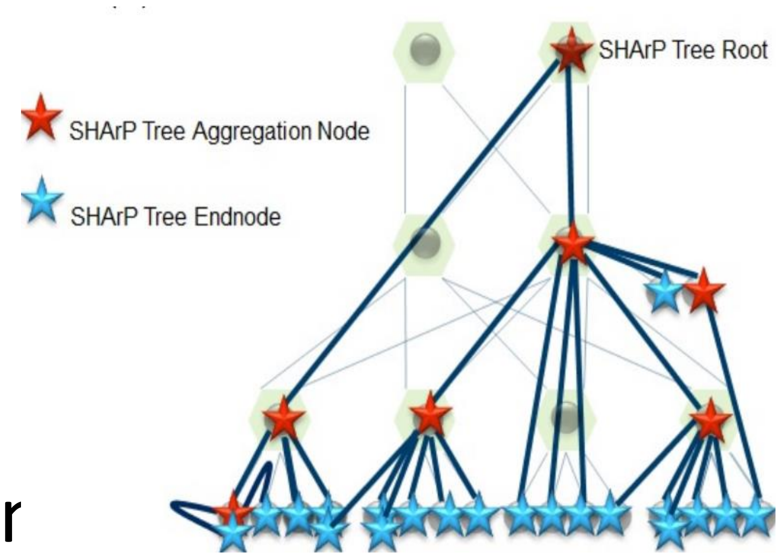
M. Bayatpour, S. Chakraborty , H. Subramoni, X. Lu, and D. K. Panda

Department of Computer Science and Engineering
The Ohio State University

MPI Reduction Collectives 101

- Convenient abstraction to implement group communication operations
- Widely used across various scientific domains
  - Owing to their ease of use and performance portability
- One of the most popular collective operations: MPI_Allreduce
  - 37% of communication time
- MPI_Allreduce reduces values from all processes and distribute the result back to all processes

Existing Designs for MPI_Allreduce



- Hierarchical strategy
  - A two-level approach
  - Intra-node reduction by root + inter-r
    - Computations are done by the root process of each node
  - QPI transfer to send everything to same socket

- Tree-based strategies
- Recursive Doubling
  - Based on point-to-point aggregation
    - High parallelism for computation
  - All the process are involved in computation
    - Pairs distance doubles after each step
    - Log (P*) steps

- Network offload based mechanism
  - Scalable Hierarchical Aggregation Protocol (SHArP*) by Mellanox
    - Management and execution of MPI operations in the network by using SHArP
    - Similar to hierarchical strategy

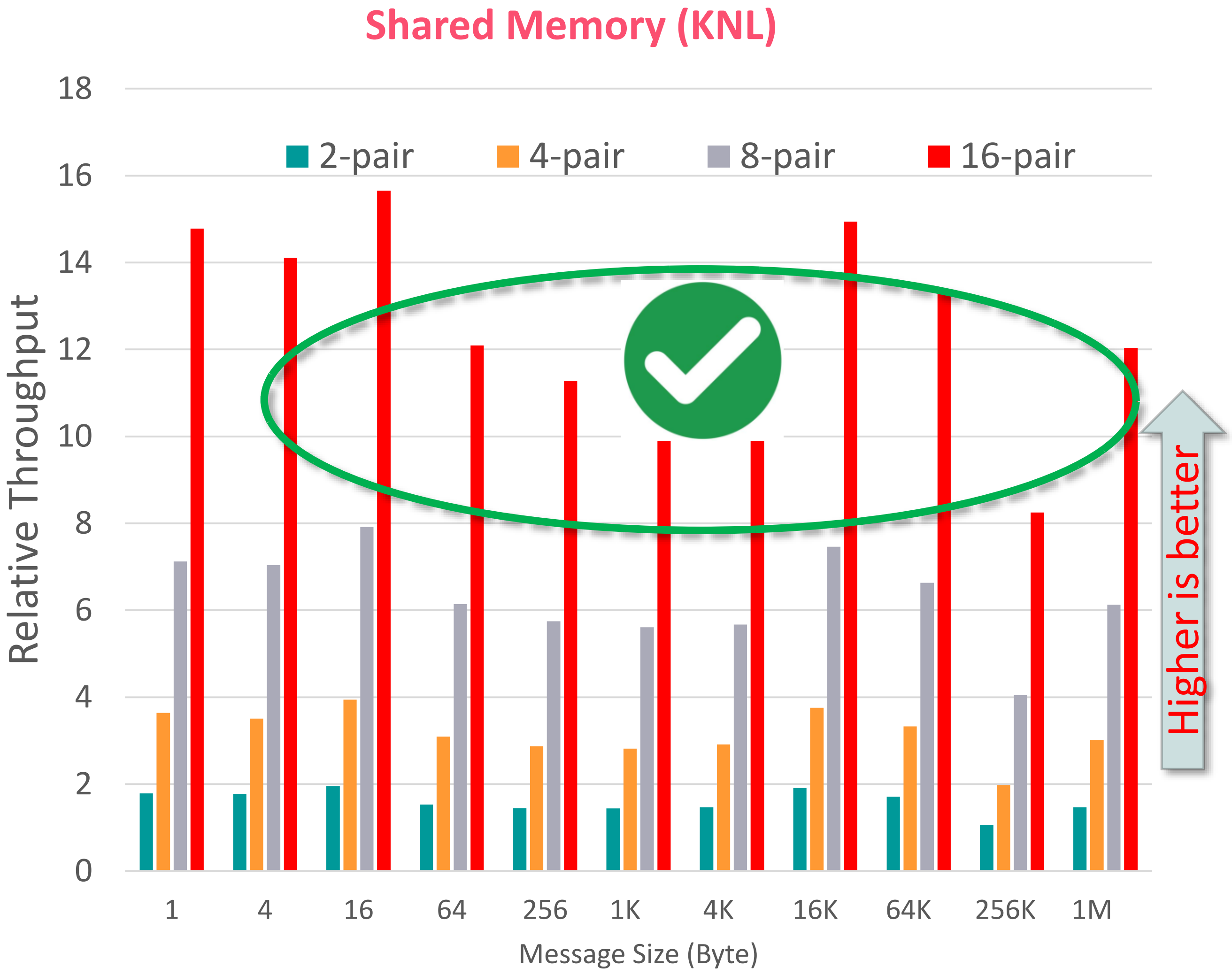
* Bloch et al. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction

Relative Throughput of Different Architectures

- Using OSU Micro benchmark suite*
- “Multiple Bandwidth Test”
  - Back-to-back messages
    - Sent to a pair before waiting for receive
- Evaluates the aggregate unidirectional bandwidth between multiple pairs of processes
- 1) Xeon + IB, 2)Xeon + Omni-Path, and 3) KNL + Omni-Path

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

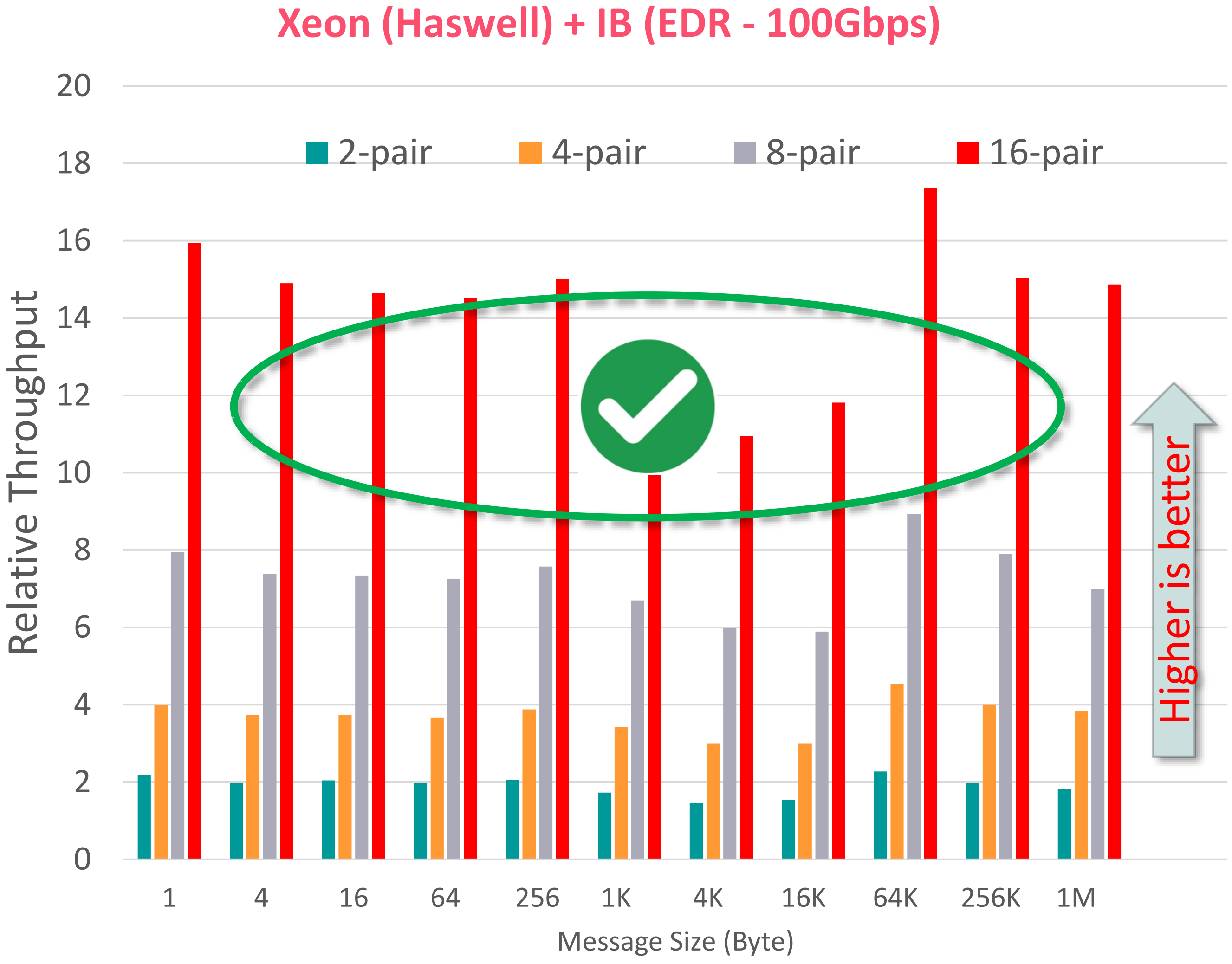
Communication Characteristics of Modern Architectures: Intra-node Communication



Multiple pair test vs. one pair test

- The relative throughput very close to the number of pairs
- Support many concurrent intra-node communication

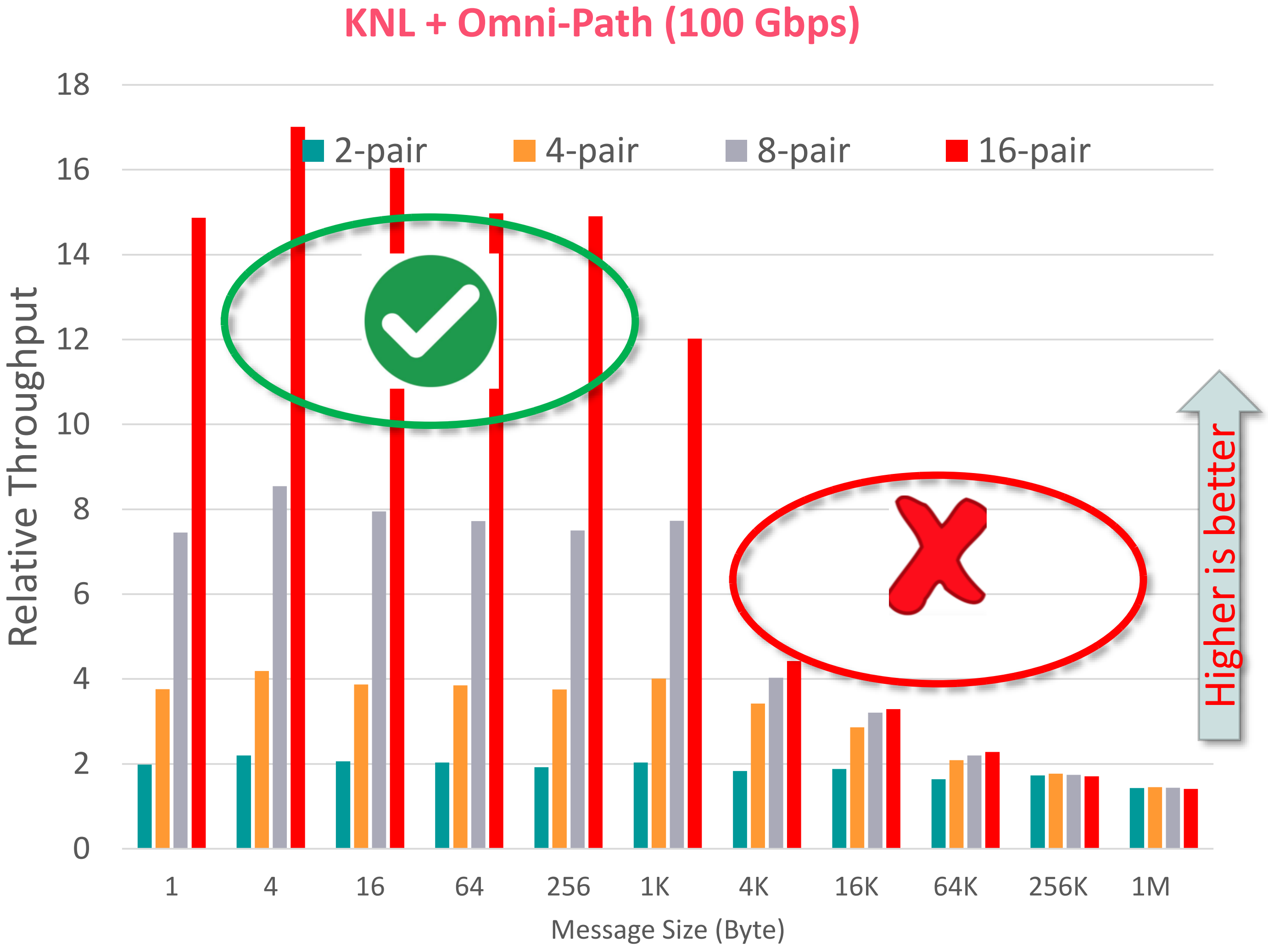
Communication Characteristics of Modern Architectures: InfiniBand Interconnect



Multiple pair test vs. one pair test

- The relative throughput close to the number of communicating processes per node
- Support many concurrent intra-node communication

Communication Characteristics of Modern Architectures: Omni-Path Interconnect



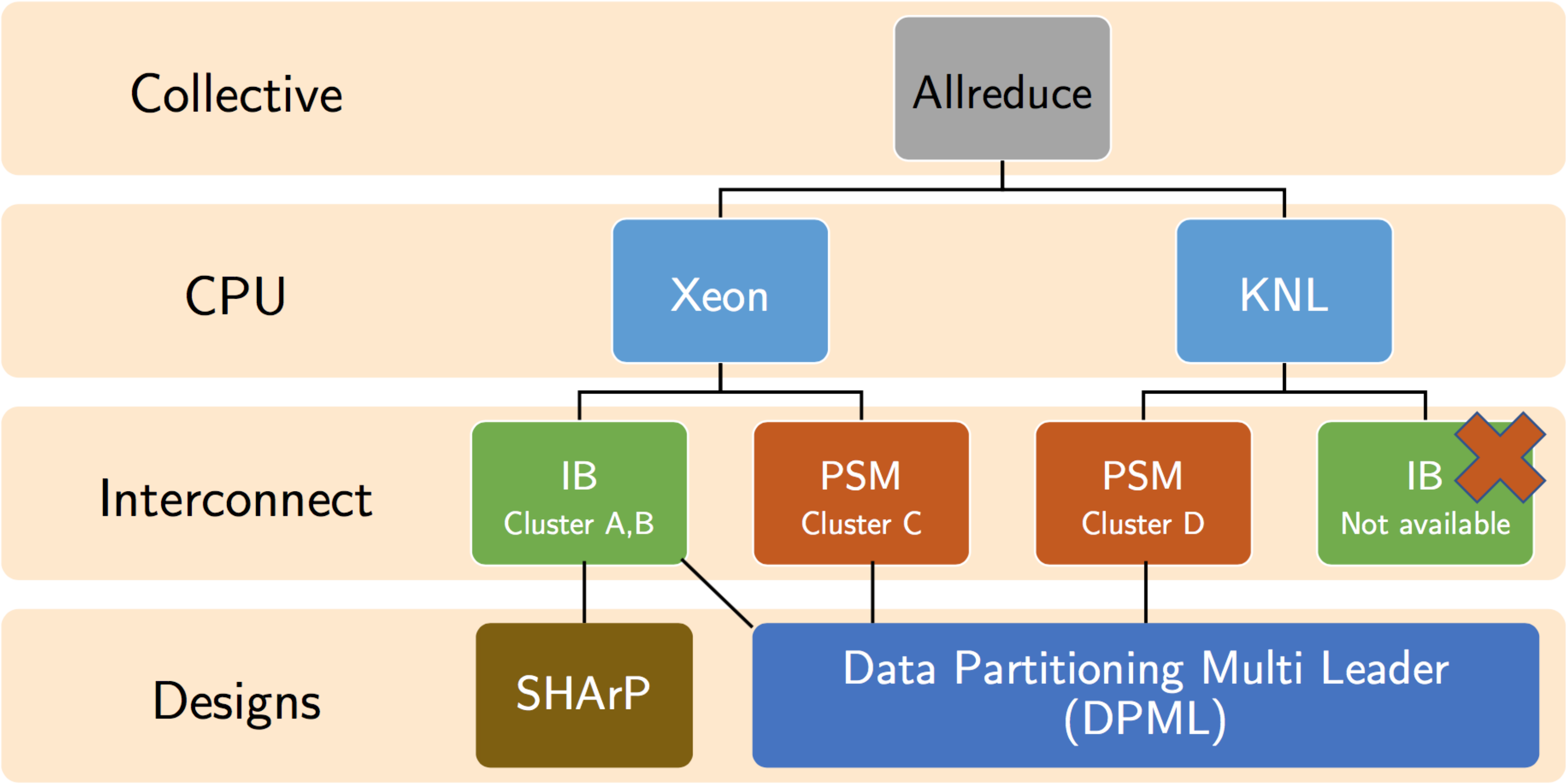
Multiple pair test vs. one pair test

- The relative throughput of one for large messages
- Supports many concurrent communications for small and medium message range
- Similar behavior observed for Xeon + Omni-Path

Performance limitations of Existing Designs for MPI_Allreduce

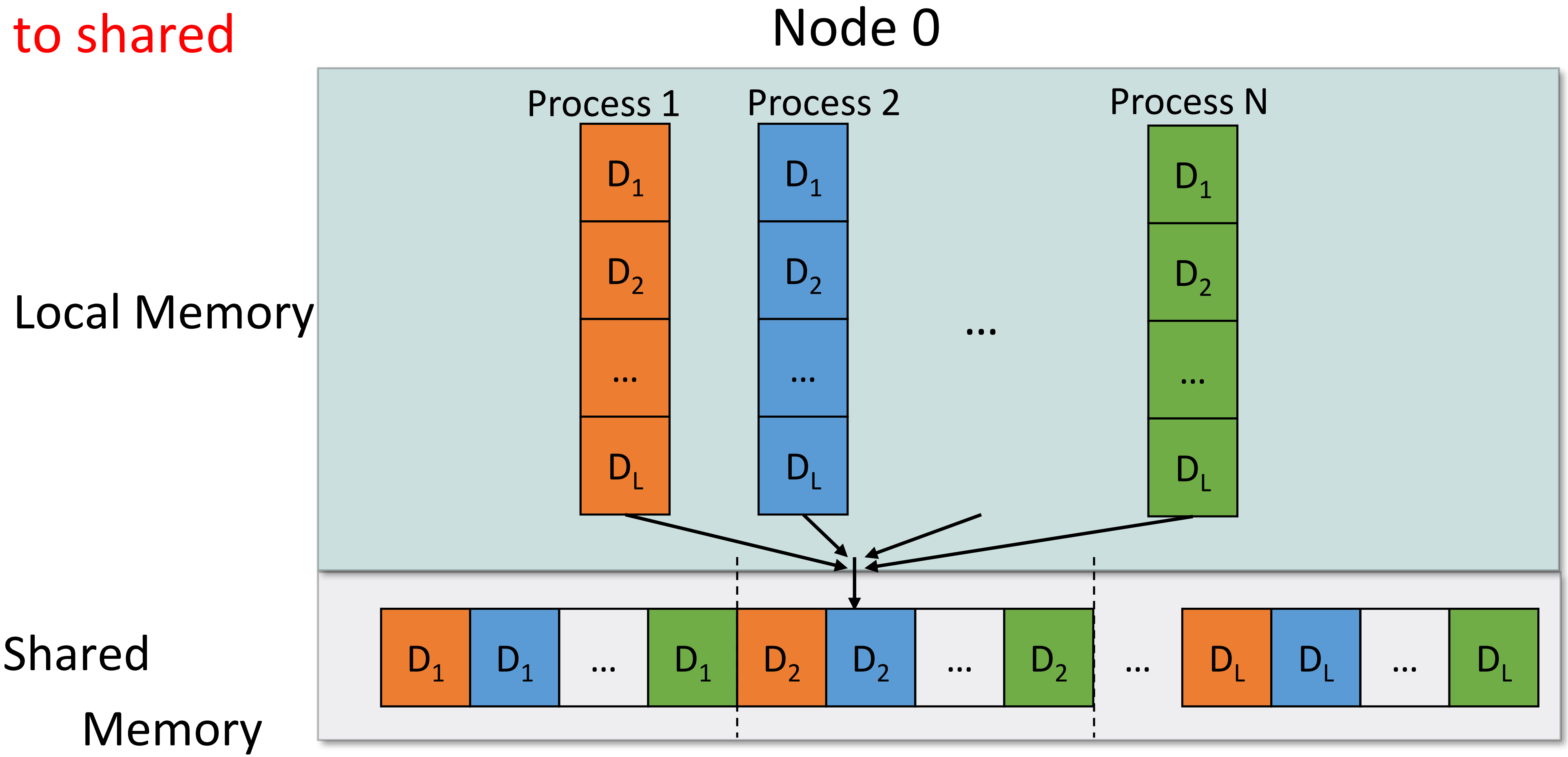
- Does not take advantage of large number of cores and high concurrency in communication
- Does not take advantage of shared memory collectives
  - Needs kernel support for zero-copy communication for large messages in same node
- Too many inter-node communication for large PPNs
- Limited performance due to extra QPI transfers
- Limited computing power of switches limits its performance for medium and large message ranges

Design Outline



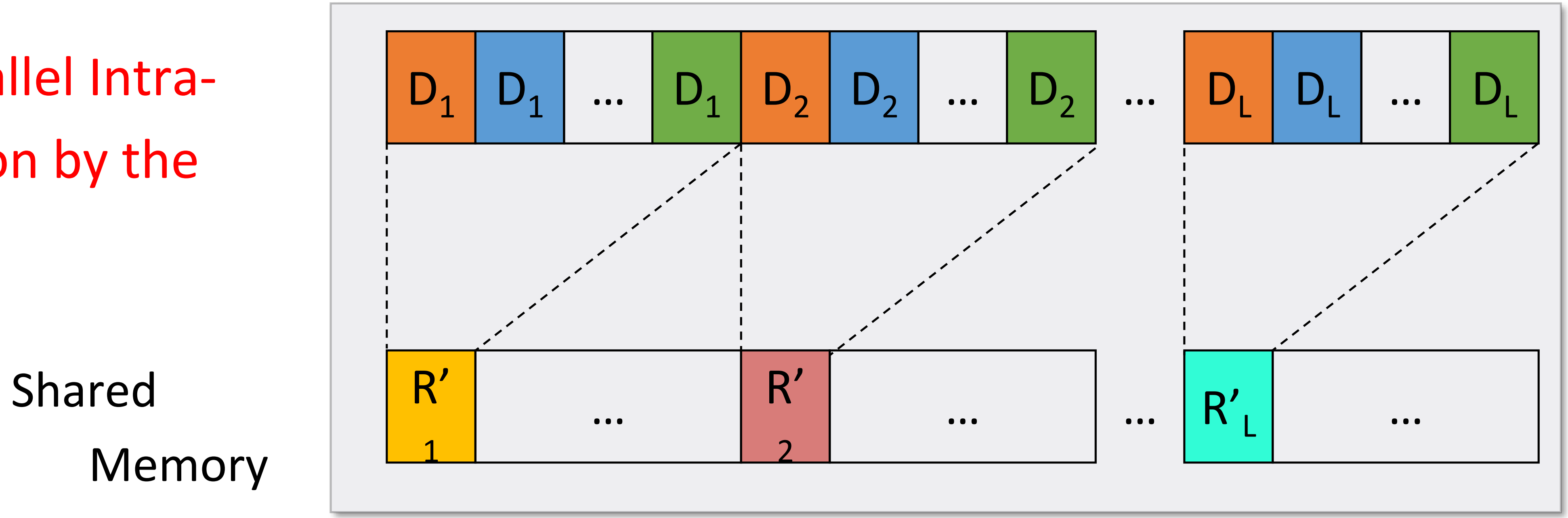
DPML Design Phases

- Phase 1: Copy to shared Memory



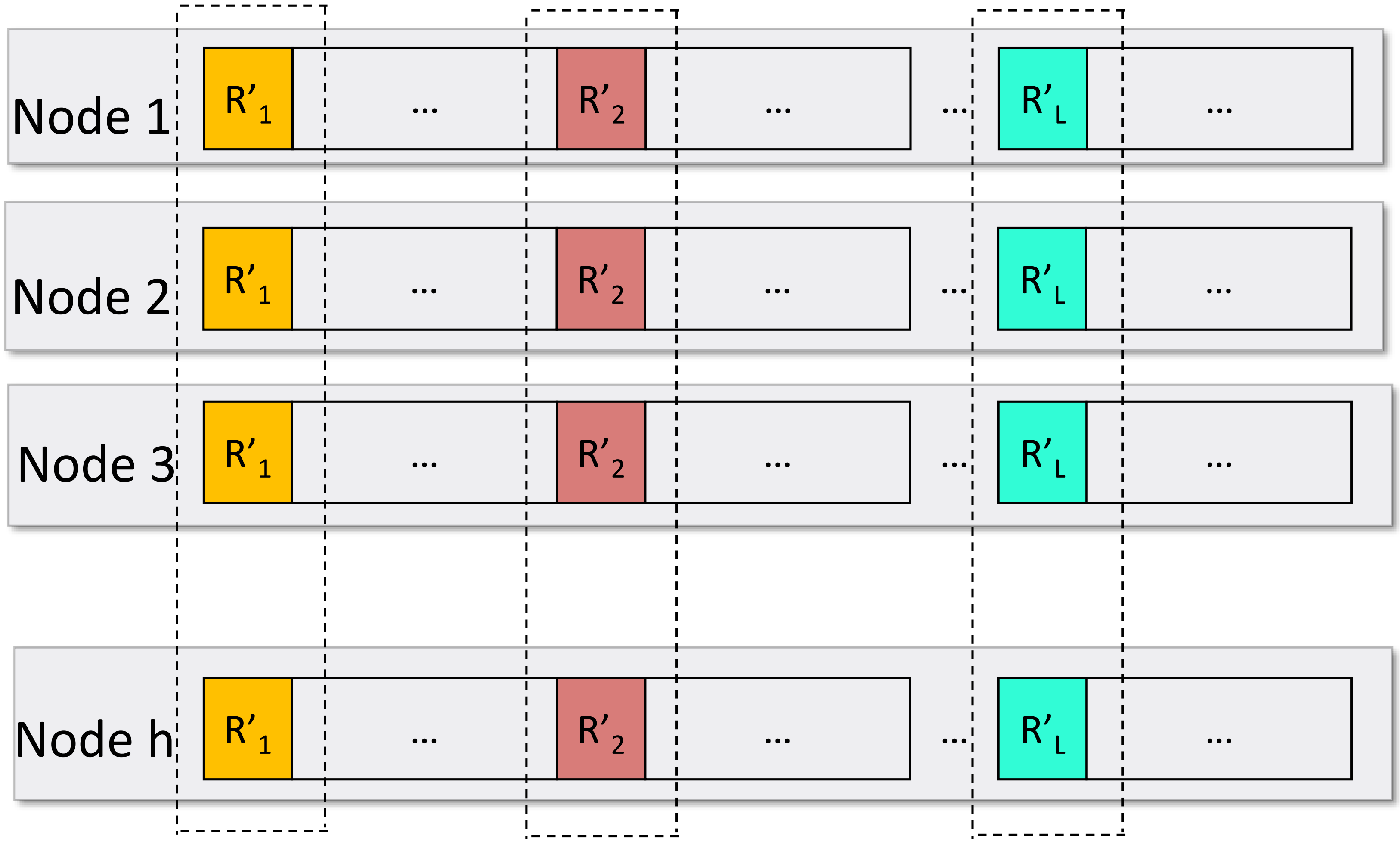
DPML Design Phases

- Phase 1: Copy to shared Memory
- Phase 2: Parallel Intra-node reduction by the leaders



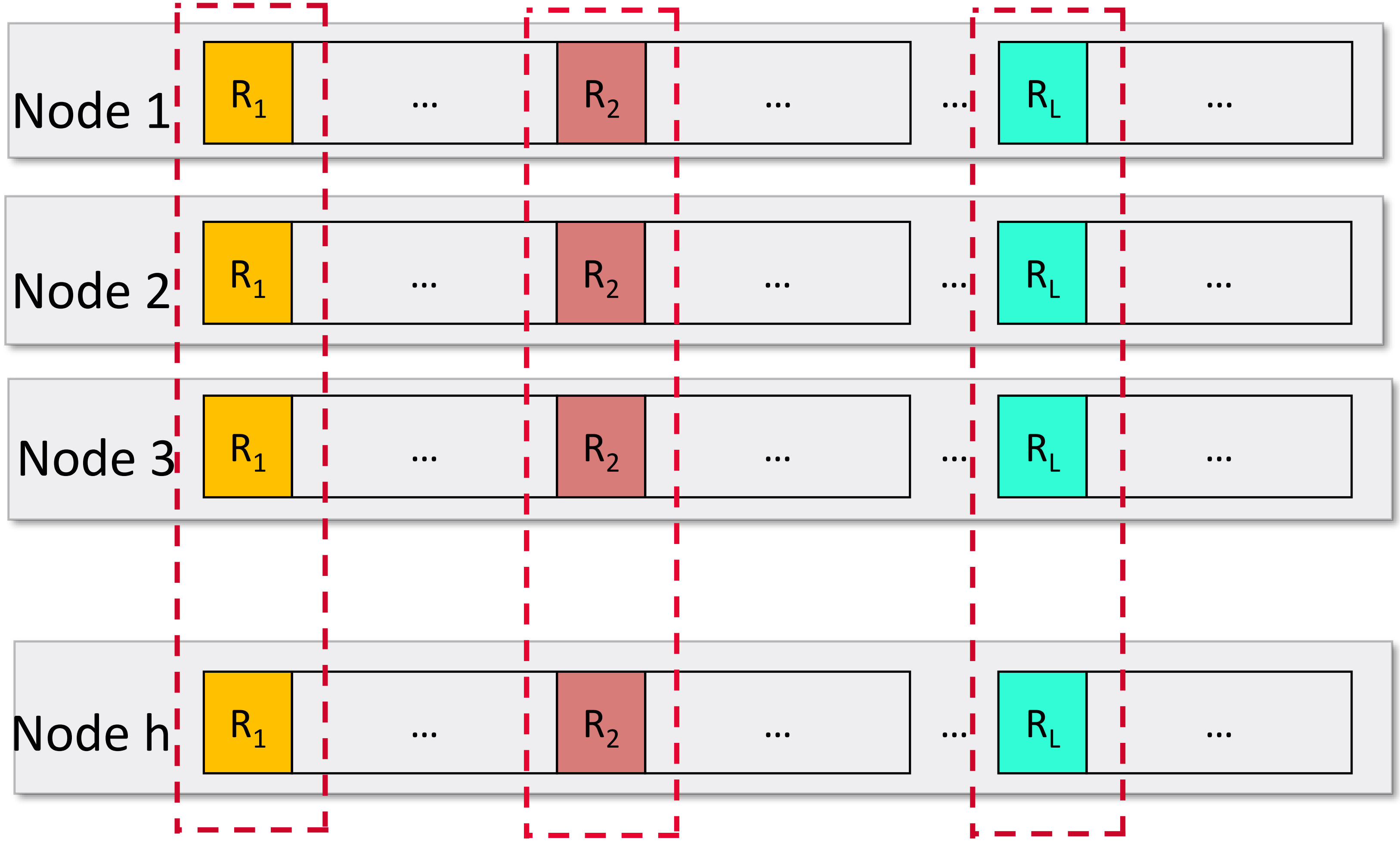
DPML Design Phases

- Phase 1: Copy to shared Memory
- Phase 2: Parallel Intra-node reduction by the leaders



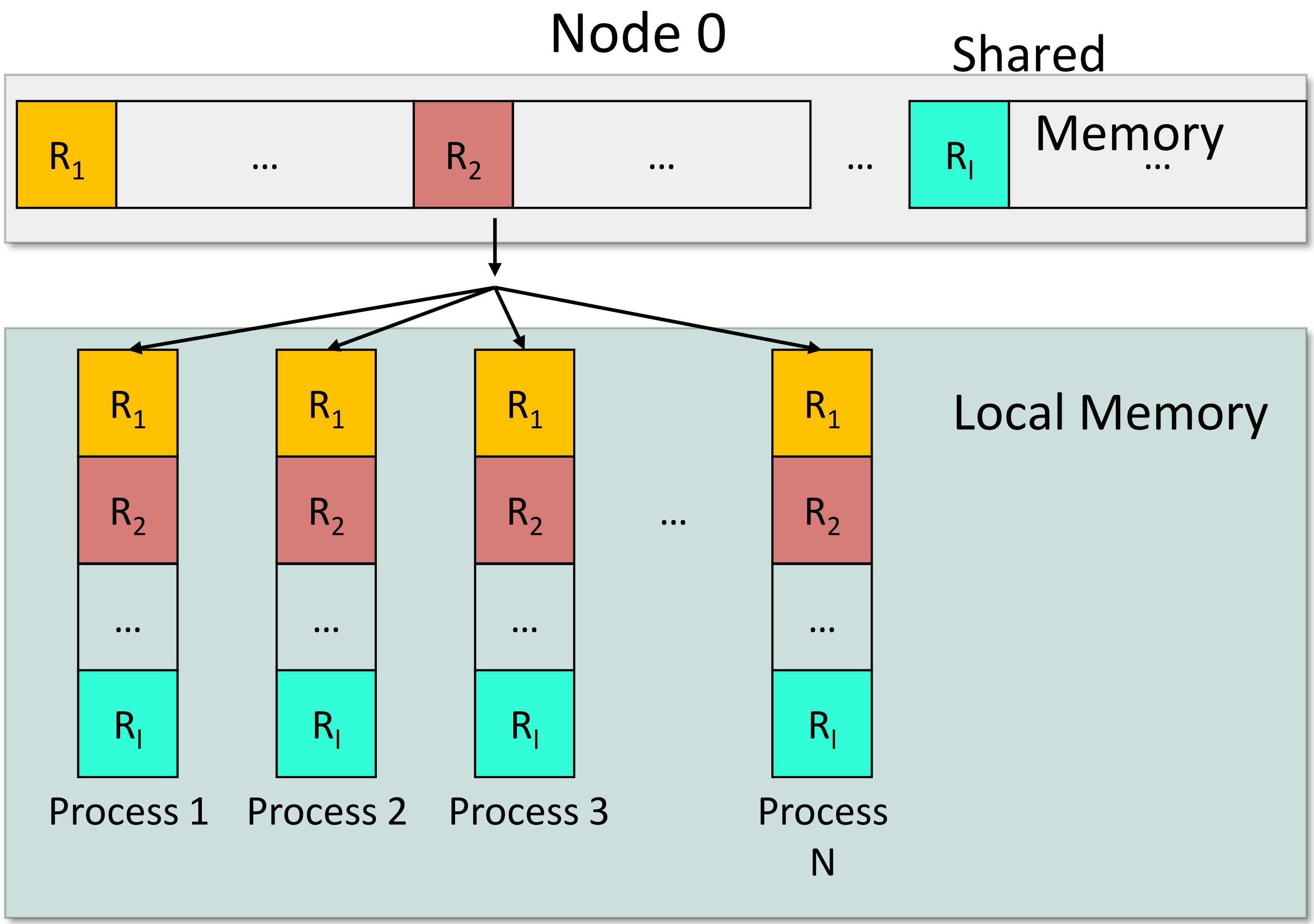
DPML Design Phases

- Phase 1: Copy to shared Memory
- Phase 2: Parallel Intra-node reduction by the leaders
- Phase 3: Parallel Inter-node Allreduce by the leaders with same index

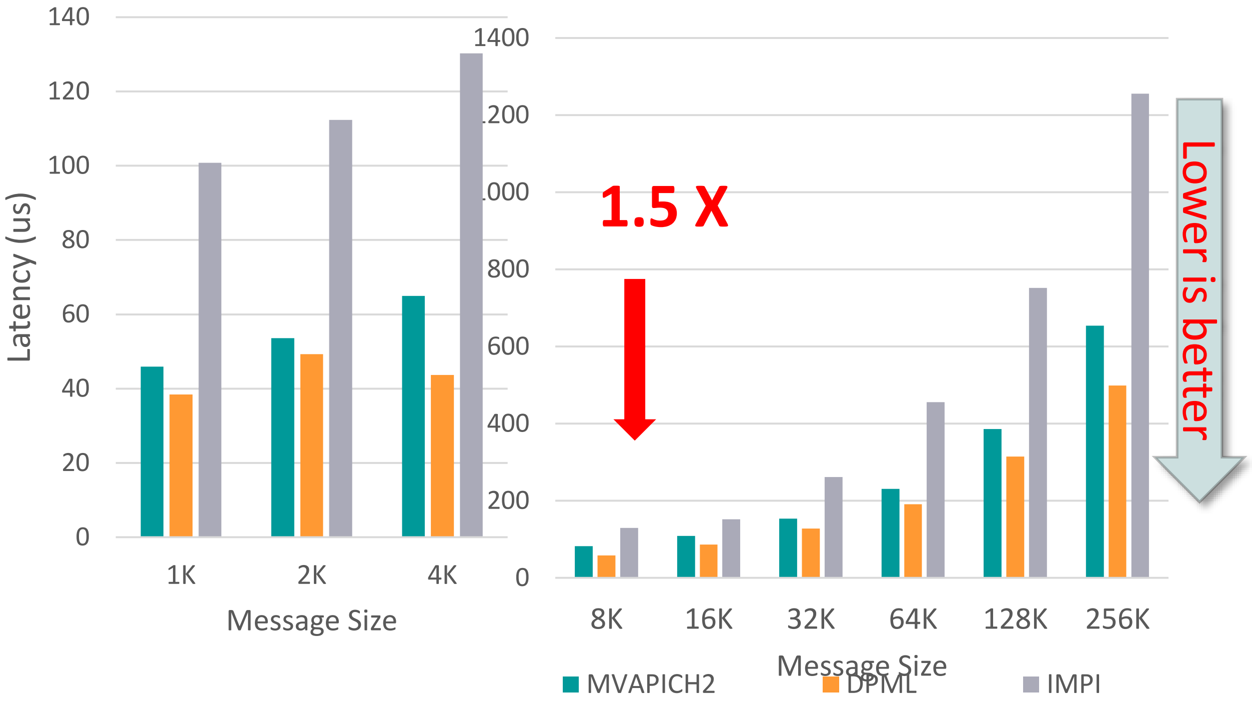


DPML Design Phases

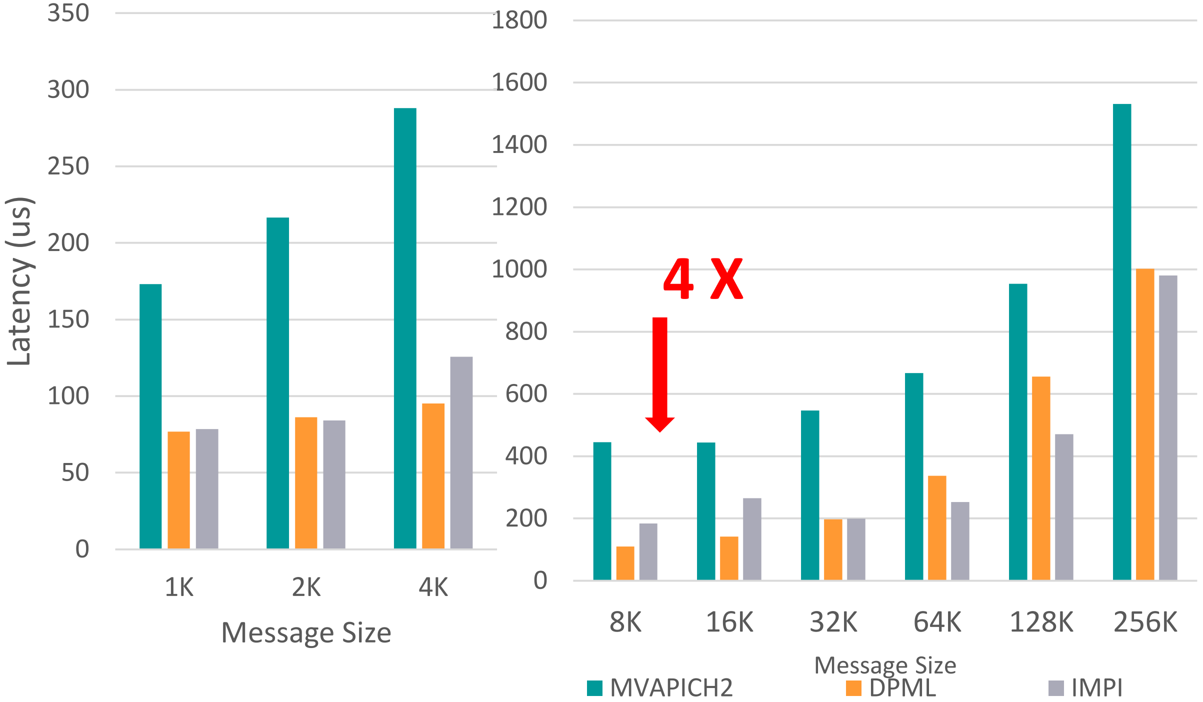
- Phase 1: Copy to shared Memory
- Phase 2: Parallel Intra-node reduction by the leaders
- Phase 3: Parallel Inter-node Allreduce by the leaders with same index
- Phase 4: Parallel distribution of Allreduce results to local buffers



Performance of MPI_Allreduce On Omni-Path



XEON + Omni-Path (64 Nodes, 28 PPN*)

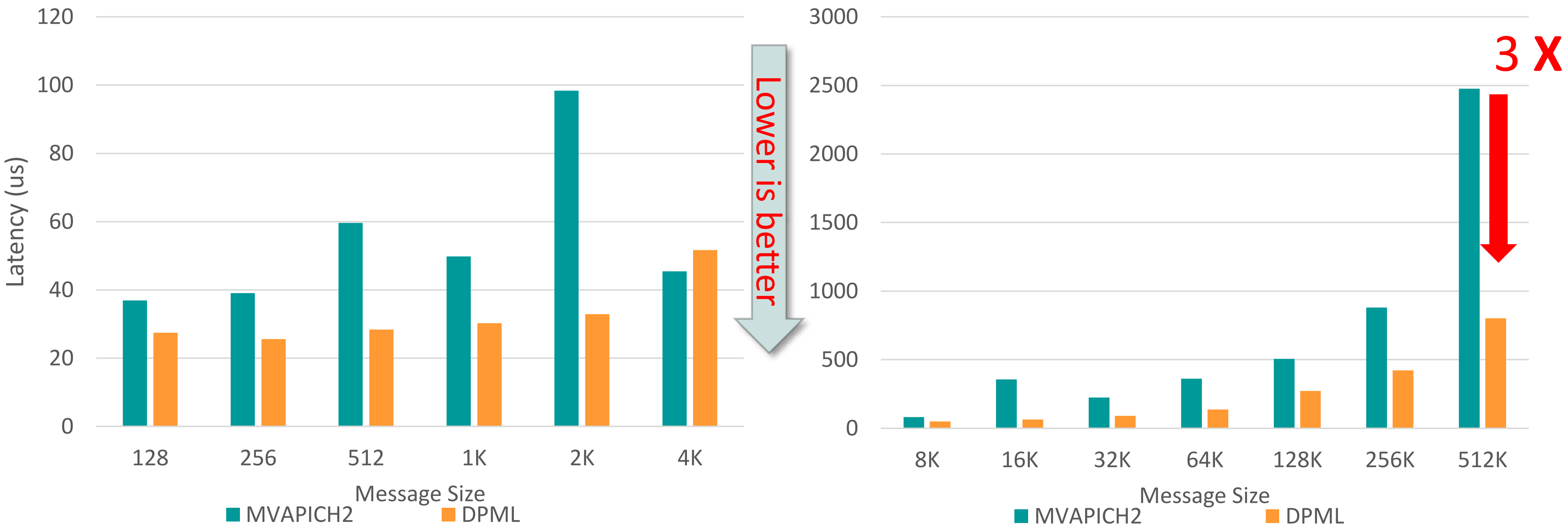


KNL + Omni-Path (32 Nodes, 32 PPN)

- DPML always outperform MVAPICH2 for all medium and large message range
- DPML outperform IMPI in medium message range
- High parallelism of DPML benefits KNL more than XEON

*Processes Per Node

Performance of MPI_Allreduce On InfiniBand



XEON + IB (64 Nodes, 28 PPN)

- DPML outperform MVAPICH2 for most of the medium and large message range
  - With 512K bytes, 3X improvement of DPML
- Higher benefits of DPML as the message size increases

Conclusions & Future Work

- Designed multi-leader based collective operations
  - Capable of taking advantage of high-end features offered by modern network interconnects
- Modeled and analyzed proposed design theoretically
- The benefits were evaluated on different architectures
- The DPML design is released as a part of MVAPICH2-X 2.3b! Check out:
  - http://mvapich.cse.ohio-state.edu/overview/#mv2X
- Studied the interplay between communication pattern of applications and different tag matching schemes
- Proposes, designed and implemented a dynamic and adaptive tag matching scheme capable to adapting dynamically to the communication characteristics of applications
- The adaptive approach opens up a new direction to design tag matching schemes for next-generation exascale systems