MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Designing High-Performance and Scalable Collectives for the Many-core Era: The MVAPICH2 Approach

IXPUG ’18 Presentation

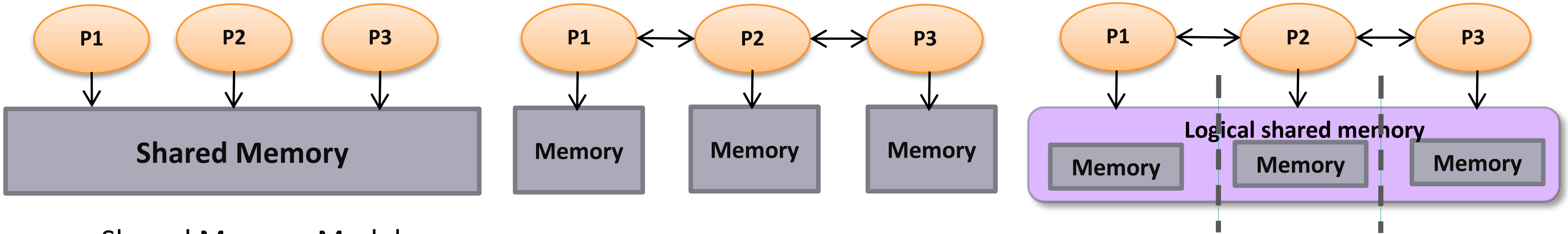
Presenter: **Jahanzeb Hashmi**

S. Chakraborty, M. Bayatpour, J. Hashmi, H. Subramoni and DK Panda

The Ohio State University

E-mail: {chakraborty.52,bayatpour.1,hashmi.29,subramoni.1,panda.2}@osu.edu

Parallel Programming Models Overview



Shared Memory Model

SHMEM, DSM

Distributed Memory Model

MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)

Global Arrays, UPC, Chapel, X10, CAF, …

- Programming models provide abstract machine models
- Models can be mapped on different types of systems
  - e.g. Distributed Shared Memory (DSM), MPI within a node, etc.
- Programming models offer various communication primitives
  - Point-to-point (between pair of processes/threads)
  - Remote Memory Access (directly access memory of another process)
  - **Collectives (group communication)**

Supporting Programming Models for Multi-Petaflop and Exaflop Systems: Challenges

Application Kernels/Applications

Middleware

Programming Models

MPI, PGAS (UPC, Global Arrays, OpenSHMEM), CUDA, OpenMP, OpenACC, Cilk, Hadoop (MapReduce), Spark (RDD, DAG), etc.

Communication Library or Runtime for Programming Models

- Point-to-point Communication
- Collective Communication
- Energy-Awareness
- Synchronization and Locks
- I/O and File Systems
- Fault Tolerance

Networking Technologies (InfiniBand, 40/100GigE, Aries, and Omni-Path)

Multi-/Many-core Architectures

Accelerators (GPU and FPGA)

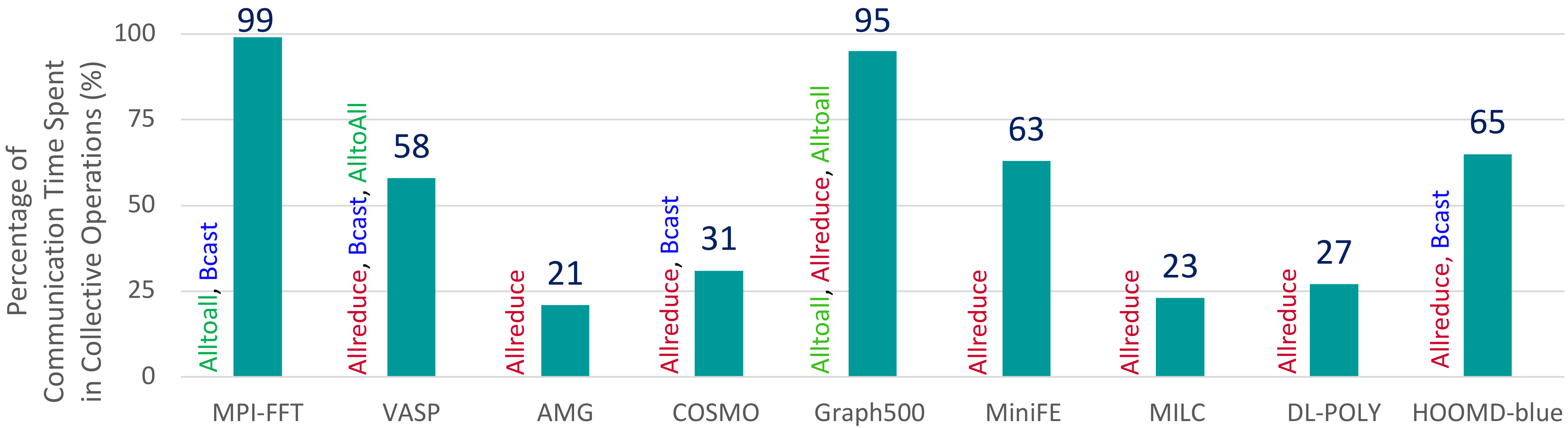
Co-Design Opportunities and Challenges across Various Layers

Performance

Scalability

Resilience

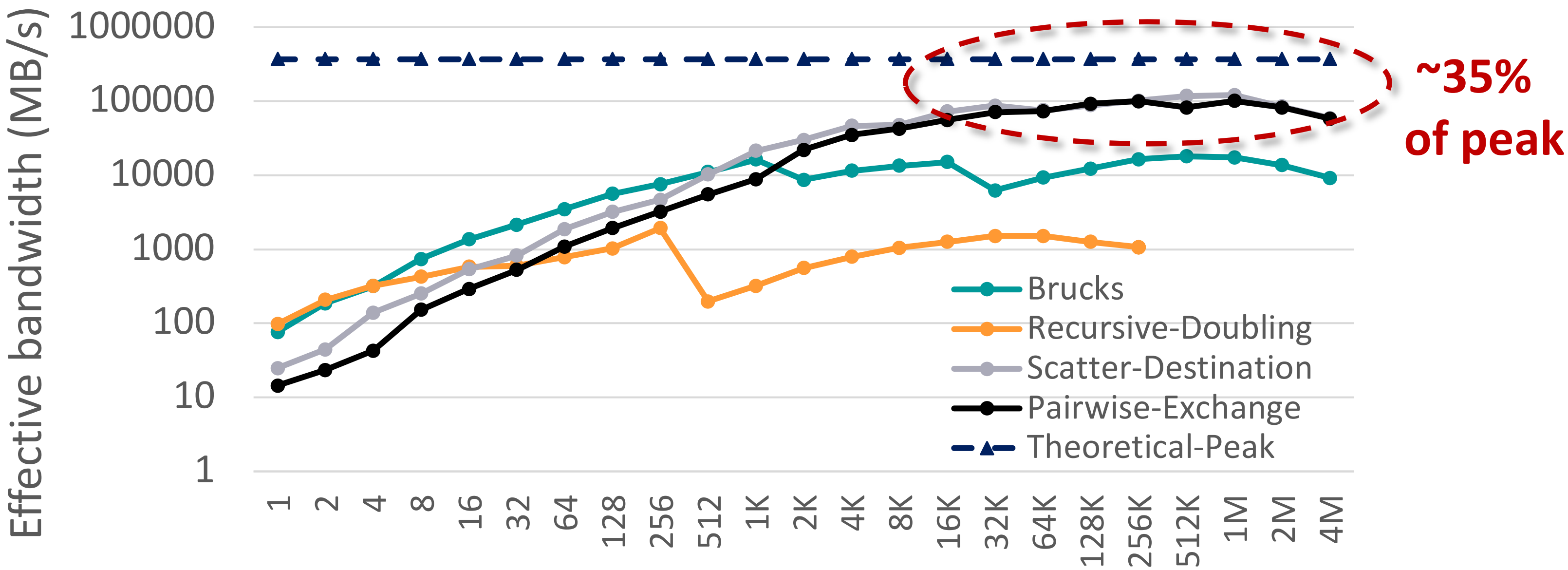
Why Collective Communication Matters?



- HPC Advisory Council (HPCAC) MPI application profiles
- Most application profiles showed majority of time spent in collective operations
- Optimizing collective communication directly impacts scientific applications leading to accelerated scientific discovery

http://www.hpcadvisorycouncil.com

Are Collective Designs in MPI ready for Manycore Era?



Alltoall Algorithms on single KNL 7250 in Cache-mode on 64 MPI processes using MVAPICH2-2.3rc1

Why different algorithms of even a dense collective such as Alltoall do not achieve theoretical peak bandwidth offered by the system?

Broad Challenges due to Architectural Advances

- *Exploiting high concurrency and high bandwidth offered by modern architectures*
- *Designing "zero-copy" and "contention-free" Collective Communication*
- *Efficient hardware offloading for better overlap of communication and computation*

How does MVAPICH2 as an MPI library tackles these challenges and provide optimal collective designs for emerging multi-/many-cores?

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  - MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.1), Started in 2001, First version available in 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2011
  - Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  - Support for Virtualization (MVAPICH2-Virt), Available since 2015
  - Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  - Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  - **Used by more than 2,875 organizations in 86 countries**
  - **More than 464,000 (> 0.46 million) downloads from the OSU site directl**
  - Empowering many TOP500 clusters (Nov '17 ranking)
    - **1st, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China**
    - 9th, 556,104 cores (Oakforest-PACS) in Japan
    - 12th, 368,928-core (Stampede2) at TACC
    - 17th, 241,108-core (Pleiades) at NASA
    - 48th, 76,032-core (Tsubame 2.5) at Tokyo Institute of Technology
  - Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  - **http://mvapich.cse.ohio-state.edu**
- Empowering Top500 systems for over a decade

Architecture of MVAPICH2 Software Family

High Performance Parallel Programming Models

- Message Passing Interface (MPI)
- PGAS (UPC, OpenSHMEM, CAF, UPC++)
- Hybrid --- MPI + X (MPI + PGAS + OpenMP/Cilk)

High Performance and Scalable Communication Runtime

Diverse APIs and Mechanisms

- Point-to-point Primitives
- Collectives Algorithms
- Job Startup
- Energy-Awareness
- Remote Memory Access
- I/O and File Systems
- Fault Tolerance
- Virtualization
- Active Messages
- Introspection & Analysis

Support for Modern Networking Technology (InfiniBand, iWARP, RoCE, Omni-Path)

Transport Protocols: RC, XRC, UD, DC

Modern Features: UMR, ODP, SR-IOV, Multi Rail

Support for Modern Multi-/Many-core Architectures (Intel-Xeon, OpenPower, Xeon-Phi, ARM, NVIDIA GPGPU)

Transport Mechanisms: Shared Memory, CMA, IVSHMEM, XPMEM*

Modern Features: MCDRAM*, NVLink*, CAPI*

* Upcoming

MVAPICH2 Software Family

High-Performance Parallel Programming Libraries	
MVAPICH2	Support for InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE
MVAPICH2-X	Advanced MPI features, OSU INAM, PGAS (OpenSHMEM, UPC, UPC++, and CAF), and MPI+PGAS programming models with unified communication runtime
MVAPICH2-GDR	Optimized MPI for clusters with NVIDIA GPUs
MVAPICH2-Virt	High-performance and scalable MPI for hypervisor and container based HPC cloud
MVAPICH2-EA	Energy aware and High-performance MPI
MVAPICH2-MIC	Optimized MPI for clusters with Intel KNC
Microbenchmarks	
OMB	Microbenchmarks suite to evaluate MPI and PGAS (OpenSHMEM, UPC, and UPC++) libraries for CPUs and GPUs
Tools	
OSU INAM	Network monitoring, profiling, and analysis for clusters with MPI and scheduler integration
OEMT	Utility to measure the energy consumption of MPI applications

Agenda

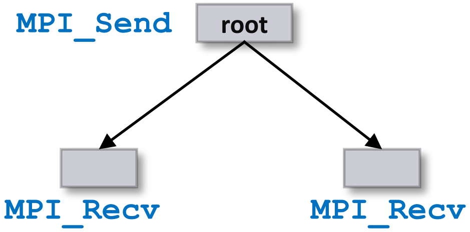
- **Exploiting high concurrency and high bandwidth offered by modern architectures for MPI collectives design**
  - **Point-to-point**
  - **Direct Shared-memory**
  - **Data Partitioned Multi-Leader (DPML)**
- Designing “zero-copy” and “contention-free” Collective Communication
  - Contention-aware designs
  - True zero-copy collectives
- Hardware offloading for better communication and computation overlap
  - SHARP based offloaded collectives
  - CORE-Direct based Non-blocking collectives

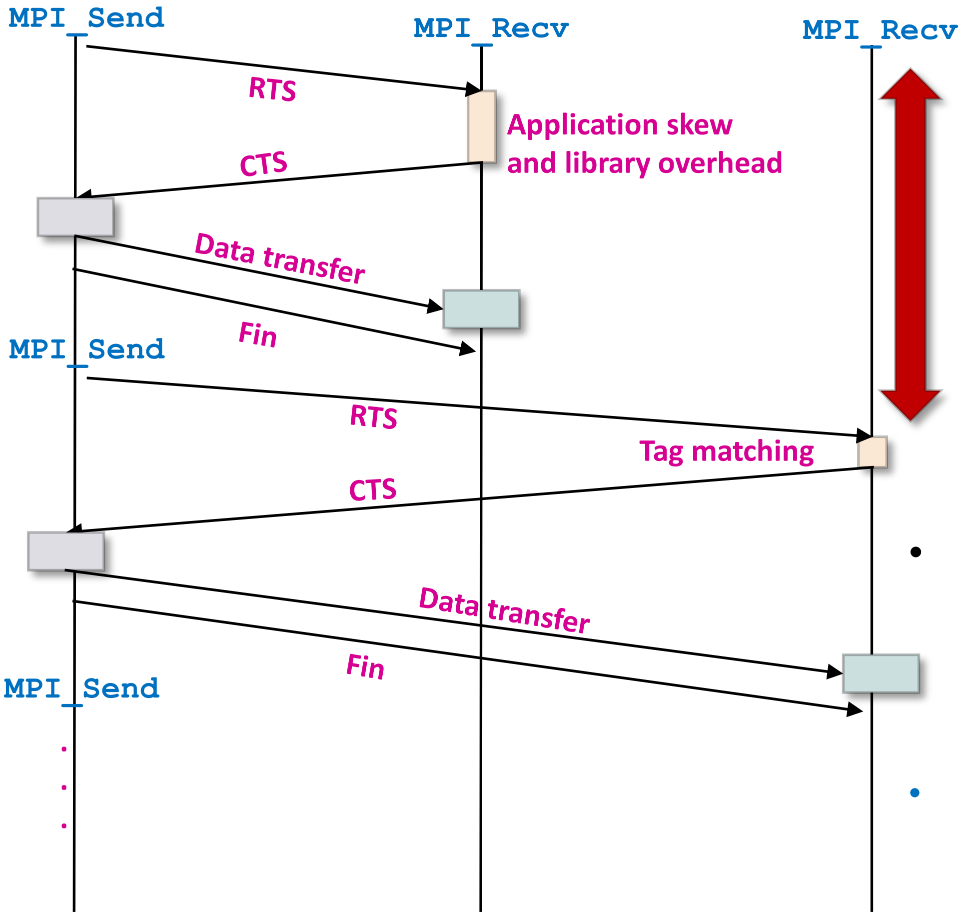
Collective Designs based on Point-to-point Primitives

- Commonly used approach in implementing collectives
- Easy to express algorithms in message passing semantics
- A naïve Broadcast could be a series of “*send*” operations from root to all the non-root processes
- Relies on the implementation of point to point primitives
- Limited by the overheads exposed by these primitives
  - Tag-matching
  - Rendezvous hand-shake

A Naïve Example of MPI_Bcast when using MPI_Send/MPI_Recv

```
// root = 0
// msg-size > eager-size
if (rank == 0) {
  for (i = 1 to n-1) {
    MPI_Send(buf,...i,...);
  }
} else {
  MPI_Recv(buf,...0,...);
}
```



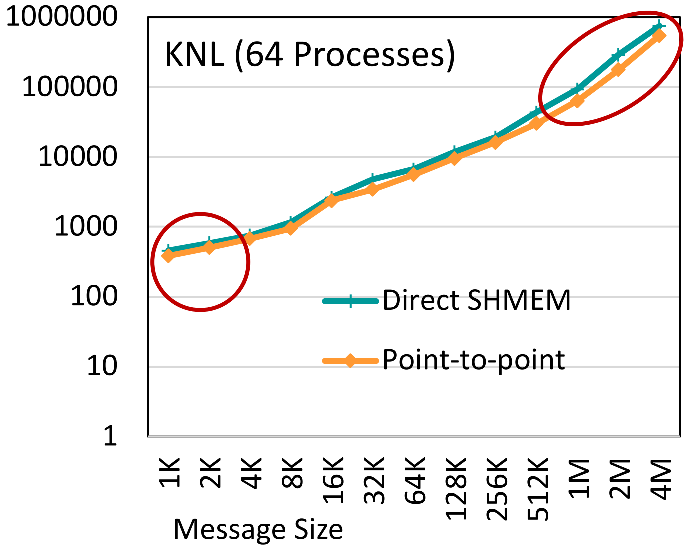


Receiver wasting CPU cycles waiting for Sender’s request

- Overheads of handshake for each rendezvous message transfer
- ***Is there a better way?***

Direct Shared Memory based Collectives

- A large shared-memory region
  - Collective algorithms are realized by shared-memory copies and synchronizations
  - Good performance for small message via exploiting cache locality
  - Avoid overheads associated with MPI point-to-point implementations
- **Requires one additional copy for each transfer**
- Performance degradations for large message communication
  - memcpy () is the dominant cost for large messages
- Most MPI libraries use some variant of Direct SHMEM collectives



Personalized All-to-All

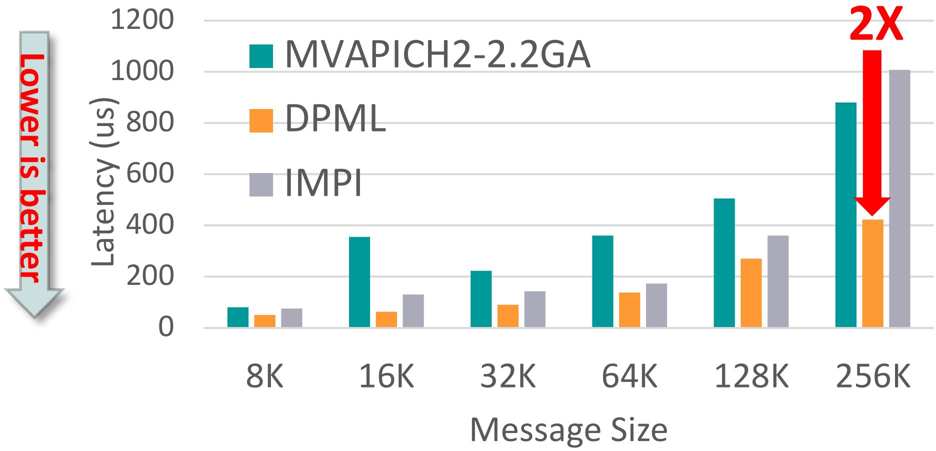
Reduction collectives perform even worse with SHMEM based design because of compute + memcpy

Data Partitioning based Multi-Leader (DPML) Designs

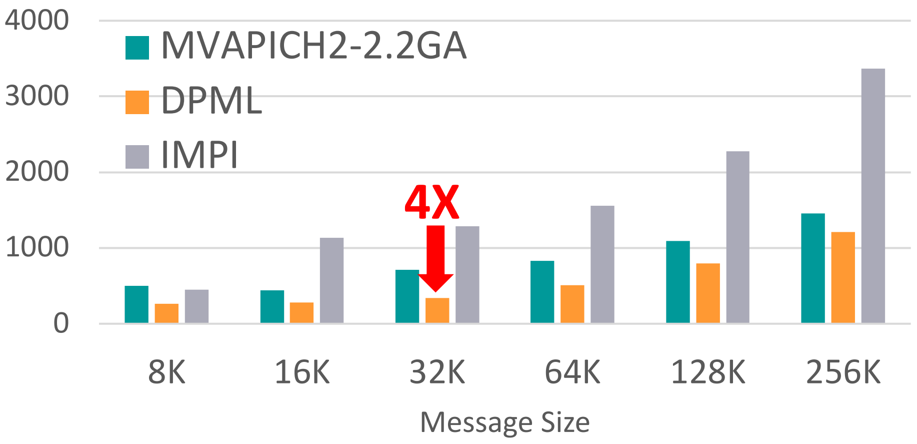
- Hierarchical algorithms delegate lot of computation on the ***"node-leader"***
  - Leader process responsible for inter-node reductions while intra-node non-root processes wait for the leader
- Existing designs for MPI_Allreduce do not take advantage of the vast parallelism available in modern multi-/many-core processors
- DPML - a new solution for MPI_Allreduce
- Takes advantage of the parallelism offered by
  - Multi-/many-core architectures
  - High throughput and high-end features offered by InfiniBand and Omni-Path
- Multiple partitions of reduction vectors for arbitrary number of leaders

M. Bayatpour, S. Chakraborty, H. Subramoni, X. Lu, and D. K. Panda, Scalable Reduction Collectives with Data Partitioning-based Multi-Leader Design, Supercomputing '17.

Performance of DPML MPI_Allreduce On Different Networks and Architectures



XEON + IB (64 Nodes, 28 PPN)

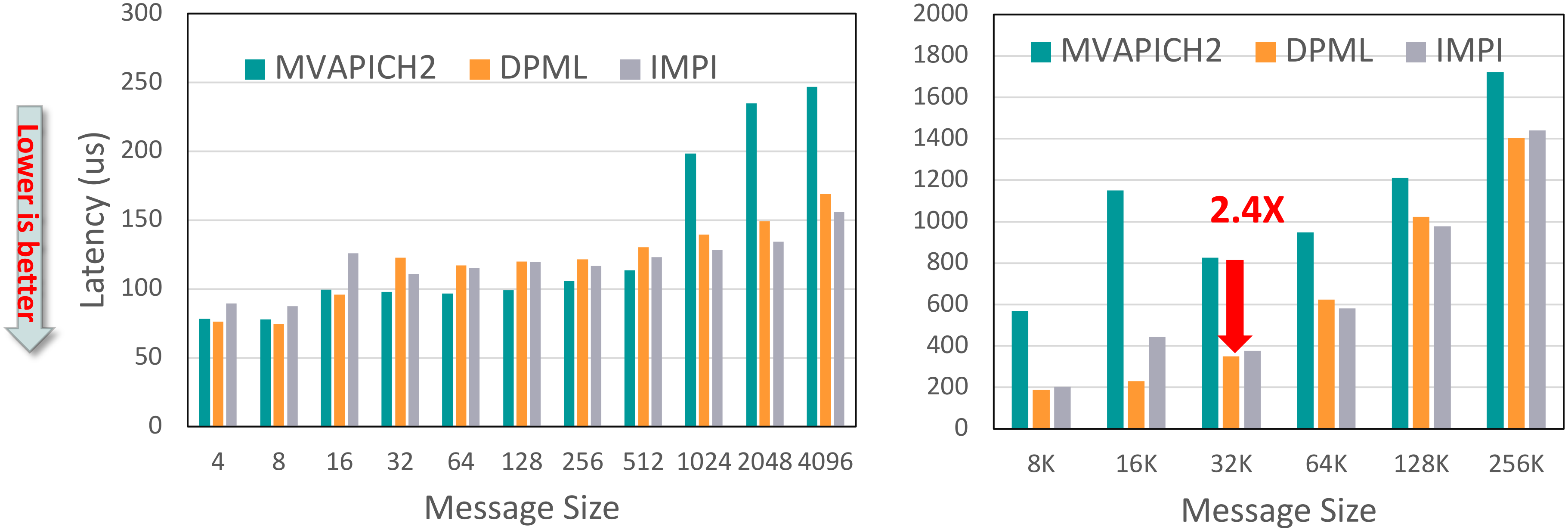


KNL + OmniPath (64 Nodes 64 PPN)

- 2X improvement of over MVAPICH2 at 256K
- Higher benefits of DPML as the message size increases

- Benefits of DPML sustained on KNL+OmniPath even at **4096** processes
- With 32K bytes, 4X improvement over MVAPICH2

Scalability of DPML Allreduce On Stampede2-KNL (10,240 Processes)

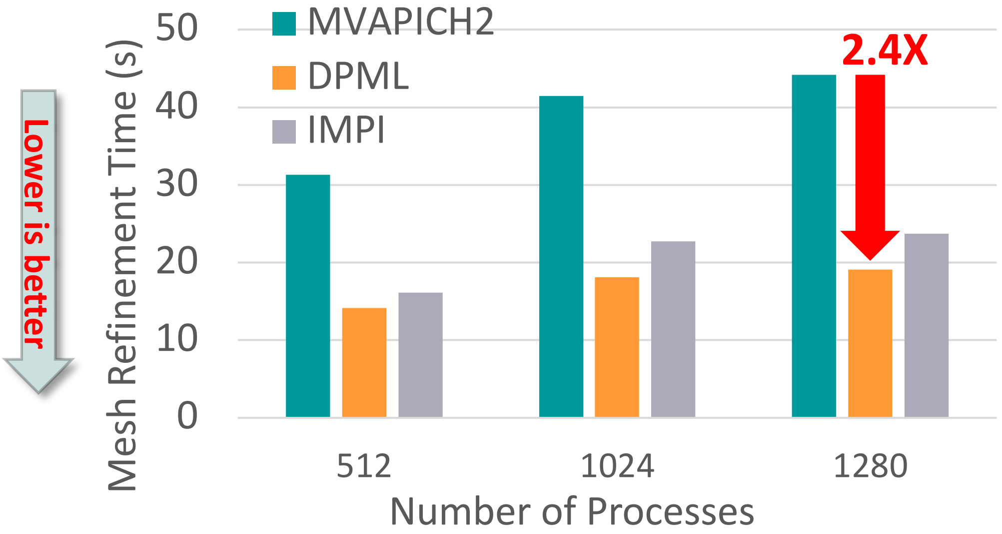


OSU Micro Benchmark (64 PPN)

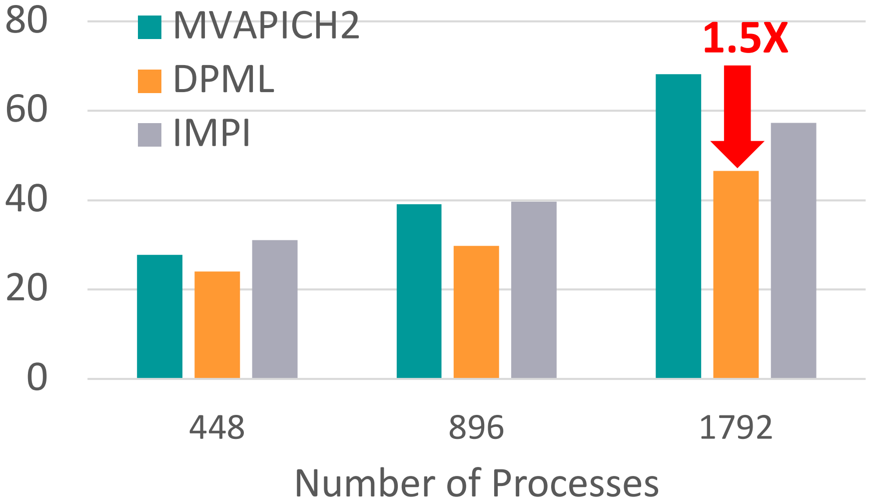
- For MPI_Allreduce latency with 32K bytes, DPML design can reduce the latency by 2.4X

Available in MVAPICH2-X 2.3b

Performance Benefits of DPML AllReduce on MiniAMR Kernel



KNL + Omni-Path (32 PPN)



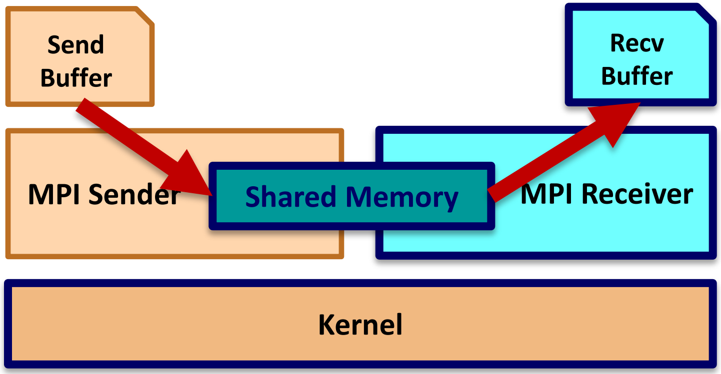
XEON + Omni-Path (28 PPN)

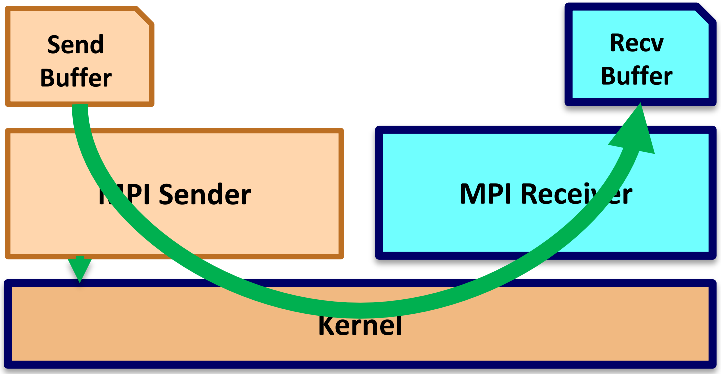
- For MiniAMR Application with 4096 processes, DPML can reduce the latency by 2.4X on KNL + Omni-Path cluster
- On XEON + Omni-Path, with 1792 processes, DPML can reduce the latency by 1.5X

Agenda

- Exploiting high concurrency and high bandwidth offered by modern architectures for MPI collectives design
  - Point-to-point
  - Direct Shared-memory
  - Data Partitioned Multi-Leader (DPML)
- **Designing "Zero-copy" and "contention-free" Collective Communication**
  - **Contention-aware designs**
  - **True zero-copy collectives**
- Hardware offloading for better communication and computation overlap
  - SHARP based offloaded collectives
  - CORE-Direct based Non-blocking collectives

How Kernel-assisted “Zero-copy” works?





Shared Memory – SHMEM

Requires two copies

No system call overhead

Better for Small Messages

Kernel-Assisted Copy

Requires single copy

System call overhead

Better for Large Messages

A Variety of Available “Zero”-Copy Mechanisms

	LiMIC	KNEM	CMA	XPMEM
Permission Check	Not Supported	Supported	Supported	Supported
Availability	Kernel Module	Kernel Module	Included in Linux 3.2+	Kernel Module
memcpy() invoked at	Kernel-space	Kernel-space	Kernel-space	User-space
memcpy() granularity	Page size	Page size	Page size	Any size

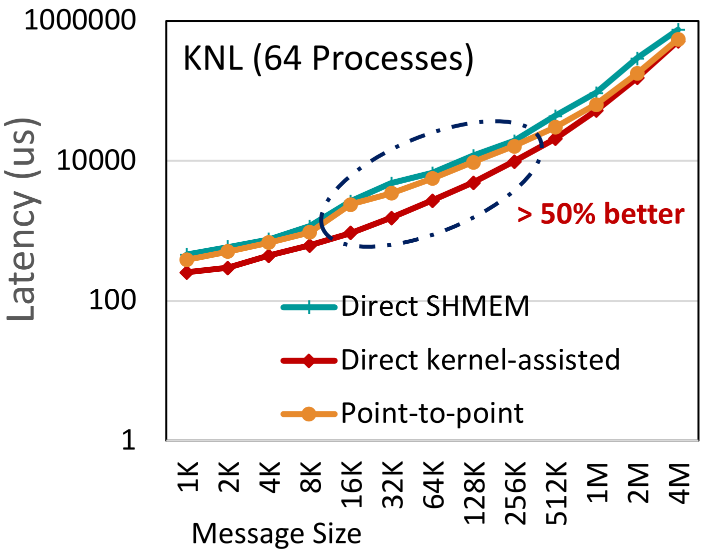
MPI Library Support

	LiMIC	KNEM	CMA	XPMEM
MVAPICH2	√	x	√	√ (upcoming release)
OpenMPI 2.1.0	x	√	√	√
Intel MPI 2017	x	x	√	x
Cray MPI	x	x	√	√

Cross Memory Attach(CMA) is widely supported kernel-assisted mechanism

Direct Kernel-assisted (CMA-based) Collectives

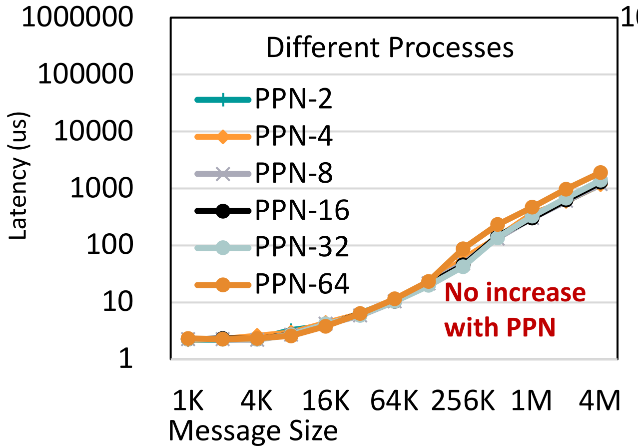
- Direct algorithm designs based on kernel-assisted zero-copy mechanism
  - *"Map"* application buffer pages inside kernel
  - Issue "*Put*" or "*Get*" operations directly on the application buffers
- Good performance for large messages
  - Avoid unnecessary copy overheads of SHMEM
- Performance depends on the communication pattern of the collective primitive
- Does not offer "zero-copy" for Reduction Collectives

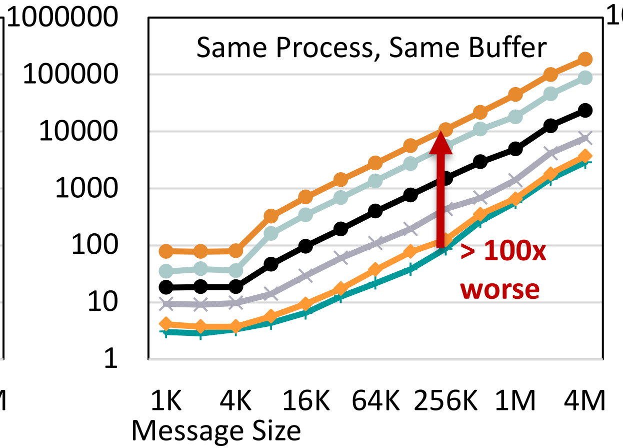


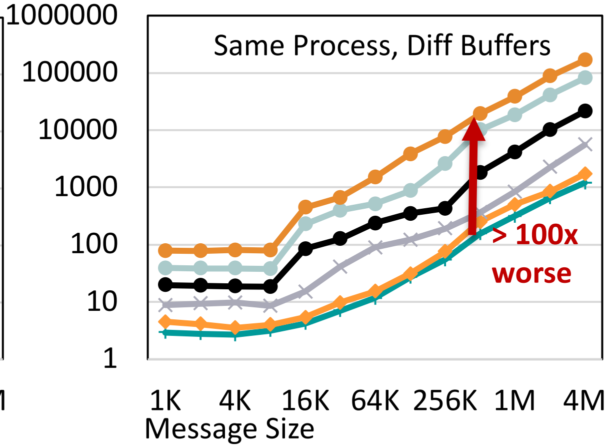
CMA based Personalized All-to-All

What about contention?

Impact of Collective Communication Pattern on CMA Collectives



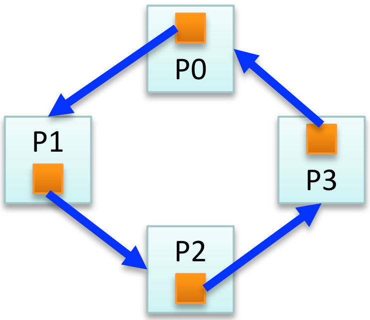


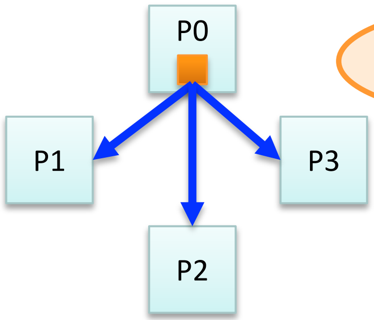


All-to-All – Good Scalability

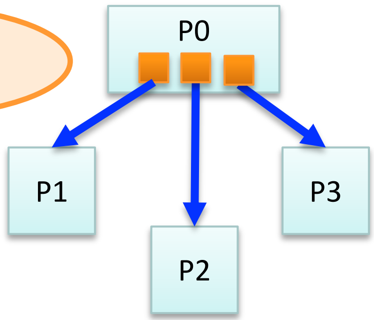
One-to-All - Poor Scalability

One-to-All – Poor Scalability

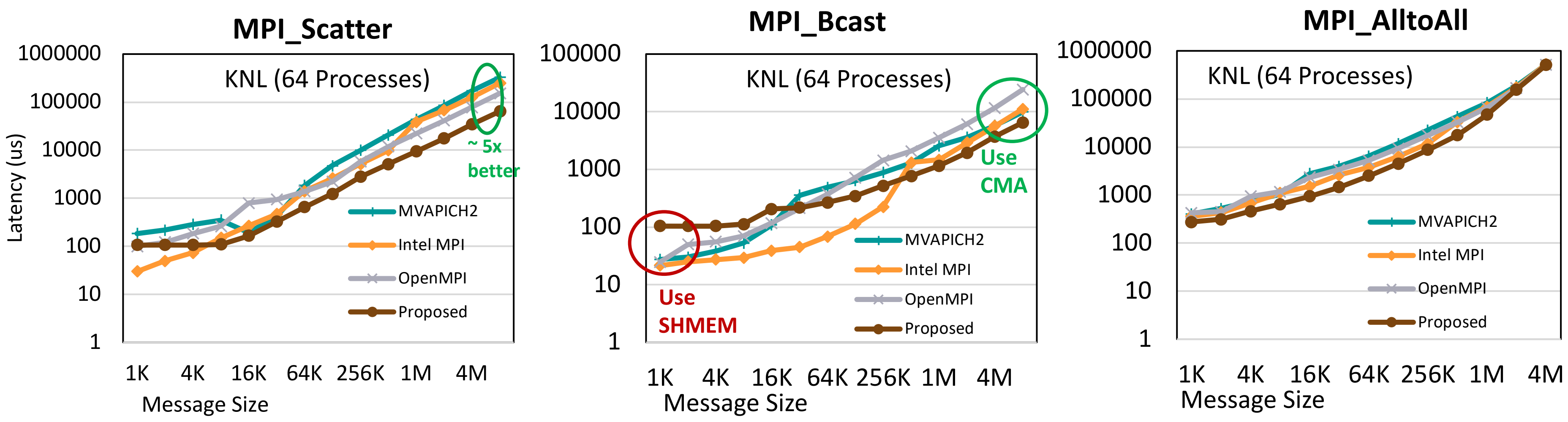




Contention is at Process level



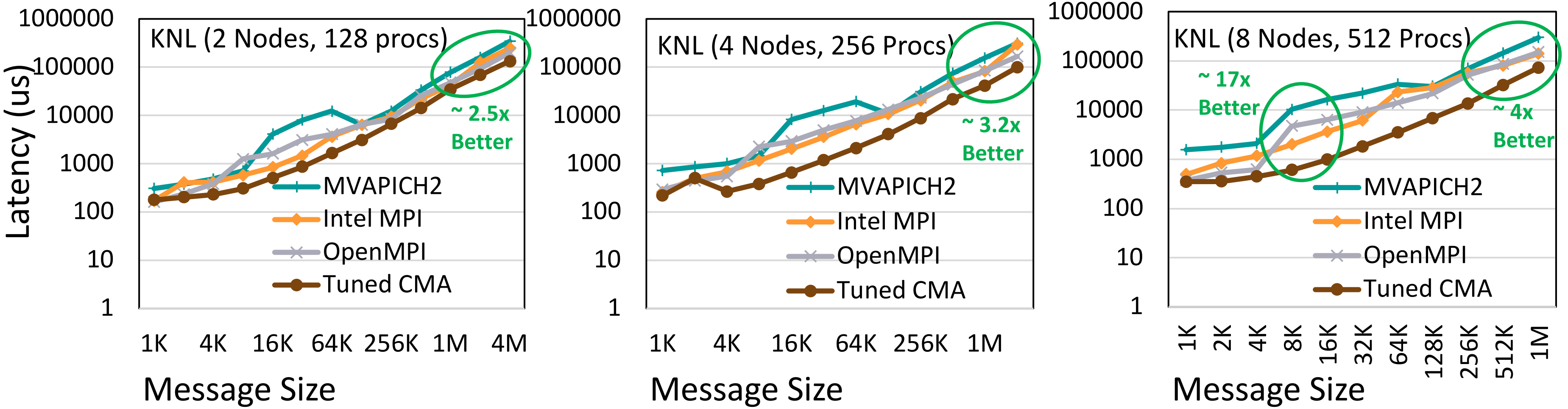
Contention-aware CMA Collective



- Up to 5x and 2x improvement for MPI_Scatter and MPI_Bcast on KNL
  - For Bcast, improvements obtained for large messages only (p-1 copies with CMA, p copies with Shared memory)
- AlltoAll Large message performance bound by system bandwidth (5%-20% improvement)
- Fallback to SHMEM for small messages

S. Chakraborty, H. Subramoni, and D. K. Panda, Contention Aware Kernel-Assisted MPI Collectives for Multi/Many-core Systems, *IEEE Cluster '17, BEST Paper Finalist*

Multi-Node Scalability Using Two-Level Algorithms



- Significantly faster intra-node communication
- New two-level collective designs can be composed
- 4x-17x improvement in 8 node Scatter and Gather compared to default MVAPICH2

Can we have zero-copy "Reduction" collectives with this approach?

Do you see the problem here???

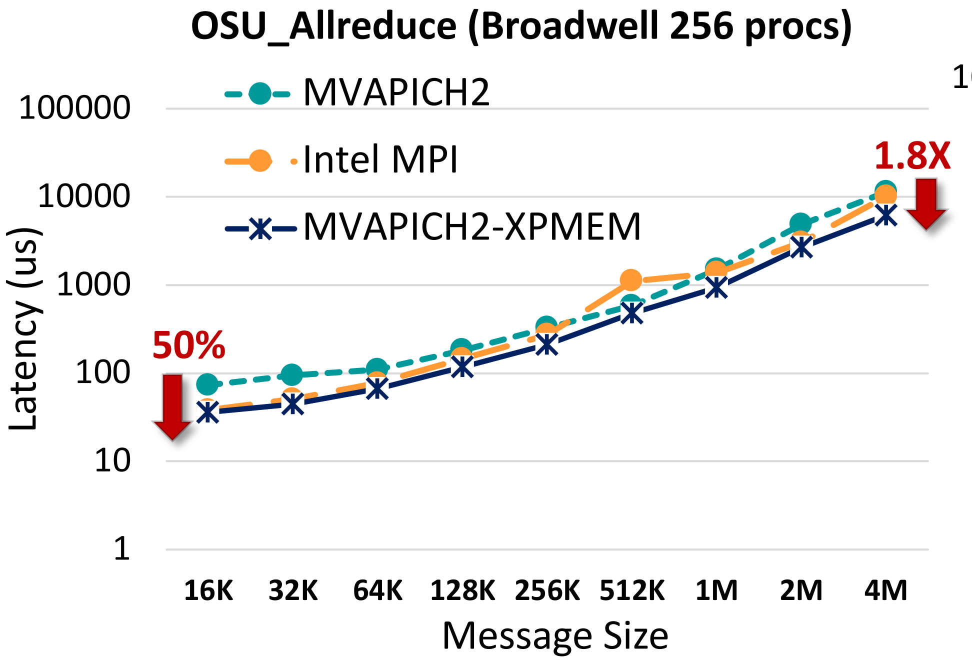
1. *Contention "avoidance" – Not removal*
2. *Reduction requires extra copies*

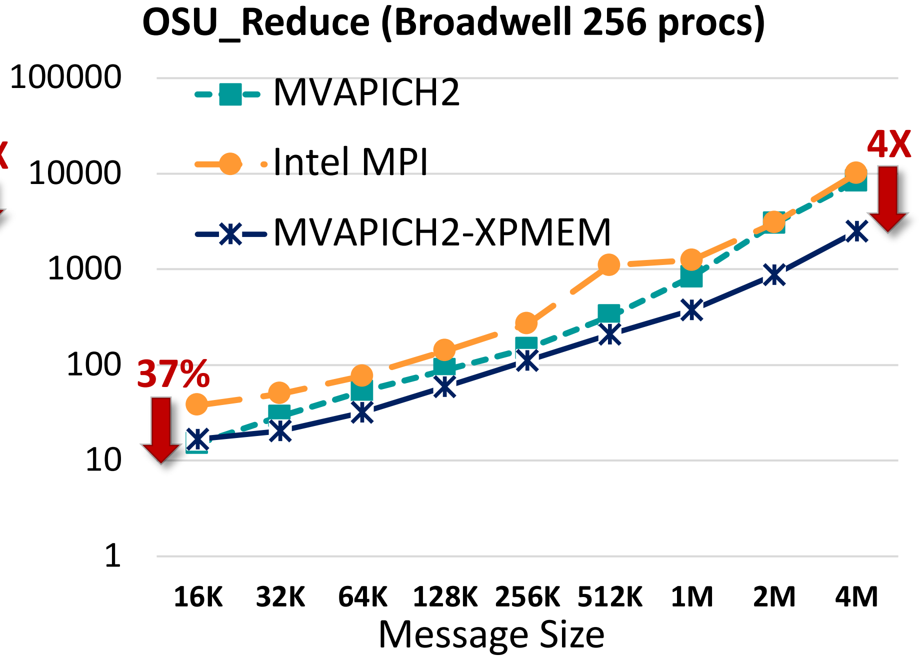
Shared Address Space (XPMEM-based) Collectives

- Offload Reduction computation and communication to peer MPI ranks
  - Every Peer has direct "load/store" access to other peer's buffers
  - Multiple pseudo roots independently carry-out reductions for intra-and inter-node
  - Directly put reduced data into root's receive buffer
- *True "Zero-copy"* design for Allreduce and Reduce
  - No copies require during the entire duration of Reduction operation
  - Scalable to multiple nodes
- *Zero contention* overheads as memory copies happen in *"user-space"*

J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. Panda, Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores, International Parallel & Distributed Processing Symposium (IPDPS '18), May 2018.

Shared Address Space (XPMEM)-based Collectives Design



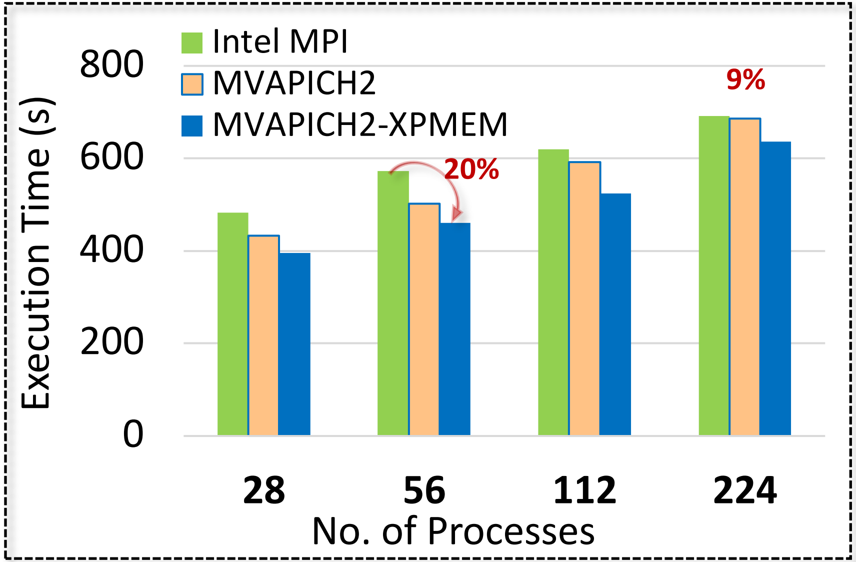


- "*Shared Address Space*"-based true *zero-copy* Reduction collective designs in MVAPICH2
- Offloaded computation/communication to peers ranks in reduction collective operation
- Up to **4X** improvement for 4MB Reduce and up to **1.8X** improvement for 4M AllReduce

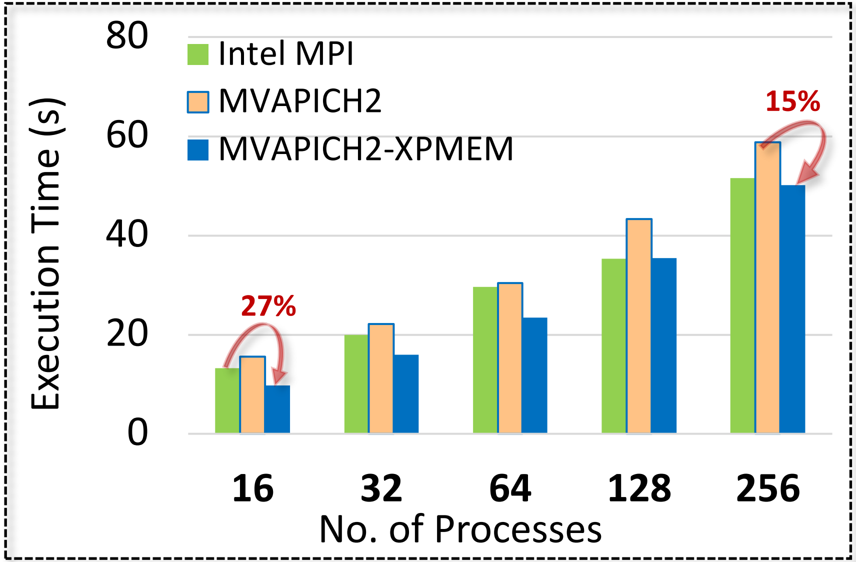
Will be available in upcoming MVAPICH2-X release

Application-Level Benefits of XPMEM-Based Collectives

CNTK AlexNet Training
(Broadwell, B.S=default, iteration=50, ppn=28)



MiniAMR (Broadwell, ppn=16)

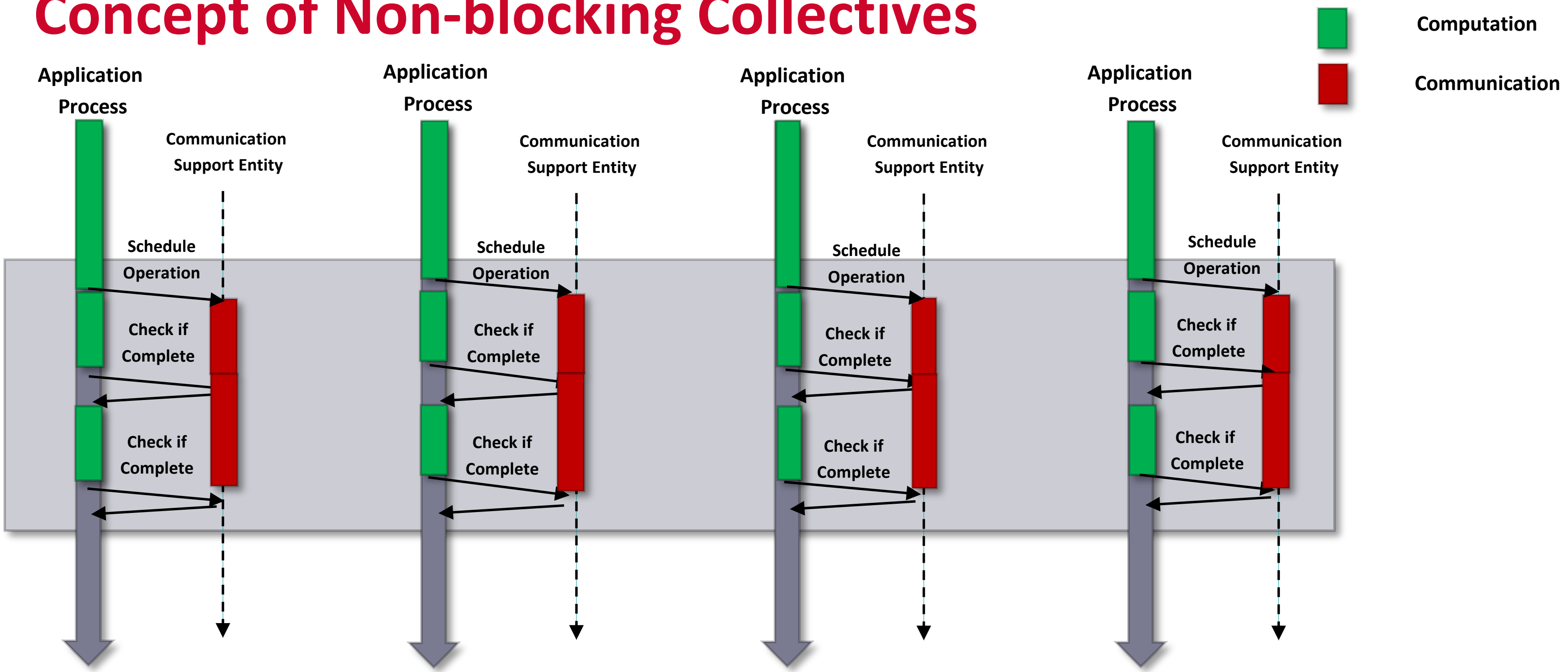


- Up to **20%** benefits over IMPI for CNTK DNN training using AllReduce
- Up to **27%** benefits over IMPI and up to **15%** improvement over MVAPICH2 for MiniAMR application kernel

Agenda

- Exploiting high concurrency and high bandwidth offered by modern architectures for MPI collectives design
  - Point-to-point
  - Direct Shared-memory
  - Data Partitioned Multi-Leader (DPML)
- Designing “zero-copy” and “contention-free” Collective Communication
  - Contention-aware designs
  - True zero-copy collectives
- **Hardware offloading for better communication and computation overlap**
  - **SHARP based offloaded collectives**
  - **CORE-Direct based Non-blocking collectives**

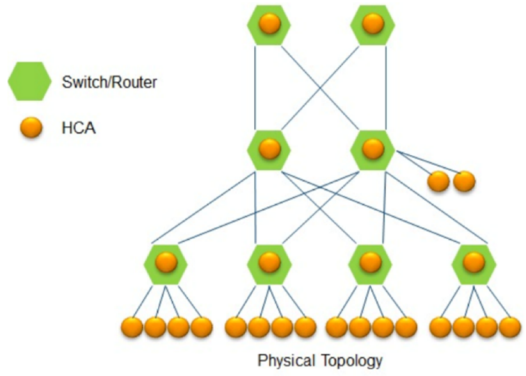
Concept of Non-blocking Collectives



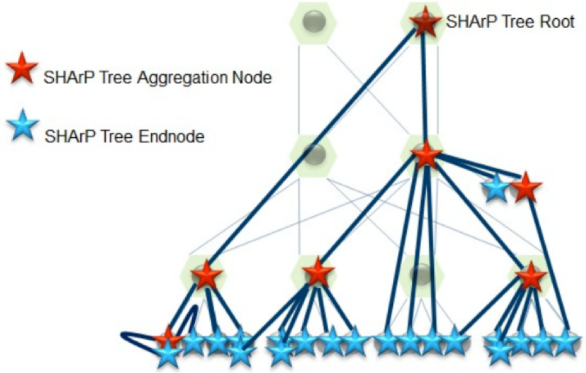
- Application processes schedule collective operation
- Check periodically if operation is complete
- **Overlap of computation and communication => Better Performance**
- *Catch: Who will progress communication*

Offloading with Scalable Hierarchical Aggregation Protocol (SHArP)

- Management and execution of MPI operations in the network by using SHArP
  - Manipulation of data while it is being transferred in the switch network
- SHArP provides an abstraction to realize the reduction operation
  - Defines Aggregation Nodes (AN), Aggregation Tree, and Aggregation Groups
  - AN logic is implemented as an InfiniBand Target Channel Adapter (TCA) integrated into the switch ASIC *
  - Uses RC for communication between ANs and between AN and hosts in the Aggregation Tree *



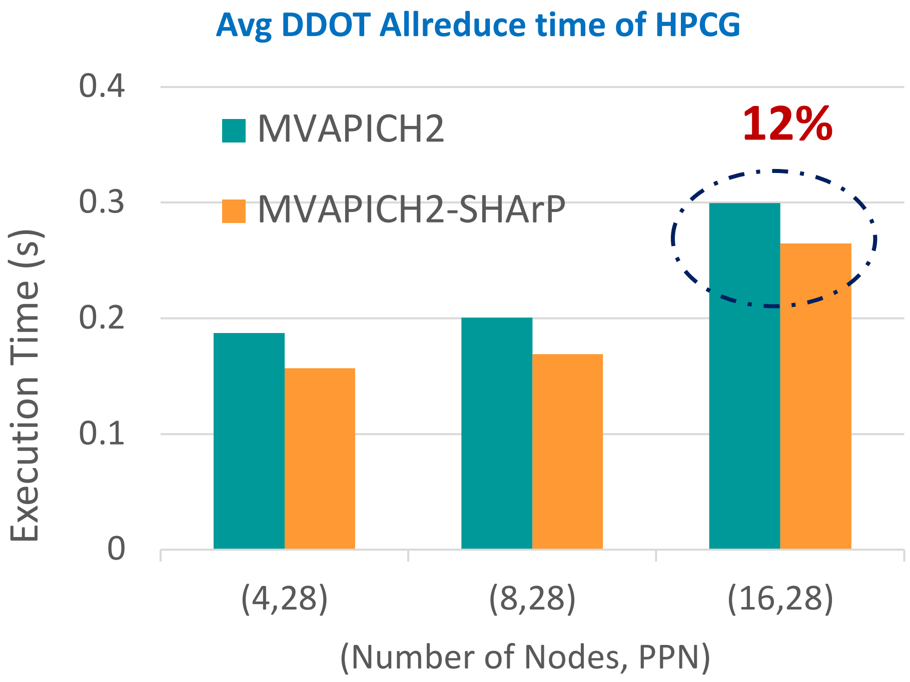
Physical Network Topology*

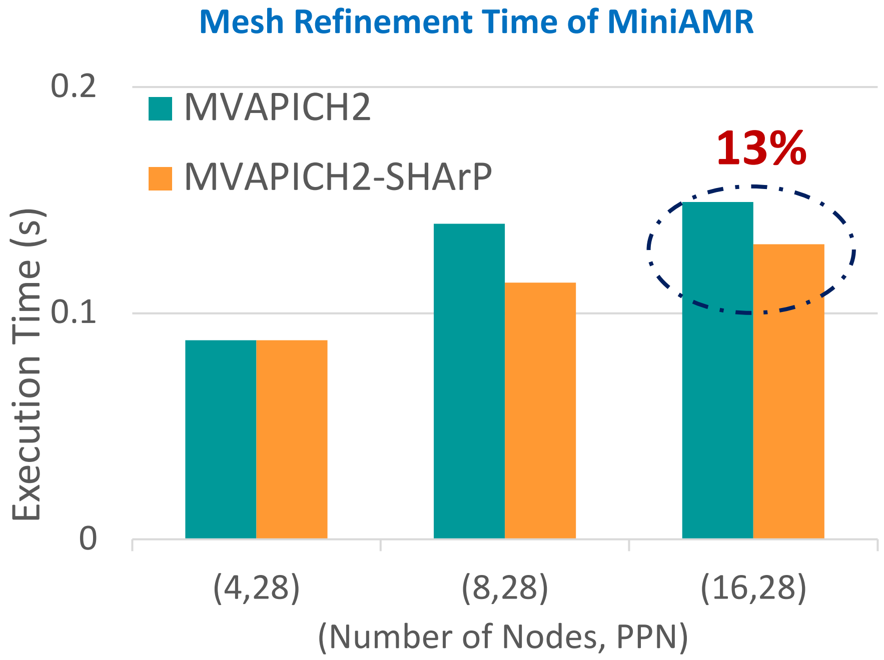


Logical SHArP Tree*

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

SHArP based blocking Allreduce Collective Designs in MVAPICH2

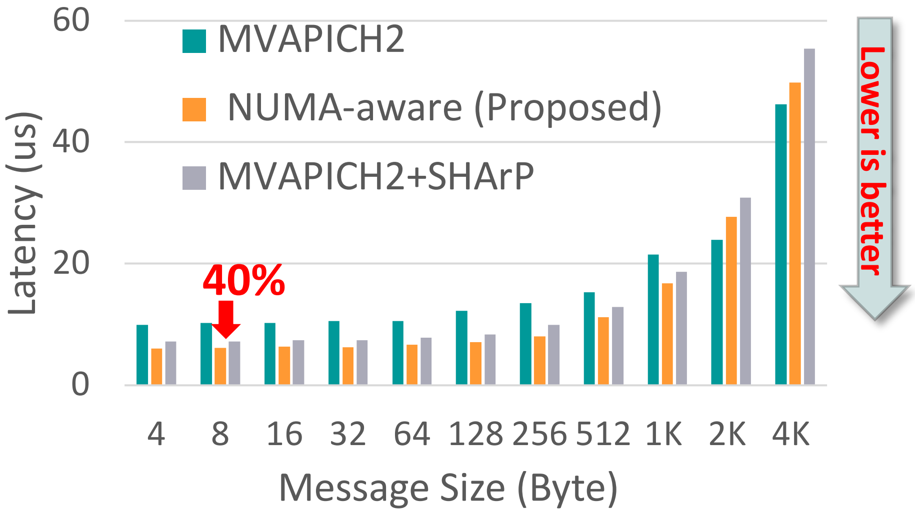


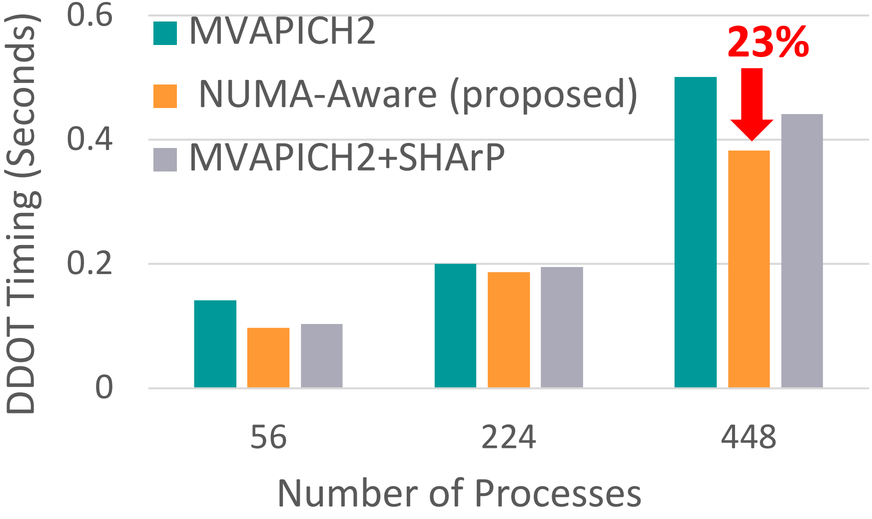


M. Bayatpour, S. Chakraborty, H. Subramoni, X. Lu, and D. K. Panda, Scalable Reduction Collectives with Data Partitioning-based Multi-Leader Design, SuperComputing '17.

SHArP Support is available since MVAPICH2 2.3a

Performance of NUMA-aware SHArP Design on XEON + IB Cluster





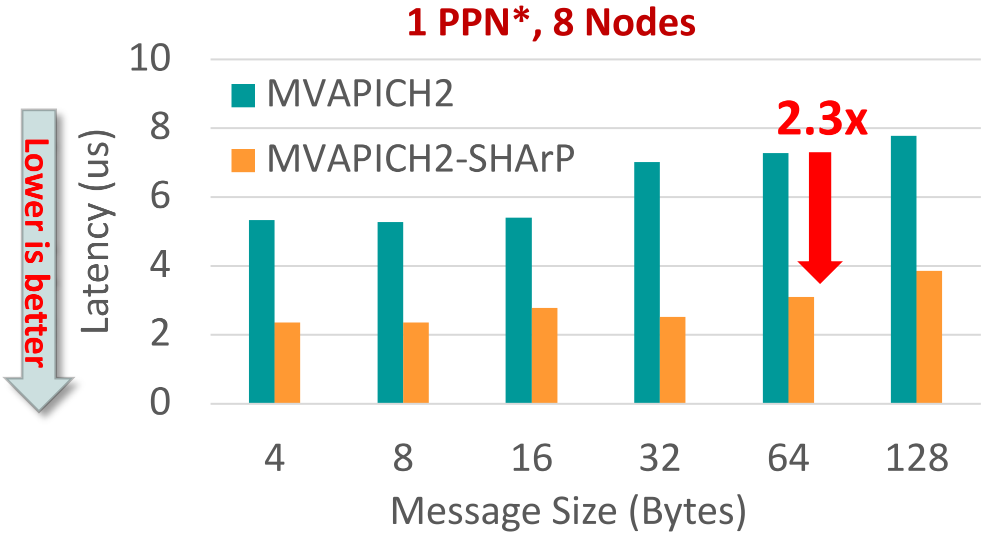
OSU Micro Benchmark (16 Nodes, 28 PPN)

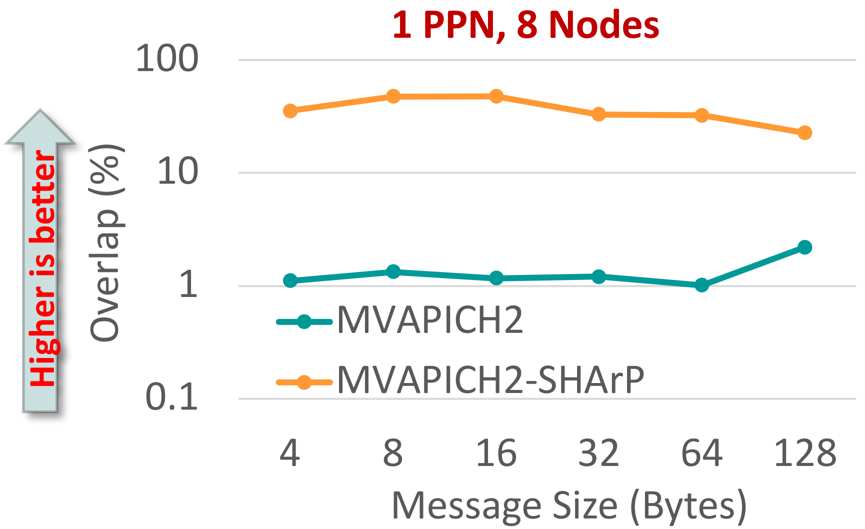
HPCG (16 nodes, 28 PPN)

- As the message size decreases, the benefits of using Socket-based design increases
- NUMA-aware design can reduce the latency by up to 23% for DDOT phase of HPCG and up to 40% for micro-benchmarks

SHArP based Non-Blocking Allreduce in MVAPICH2

MPI_Iallreduce Benchmark





- Complete offload of Allreduce collective operation to "*Switch*"
  - higher overlap of communication and computation

Available since MVAPICH2 2.3a

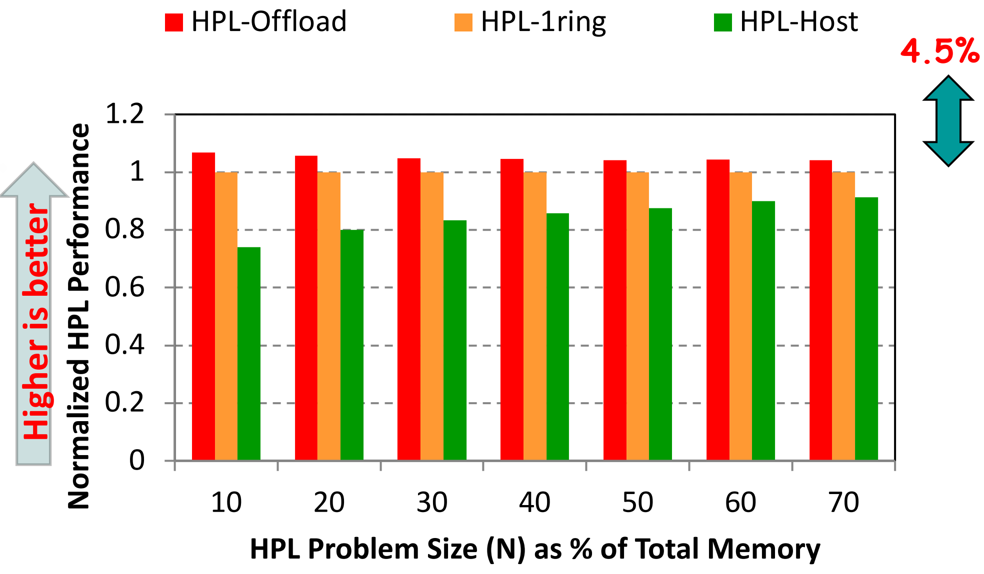
**PPN: Processes Per Node*

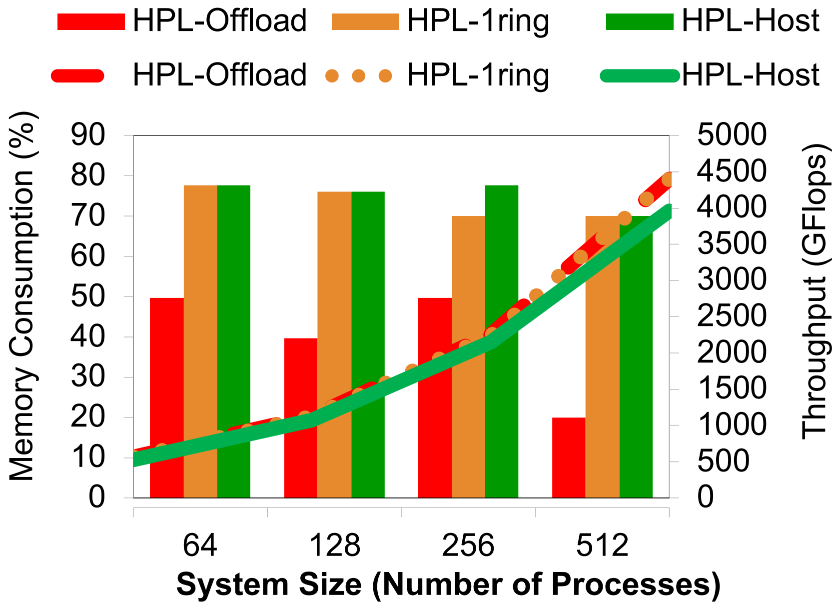
NIC offload based Non-blocking Collectives using CORE-Direct

- Mellanox CORE-Direct technology allows for offloading the collective communication to the network adapter
- MVAPICH2 supports CORE-Direct based offloading of non-blocking collectives
  - Covers all the non-blocking collectives
  - Enabled by configure and runtime parameters
- CORE-Direct based MPI_Ibcast design improves the performance of High Performance Linpack (HPL) benchmark

Available since MVAPICH2-X 2.2a

Co-designing HPL with Core-Direct and Performance Benefits





HPL Performance Comparison with 512 Processes

HPL-Offload consistently offers higher throughput than HPL-1ring and HPL-Host. Improves peak throughput by up to 4.5 % for large problem sizes

HPL-Offload surpasses the peak throughput of HPL-1ring with significantly smaller problem sizes and run-times!

K. Kandalla, H. Subramoni, J. Vienne, S. Pai Raikar, K. Tomko, S. Sur, and D K Panda,
Designing Non-blocking Broadcast with Collective Offload on InfiniBand Clusters: A Case Study with HPL, (HOTI 2011)

Concluding Remarks

- Many-core nodes will be the foundation blocks for emerging Exascale systems
- Communication mechanisms and runtimes need to be **re-designed** to take advantage of the **high concurrency** offered by manycores
- Presented a set of **novel designs** for **collective communication** primitives in MPI that **address several challenges**
- Demonstrated the **performance benefits** of our proposed designs under a variety of **multi-/many-cores and high-speed networks**
- Some of these designs are already available in MVAPICH2 libraries
- The new designs will be available in upcoming MVAPICH2 libraries

Thank You!

hashmi.29@osu.edu

Network Based Computing Laboratory

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

HiBD
High-Performance Big Data

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

HiDL
High-Performance Deep Learning

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/