OPENFABRICS ALLIANCE

2021 OFA Virtual Workshop

DESIGNING A HIGH-PERFORMANCE MPI LIBRARY USING IN-NETWORK COMPUTING

Mohammadreza Bayatpour, Bharath Ramesh, Kaushik Kandadi Suresh, Hari Subramoni, Dhabaleswar K. Panda

The Ohio State University

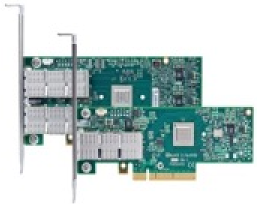
subramon@cse.ohio-state.edu

OUTLINE

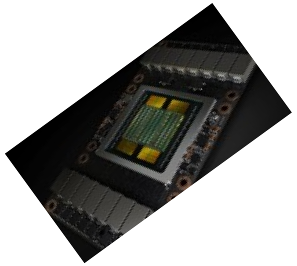
- Introduction and Motivation
- Overview of the MVAPICH2 project
- Hardware tag matching in MPI
- Offloading with Scalable Hierarchical Aggregation Protocol (SHARP)
- Conclusions and Future Work

HIGH-PERFORMANCE ARCHITECTURES: DRIVERS OF MODERN HPC CLUSTERS

Multi-/Many-core Processors



High Performance Interconnects - InfiniBand
<1usec latency, 200Gbps Bandwidth>

Accelerators
high compute density, high performance/watt
>1 TFlop DP on a chip

SSD, NVMe-SSD, NVRAM

- **Multi-core/many-core technologies**
- **Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)**
- **Solid State Drives (SSDs), Non-Volatile Random-Access Memory (NVRAM), NVMe-SSD**
- **Accelerators (NVIDIA GPGPUs)**

Summit

Sunway TaihuLight

Sierra

K - Computer

INTRODUCTION AND MOTIVATION

- **Applications exchange large amounts of data during their run-time**
- **Communication runtimes have the following goals in order to provide high performance**
  - Overlap computation and communication
  - Maximally utilize CPU resources
  - Scale-out and Scale-up efficiency
  - Ideally no changes to application code for performance
  - Utilize high levels of parallelism
- **Some tasks can be offloaded to other computing elements, freeing up CPU resources for other important tasks like application-level compute**
  - “In-network” computing is an emerging technology that enables this. Examples : SHARP, Hardware Tag Matching
- **Support for in-network computing is critical for efficient scale-out of HPC and AI applications in the “Exascale” era.**

OUTLINE

- Introduction and Motivation
- Overview of the MVAPICH2 project
- Hardware tag matching in MPI
- Offloading with Scalable Hierarchical Aggregation Protocol (SHARP)
- Conclusions and Future Work

OVERVIEW OF THE MVAPICH2 PROJECT

- **High Performance open-source MPI Library**
- **Support for multiple interconnects**
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), and AWS EFA
- **Support for multiple platforms**
  - x86, OpenPOWER, ARM, Xeon-Phi, GPGPUs (NVIDIA and AMD)
- **Started in 2001, first open-source version demonstrated at SC ‘02**
- **Supports the latest MPI-3.1 standard**
- **http://mvapich.cse.ohio-state.edu**
- **Additional optimized versions for different systems/environments:**
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA GPGPUs, since 2014
  - **MVAPICH2-GDR with support for AMD GPUs since MVAPICH2-GDR-2.3.5**
  - MVAPICH2-MIC with support for Intel Xeon-Phi, since 2014
  - MVAPICH2-Virt with virtualization support, since 2015
  - MVAPICH2-EA with support for Energy-Awareness, since 2015
  - MVAPICH2-Azure for Azure HPC IB instances, since 2019
  - MVAPICH2-X-AWS for AWS HPC+EFA instances, since 2019
- **Tools:**
  - **OSU MPI Micro-Benchmarks (OMB),** since 2003
    - **Support for AMD GPUs with ROCm-aware MPI in Release Version 5.7**
  - OSU InfiniBand Network Analysis and Monitoring (INAM), since 2015

- Used by more than 3,150 organizations in 89 countries
- More than 1.26 Million downloads from the OSU site directly
- Empowering many TOP500 clusters (Nov ‘20 ranking)
  - 4^{th} , 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 9^{th}, 448, 448 cores (Frontera) at TACC
  - 14^{th}, 391,680 cores (ABCI) in Japan
  - 21^{st}, 570,020 cores (Nurion) in South Korea and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 9^{th} ranked TACC Frontera system
- Empowering Top500 systems for more than 16 years

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, Elastic Fabric Adapter)

Transport Protocols: RC, XRC, UD, DC

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

Modern HCA Features: Burst, Poll, Tag Matching

Modern IB Features: Multicast, SHARP, BlueField*

* Upcoming

OUTLINE

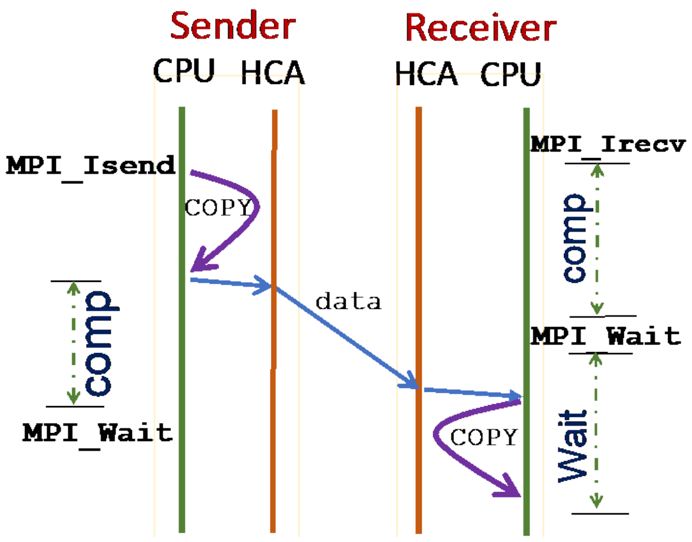
- Introduction and Motivation
- Overview of the MVAPICH2 project
- Hardware tag matching in MPI
- Offloading with Scalable Hierarchical Aggregation Protocol (SHARP)
- Conclusions and Future Work

HARDWARE TAG MATCHING IN MPI

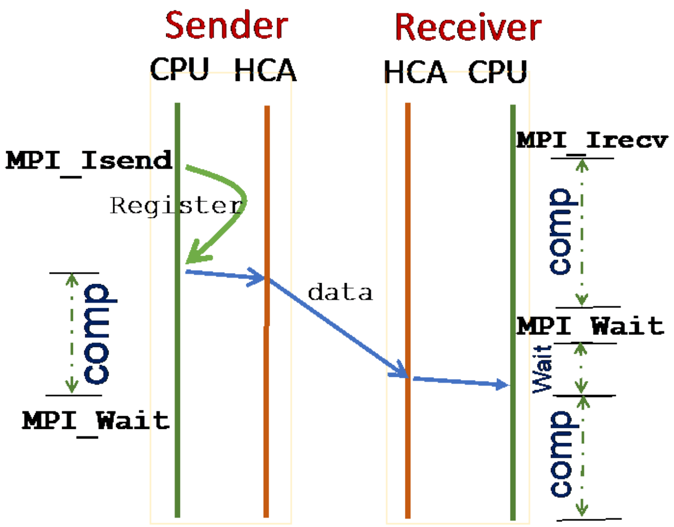
- **Offloading the processing of MPI Tag Matching from the host processor to HCA**
  - Software and Hardware need to synchronized to avoid message ordering issue
- **Enabling zero copy of MPI message transfers in Eager protocol**
  - Messages are written directly to the user's buffer without extra buffering and copies
  - Applicable to expected messages, unexpected messages are handled by host
- **Provides Rendezvous progress offload to HCA**
  - Upon finding a match, HCA initiates the RDMA Read without host involvement
  - Increases the overlap of communication and computation

- Mohammadreza Bayatpour, Jahanzeb Hashmi Maqbool, Sourav Chakraborty, Kaushik Kandadi Suresh, Seyedeh Mahdieh Ghazimirsaeed, Bharath Ramesh, Hari Subramoni and Dhabaleswar K. Panda.: Communication-Aware Hardware-Assisted MPI Overlap Engine. In International Conference on High Performance Computing - ISC High Performance 2020.
- Bayatpour, M., Ghazimirsaeed, S.M., Xu, S., Subramoni, H., Panda, D.K.: Design and Characterization of Infiniband Hardware Tag Matching in MPI. In proceedings of 20th Annual IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing (2020)

EAGER PROTOCOL USING HARDWARE TAG MATCHING



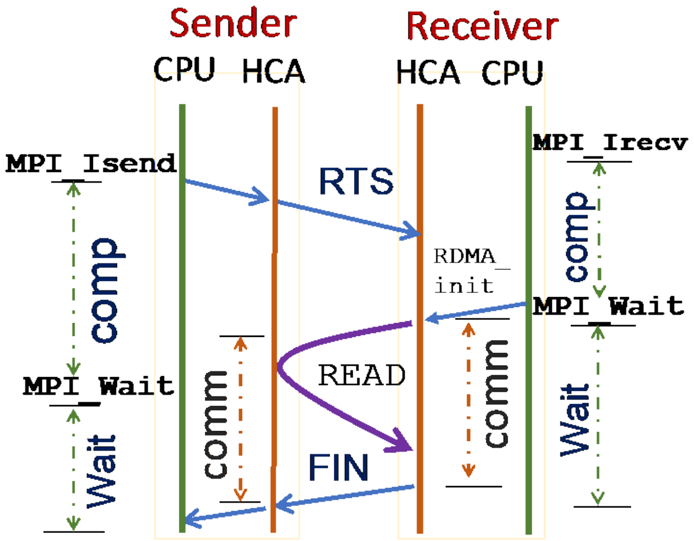
Without Hardware Tag Matching



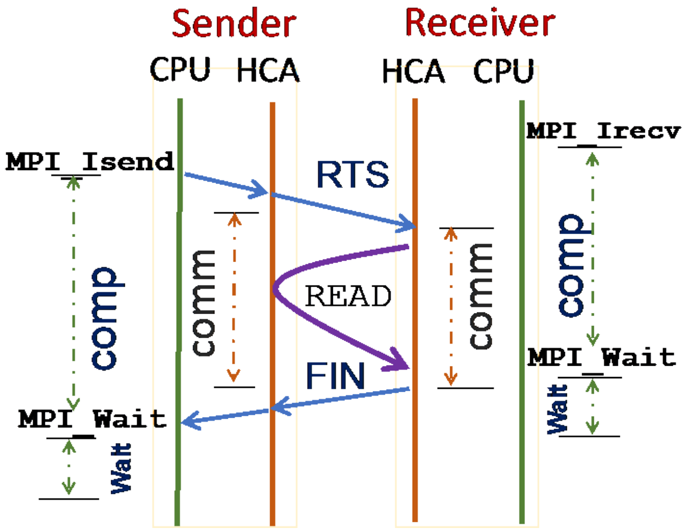
With Hardware Tag Matching

- Using Hardware Tag Matching, there is no extra overhead of copy operation in eager protocol

RENDEZVOUS PROTOCOL USING HARDWARE TAG MATCHING



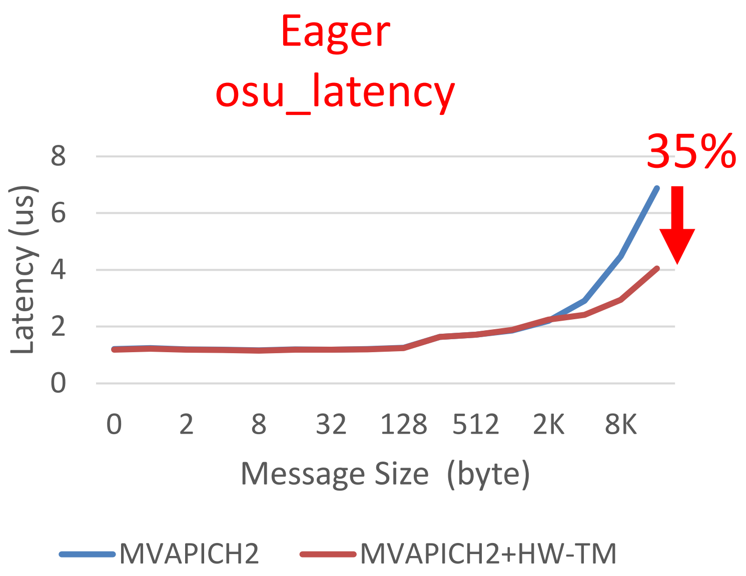
Without Hardware Tag Matching

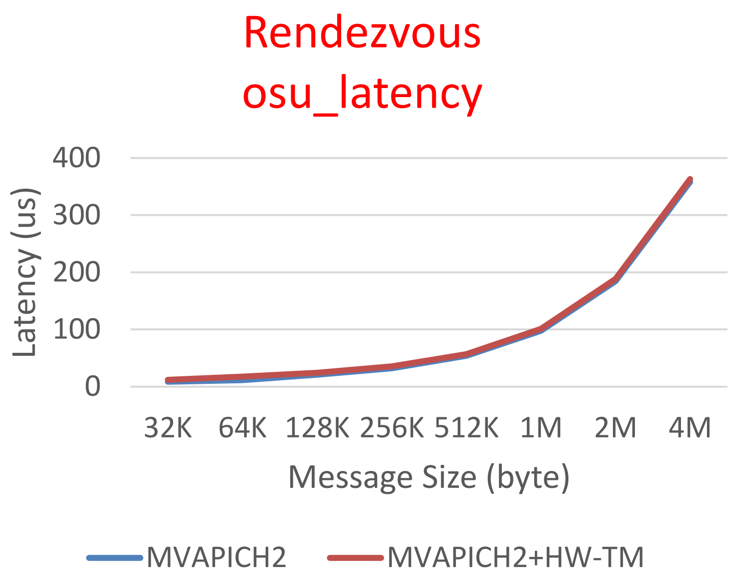


With Hardware Tag Matching

- Using Hardware Tag Matching, the overlap of communication and computation is increased in Rendezvous protocol

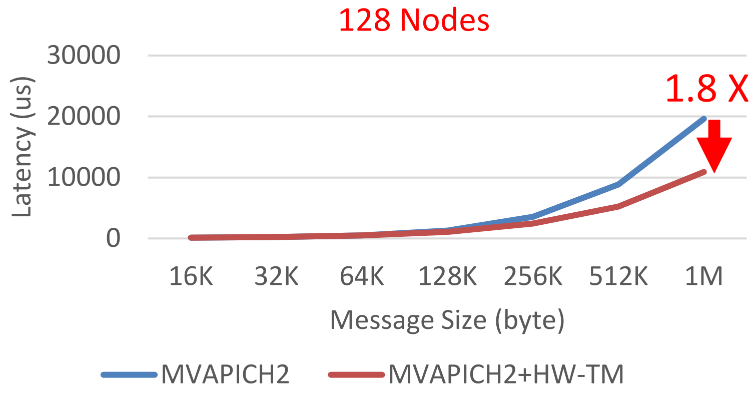
IMPACT OF ZERO COPY MPI MESSAGE PASSING USING HW TAG MATCHING (POINT-TO-POINT) ON TACC FRONTERA

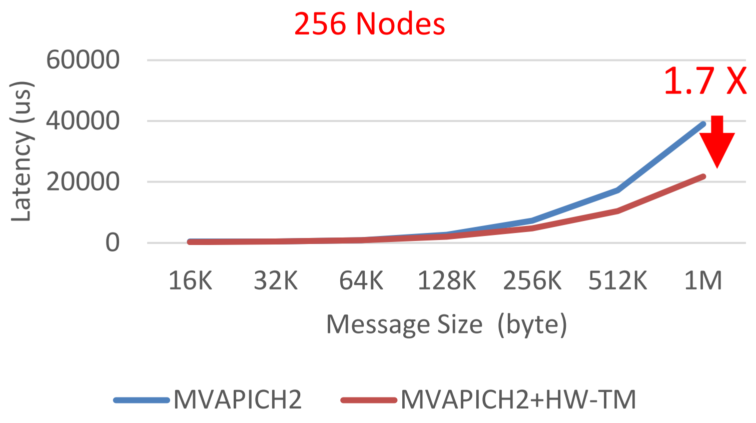


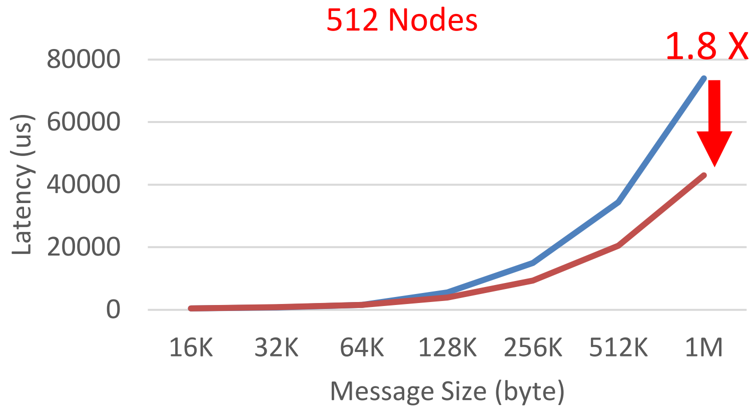


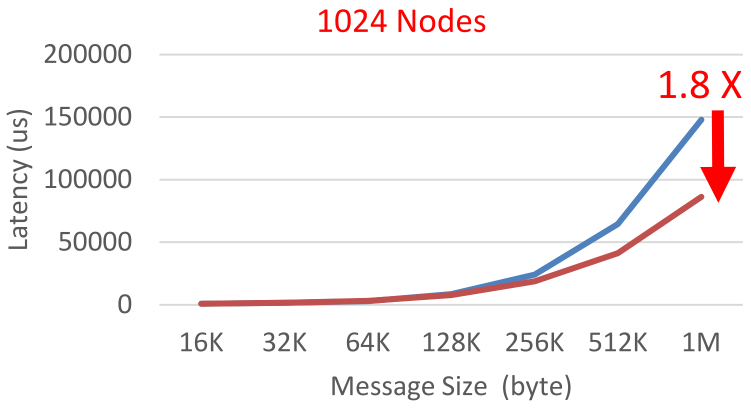
Removal of intermediate buffering/copies can lead up to 35% performance improvement in latency of medium messages on TACC Frontera

PERFORMANCE OF MPI_ISCATTERV USING HW TAG MATCHING



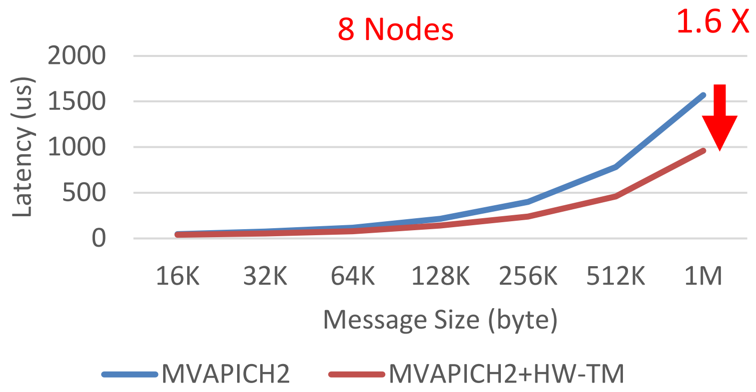


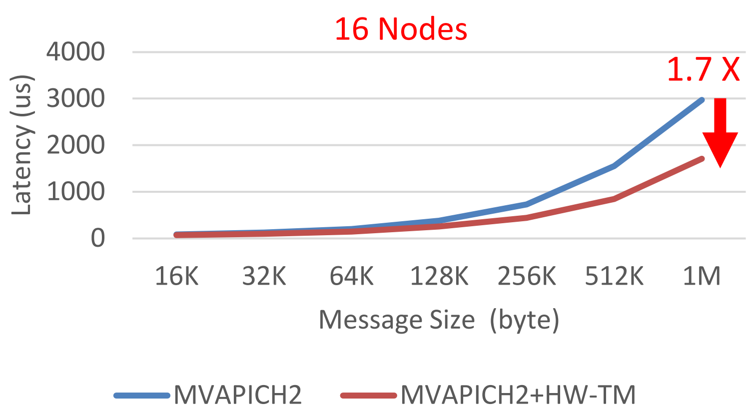


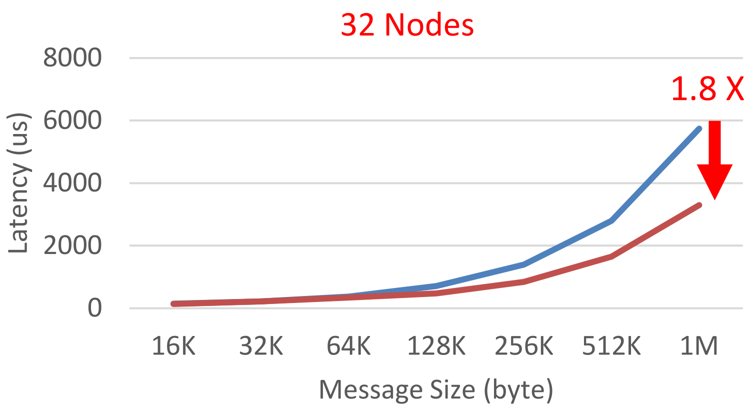


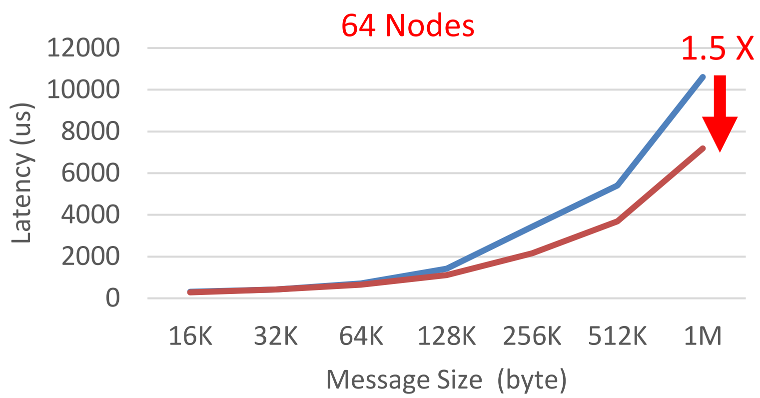
- Up to 1.8x Performance Improvement, Sustained benefits as system size increases

PERFORMANCE OF MPI_IALLTOALL USING HW TAG MATCHING



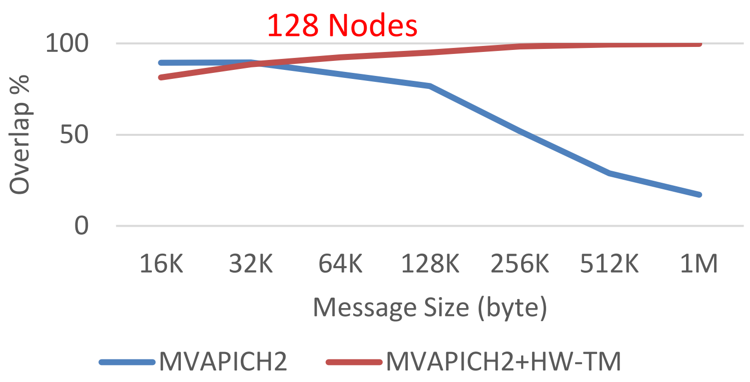


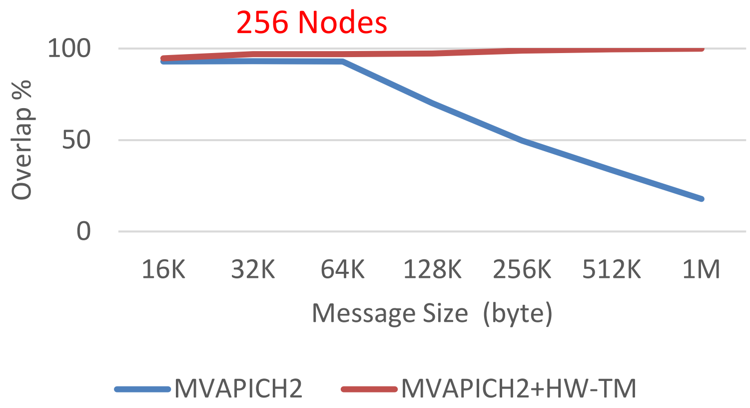


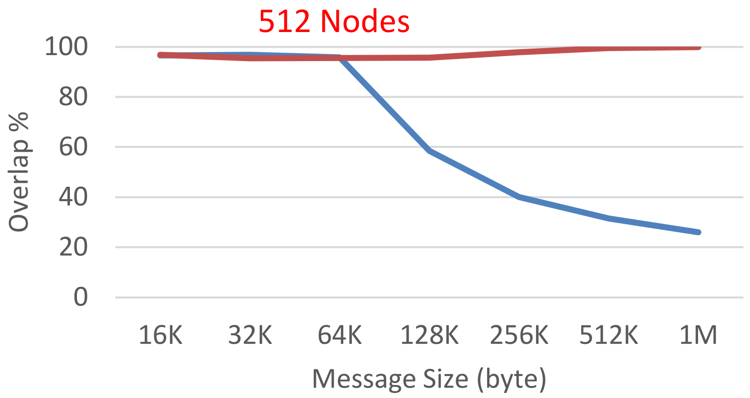


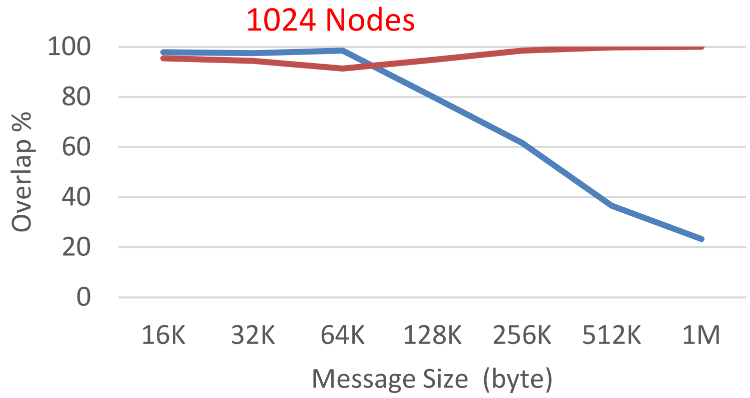
- Up to 1.8x Performance Improvement, Sustained benefits as system size increases

OVERLAP WITH MPI_ISCATTERV USING HW TAG MATCHING



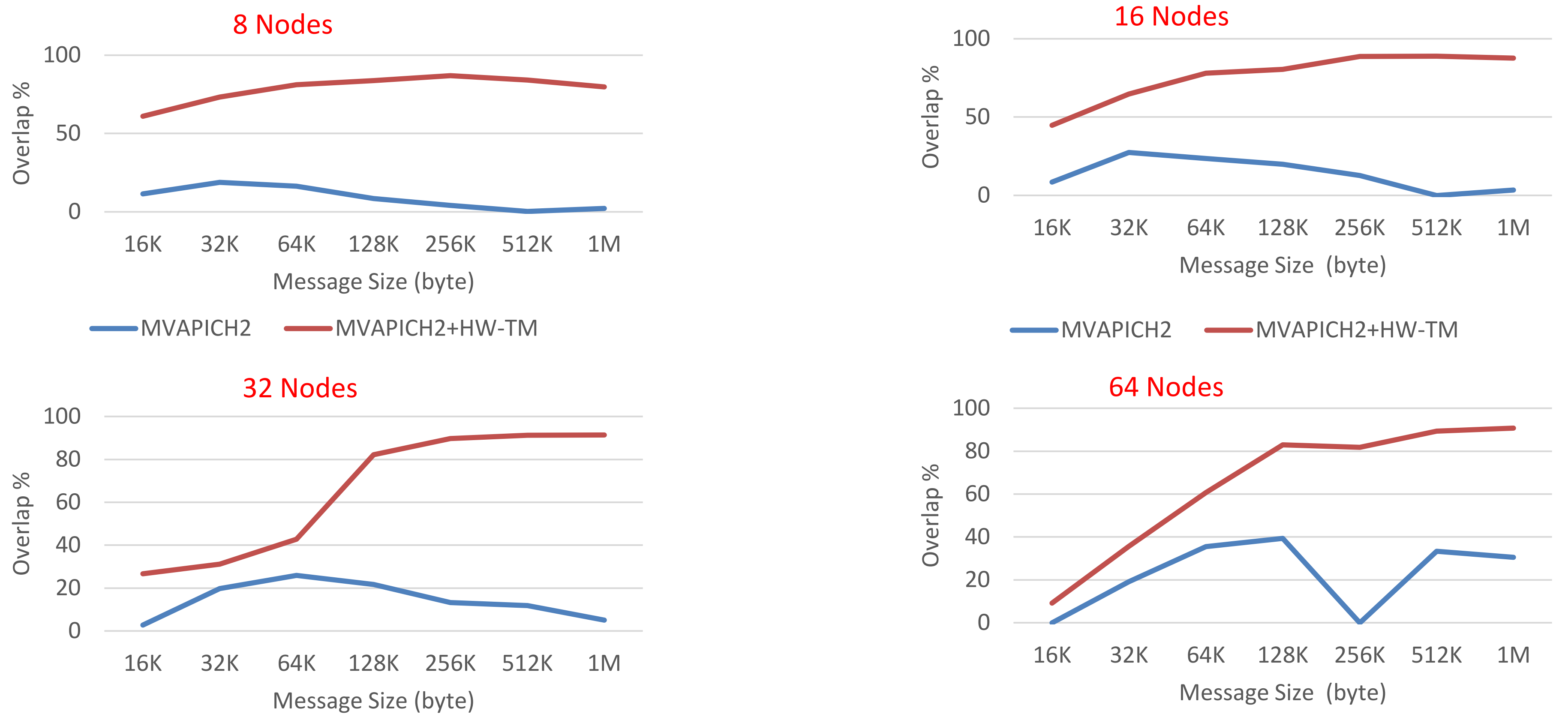






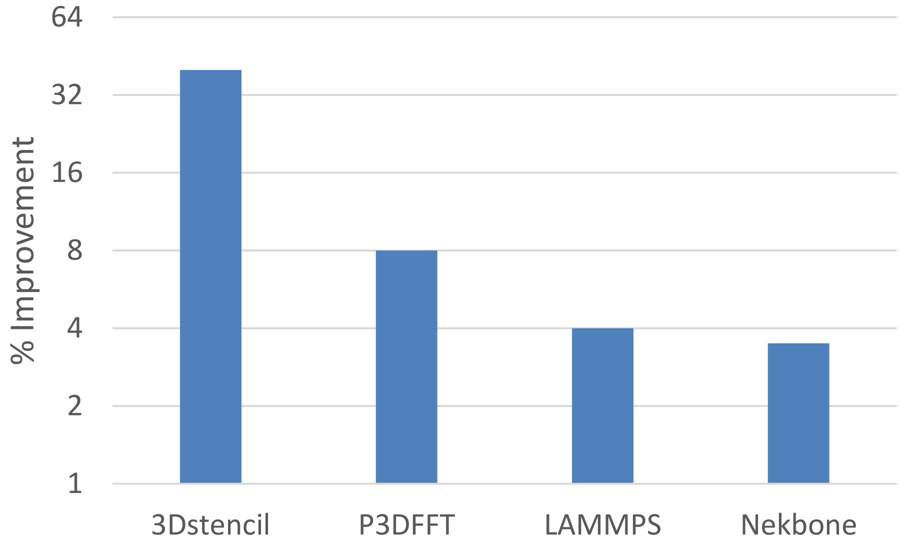
- Maximizing the overlap of communication and computation, Sustained benefits as system size increases

OVERLAP WITH MPI_IALLTOALL USING HW TAG MATCHING



- Maximizing the overlap of communication and computation, Sustained benefits as system size increases
- Tag matching support will be available in future releases of MVAPICH2

APPLICATION-LEVEL BENEFITS FOR TAG MATCHING



Tests Setup			
App.	N	PPN	Measured Unit
3DStencil	128	56	Overall Runtime
P3DFFT	16	32	Avg. Total Time Per Forward Backward FFT Loop
LAMMPS	16	32	Execution Time
Nekbone	16	32	Avg. MFlops

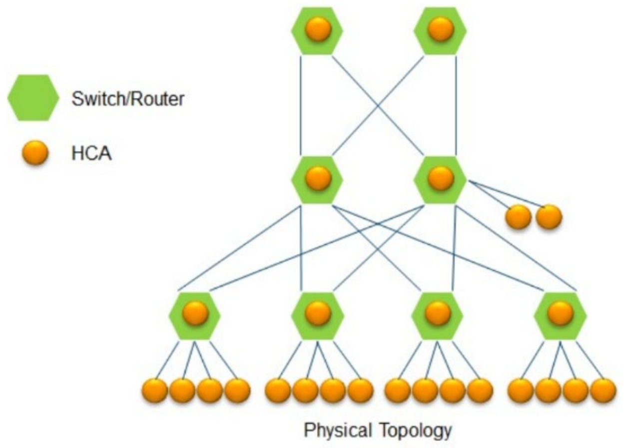
- Maximizing the overlap of communication and computation leads to application-level improvements in total execution time compared to default MVAPICH2 on Frontera

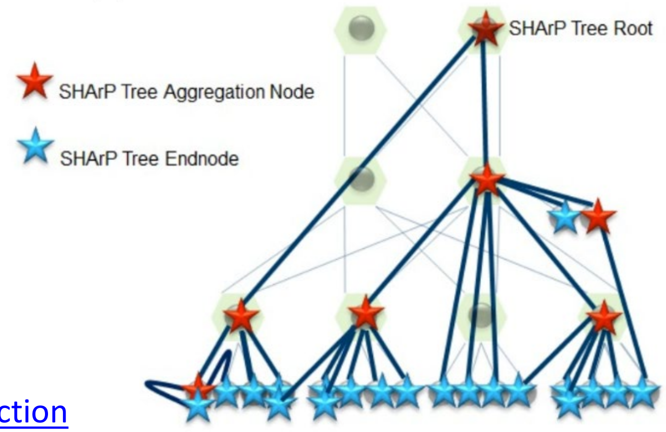
OUTLINE

- Introduction and Motivation
- Overview of the MVAPICH2 project
- Hardware tag matching in MPI
- Offloading with Scalable Hierarchical Aggregation Protocol (SHARP)
- Conclusions and Future Work

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 *





More details in the tutorial "SHARPv2: In-Network Scalable Streaming Hierarchical Aggregation and Reduction Protocol" by Devendar Bureddy (NVIDIA/Mellanox)

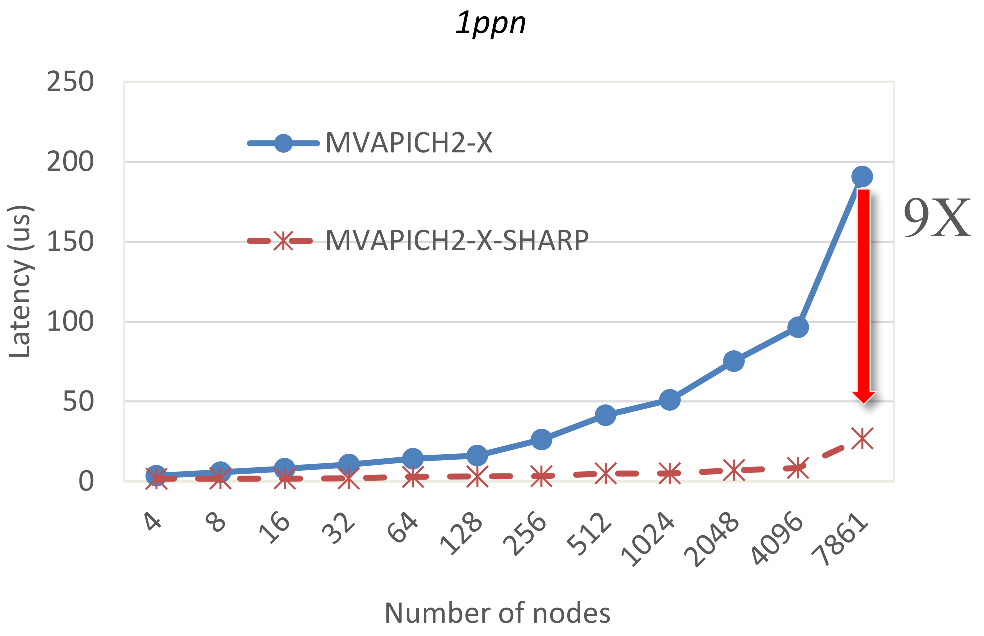
Logical SHArP Tree*

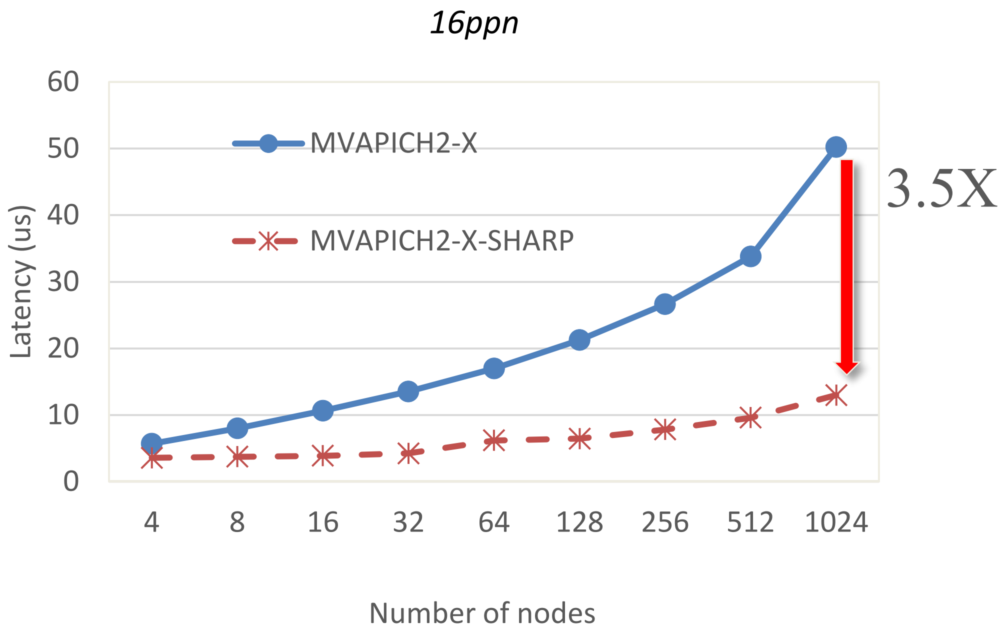
* Bloch et al. Scalable Hierarchical Aggregation Protocol (SHArP): A Hardware Architecture for Efficient Data Reduction

HIGH-LEVEL SHARP DESIGNS IN MVAPICH2

- **Support for SHARP based offload implemented for MPI_Allreduce, MPI_Barrier and MPI_Reduce**
- **Only one process per node participates in the SHARP operation due to circumvent hardware limitations**
- **Intra-node operations happen through Shared Memory**
  - “Two-copy” mechanism”, cache-aligned and very efficient for smaller message sizes
- **Algorithms are usually “two-level” in nature, with intra-node and inter-node steps**
- **MPI_Allreduce design**
  - First, socket-level leaders perform an intra-socket reduction amongst processes in their socket
  - Second, socket-level leaders reduce values to a designated node-level leader using shared memory
  - Third, use SHARP APIs to perform an allreduce over node-level leaders. This is the only inter-node step.
  - For the final two steps, a broadcast of these results is done within the node (as mirror of the first and second steps)
  - Algorithm for MPI_Barrier is the same – reductions are replaced with basic gather of “flag” arrays
- **Support for MPI_Reduce is implemented using the allreduce SHARP primitive by ignoring recvbuf at non-root processes**

PERFORMANCE OF MPI_BARRIER WITH SHARP



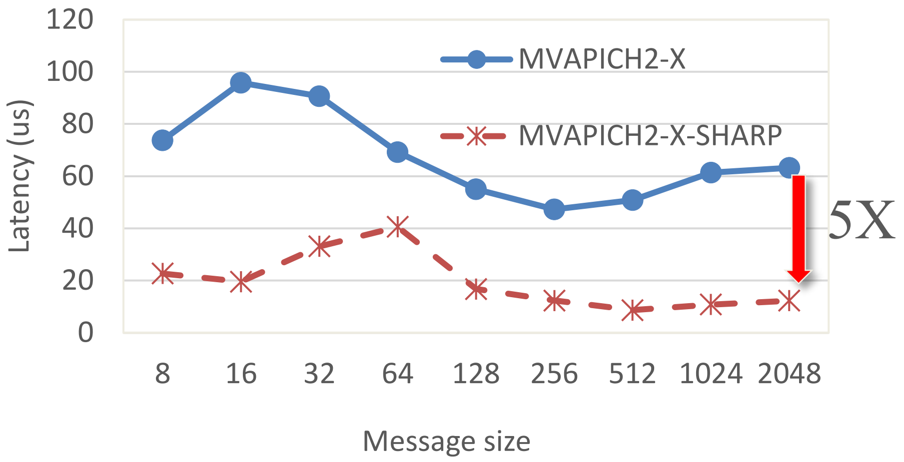


- Near flat scaling with SHARP
- Up to 9X for 1ppn at full system scale and 3.5X for 16ppn over default MVAPICH2-X

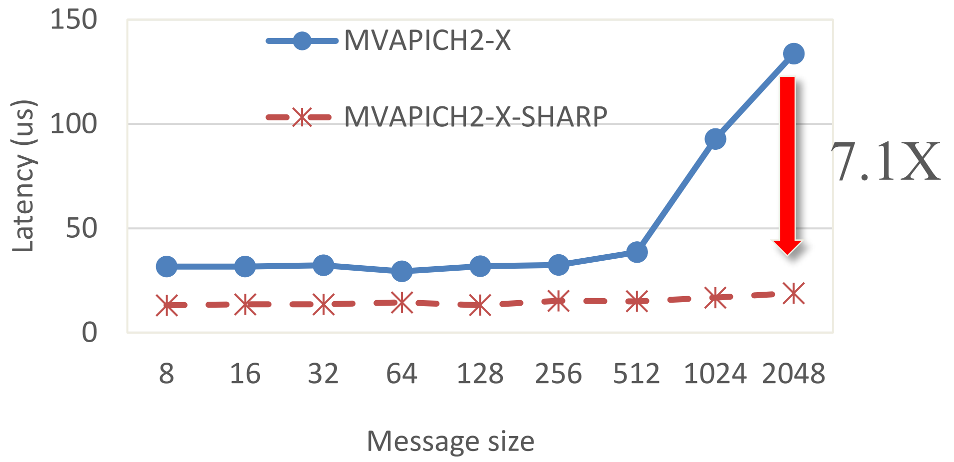
B. Ramesh , K. Suresh , N. Sarkauskas , M. Bayatpour , J. Hashmi , H. Subramoni , and D. K. Panda, Scalable MPI Collectives using SHARP: Large Scale Performance Evaluation on the TACC Frontera System, ExaMPI2020 - Workshop on Exascale MPI 2020, Nov 2020.

PERFORMANCE OF MPI_ALLREDUCE WITH SHARP

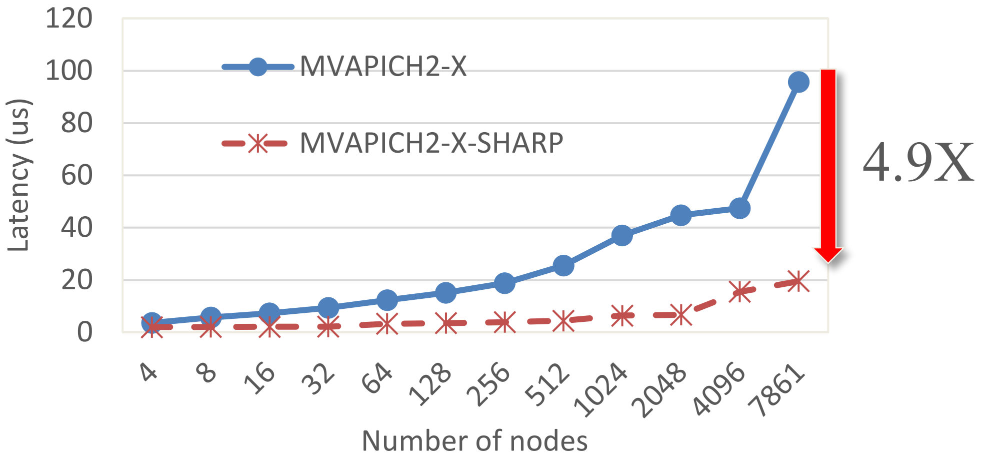
MPI_Allreduuce
(PPN = 1, Nodes = 7861)



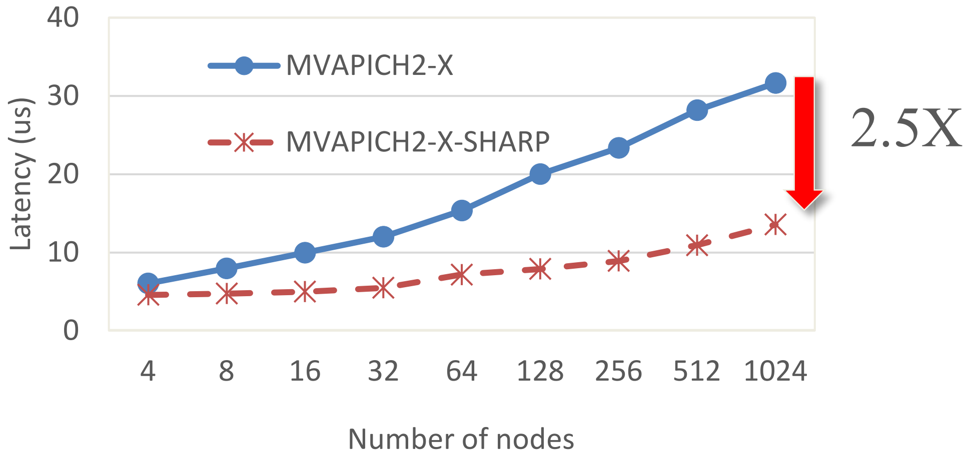
MPI_Allreduuce
(PPN = 16, Nodes = 7861)



MPI_Allreduce
PPN = 1, Size = 16 bytes

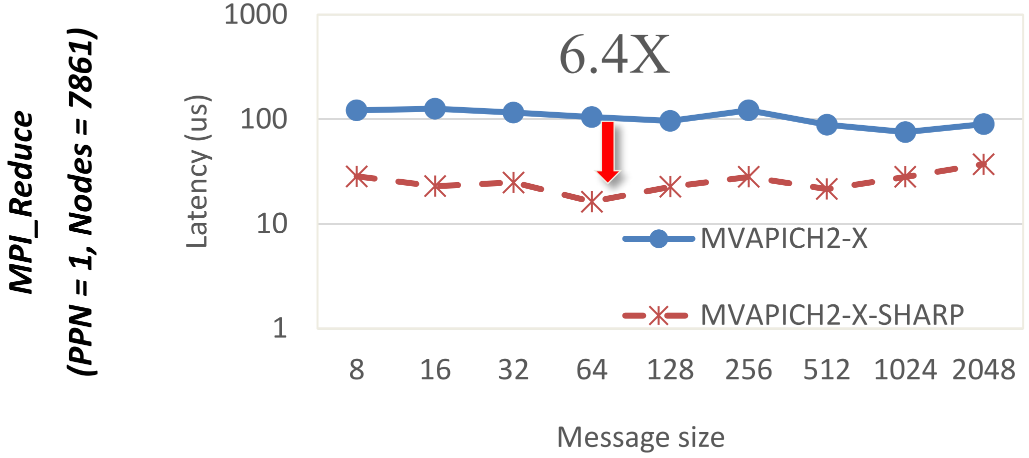


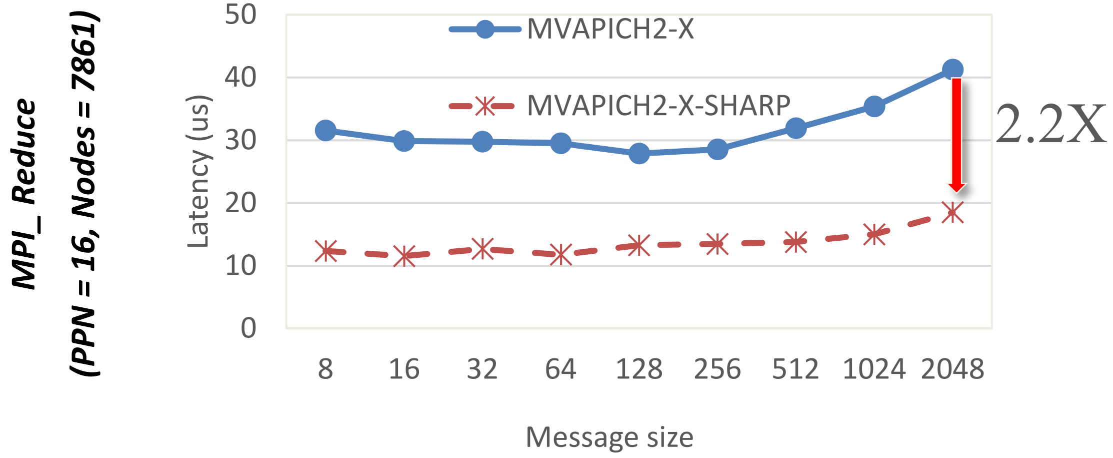
MPI_Allreduce
PPN = 16, Size = 16 bytes

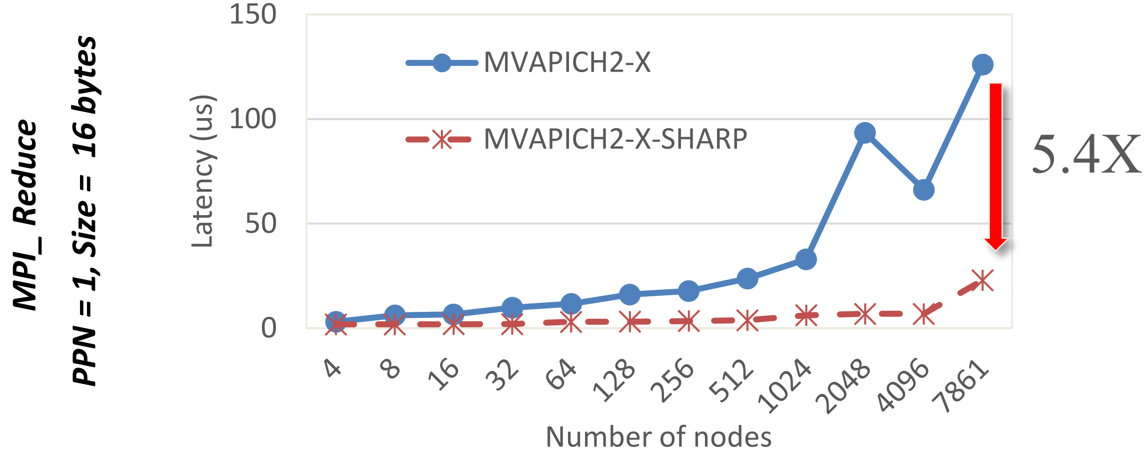


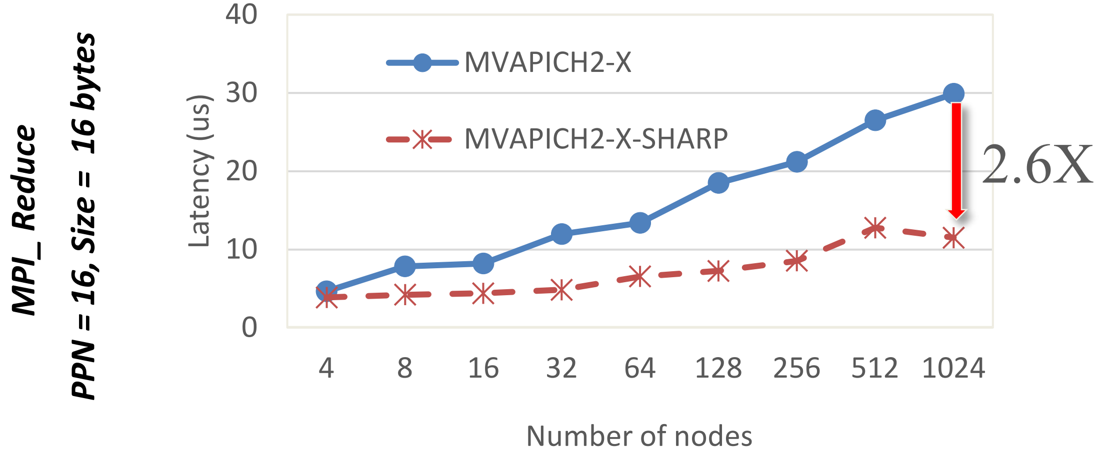
- Near flat scaling with SHARP for varying message sizes and node counts

PERFORMANCE OF MPI_REDUCE WITH SHARP



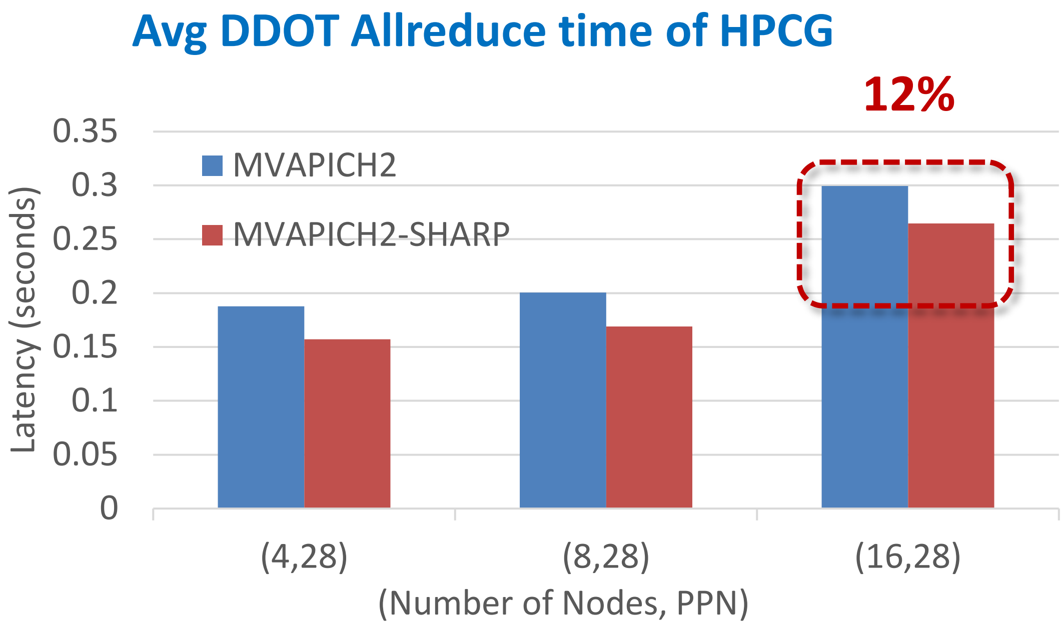






- Near flat scaling with SHARP for varying message sizes and node counts

APPLICATION-LEVEL BENEFITS WITH MVAPICH2-SHARP



- Refer to **Running Collectives with Hardware based SHARP support** section of MVAPICH2 user guide for more information
- https://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3.5-userguide.html#x1-1050006.27

OUTLINE

- Introduction and Motivation
- Overview of the MVAPICH2 project
- Hardware tag matching in MPI
- Offloading with Scalable Hierarchical Aggregation Protocol (SHArP)
- Conclusions and Future Work

CONCLUSION AND FUTURE WORK

- **Hardware based Tag Matching** designs show good benefits, especially at scale
  - Improved the performance of non-blocking collectives up to 42% on 1,024 nodes
  - Up to a 35% improvement for point-to-point operations
- **SHARP based offload has significant benefits in latency**
  - Up to 5.4X reduction in latency for MPI_Reduce at 7,861 nodes over baseline
  - Up to 7.1X reduction in latency for MPI_Barrier
  - Up to 5.1X reduction in latency for MPI_Allreduce
- **Future work**
  - Comprehensive evaluation with applications at large scales
  - Scaling studies with larger number of processes per node
  - Exploration of streaming aggregation with SHARP
  - All features presented will be available in future releases of MVAPICH2

THANK YOU!

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

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

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

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