Hybrid MPI - A Case Study on the Xeon Phi Platform

Udayanga Wickramasinghe

Center for Research on Extreme Scale Technologies (CREST)
Indiana University

Greg Bronevetsky
Lawrence Livermore National Laboratory

Andrew Friedley
Intel Corporation

Andrew Lumsdaine
Center for Research on Extreme Scale Technologies (CREST)
Indiana University

ROSS 2014 Munich, Germany

Hybrid MPI - Motivation

- MPI – dominant programming model in HPC
- Hybrid MPI – MPI implementation specialized for intra-node point to point communication
  - Fast point to point communication over shared memory hardware
- Evolving processor architectures
  - Single Core → Dual Core → Quad Core → Multi-Core → Many-Core/Clusters
  - High compute density and performance per watt
  - Robust shared memory hardware
- Motivation – maximize use of many core hardware
  - Maximum use of shared memory hardware of the Xeon Phi
  - Gain Maximum communication performance from available bandwidth of the Xeon Phi hardware

Agenda

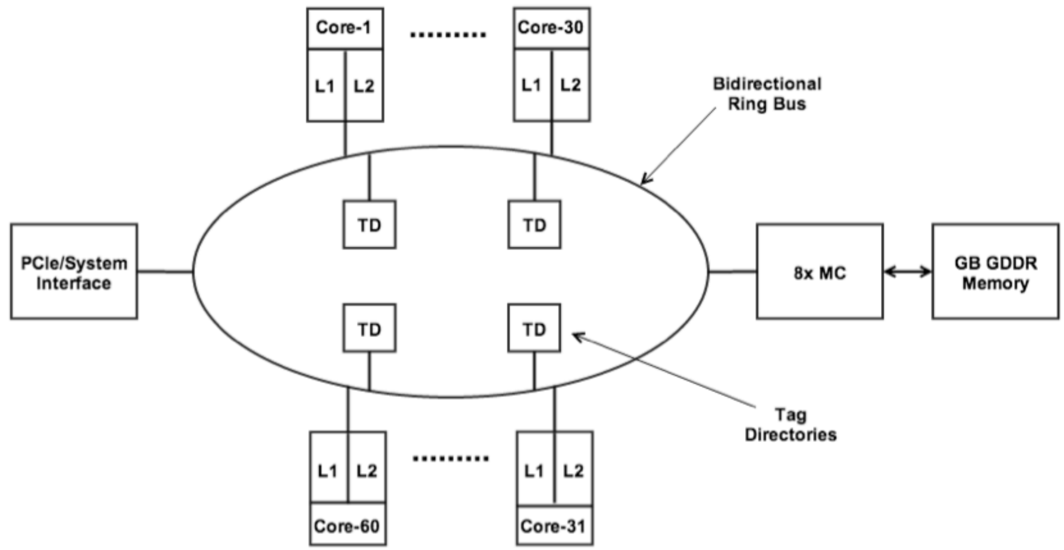
- Xeon Phi Platform
- Traditional MPI Design
- Hybrid MPI
  - A Shared Heap
  - Communication
- Experimental Evaluation
  - Micro-benchmarks
  - Applications
- Hybrid MPI Highlights
- Towards Hybrid MPI Future

Xeon Phi Platform

- Intel Many Integrated Core Architecture (MIC) → Xeon Phi (earlier known as Knights Corner - 50 cores)
  - Utilized in #1 supercomputing cluster – Tianhe-2 (http://top500.org/)
  - STAMPEDE @ TACC
- Xeon Phi processor → 61 cores with 4 Hardware Threads
  - No out of order execution
  - x86 compatibility
  - Shorter instruction set pipeline
- Simpler cores → higher power efficiency

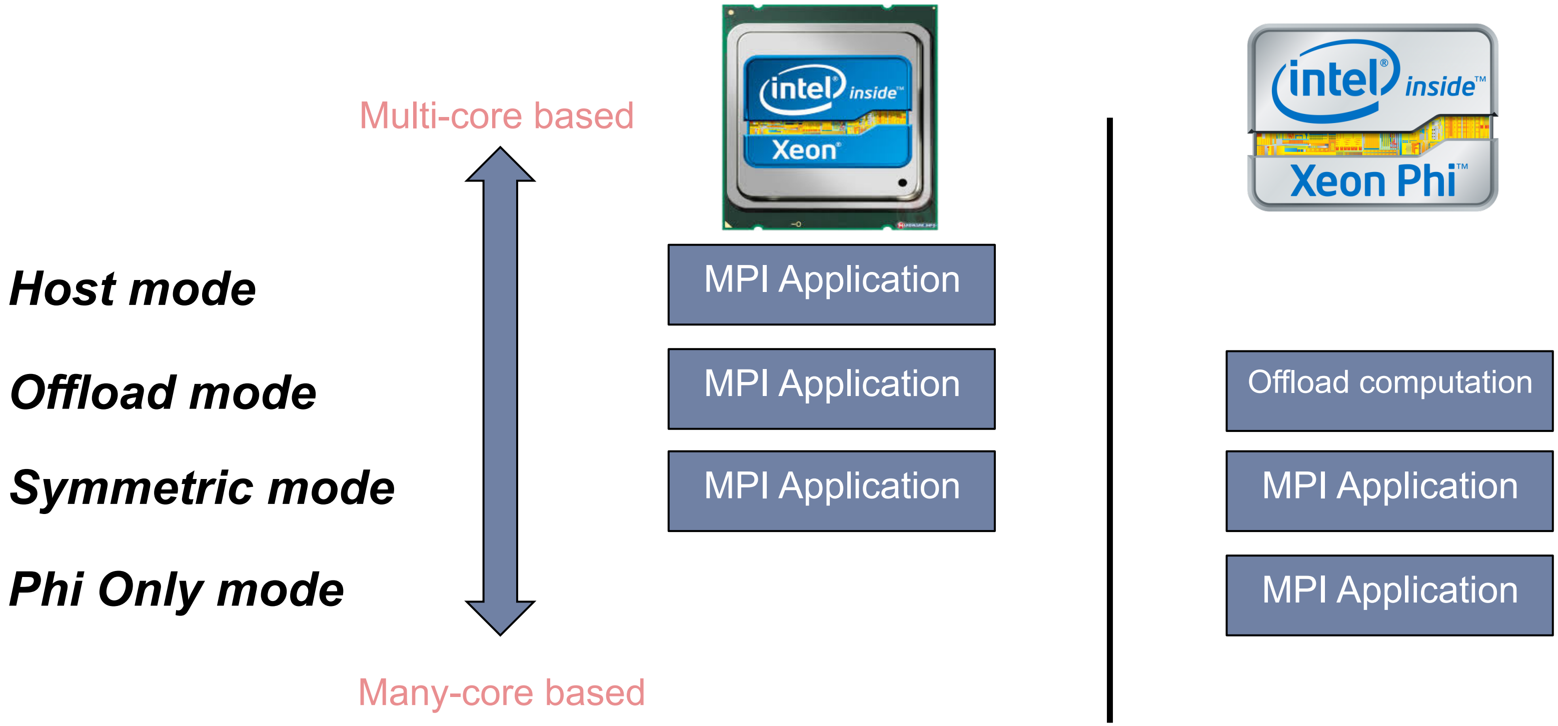
Xeon Phi Platform

- Inter core communication
  - Bi-directional ring topology interconnect
- ~320GB/s Aggregated Theoretical bandwidth



- 4 modes of operation (MPSS)
  - Host
  - Offload – offloads computation
  - Symmetric – ranks in both Host and Phi
  - Phi Only

Xeon Phi Software Model/Stack (MPSS)



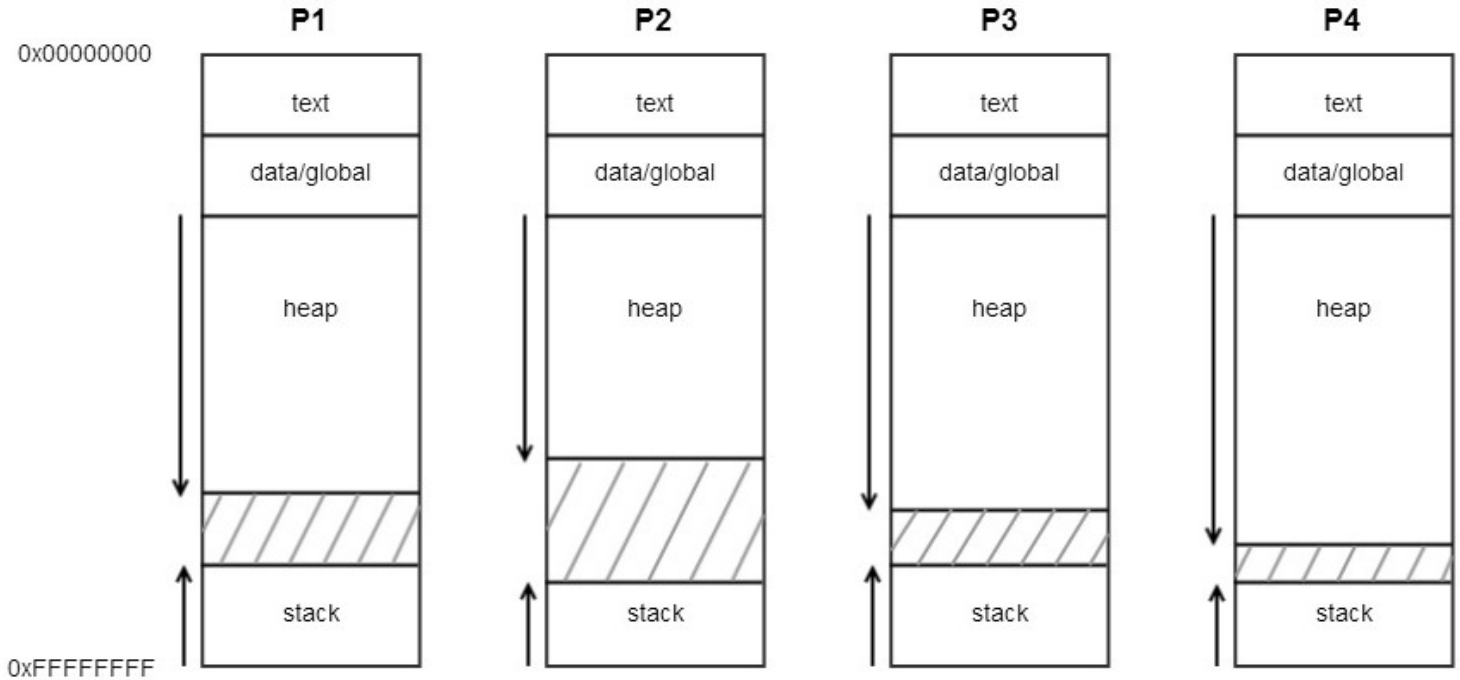
- Offload/Symmetric/Phi-only supported via Intel Many-core Platform Software Stack(MPSS)
  - Shared memory/ SHM
  - SCIF
  - IB verbs/ IB-SCIF

INDIANA UNIVERSITY
SCHOOL OF INFORMATICS AND COMPUTING
Bloomington

Agenda

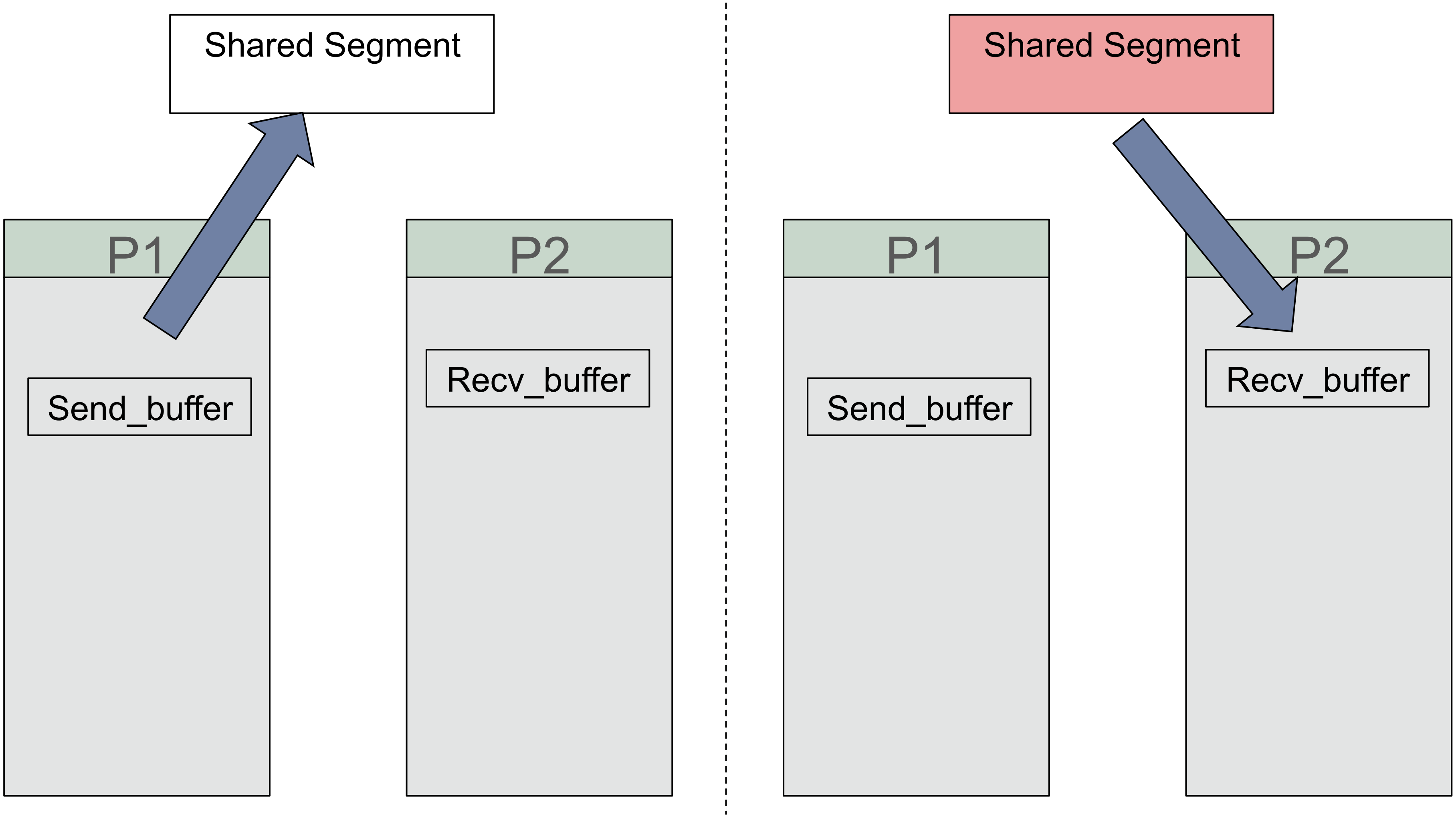
- Xeon Phi Platform
- Traditional MPI Design
- Hybrid MPI
  - A Shared Heap
  - Communication
- Experimental Evaluation
  - Micro-benchmarks
  - Applications
- Hybrid MPI Highlights
- Towards Hybrid MPI Future

Traditional MPI with Disjoint Address Spaces



- Process based ranks - Regular process abstraction - Shared nothing
- Communication
  - Disjoint address spaces – multiple copies
  - IPC/Kernel buffers/ shared buffers – resources grow rapidly with number of ranks

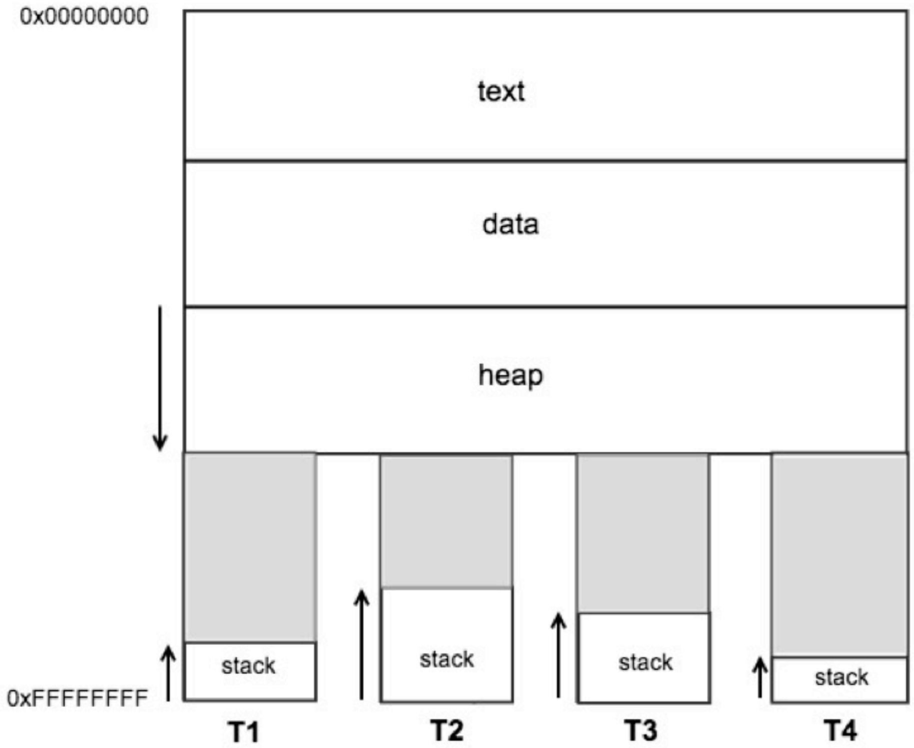
Traditional MPI with Disjoint Address Spaces (contd..)



- Two Copies – resources grow as ranks increase

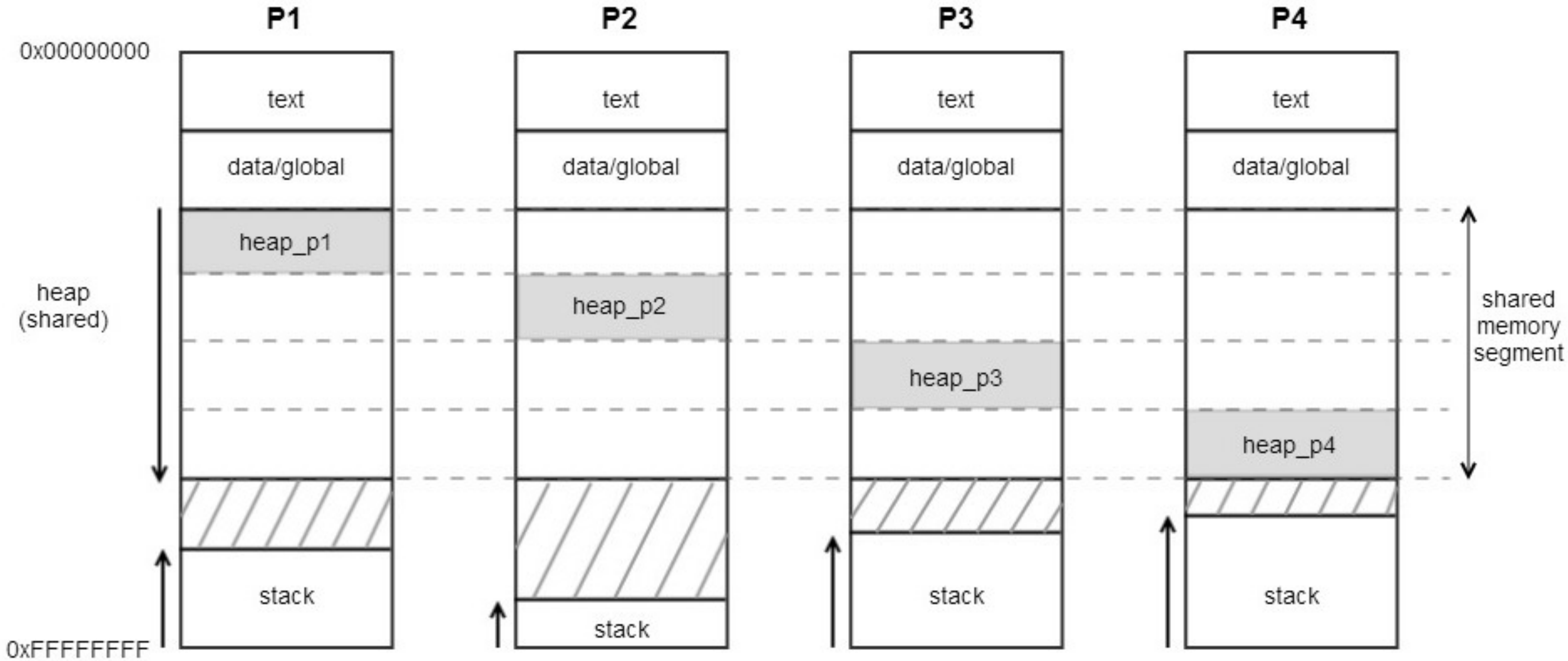
Alternative MPI – Avoid Copies

- Necessary to share memory
  - Thread based
  - Share everything
    - Heap/data/text segments are shared among threads
- Thread pinning to Xeon Phi cores via KMP_AFFINITY
  - scatter/compact/fine



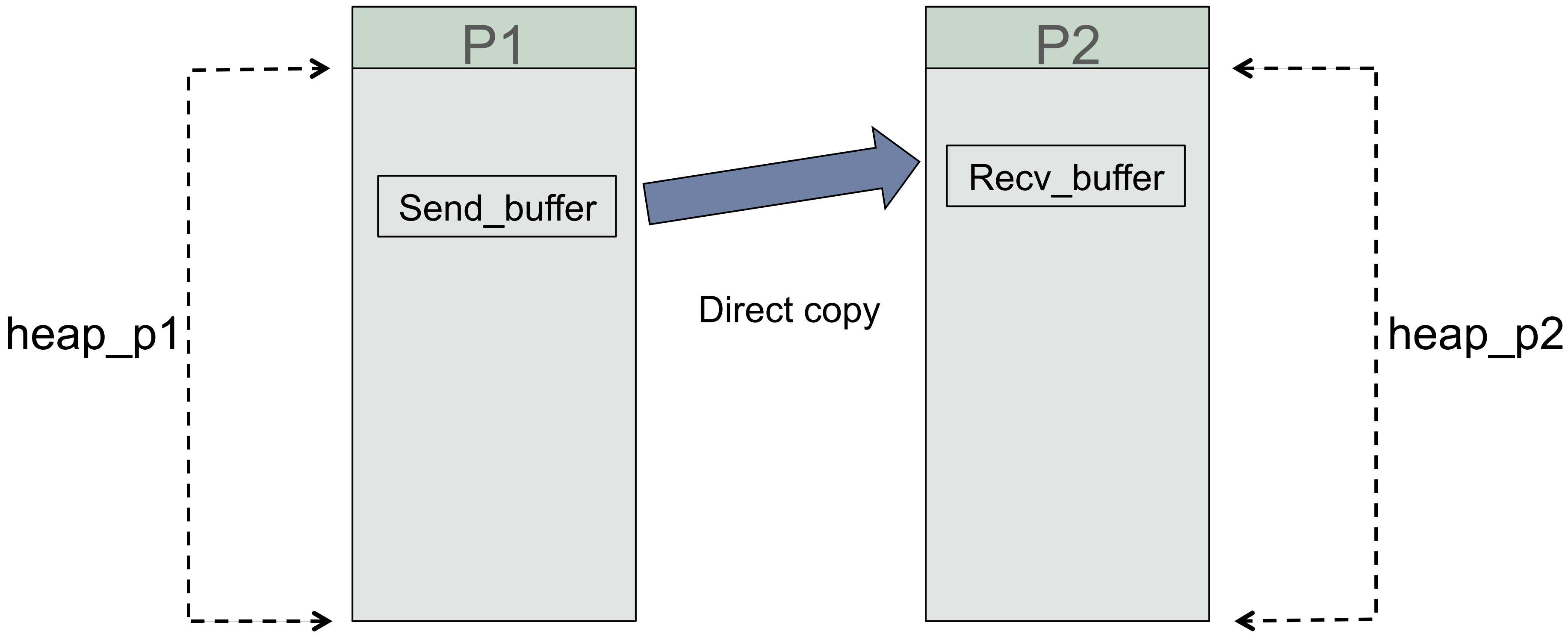
- However few problems arise when resources are shared
  - Ensure mutual exclusion
  - Transform globals/heap vars to thread local
  - *Network resource contention*

Hybrid MPI – A Shared Heap



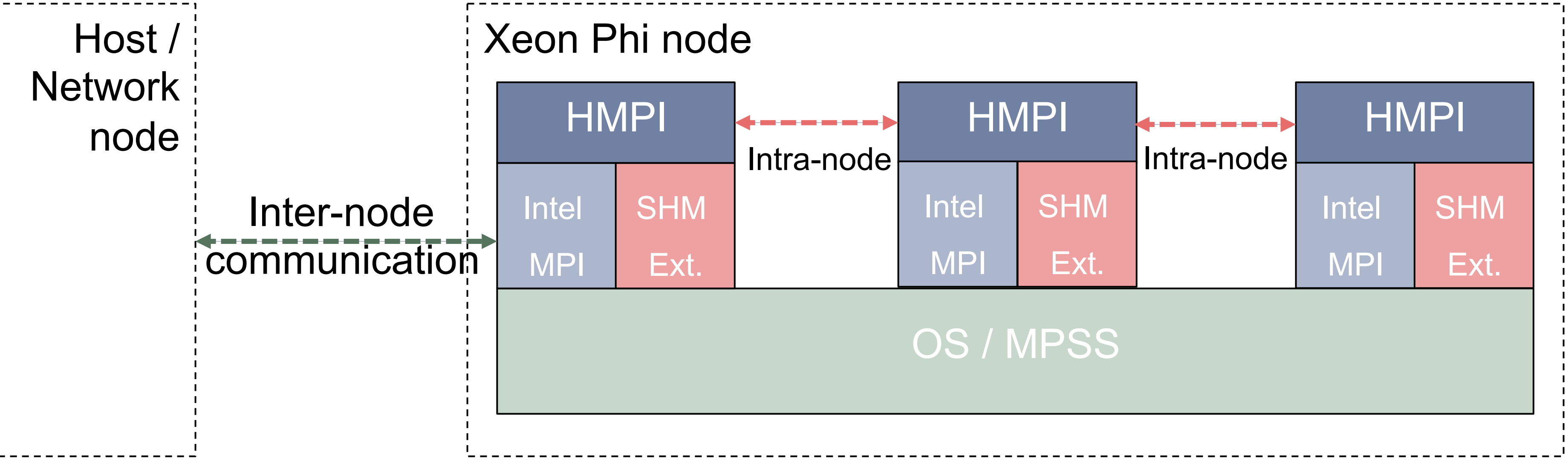
- Hybrid MPI approach
- Each rank P1, P2, P3, P4 heap is *mmap()* ed to a shared segment
  - Has access to entire shared segment
- Each process allocates memory on their own chunk → heap_p1, heap_p2, heap_p3, heap_p4

Hybrid MPI – A Shared Heap (contd..)



- Single Copy using the unified shared address space of Hybrid MPI
- Implementation with mmap()
  - **MAP_SHARED , MAP_FIXED** features

Hybrid MPI View on Xeon Phi

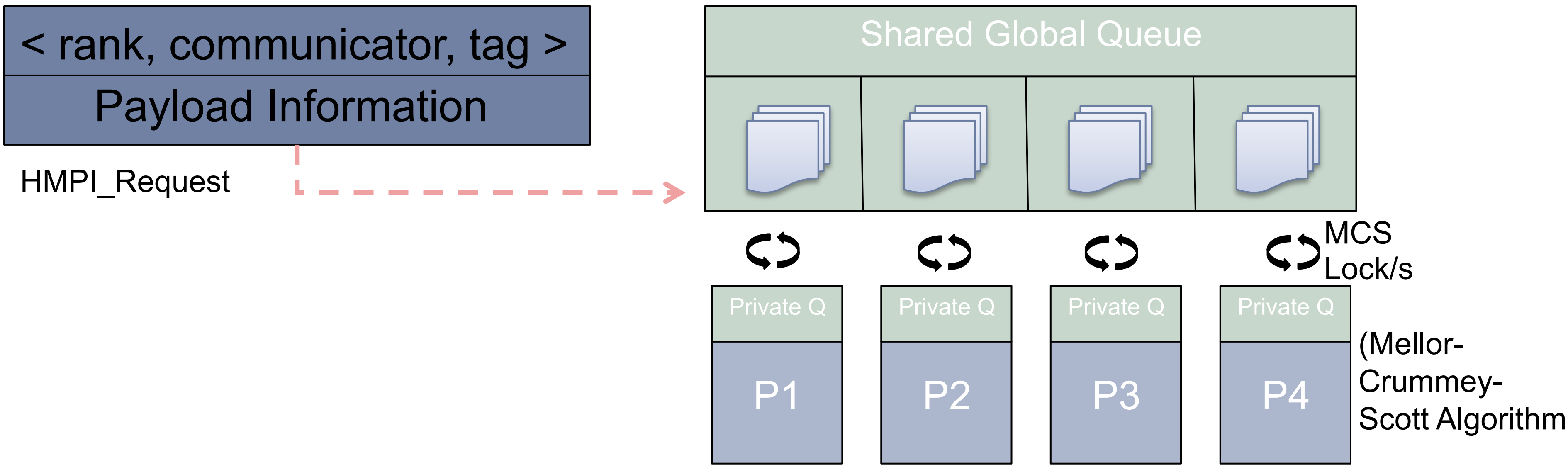


- Hybrid MPI has its own Shared Memory extension for Intra-node communication
- Inter-node communication via Intel MPI
  - Infiniband network
  - TCP/IP
  - PCIe(PCI express) / SCIF (Symmetric Communication Interface)

Agenda

- Xeon Phi Platform
- Traditional MPI Design
- Hybrid MPI
  - A Shared Heap
  - Communication
- Experimental Evaluation
  - Micro-benchmarks
  - Applications
- Hybrid MPI Highlights
- Towards Hybrid MPI Future

Hybrid MPI – Message matching



- ‘send’ requests are matched with local ‘receive’ requests in *HMPI_Progress*
  - Match for tuples <rank, comm, tag>
- Two Queues used
  - Shared – protected by global MCS lock
  - Private – where match is performed, drained from global queue
  - Minimize contention

Hybrid MPI – Communication protocols

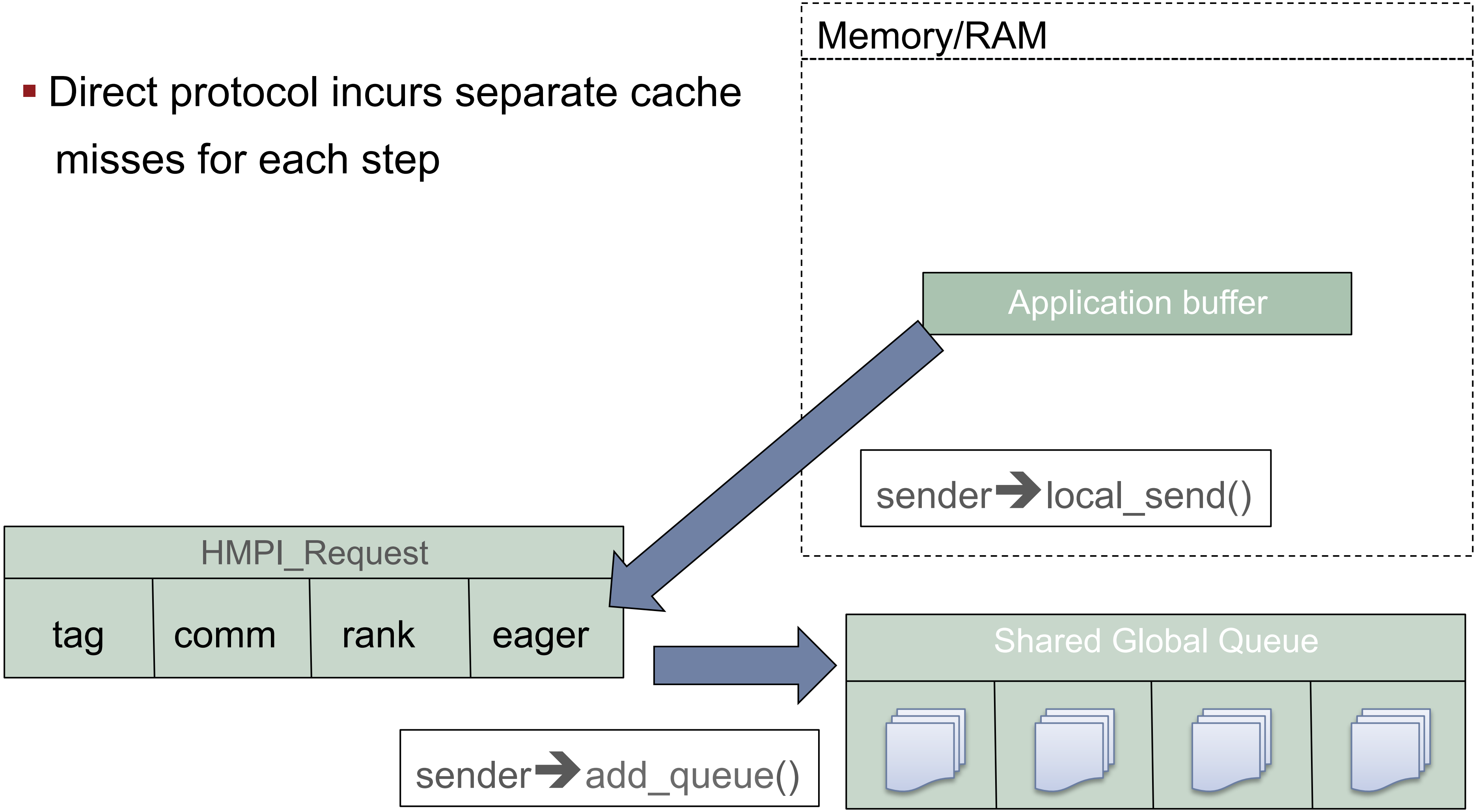
- 3 protocols
  - Direct Transfer
  - Immediate Transfer
  - Synergistic Transfer

- Direct Transfer
  - Single *memcpy*() to transfer from sender's buffer to receive
  - Applied when message is medium sized (512b ≤ m ≤ 8KB)

- Immediate Transfer
  - Applied when message size is small (≤ 512 bytes)
  - Payload is transferred immediately with HMPI request (header)
  - Message is cache aligned to fit the cache lines and 32KB L1 cache
    - Avoids 2 copies → use temporal locality, payload will already be in receivers' L1/L2 cache

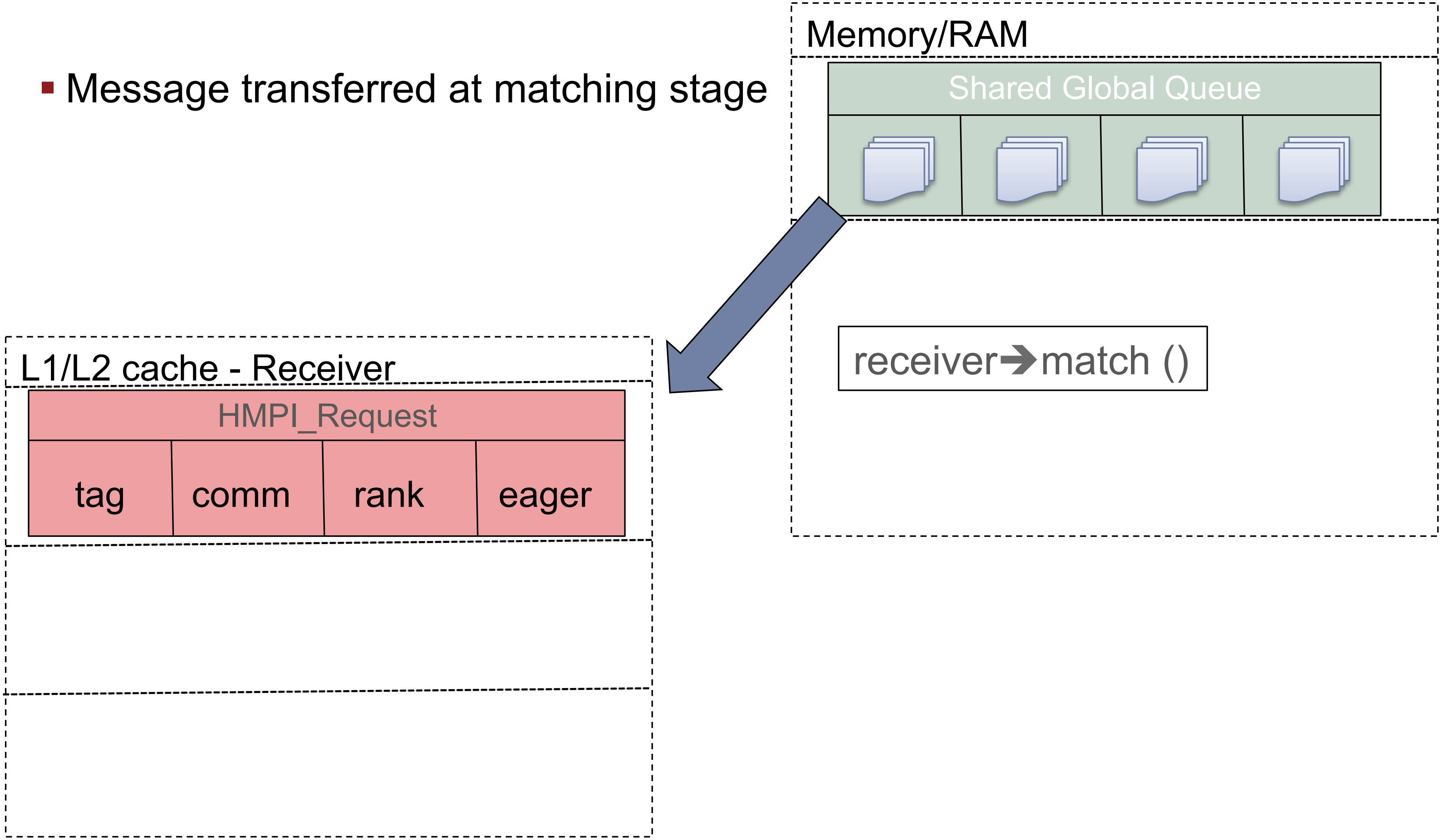
Hybrid MPI – Immediate protocol

- Direct protocol incurs separate cache misses for each step



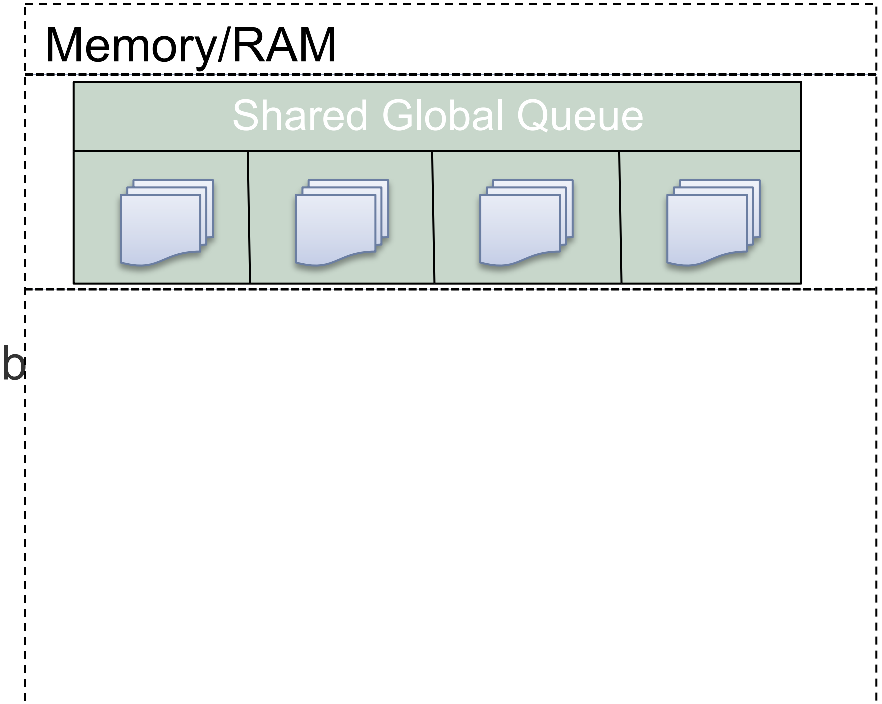
Hybrid MPI – Immediate protocol

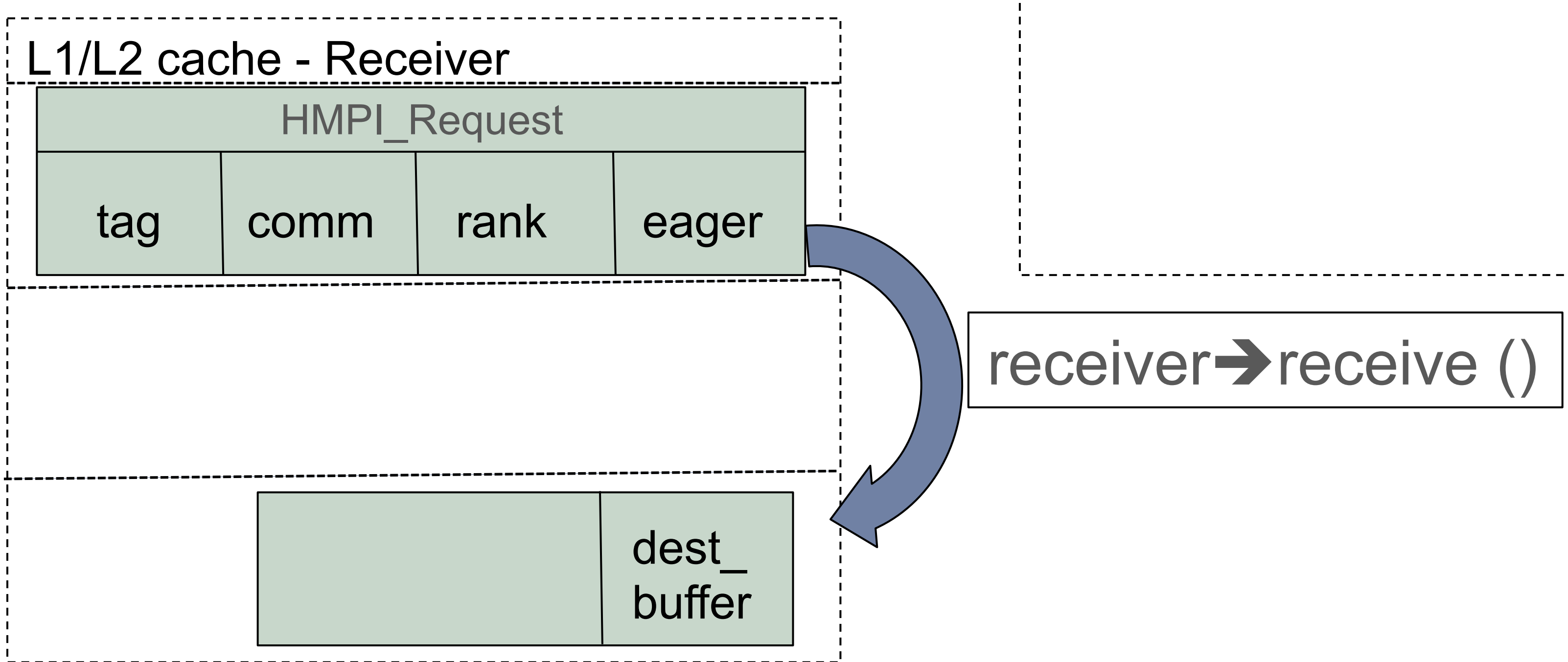
- Message transferred at matching stage



Hybrid MPI – Immediate protocol

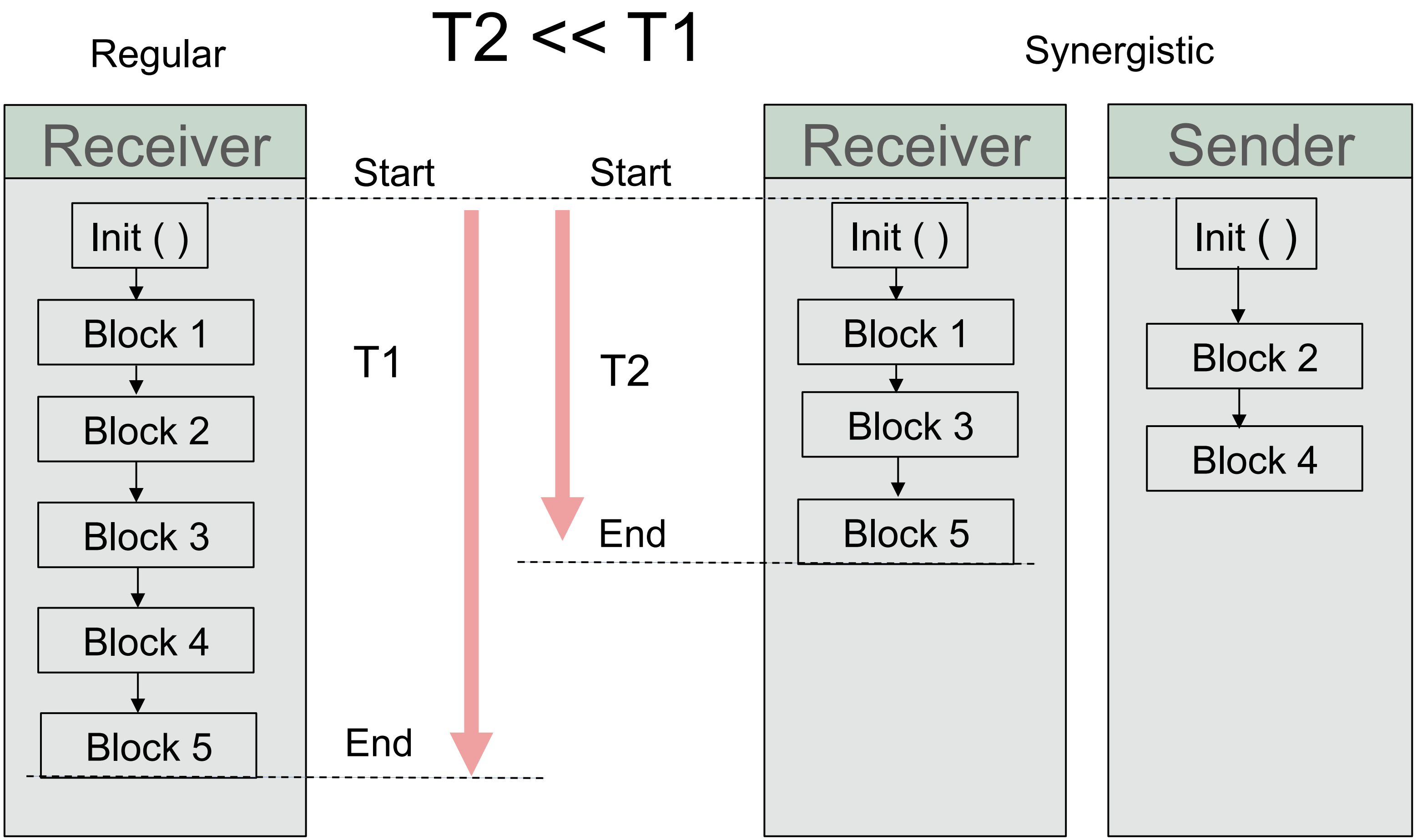
- At the data transfer
  - No cache miss to fetch data
  - If destination buffer is already on cache then extremely fast copying
  - 43% - 70% improvement over 32b – 512b





Hybrid MPI – Communication protocols

- Synergistic Transfer
  - Large messages (>=8KB) both sender and receiver engage actively in copying the message to destination



Agenda

- Xeon Phi Platform
- Traditional MPI Design
- Hybrid MPI
  - A Shared Heap
  - Communication
- Experimental Evaluation
  - Micro-benchmarks
  - Applications
- Hybrid MPI Highlights
- Towards Hybrid MPI Future

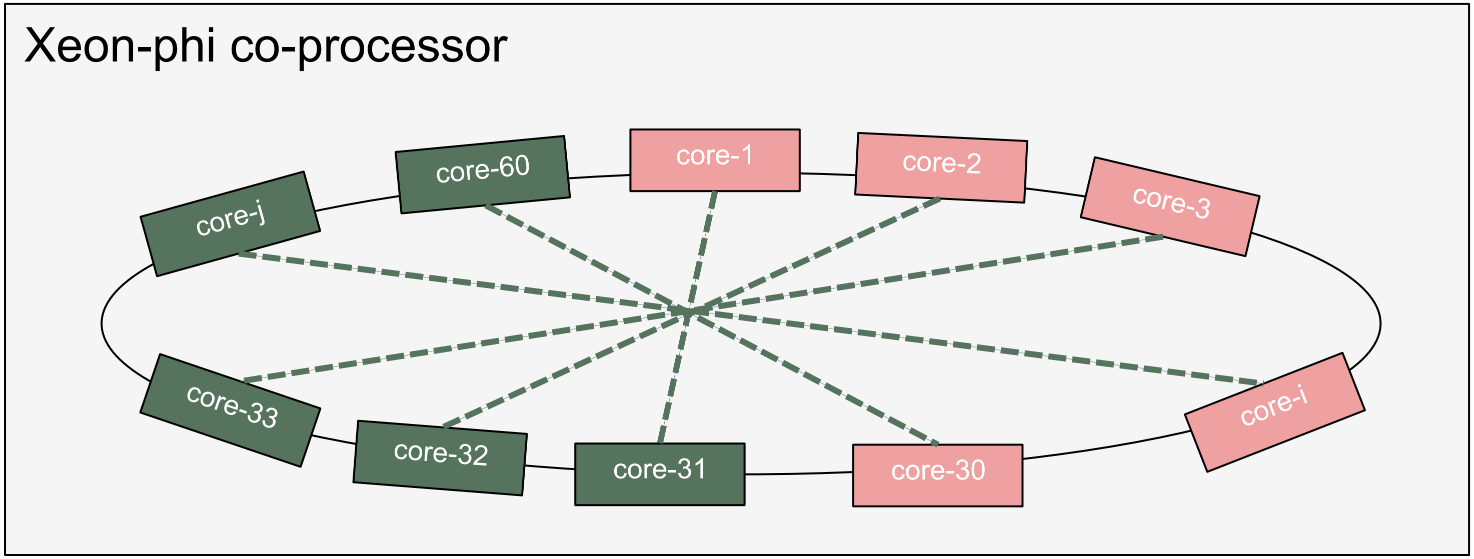
Experimental Setup

- TACC STAMPEDE node
  - Host processor
    - Xeon E5, 8 core, 2.7GHz, 32GB DDR3 RAM , Cent OS 6.3
  - Co processor
    - Xeon Phi, 61 cores, 1.1 GHz, 8GB DDR5 RAM
    - Linux based Busy Box OS (kernel version 2.6) / MPSS
    - Intel icc/mpicc Compiler – cross compile for Xeon Phi
- ***Presta*** benchmark*, "purple suite"*
- 2 types of experiments
  - Intra-node
    - Single STAMPEDE node (from 2 ranks to 240 ranks in one node)
    - All experiments run in *'Phi-Only'* mode – only in coprocessor
    - Benchmarks used – **Presta Stress Benchmark - *com / latency***
  - *Inter-node*
    - *Between nodes but* in *'Phi-Only'* mode
    - Communication via infiniband FDR interconnect

INDIANA UNIVERSITY
SCHOOL OF INFORMATICS AND COMPUTING
Bloomington

Intra-node Setup

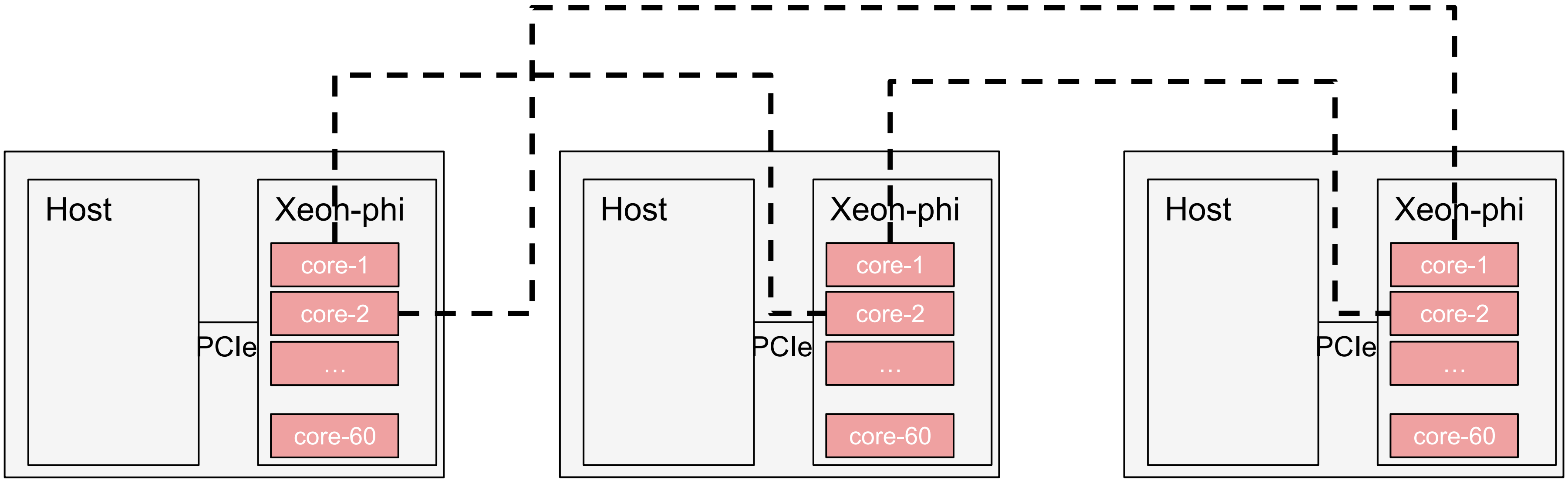
- Intra-node setup



- Each core is bound to a rank
  - All nodes tested have one Xeon Phi coprocessor
  - Rank pairs are formed in on opposite sides of ring interconnect

Inter-node Setup

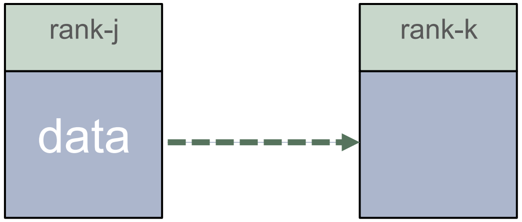
- Inter-node setup



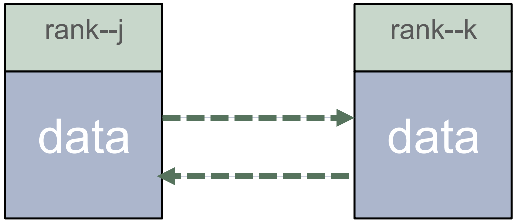
- Subset of cores/ranks from each node are selected
  - Communication in symmetric mode – Phi to Phi
  - RDMA with Infiniband

Presta *com* benchmark

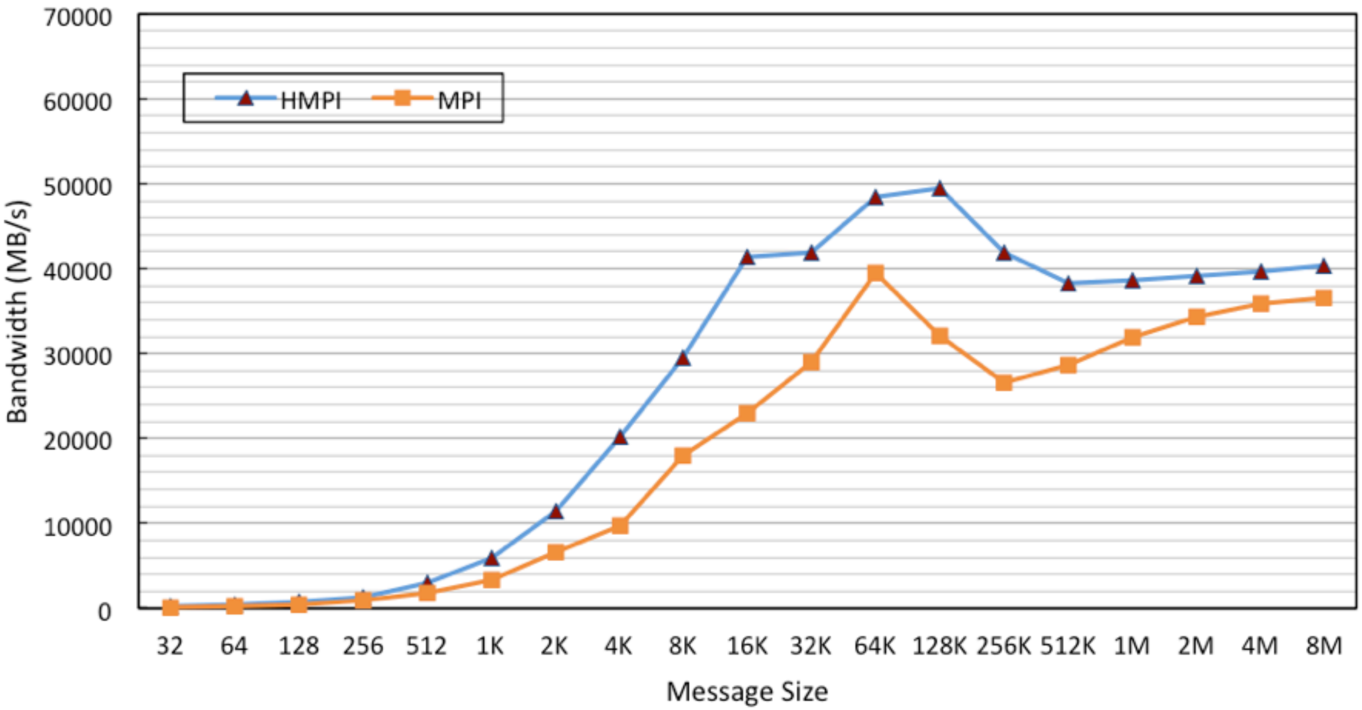
- 2 types of Presta ***com benchmark*** measurements
  - Uni-directional
    - One-way communication
    - MPI_Send / MPI_Recv



  - Bi-Directional
    - Two-way communication
    - MPI_Sendrecv
    - Full duplex – both sender and receiver transfer data at the same time
    - Generate rank pairs, similar to Uni-directional benchmark



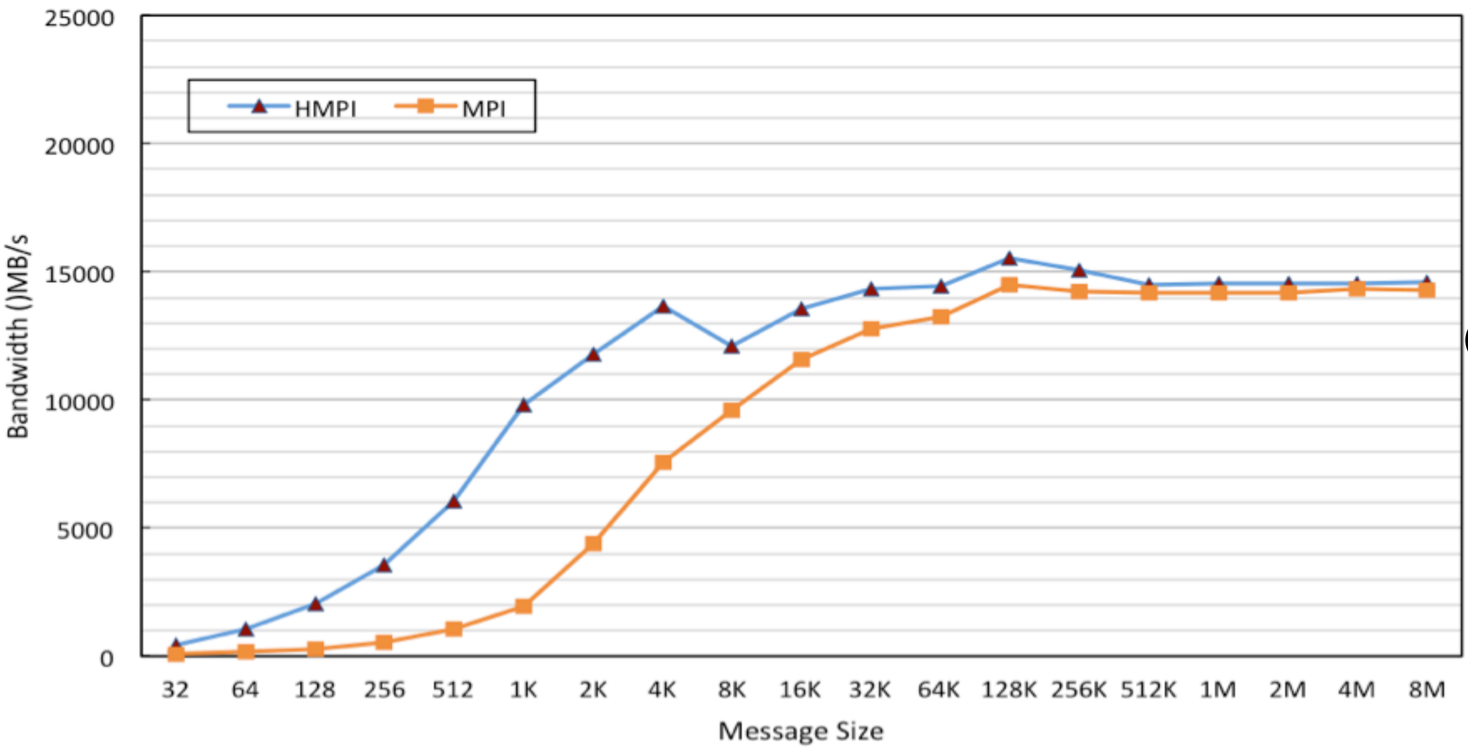
Intra vs Inter node Point to Point Communication



Intra-node BW
(60 ranks – 1 node)

- Intra node Hybrid MPI peak bandwidth ~50GB/s >> MPI peak bandwidth ~40GB/
- For Intra node communication with 60
  - For small messages , Hybrid MPI outperforms Intel MPI - speedup due to immediate protocol
  - Medium/Large messages → direct copy, synergistic protocol

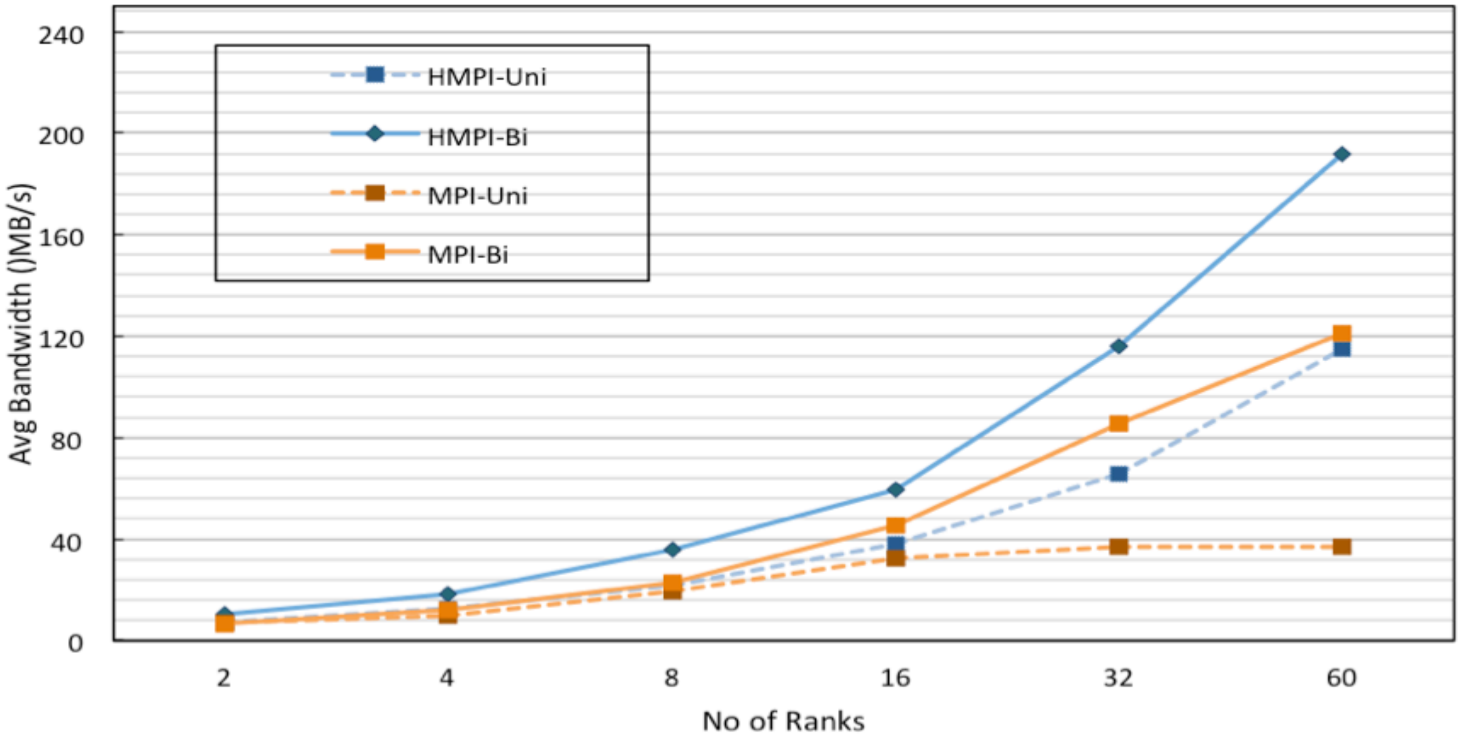
Inter node Point to Point Communication



Inter-node BW
(960 ranks – 16 nodes)

- Inter-node Bi-directional bandwidth
  - Smaller bandwidth difference
  - Due to noise in measurements, subtleties in message patterns, etc

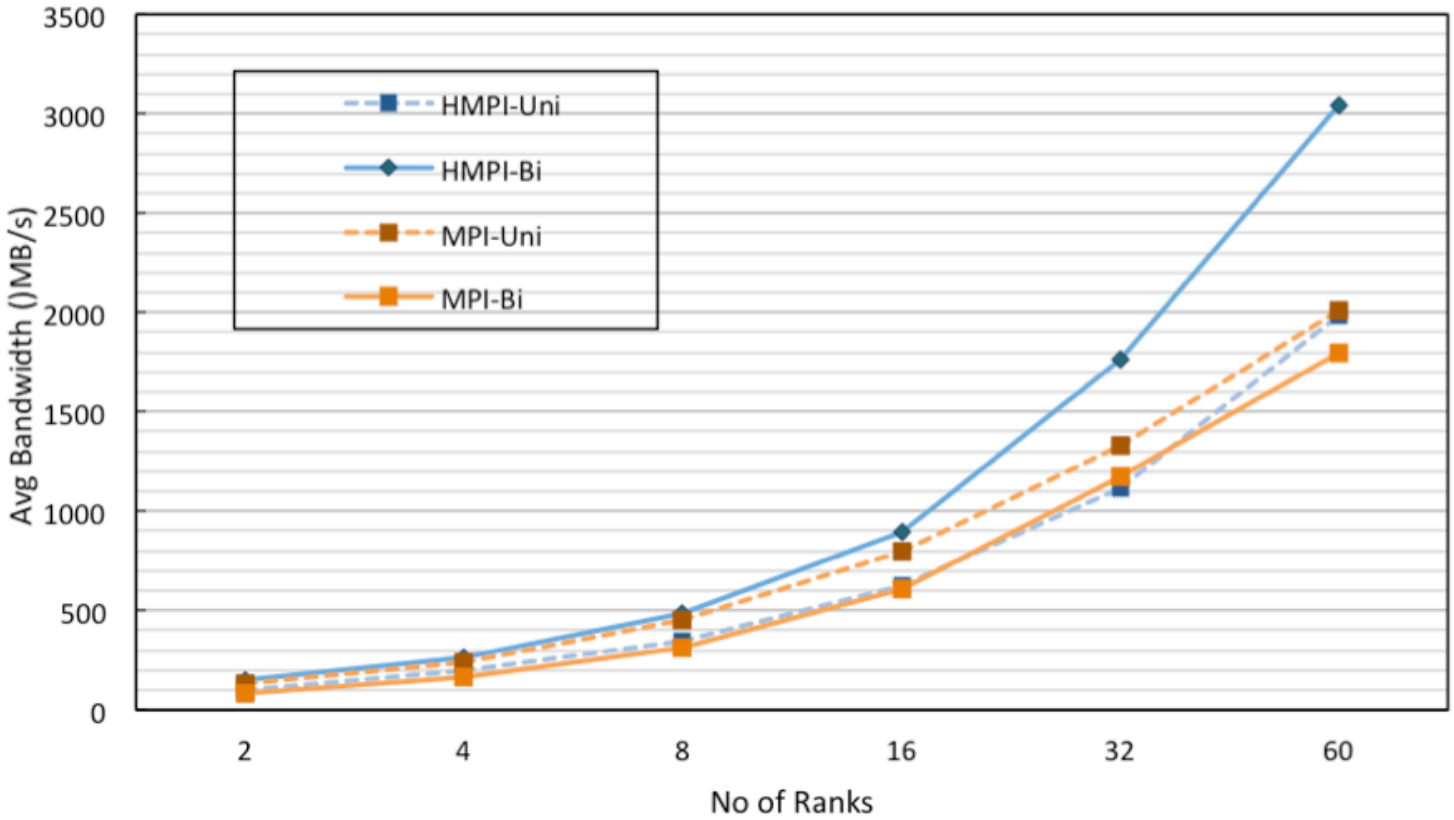
Intra-node Message Size Specific - small



Message size
32 bytes

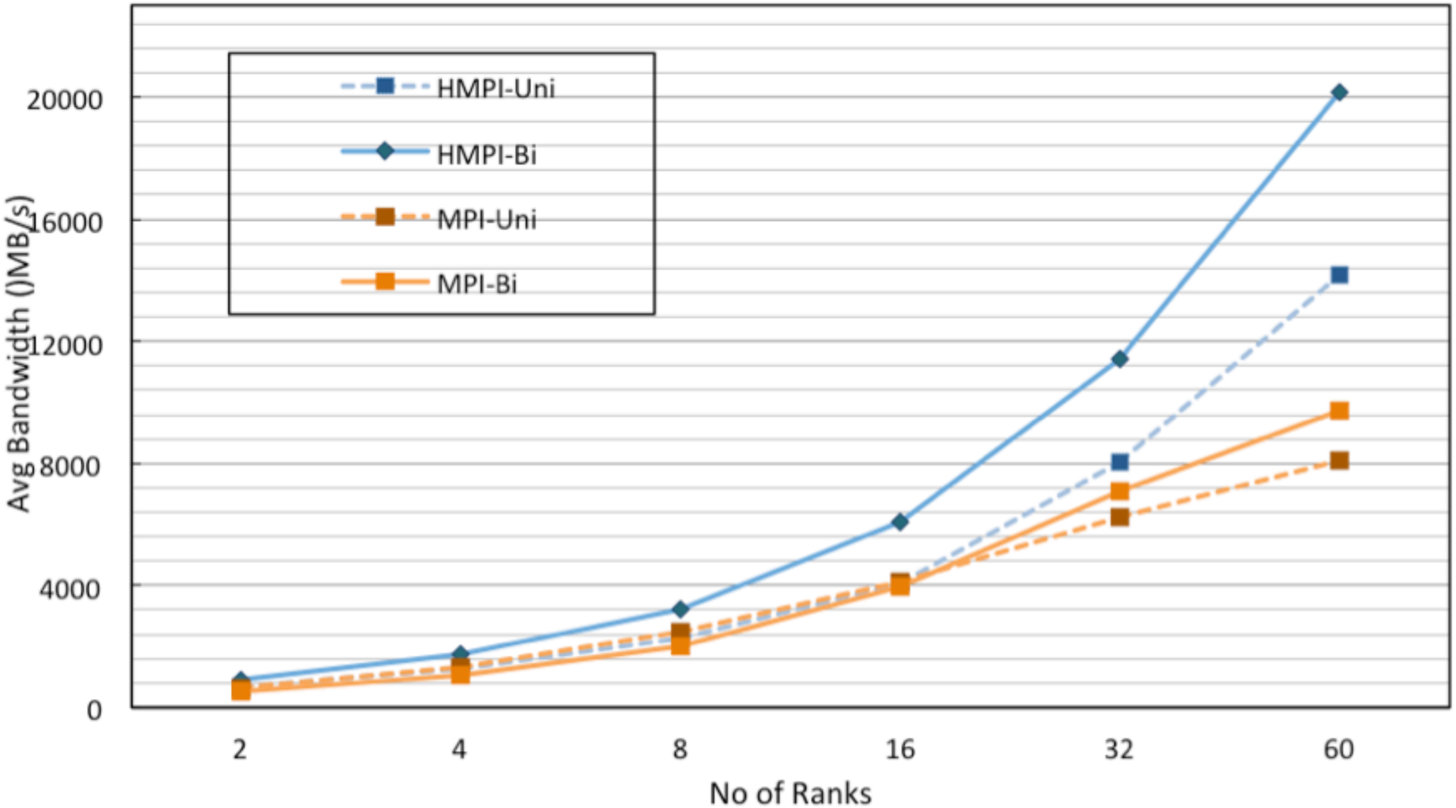
- Shows the effect of Hybrid MPI Immediate protocol
  - fast copying due to temporal locality
- Message size (32 bytes) fit a cache line on Xeon Phi core
- Hybrid MPI bi-directional benchmark outperforms others types for all ranks

Intra-node Message Size Specific - small



Message size
512 bytes

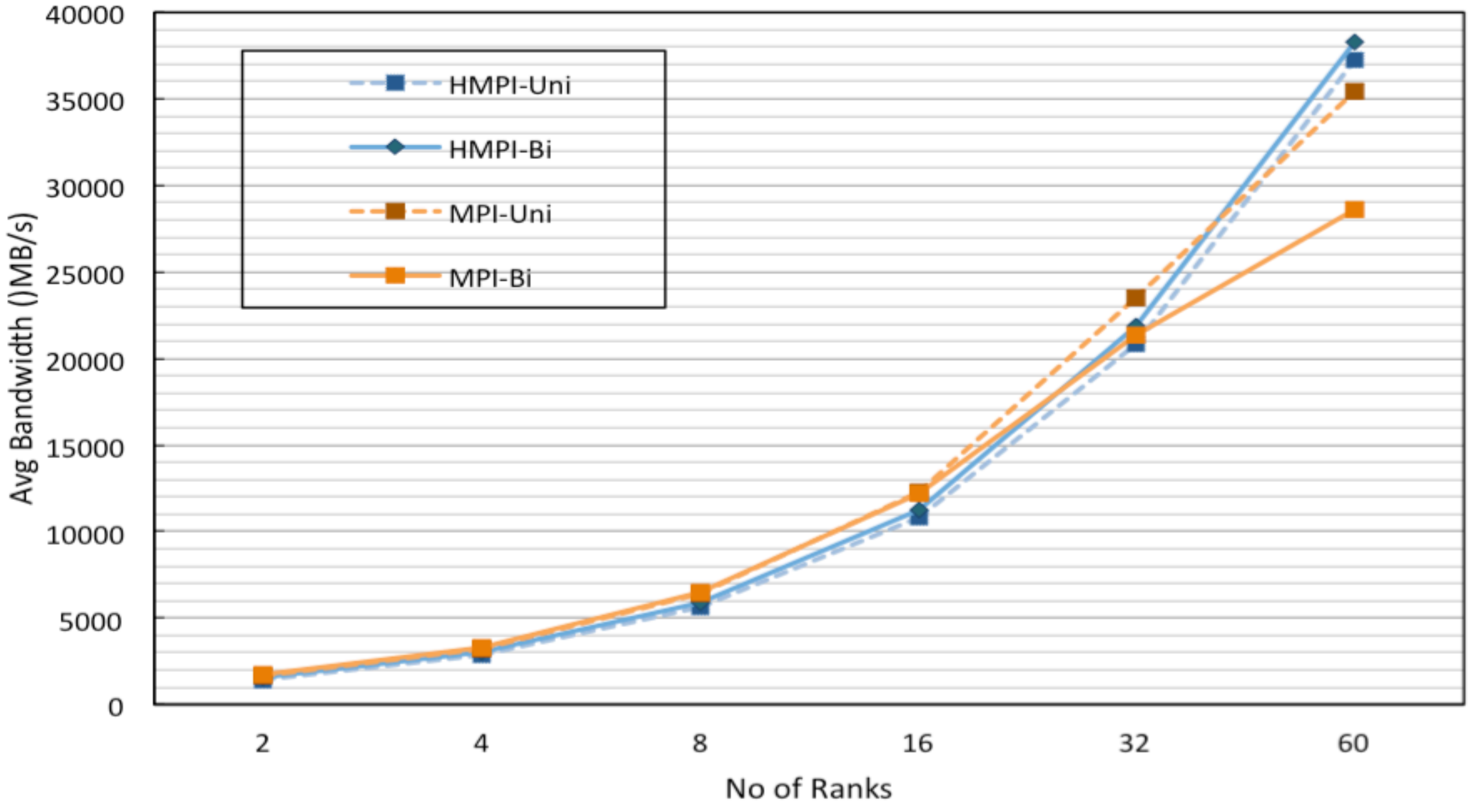
Intra-node Message Size Specific - medium



Message size
4 KB

- Hybrid MPI direct protocol
- Both Hybrid MPI's Uni-directional and Bi-directional transfers performs well over Intel MPI
  - Bi-directional BW >> Uni-directional

Intra-node Message Size Specific - large



Message size 512 KB

- Positive impact of Hybrid MPI ‘s synergistic protocol visible when number of ranks are 60
- For 512KB messages → 39GB/s peak BW, but 8MB is even better
- For 8MB messages → 50GB/s peak BW

Intra-node Message Size Specific Performance

- Bandwidth increases rapidly with the number of ranks
  - More cores are engaged in active data transfer
  - More memory Load/Store requests dispatched to controllers
  - Prefetching and cache coherence effects during transfer
  - More activity implies higher aggregated bandwidth

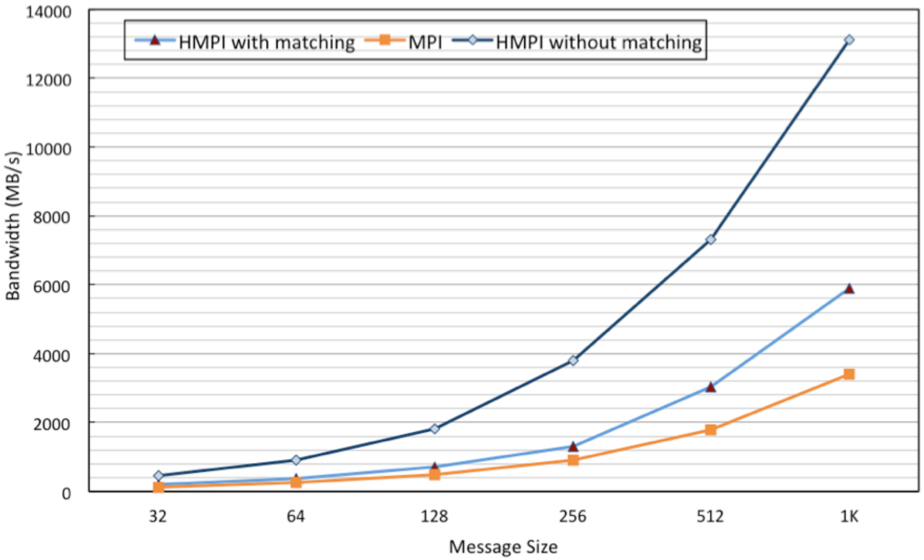
- In general for Medium/Large Messages, Bi-directional BW > Uni-directional BW
  - Hybrid MPI Peak Bi-directional BW ~50GB/s vs Intel MPI ~ 32GB/s - message size 128K 60 ranks
  - At synergistic transfer – multiple pairs of ranks can use multiple channels (on the ring interconnect) for simulataneous memcpy() in both directions

A Benchmark Without Message Matching

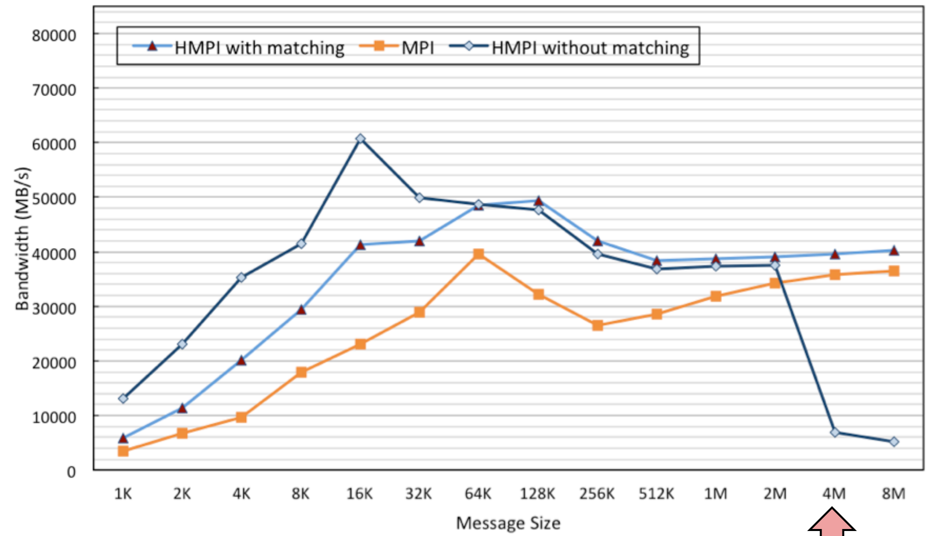
- Experimentally controlled to measure cost of message matching in MPI
  - Upper limit on Bandwidth and latency

- Algorithm
  - Initialize a shared memory pool to store source and destination memory pointers for messages
    - Use the extended heap of Hybrid MPI for shared access
  - Presta com benchmark with MPI message matching replaced by atomic synchronization
    - All Hybrid MPI protocols (direct, immediate, synergistic) in-lined in the benchmark
    - Use atomic spin locks (ie:- *sync_bool_compare_and_swap* ()) to synchronize between sender and receiver – synchronize sender/receiver → next iteration

A Benchmark Without Message Matching (contd..)



a) Small messages (<= 1KB)



b) Large messages (>= 1KB)

- Too much strain on memory sub system when → message size >> cache
  - saturates memory channels/interconnect quickly
- Peak BW of ~61 GB/s w/o message matching vs ~50 GB/s regular mode
  - 35% overhead for message matching at peak

INDIANA UNIVERSITY
SCHOOL OF INFORMATICS AND COMPUTING
Bloomington

Agenda

- Xeon Phi Platform
- Traditional MPI Design
- Hybrid MPI
  - A Shared Heap
  - Communication
- Experimental Evaluation
  - Micro-benchmarks
  - Applications
- Hybrid MPI Highlights
- Towards Hybrid MPI Future

Application Benchmarks

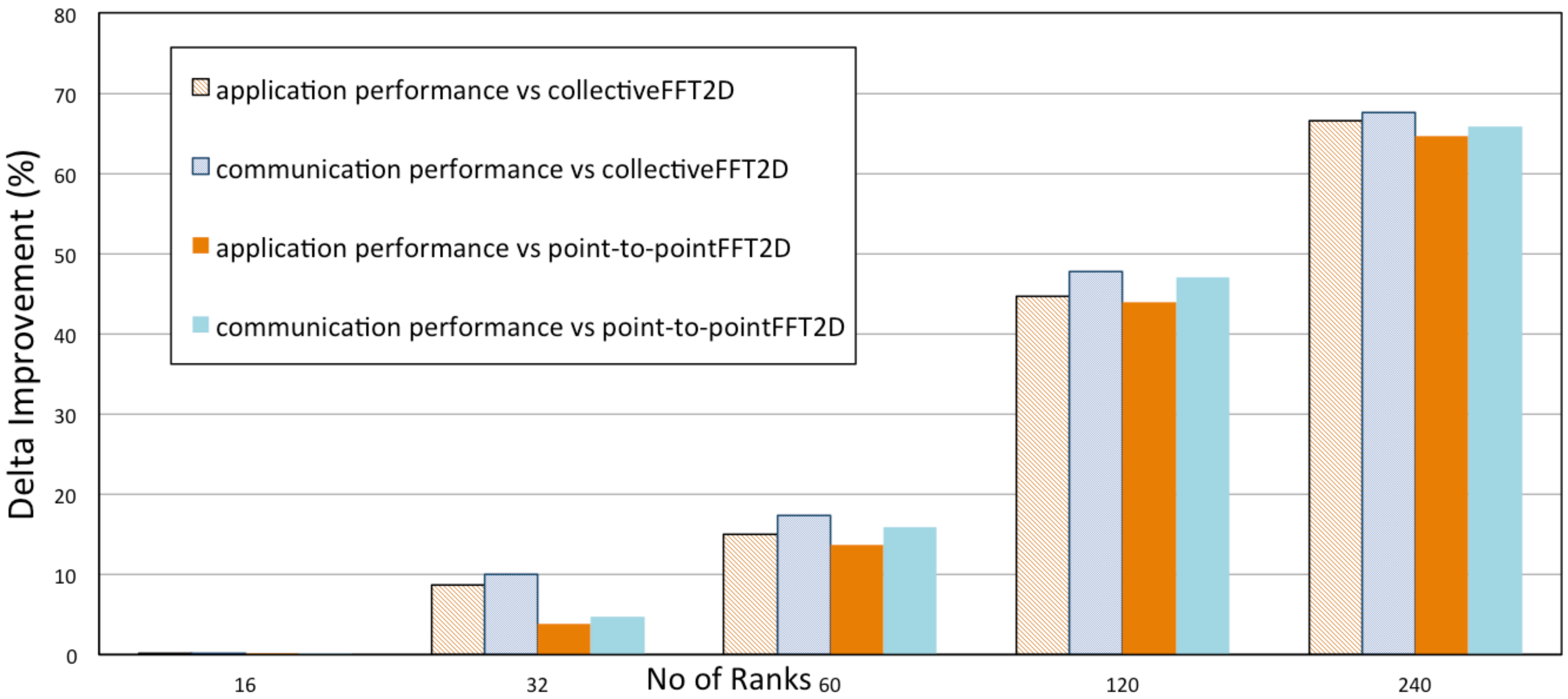
- FFT2D Application
  - Representative benchmark developed by T. Hoefler and S. Gottlieb
    - Implements a simple parallel FFT (Fast Fourier) on a 2D array
    - Uses FFTW library (developed by M.I.T.) for 1-d decomposition

- Application performance based on FFT2D variants
  - FFT2D collective
    - Original MPI collective based implementation
    - Communication with MPI_Alltoall, MPI_Scatter, MPI_Gather ,etc
  - FFT2D Point to point
    - Since Hybrid MPI implements only Point to Point primitives → transform collectives to MPI_Send/Recv/Isend/Irecv/Wait pattern/s

- Performance measurements
  - Application time – time to complete the program
  - Comm time – time spent on the data exchange between ranks

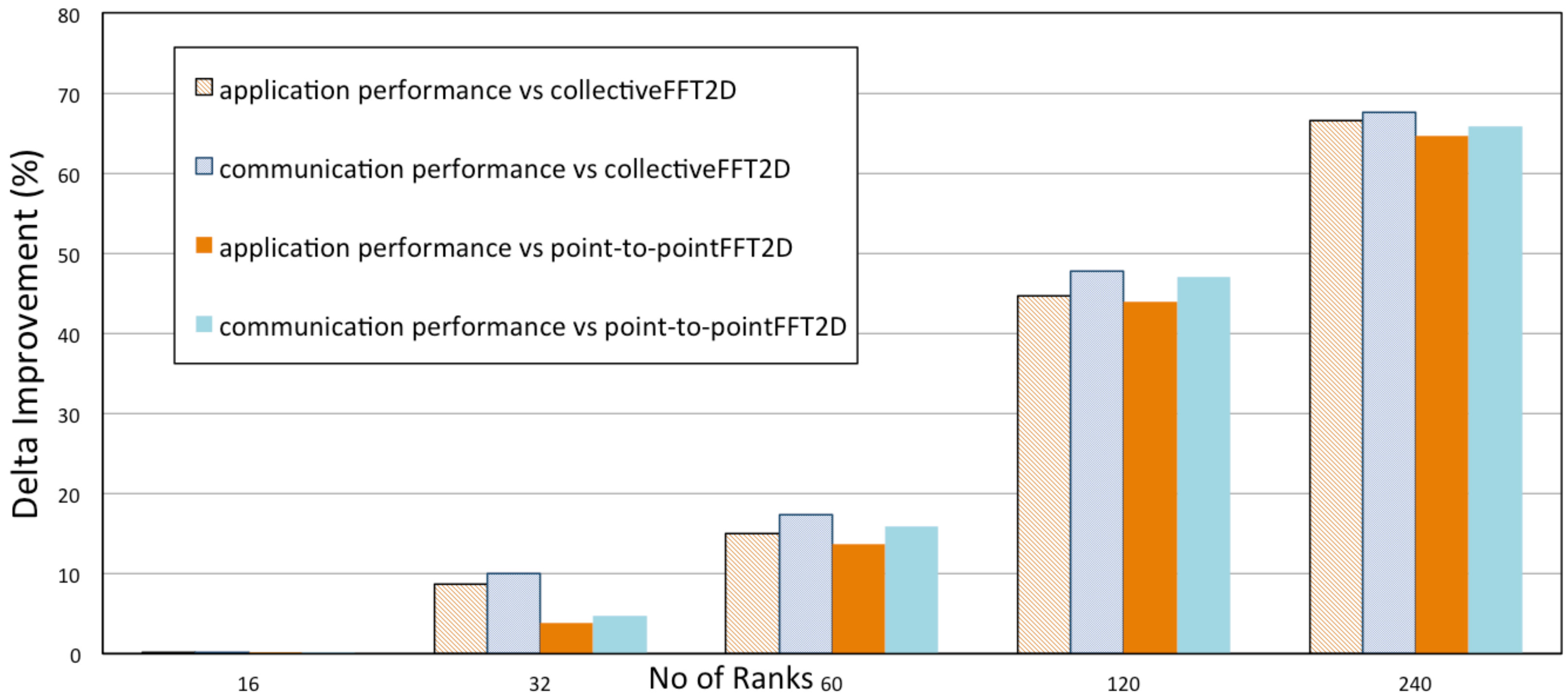
INDIANA UNIVERSITY
SCHOOL OF INFORMATICS AND COMPUTING
Bloomington

FFT2D Benchmark Intra-node



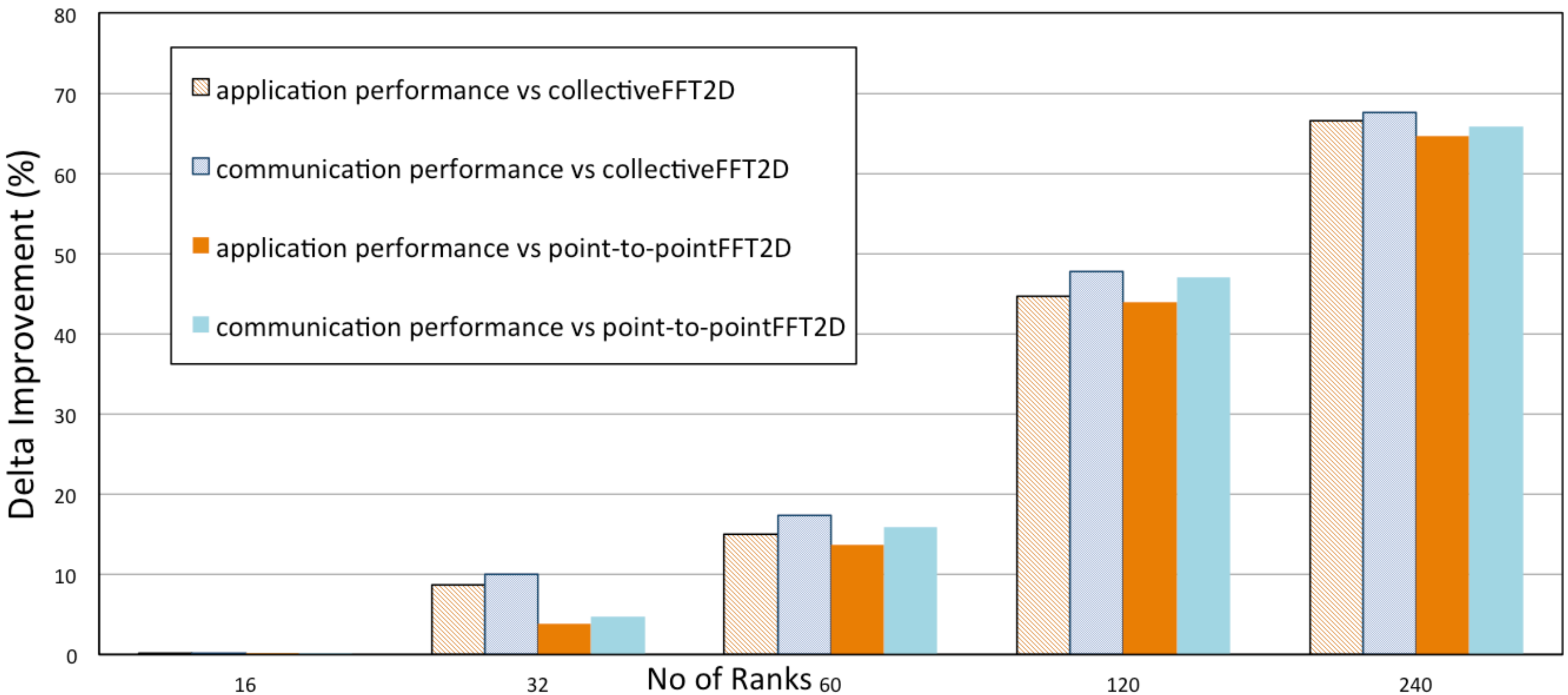
- Delta Improvement $= \dfrac{\text{Intel MPI time} - \text{Hybrid MPI time}}{\text{Intel MPI time}}\ \%$

FFT2D Benchmark Intra-node (contd..)



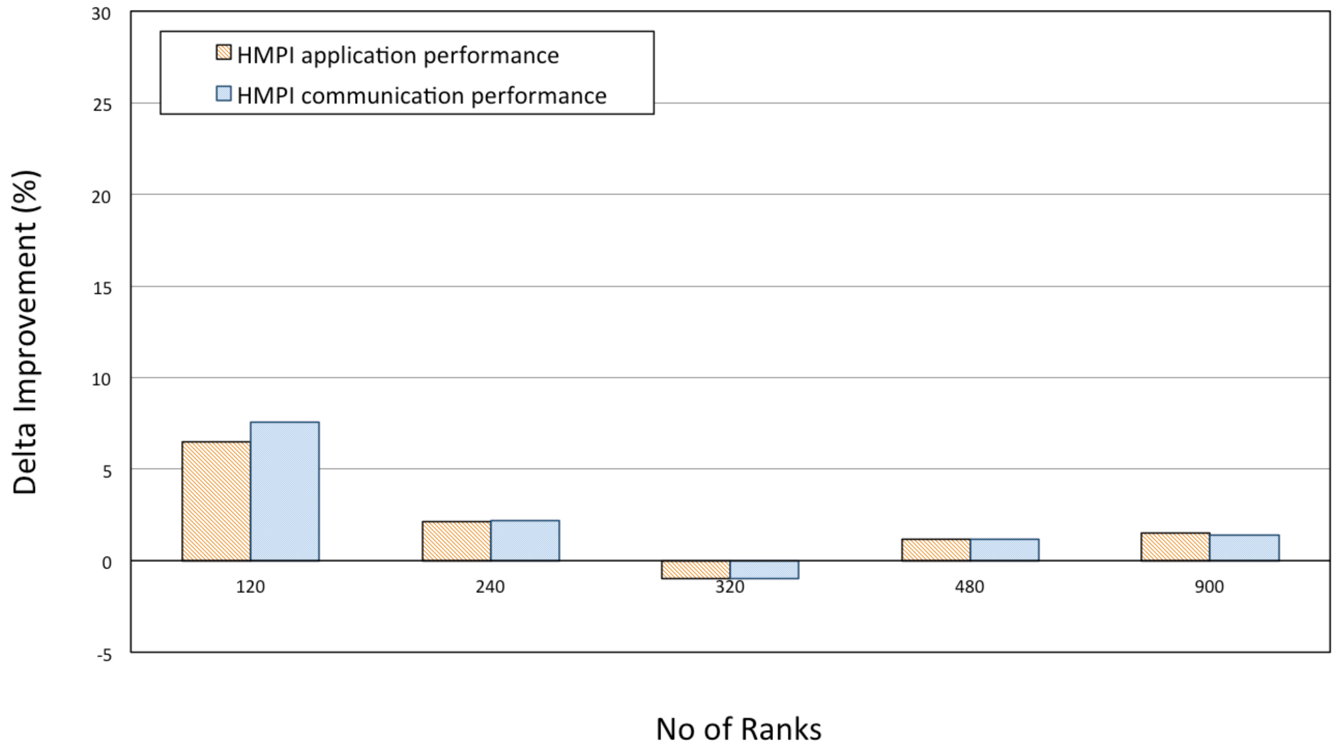
- Up to 240 ranks on phi (using 4 Hardware Threads per core)
- [app/comm]***-relative to point-to-pointFFT2D*** – Intel MPI baseline taken as modified point to point benchmark
- [app/comm]***-relative to collectiveFFT2D*** – Intel MPI baseline taken as original collective based benchmark

FFT2D Benchmark Intra-node (contd..)



- Considerable improvement in operational times
  - 5% to 66% - communication time, 4% to 65% - application time
  - Higher on phi ranks → higher improvement
  - ranks ≤ 16 → zero improvement
- Data don't show significant difference relative to point-to-point OR collective baselines – doesn't affect validity with P2P version

FFT2D Benchmark Inter-node



- Up to 900 ranks spanning 30 nodes
- Internode improvement/bottleneck is marginal – network overhead
  - Hybrid MPI delegates inter-node communication to underlying MPI layer
  - 6% improvement for 120 ranks → noise or other factors

Hybrid MPI Highlights

- Hybrid MPI highlights
  - Extremely high throughput via shared memory and single/zero copy techniques
    - 50 GB/s peak BW measurements
    - Overall significant improvements for all message sizes (use of Hybrid MPI protocols –immediate/direct/synergistic)

Message size	Improvement
Small (< 512b)	12% - 68%
Medium (512b – 8KB)	45% - 72%
Large (> 8KB)	65%

  - Results show improvement In FFT2D application and communication time
    - Upto 65% communication time improvement
  - Higher the number of ranks, higher improvement gained by Hybrid MPI

Towards a Hybrid MPI Future

- Efficient use of Xeon Phi cores and memory channels
  - Throughput proportional to number of cores used
    - Ranks ↑ → Bandwidth ↑
  - Achieve higher throughput via balancing the communication load between the available cores
- Optimizing message matching
  - At peak 35% time spent on matching on coming receives
  - Efficient data structures and algorithms to reduce matching overhead
- Collectives and Inter-node implementation
  - Currently Hybrid MPI does not support collectives or native inter-node mode
  - Use available technologies (ie:- SCIF, IB, etc) to improve off Phi bandwidth and latency