



MVAPICH

MPI, PGAS and Hybrid MPI+PGAS Library

Latest version of the slides available at

<http://cse.osu.edu/~subramon/mug19-mvapich2-tutorial.pdf>

How to Boost the Performance of Your HPC/AI Applications with MVAPICH2 Libraries?

A Tutorial at MUG'19

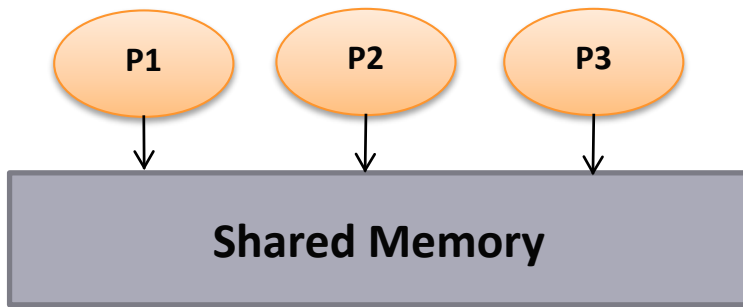
by

The MVAPICH Team

The Ohio State University

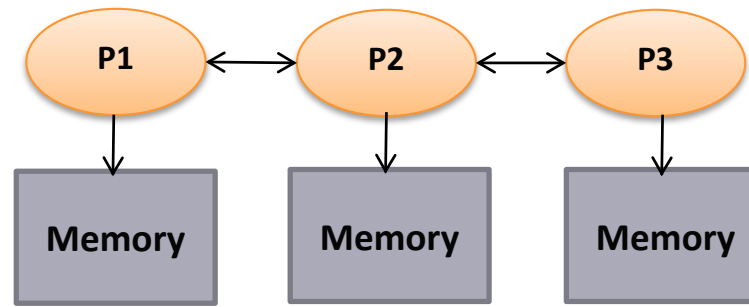
<http://mvapich.cse.ohio-state.edu/>

Parallel Programming Models Overview



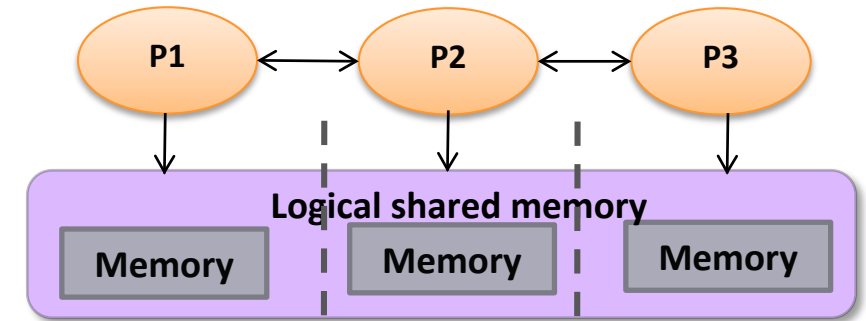
Shared Memory Model

SHMEM, DSM



Distributed Memory Model

MPI (Message Passing Interface)

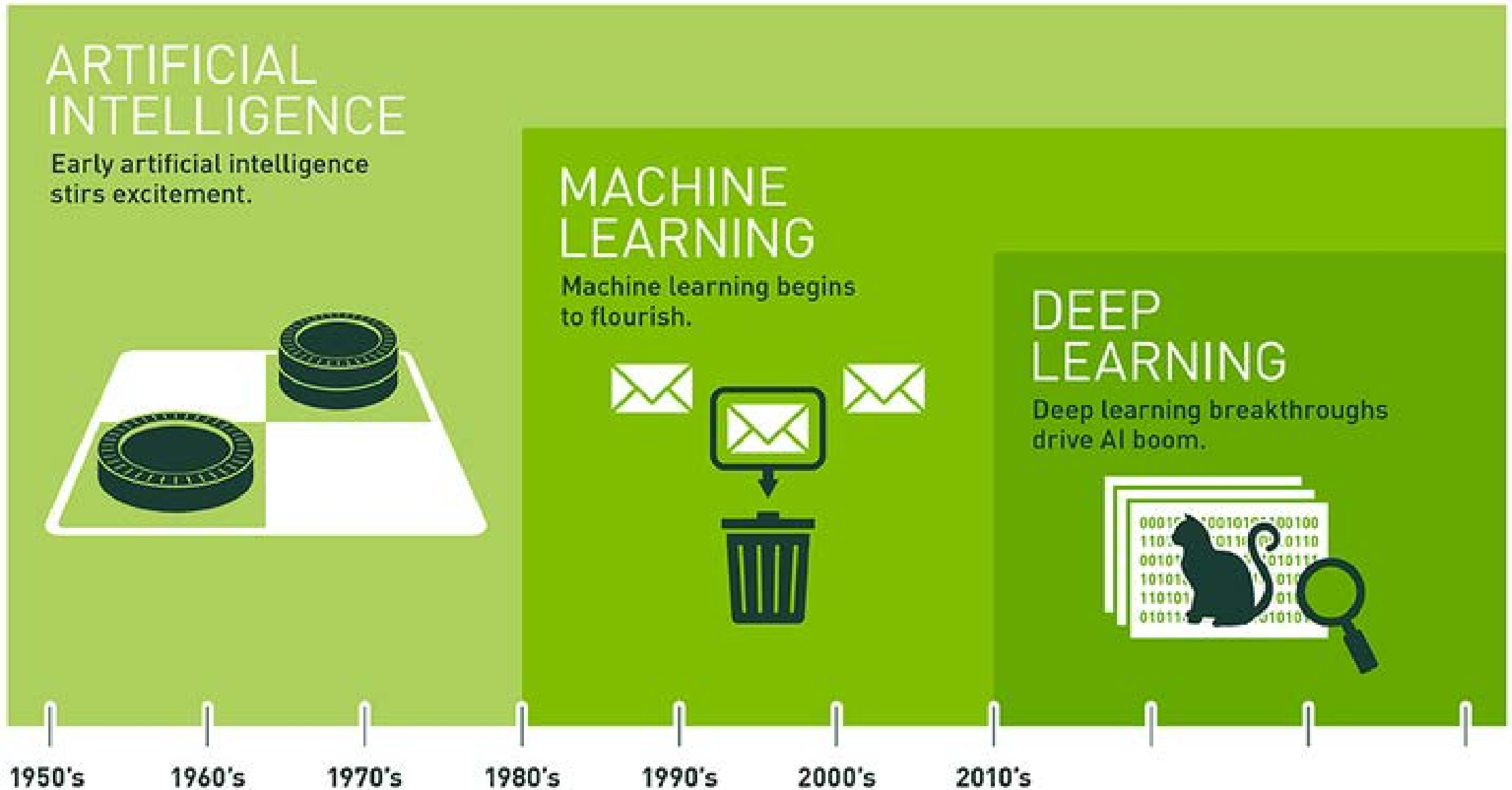


Partitioned Global Address Space (PGAS)

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.
- PGAS models and Hybrid MPI+PGAS models are gradually receiving importance

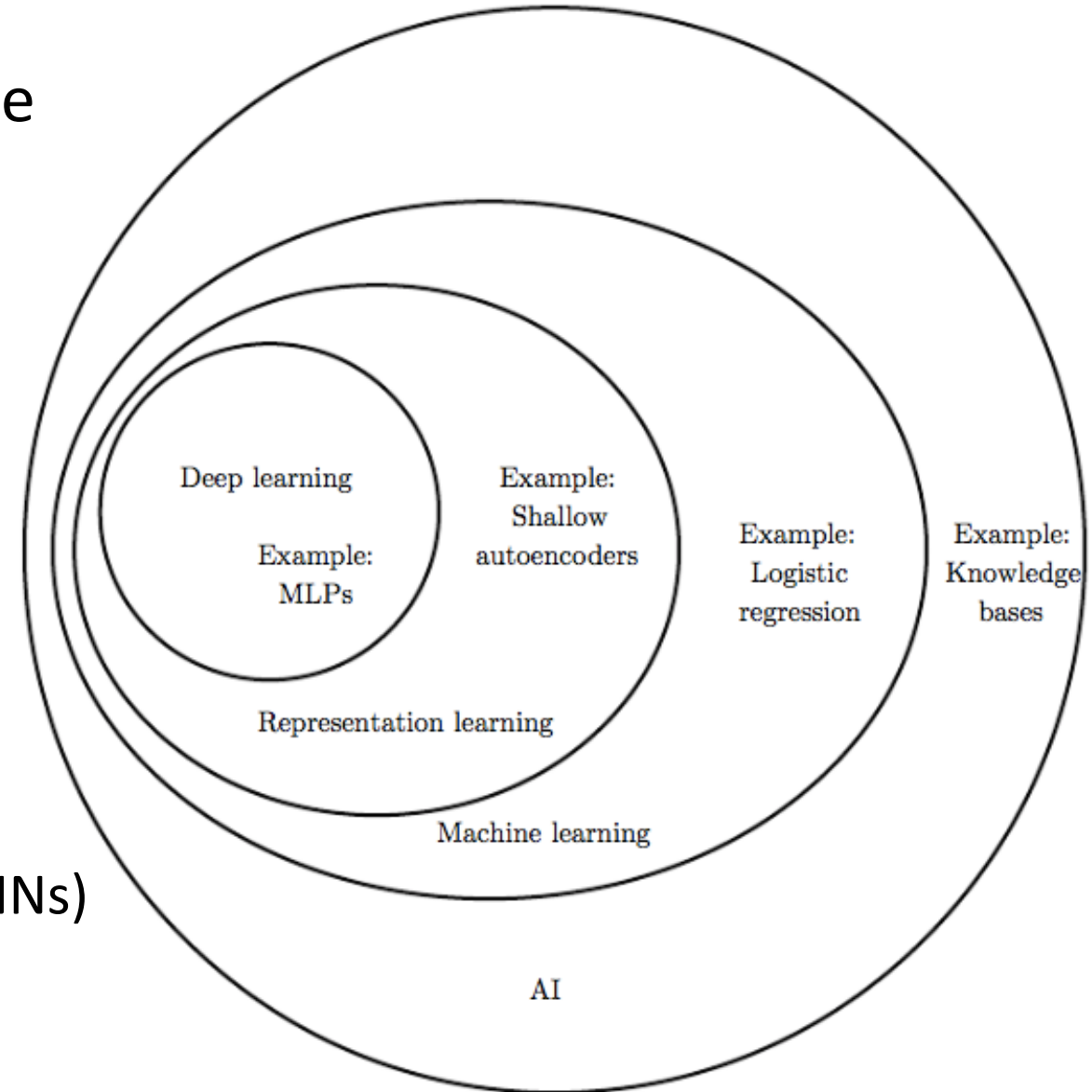
Brief History of Deep Learning (DL)



Courtesy: <http://www.zdnet.com/article/caffe2-deep-learning-wide-ambitions-flexibility-scalability-and-advocacy/>

Understanding the Deep Learning Resurgence

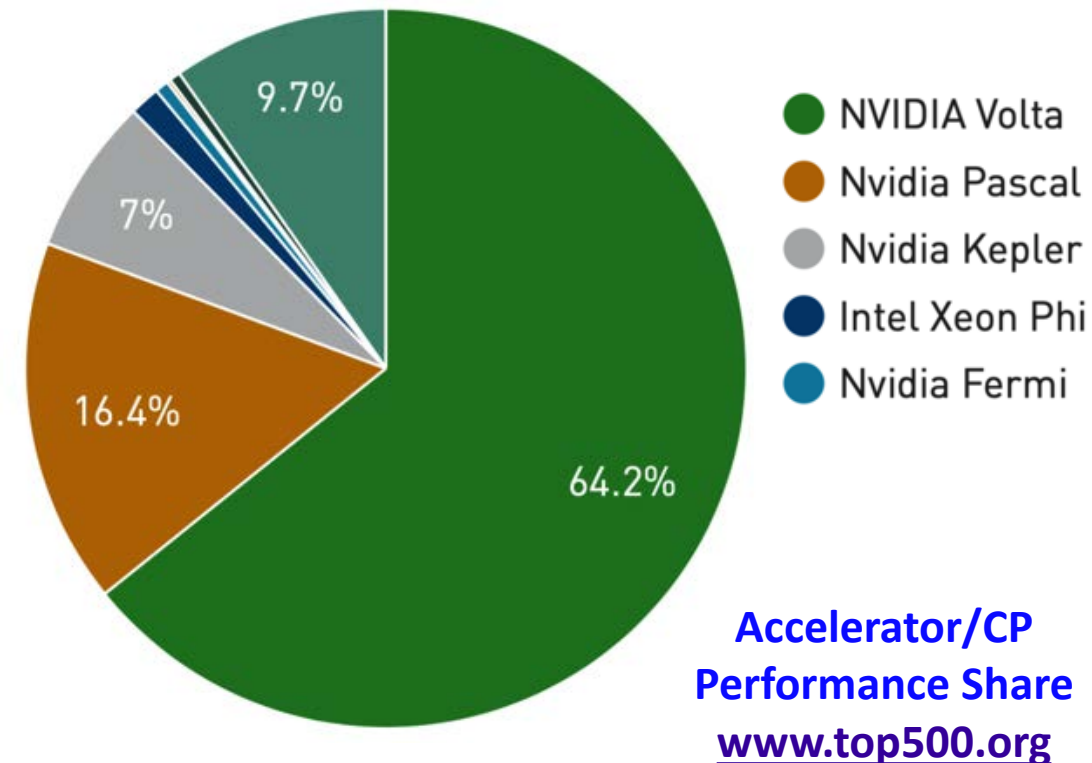
- Deep Learning is a sub-set of Machine Learning
 - But, it is perhaps the most radical and revolutionary subset
 - Automatic feature extraction vs. hand-crafted features
- Deep Learning
 - A renewed interest and a lot of hype!
 - Key success: Deep Neural Networks (DNNs)
 - Everything was there since the late 80s except the “**computability of DNNs**”



Courtesy: <http://www.deeplearningbook.org/contents/intro.html>

Deep Learning, Many-cores, and HPC

- NVIDIA GPUs are the main driving force for faster training of DL models
 - The ImageNet Challenge - (ILSVRC)
 - 90% of the ImageNet teams used GPUs in 2014*
 - Deep Neural Networks (DNNs) like AlexNet, GoogLeNet, and VGG are used
 - A natural fit for DL due to the throughput-oriented nature
- In the High Performance Computing (HPC) arena
 - 124/500 Top HPC systems use NVIDIA GPUs (Jun '19)
 - CUDA-Aware Message Passing Interface (MPI)
 - NVIDIA Fermi, Kepler, Pascal, and Volta architecture
 - DGX-1 (Pascal) and DGX-2 (Volta)
 - Dedicated DL supercomputers



*<https://blogs.nvidia.com/blog/2014/09/07/imagenet/>

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

Application Kernels/Applications (HPC and DL)

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, 40/100/200GigE,
Aries, and Omni-Path)

**Multi-/Many-core
Architectures**

**Accelerators
(GPU and FPGA)**

**Co-Design
Opportunities
and Challenges
across Various
Layers**

**Performance
Scalability
Resilience**

Designing (MPI+X) for Exascale

- Scalability for million to billion processors
 - Support for highly-efficient inter-node and intra-node communication (both two-sided and one-sided)
- Scalable Collective communication
 - Offloaded
 - Non-blocking
 - Topology-aware
- Balancing intra-node and inter-node communication for next generation multi-/many-core (128-1024 cores/node)
 - Multiple end-points per node
- Support for efficient multi-threading
- Integrated Support for GPGPUs and Accelerators
- Fault-tolerance/resiliency
- QoS support for communication and I/O
- Support for Hybrid MPI+PGAS programming
 - MPI + OpenMP, MPI + UPC, MPI + OpenSHMEM, CAF, MPI + UPC++...
- Virtualization
- Energy-Awareness

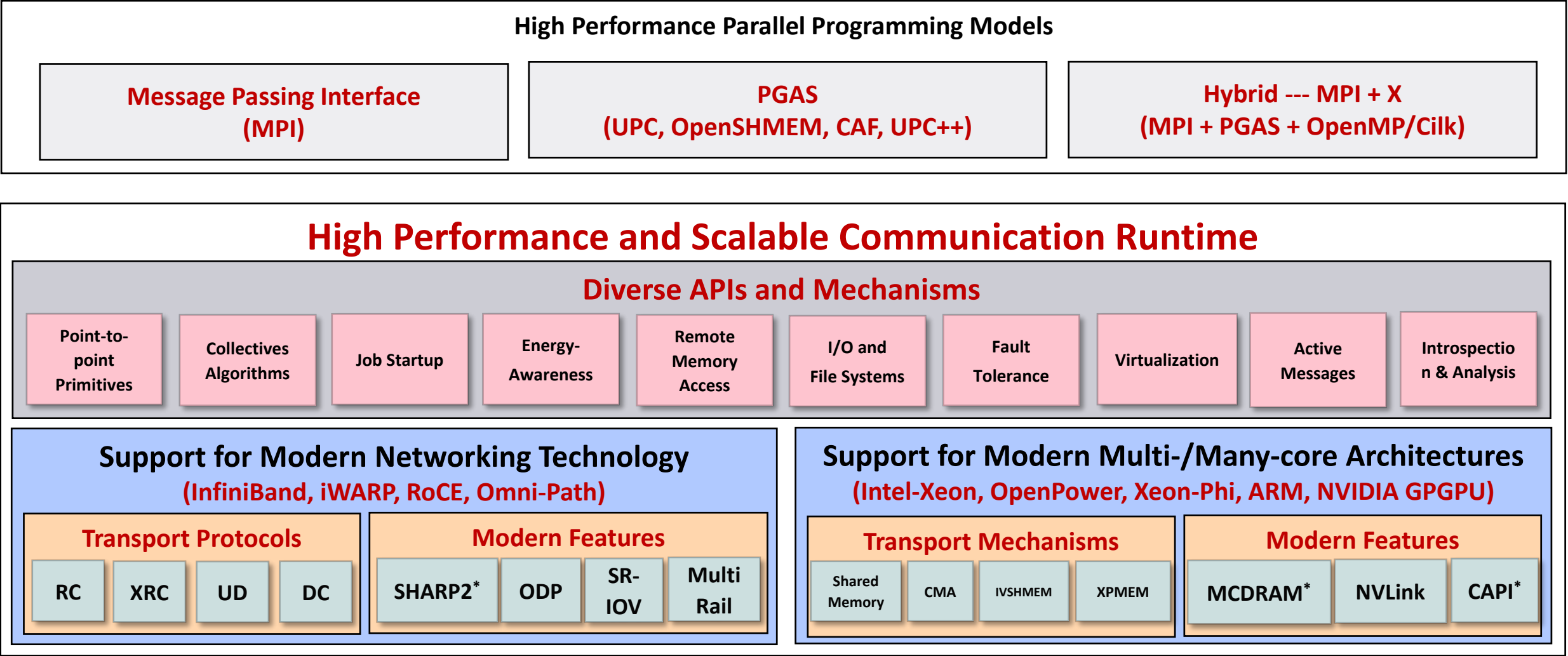
Overview of the MVAPICH2 Project

- 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 3,025 organizations in 89 countries**
 - **More than 563,000 (> 0.5 million) downloads from the OSU site directly**
 - Empowering many TOP500 clusters (Nov '18 ranking)
 - 3rd, 10,649,600-core (Sunway TaihuLight) at National Supercomputing Center in Wuxi, China
 - 5th, 448, 448 cores (Frontera) at TACC
 - 8th, 391,680 cores (ABCI) in Japan
 - 15th, 570,020 cores (Neurion) in South Korea and many others
 - Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, and OpenHPC)
 - <http://mvapich.cse.ohio-state.edu>
- Empowering Top500 systems for over a decade



Partner in the TACC Frontera System

Architecture of MVAPICH2 Software Family (for HPC and DL)



* Upcoming

Strong Procedure for Design, Development and Release

- Research is done for exploring new designs
- Designs are first presented to conference/journal publications
- Best performing designs are incorporated into the codebase
- Rigorous Q&A procedure before making a release
 - Exhaustive unit testing
 - Various test procedures on diverse range of platforms and interconnects
 - Performance tuning
 - Applications-based evaluation
 - Evaluation on large-scale systems
- Even alpha and beta versions go through the above testing

MVAPICH2 Software Family

Requirements	Library
MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)	MVAPICH2
Optimized Support for Microsoft Azure Platform with InfiniBand	MVAPICH2-Azure
Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis),	MVAPICH2-X
Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric Adapter (EFA)	MVAPICH2-X-AWS
Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications	MVAPICH2-GDR
Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)	MVAPICH2-EA
MPI Energy Monitoring Tool	OEMT
InfiniBand Network Analysis and Monitoring	OSU INAM
Microbenchmarks for Measuring MPI and PGAS Performance	OMB

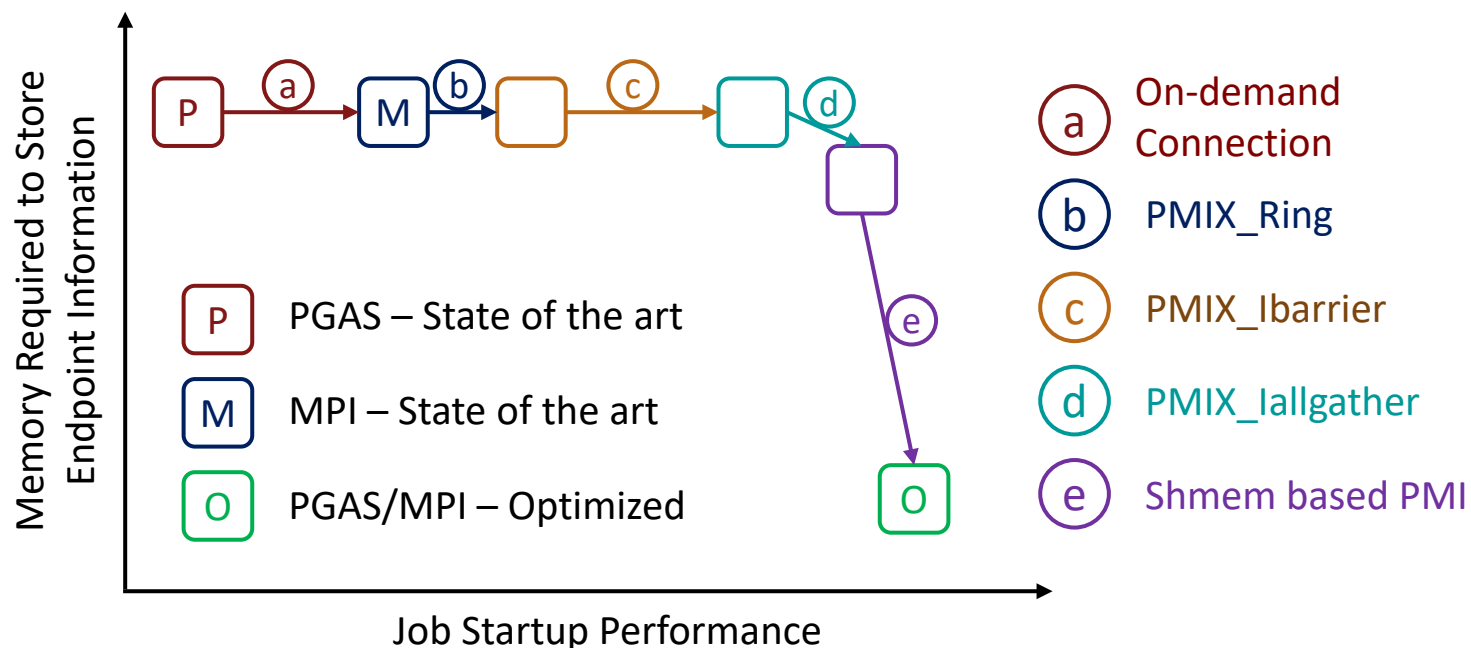
MVAPICH2 2.3.2

- Released on 08/09/2019
- Major Features and Enhancements
 - Improved performance for inter-node communication
 - Improved performance for Gather, Reduce, and Allreduce with cyclic hostfile
 - Thanks to X-ScaleSolutions for the patch
 - Improved performance for intra-node point-to-point communication
 - Add support for Mellanox HDR adapters
 - Add support for Cascade lake systems
 - Add support for Microsoft Azure platform
 - Enhanced point-to-point and collective tuning for Microsoft Azure
 - Add support for new NUMA-aware hybrid binding policy
 - Add support for AMD EPYC Rome architecture
 - Improved multi-rail selection logic
 - Enhanced heterogeneity detection logic
 - Enhanced point-to-point and collective tuning for AMD EPYC Rome, Frontera@TACC, Mayer@Sandia, Pitzer@OSC, Summit@ORNL, Lassen@LLNL, and Sierra@LLNL systems
 - Add multiple PVARs and CVARs for point-to-point and collective operations

Overview of MVAPICH2 Features

- **Job start-up**
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

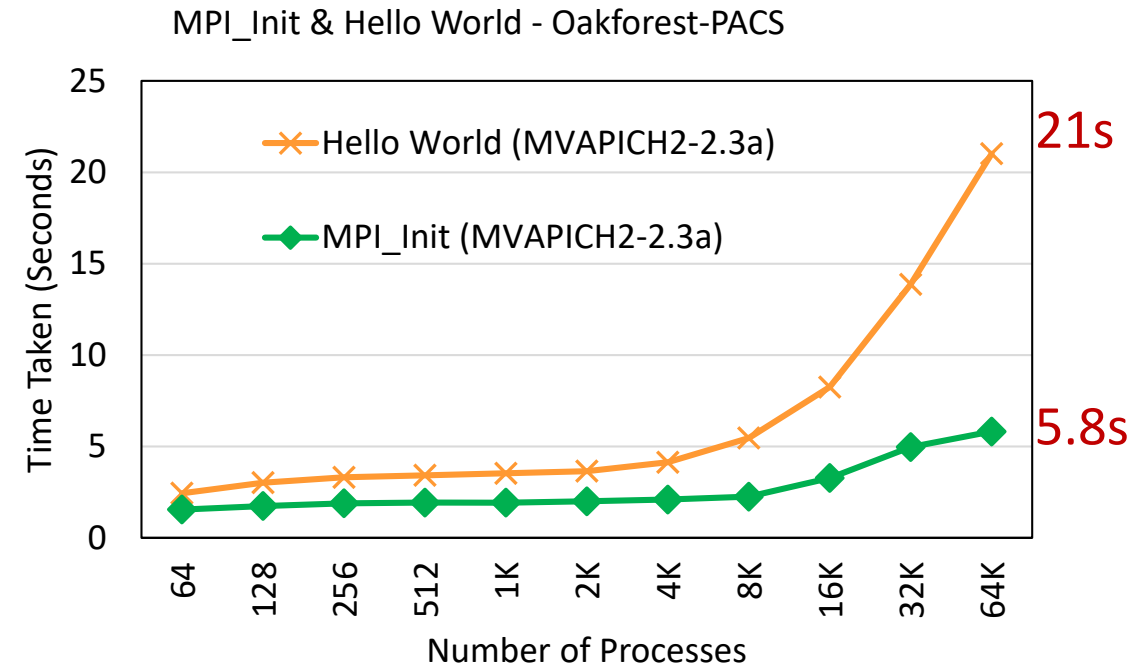
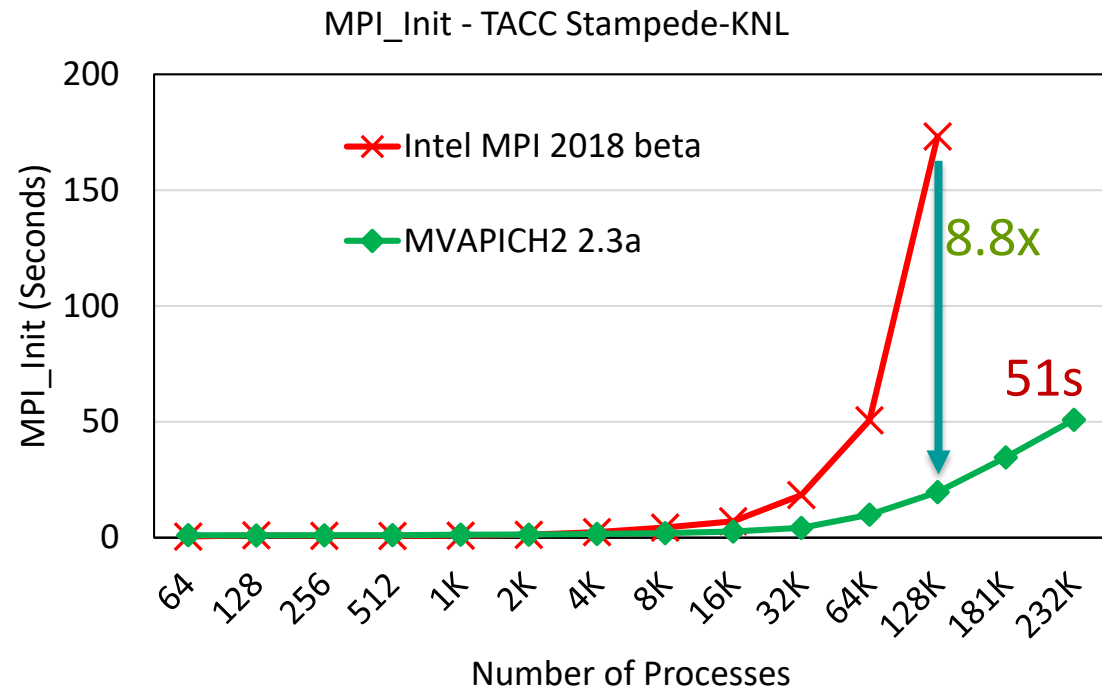
Towards High Performance and Scalable Startup at Exascale



- Near-constant MPI and OpenSHMEM initialization time at any process count
- 10x and 30x improvement in startup time of MPI and OpenSHMEM respectively at 16,384 processes
- Memory consumption reduced for remote endpoint information by $O(\text{processes per node})$
- 1GB Memory saved per node with 1M processes and 16 processes per node

- (a) **On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI.** S. Chakraborty, H. Subramoni, J. Perkins, A. A. Awan, and D K Panda, 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS '15)
- (b) **PMI Extensions for Scalable MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, J. Perkins, M. Arnold, and D K Panda, Proceedings of the 21st European MPI Users' Group Meeting (EuroMPI/Asia '14)
- (c) (d) **Non-blocking PMI Extensions for Fast MPI Startup.** S. Chakraborty, H. Subramoni, A. Moody, A. Venkatesh, J. Perkins, and D K Panda, 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '15)
- (e) **SHMEMPMI – Shared Memory based PMI for Improved Performance and Scalability.** S. Chakraborty, H. Subramoni, J. Perkins, and D K Panda, 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '16)

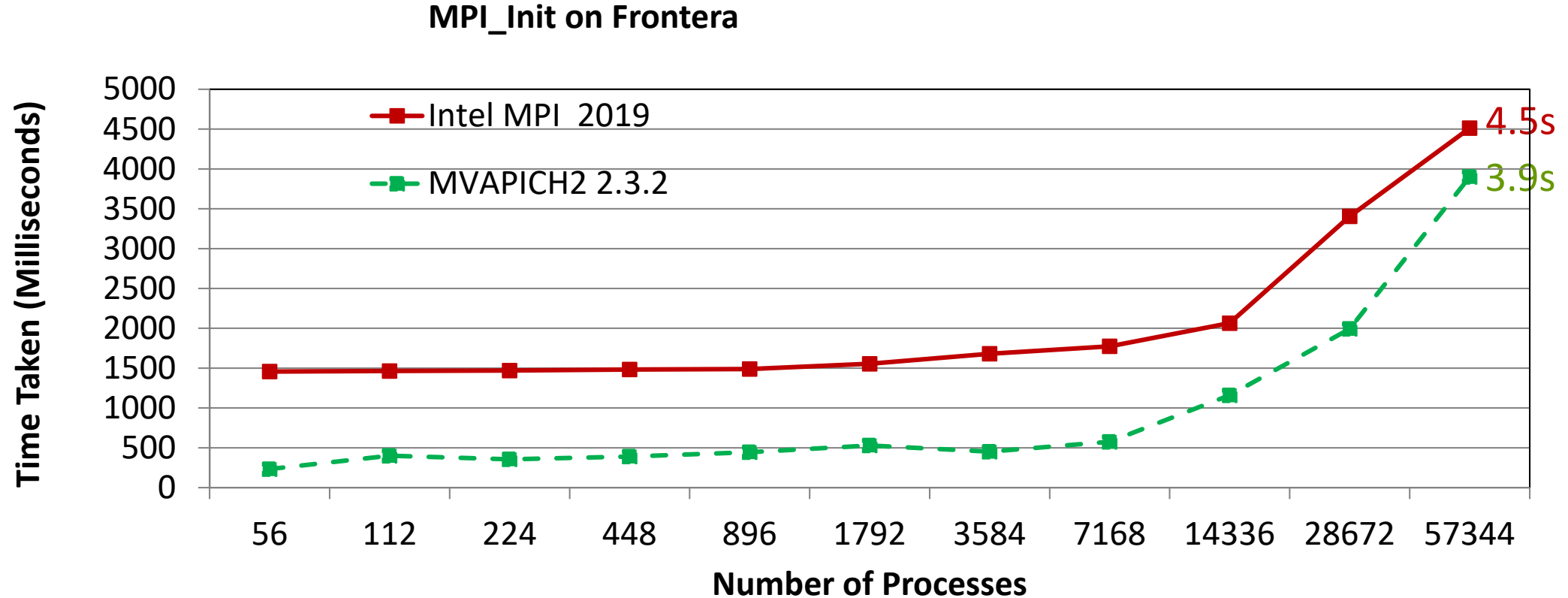
Startup Performance on KNL + Omni-Path



- MPI_Init takes 51 seconds on 231,956 processes on 3,624 KNL nodes (Stampede – Full scale)
- 8.8 times faster than Intel MPI at 128K processes (Courtesy: TACC)
- At 64K processes, MPI_Init and Hello World takes 5.8s and 21s respectively (Oakforest-PACS)
- All numbers reported with 64 processes per node

New designs available in MVAPICH2-2.3a and as patch for SLURM-15.08.8 and SLURM-16.05.1

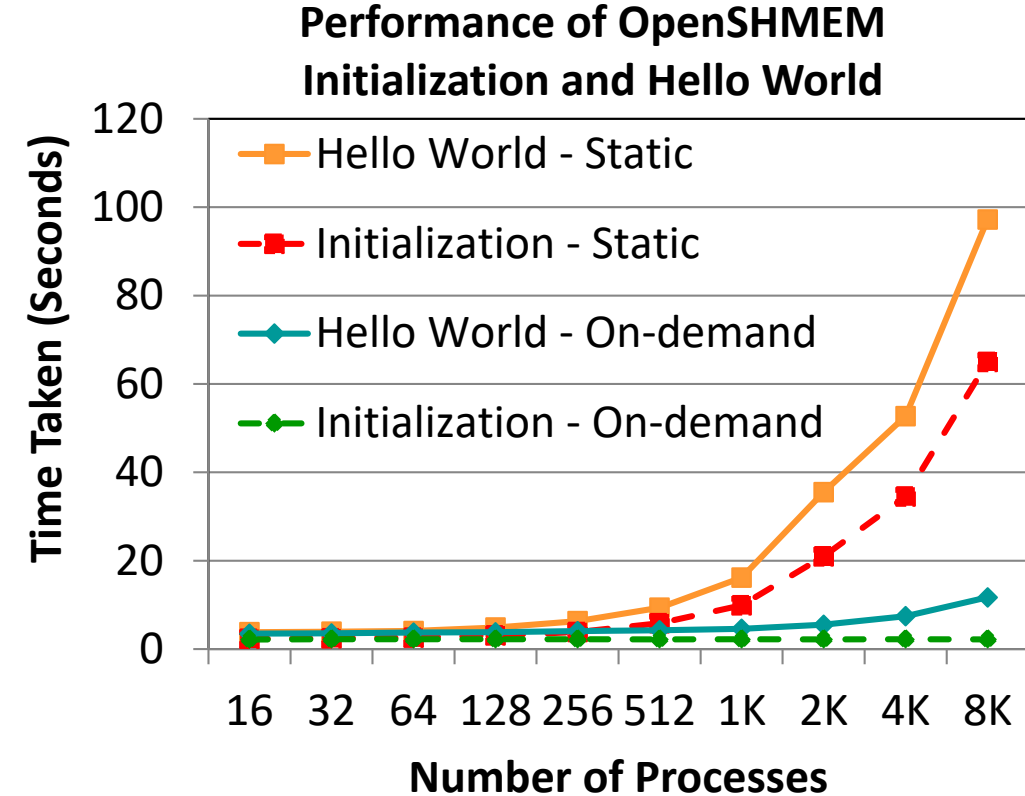
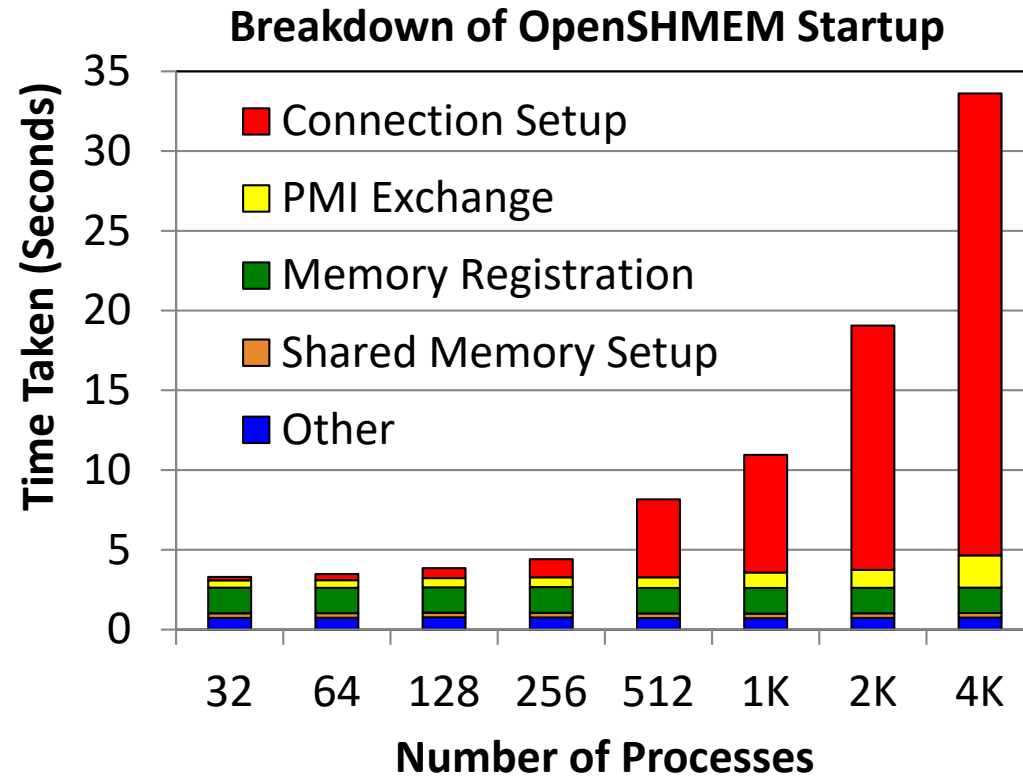
Startup Performance on TACC Frontera



- MPI_Init takes 3.9 seconds on 57,344 processes on 1,024 nodes
- All numbers reported with 56 processes per node

New designs available in MVAPICH2-2.3.2

On-demand Connection Management for OpenSHMEM+MPI



- Static connection establishment wastes memory and takes a lot of time
- On-demand connection management improves OpenSHMEM initialization time by **29.6 times**
- Time taken for Hello World reduced by **8.31 times** at 8,192 processes
- **Available since MVAPICH2-X 2.1rc1**

How to Get the Best Startup Performance with MVAPICH2?

- **MV2_HOMOGENEOUS_CLUSTER=1** //Set for homogenous clusters
- **MV2_ON_DEMAND_UD_INFO_EXCHANGE=1** //Enable UD based address exchange

Using SLURM as launcher

- **Use PMI2**
 - ./configure --with-pm=slurm --with-pmi=pmi2
 - srun --mpi=pmi2 ./a.out
- **Use PMI Extensions**
 - Patch for SLURM available at <http://mvapich.cse.ohio-state.edu/download/>
 - Patches available for SLURM 15, 16, and 17
 - PMI Extensions are automatically detected by MVAPICH2

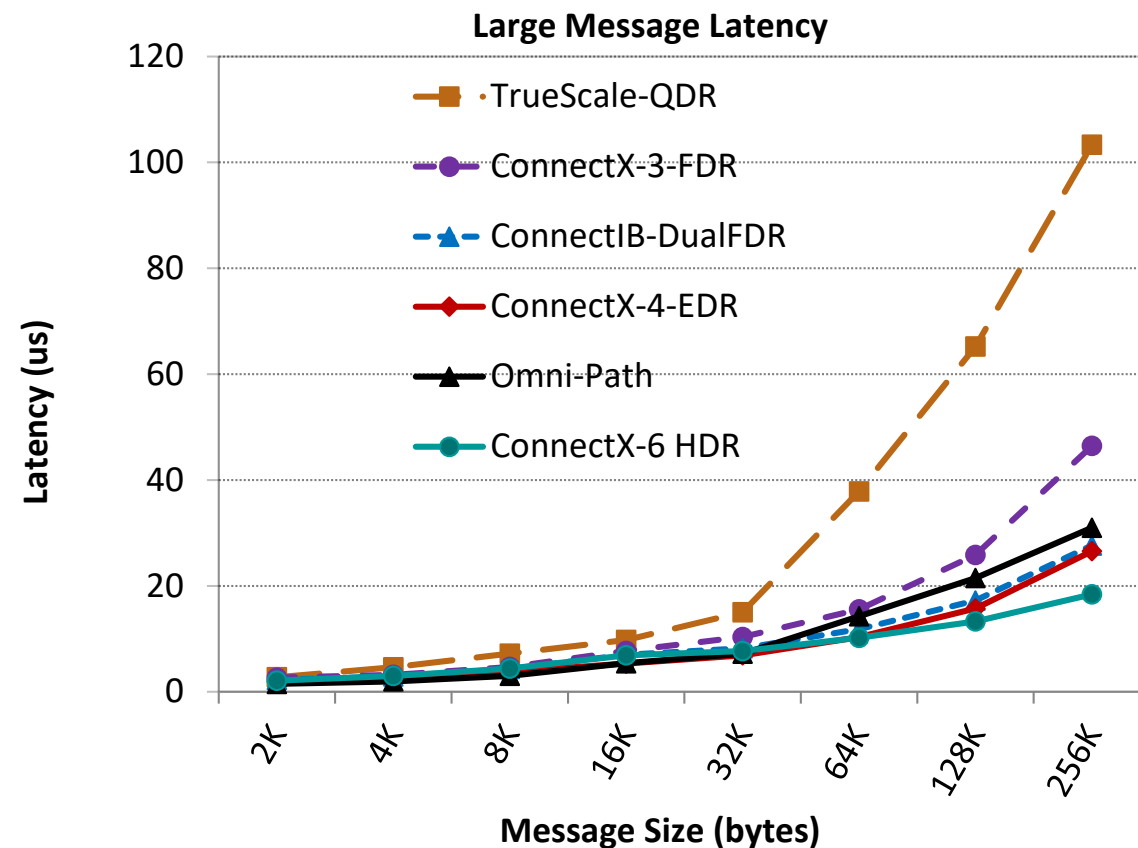
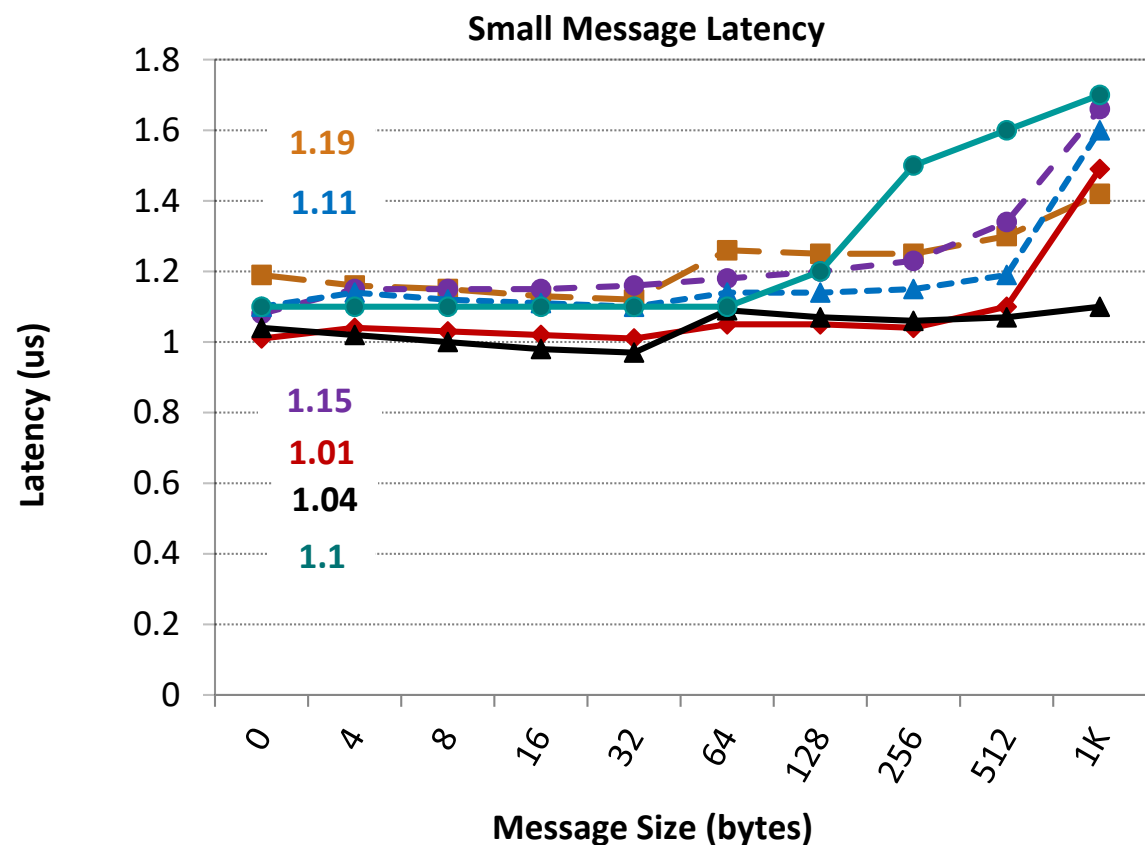
Using mpirun_rsh as launcher

- **MV2_MT_DEGREE**
 - degree of the hierarchical tree used by mpirun_rsh
- **MV2_FASTSSH_THRESHOLD**
 - #nodes beyond which hierarchical-ssh scheme is used
- **MV2_NPROCS_THRESHOLD**
 - #nodes beyond which file-based communication is used for hierarchical-ssh during start up

Overview of MVAPICH2 Features

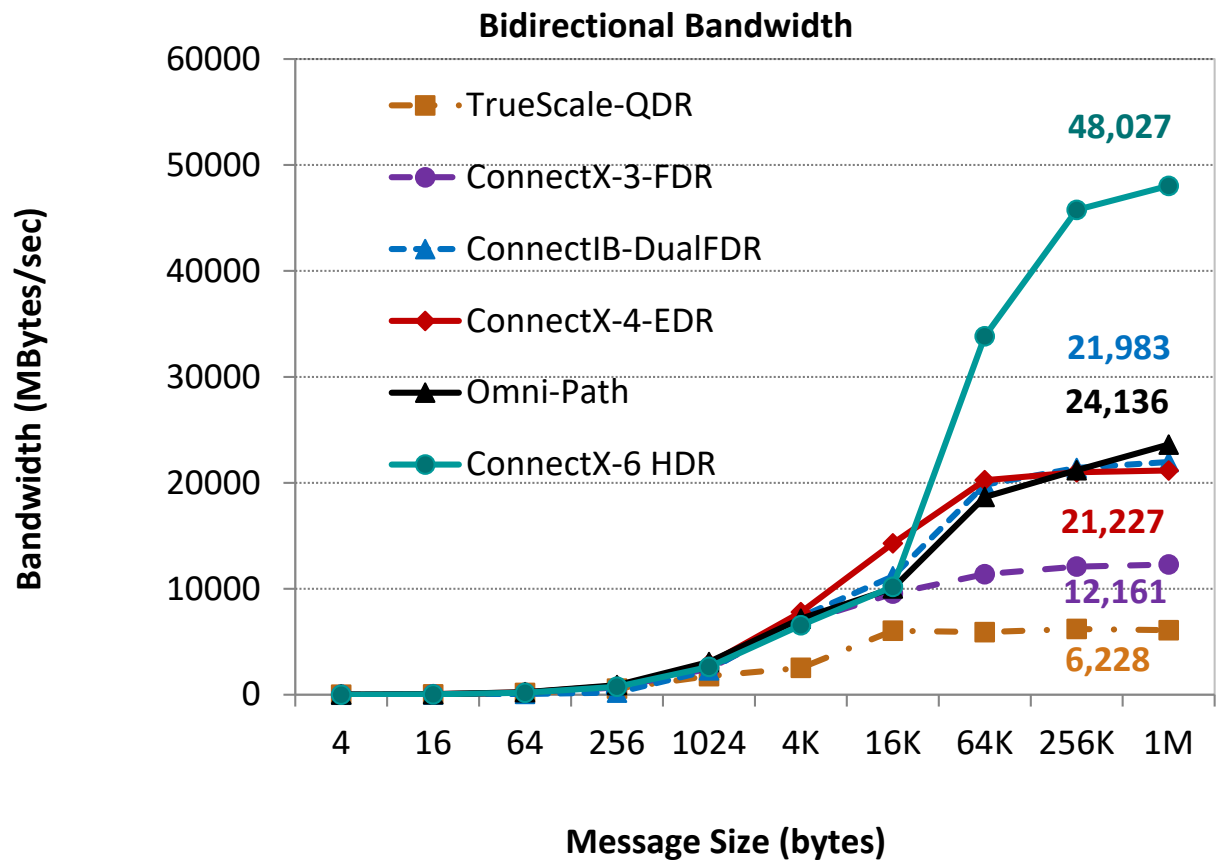
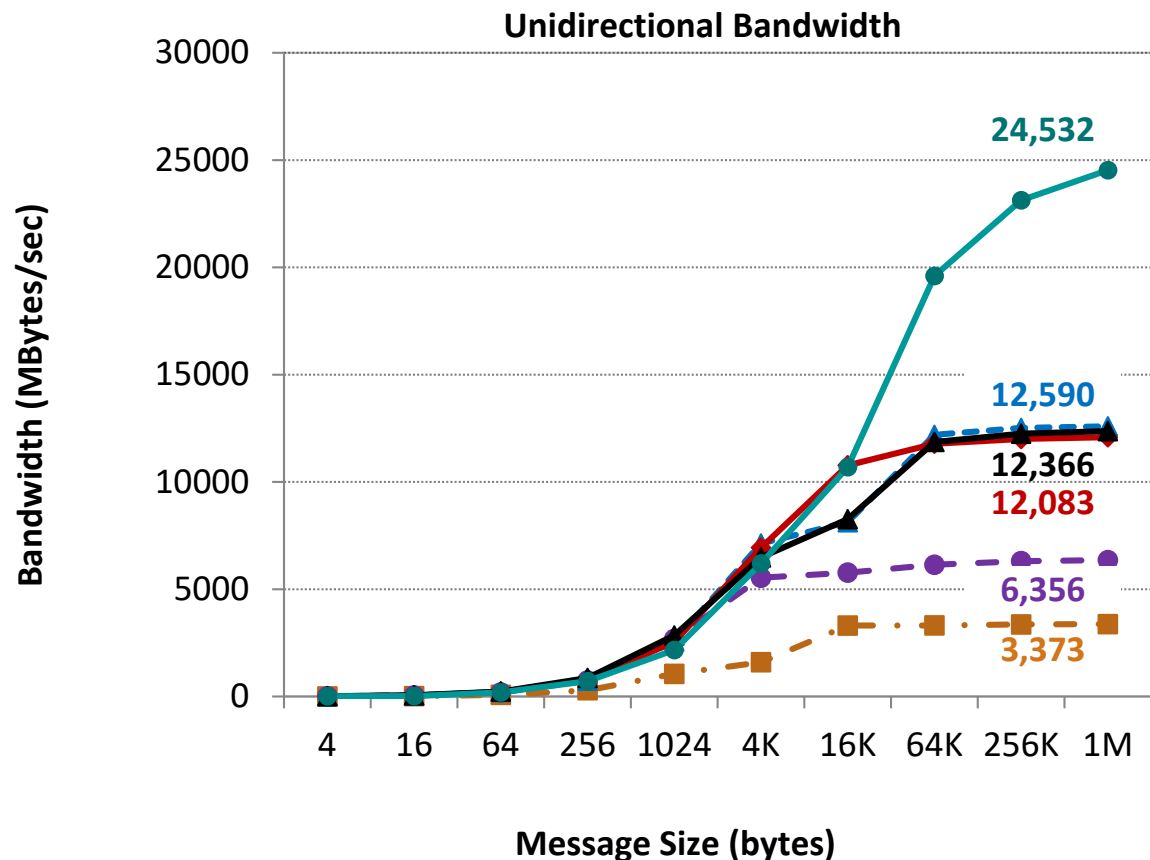
- Job start-up
- **Point-to-point Inter-node Protocol**
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

One-way Latency: MPI over IB with MVAPICH2



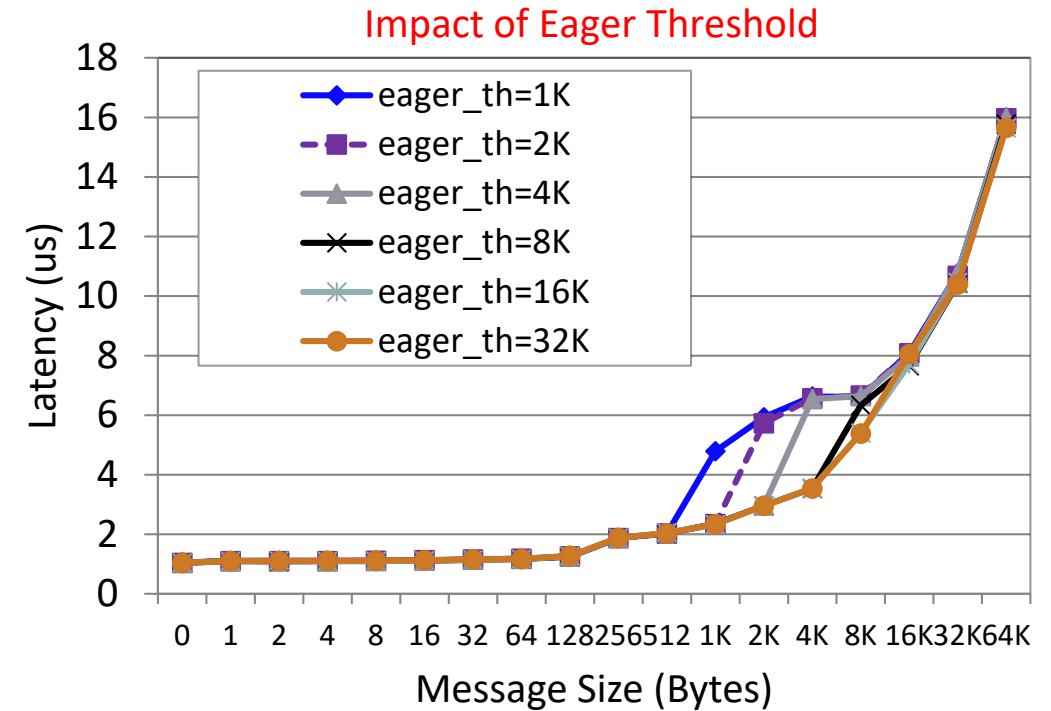
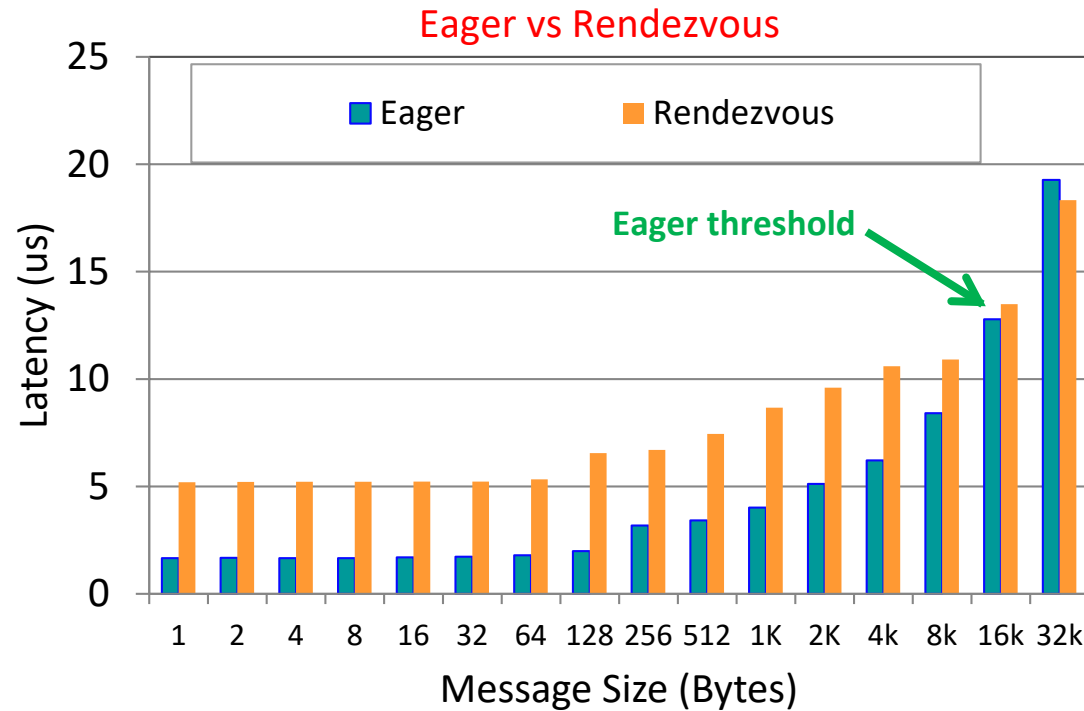
TrueScale-QDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch
ConnectIB-Dual FDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
ConnectX-4-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch
Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch
ConnectX-6-HDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch

Bandwidth: MPI over IB with MVAPICH2



TrueScale-QDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
 ConnectX-3-FDR - 2.8 GHz Deca-core (IvyBridge) Intel PCI Gen3 with IB switch
 ConnectIB-Dual FDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB switch
 ConnectX-4-EDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch
 Omni-Path - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with Omni-Path switch
 ConnectX-6-HDR - 3.1 GHz Deca-core (Haswell) Intel PCI Gen3 with IB Switch

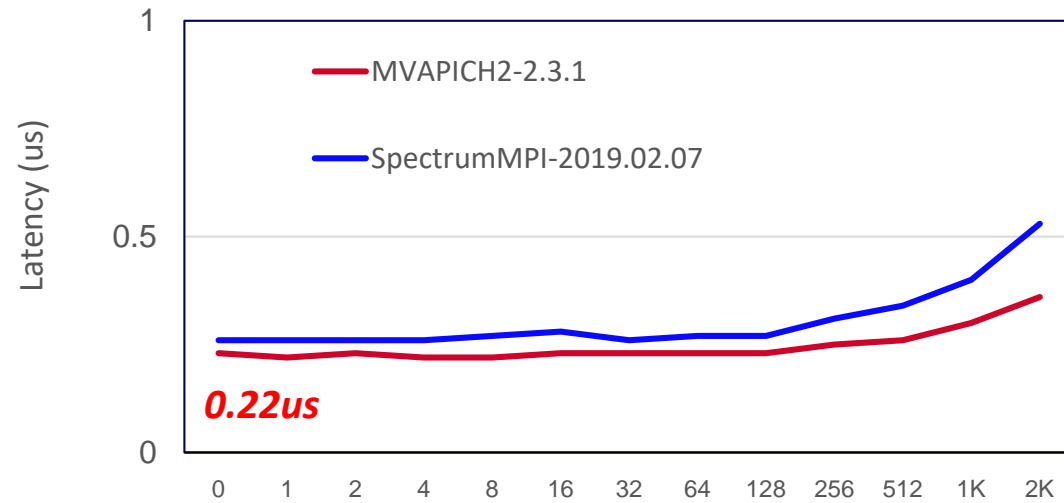
Inter-node Point-to-Point Tuning: Eager Thresholds



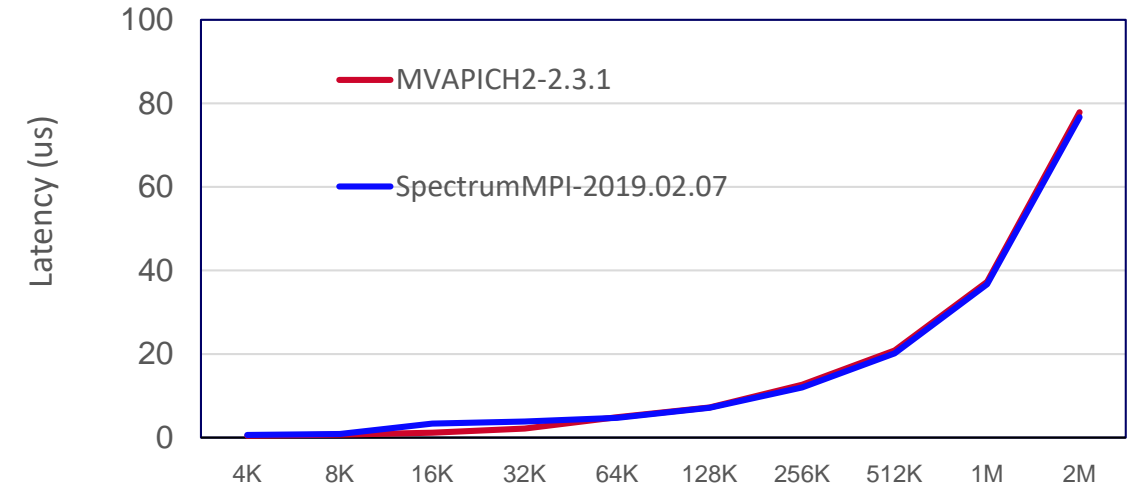
- Switching Eager to Rendezvous transfer
 - Default: Architecture dependent on common platforms, in order to achieve both best performance and memory footprint
- Threshold can be modified by users to get smooth performance across message sizes
 - `mpirun_rsh -np 2 -hostfile hostfile MV2_IBA_EAGER_THRESHOLD=32K a.out`
 - Memory footprint can increase along with eager threshold

Intra-node Point-to-Point Performance on OpenPower

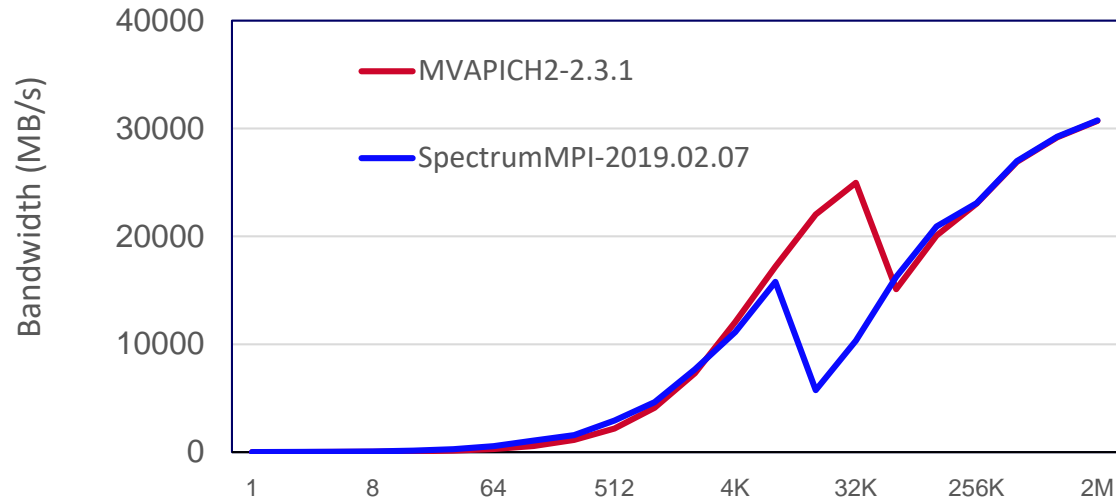
Intra-Socket Small Message Latency



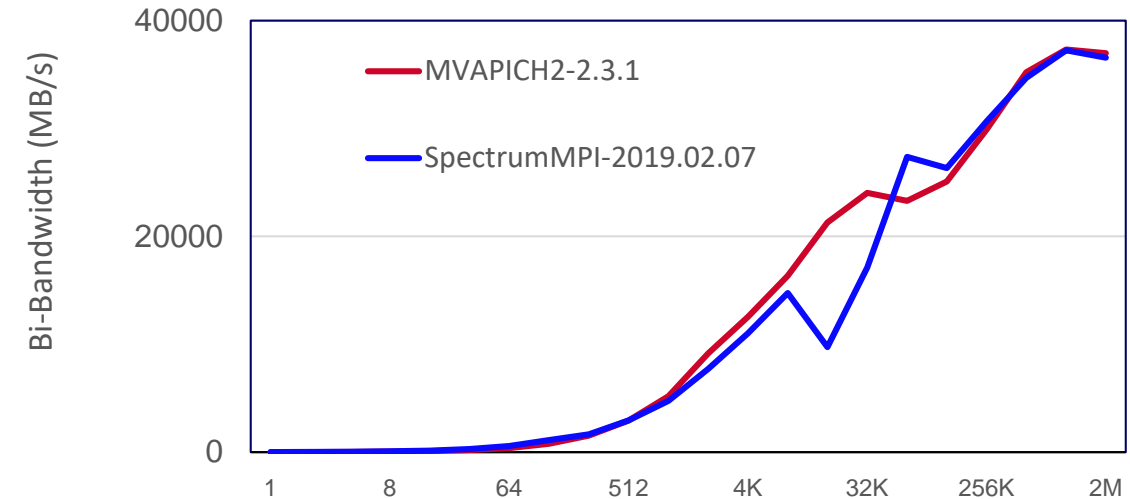
Intra-Socket Large Message Latency



Intra-Socket Bandwidth



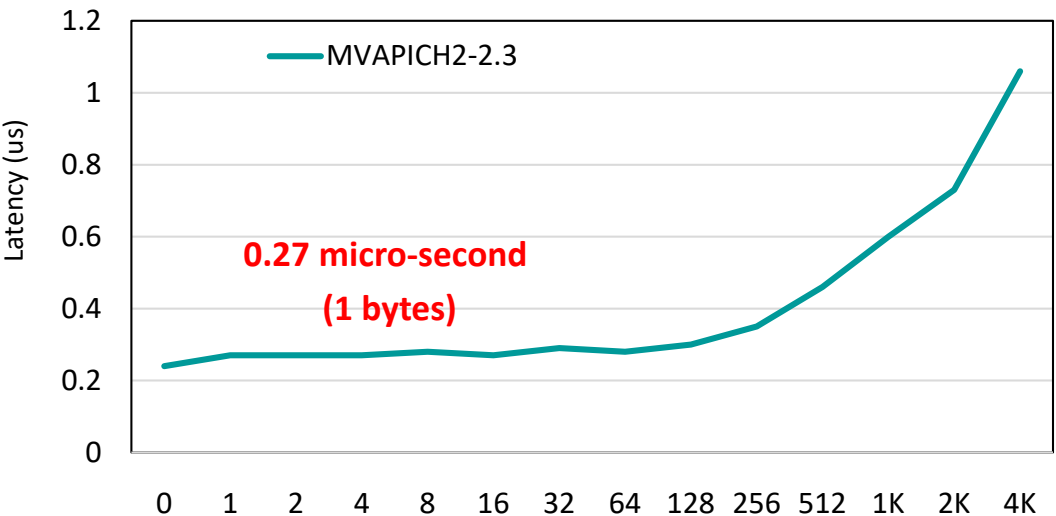
Intra-Socket Bi-directional Bandwidth



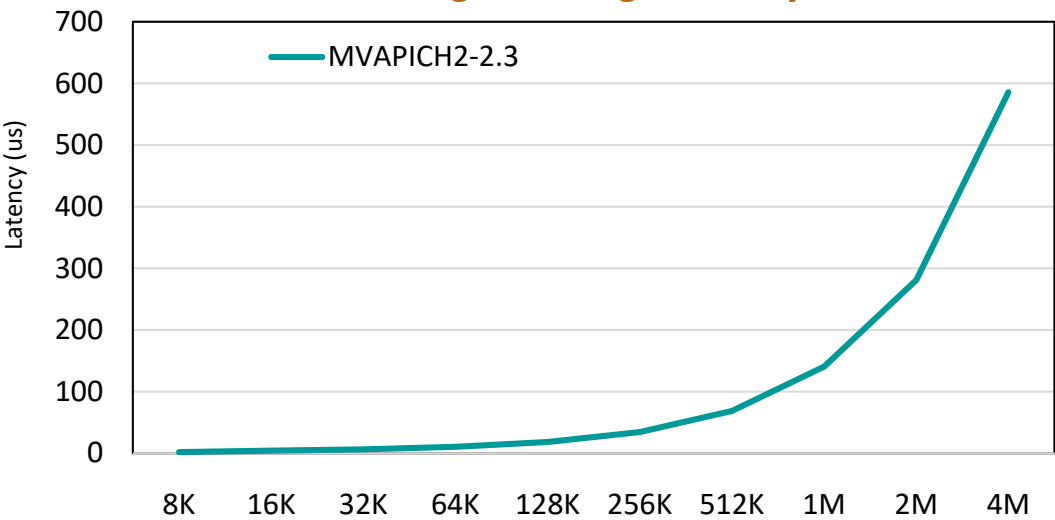
Platform: Two nodes of OpenPOWER (POWER9-ppc64le) CPU using Mellanox EDR (MT4121) HCA

Intra-node Point-to-point Performance on ARM Cortex-A72

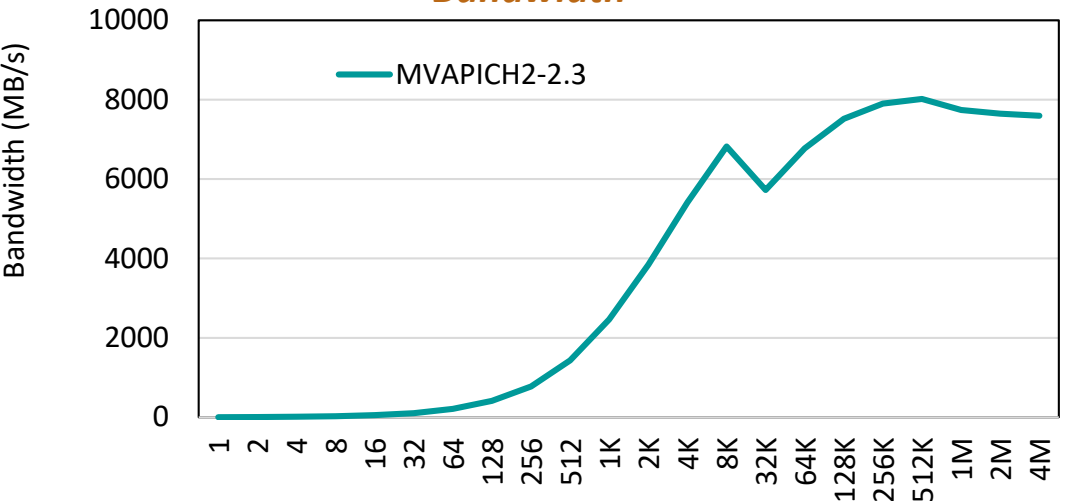
Small Message Latency



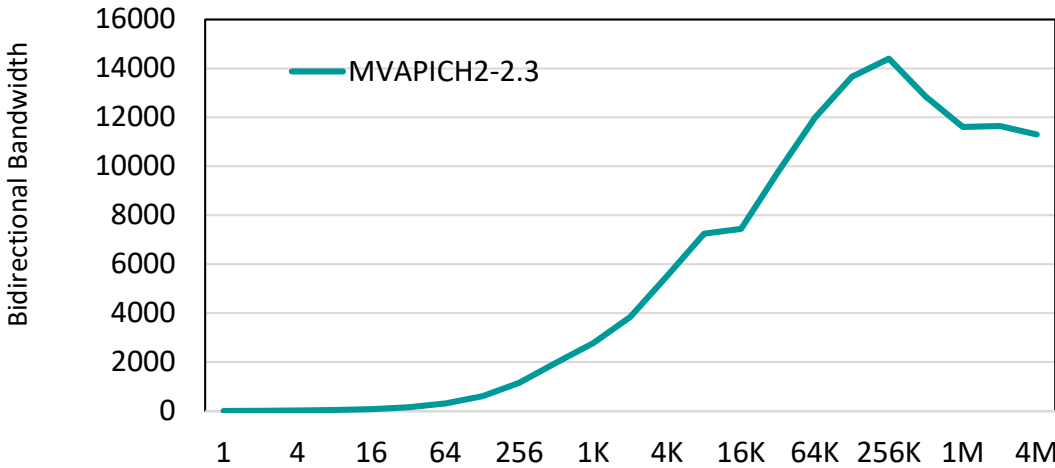
Large Message Latency



Bandwidth



Bi-directional Bandwidth

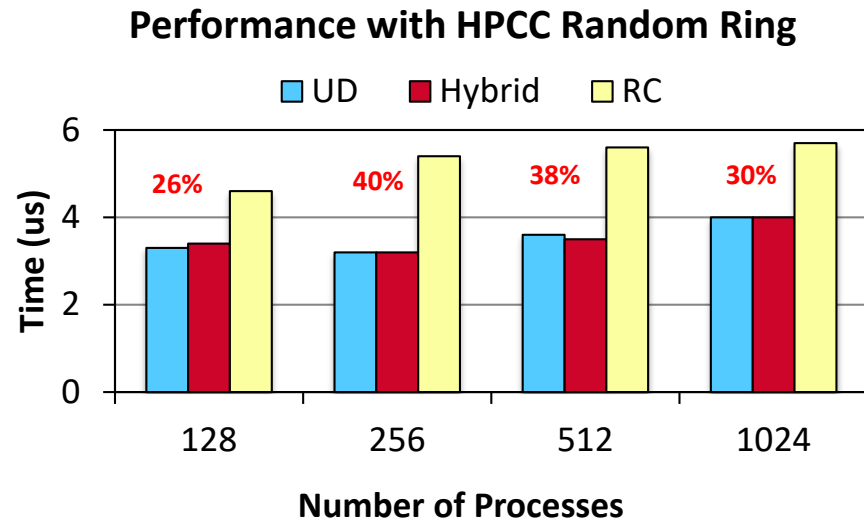


Platform: ARM Cortex A72 (aarch64) processor with 64 cores dual-socket CPU. Each socket contains 32 cores.

Overview of MVAPICH2 Features

- Job start-up
- Point-to-point Inter-node Protocol
- **Transport Type Selection**
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

Hybrid (UD/RC/XRC) Mode in MVAPICH2



- Both UD and RC/XRC have benefits
 - Hybrid for the best of both
- Enabled by configuring MVAPICH2 with the `--enable-hybrid`
- Available since MVAPICH2 1.7 as integrated interface

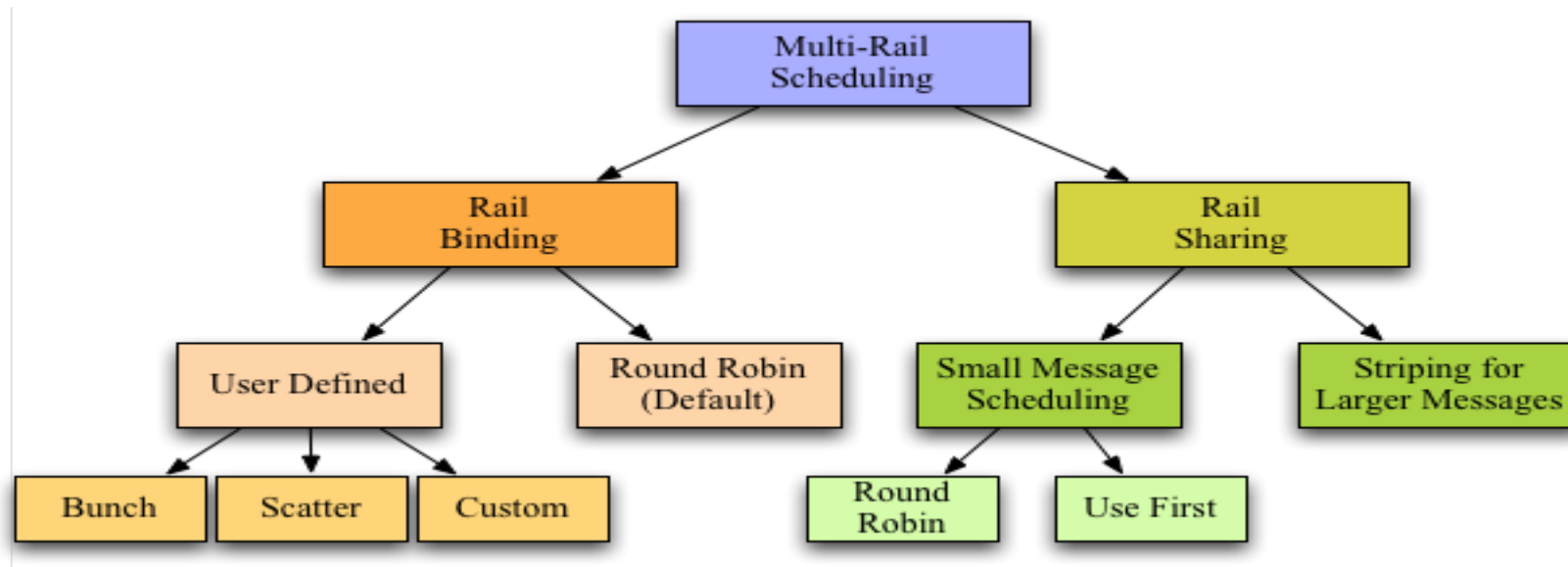
Parameter	Significance	Default	Notes
MV2_USE_UD_HYBRID	• Enable / Disable use of UD transport in Hybrid mode	Enabled	• Always Enable
MV2_HYBRID_ENABLE_THRESHOLD_SIZE	• Job size in number of processes beyond which hybrid mode will be enabled	1024	• Uses RC/XRC connection until job size < threshold
MV2_HYBRID_MAX_RC_CONN	• Maximum number of RC or XRC connections created per process • Limits the amount of connection memory	64	• Prevents HCA QP cache thrashing

- Refer to **Running with Hybrid UD-RC/XRC** section of MVAPICH2 user guide for more information
- <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3a-userguide.html#x1-690006.11>

Overview of MVAPICH2 Features

- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- **Multi-rail**
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- MPI_T Support

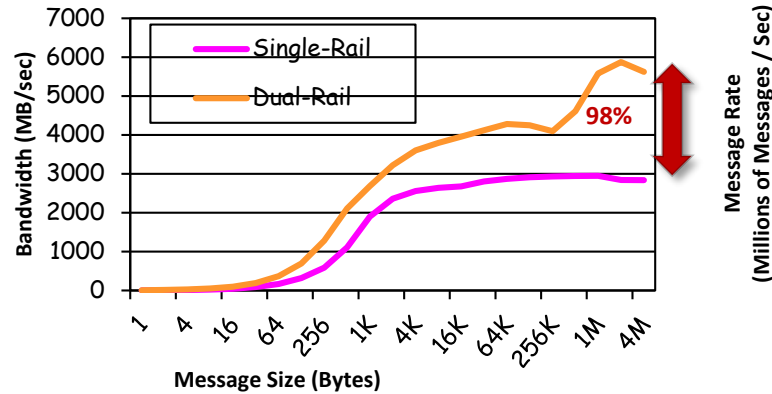
MVAPICH2 Multi-Rail Design



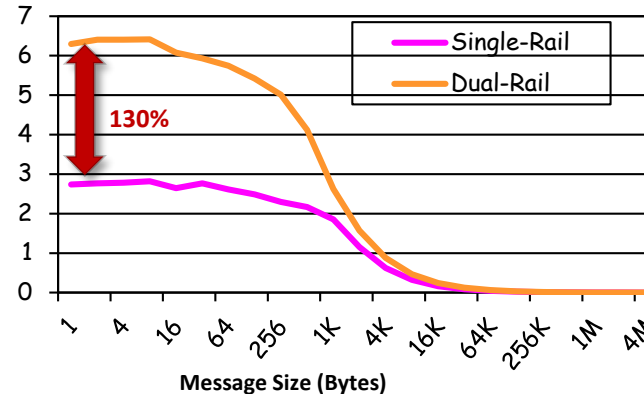
- What is a rail?
 - **HCA, Port, Queue Pair**
- Automatically detects and uses all active HCAs in a system
 - Automatically handles heterogeneity
- Supports multiple rail usage policies
 - Rail Sharing – Processes share all available rails
 - Rail Binding – Specific processes are bound to specific rails

Performance Tuning on Multi-Rail Clusters

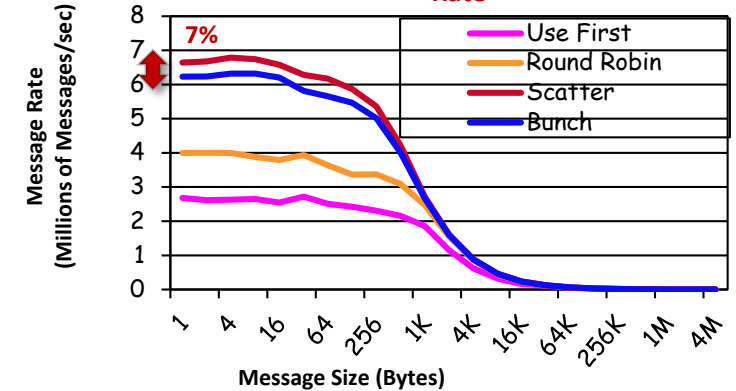
Impact of Default Message Striping on Bandwidth



Impact of Default Rail Binding on Message Rate



Impact of Advanced Multi-rail Tuning on Message Rate



Two 24-core Magny Cours nodes with two Mellanox ConnectX QDR adapters
Six pairs with OSU Multi-Pair bandwidth and messaging rate benchmark

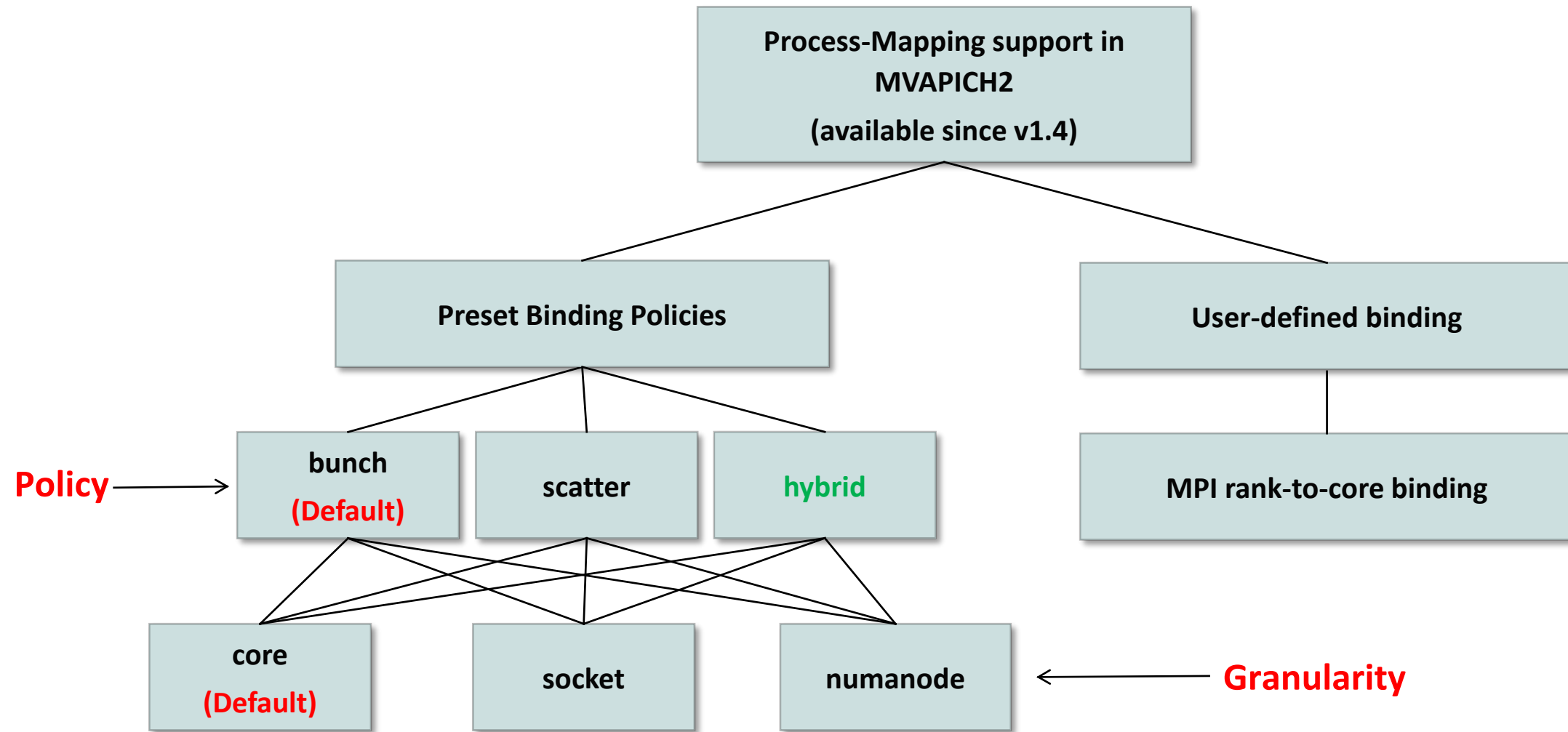
Parameter	Significance	Default	Notes
MV2_IBA_HCA	• Manually set the HCA to be used	Unset	• To get names of HCA ibstat grep “^CA”
MV2_DEFAULT_PORT	• Select the port to use on a active multi port HCA	0	• Set to use different port
MV2_RAIL_SHARING_LARGE_MSG_THRESHOLD	• Threshold beyond which striping will take place	16 Kbyte	
MV2_RAIL_SHARING_POLICY	• Choose multi-rail rail sharing / binding policy • For Rail Sharing set to USE_FIRST or ROUND_ROBIN • Set to FIXED_MAPPING for advanced rail binding options	Rail Binding in Round Robin mode	• Advanced tuning can result in better performance
MV2_PROCESS_TO_RAIL_MAPPING	• Determines how HCAs will be mapped to the rails	BUNCH	• Options: SCATTER and custom list

- Refer to **Enhanced design for Multiple-Rail** section of MVAPICH2 user guide for more information
- <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3a-userguide.html#x1-700006.12>

Overview of MVAPICH2 Features

- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- **Process Mapping and Point-to-point Intra-node Protocols**
- Collectives
- MPI_T Support

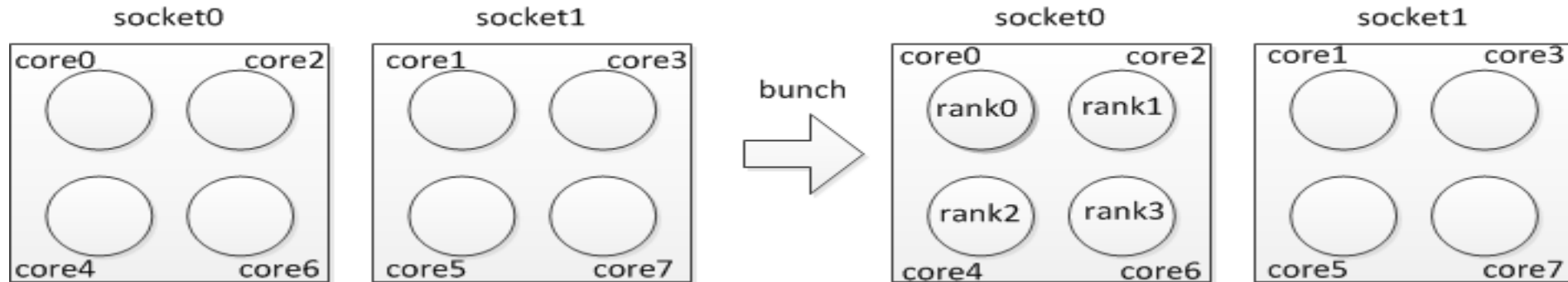
Process Mapping support in MVAPICH2



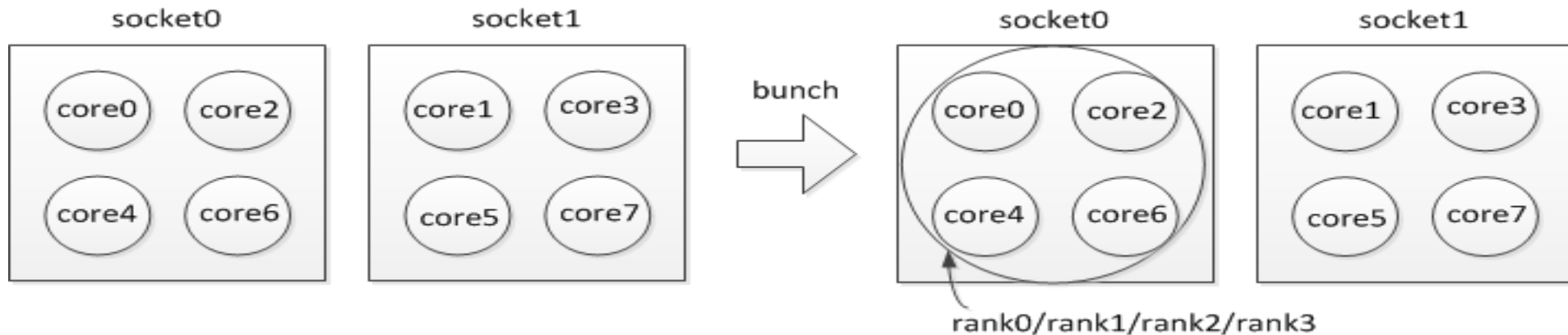
MVAPICH2 detects processor architecture at job-launch

Preset Process-binding Policies – Bunch

- “Core” level “Bunch” mapping (Default)
 - MV2_CPU_BINDING_POLICY=bunch

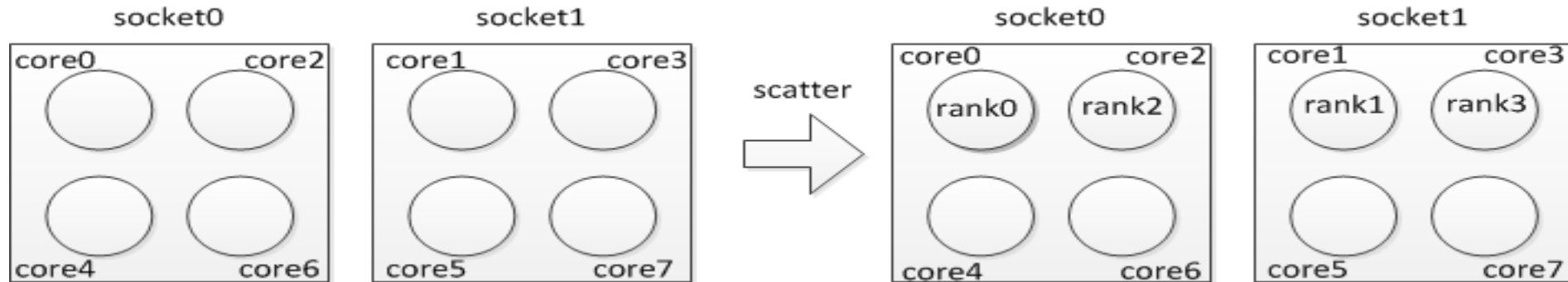


- “Socket/Numanode” level “Bunch” mapping
 - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=bunch

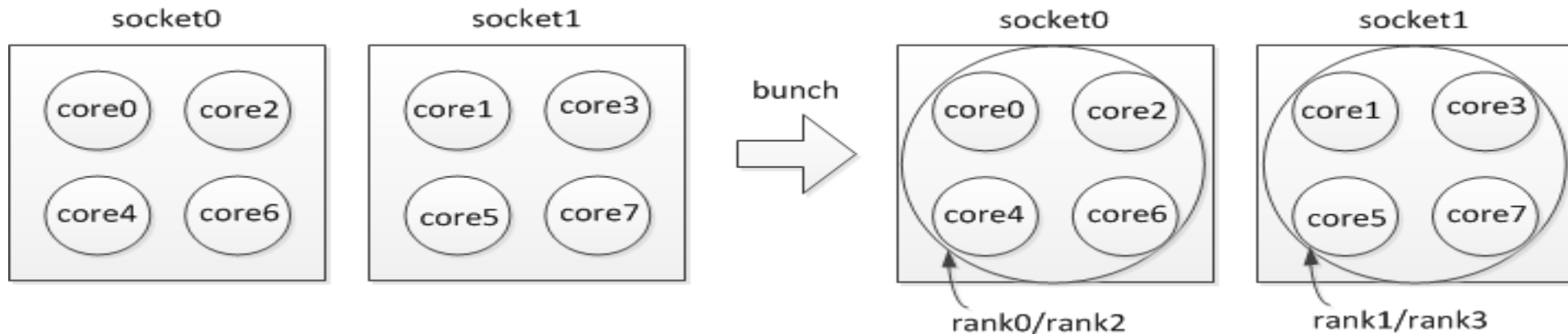


Preset Process-binding Policies – Scatter

- “Core” level “Scatter” mapping
 - MV2_CPU_BINDING_POLICY=scatter



- “Socket/Numanode” level “Scatter” mapping
 - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=scatter

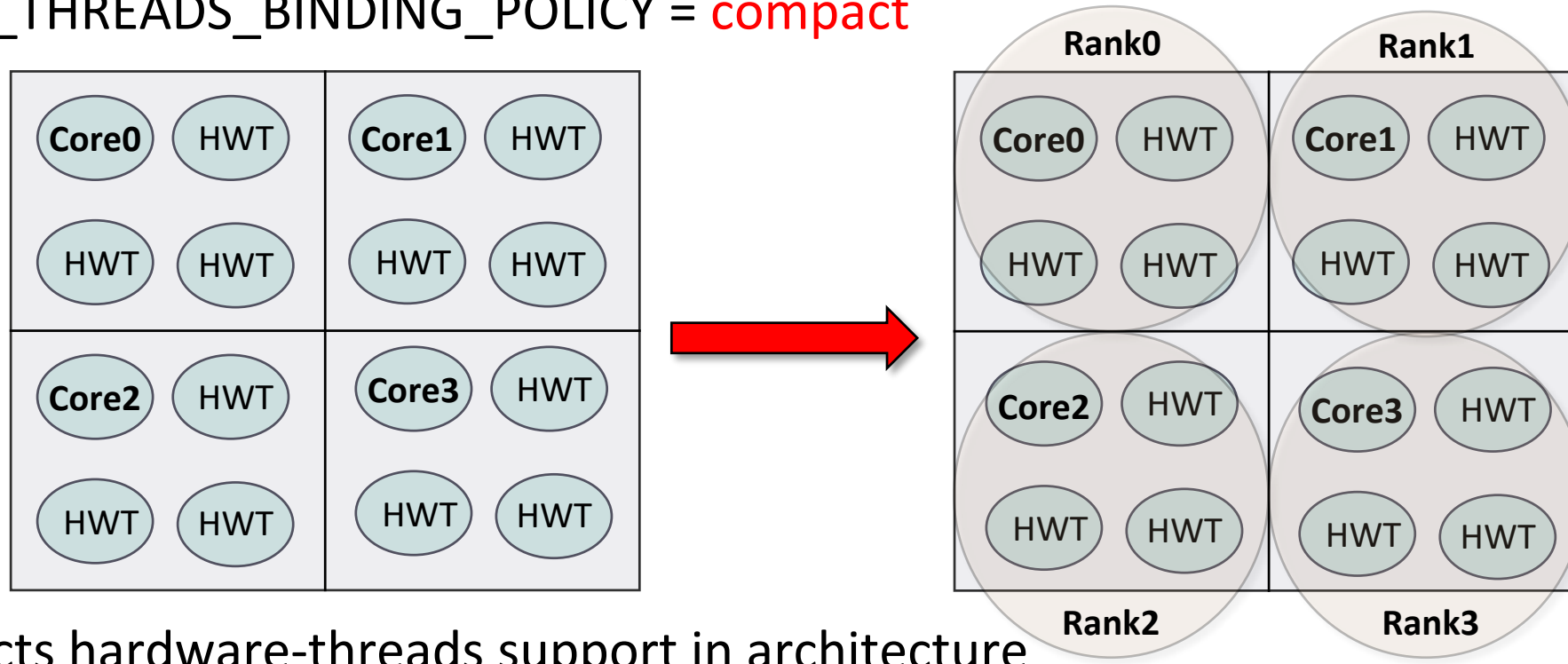


Process and thread binding policies in hybrid MPI+Threads

- A new process binding policy – “**hybrid**”
 - **MV2_CPU_BINDING_POLICY** = hybrid
- A new environment variable for co-locating Threads with MPI Processes
 - **MV2_THREADS_PER_PROCESS** = k
 - Automatically set to OMP_NUM_THREADS if OpenMP is being used
 - Provides a hint to the MPI runtime to spare resources for application threads.
- New variable for threads bindings with respect to parent process and architecture
 - **MV2_HYBRID_BINDING_POLICY** = {bunch|scatter|linear|compact|spread|numa}
 - Linear – binds MPI ranks and OpenMP threads sequentially (one after the other)
 - Recommended to be used on non-hyper threaded systems with MPI+OpenMP
 - Compact – binds MPI rank to physical-core and locates respective OpenMP threads on hardware threads
 - Recommended to be used on multi-/many-cores e.g., KNL, POWER8, and hyper-threaded Xeon, etc.

Binding Example in Hybrid (MPI+Threads)

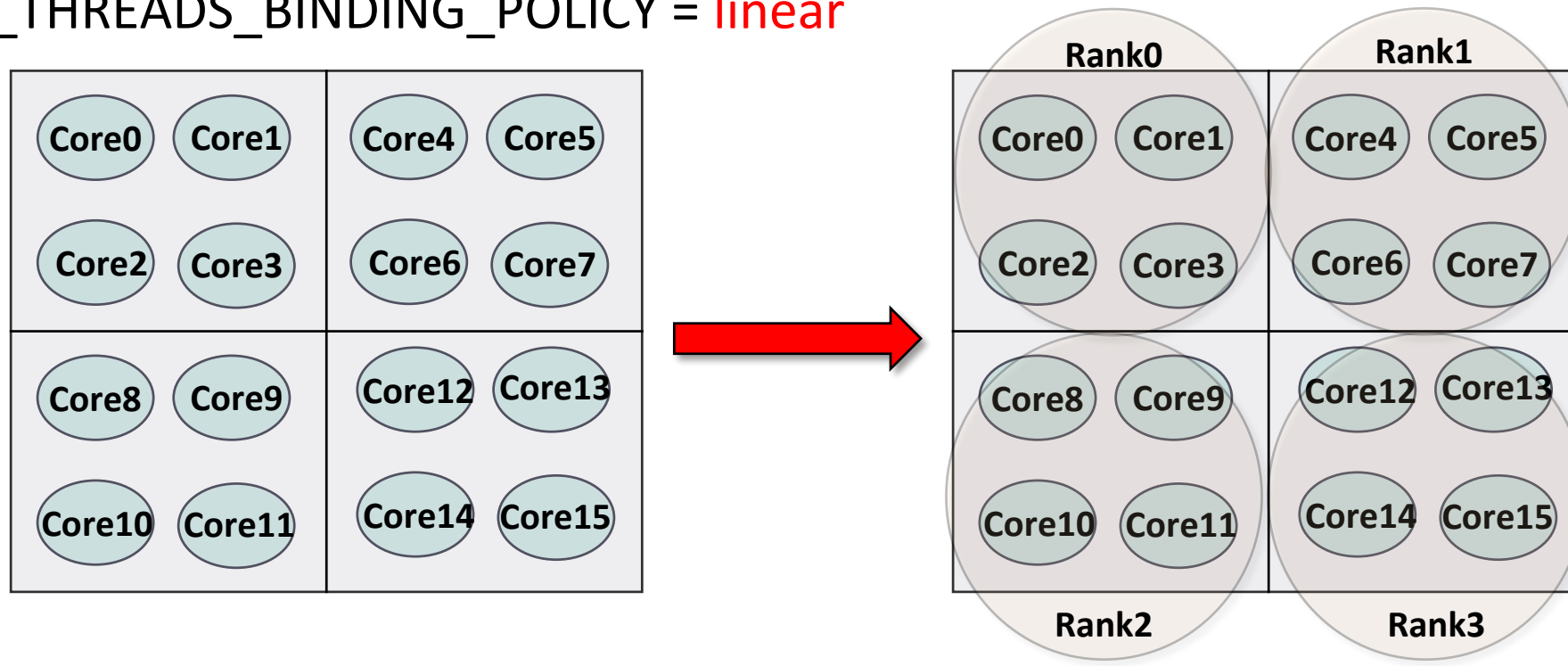
- MPI Processes = 4, OpenMP Threads per Process = 4
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_THREADS_PER_PROCESS = 4
- MV2_THREADS_BINDING_POLICY = compact



- Detects hardware-threads support in architecture
- Assigns MPI ranks to physical cores and respective OpenMP Threads to HW threads

Binding Example in Hybrid (MPI+Threads) ---- Cont'd

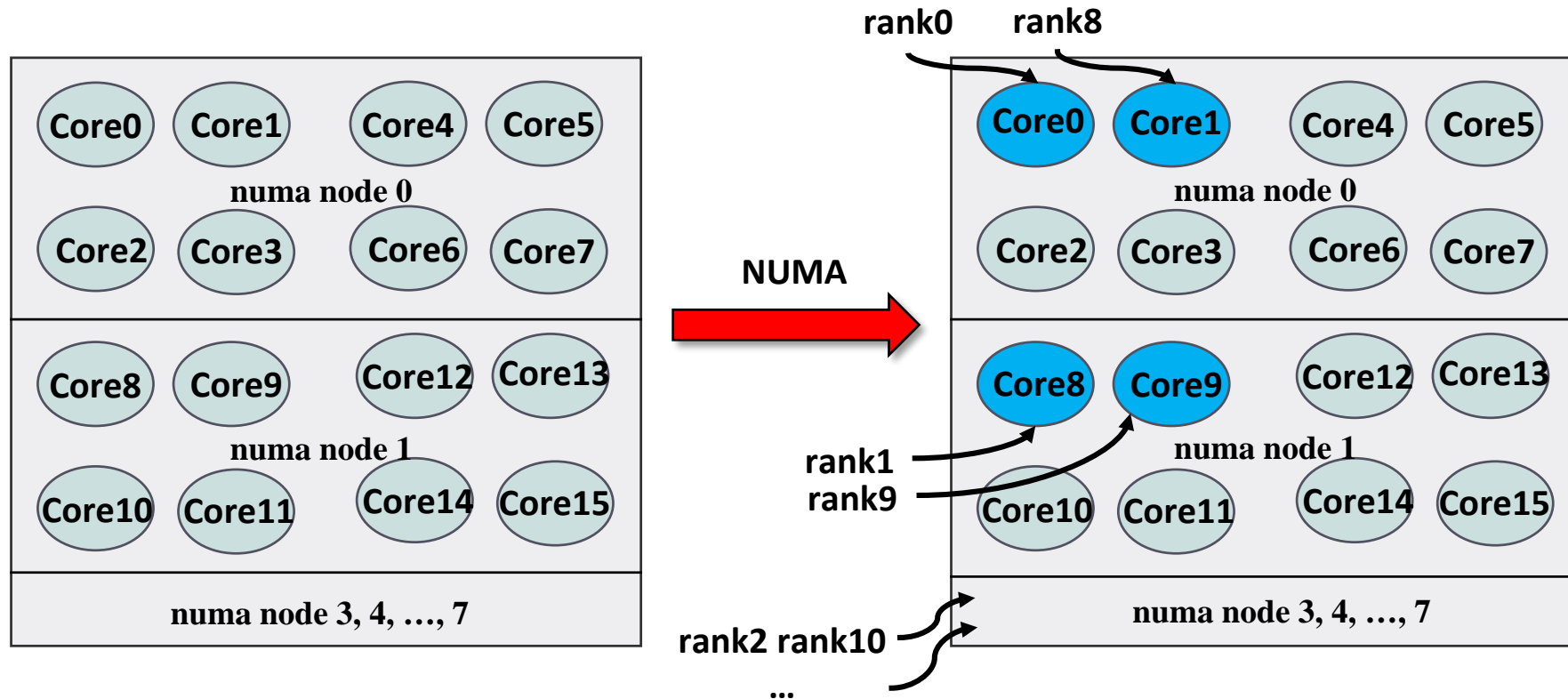
- MPI Processes = 4, OpenMP Threads per Process = 4
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_THREADS_PER_PROCESS = 4
- MV2_THREADS_BINDING_POLICY = **linear**



- MPI Rank-0 with its 4-OpenMP threads gets bound on Core-0 through Core-3, and so on

Binding Example in Hybrid (MPI+Threads) ---- Cont'd

- MPI Processes = 16
- Example: AMD EPYC 7551 processor with 8 NUMA domains
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_HYBRID_BINDING_POLICY = **numa**



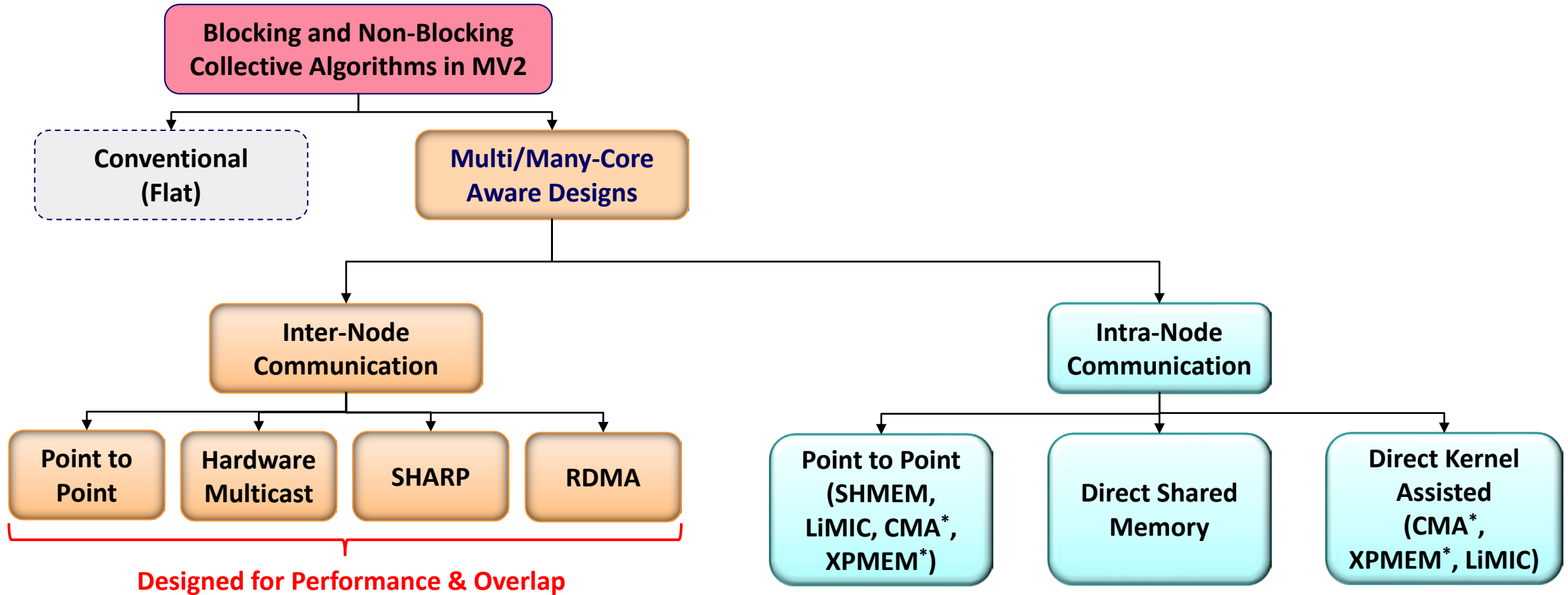
User-Defined Process Mapping

- User has complete-control over process-mapping
 - To run 4 processes on cores 0, 1, 4, 5:
 - \$ mpirun_rsh -np 4 -hostfile hosts **MV2_CPU_MAPPING=0:1:4:5** ./a.out
 - Use ',' or '-' to bind to a set of cores:
 - \$ mpirun_rsh -np 64 -hostfile hosts **MV2_CPU_MAPPING=0,2-4:1:5:6** ./a.out
 - Is process binding working as expected?
 - **MV2_SHOW_CPU_BINDING=1**
 - Display CPU binding information
 - Launcher independent
 - Example
 - MV2_SHOW_CPU_BINDING=1 MV2_CPU_BINDING_POLICY=scatter
- ```
-----CPU AFFINITY-----
RANK:0 CPU_SET: 0
RANK:1 CPU_SET: 8
```
- Refer to **Running with Efficient CPU (Core) Mapping** section of MVAPICH2 user guide for more information
  - <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3rc1-userguide.html#x1-600006.5>

# Overview of MVAPICH2 Features

- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- **Collectives**
- MPI\_T Support

# Collective Communication in MVAPICH2



Run-time flags:

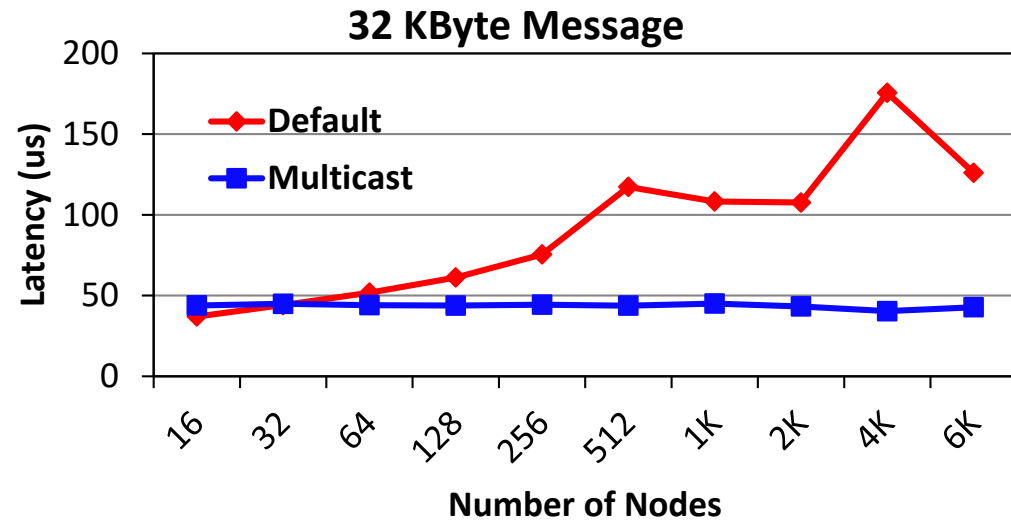
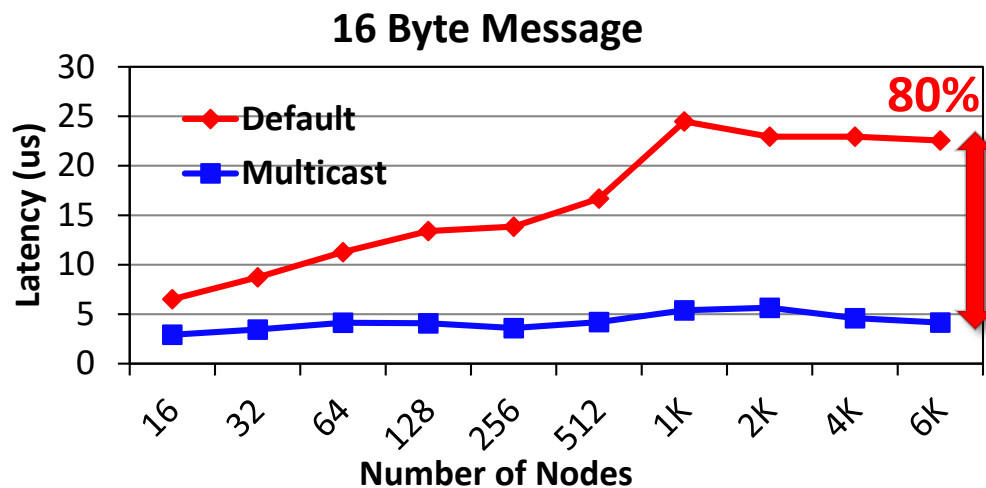
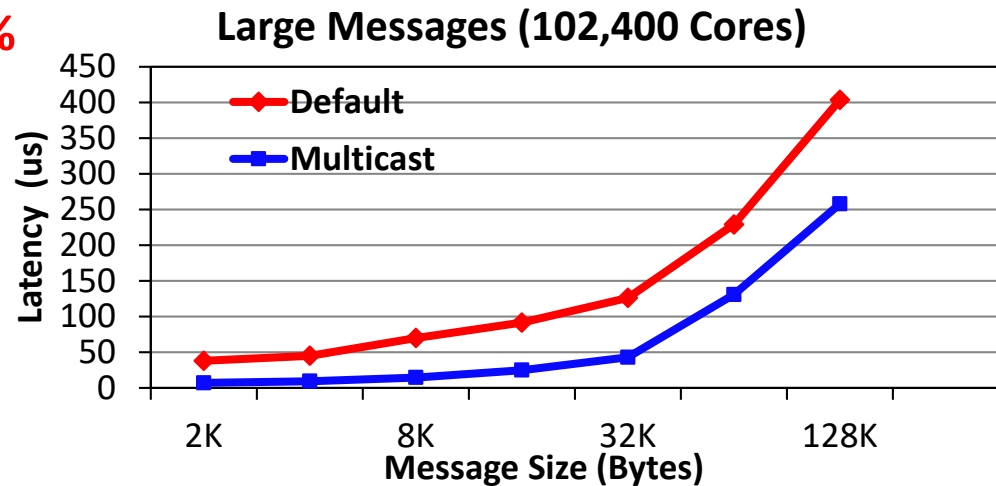
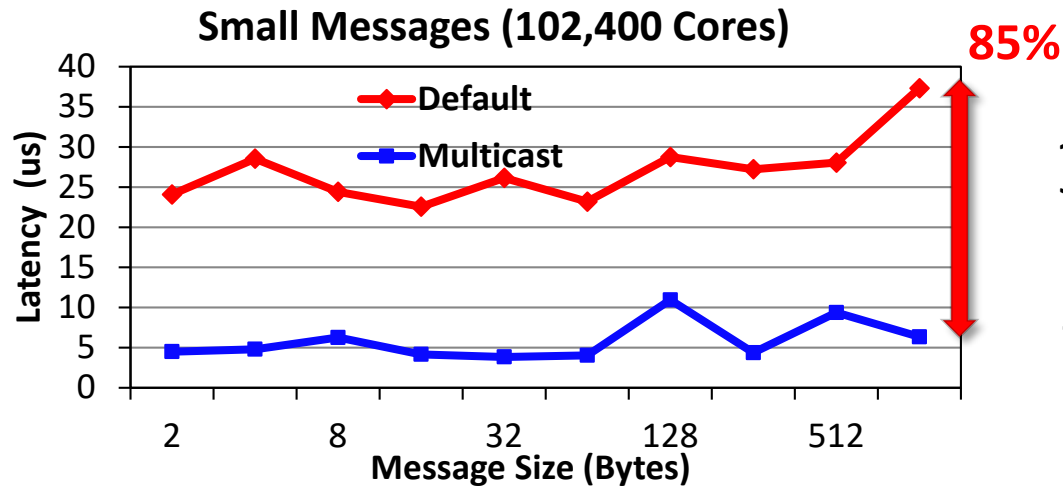
All shared-memory based collectives : MV2\_USE\_SHMEM\_COLL (Default: ON)

Hardware Mcast-based collectives : MV2\_USE\_MCAST (Default : OFF)

CMA and XPMEM-based collectives are in MVAPICH2-X

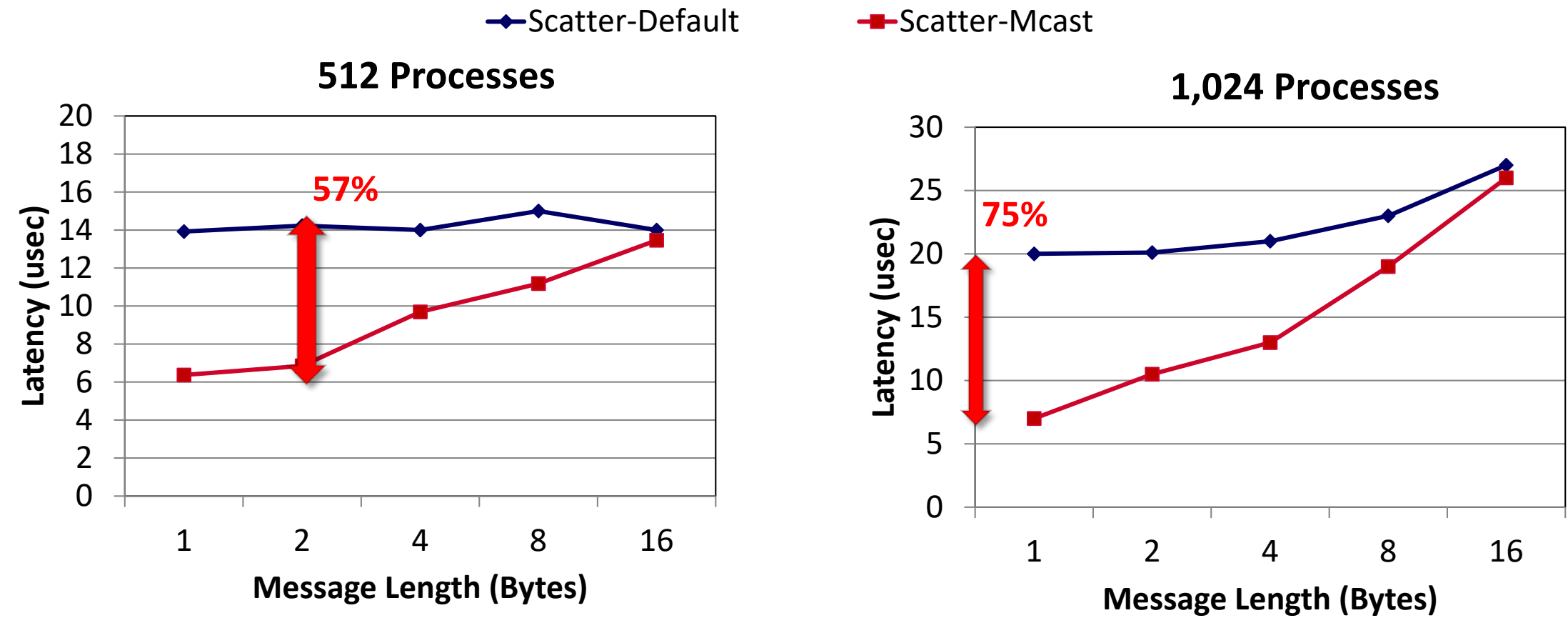


# Hardware Multicast-aware MPI\_Bcast on TACC Stampede



- MCAST-based designs improve latency of MPI\_Bcast by up to **85%**
- Use MV2\_USE\_MCAST=1 to enable MCAST-based designs

# MPI\_Scatter - Benefits of using Hardware-Mcast

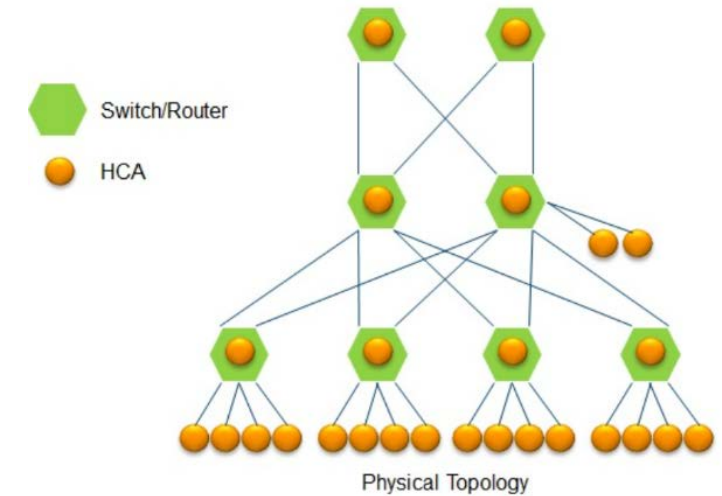


- Enabling MCAST-based designs for MPI\_Scatter improves small message up to **75%**

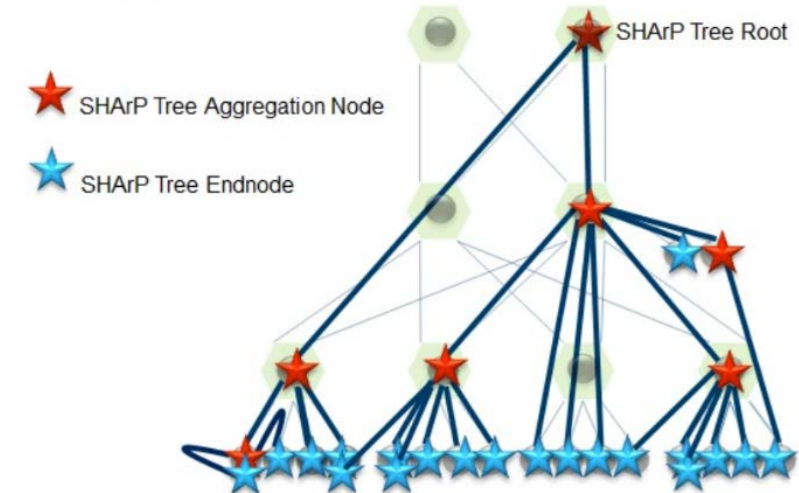
| Parameter         | Description                         | Default  |
|-------------------|-------------------------------------|----------|
| MV2_USE_MCAST = 1 | Enables hardware Multicast features | Disabled |
| --enable-mcast    | Configure flag to enable            | Enabled  |

# 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\*

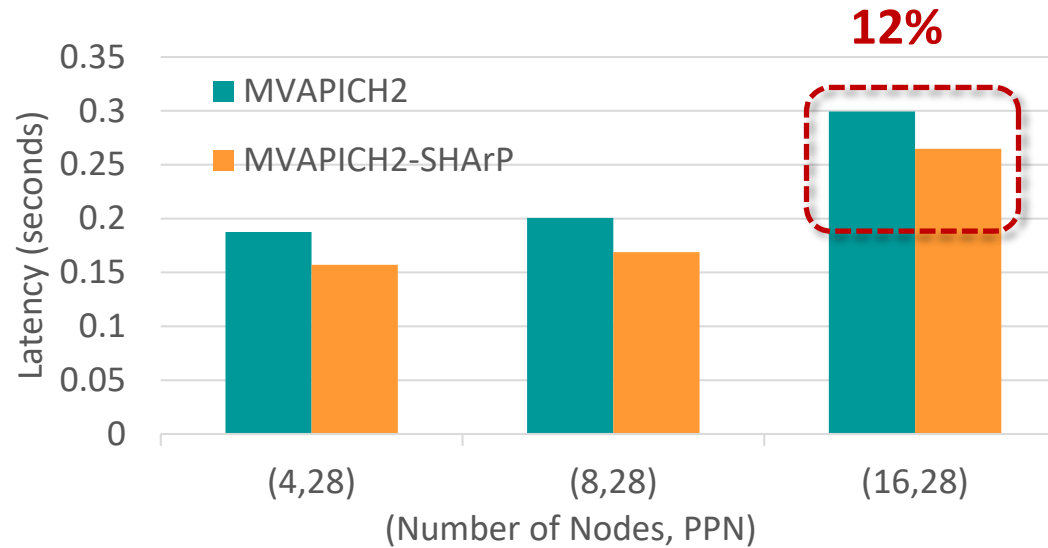


Logical SHArP Tree\*

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

# Benefits of SHARP Allreduce at Application Level

Avg DDOT Allreduce time of HPCG

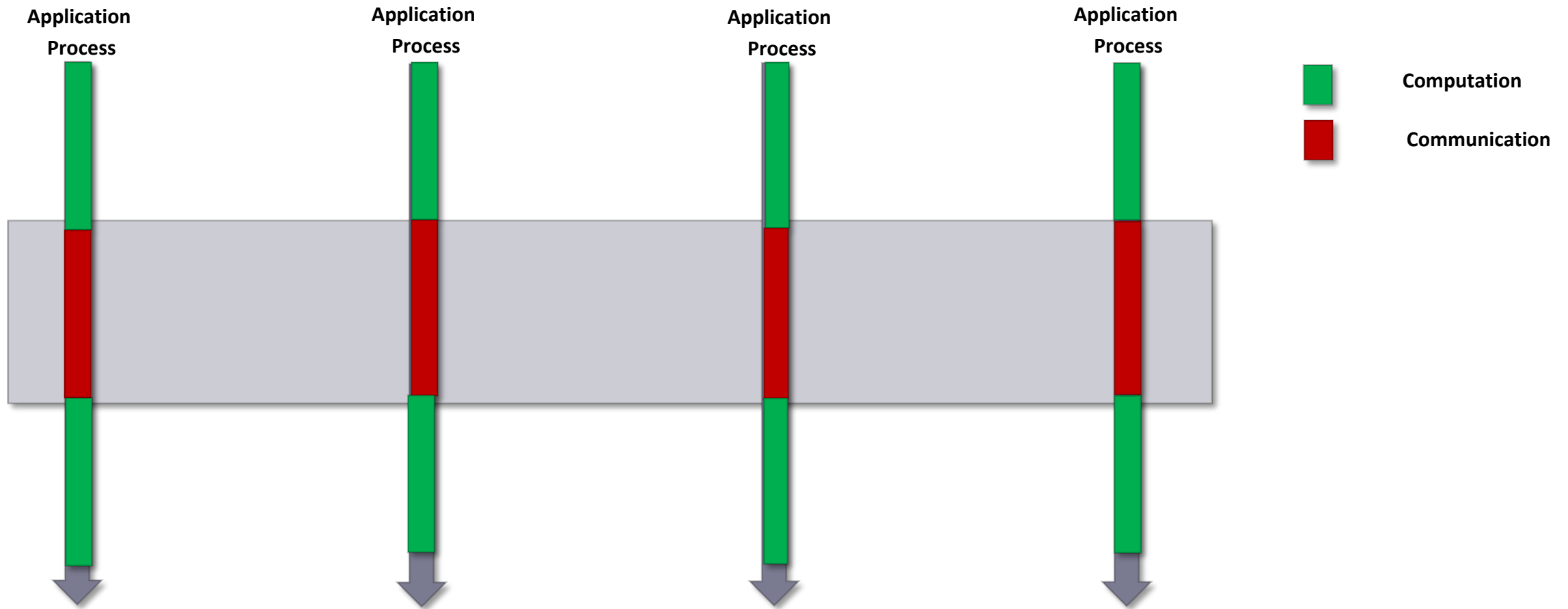


SHARP support available since MVAPICH2 2.3a

| Parameter          | Description                     | Default  |
|--------------------|---------------------------------|----------|
| MV2_ENABLE_SHARP=1 | Enables SHARP-based collectives | Disabled |
| --enable-sharp     | Configure flag to enable SHARP  | Disabled |

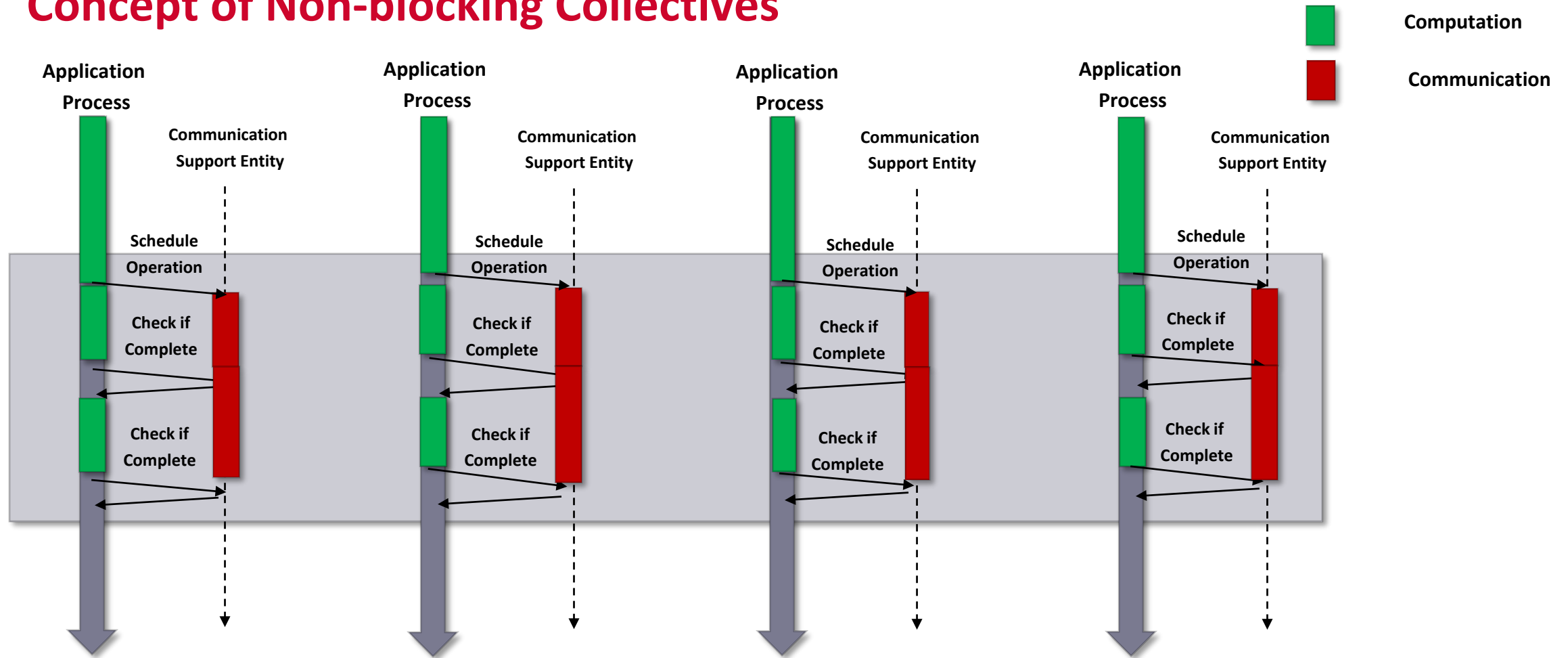
- Refer to **Running Collectives with Hardware based SHARP support** section of MVAPICH2 user guide for more information
- <http://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-2.3-userguide.html#x1-990006.26>

# Problems with Blocking Collective Operations



- Communication time cannot be used for compute
  - No overlap of computation and communication
  - Inefficient

# 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*

# Non-blocking Collective (NBC) Operations

- Enables overlap of computation with communication
- Non-blocking calls do not match blocking collective calls
  - MPI may use different algorithms for blocking and non-blocking collectives
  - Blocking collectives: Optimized for latency
  - Non-blocking collectives: Optimized for overlap
- A process calling a NBC operation
  - Schedules collective operation and immediately returns
  - Executes application computation code
  - Waits for the end of the collective
- The communication progress by
  - Application code through MPI\_Test
  - Network adapter (HCA) with hardware support
  - Dedicated processes / thread in MPI library
- There is a non-blocking equivalent for each blocking operation
  - Has an “I” in the name
    - MPI\_Bcast -> MPI\_Ibcast; MPI\_Reduce -> MPI\_Ireduce

# How do I write applications with NBC?

```
void main()
{
 MPI_Init()

 MPI_Ialltoall(...)

 Computation that does not depend on result of Alltoall

 MPI_Test(for Ialltoall) /* Check if complete (non-blocking) */

 Computation that does not depend on result of Alltoall

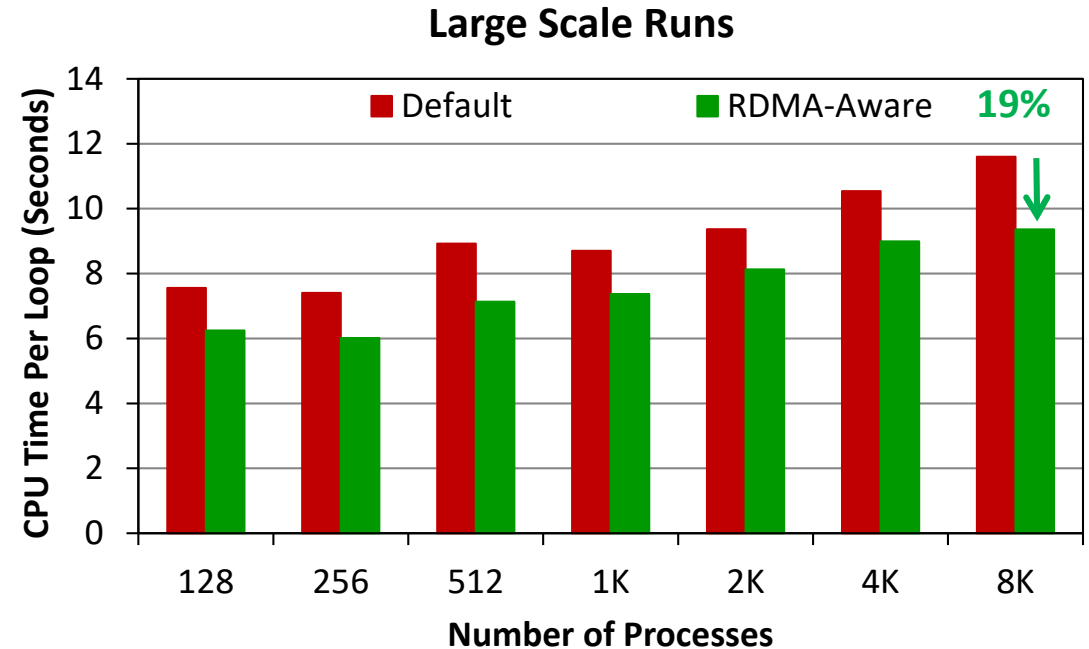
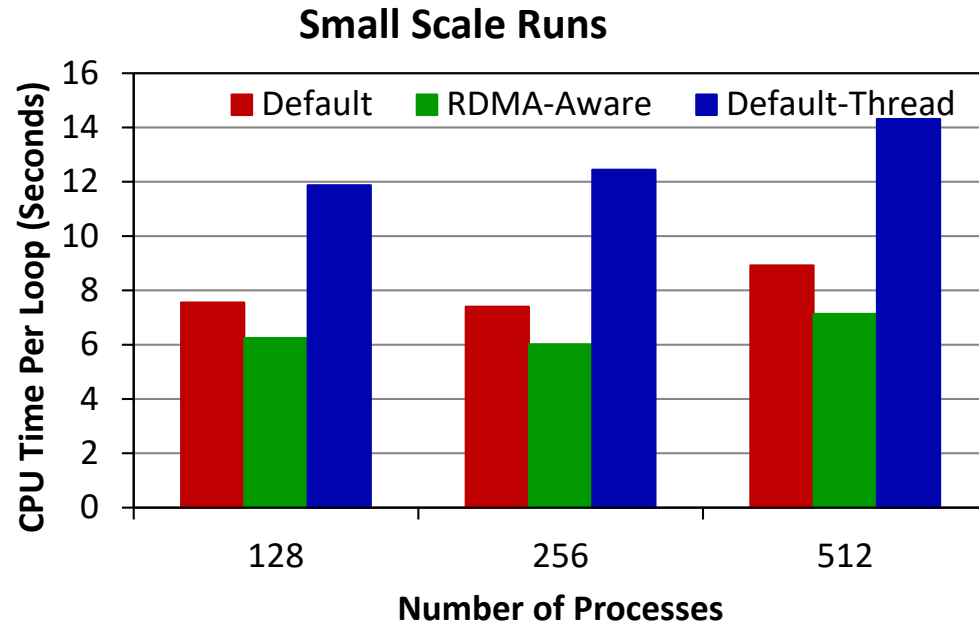
 MPI_Wait(for Ialltoall) /* Wait till complete (Blocking) */

 ...

 MPI_Finalize()
}
```



# P3DFFT Performance with Non-Blocking Alltoall using RDMA Primitives

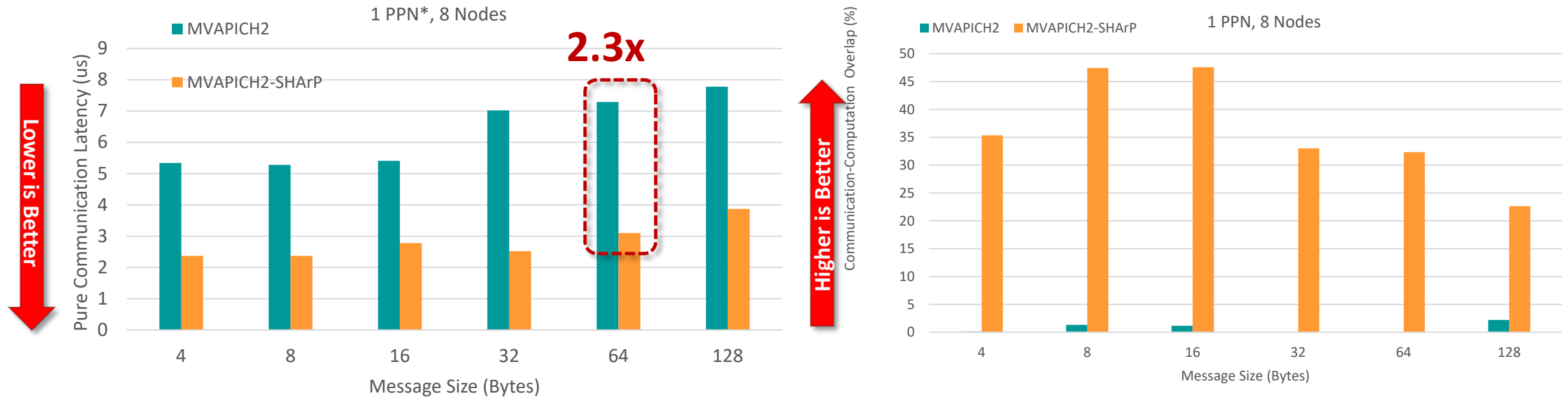


- Weak scaling experiments; problem size increases with job size
- RDMA-Aware delivers 19% improvement over Default @ 8,192 procs
- Default-Thread exhibits worst performance
  - Possibly because threads steal CPU cycles from P3DFFT
  - Do not consider for large scale experiments

**Will be available in future**

# Evaluation of SHArP based Non Blocking Allreduce

## MPI\_allreduce Benchmark



- Complete offload of Allreduce collective operation to Switch helps to have much higher overlap of communication and computation

**Available since MVAPICH2 2.3a**

\*PPN: Processes Per Node

# Presentation Overview

- Job start-up
- Point-to-point Inter-node Protocol
- Transport Type Selection
- Multi-rail
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives
- **MPI\_T Support**

# MPI Tools Information Interface (MPI\_T)

- Introduced in MPI 3.0 standard to expose internals of MPI to tools and applications
- Generalized interface – no defined variables in the standard
- Variables can differ between
  - MPI implementations
  - Compilations of same MPI library (production vs debug)
  - Executions of the same application/MPI library
  - There could be no variables provided
- Control Variables (CVARS) and Performance Variables (PVARs)
- More about the interface: [mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf](https://mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf)

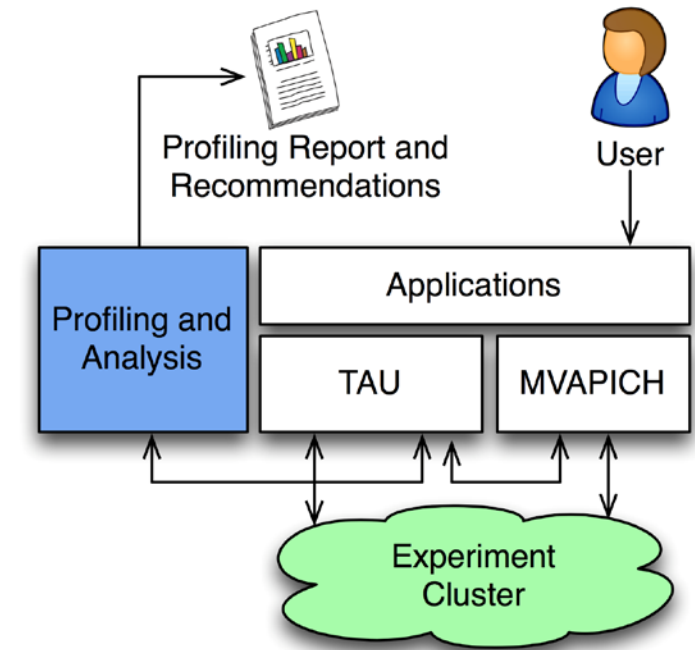
# Co-designing Applications to use MPI-T

Example Pseudo-code: Optimizing the eager limit dynamically:

```
MPI_T_init_thread(..)
MPI_T_cvar_get_info(MV2_EAGER_THRESHOLD)
if (msg_size < MV2_EAGER_THRESHOLD + 1KB)
 MPI_T_cvar_write(MV2_EAGER_THRESHOLD, +1024)
MPI_Send(..)
MPI_T_finalize(..)
```

# Performance Engineering Applications using MVAPICH2 and TAU

- Enhance existing support for MPI\_T in MVAPICH2 to expose a richer set of performance and control variables
- Get and display MPI Performance Variables (PVARs) made available by the runtime in TAU
- Control the runtime's behavior via MPI Control Variables (CVARs)
- Introduced support for new MPI\_T based CVARs to MVAPICH2
  - MPIR\_CVAR\_MAX\_INLINE\_MSG\_SZ, MPIR\_CVAR\_VBUF\_POOL\_SIZE, MPIR\_CVAR\_VBUF\_SECONDARY\_POOL\_SIZE
- TAU enhanced with support for setting MPI\_T CVARs in a non-interactive mode for uninstrumented applications
- S. Ramesh, A. Maheo, S. Shende, A. Malony, H. Subramoni, and D. K. Panda, *MPI Performance Engineering with the MPI Tool Interface: the Integration of MVAPICH and TAU*, *EuroMPI/USA '17, Best Paper Finalist*



Available in MVAPICH2

**VBUF usage without CVAR based tuning as displayed by ParaProf**

| Name                                                                   | MaxValue  | MinValue  | MeanValue | Std. Dev. | NumSamples | Total     |
|------------------------------------------------------------------------|-----------|-----------|-----------|-----------|------------|-----------|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs) | 3,313,056 | 3,313,056 | 3,313,056 | 0         | 1          | 3,313,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_available (Number of UD VBUFs available)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)                  | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_vbuf_allocated (Number of VBUFs allocated)                         | 320       | 320       | 320       | 0         | 1          | 320       |
| mv2_vbuf_available (Number of VBUFs available)                         | 255       | 255       | 255       | 0         | 1          | 255       |
| mv2_vbuf_freed (Number of VBUFs freed)                                 | 25,545    | 25,545    | 25,545    | 0         | 1          | 25,545    |
| mv2_vbuf_inuse (Number of VBUFs inuse)                                 | 65        | 65        | 65        | 0         | 1          | 65        |
| mv2_vbuf_max_use (Maximum number of VBUFs used)                        | 65        | 65        | 65        | 0         | 1          | 65        |
| num_calloc_calls (Number of MPIT_calloc calls)                         | 89        | 89        | 89        | 0         | 1          | 89        |

**VBUF usage with CVAR based tuning as displayed by ParaProf**

| Name                                                                   | MaxValue  | MinValue  | MeanValue | Std. Dev. | NumSamples | Total     |
|------------------------------------------------------------------------|-----------|-----------|-----------|-----------|------------|-----------|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs) | 1,815,056 | 1,815,056 | 1,815,056 | 0         | 1          | 1,815,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_available (Number of UD VBUFs available)                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)                  | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_vbuf_allocated (Number of VBUFs allocated)                         | 160       | 160       | 160       | 0         | 1          | 160       |
| mv2_vbuf_available (Number of VBUFs available)                         | 94        | 94        | 94        | 0         | 1          | 94        |
| mv2_vbuf_freed (Number of VBUFs freed)                                 | 5,479     | 5,479     | 5,479     | 0         | 1          | 5,479     |
| mv2_vbuf_inuse (Number of VBUFs inuse)                                 | 66        | 66        | 66        | 0         | 1          | 66        |

# Enhancing MPI\_T Support

- Introduced support for new MPI\_T based CVARs to MVAPICH2
  - `MPIR_CVAR_MAX_INLINE_MSG_SZ`
    - Controls the message size up to which “inline” transmission of data is supported by MVAPICH2
  - `MPIR_CVAR_VBUF_POOL_SIZE`
    - Controls the number of internal communication buffers (VBUFs) MVAPICH2 allocates initially. Also, `MPIR_CVAR_VBUF_POOL_REDUCED_VALUE[1] ([2...n])`
  - `MPIR_CVAR_VBUF_SECONDARY_POOL_SIZE`
    - Controls the number of VBUFs MVAPICH2 allocates when there are no more free VBUFs available
  - `MPIR_CVAR_IBA_EAGER_THRESHOLD`
    - Controls the message size where MVAPICH2 switches from eager to rendezvous protocol for large messages
- TAU enhanced with support for setting MPI\_T CVARs in a non-interactive mode for uninstrumented applications

# PVARs Exposed by MVAPICH2

TAU: ParaProf Manager

File Options Help

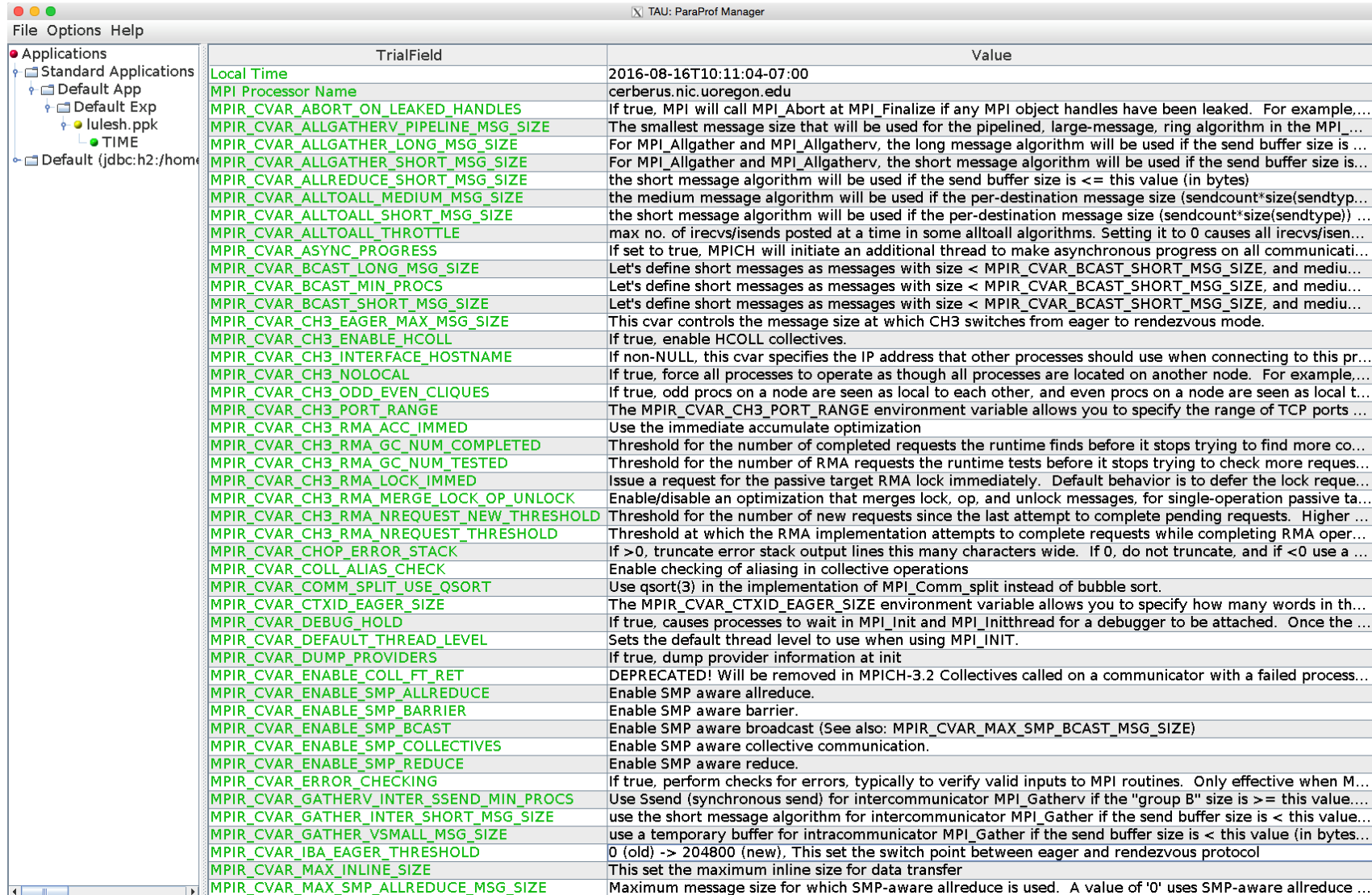
Applications

- Standard Applications
  - Default App
    - Default Exp
      - lulesh.ppk
        - TIME
  - Default (jdbc:h2:/home)

Courtesy: The TAU Team



# CVARs Exposed by MVAPICH2



The screenshot shows the TAU ParaProf Manager application window. On the left is a sidebar with a tree view containing 'Applications', 'Standard Applications', 'Default App', 'Default Exp', 'lulesh.ppk', 'TIME', and 'Default (jdbc:h2:/home)'. The main area displays a table of MPI Cvars. The table has two columns: 'TrialField' and 'Value'. The 'TrialField' column lists various MPI Cvars, and the 'Value' column shows their current values or descriptions. The Cvars listed include 'Local Time', 'MPI Processor Name', 'MPIR\_CVAR\_ABORT\_ON\_LEAKED\_HANDLES', 'MPIR\_CVAR\_ALLGATHERV\_PIPELINE\_MSG\_SIZE', 'MPIR\_CVAR\_ALLGATHER\_LONG\_MSG\_SIZE', 'MPIR\_CVAR\_ALLGATHER\_SHORT\_MSG\_SIZE', 'MPIR\_CVAR\_ALLREDUCE\_SHORT\_MSG\_SIZE', 'MPIR\_CVAR\_ALLTOALL\_MEDIUM\_MSG\_SIZE', 'MPIR\_CVAR\_ALLTOALL\_SHORT\_MSG\_SIZE', 'MPIR\_CVAR\_ALLTOALL\_THROTTLE', 'MPIR\_CVAR\_ASYNC\_PROGRESS', 'MPIR\_CVAR\_BCAST\_LONG\_MSG\_SIZE', 'MPIR\_CVAR\_BCAST\_MIN\_PROCS', 'MPIR\_CVAR\_BCAST\_SHORT\_MSG\_SIZE', 'MPIR\_CVAR\_CH3\_EAGER\_MAX\_MSG\_SIZE', 'MPIR\_CVAR\_CH3\_ENABLE\_HCOLL', 'MPIR\_CVAR\_CH3\_INTERFACE\_HOSTNAME', 'MPIR\_CVAR\_CH3\_NOLOCAL', 'MPIR\_CVAR\_CH3\_ODD\_EVEN\_CLIQUES', 'MPIR\_CVAR\_CH3\_PORT\_RANGE', 'MPIR\_CVAR\_CH3\_RMA\_ACC\_IMMED', 'MPIR\_CVAR\_CH3\_RMA\_GC\_NUM\_COMPLETED', 'MPIR\_CVAR\_CH3\_RMA\_GC\_NUM\_TESTED', 'MPIR\_CVAR\_CH3\_RMA\_LOCK\_IMMED', 'MPIR\_CVAR\_CH3\_RMA\_MERGE\_LOCK\_OP\_UNLOCK', 'MPIR\_CVAR\_CH3\_RMA\_NREQUEST\_NEW\_THRESHOLD', 'MPIR\_CVAR\_CH3\_RMA\_NREQUEST\_THRESHOLD', 'MPIR\_CVAR\_CHOP\_ERROR\_STACK', 'MPIR\_CVAR\_COLL\_ALIAS\_CHECK', 'MPIR\_CVAR\_COMM\_SPLIT\_USE\_QSORT', 'MPIR\_CVAR\_CTXID\_EAGER\_SIZE', 'MPIR\_CVAR\_DEBUG\_HOLD', 'MPIR\_CVAR\_DEFAULT\_THREAD\_LEVEL', 'MPIR\_CVAR\_DUMP\_PROVIDERS', 'MPIR\_CVAR\_ENABLE\_COLL\_FT\_RET', 'MPIR\_CVAR\_ENABLE\_SMP\_ALLREDUCE', 'MPIR\_CVAR\_ENABLE\_SMP\_BARRIER', 'MPIR\_CVAR\_ENABLE\_SMP\_BCAST', 'MPIR\_CVAR\_ENABLE\_SMP\_COLLECTIVES', 'MPIR\_CVAR\_ENABLE\_SMP\_REDUCE', 'MPIR\_CVAR\_ERROR\_CHECKING', 'MPIR\_CVAR\_GATHERV\_INTER\_SSEND\_MIN\_PROCS', 'MPIR\_CVAR\_GATHER\_INTER\_SHORT\_MSG\_SIZE', 'MPIR\_CVAR\_GATHER\_VSMALL\_MSG\_SIZE', 'MPIR\_CVAR\_IBA\_EAGER\_THRESHOLD', 'MPIR\_CVAR\_MAX\_INLINE\_SIZE', and 'MPIR\_CVAR\_MAX\_SMP\_ALLREDUCE\_MSG\_SIZE'.

| TrialField                               | Value                                                                                                           |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| Local Time                               | 2016-08-16T10:11:04-07:00                                                                                       |
| MPI Processor Name                       | cerberus.nic.uoregon.edu                                                                                        |
| MPIR_CVAR_ABORT_ON_LEAKED_HANDLES        | If true, MPI will call MPI_Abort at MPI_Finalize if any MPI object handles have been leaked. For example,...    |
| MPIR_CVAR_ALLGATHERV_PIPELINE_MSG_SIZE   | The smallest message size that will be used for the pipelined, large-message, ring algorithm in the MPI_...     |
| MPIR_CVAR_ALLGATHER_LONG_MSG_SIZE        | For MPI_Allgather and MPI_Allgatherv, the long message algorithm will be used if the send buffer size is ...    |
| MPIR_CVAR_ALLGATHER_SHORT_MSG_SIZE       | For MPI_Allgather and MPI_Allgatherv, the short message algorithm will be used if the send buffer size is...    |
| MPIR_CVAR_ALLREDUCE_SHORT_MSG_SIZE       | the short message algorithm will be used if the send buffer size is <= this value (in bytes)                    |
| MPIR_CVAR_ALLTOALL_MEDIUM_MSG_SIZE       | the medium message algorithm will be used if the per-destination message size (sendcount*size(sendtyp...        |
| MPIR_CVAR_ALLTOALL_SHORT_MSG_SIZE        | the short message algorithm will be used if the per-destination message size (sendcount*size(sendtype)) ...     |
| MPIR_CVAR_ALLTOALL_THROTTLE              | max no. of irecv/isends posted at a time in some alltoall algorithms. Setting it to 0 causes all irecvs/isen... |
| MPIR_CVAR_ASYNC_PROGRESS                 | If set to true, MPICH will initiate an additional thread to make asynchronous progress on all communicati...    |
| MPIR_CVAR_BCAST_LONG_MSG_SIZE            | Let's define short messages as messages with size < MPIR_CVAR_BCAST_SHORT_MSG_SIZE, and mediu...                |
| MPIR_CVAR_BCAST_MIN_PROCS                | Let's define short messages as messages with size < MPIR_CVAR_BCAST_SHORT_MSG_SIZE, and mediu...                |
| MPIR_CVAR_BCAST_SHORT_MSG_SIZE           | Let's define short messages as messages with size < MPIR_CVAR_BCAST_SHORT_MSG_SIZE, and mediu...                |
| MPIR_CVAR_CH3_EAGER_MAX_MSG_SIZE         | This cvar controls the message size at which CH3 switches from eager to rendezvous mode.                        |
| MPIR_CVAR_CH3_ENABLE_HCOLL               | If true, enable HCOLL collectives.                                                                              |
| MPIR_CVAR_CH3_INTERFACE_HOSTNAME         | If non-NULL, this cvar specifies the IP address that other processes should use when connecting to this pr...   |
| MPIR_CVAR_CH3_NOLOCAL                    | If true, force all processes to operate as though all processes are located on another node. For example,...    |
| MPIR_CVAR_CH3_ODD_EVEN_CLIQUES           | If true, odd procs on a node are seen as local to each other, and even procs on a node are seen as local t...   |
| MPIR_CVAR_CH3_PORT_RANGE                 | The MPIR_CVAR_CH3_PORT_RANGE environment variable allows you to specify the range of TCP ports ...              |
| MPIR_CVAR_CH3_RMA_ACC_IMMED              | Use the immediate accumulate optimization                                                                       |
| MPIR_CVAR_CH3_RMA_GC_NUM_COMPLETED       | Threshold for the number of completed requests the runtime finds before it stops trying to find more co...      |
| MPIR_CVAR_CH3_RMA_GC_NUM_TESTED          | Threshold for the number of RMA requests the runtime tests before it stops trying to check more reques...       |
| MPIR_CVAR_CH3_RMA_LOCK_IMMED             | Issue a request for the passive target RMA lock immediately. Default behavior is to defer the lock reque...     |
| MPIR_CVAR_CH3_RMA_MERGE_LOCK_OP_UNLOCK   | Enable/disable an optimization that merges lock, op, and unlock messages, for single-operation passive ta...    |
| MPIR_CVAR_CH3_RMA_NREQUEST_NEW_THRESHOLD | Threshold for the number of new requests since the last attempt to complete pending requests. Higher ...        |
| MPIR_CVAR_CH3_RMA_NREQUEST_THRESHOLD     | Threshold at which the RMA implementation attempts to complete requests while completing RMA oper...            |
| MPIR_CVAR_CHOP_ERROR_STACK               | If >0, truncate error stack output lines this many characters wide. If 0, do not truncate, and if <0 use a ...  |
| MPIR_CVAR_COLL_ALIAS_CHECK               | Enable checking of aliasing in collective operations                                                            |
| MPIR_CVAR_COMM_SPLIT_USE_QSORT           | Use qsort(3) in the implementation of MPI_Comm_split instead of bubble sort.                                    |
| MPIR_CVAR_CTXID_EAGER_SIZE               | The MPIR_CVAR_CTXID_EAGER_SIZE environment variable allows you to specify how many words in th...               |
| MPIR_CVAR_DEBUG_HOLD                     | If true, causes processes to wait in MPI_Init and MPI_Initthread for a debugger to be attached. Once the ...    |
| MPIR_CVAR_DEFAULT_THREAD_LEVEL           | Sets the default thread level to use when using MPI_INIT.                                                       |
| MPIR_CVAR_DUMP_PROVIDERS                 | If true, dump provider information at init                                                                      |
| MPIR_CVAR_ENABLE_COLL_FT_RET             | DEPRECATED! Will be removed in MPICH-3.2 Collectives called on a communicator with a failed process...          |
| MPIR_CVAR_ENABLE_SMP_ALLREDUCE           | Enable SMP aware allreduce.                                                                                     |
| MPIR_CVAR_ENABLE_SMP_BARRIER             | Enable SMP aware barrier.                                                                                       |
| MPIR_CVAR_ENABLE_SMP_BCAST               | Enable SMP aware broadcast (See also: MPIR_CVAR_MAX_SMP_BCAST_MSG_SIZE)                                         |
| MPIR_CVAR_ENABLE_SMP_COLLECTIVES         | Enable SMP aware collective communication.                                                                      |
| MPIR_CVAR_ENABLE_SMP_REDUCE              | Enable SMP aware reduce.                                                                                        |
| MPIR_CVAR_ERROR_CHECKING                 | If true, perform checks for errors, typically to verify valid inputs to MPI routines. Only effective when M...  |
| MPIR_CVAR_GATHERV_INTER_SSEND_MIN_PROCS  | Use Ssend (synchronous send) for intercommunicator MPI_Gatherv if the "group B" size is >= this value....       |
| MPIR_CVAR_GATHER_INTER_SHORT_MSG_SIZE    | use the short message algorithm for intercommunicator MPI_Gather if the send buffer size is < this value...     |
| MPIR_CVAR_GATHER_VSMALL_MSG_SIZE         | use a temporary buffer for intracommunicator MPI_Gather if the send buffer size is < this value (in bytes...    |
| MPIR_CVAR_IBA_EAGER_THRESHOLD            | 0 (old) -> 204800 (new), This set the switch point between eager and rendezvous protocol                        |
| MPIR_CVAR_MAX_INLINE_SIZE                | This set the maximum inline size for data transfer                                                              |
| MPIR_CVAR_MAX_SMP_ALLREDUCE_MSG_SIZE     | Maximum message size for which SMP-aware allreduce is used. A value of '0' uses SMP-aware allreduce ...         |

Courtesy: The TAU Team


# Using MVAPICH2 and TAU

- To set CVARs or read PVARs using TAU for an uninstrumented binary:  
% export TAU\_TRACK\_MPI\_T\_PVARS=1  
% export TAU\_MPI\_T\_CVAR\_METRICS=  
    MPIR\_CVAR\_VBUF\_POOL\_REDUCED\_VALUE[1],  
    MPIR\_CVAR\_IBA\_EAGER\_THRESHOLD  
% export TAU\_MPI\_T\_CVAR\_VALUES=32,64000  
% export PATH=/path/to/tau/x86\_64/bin:\$PATH  
% mpirun -np 1024 *tau\_exec -T mvapich2,mpit* ./a.out  
% paraprof

Courtesy: The TAU Team

# VBUF usage without CVARs

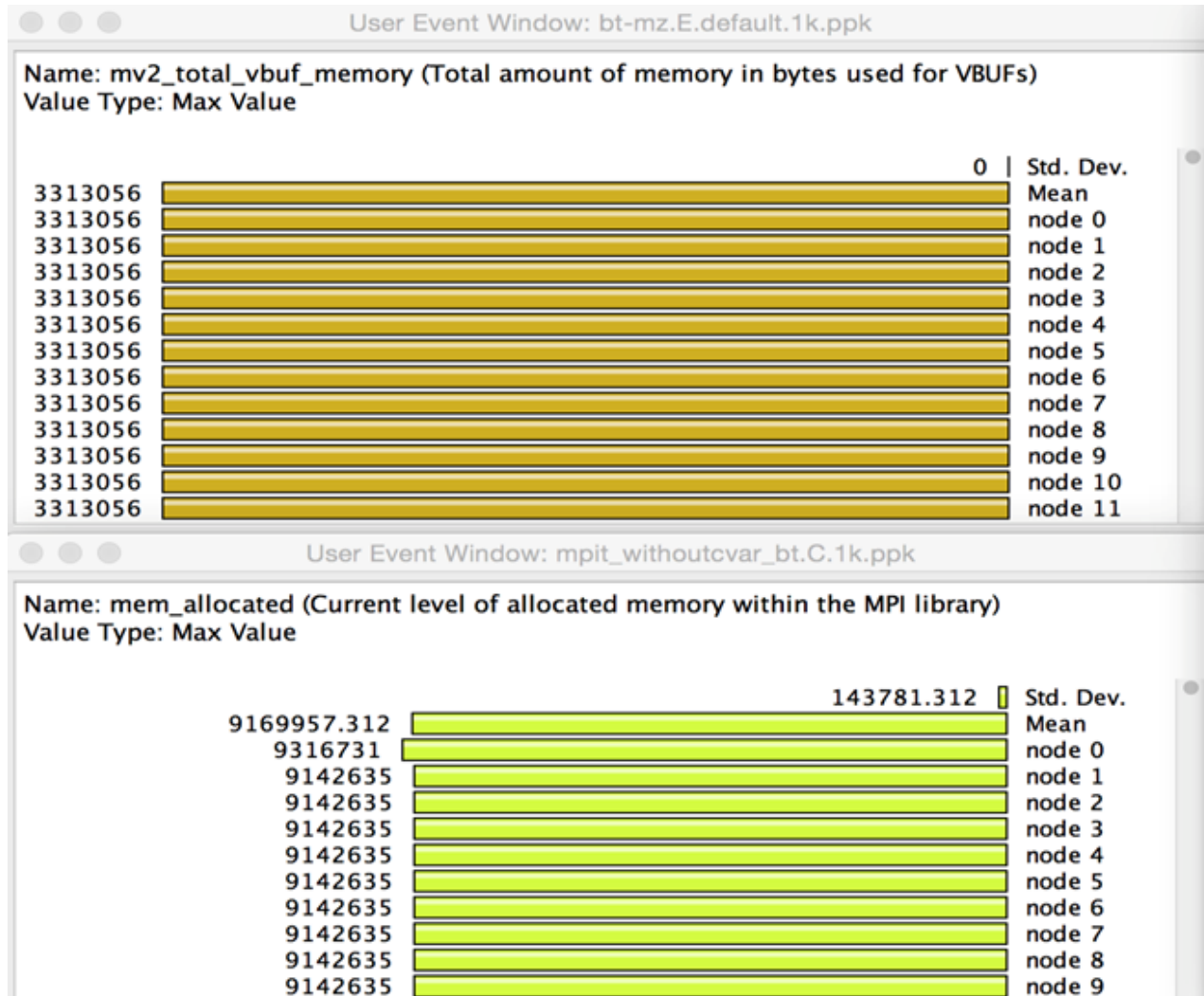
TAU: ParaProf: Context Events for: node 0 - mpit\_withoutcvar\_bt.C.1k.ppk

| Name  | MaxValue  | MinValue  | MeanValue | Std. Dev. | NumSamples | Total     |
|----------------------------------------------------------------------------------------|-----------|-----------|-----------|-----------|------------|-----------|
| mv2_total_vbuf_memory (Total amount of memory in bytes used for VBUFs)                 | 3,313,056 | 3,313,056 | 3,313,056 | 0         | 1          | 3,313,056 |
| mv2_ud_vbuf_allocated (Number of UD VBUFs allocated)                                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_available (Number of UD VBUFs available)                                   | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_freed (Number of UD VBUFs freed)                                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_inuse (Number of UD VBUFs inuse)                                           | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_ud_vbuf_max_use (Maximum number of UD VBUFs used)                                  | 0         | 0         | 0         | 0         | 0          | 0         |
| mv2_vbuf_allocated (Number of VBUFs allocated)                                         | 320       | 320       | 320       | 0         | 1          | 320       |
| mv2_vbuf_available (Number of VBUFs available)                                         | 255       | 255       | 255       | 0         | 1          | 255       |
| mv2_vbuf_freed (Number of VBUFs freed)                                                 | 25,545    | 25,545    | 25,545    | 0         | 1          | 25,545    |
| mv2_vbuf_inuse (Number of VBUFs inuse)                                                 | 65        | 65        | 65        | 0         | 1          | 65        |
| mv2_vbuf_max_use (Maximum number of VBUFs used)                                        | 65        | 65        | 65        | 0         | 1          | 65        |
| num_calloc_calls (Number of MPIT_calloc calls)                                         | 89        | 89        | 89        | 0         | 1          | 89        |
| num_free_calls (Number of MPIT_free calls)                                             | 47,801    | 47,801    | 47,801    | 0         | 1          | 47,801    |
| num_malloc_calls (Number of MPIT_malloc calls)                                         | 49,258    | 49,258    | 49,258    | 0         | 1          | 49,258    |
| num_memalign_calls (Number of MPIT_memalign calls)                                     | 34        | 34        | 34        | 0         | 1          | 34        |
| num_memalign_free_calls (Number of MPIT_memalign_free calls)                           | 0         | 0         | 0         | 0         | 0          | 0         |

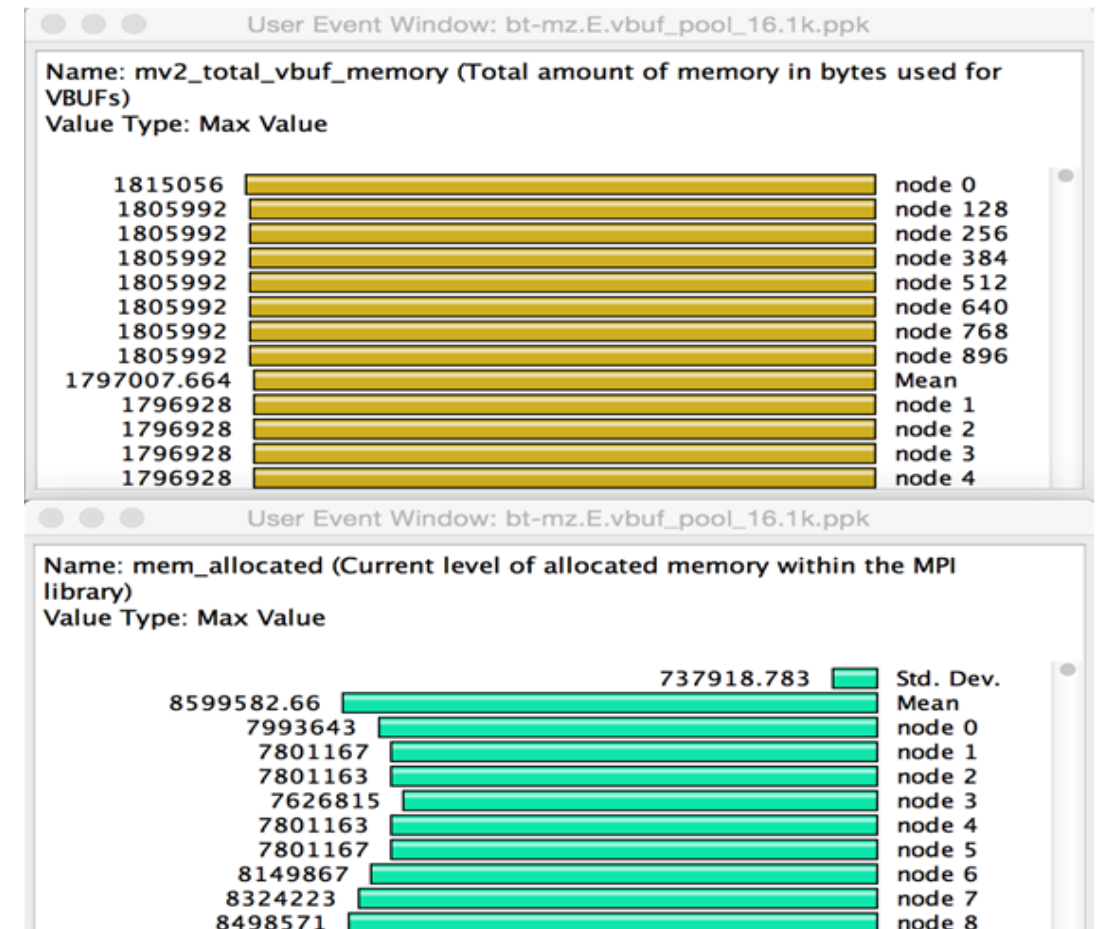
Courtesy: The TAU Team

# VBUF Memory Usage Without and With CVAR

## Without CVAR



## With CVAR



```
% export TAU_TRACK_MPI_T_PVARS=1
% export TAU_MPI_T_CVAR_METRICS=MPIR_CVAR_VBUF_POOL_SIZE
% export TAU_MPI_T_CVAR_VALUES=16
% mpirun -np 1024 tau_exec -T mvapich2 ./a.out
```

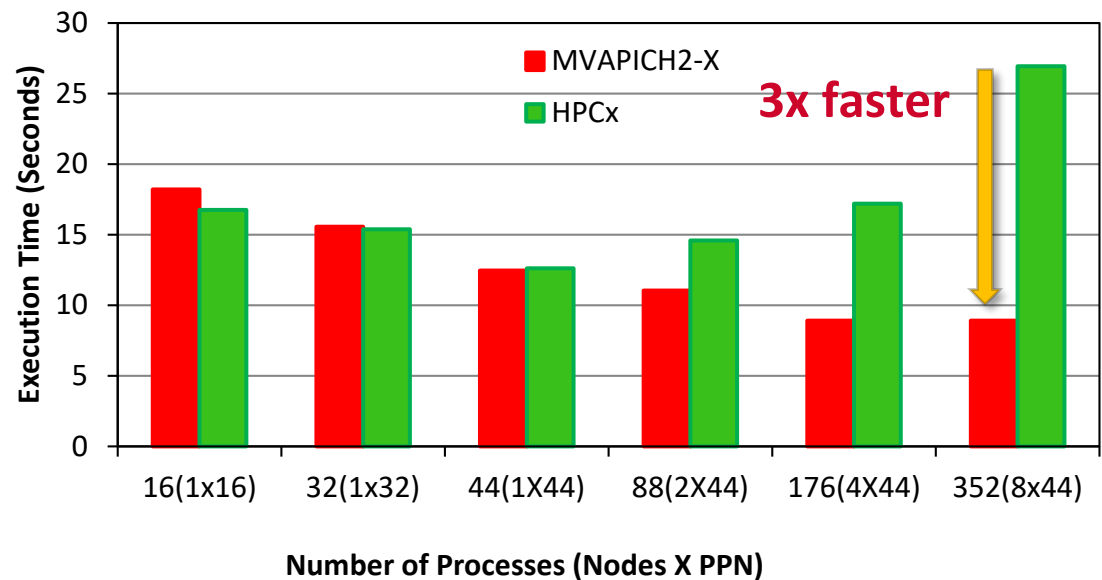
Courtesy: The TAU Team

# MVAPICH2-Azure 2.3.2

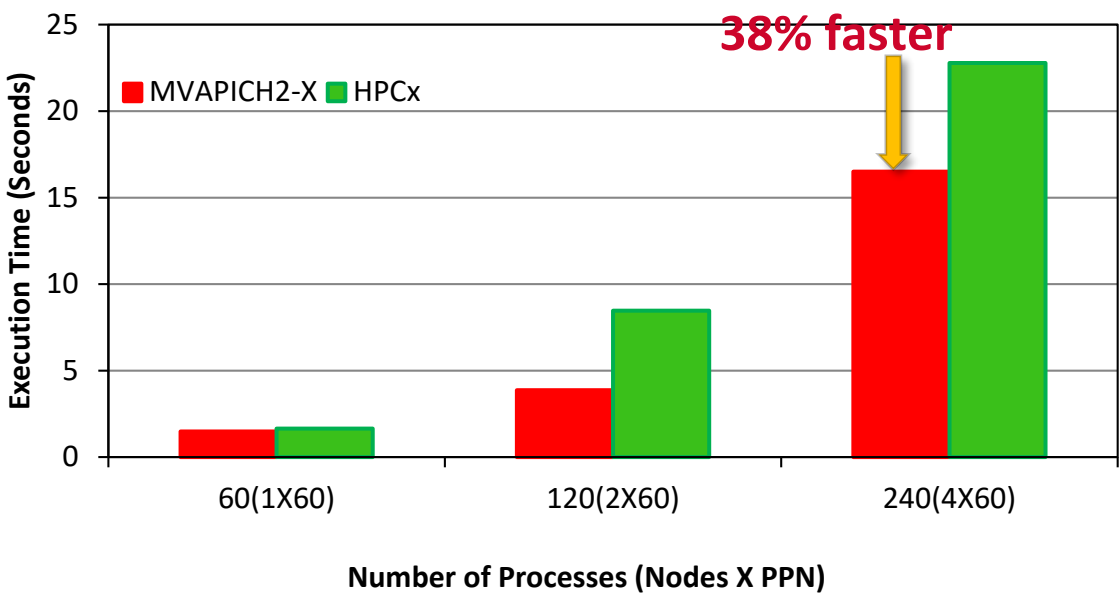
- Released on 08/16/2019
- Major Features and Enhancements
  - Based on MVAPICH2-2.3.2
  - Enhanced tuning for point-to-point and collective operations
  - Targeted for Azure HB & HC virtual machine instances
  - Flexibility for 'one-click' deployment
  - Tested with Azure HB & HC VM instances

# Performance of Radix

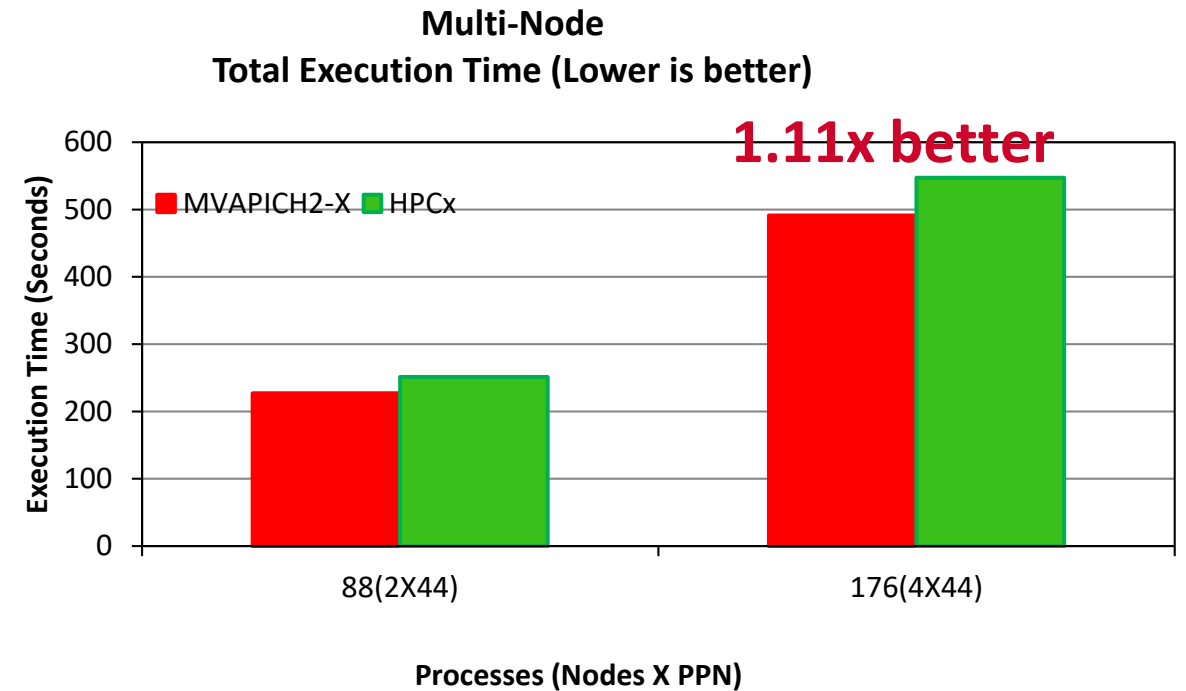
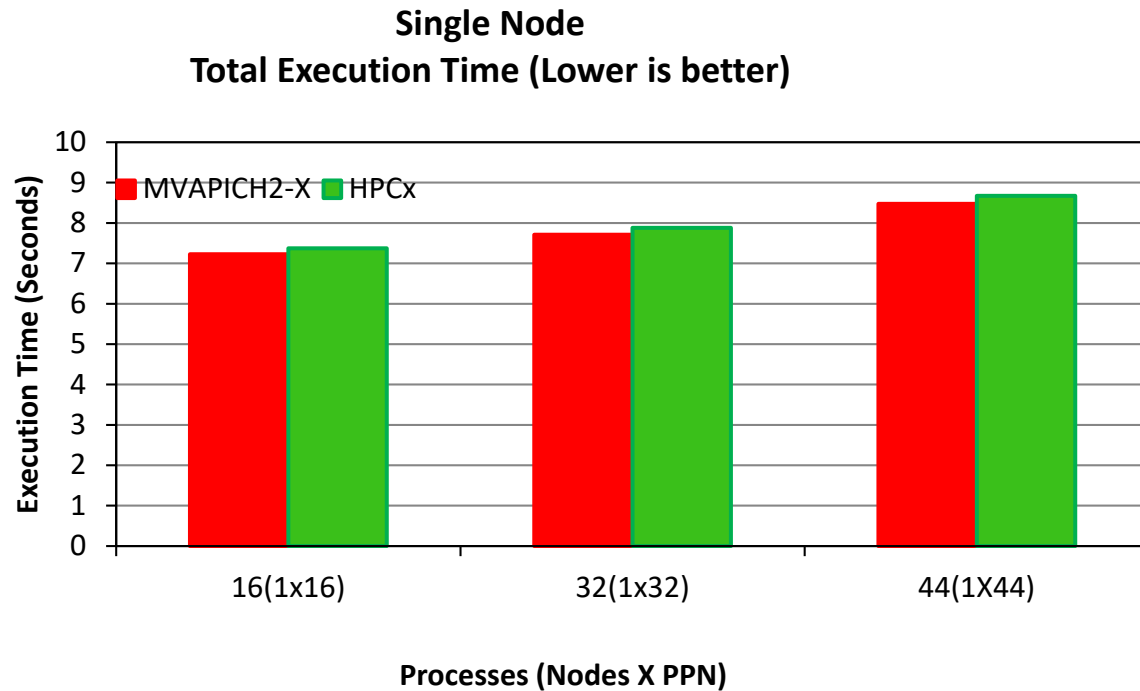
Total Execution Time on HC (Lower is better)



Total Execution Time on HB (Lower is better)



# Performance of FDS (HC)



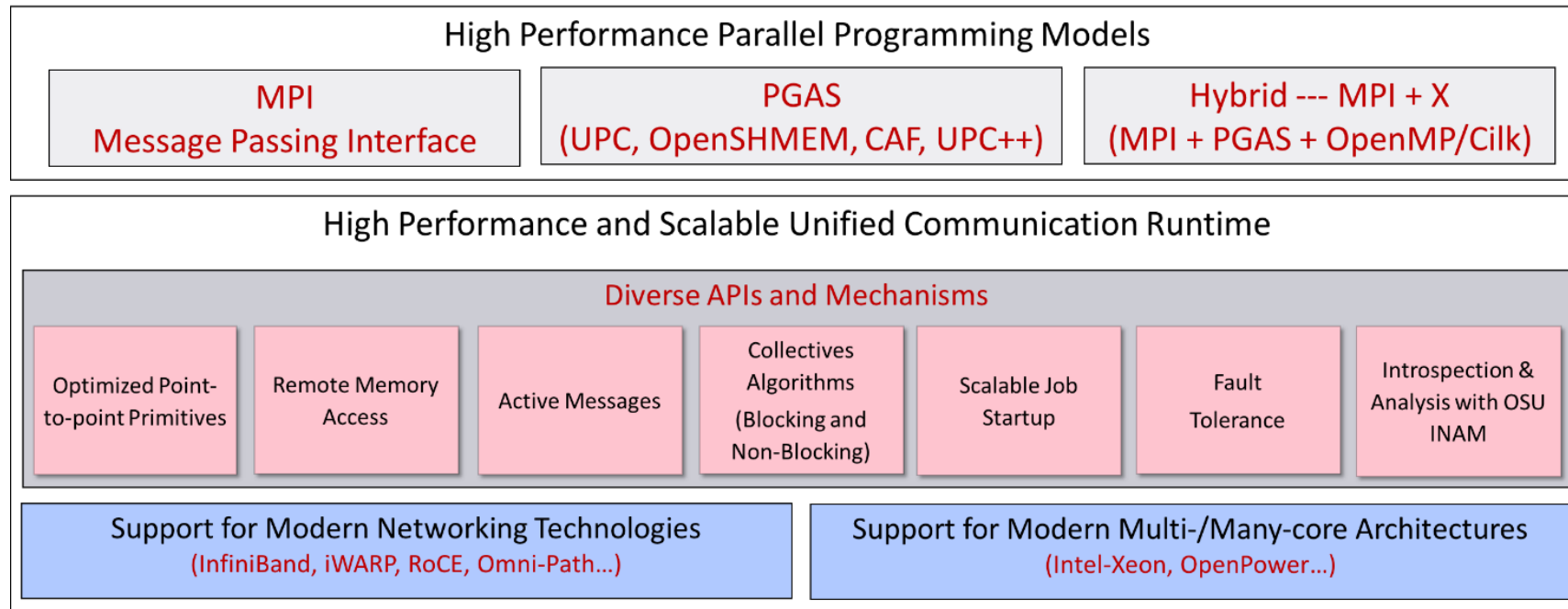
Part of input parameter: MESH IJK=5,5,5, XB=-1.0,0.0,-1.0,0.0,0.0,1.0, MULT\_ID='mesh array'

# MVAPICH2 Software Family

| Requirements                                                                                                                    | Library        |
|---------------------------------------------------------------------------------------------------------------------------------|----------------|
| MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                                    | MVAPICH2       |
| Optimized Support for Microsoft Azure Platform with InfiniBand                                                                  | MVAPICH2-Azure |
| Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis), | MVAPICH2-X     |
| Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric Adapter (EFA)                                      | MVAPICH2-X-AWS |
| Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications                                      | MVAPICH2-GDR   |
| Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                       | MVAPICH2-EA    |
| MPI Energy Monitoring Tool                                                                                                      | OEMT           |
| InfiniBand Network Analysis and Monitoring                                                                                      | OSU INAM       |
| Microbenchmarks for Measuring MPI and PGAS Performance                                                                          | OMB            |



# MVAPICH2-X for MPI and Hybrid MPI + PGAS Applications



- Current Model – Separate Runtimes for OpenSHMEM/UPC/UPC++/CAF and MPI
  - Possible deadlock if both runtimes are not progressed
  - Consumes more network resource
- Unified communication runtime for MPI, UPC, UPC++, OpenSHMEM, CAF
  - Available with since 2012 (starting with MVAPICH2-X 1.9)
  - <http://mvapich.cse.ohio-state.edu>

# MVAPICH2-X 2.3rc2

- Released on 03/01/2019
- Major Features and Enhancements
  - **MPI Features**
  - **Based on MVAPICH2 2.3.1**
    - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
  - **MPI (Advanced) Features**
    - Improved performance of large message communication
    - Support for advanced co-operative (COOP) rendezvous protocols in SMP channel
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Support for RGET, RPUT, and COOP protocols for CMA and XPMEM
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Support for load balanced and dynamic rendezvous protocol selection
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Support for XPMEM-based MPI collective operations (Broadcast, Gather, Scatter, Allgather)
      - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
    - Extend support for XPMEM-based MPI collective operations (Reduce and All-Reduce for PSM-CH3 and PSM2-CH3 interfaces)
  - Improved connection establishment for DC transport
    - OFA-IB-CH3 interface
  - Add improved Alltoallv algorithm for small messages
  - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
- **OpenSHMEM Features**
  - Support for XPMEM-based collective operations (Broadcast, Collect, Reduce\_all, Reduce, Scatter, Gather)
- **UPC Features**
  - Support for XPMEM-based collective operations (Broadcast, Collect, Scatter, Gather)
- **UPC++ Features**
  - Support for XPMEM-based collective operations (Broadcast, Collect, Scatter, Gather)
- **Unified Runtime Features**
  - Based on MVAPICH2 2.3.1 (OFA-IB-CH3 interface). All the runtime features enabled by default in OFA-IB-CH3 and OFA-IB-RoCE interface of MVAPICH2 2.3.1 are available in MVAPICH2-X 2.3rc2

# MVAPICH2-X Feature Table

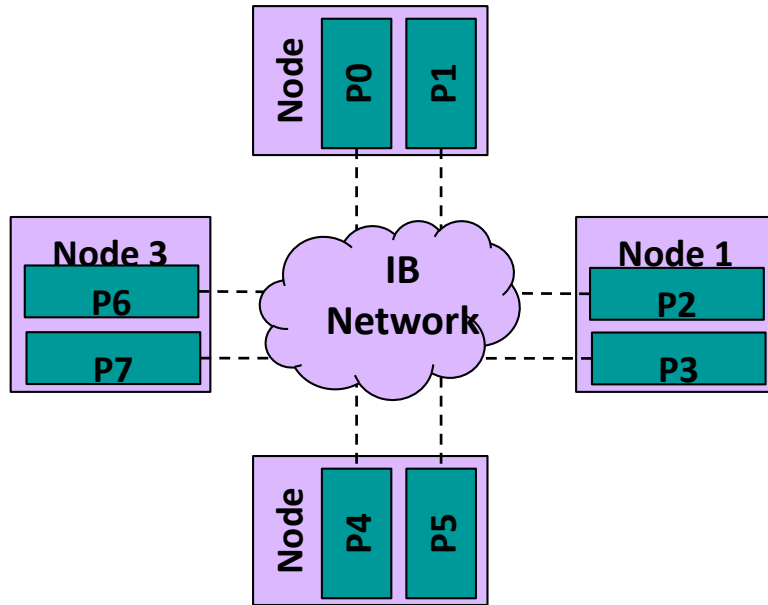
| Features for InfiniBand (OFA-IB-CH3) and RoCE (OFA-RoCE-CH3)                                  | Basic | Basic-XPMEM | Intermediate | Advanced |
|-----------------------------------------------------------------------------------------------|-------|-------------|--------------|----------|
| Architecture Specific Point-to-point and Collective Optimizations for x86, OpenPOWER, and ARM | ✓     | ✓           | ✓            | ✓        |
| Optimized Support for PGAS models (UPC, UPC++, OpenSHMEM, CAF) and Hybrid MPI+PGAS models     | ✓     | ✓           | ✓            | ✓        |
| CMA-Aware Collectives                                                                         | ✓     | ✓           | ✓            | ✓        |
| Optimized Asynchronous Progress*                                                              | ✓     | ✓           | ✓            | ✓        |
| InfiniBand Hardware Multicast-based MPI_Bcast**                                               | ✓     | ✓           | ✓            | ✓        |
| OSU InfiniBand Network Analysis and Monitoring (INAM)**                                       |       |             |              | ✓        |
| XPMEM-based Point-to-Point and Collectives                                                    |       | ✓           | ✓            | ✓        |
| Direct Connected (DC) Transport Protocol**                                                    |       |             | ✓            | ✓        |
| User mode Memory Registration (UMR)**                                                         |       |             |              | ✓        |
| On Demand Paging (ODP)**                                                                      |       |             |              | ✓        |
| Core-direct based Collective Offload**                                                        |       |             |              | ✓        |
| SHARP-based Collective Offload**                                                              |       |             |              | ✓        |

- \* indicates disabled by default at runtime. Must use appropriate environment variable in MVAPICH2-X user guide to enable it.
- + indicates features only tested with InfiniBand network

# Overview of MVAPICH2-X Features

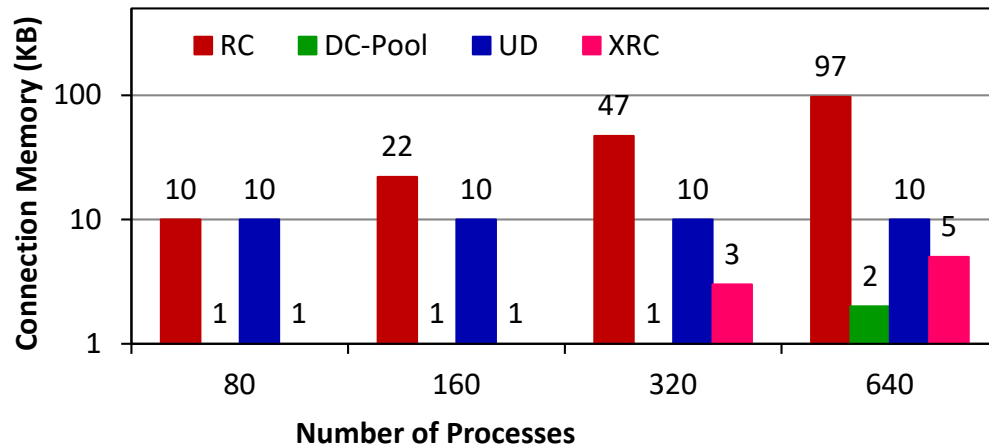
- **Direct Connect (DC) Transport**
  - Available from MVAPICH2-X 2.3rc1 onwards
- **Understanding Basic Intra-node Communication Mechanisms**
  - POSIX SHMEM vs. CMA vs. XPMEM
- **CMA-based Collectives**
  - Available from MVAPICH2-X 2.3rc1 onwards
- **Asynchronous Progress**
  - Available from MVAPICH2-X 2.3rc1 onwards
- **XPMEM-based Reduction Collectives**
  - Available from MVAPICH2-X 2.3rc1 onwards
- **XPMEM-based Non-reduction Collectives**
  - Available from MVAPICH2-X 2.3rc2 onwards
- **XPMEM-based MPI Derived Datatype Designs**
  - Will be available in future MVAPICH2-X releases
- **Optimized Collective Communication and Advanced Transport Protocols**
  - Available from MVAPICH2-X 2.3rc2 onwards
- **PGAS and Hybrid MPI+PGAS Support**
  - Available from MVAPICH2-X 2.1.9 onwards

# Minimizing Memory Footprint by Direct Connect (DC) Transport

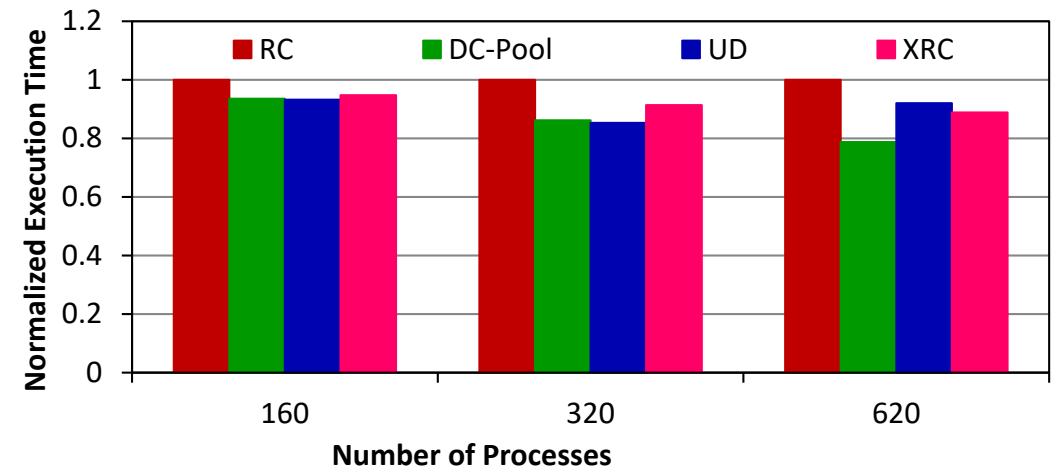


- Constant connection cost (*One QP for any peer*)
- Full Feature Set (RDMA, Atomics etc)
- Separate objects for send (DC Initiator) and receive (DC Target)
  - DC Target identified by “DCT Number”
  - Messages routed with (DCT Number, LID)
  - Requires same “DC Key” to enable communication
- Available since MVAPICH2-X 2.2a

Memory Footprint for Alltoall



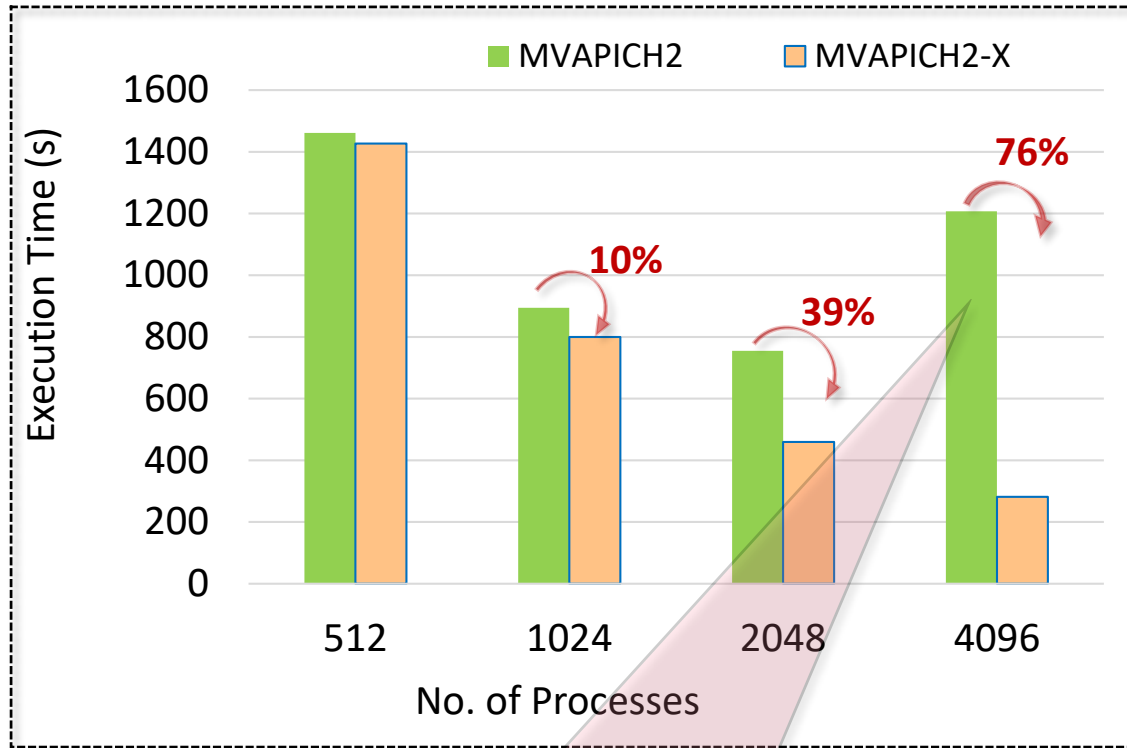
NAMD - Apoa1: Large data set



H. Subramoni, K. Hamidouche, A. Venkatesh, S. Chakraborty and D. K. Panda, Designing MPI Library with Dynamic Connected Transport (DCT) of InfiniBand : Early Experiences. IEEE International Supercomputing Conference (ISC '14)

# Impact of DC Transport Protocol on Neuron

## Neuron with YuEtAl2012



**Overhead of RC protocol for  
connection establishment and  
communication**

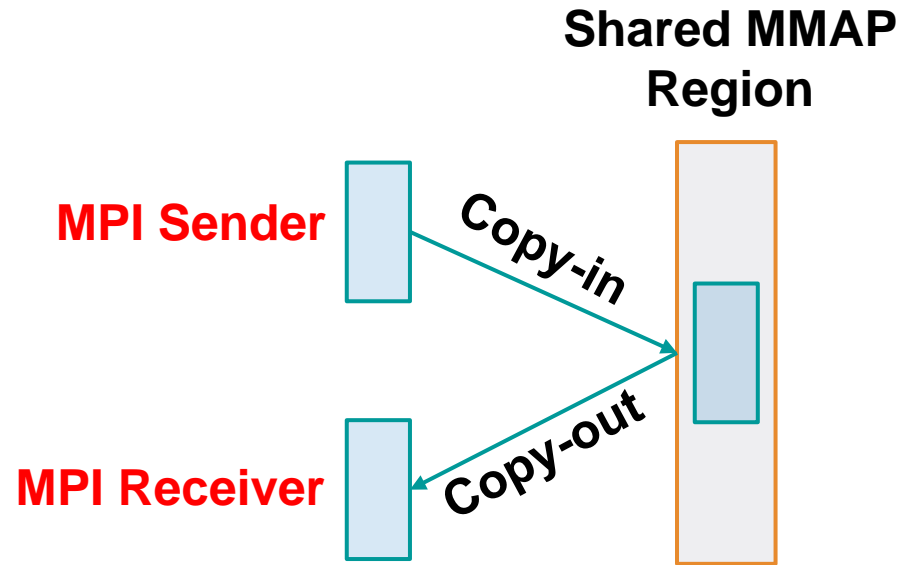
- Up to **76%** benefits over MVAPICH2 for Neuron using Direct Connected transport protocol at scale
  - VERSION 7.6.2 master (f5a1284) 2018-08-15
- Numbers taken on bbpv2.epfl.ch
  - Knights Landing nodes with 64 ppn
  - ./x86\_64/special -mpi -c stop\_time=2000 -c is\_split=1 parinit.hoc
  - Used “runtime” reported by execution to measure performance
- Environment variables used
  - MV2\_USE\_DC=1
  - MV2\_NUM\_DC\_TGT=64
  - MV2\_SMALL\_MSG\_DC\_POOL=96
  - MV2\_LARGE\_MSG\_DC\_POOL=96
  - MV2\_USE\_RDMA\_CM=0

*Available from MVAPICH2-X 2.3rc2 onwards*

# Overview of MVAPICH2-X Features

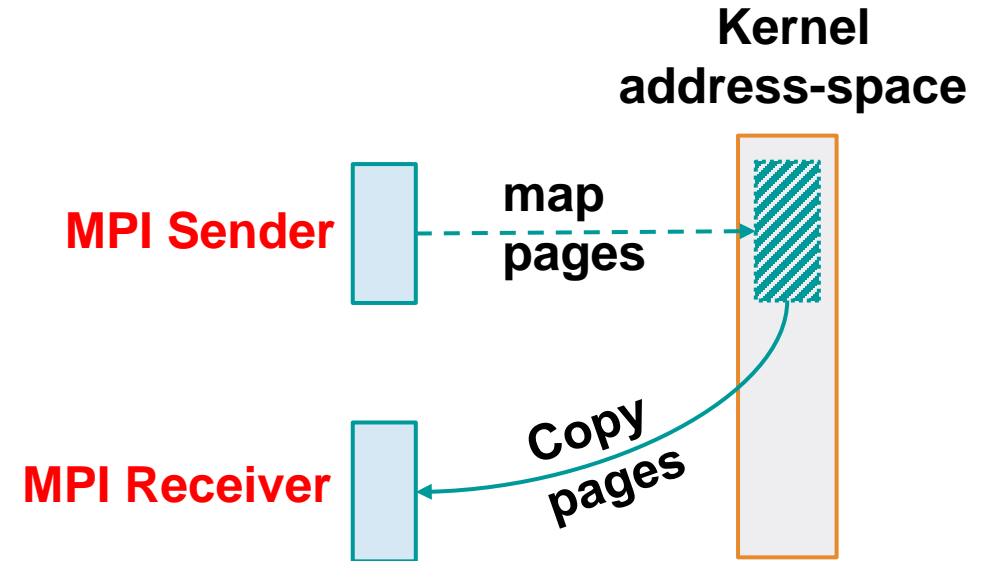
- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- **Understanding Basic Intra-node Communication Mechanisms**
  - **POSIX SHMEM vs. CMA vs. XPMEM**
- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
- XPMEM-based MPI Derived Datatype Designs
  - Will be available in future MVAPICH2-X releases
- Optimized Collective Communication and Advanced Transport Protocols
  - Available from MVAPICH2-X 2.3rc2 onwards
- PGAS and Hybrid MPI+PGAS Support
  - Available from MVAPICH2-X 2.1.9 onwards

# Existing Intra-Node Communication Mechanism in MPI



## Shared Memory – SHMEM

Requires two copies  
No system call overhead  
Better for Small Messages



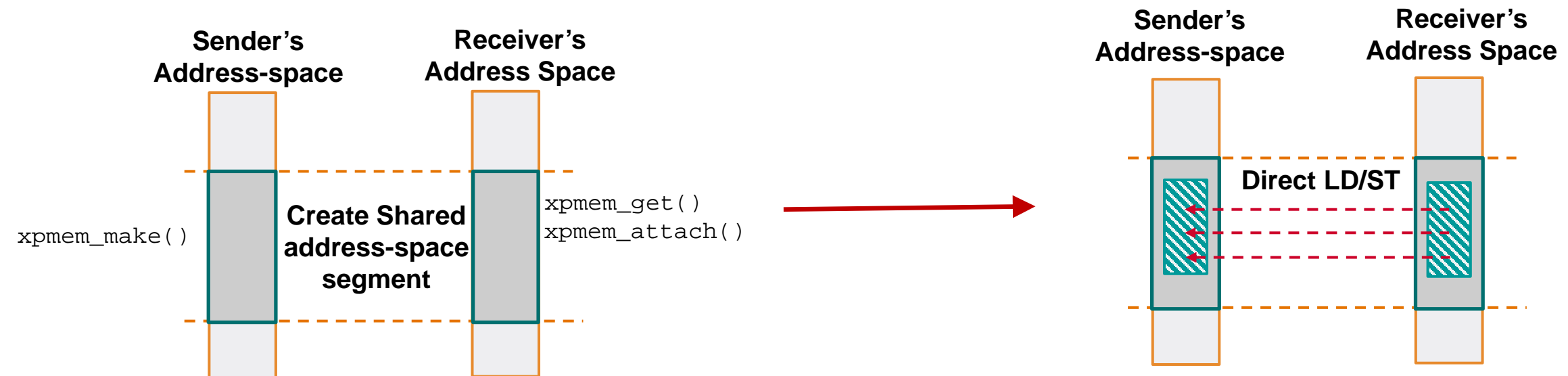
## Kernel-Assisted Copy

System call overhead  
Requires single(a.k.a “zero”) copy  
Better for Large Messages  
(CMA, KNEM, LiMIC)

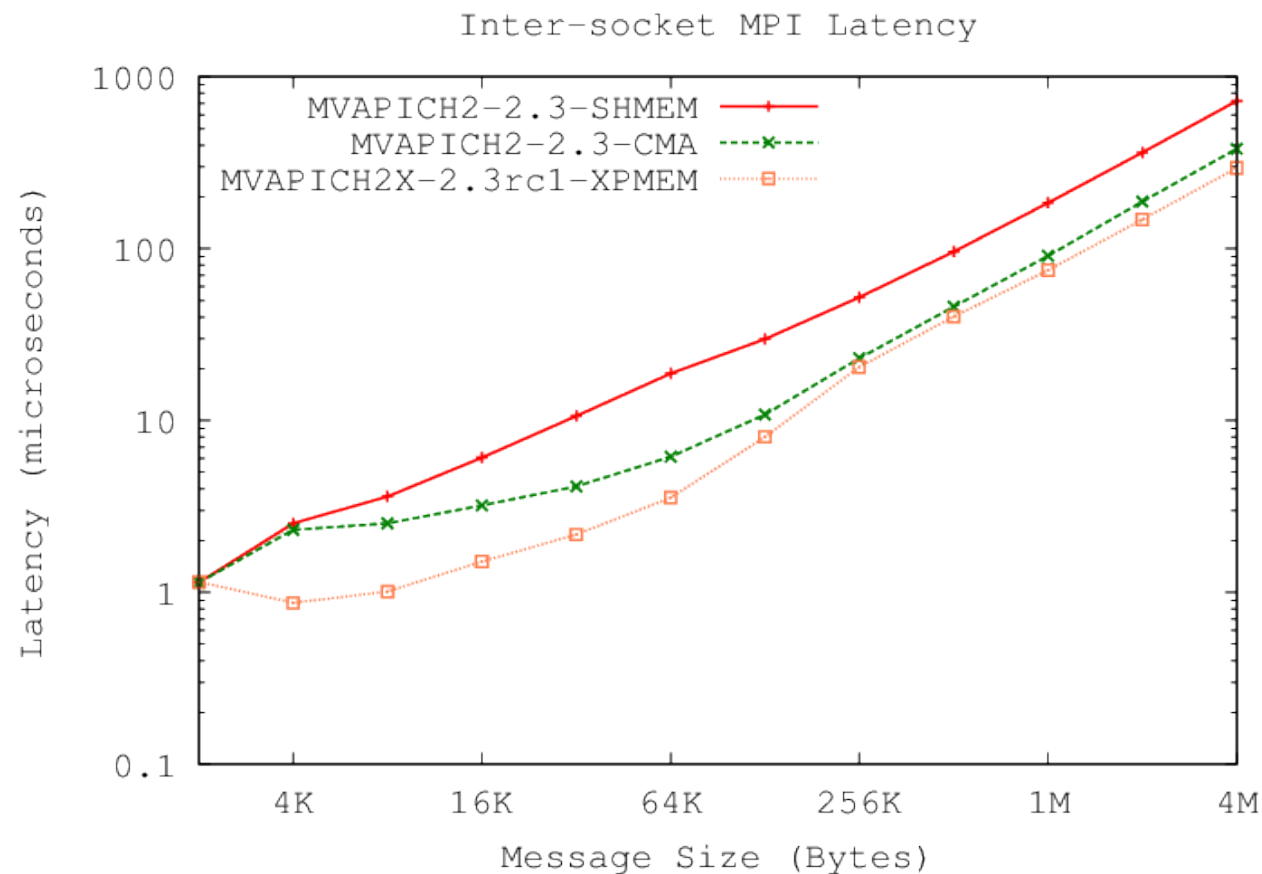
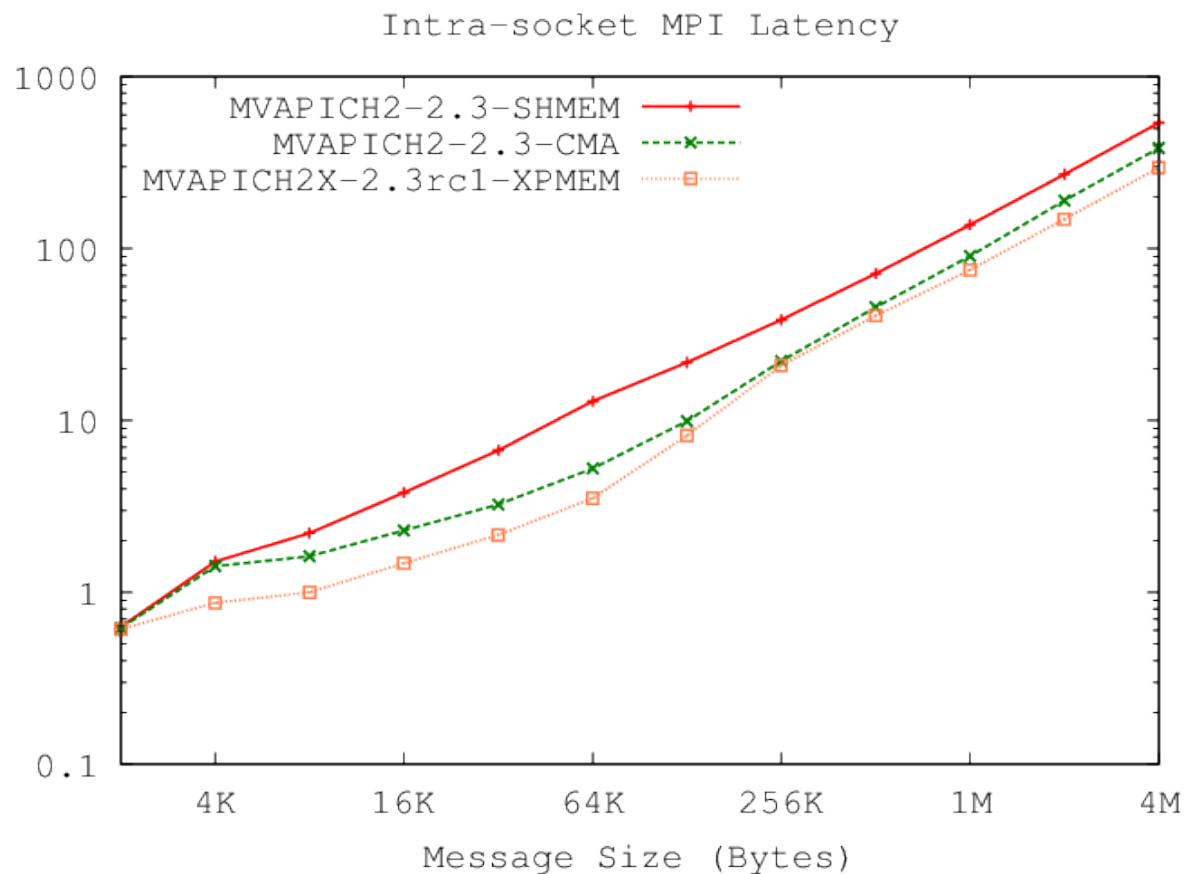


# Shared Address-space based Communication

- XPMEM (<https://github.com/hjelmn/xpmem>) --- “Cross-partition Memory”
  - Mechanisms for a process to “*attach*” to the virtual memory segment of a remote process
  - Consists of a user-space API and a kernel module
- The sender process calls “`xpmem_make ( )`” to create a shared segment
  - Segment information is then shared with the receiver
- The receiver process calls “`xpmem_get ( )`” followed by “`xpmem_attach ( )`”
- The receiver process can directly read/write on the remote process’ memory

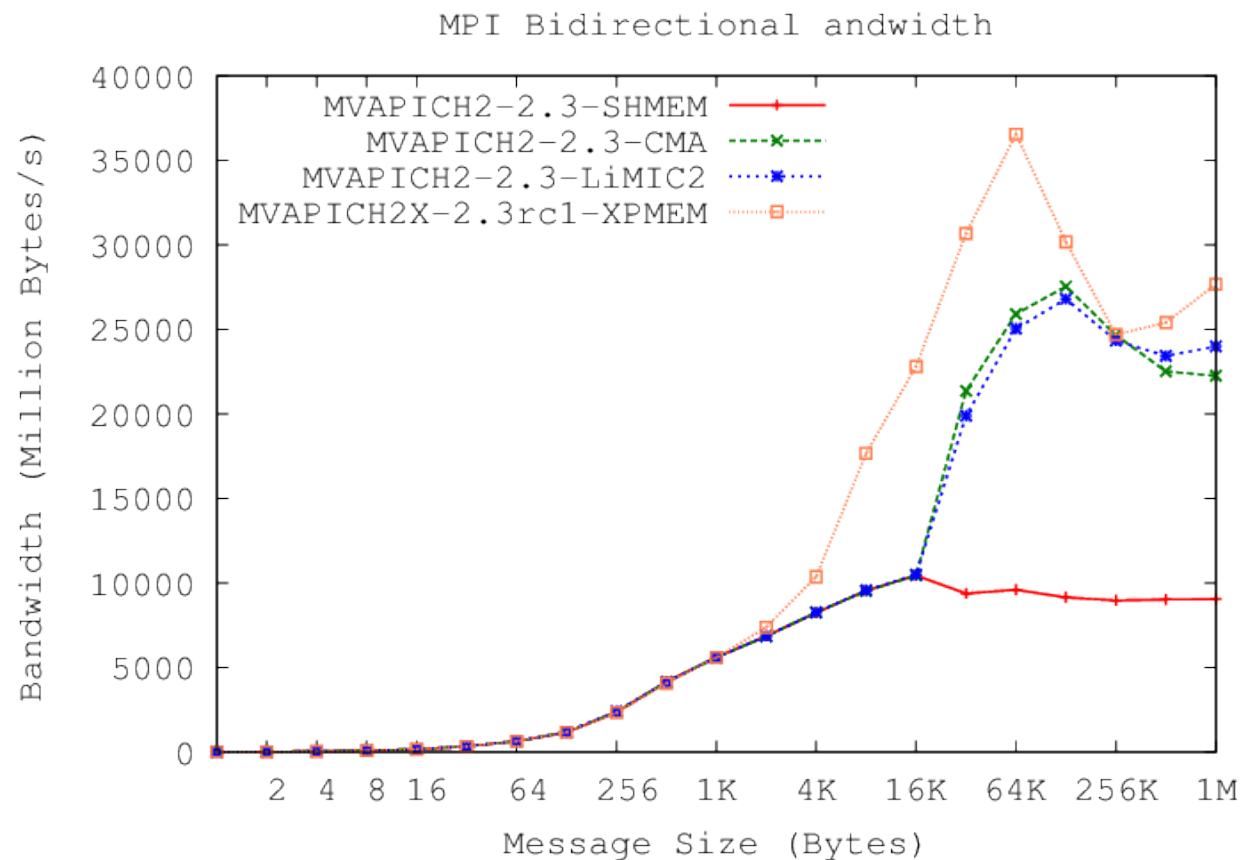
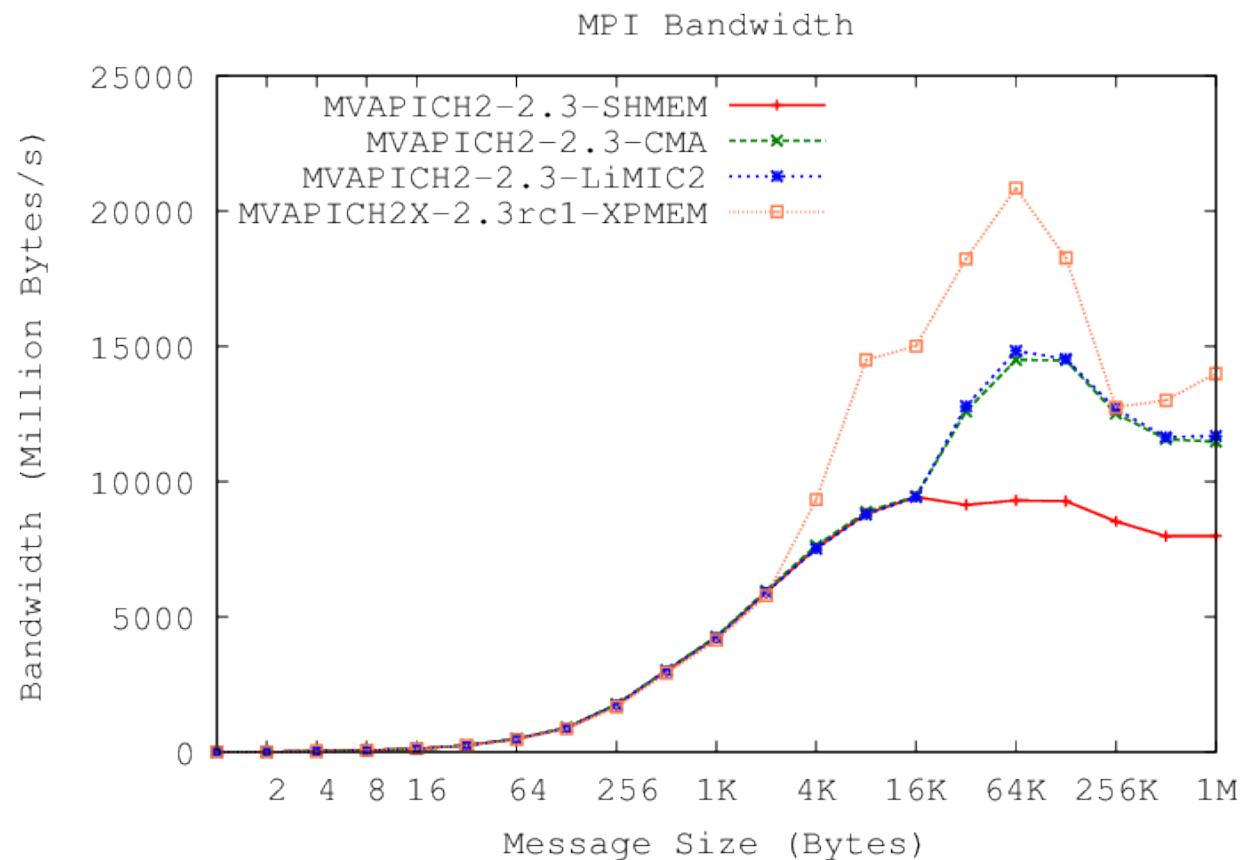


# MPI Level Point-to-Point Latency



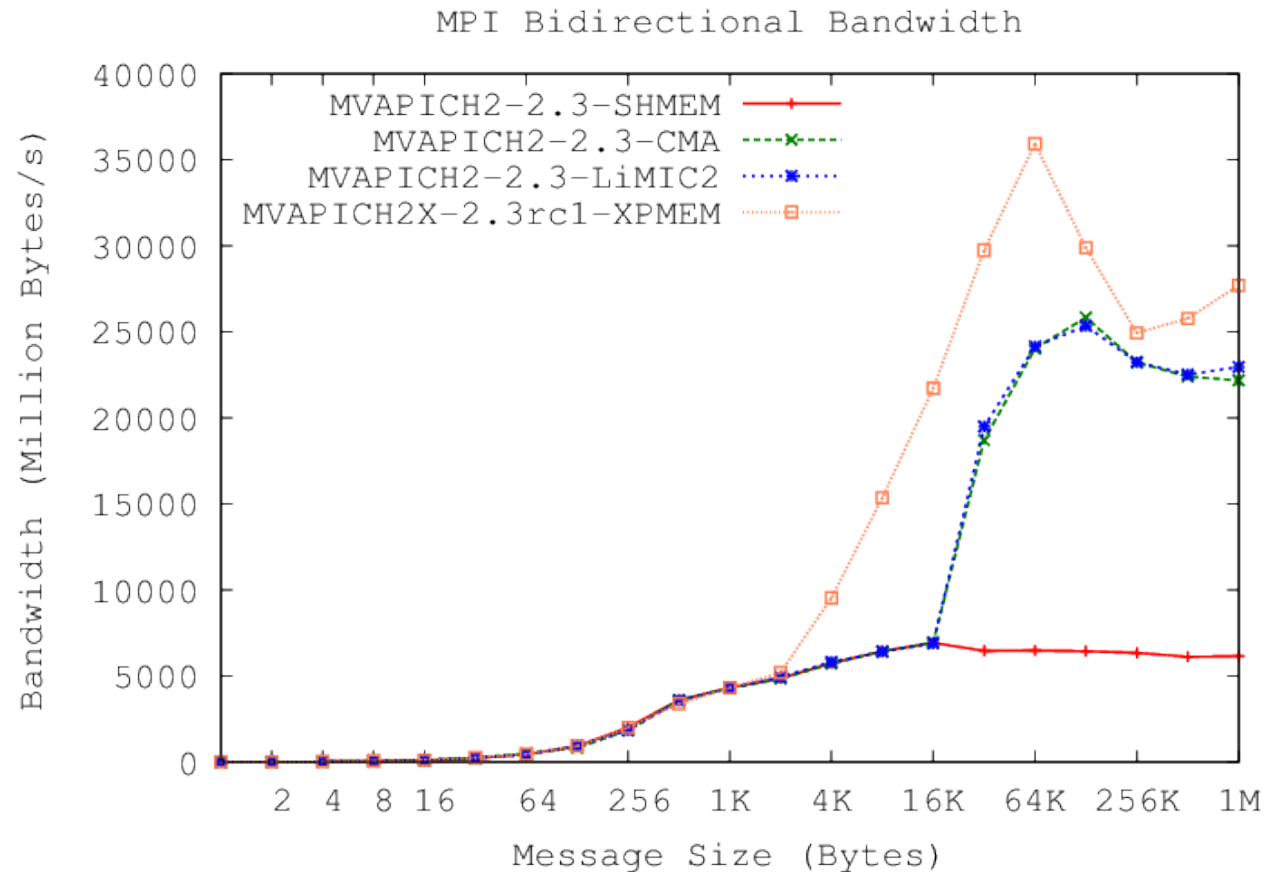
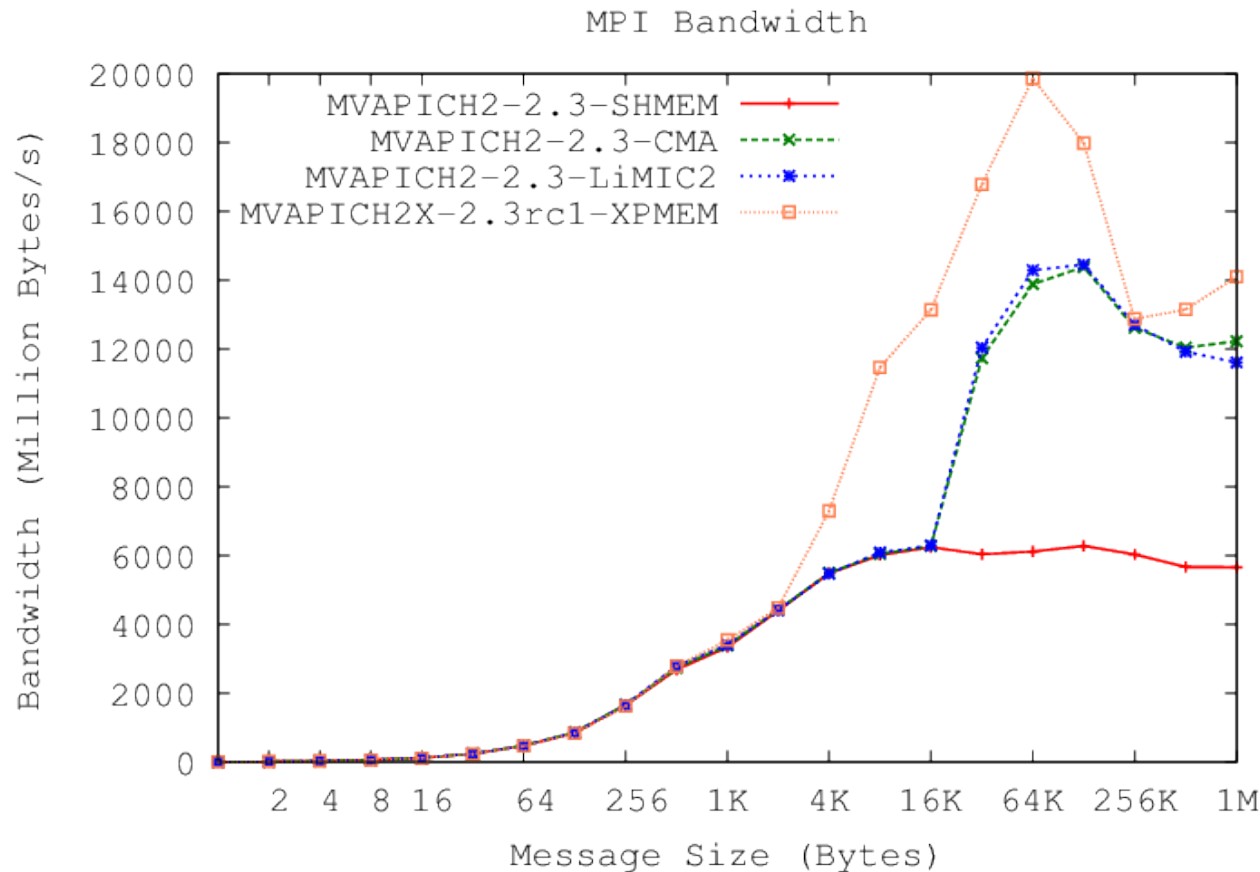
- Intel Xeon CPU E5-2687W v3 @ 3.10GHz (10-core, 2-socket)
- Used osu\_latency from OSU Microbenchmarks v5.5

# MPI Level Intra-socket Point-to-Point Bandwidth



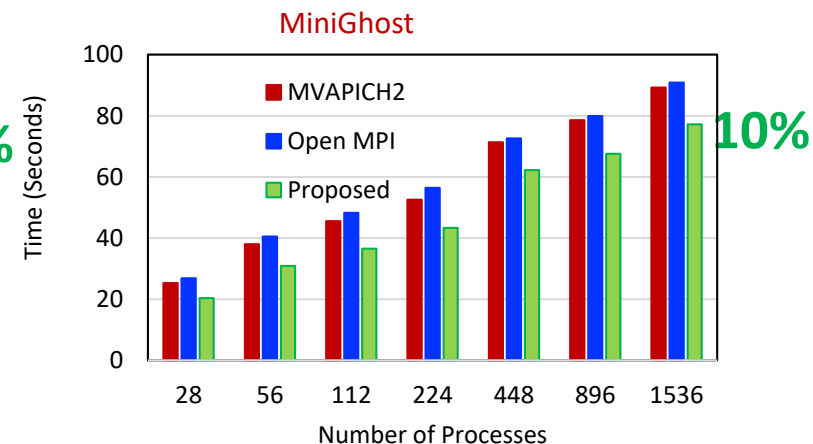
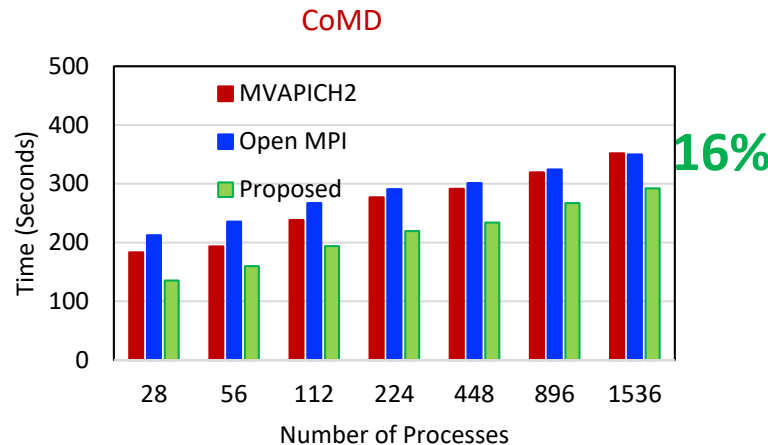
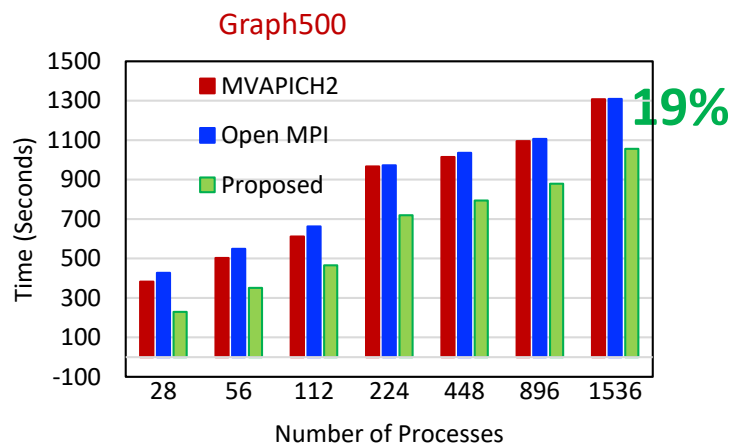
- Intel Xeon CPU E5-2687W v3 @ 3.10GHz (10-core, 2-socket)
- Used osu\_bw and osu\_bibw from OSU Microbenchmarks v5.5

# MPI Level Inter-Socket Point-to-Point Bandwidth



- Intel XeonCPU E5-2687W v3 @ 3.10GHz (10-core, 2-socket)
- Used osu\_bw and osu\_bibw from OSU Microbenchmarks v5.5

# Cooperative Rendezvous Protocols



- Use both sender and receiver CPUs to progress communication concurrently
- Dynamically select rendezvous protocol based on communication primitives and sender/receiver availability (load balancing)
- Up to 2x improvement in large message latency and bandwidth
- Up to 19% improvement for Graph500 at 1536 processes

Cooperative Rendezvous Protocols for Improved Performance and Overlap

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

SC '18 (Best Student Paper Award Finalist)

Platform: 2x14 core Broadwell 2680 (2.4 GHz)

Mellanox EDR ConnectX-5 (100 GBps)

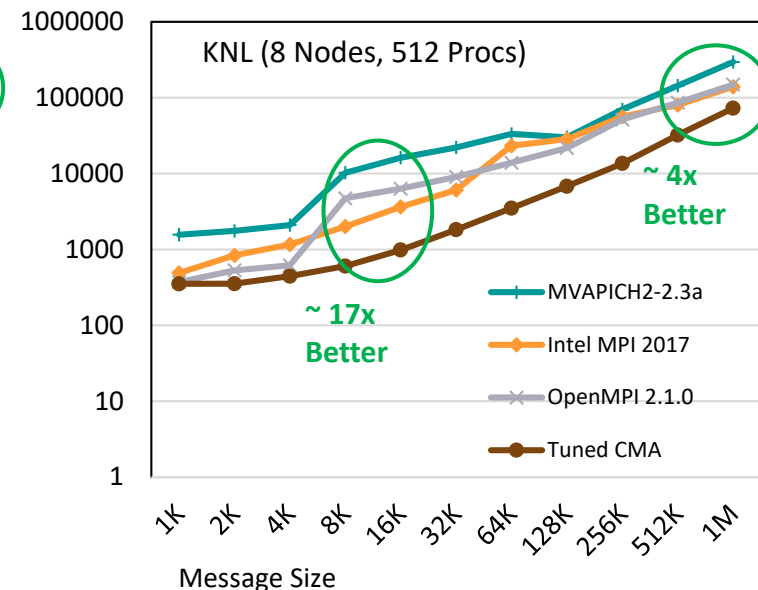
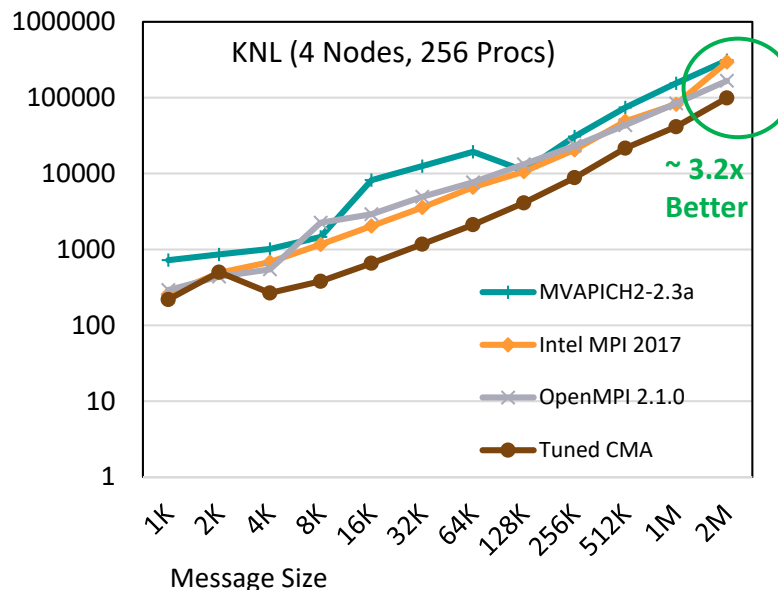
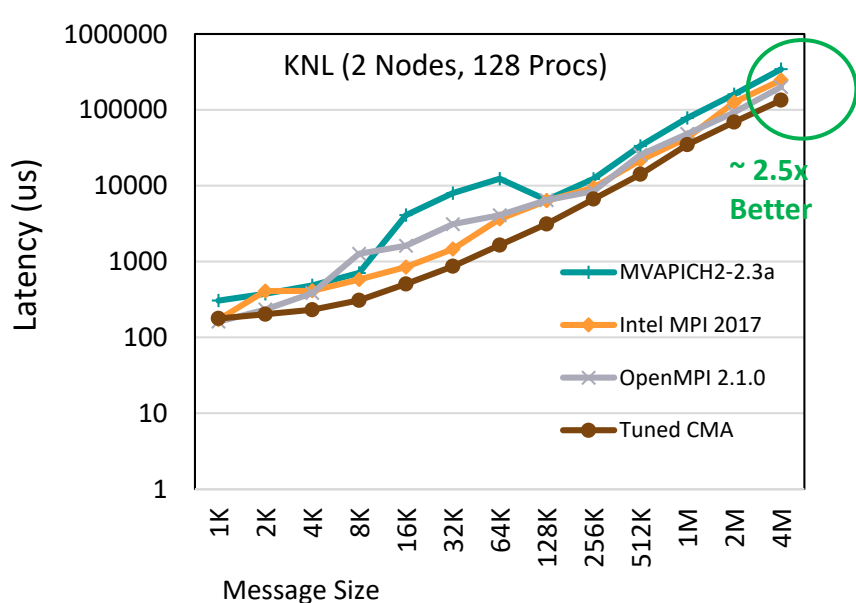
Baseline: MVAPICH2X-2.3rc1, Open MPI v3.1.0

Available in MVAPICH2-X 2.3rc2

# Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- Understanding Basic Intra-node Communication Mechanisms
  - POSIX SHMEM vs. CMA vs. XPMEM
- **CMA-based Collectives**
  - **Available from MVAPICH2-X 2.3rc1 onwards**
- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
- XPMEM-based MPI Derived Datatype Designs
  - Will be available in future MVAPICH2-X releases
- Optimized Collective Communication and Advanced Transport Protocols
  - Available from MVAPICH2-X 2.3rc2 onwards
- PGAS and Hybrid MPI+PGAS Support
  - Available from MVAPICH2-X 2.1.9 onwards

# Optimized CMA-based Collectives for Large Messages



Performance of MPI\_Gather on KNL nodes (64PPN)

- Significant improvement over existing implementation for Scatter/Gather with 1MB messages (up to 4x on KNL, 2x on Broadwell, 14x on OpenPower)
- New two-level algorithms for better scalability
- Improved performance for other collectives (Bcast, Allgather, and Alltoall)

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

Available since MVAPICH2-X 2.3b

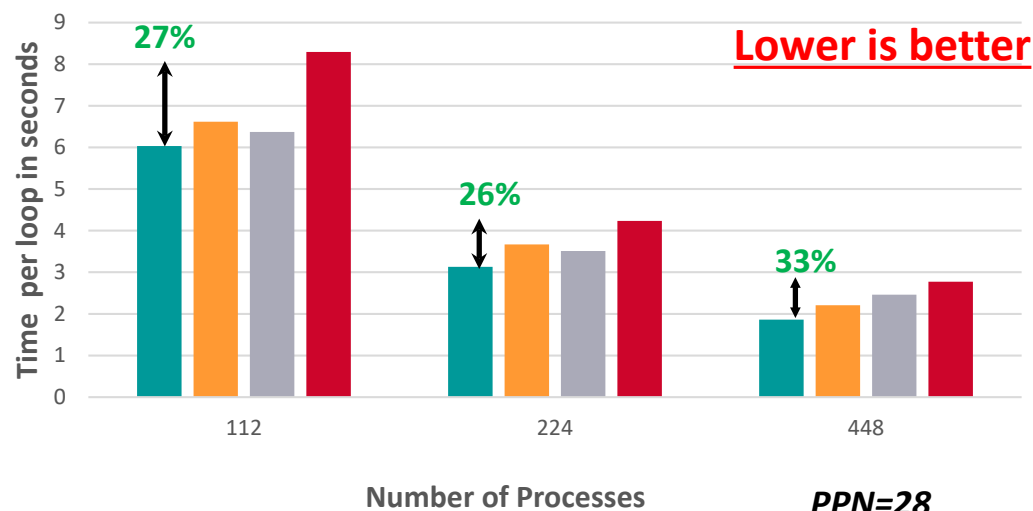
# Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- Understanding Basic Intra-node Communication Mechanisms
  - POSIX SHMEM vs. CMA vs. XPMEM
- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- **Asynchronous Progress**
  - **Available from MVAPICH2-X 2.3rc1 onwards**
- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
- XPMEM-based MPI Derived Datatype Designs
  - Will be available in future MVAPICH2-X releases
- Optimized Collective Communication and Advanced Transport Protocols
  - Available from MVAPICH2-X 2.3rc2 onwards
- PGAS and Hybrid MPI+PGAS Support
  - Available from MVAPICH2-X 2.1.9 onwards



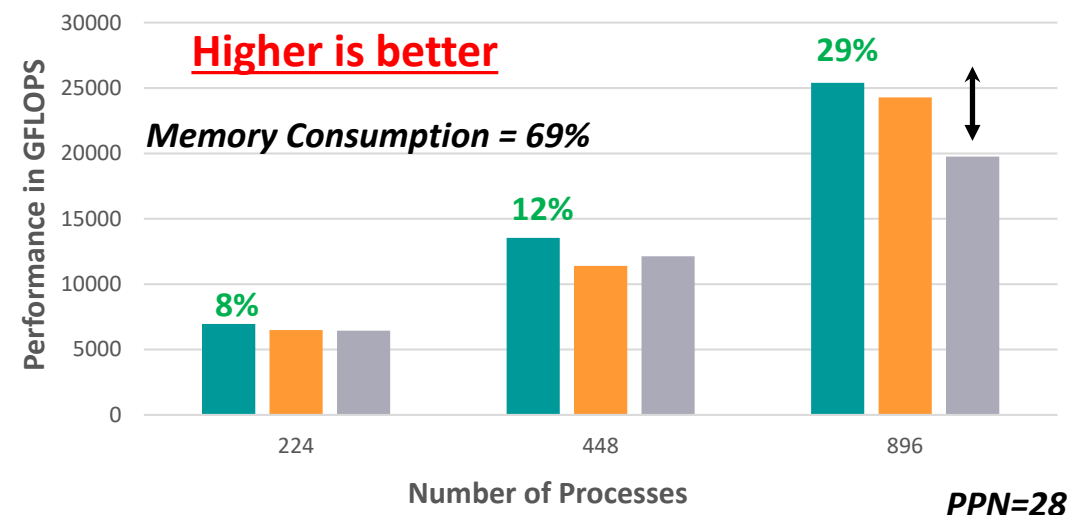
# Benefits of the New Asynchronous Progress Design: Broadwell + InfiniBand

## P3DFFT



■ MVAPICH2 Async ■ MVAPICH2 Default ■ IMPI 2019 Default ■ IMPI 2019 Async

## High Performance Linpack (HPL)



■ MVAPICH2 Async ■ MVAPICH2 Default ■ IMPI 2019 Default

Up to **33%** performance improvement in P3DFFT application with 448 processes

Up to **29%** performance improvement in HPL application with 896 processes

A. Ruhela, H. Subramoni, S. Chakraborty, M. Bayatpour, P. Kousha, and D.K. Panda,  
“Efficient design for MPI Asynchronous Progress without Dedicated Resources”, Parallel Computing 2019

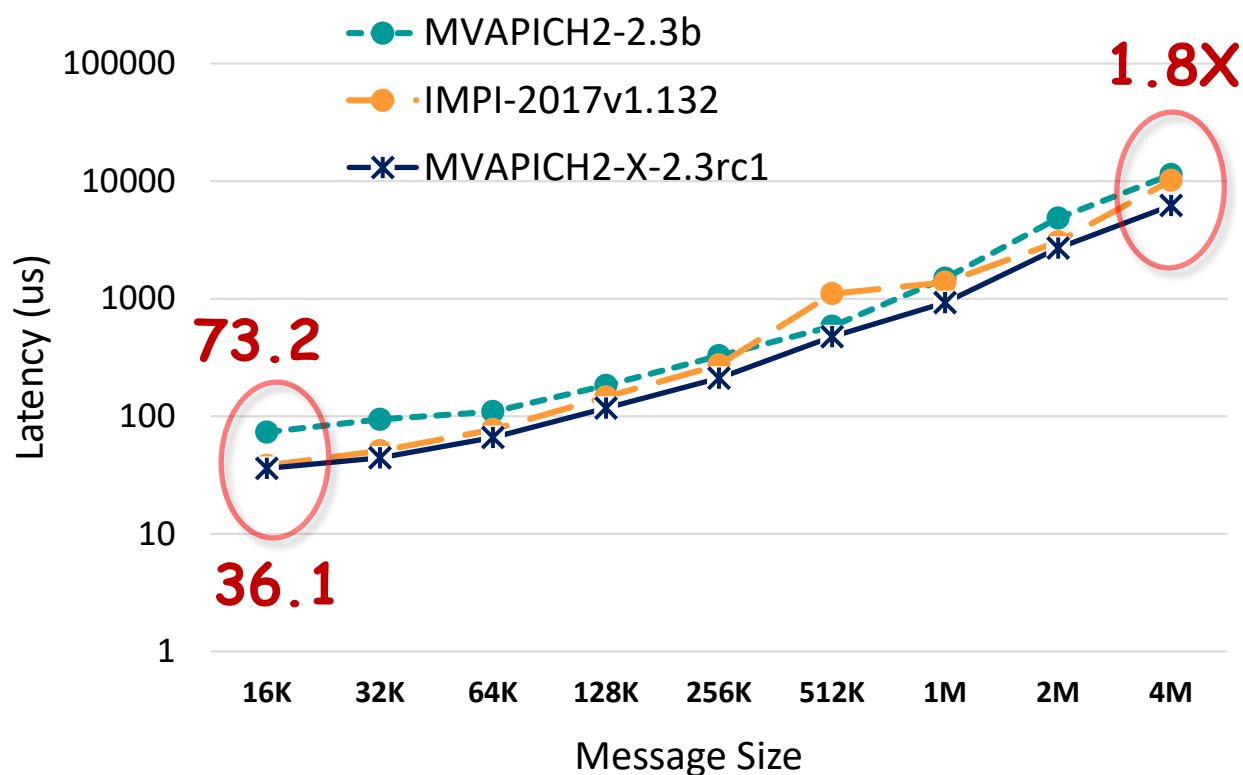
Available since MVAPICH2-X 2.3rc1

# Overview of MVAPICH2-X Features

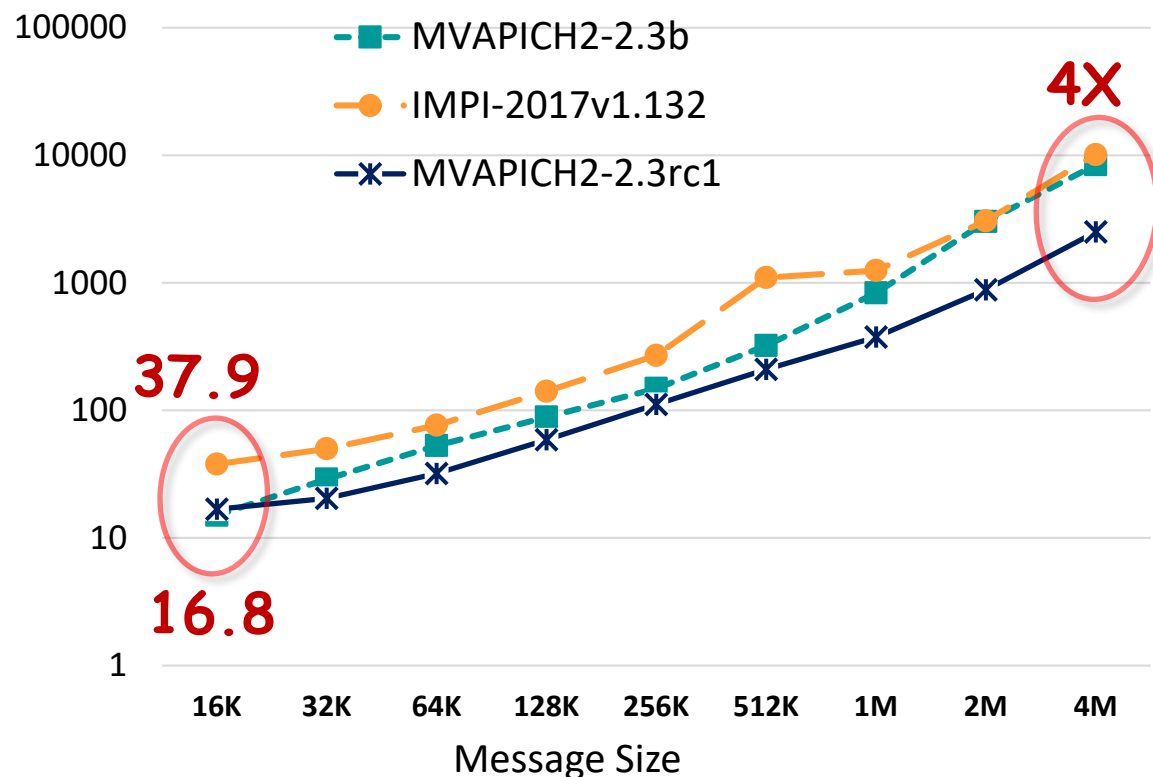
- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- Understanding Basic Intra-node Communication Mechanisms
  - POSIX SHMEM vs. CMA vs. XPMEM
- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards
- **XPMEM-based Reduction Collectives**
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
- XPMEM-based MPI Derived Datatype Designs
  - Will be available in future MVAPICH2-X releases
- Optimized Collective Communication and Advanced Transport Protocols
  - Available from MVAPICH2-X 2.3rc2 onwards
- PGAS and Hybrid MPI+PGAS Support
  - Available from MVAPICH2-X 2.1.9 onwards

# Shared Address Space (XPMEM)-based Collectives Design

## OSU\_Allreduce (Broadwell 256 procs)



## OSU\_Reduce (Broadwell 256 procs)

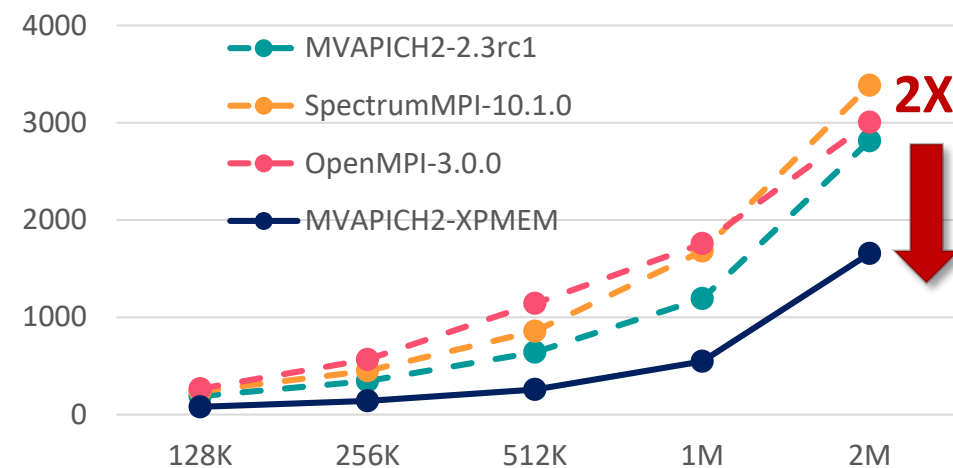
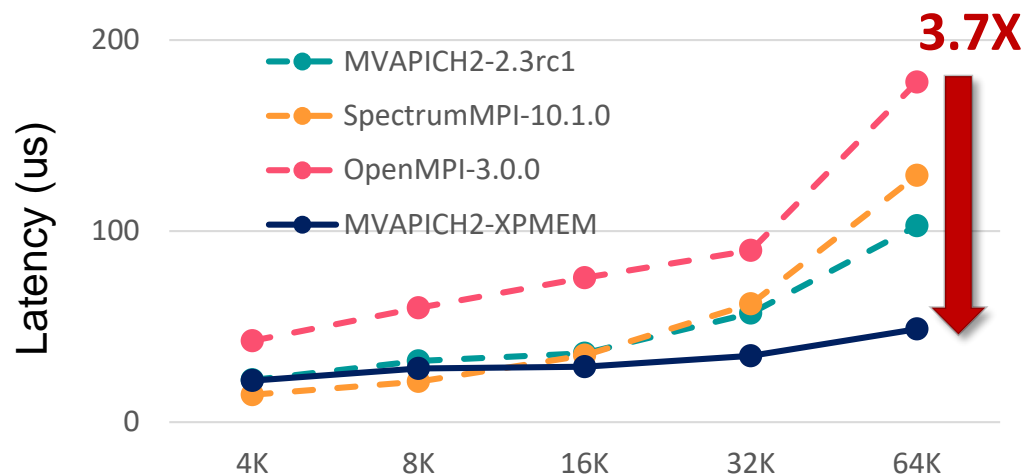


- “Shared Address Space”-based true zero-copy Reduction collective designs in MVAPICH2
- Offloaded computation/communication to peers ranks in reduction collective operation
- Up to **4X** improvement for 4MB Reduce and up to **1.8X** improvement for 4M AllReduce

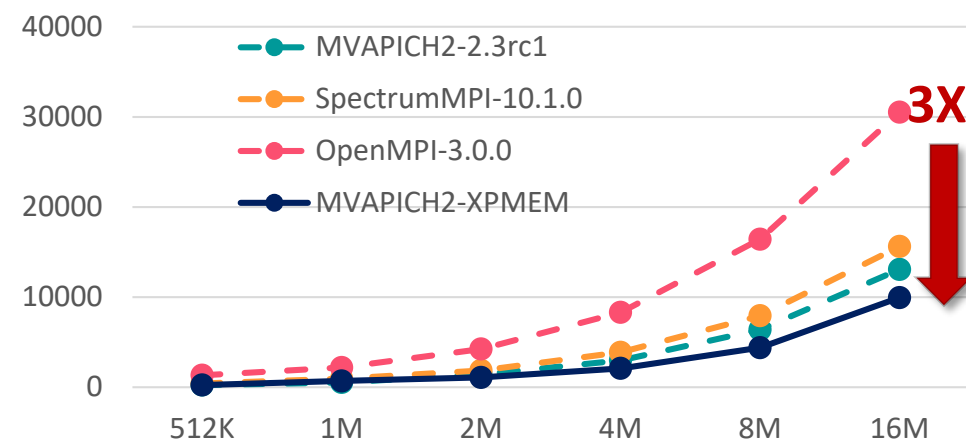
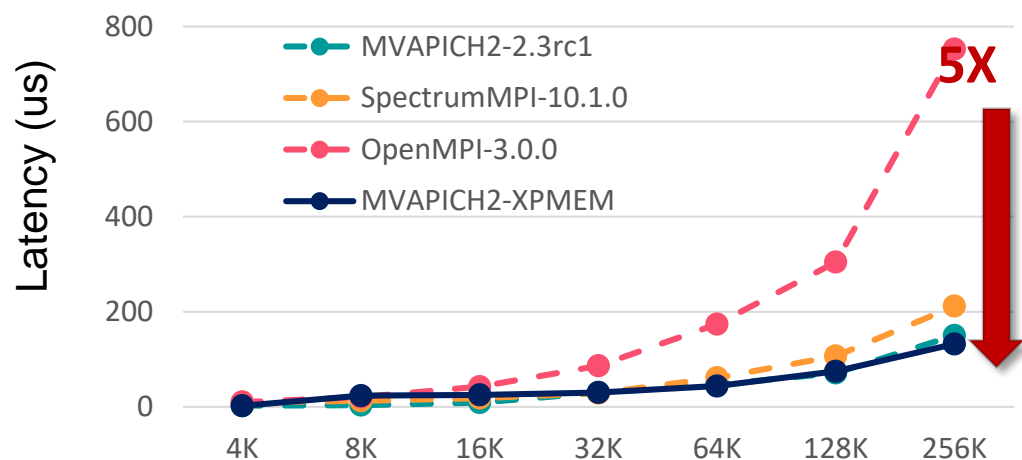
J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and D. Panda, Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores, International Parallel & Distributed Processing Symposium (IPDPS '18), May 2018. Available since MVAPICH2-X 2.3rc1

# Reduction Collectives on IBM OpenPOWER

MPI\_Allreduce



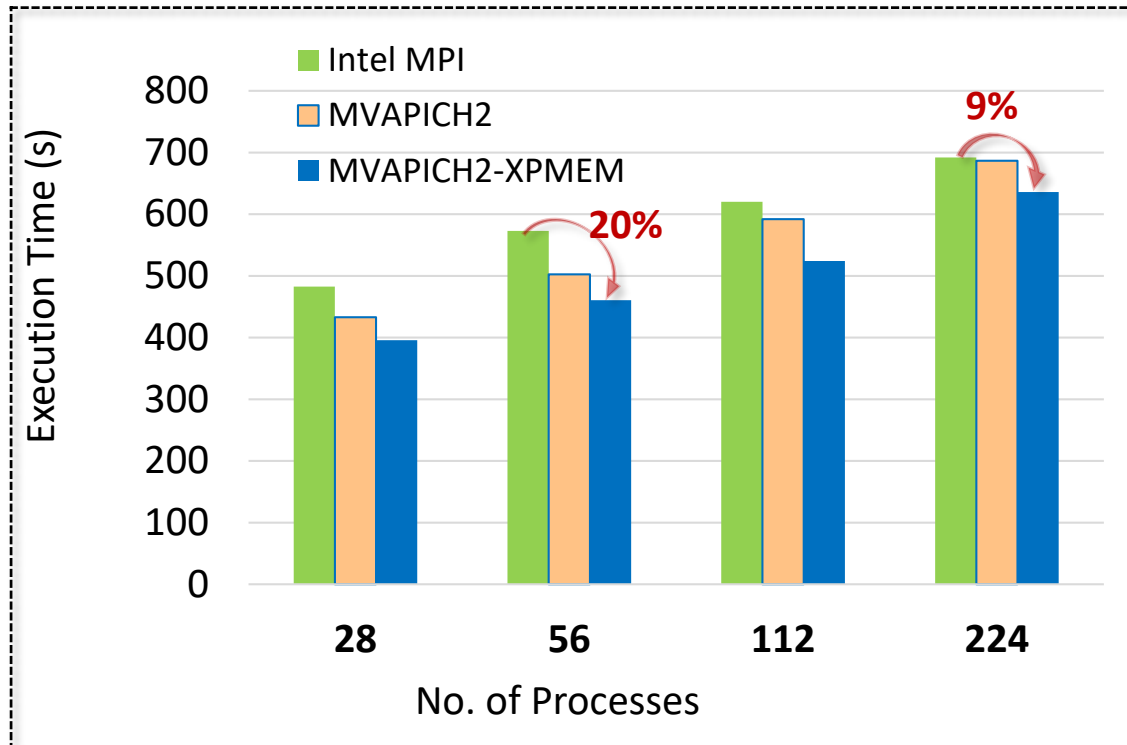
MPI\_Reduce



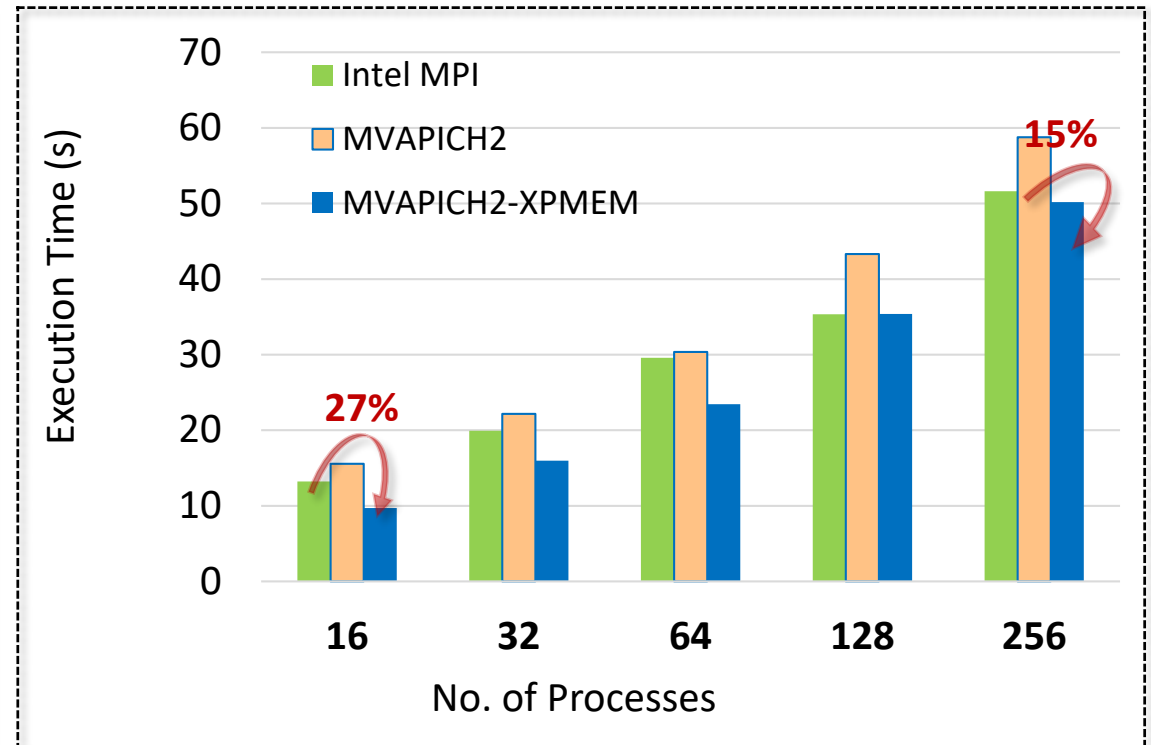
- Two POWER8 dual-socket nodes each with 20 ppn
- Up to **2X** improvement for Allreduce and **3X** improvement for Reduce at 4MB message
- Used osu\_reduce and osu\_allreduce from OSU Microbenchmarks v5.5

# Application Level Benefits of XPMEM-based Designs

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

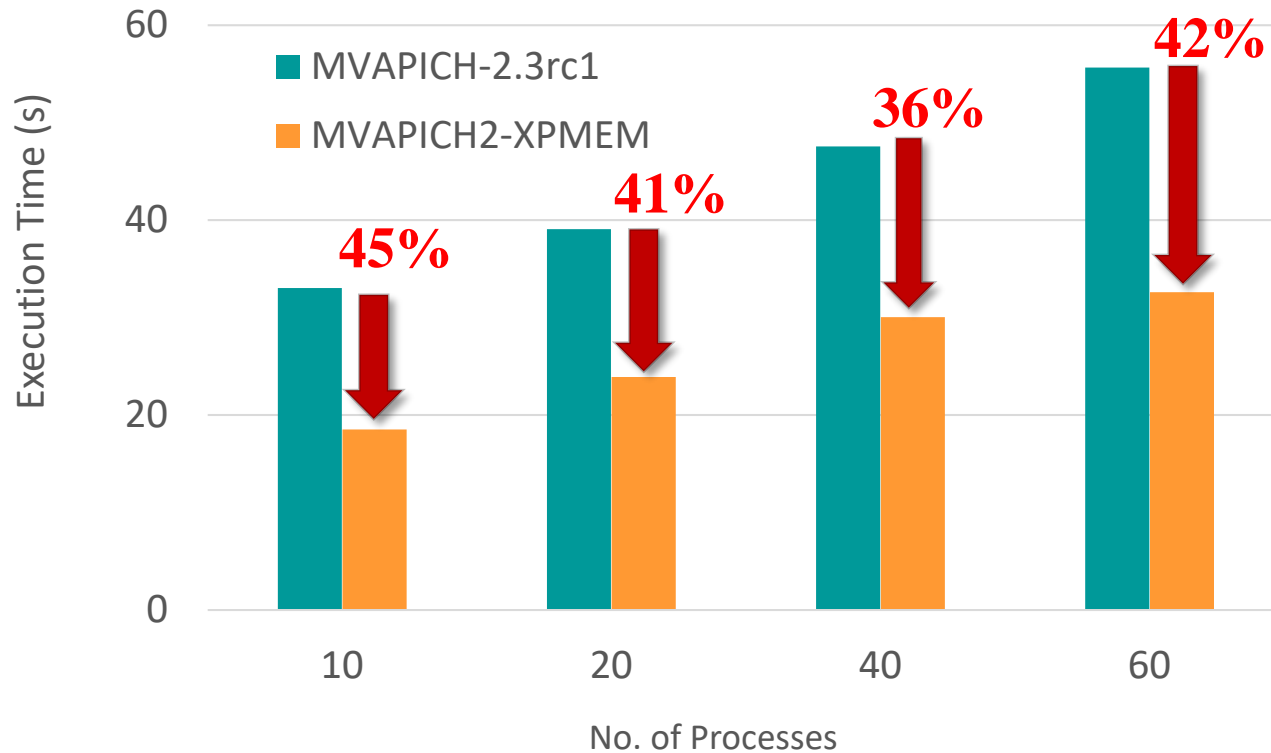


MiniAMR (dual-socket, ppn=16)



- Intel XeonCPU E5-2687W v3 @ 3.10GHz (10-core, 2-socket)
- 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

# Impact of XPMEM-based Designs on MiniAMR



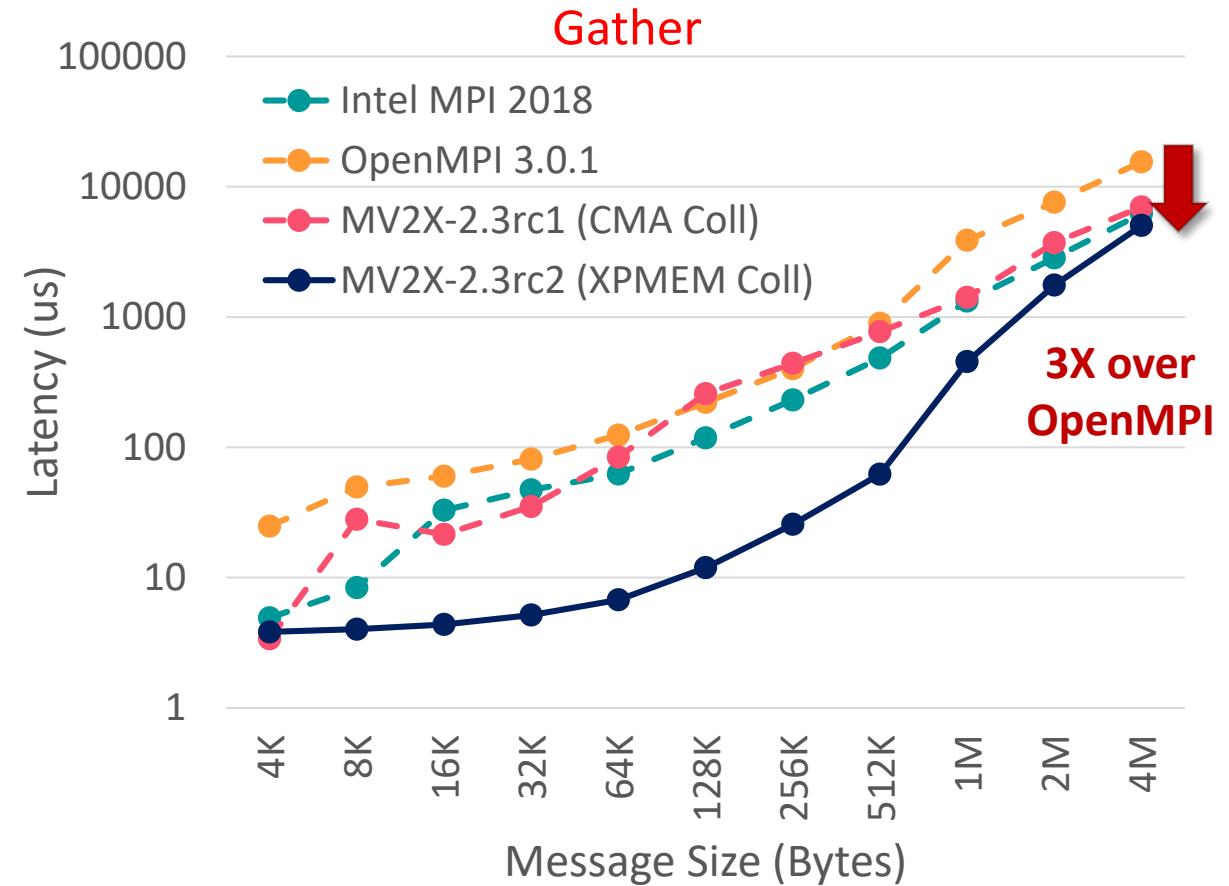
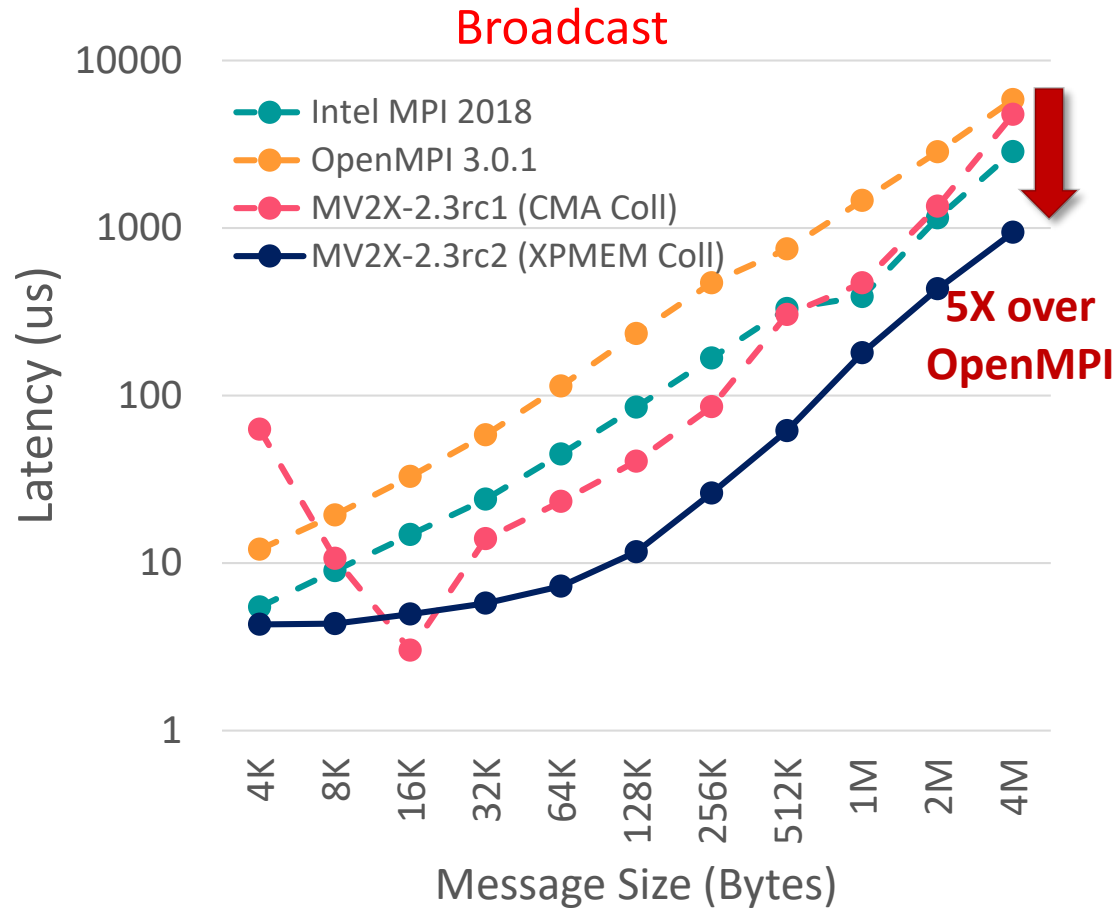
OpenPOWER (weak-scaling, 3 nodes, ppn=20)

- Two POWER8 dual-socket nodes each with 20 ppn
- MiniAMR application execution time comparing MVAPICH2-2.3rc1 and optimized All-Reduce design
  - MiniAMR application for weak-scaling workload on up to three POWER8 nodes.
  - Up to 45% improvement over MVAPICH2-2.3rc1 in mesh-refinement time

# Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- Understanding Basic Intra-node Communication Mechanisms
  - POSIX SHMEM vs. CMA vs. XPMEM
- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- **XPMEM-based Non-reduction Collectives**
  - Available from MVAPICH2-X 2.3rc2 onwards
- XPMEM-based MPI Derived Datatype Designs
  - Will be available in future MVAPICH2-X releases
- Optimized Collective Communication and Advanced Transport Protocols
  - Available from MVAPICH2-X 2.3rc2 onwards
- PGAS and Hybrid MPI+PGAS Support
  - Available from MVAPICH2-X 2.1.9 onwards

# Performance of Non-Reduction Collectives with XPMEM



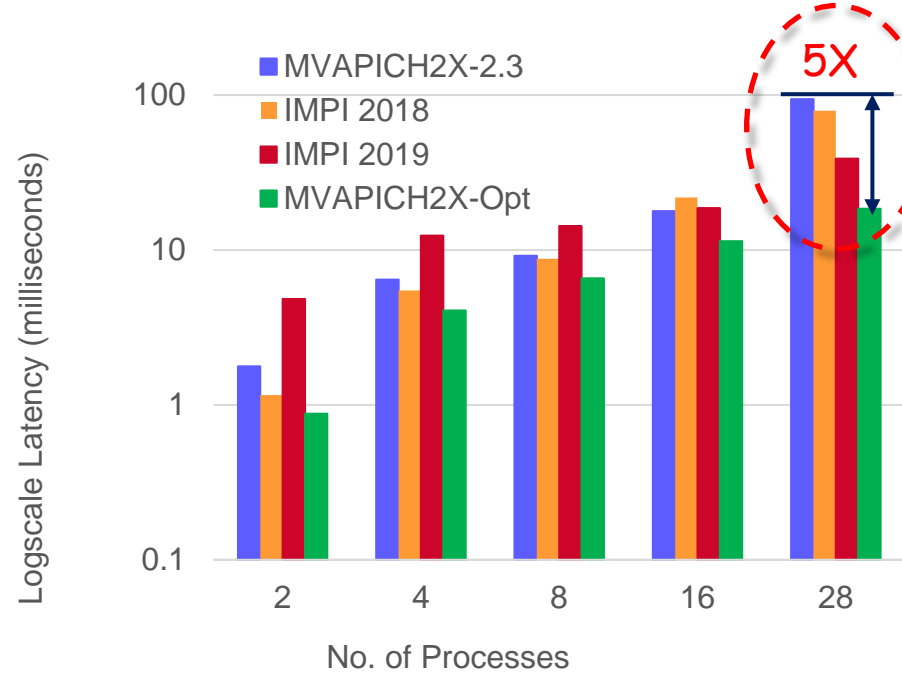
- **28 MPI Processes** on single dual-socket Broadwell E5-2680v4, 2x14 core processor
- Used `osu_bcast` from OSU Microbenchmarks v5.5



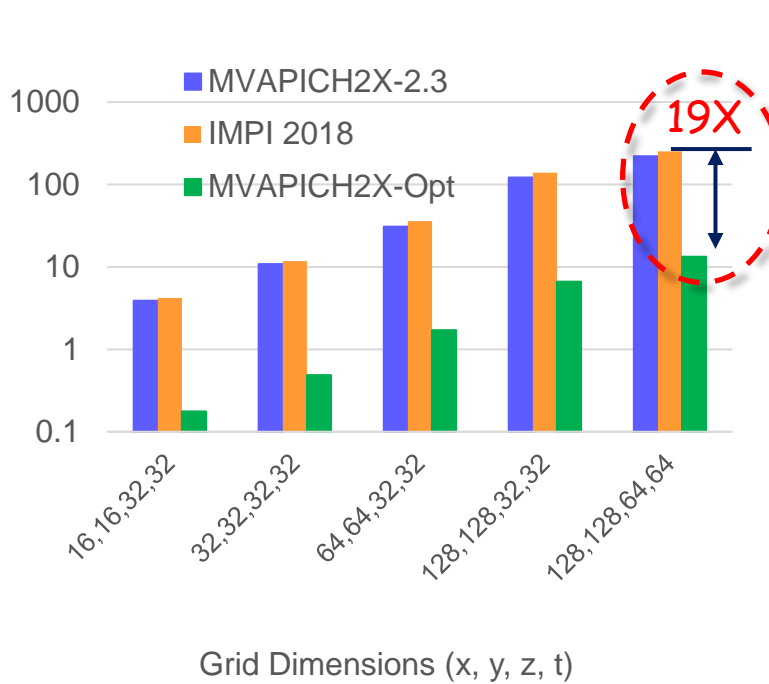
# Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- Understanding Basic Intra-node Communication Mechanisms
  - POSIX SHMEM vs. CMA vs. XPMEM
- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
- **XPMEM-based MPI Derived Datatype Designs**
  - Will be available in future MVAPICH2-X releases
- Optimized Collective Communication and Advanced Transport Protocols
  - Available from MVAPICH2-X 2.3rc2 onwards
- PGAS and Hybrid MPI+PGAS Support
  - Available from MVAPICH2-X 2.1.9 onwards

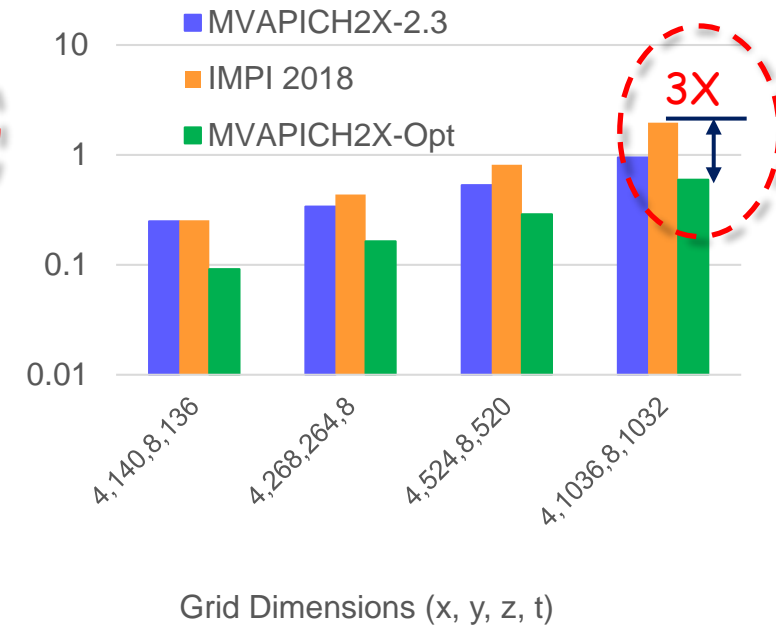
# Efficient Zero-copy MPI Datatypes for Emerging Architectures



**3D-Stencil** Datatype Kernel on  
**Broadwell** (2x14 core)



**MILC** Datatype Kernel on **KNL 7250**  
in Flat-Quadrant Mode (64-core)



**NAS-MG** Datatype Kernel on  
**OpenPOWER** (20-core)

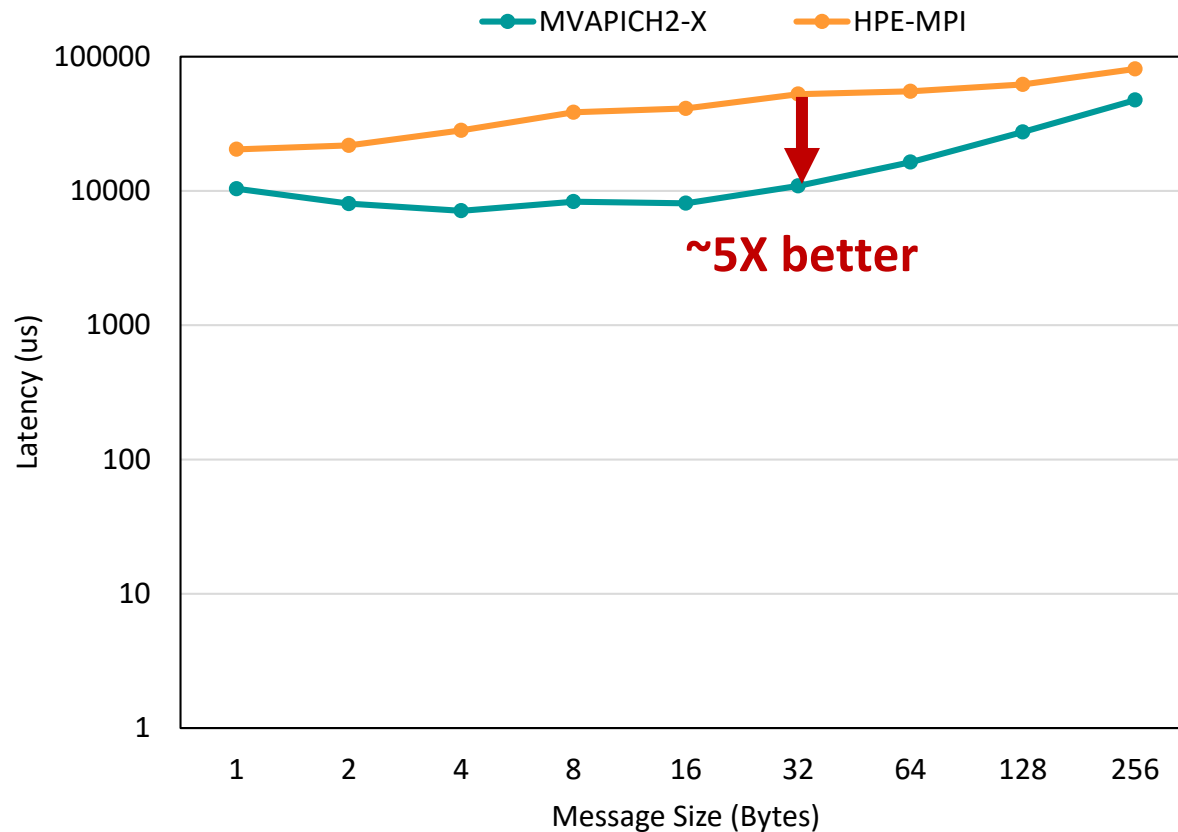
- New designs for efficient zero-copy based MPI derived datatype processing
- Efficient schemes mitigate datatype translation, packing, and exchange overheads
- Demonstrated benefits over prevalent MPI libraries for various application kernels
- To be available in the upcoming MVAPICH2-X release

# Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- Understanding Basic Intra-node Communication Mechanisms
  - POSIX SHMEM vs. CMA vs. XPMEM
- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
- XPMEM-based MPI Derived Datatype Designs
  - Will be available in future MVAPICH2-X releases
- **Optimized Collective Communication and Advanced Transport Protocols**
  - **Available from MVAPICH2-X 2.3rc2 onwards**
- PGAS and Hybrid MPI+PGAS Support
  - Available from MVAPICH2-X 2.1.9 onwards

# Impact of Optimized Small Message MPI\_Alltoallv Algorithm

- Optimized designs in MVAPICH2-X offer significantly improved performance for small message MPI\_Alltoallv



*Courtesy: Pramod Shivaji Kumbhar@EPFL*

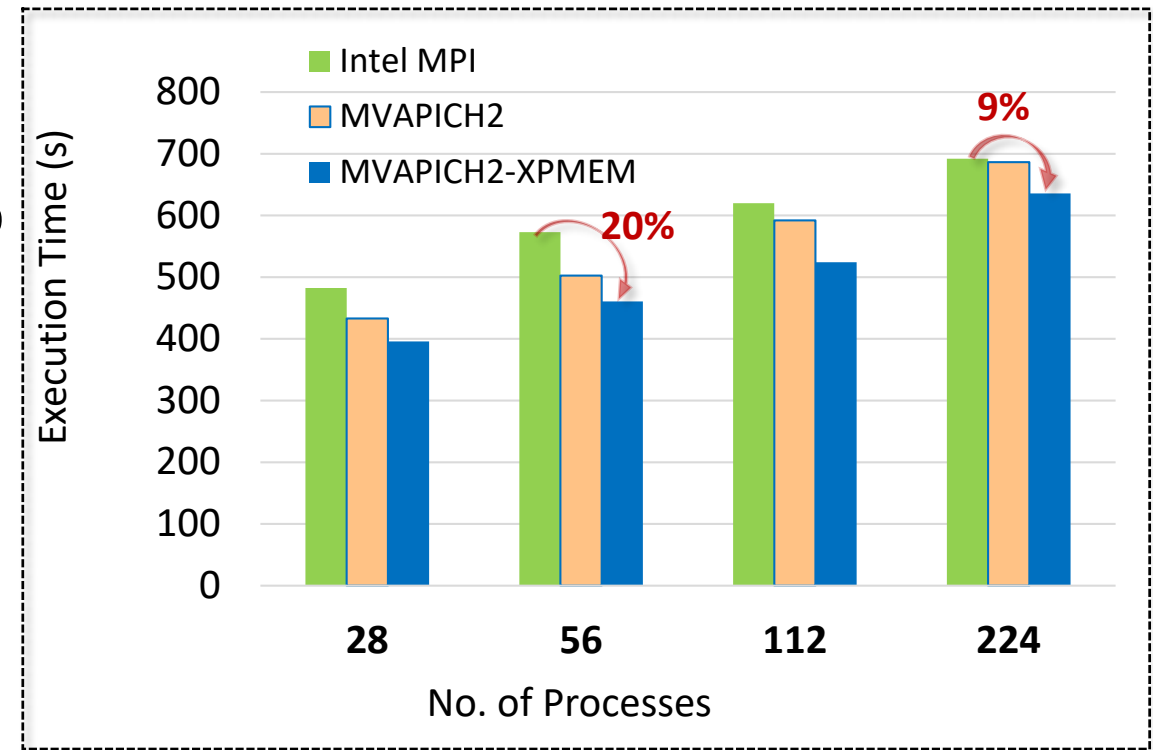
- Up to **5X** benefits over HPE-MPI using optimized using optimized Alltoallv algorithm and Direct Connected transport protocol
- Numbers taken on bbpv2.epfl.ch
  - 96 KNL nodes with 64 ppn (6,144 processes)
  - osu\_alltoallv from OSU Micro Benchmarks
- Environment variables used
  - MV2\_USE\_DC=1
  - MV2\_NUM\_DC\_TGT=64
  - MV2\_SMALL\_MSG\_DC\_POOL=96
  - MV2\_LARGE\_MSG\_DC\_POOL=96
  - MV2\_USE\_RDMA\_CM=0

*Available from MVAPICH2-X 2.3rc2 onwards*

# Performance of CNTK with MVAPICH2-X on CPU-based Deep Learning

- CPU-based training of AlexNet neural network using ImageNet ILSVRC2012 dataset
- Advanced XPMEM-based designs show up to **20%** benefits over Intel MPI (IMPI) for CNTK DNN training using All\_Reduce
- The proposed designs show good scalability with increasing system size

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

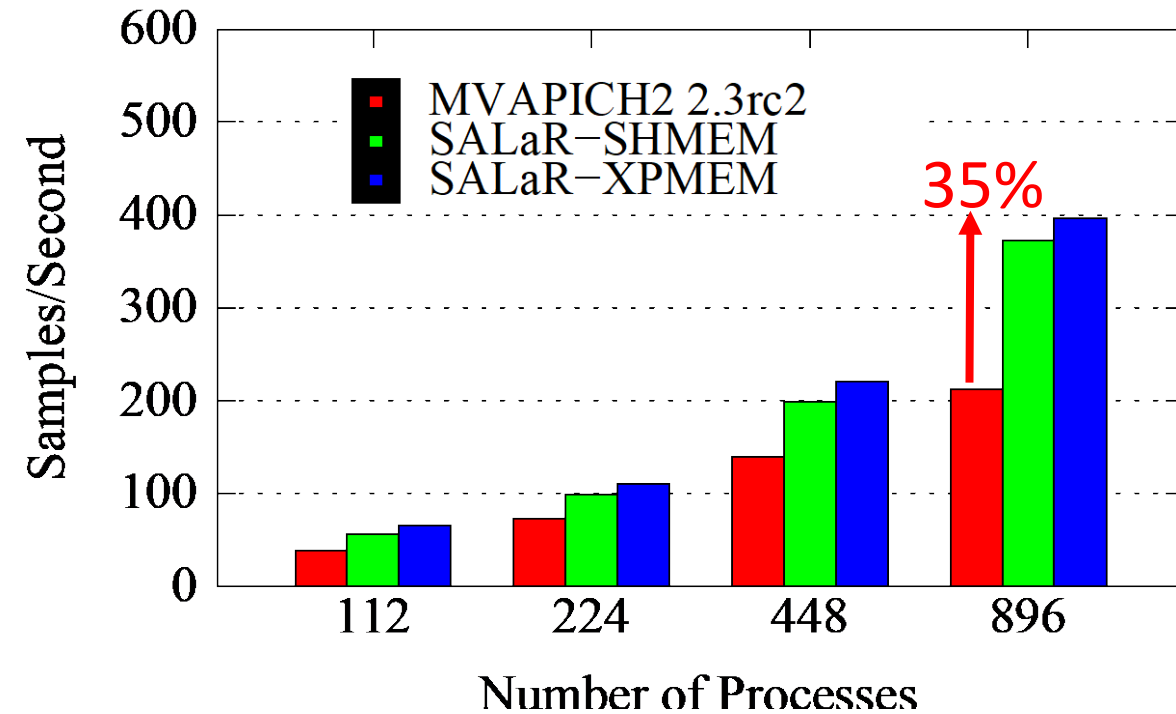


Available since MVAPICH2-X 2.3rc1 release

Designing Efficient Shared Address Space Reduction Collectives for Multi-/Many-cores, J. Hashmi, S. Chakraborty, M. Bayatpour, H. Subramoni, and DK Panda, 32nd IEEE International Parallel & Distributed Processing Symposium (IPDPS '18), May 2018

# Performance of TensorFlow with MVAPICH2-X on CPU

- CPU-based distributed TensorFlow Benchmarks (TF) benchmark
  - tf\_cnn\_benchmark tests
- AlexNet model training
  - ImageNet ILSVRC2012 dataset
- Advanced SALaR and XPMEM based designs in MVAPICH-X showed good scalability
- Up to **15%** and **35%** improvements in number of images per second at 448 and 896 processes, respectively.



TensorFlow Images per Second  
(higher is better)

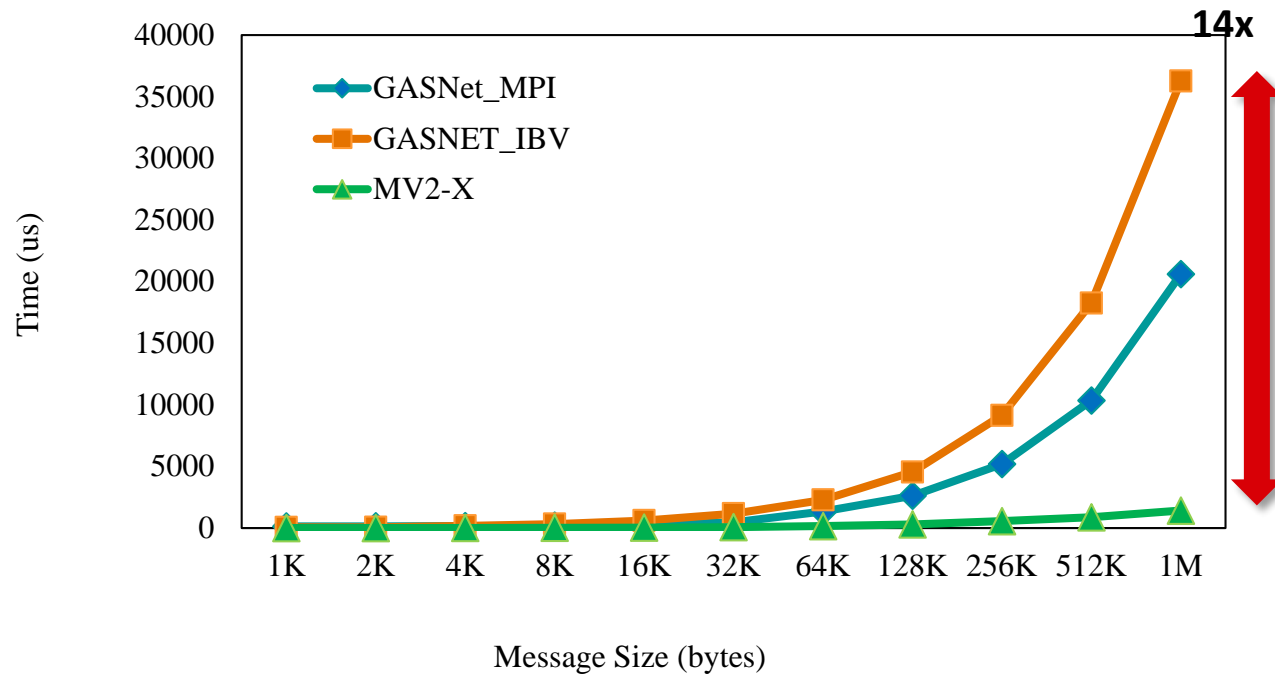
**Will be available in future MVAPICH2-X releases**

SALaR: Scalable and Adaptive Designs for Large Message Reduction Collectives, M. Bayatpour, J. Hashmi, S. Chakraborty, H. Subramoni, P. Kousha, and DK Panda IEEE Cluster 2018, Sep 2018 [Best Paper in Architecture Track]

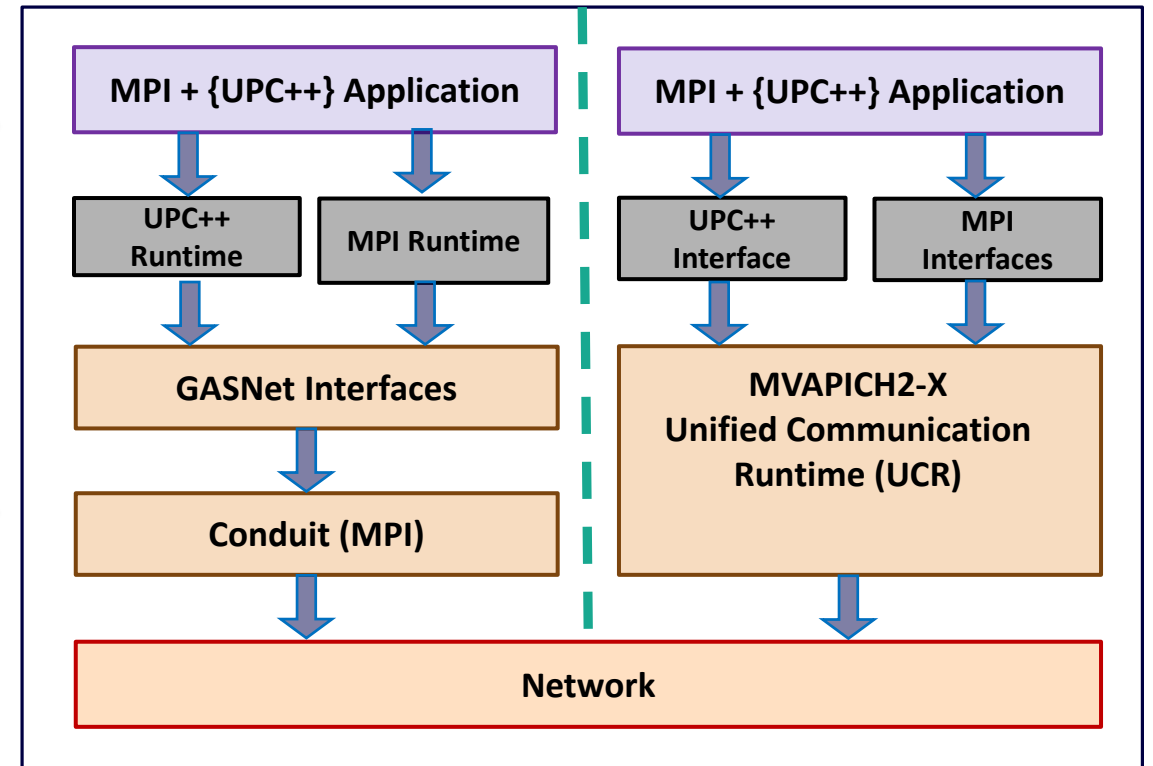
# Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- Understanding Basic Intra-node Communication Mechanisms
  - POSIX SHMEM vs. CMA vs. XPMEM
- CMA-based Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- Asynchronous Progress
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Reduction Collectives
  - Available from MVAPICH2-X 2.3rc1 onwards
- XPMEM-based Non-reduction Collectives
  - Available from MVAPICH2-X 2.3rc2 onwards
- XPMEM-based MPI Derived Datatype Designs
  - Will be available in future MVAPICH2-X releases
- Optimized Collective Communication and Advanced Transport Protocols
  - Available from MVAPICH2-X 2.3rc2 onwards
- **PGAS and Hybrid MPI+PGAS Support**
  - **Available from MVAPICH2-X 2.1.9 onwards**

# UPC++ Support in MVAPICH2-X



Inter-node Broadcast (64 nodes 1:ppn)

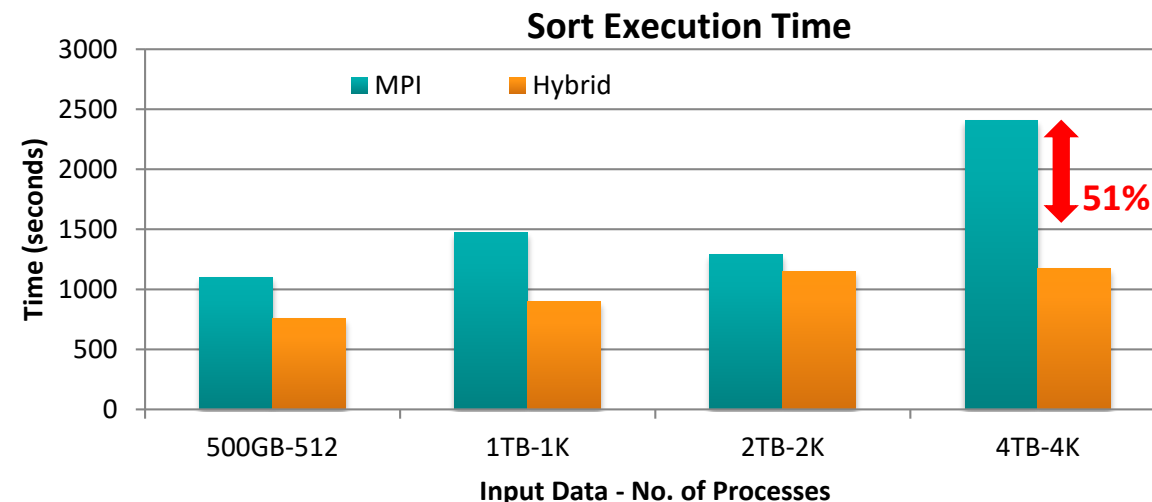
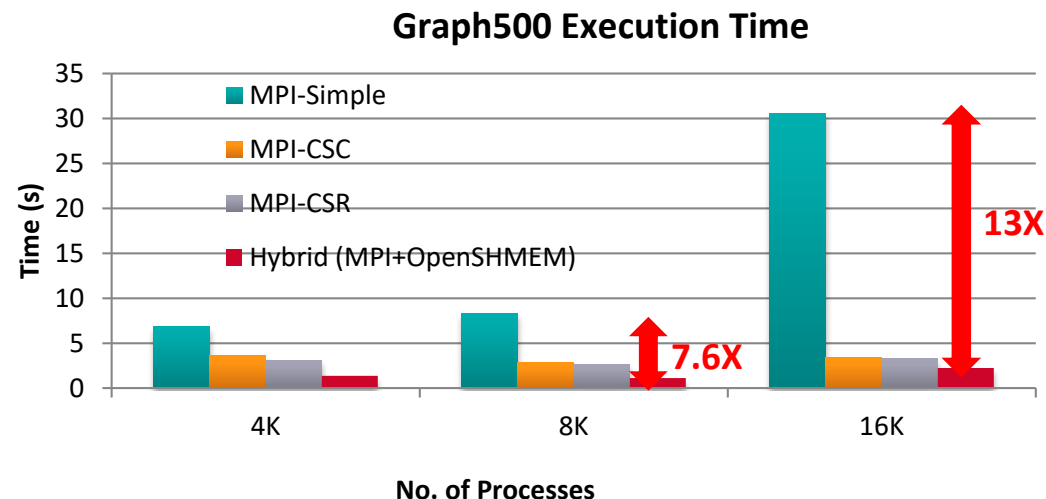


- Full and native support for hybrid MPI + UPC++ applications
- Better performance compared to IBV and MPI conduits
- OSU Micro-benchmarks (OMB) support for UPC++
- Available since MVAPICH2-X (2.2rc1)

More Details in Student Poster  
Presentation



# Application Level Performance with Graph500 and Sort



- Performance of Hybrid (MPI+ OpenSHMEM) Graph500 Design
  - 8,192 processes
    - **2.4X** improvement over MPI-CSR
    - **7.6X** improvement over MPI-Simple
  - 16,384 processes
    - **1.5X** improvement over MPI-CSR
    - **13X** improvement over MPI-Simple
- Performance of Hybrid (MPI+OpenSHMEM) Sort Application
  - 4,096 processes, 4 TB Input Size
    - MPI – **2408 sec**; **0.16 TB/min**
    - Hybrid – **1172 sec**; **0.36 TB/min**
    - **51%** improvement over MPI-design

J. Jose, K. Kandalla, S. Potluri, J. Zhang and D. K. Panda, Optimizing Collective Communication in OpenSHMEM, Int'l Conference on Partitioned Global Address Space Programming Models (PGAS '13), October 2013.

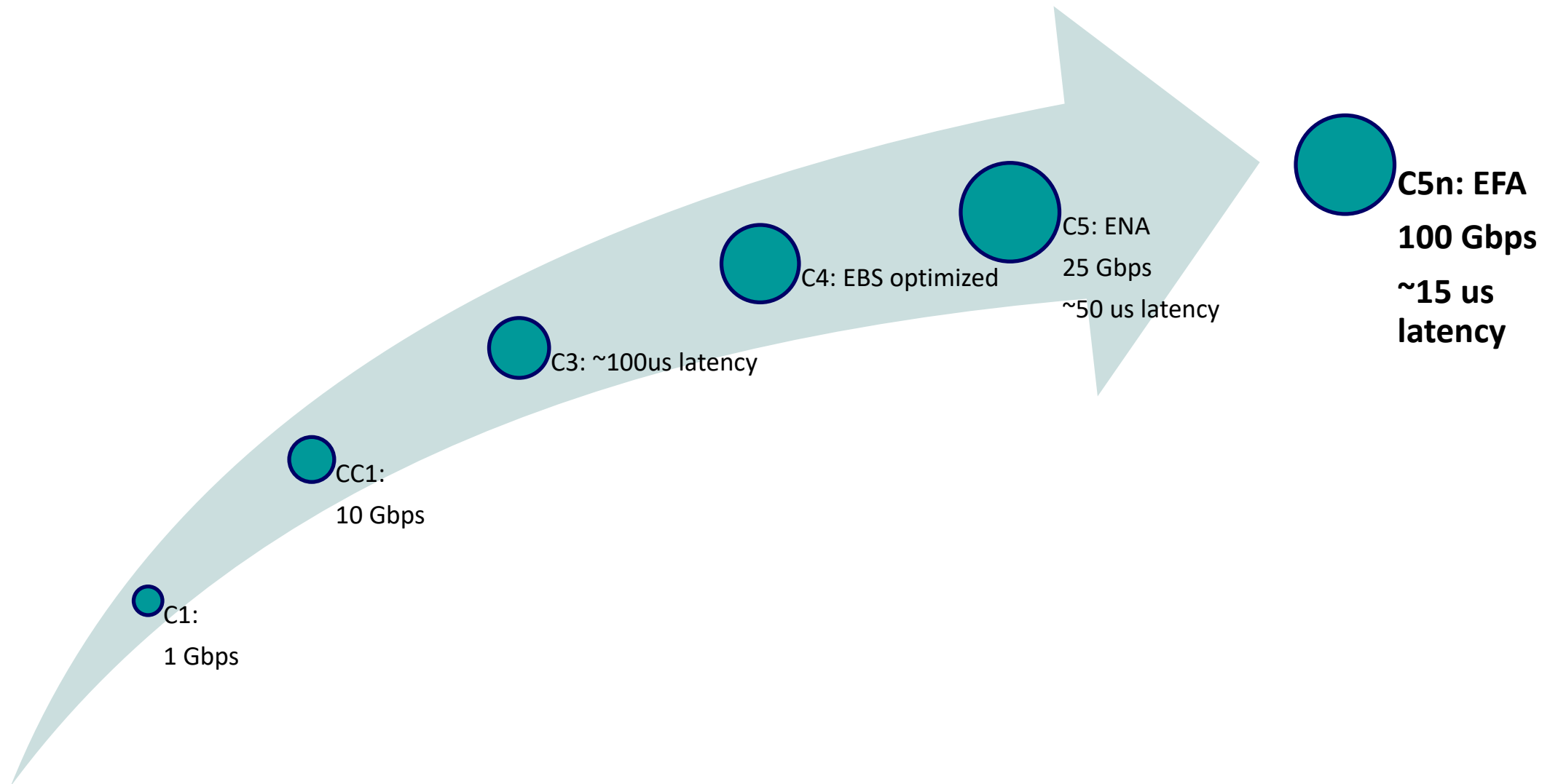
J. Jose, S. Potluri, K. Tomko and D. K. Panda, Designing Scalable Graph500 Benchmark with Hybrid MPI+OpenSHMEM Programming Models, International Supercomputing Conference (ISC'13), June 2013

J. Jose, K. Kandalla, M. Luo and D. K. Panda, Supporting Hybrid MPI and OpenSHMEM over InfiniBand: Design and Performance Evaluation, Int'l Conference on Parallel Processing (ICPP '12), September 2012

# MVAPICH2-X-AWS 2.3

- Released on 08/12/2019
- Major Features and Enhancements
  - Based on MVAPICH2-X 2.3
  - New design based on Amazon EFA adapter's Scalable Reliable Datagram (SRD) transport protocol
  - Support for XPMEM based intra-node communication for point-to-point and collectives
  - Enhanced tuning for point-to-point and collective operations
  - Targeted for AWS instances with Amazon Linux 2 AMI and EFA support
  - Tested with c5n.18xlarge instance

# Evolution of networking on AWS



Deep Dive on OpenMPI and Elastic Fabric Adapter (EFA) - AWS Online Tech Talks, Linda Hedges

# Amazon Elastic Fabric Adapter (EFA)

- Enhanced version of Elastic Network Adapter (ENA)
- Allows OS bypass, up to 100 Gbps bandwidth
- Network aware multi-path routing
- Exposed through libibverbs and libfabric interfaces
- Introduces new Queue-Pair (QP) type
  - Scalable Reliable Datagram (SRD)
  - Also supports Unreliable Datagram (UD)
  - No support for Reliable Connected (RC)

# IB Transport Types and Associated Trade-offs

| Attribute                             |                         | Reliable Connection                                                                                                                                                 | Reliable Datagram                   | Dynamic Connected | Scalable Reliable Datagram | Unreliable Connection              | Unreliable Datagram | Raw Datagram |
|---------------------------------------|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-------------------|----------------------------|------------------------------------|---------------------|--------------|
| Scalability<br>(M processes, N nodes) |                         | M <sup>2</sup> N QPs per HCA                                                                                                                                        | M QPs per HCA                       | M QPs per HCA     | M QPs per HCA              | M <sup>2</sup> N QPs per HCA       | M QPs per HCA       | 1 QP per HCA |
| Reliability                           | Corrupt data detected   | Yes                                                                                                                                                                 |                                     |                   |                            |                                    |                     |              |
|                                       | Data Delivery Guarantee | Data delivered exactly once                                                                                                                                         |                                     |                   |                            | No guarantees                      |                     |              |
|                                       | Data Order Guarantees   | Per connection                                                                                                                                                      | One source to multiple destinations | Per connection    | No                         | Unordered, duplicate data detected | No                  | No           |
|                                       | Data Loss Detected      | Yes                                                                                                                                                                 |                                     |                   |                            |                                    | No                  | No           |
|                                       | Error Recovery          | Errors (retransmissions, alternate path, etc.) handled by transport layer. Client only involved in handling fatal errors (links broken, protection violation, etc.) |                                     |                   |                            | Errors are reported to responder   | None                | None         |

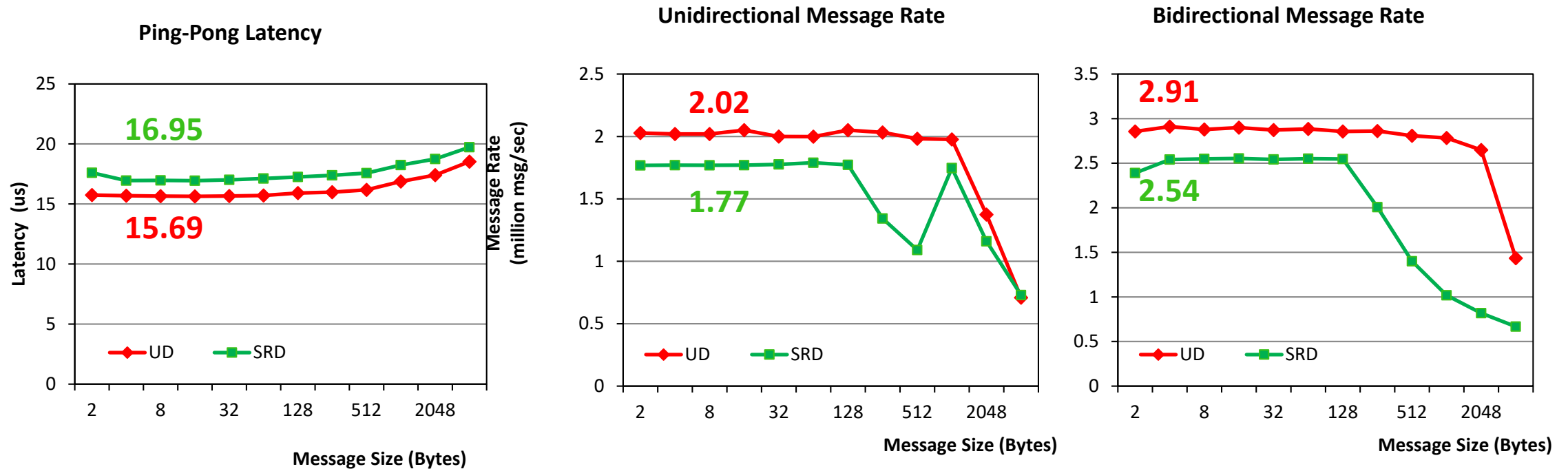
# Scalable Reliable Datagrams (SRD): Features & Limitations

| Feature                   | UD  | SRD |
|---------------------------|-----|-----|
| Send/Recv                 | ✓   | ✓   |
| Send w/ Immediate         | ✗   | ✗   |
| RDMA<br>Read/Write/Atomic | ✗   | ✗   |
| Scatter Gather Lists      | ✓   | ✓   |
| Reliable Delivery         | ✗   | ✓   |
| Ordering                  | ✗   | ✗   |
| Inline Sends              | ✗   | ✗   |
| Global Routing Header     | ✓   | ✗   |
| Max Message Size          | 4KB | 8KB |

- Similar to IB Reliable Datagram
  - No limit on number of outstanding messages per context
- Out of order delivery
  - No head-of-line blocking
  - Bad fit for MPI, can suit other workloads
- Packet spraying over multiple ECMP paths
  - No hotspots
  - Fast and transparent recovery from network failures
- Congestion control designed for large scale
  - Minimize jitter and tail latency

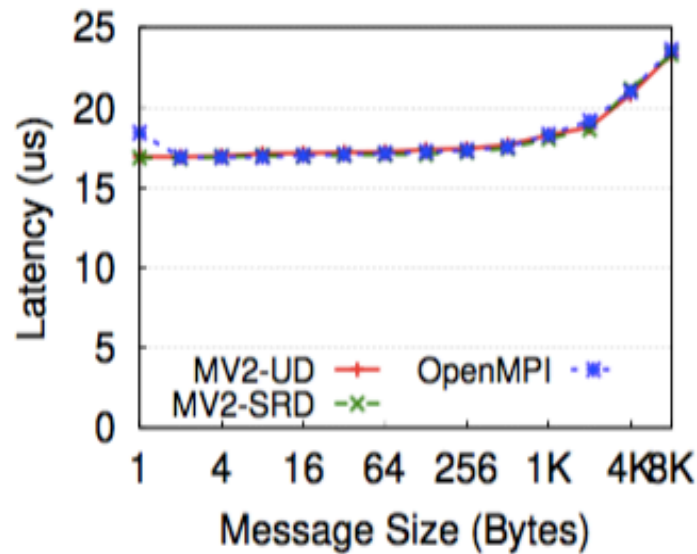
Amazon Elastic Fabric Adapter: Anatomy, Capabilities, and the Road Ahead, Raghu Raja, OpenFabrics Workshop 2019

# Verbs level evaluation of EFA performance

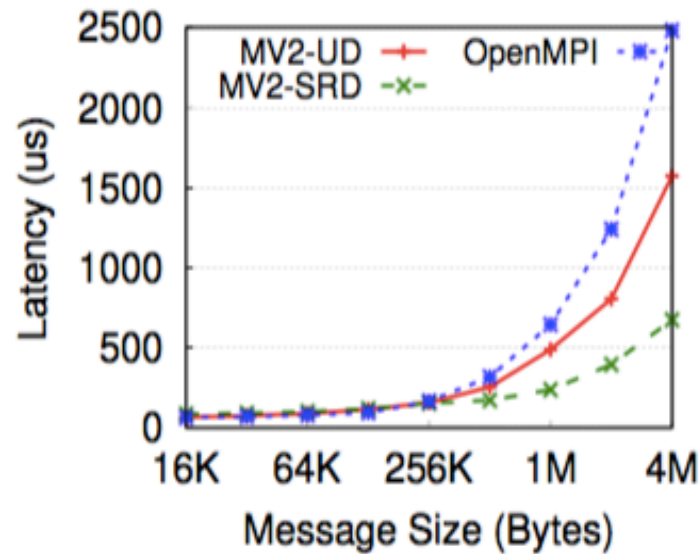


- SRD adds 8-10% overhead compared to UD
- Due to hardware based acks used for reliability
- Instance type: c5n.18xlarge
- CPU: Intel Xeon Platinum 8124M @ 3.00GHz

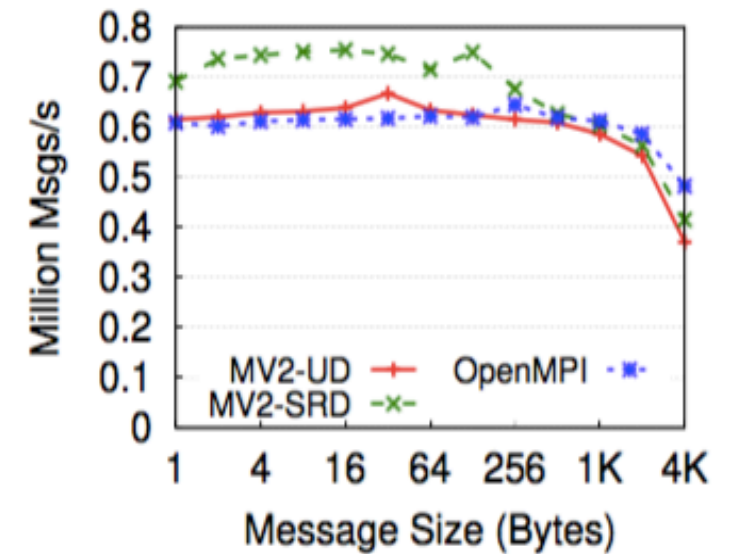
# Point-to-Point Performance



(a) Small Message Latency



(b) Large Message Latency

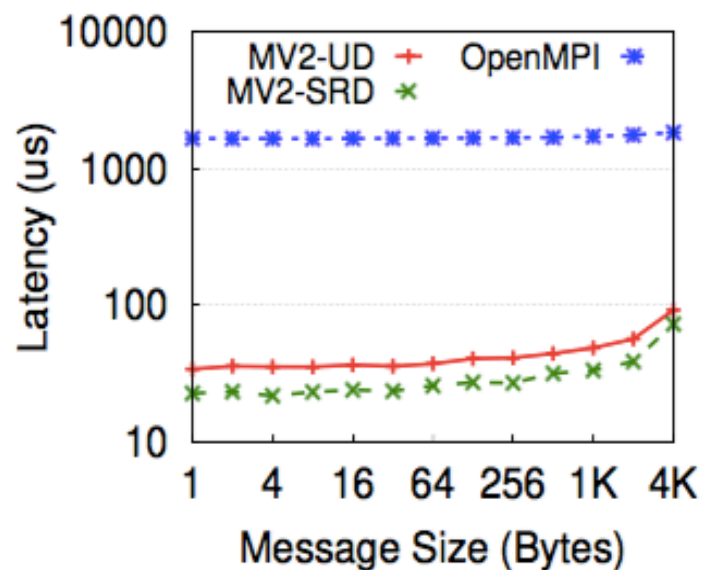


(c) Unidirectional Message Rate

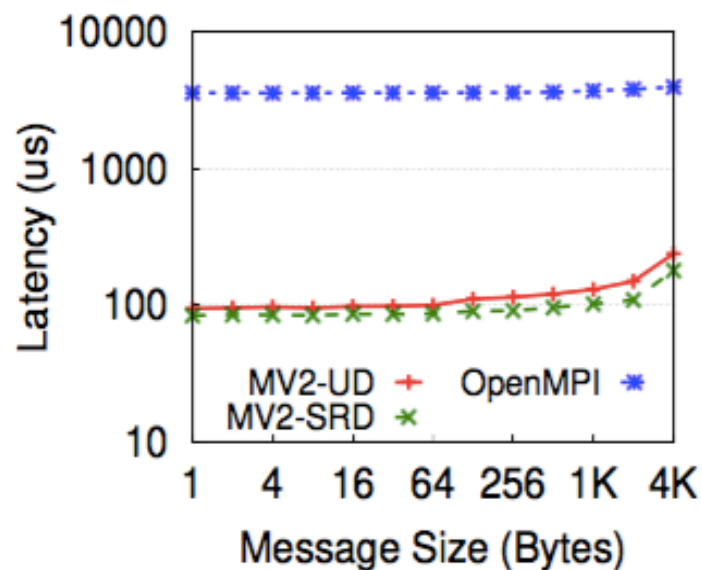
- Both UD and SRD shows similar latency for small messages
- SRD shows higher message rate due to lack of software reliability overhead
- SRD is faster for large messages due to larger MTU size



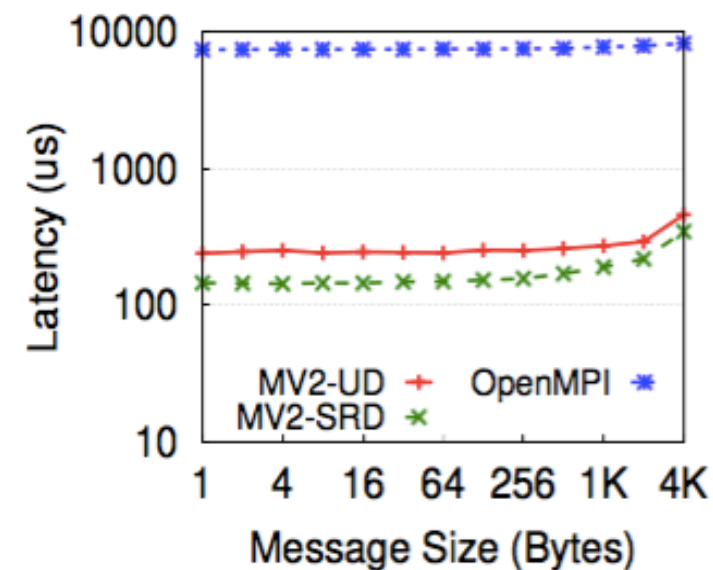
# Collective Performance: MPI Scatterv



(a) 2 Nodes, 72 Processes



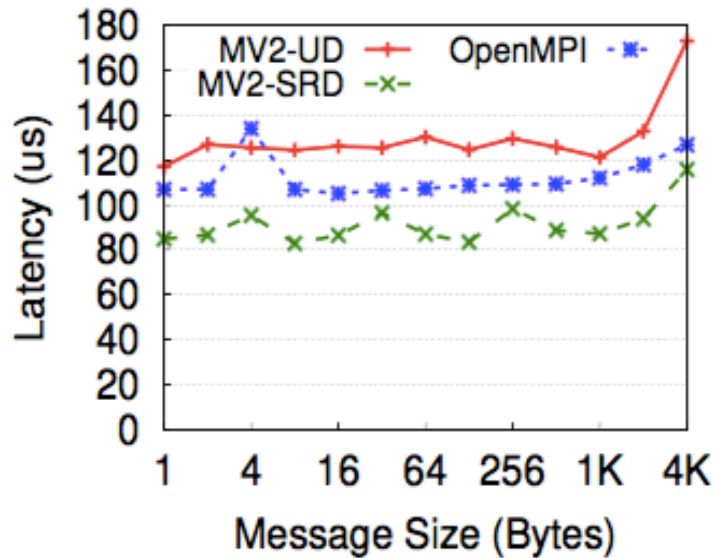
(b) 4 Nodes, 144 Processes



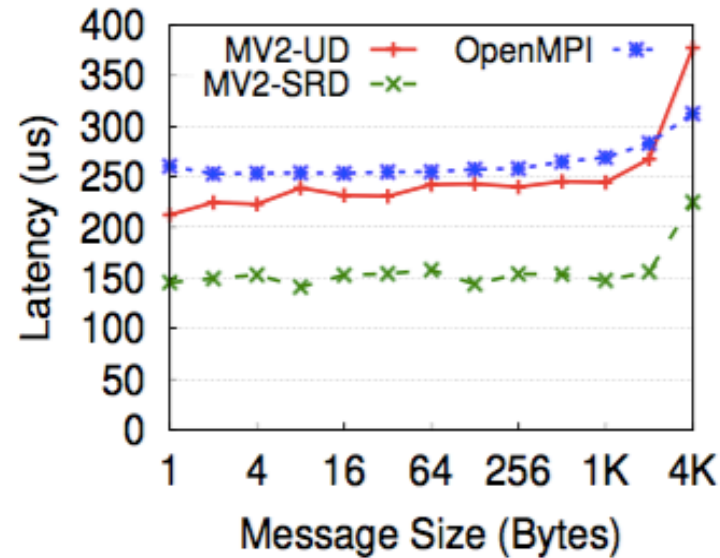
(c) 8 Nodes, 288 Processes

- SRD shows up to 60% improvement over UD
- Non-roots do not need to send back explicit acknowledgments
- Root does not need to buffer messages until ack is received

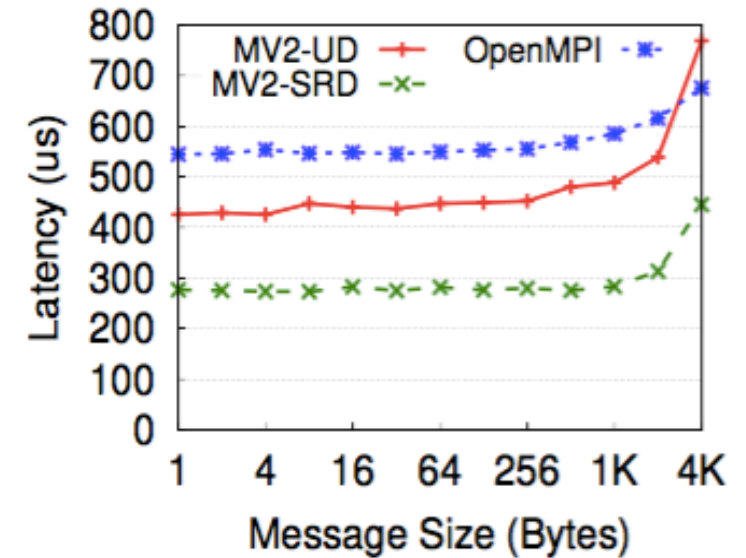
# Collective Performance: MPI Gatherv



(a) 2 Nodes, 72 Processes



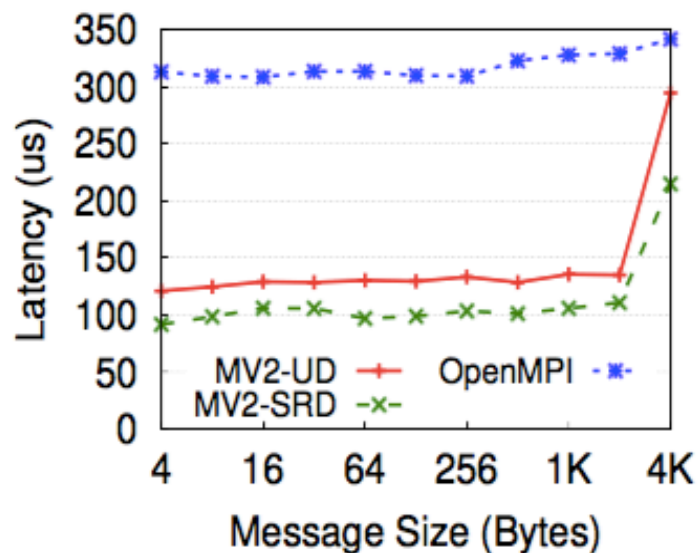
(b) 4 Nodes, 144 Processes



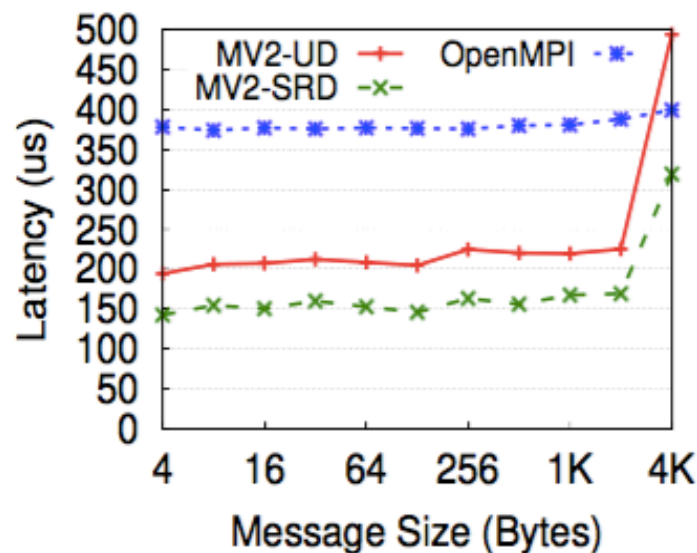
(c) 8 Nodes, 288 Processes

- Up to 33% improvement with SRD compared to UD
- Root does not need to send explicit acks to non-root processes
- Non-roots can exit as soon as the message is sent (no need to wait for acks)

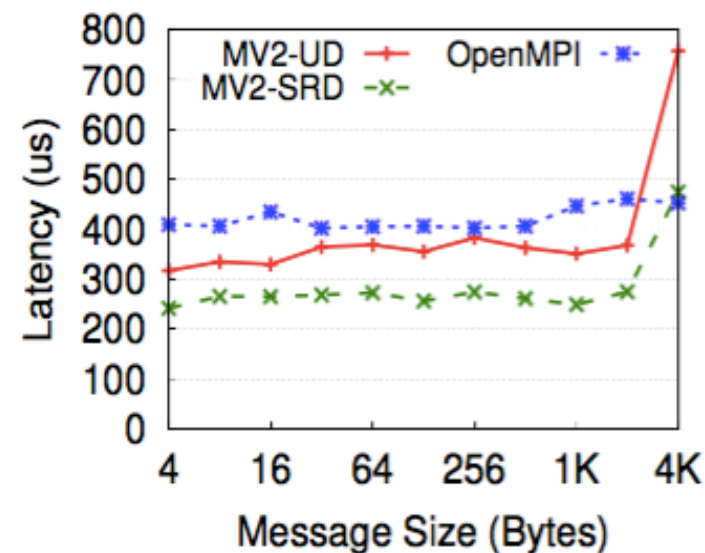
# Collective Performance: MPI Allreduce



(a) 2 Nodes, 72 Processes



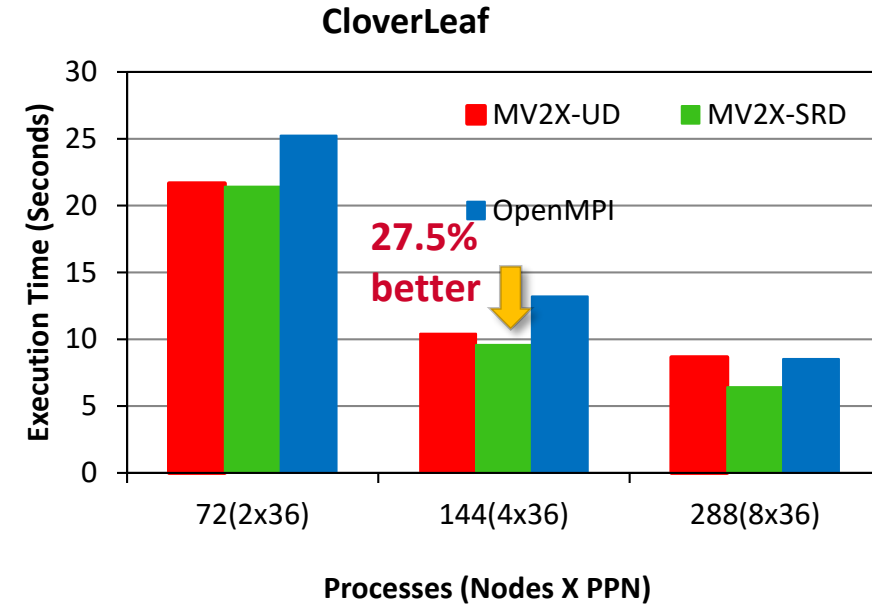
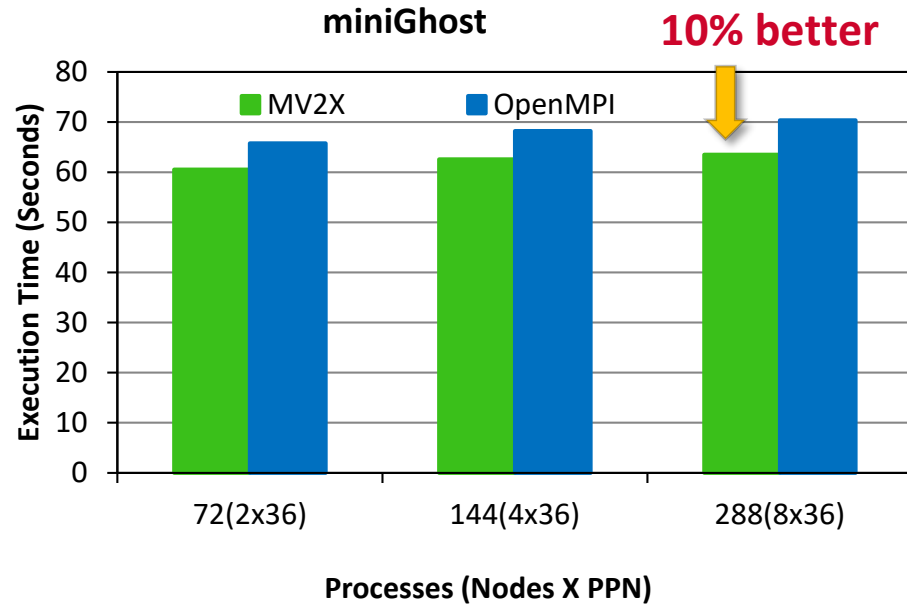
(b) 4 Nodes, 144 Processes



(c) 8 Nodes, 288 Processes

- Up to 18% improvement with SRD compared to UD
- Bidirectional communication pattern allows piggybacking of acks
- Modest improvement compared to asymmetric communication patterns

# Application Performance



- Up to 10% performance improvement for MiniGhost on 8 nodes
- Up to 27% better performance with CloverLeaf on 8 nodes

# MVAPICH2 Software Family

| Requirements                                                                                                                    | Library             |
|---------------------------------------------------------------------------------------------------------------------------------|---------------------|
| MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                                    | MVAPICH2            |
| Optimized Support for Microsoft Azure Platform with InfiniBand                                                                  | MVAPICH2-Azure      |
| Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis), | MVAPICH2-X          |
| Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric Adapter (EFA)                                      | MVAPICH2-X-AWS      |
| <b>Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications</b>                               | <b>MVAPICH2-GDR</b> |
| Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                       | MVAPICH2-EA         |
| MPI Energy Monitoring Tool                                                                                                      | OEMT                |
| InfiniBand Network Analysis and Monitoring                                                                                      | OSU INAM            |
| Microbenchmarks for Measuring MPI and PGAS Performance                                                                          | OMB                 |

## MPI + CUDA - Naive

- Data movement in applications with standard MPI and CUDA interfaces

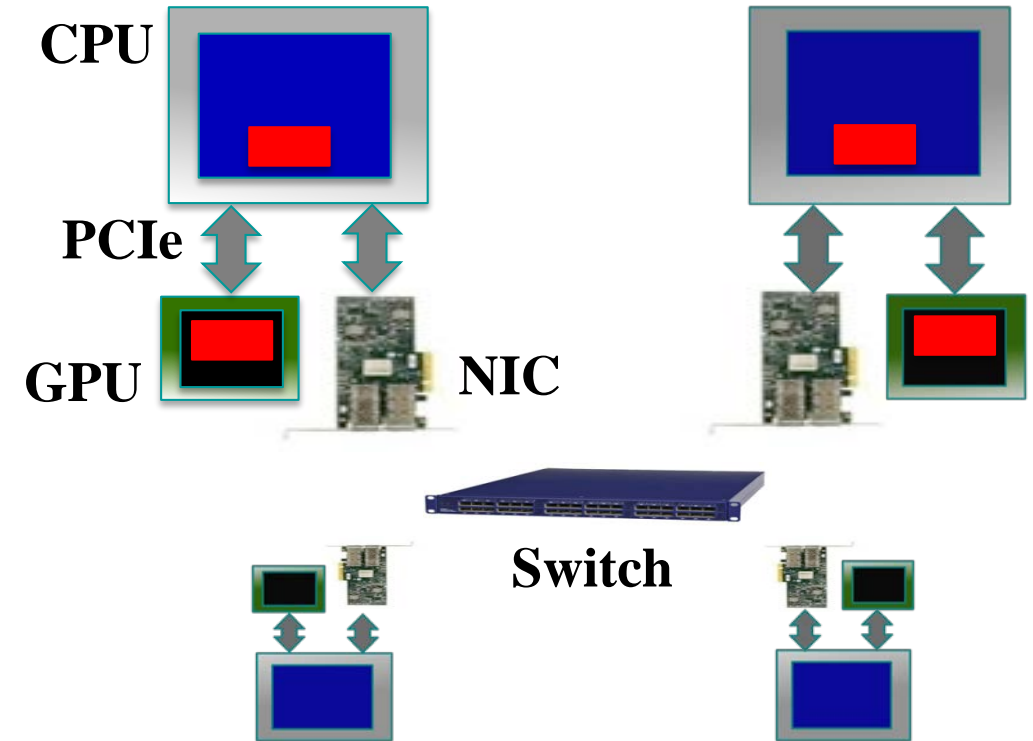
### At Sender:

```
cudaMemcpy(s_hostbuf, s_devbuf, . . .);
MPI_Send(s_hostbuf, size, . . .);
```

### At Receiver:

```
MPI_Recv(r_hostbuf, size, . . .);
cudaMemcpy(r_devbuf, r_hostbuf, . . .);
```

*High Productivity and Low Performance*



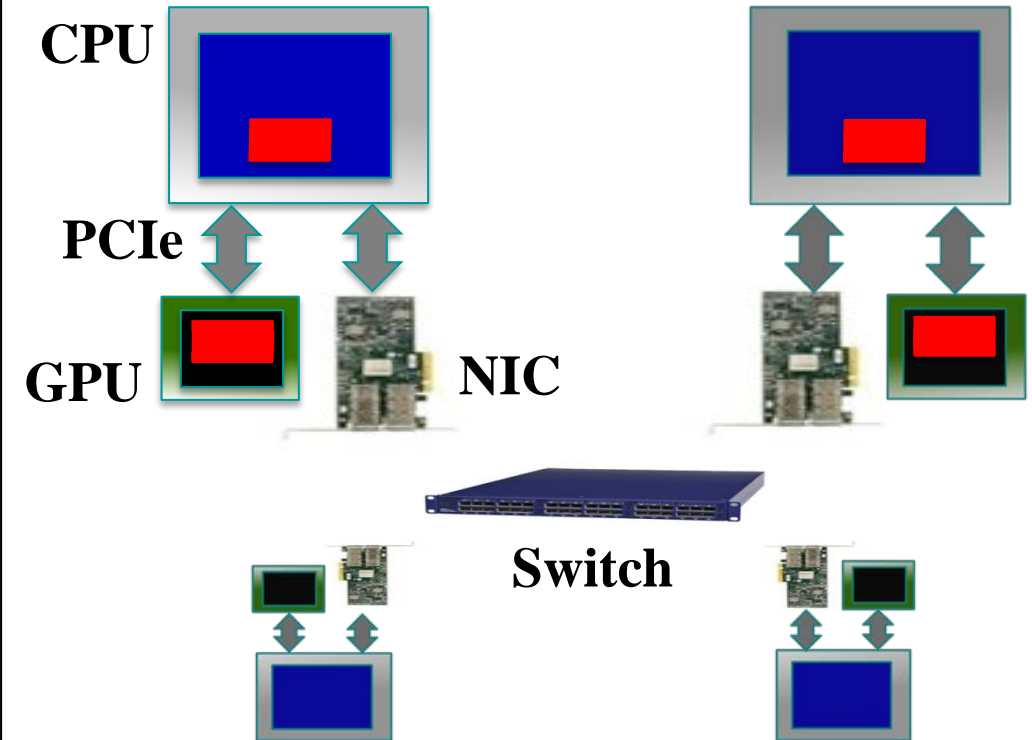
# MPI + CUDA - Advanced

- Pipelining at user level with non-blocking MPI and CUDA interfaces

## At Sender:

```
for (j = 0; j < pipeline_len; j++)
 cudaMemcpyAsync(s_hostbuf + j * blk, s_devbuf + j *
 blk_sz, ...);
for (j = 0; j < pipeline_len; j++) {
 while (result != cudaSuccess) {
 result = cudaStreamQuery(...);
 if(j > 0) MPI_Test(...);
 }
 MPI_Isend(s_hostbuf + j * block_sz, blk_sz . . .);
}
MPI_Waitall();
```

<<Similar at receiver>>



*Low Productivity and High Performance*

# GPU-Aware (CUDA-Aware) MPI Library: MVAPICH2-GPU

- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing ( $\geq$  CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

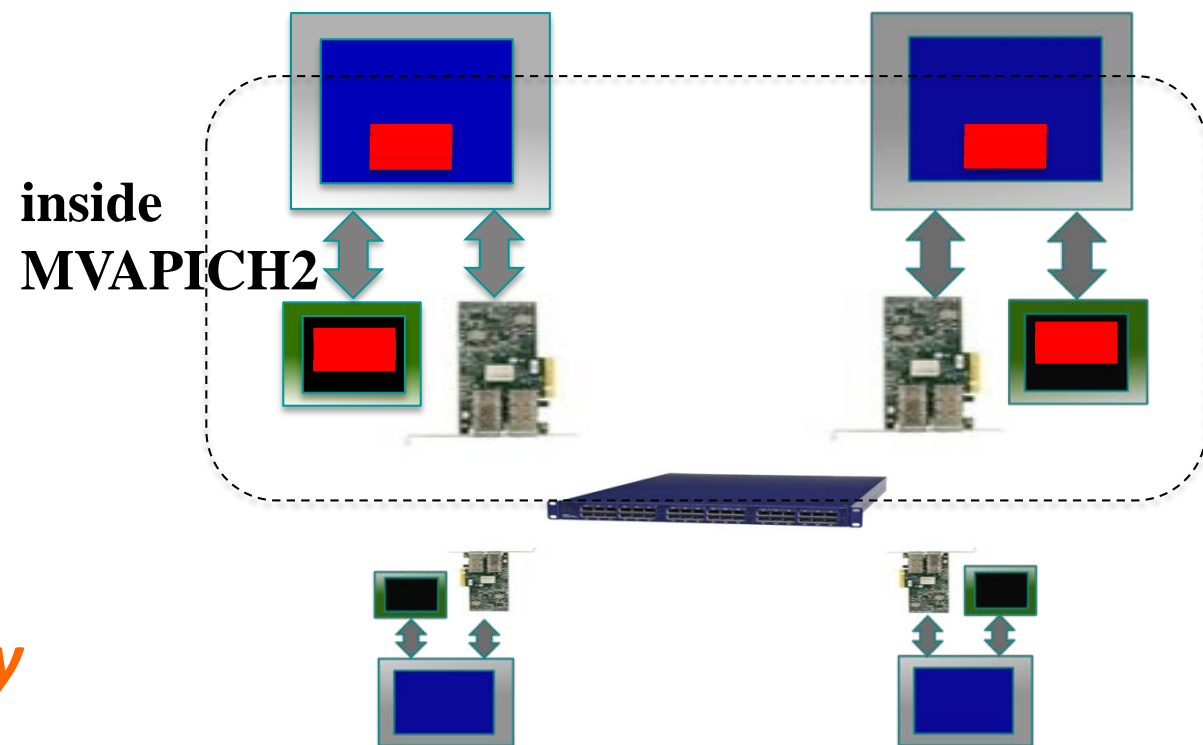
## At Sender:

```
MPI_Send(s_devbuf, size, ...);
```

## At Receiver:

```
MPI_Recv(r_devbuf, size, ...);
```

*High Performance and High Productivity*





## CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.3.2 Releases

- Support for MPI communication from NVIDIA GPU device memory
- High performance RDMA-based inter-node point-to-point communication (GPU-GPU, GPU-Host and Host-GPU)
- High performance intra-node point-to-point communication for multi-GPU adapters/node (GPU-GPU, GPU-Host and Host-GPU)
- Taking advantage of CUDA IPC (available since CUDA 4.1) in intra-node communication for multiple GPU adapters/node
- Optimized and tuned collectives for GPU device buffers
- MPI datatype support for point-to-point and collective communication from GPU device buffers
- Unified memory

# MVAPICH2-GDR: Pre-requisites for OpenPOWER & x86 Systems

- MVAPICH2-GDR 2.3.2 requires the following software to be installed on your system:
  1. [Mellanox OFED 3.2 and later](#)
  2. [NVIDIA Driver 367.48 or later](#)
  3. [NVIDIA CUDA Toolkit 7.5 and later](#)
  4. [NVIDIA Peer Memory \(nv\\_peer\\_mem\) module to enable GPUDirect RDMA \(GDR\) support](#)
- Strongly Recommended for Best Performance
  5. GDRCOPY Library by NVIDIA: <https://github.com/NVIDIA/gdrCOPY>
- Comprehensive Instructions can be seen from the MVAPICH2-GDR User Guide:
  - <http://mvapich.cse.ohio-state.edu/userguide/gdr/>

# MVAPICH2-GDR: Download and Setup on OpenPOWER & x86 Systems

- Simple Installation steps for both systems
- Pick the right MVAPICH2-GDR RPM from Downloads page:
  - <http://mvapich.cse.ohio-state.edu/downloads/>
  - e.g. [http://mvapich.cse.ohio-state.edu/download/mvapich/gdr/2.3/mofed4.5/mvapich2-gdr-mcast.cuda10.0.mofed4.5.gnu4.8.5-2.3-1.el7.x86\\_64.rpm](http://mvapich.cse.ohio-state.edu/download/mvapich/gdr/2.3/mofed4.5/mvapich2-gdr-mcast.cuda10.0.mofed4.5.gnu4.8.5-2.3-1.el7.x86_64.rpm) (== <mv2-gdr-rpm-name>.rpm)

```
$ wget http://mvapich.cse.ohio-state.edu/download/mvapich/gdr/2.3/<mv2-gdr-rpm-name>.rpm
```

## Root Users:

```
$ rpm -Uvh --nodeps <mv2-gdr-rpm-name>.rpm
```

## Non-Root Users:

```
$ rpm2cpio <mv2-gdr-rpm-name>.rpm | cpio -id
```

- Contact MVAPICH help list with any questions related to the package  
[mvapich-help@cse.ohio-state.edu](mailto:mvapich-help@cse.ohio-state.edu)

# ROCE and Optimized Collectives Support

- RoCE V1 and V2 support
- RDMA\_CM connection support
- CUDA-Aware Collective Tuning
  - Point-point Tuning (available since MVAPICH2-GDR 2.0)
    - Tuned thresholds for the different communication patterns and features
    - Depending on the system configuration (CPU, HCA and GPU models)
  - Tuning Framework for GPU based collectives
    - Select the best algorithm depending on message size, system size and system configuration
    - Support for Bcast and Gather operations for different GDR-enabled systems
- Available since **MVAPICH2-GDR 2.2RC1** release

# MVAPICH2-GDR 2.3.2

- Released on 08/08/2019
- Major Features and Enhancements
  - Based on MVAPICH2 2.3.1
  - Support for CUDA 10.1
  - Support for PGI 19.x
  - Enhanced intra-node and inter-node point-to-point performance
  - Enhanced MPI\_Allreduce performance for DGX-2 system
  - Enhanced GPU communication support in MPI\_THREAD\_MULTIPLE mode
  - Enhanced performance of datatype support for GPU-resident data
    - Zero-copy transfer when P2P access is available between GPUs through NVLink/PCIe
  - Enhanced GPU-based point-to-point and collective tuning
    - OpenPOWER systems such as ORNL Summit and LLNL Sierra ABCI system @AIST, Owens and Pitzer systems @Ohio Supercomputer Center
  - Scaled Allreduce to 24,576 Volta GPUs on Summit
  - Enhanced intra-node and inter-node point-to-point performance for DGX-2 and IBM POWER8 and IBM POWER9 systems
  - Enhanced Allreduce performance for DGX-2 and IBM POWER8/POWER9 systems
  - Enhanced small message performance for CUDA-Aware MPI\_Put and MPI\_Get
  - Flexible support for running TensorFlow (Horovod) jobs

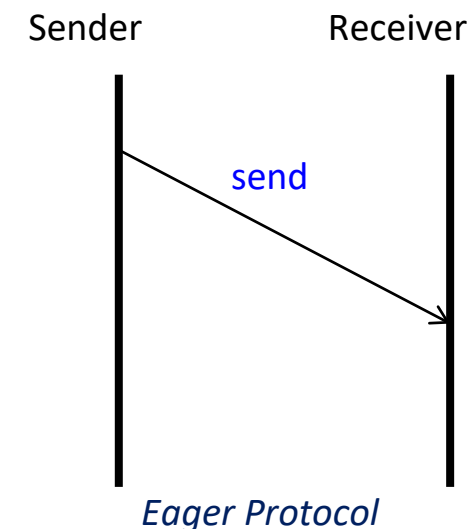
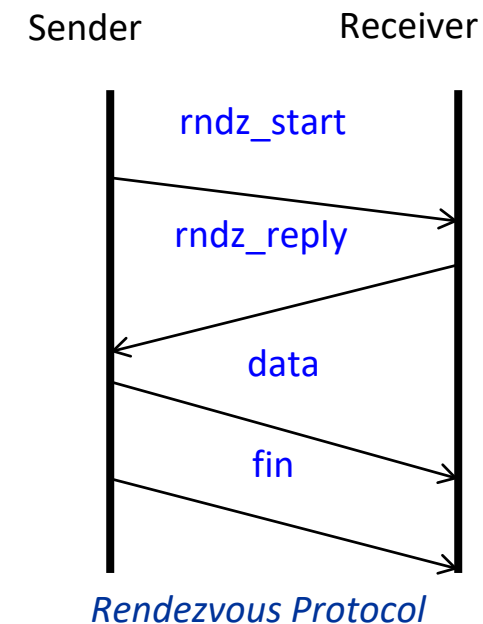
# Overview of MVAPICH2-GDR Features

- **Support for Efficient Small Message Communication with GPUDirect RDMA**
- Multi-stream Communication for IPC
- CMA- based Intra-node Host-to-Host Communication Support
- MPI Datatype Support
- Support for Managed Memory
- Optimized Support for Deep Learning

# Enhanced MPI Design with GPUDirect RDMA

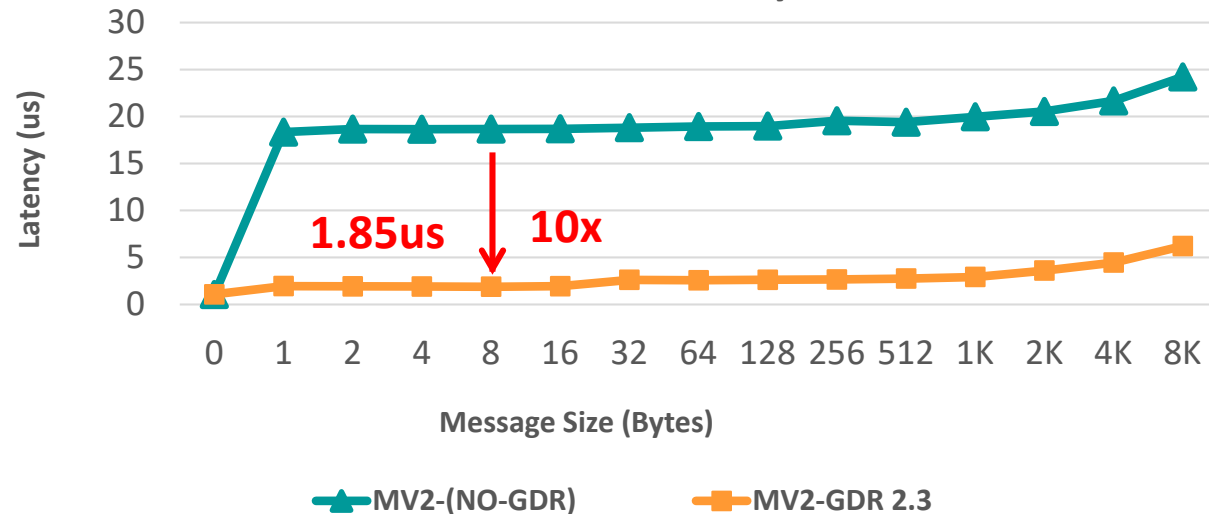
- Current MPI design using GPUDirect RDMA uses Rendezvous protocol
  - Has higher latency for small messages
- Can eager protocol be supported to improve performance for small messages?
- Two schemes proposed and used
  - **Loopback** (using network adapter to copy data)
  - **Fastcopy/GDRCOPY** (using CPU to copy data)

R. Shi, S. Potluri, K. Hamidouche M. Li, J. Perkins D. Rossetti and D. K. Panda, Designing Efficient Small Message Transfer Mechanism for Inter-node MPI Communication on InfiniBand GPU Clusters IEEE International Conference on High Performance Computing (HiPC'2014)

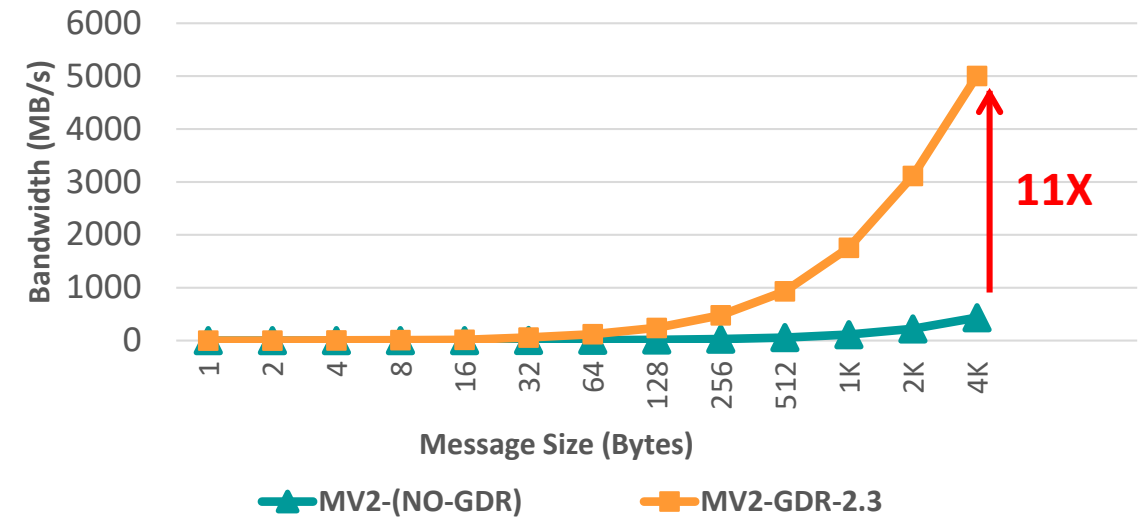


# Optimized MVAPICH2-GDR Design

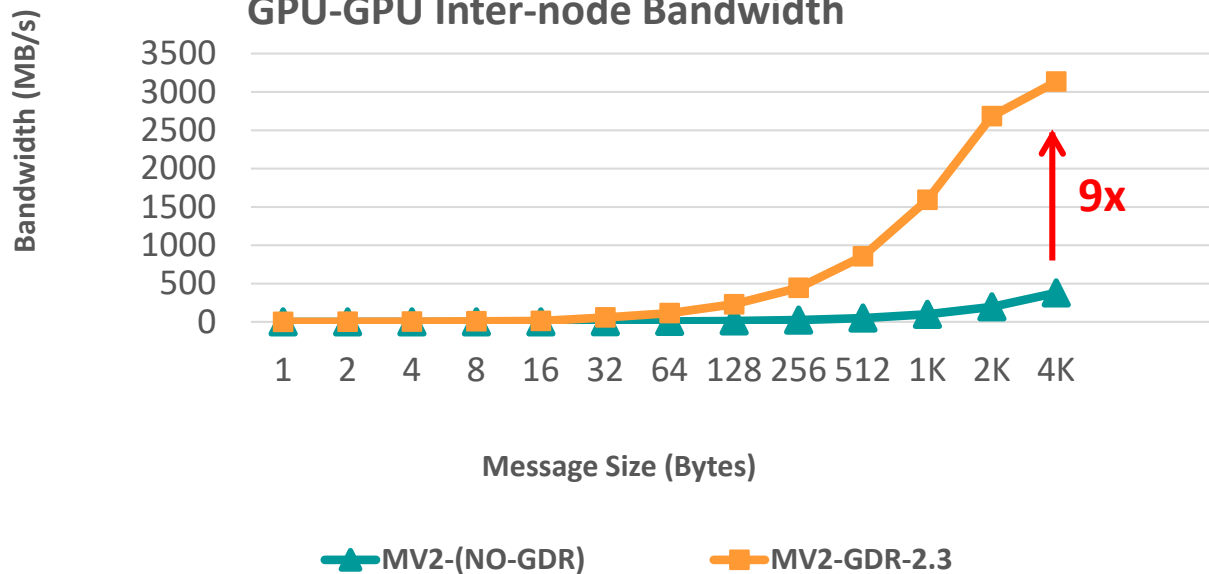
## GPU-GPU Inter-node Latency



## GPU-GPU Inter-node Bi-Bandwidth



## GPU-GPU Inter-node Bandwidth

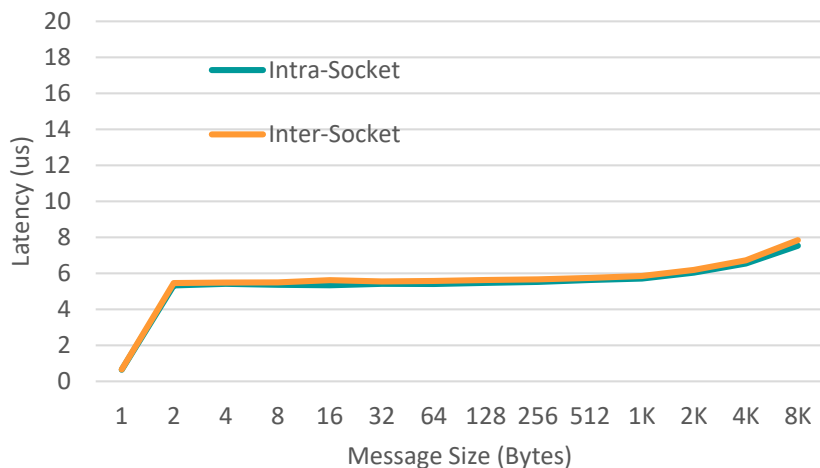


**MVAPICH2-GDR-2.3**  
**Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores**  
**NVIDIA Volta V100 GPU**  
**Mellanox Connect-X4 EDR HCA**  
**CUDA 9.0**  
**Mellanox OFED 4.0 with GPU-Direct-RDMA**



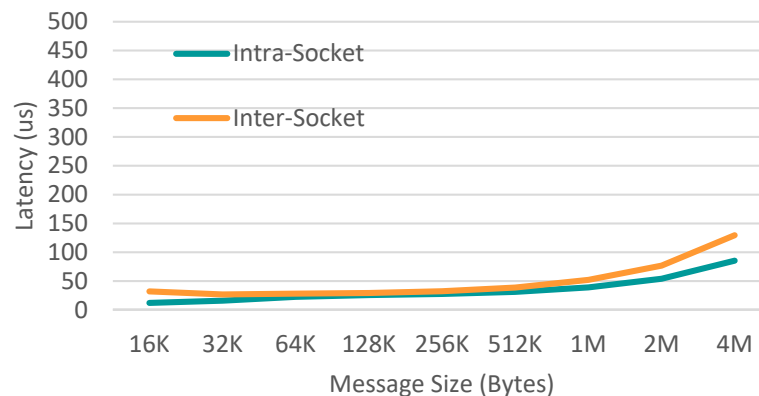
# Device-to-Device Performance on OpenPOWER (NVLink2 + Volta)

## INTRA-NODE LATENCY (SMALL)

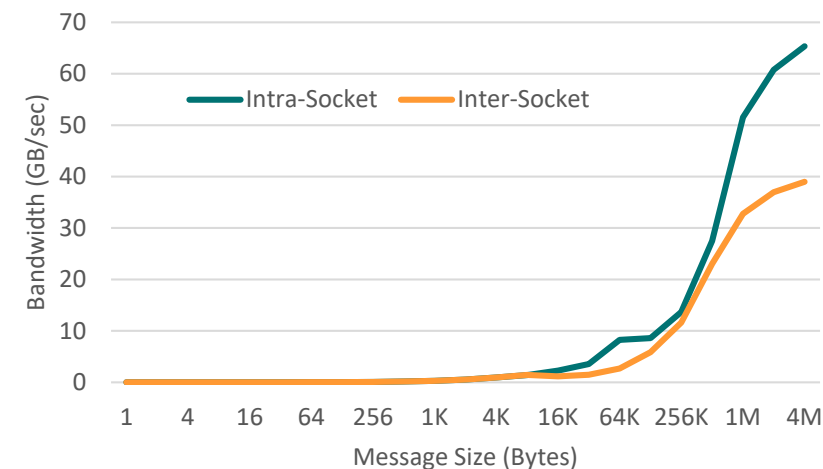


**Intra-node Latency: 5.36 us (without GDRCopy)**

## INTRA-NODE LATENCY (LARGE)

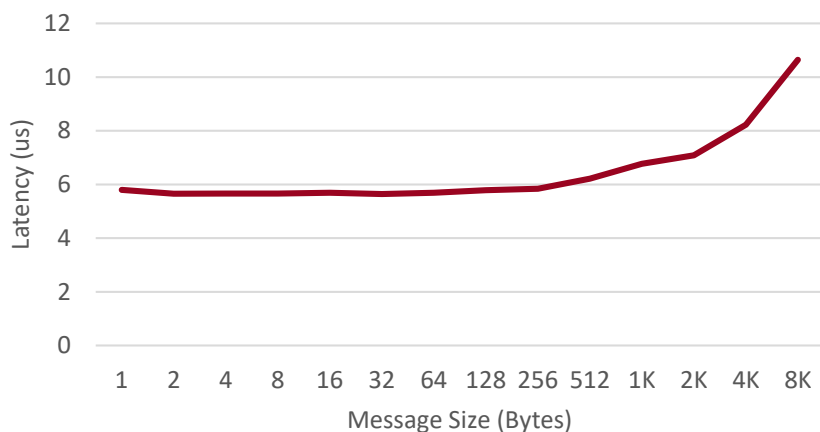


## INTRA-NODE BANDWIDTH



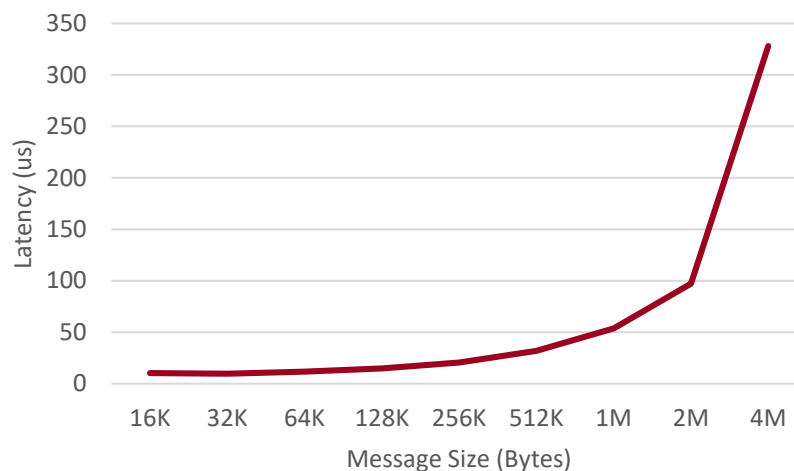
**Intra-node Bandwidth: 70.4 GB/sec for 128MB (via NVLINK2)**

## INTER-NODE LATENCY (SMALL)



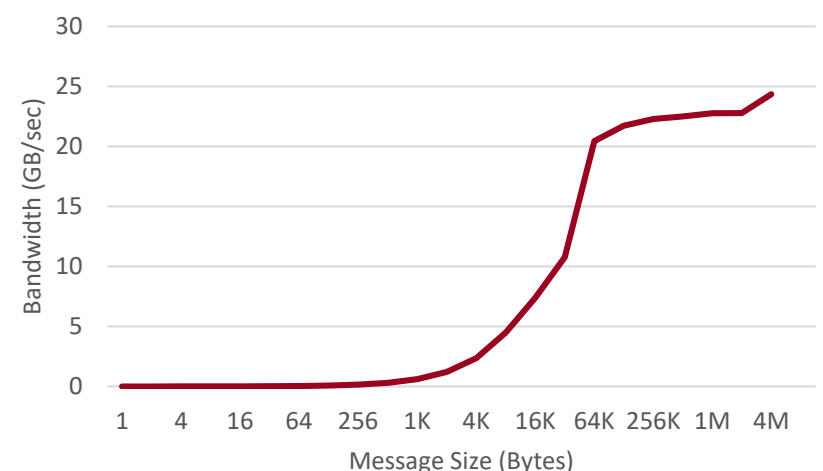
**Inter-node Latency: 5.66 us (without GDRCopy)**

## INTER-NODE LATENCY (LARGE)



**Available since MVAPICH2-GDR 2.3a**

## INTER-NODE BANDWIDTH



**Inter-node Bandwidth: 23.7 GB/sec (2 port EDR)**

**Platform: OpenPOWER (POWER9-ppc64le) nodes equipped with a dual-socket CPU, 4 Volta V100 GPUs, and 2port EDR InfiniBand Interconnect**

# Tuning GDRCOPY Designs in MVAPICH2-GDR

| Parameter                           | Significance                                                   | Default        | Notes                                                                                  |
|-------------------------------------|----------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------|
| MV2_USE_GDRCOPY                     | • Enable / Disable GDRCOPY-based designs                       | 1<br>(Enabled) | • Always enable                                                                        |
| MV2_GDRCOPY_LIMIT                   | • Controls messages size until which GDRCOPY is used           | 8 KByte        | • Tune for your system<br>• GPU type, host architecture. Impacts the eager performance |
| MV2_GPUDIRECT_GDRCOPY_LIB           | • Path to the GDRCOPY library                                  | Unset          | • Always set                                                                           |
| MV2_USE_GPUDIRECT_D2H_GDRCOPY_LIMIT | • Controls messages size until which GDRCOPY is used at sender | 16Bytes        | • Tune for your systems<br>• CPU and GPU type                                          |

- Refer to **Tuning and Usage Parameters** section of MVAPICH2-GDR user guide for more information
- [http://mvapich.cse.ohio-state.edu/userguide/gdr/#\\_tuning\\_and\\_usage\\_parameters](http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters)

# Tuning Loopback Designs in MVAPICH2-GDR

| Parameter                    | Significance                                          | Default        | Notes                                                                                                                          |
|------------------------------|-------------------------------------------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------|
| MV2_USE_GPUDIRECT_LOOPBACK   | • Enable / Disable LOOPBACK-based designs             | 1<br>(Enabled) | • Always enable                                                                                                                |
| MV2_GPUDIRECT_LOOPBACK_LIMIT | • Controls messages size until which LOOPBACK is used | 8 KByte        | • Tune for your system<br>• GPU type, host architecture and HCA. Impacts the eager performance<br>• Sensitive to the P2P issue |

- Refer to **Tuning and Usage Parameters** section of MVAPICH2-GDR user guide for more information
- [http://mvapich.cse.ohio-state.edu/userguide/gdr/#\\_tuning\\_and\\_usage\\_parameters](http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters)

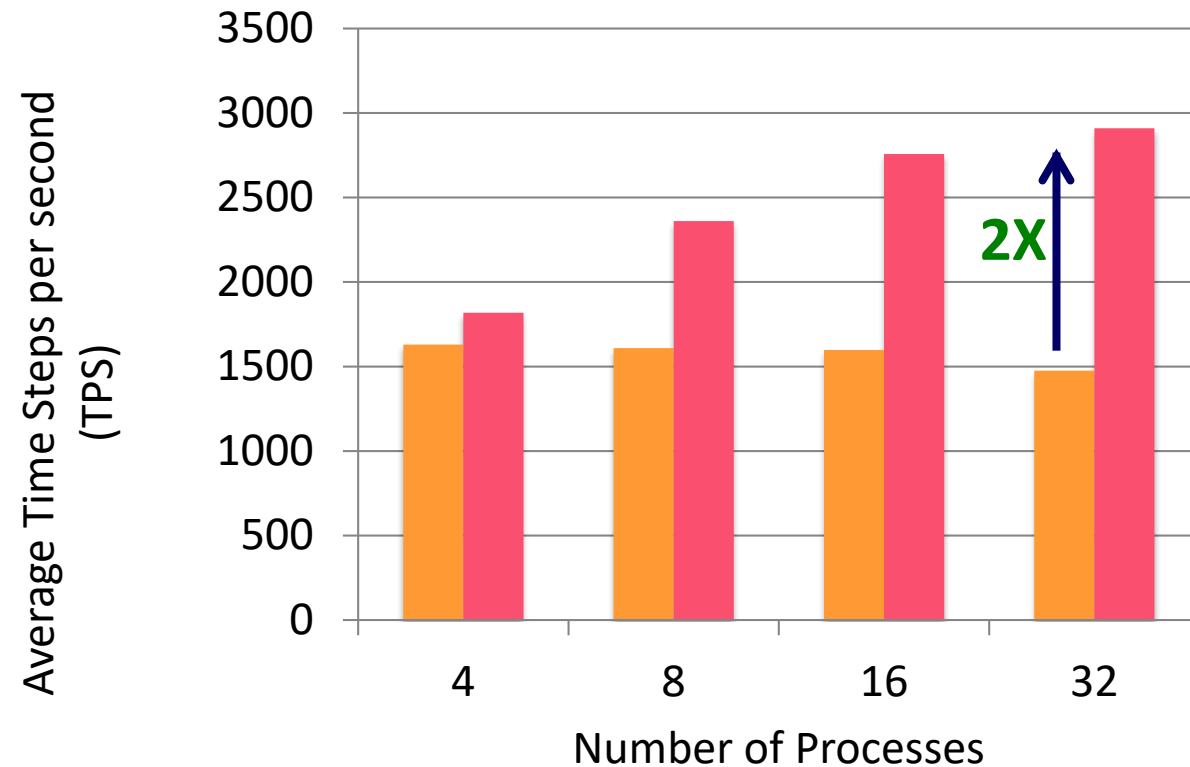
## Tuning GPUDirect RDMA (GDR) Designs in MVAPICH2-GDR

| Parameter                       | Significance                                                                          | Default        | Notes                                                                                                                                  |
|---------------------------------|---------------------------------------------------------------------------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------|
| MV2_USE_GPUDIRECT               | • Enable / Disable GDR-based designs                                                  | 1<br>(Enabled) | • Always enable                                                                                                                        |
| MV2_GPUDIRECT_LIMIT             | • Controls messages size until which GPUDirect RDMA is used                           | 8 KByte        | • Tune for your system<br>• GPU type, host architecture and<br>CUDA version: impact pipelining overheads and P2P bandwidth bottlenecks |
| MV2_USE_GPUDIRECT_RECEIVE_LIMIT | • Controls messages size until which 1 hop design is used (GDR Write at the receiver) | 256KBytes      | • Tune for your system<br>• GPU type, HCA type and configuration                                                                       |

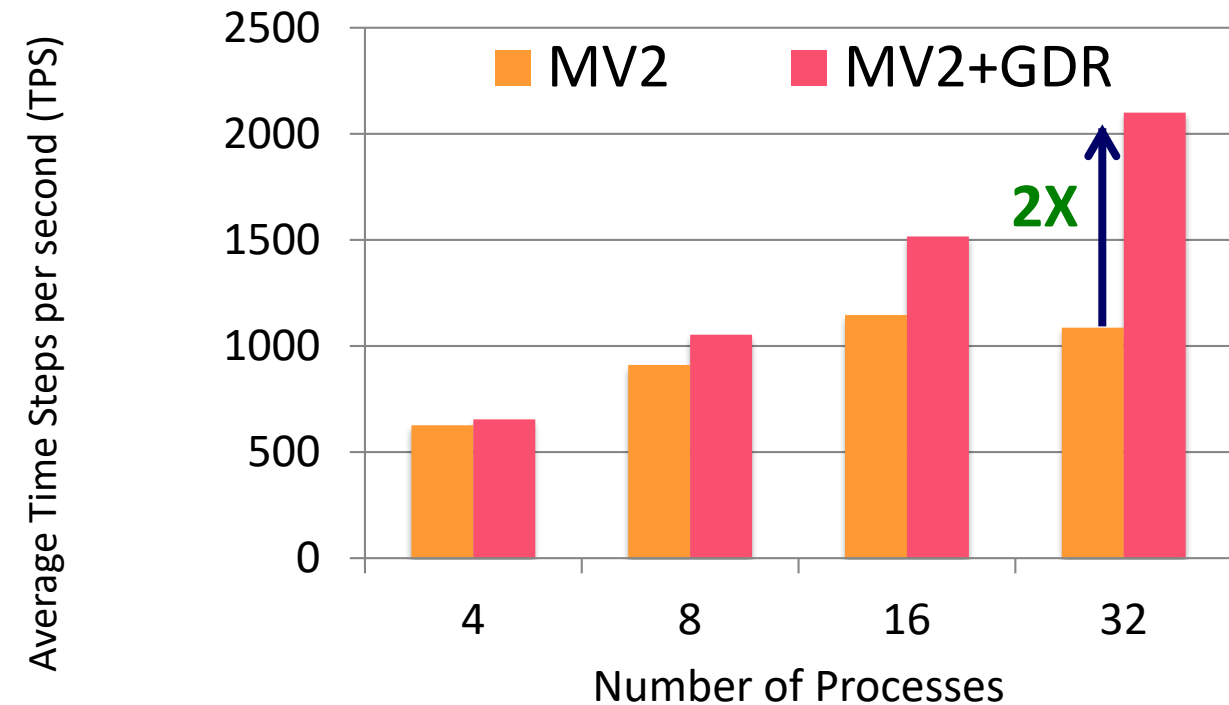
- Refer to **Tuning and Usage Parameters** section of MVAPICH2-GDR user guide for more information
- [http://mvapich.cse.ohio-state.edu/userguide/gdr/#\\_tuning\\_and\\_usage\\_parameters](http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters)

# Application-Level Evaluation (HOOMD-blue)

## 64K Particles



## 256K Particles



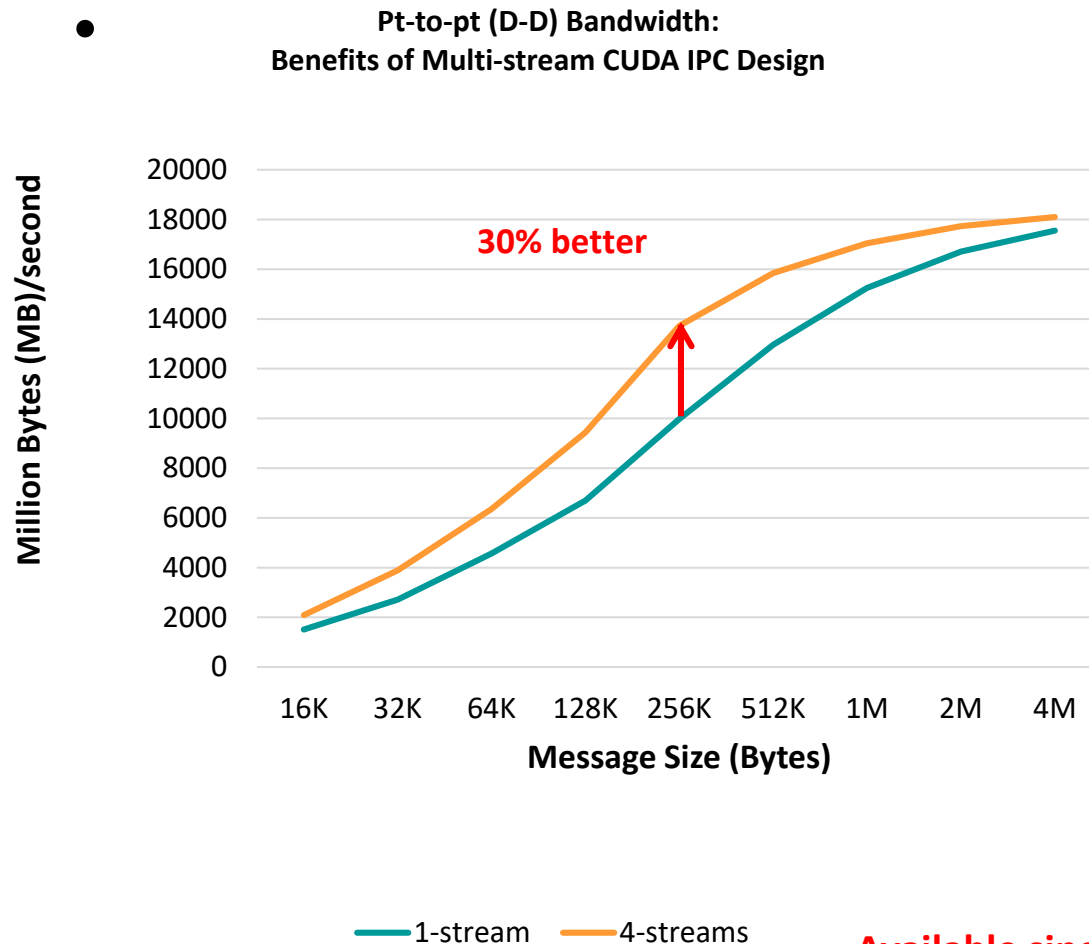
- Platform: Wilkes (Intel Ivy Bridge + NVIDIA Tesla K20c + Mellanox Connect-IB)
- HoomDBLue Version 1.0.5
  - GDRCOPY enabled: MV2\_USE\_CUDA=1 MV2\_IBA\_HCA=mlx5\_0 MV2\_IBA\_EAGER\_THRESHOLD=32768 MV2\_VBUF\_TOTAL\_SIZE=32768 MV2\_USE\_GPUDIRECT\_LOOPBACK\_LIMIT=32768 MV2\_USE\_GPUDIRECT\_GDRCOPY=1 MV2\_USE\_GPUDIRECT\_GDRCOPY\_LIMIT=16384

# Overview of MVAPICH2-GDR Features

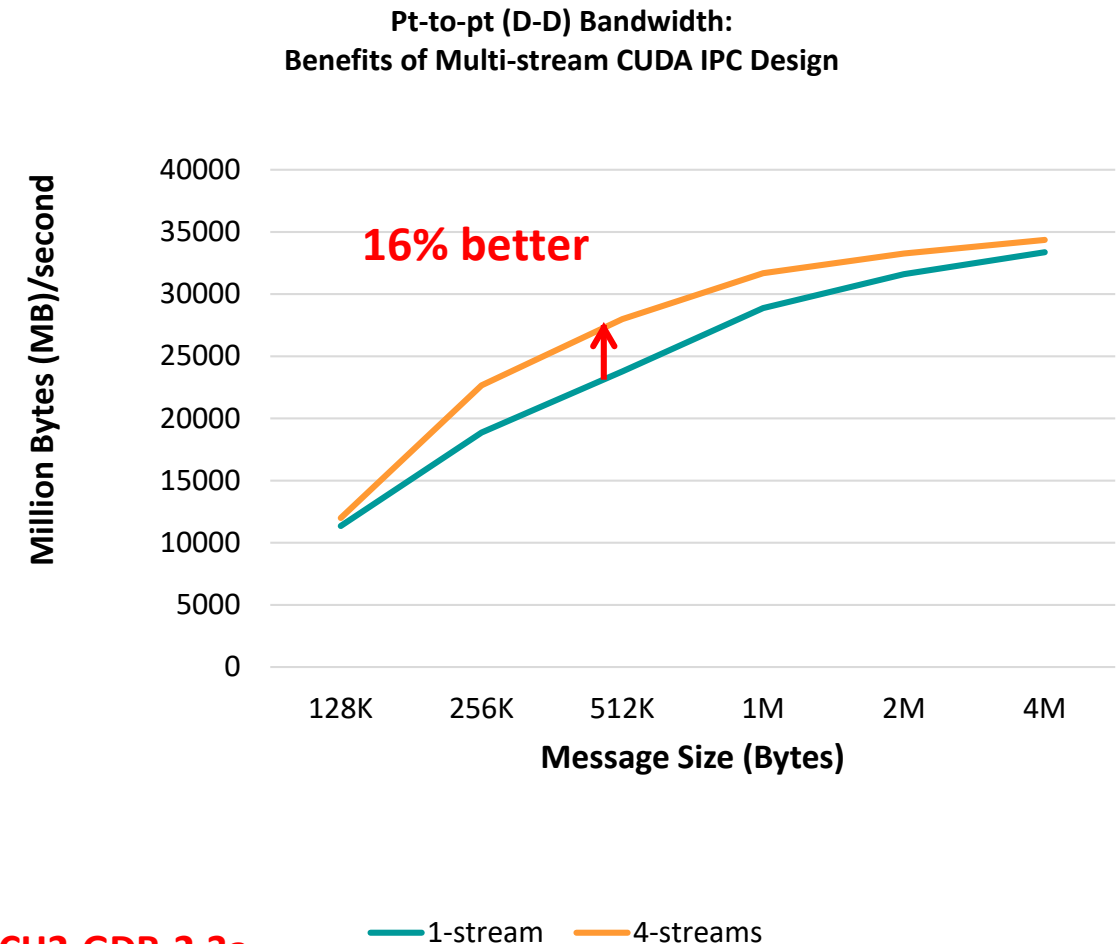
- Support for Efficient Small Message Communication with GPUDirect RDMA
- **Multi-stream Communication for IPC**
- **CMA- based Intra-node Host-to-Host Communication Support**
- MPI Datatype Support
- Support for Managed Memory
- Optimized Support for Deep Learning

# Multi-stream Communication using CUDA IPC on OpenPOWER and DGX-1

- Up to **16% higher** Device to Device (D2D) bandwidth on OpenPOWER + NVLink inter-connect
- Up to **30% higher** D2D bandwidth on DGX-1 with NVLink

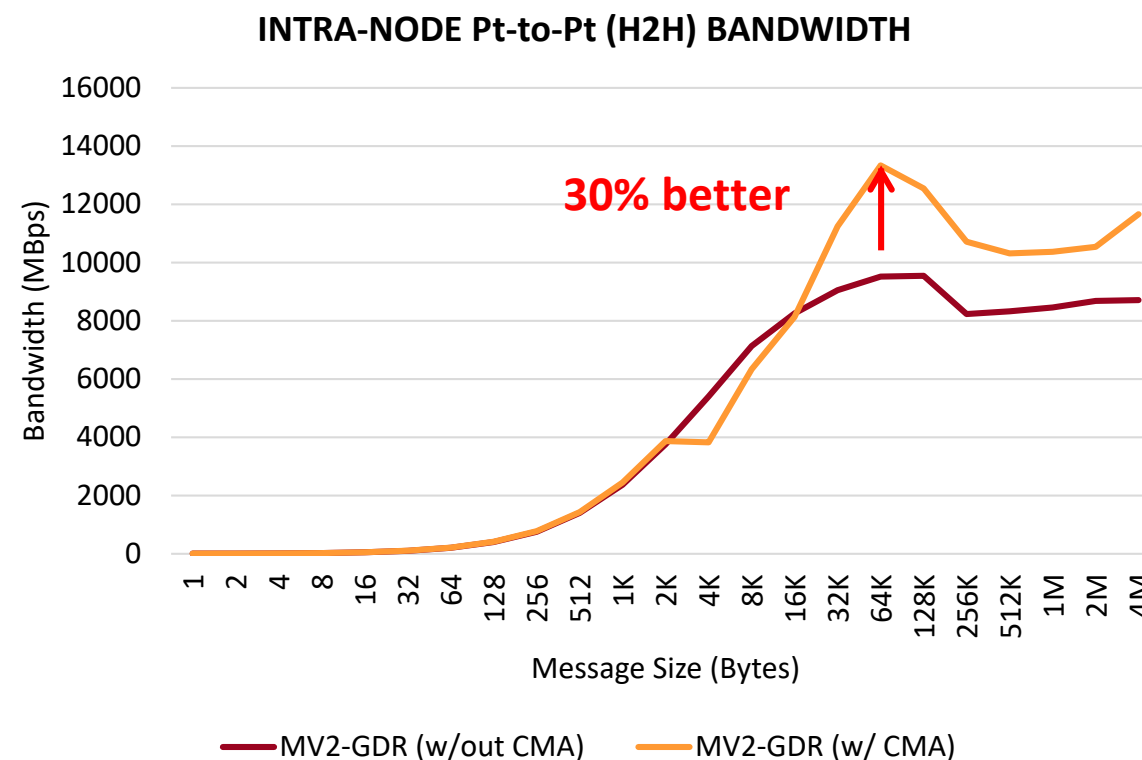
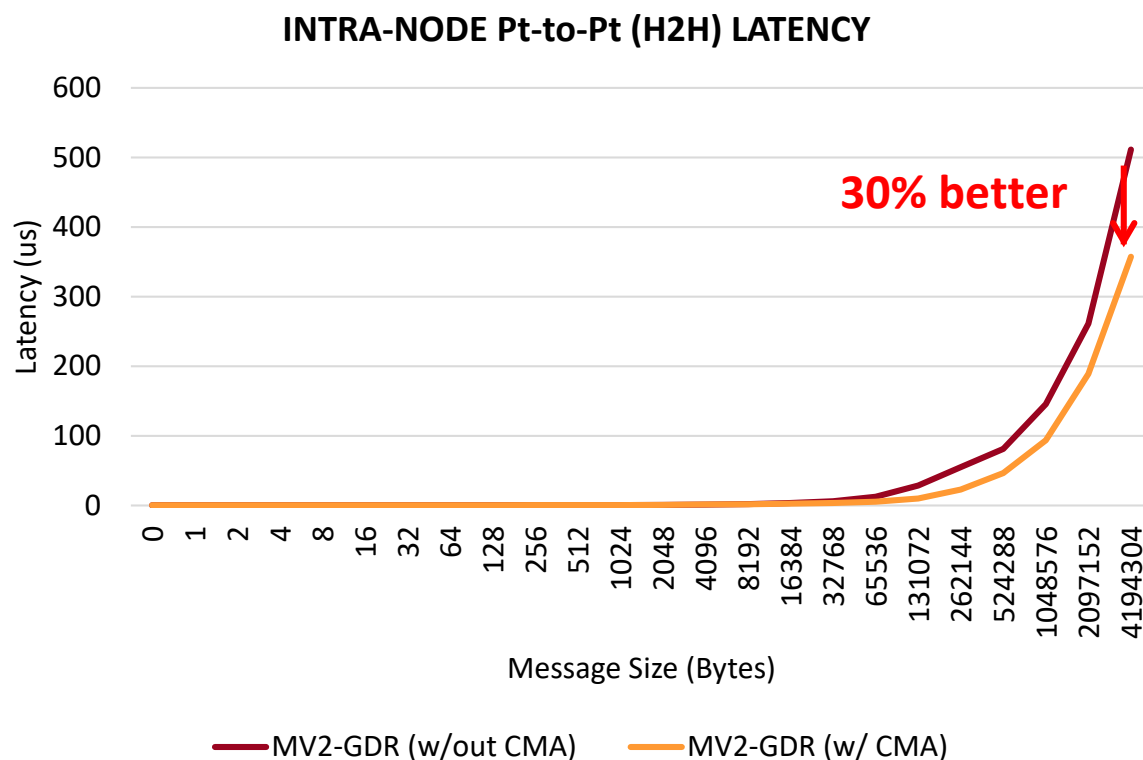


Available since **MVAPICH2-GDR-2.3a**



# CMA-based Intra-node Host-to-Host Communication Support

- Up to **30% lower** Host-to-Host (H2H) latency and **30% higher** H2H Bandwidth



**MVAPICH2-GDR-2.3a**  
**Intel Broadwell (E5-2680 v4 @ 3240 GHz) node – 28 cores**  
**NVIDIA Tesla K-80 GPU, and Mellanox Connect-X4 EDR HCA**  
**CUDA 8.0, Mellanox OFED 4.0 with GPU-Direct-RDMA**

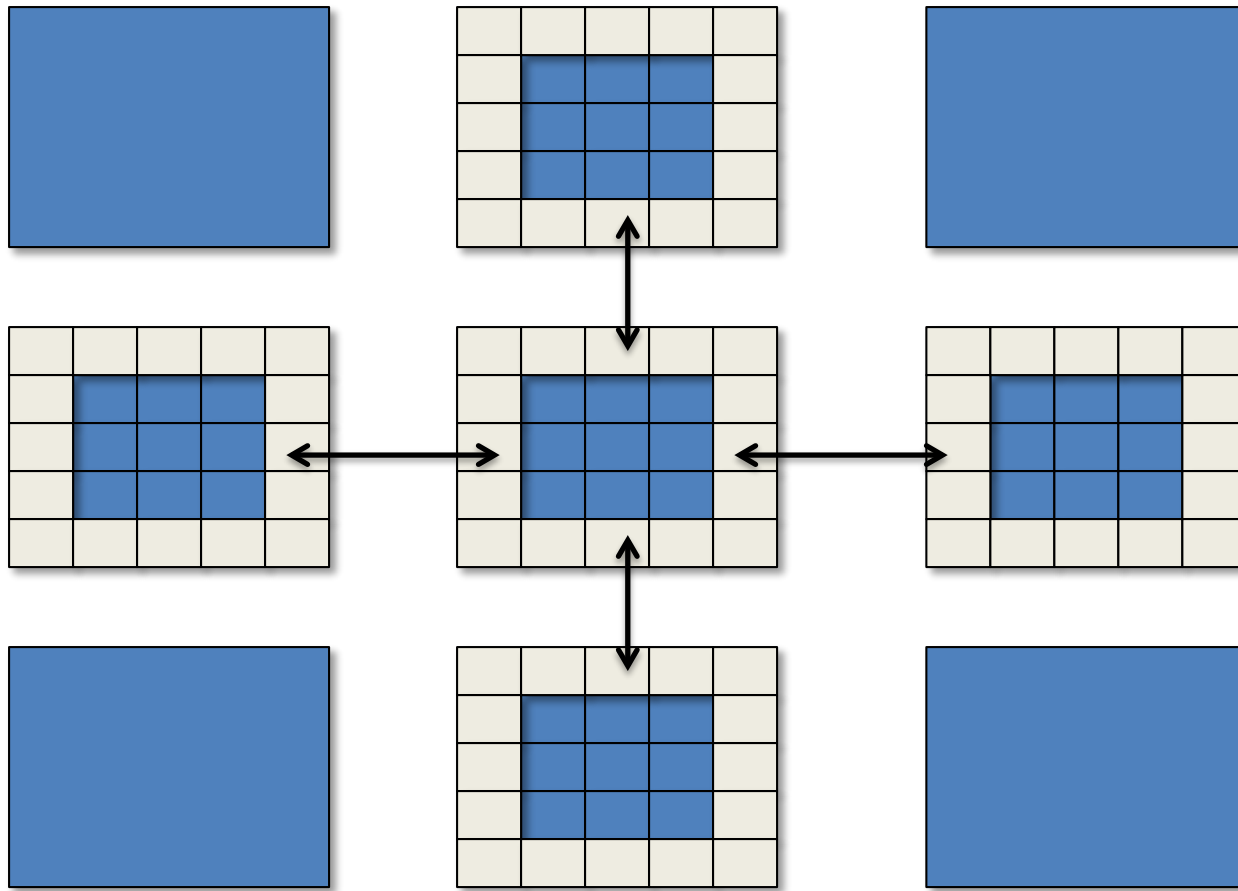


# Overview of MVAPICH2-GDR Features

- Support for Efficient Small Message Communication with GPUDirect RDMA
- Multi-stream Communication for IPC
- CMA- based Intra-node Host-to-Host Communication Support
- **MPI Datatype Support**
- Support for Managed Memory
- Optimized Support for Deep Learning

# Non-contiguous Data Exchange

## Halo data exchange



- Multi-dimensional data
  - Row based organization
  - Contiguous on one dimension
  - Non-contiguous on other dimensions
- Halo data exchange
  - Duplicate the boundary
  - Exchange the boundary in each iteration

# MPI Datatype support in MVAPICH2

- Datatypes support in MPI
  - Operate on customized datatypes to improve productivity
  - Enable MPI library to optimize non-contiguous data

## At Sender:

```
MPI_Type_vector (n_blocks, n_elements, stride, old_type, &new_type);
MPI_Type_commit(&new_type);
...
MPI_Send(s_buf, size, new_type, dest, tag, MPI_COMM_WORLD);
```

- Inside MVAPICH2
  - Use datatype specific CUDA Kernels to pack data in chunks
  - Efficiently move data between nodes using RDMA
  - In progress - currently optimizes *vector* and *hindexed* datatypes
  - Transparent to the user

*H. Wang, S. Potluri, D. Bureddy, C. Rosales and D. K. Panda, GPU-aware MPI on RDMA-Enabled Clusters: Design, Implementation and Evaluation, IEEE Transactions on Parallel and Distributed Systems, Accepted for Publication.*

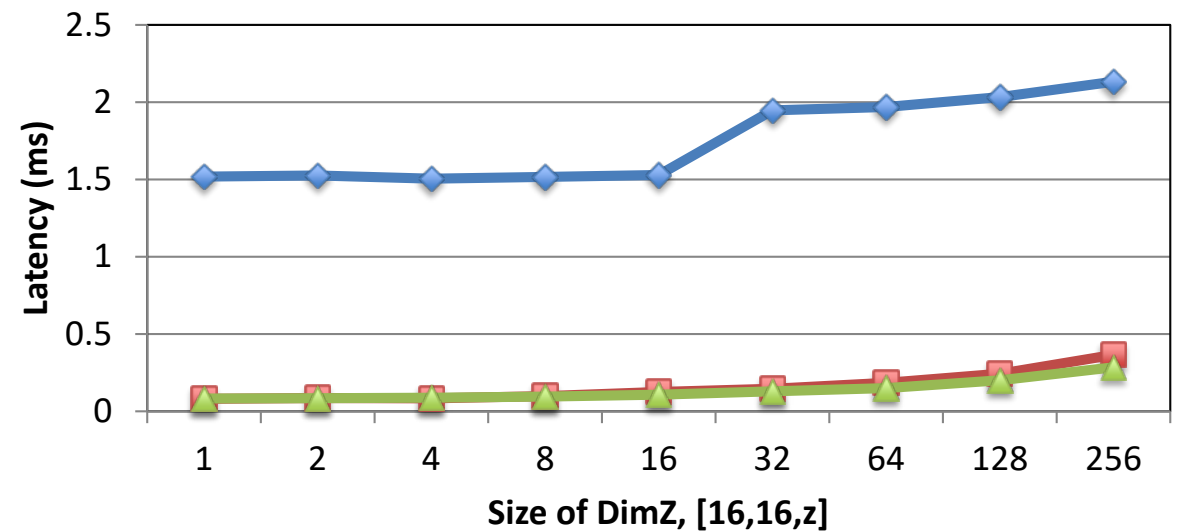
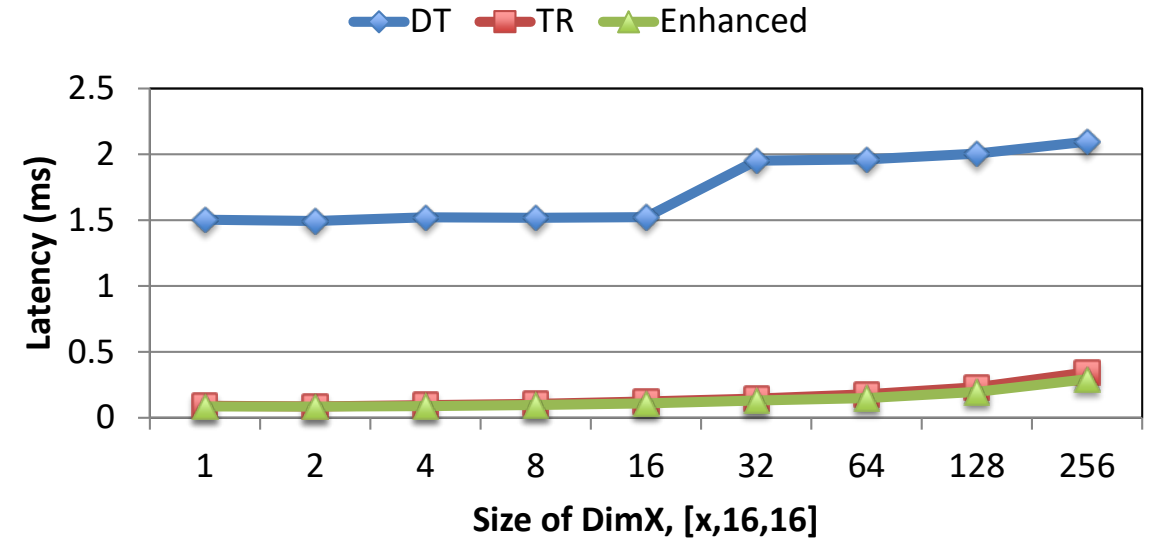
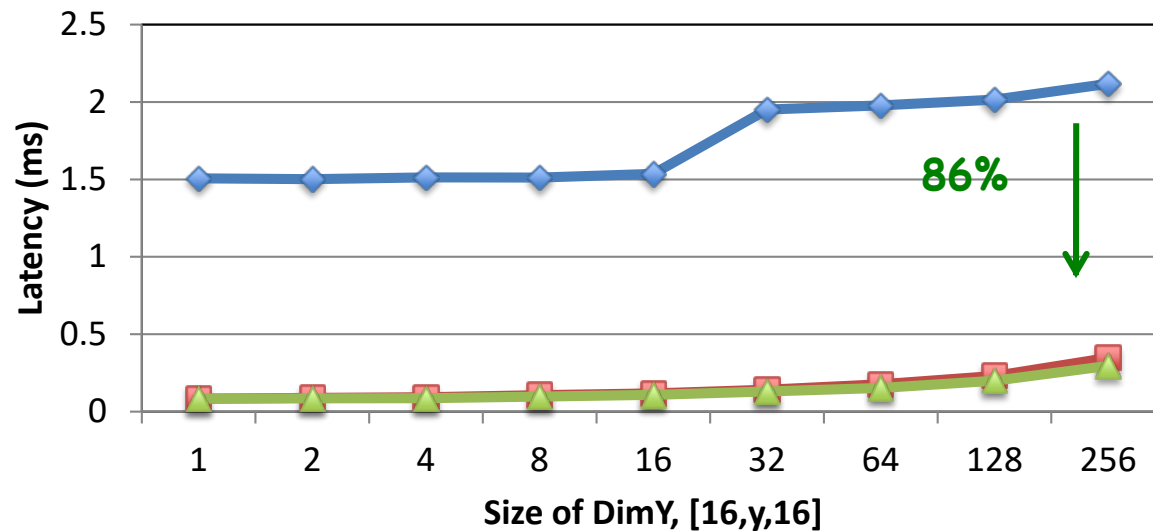
# MPI Datatype Processing (Computation Optimization )

- Comprehensive support
  - Targeted kernels for regular datatypes - vector, subarray, indexed\_block
  - Generic kernels for all other irregular datatypes
- Separate non-blocking stream for kernels launched by MPI library
  - Avoids stream conflicts with application kernels
- Flexible set of parameters for users to tune kernels
  - Vector
    - MV2\_CUDA\_KERNEL\_VECTOR\_TIDBLK\_SIZE
    - MV2\_CUDA\_KERNEL\_VECTOR\_YSIZE
  - Subarray
    - MV2\_CUDA\_KERNEL\_SUBARR\_TIDBLK\_SIZE
    - MV2\_CUDA\_KERNEL\_SUBARR\_XDIM
    - MV2\_CUDA\_KERNEL\_SUBARR\_YDIM
    - MV2\_CUDA\_KERNEL\_SUBARR\_ZDIM
  - Indexed\_block
    - MV2\_CUDA\_KERNEL\_IDXBLK\_XDIM

# Performance of Stencil3D (3D subarray)

Stencil3D communication kernel on 2 GPUs with various X, Y, Z dimensions using MPI\_Isend/Irecv

- DT: Direct Transfer, TR: Targeted Kernel
- Optimized design gains up to **15%**, **15%** and **22%** compared to TR, and more than **86%** compared to DT on X, Y and Z respectively



# MPI Datatype Processing (Communication Optimization )

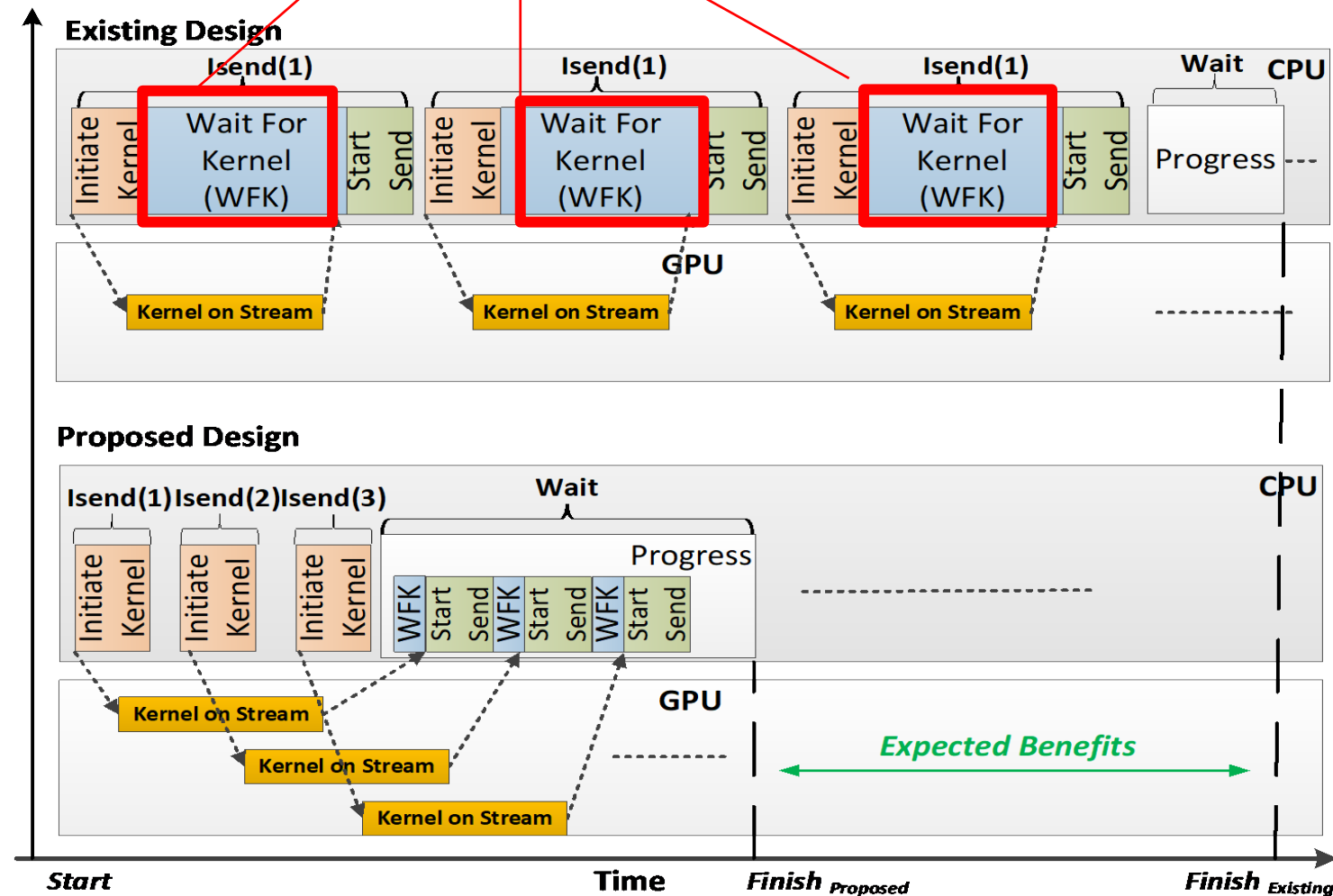
## Common Scenario

```
MPI_Isend (A,.. Datatype,...)
MPI_Isend (B,.. Datatype,...)
MPI_Isend (C,.. Datatype,...)
MPI_Isend (D,.. Datatype,...)
...

MPI_Waitall (...);
```

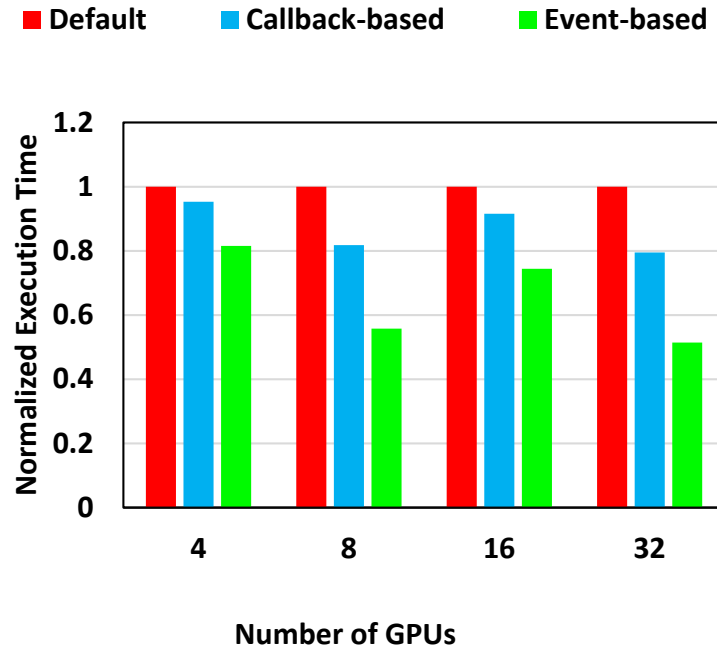
\*A, B...contain non-contiguous MPI Datatype

## Waste of computing resources on CPU and GPU

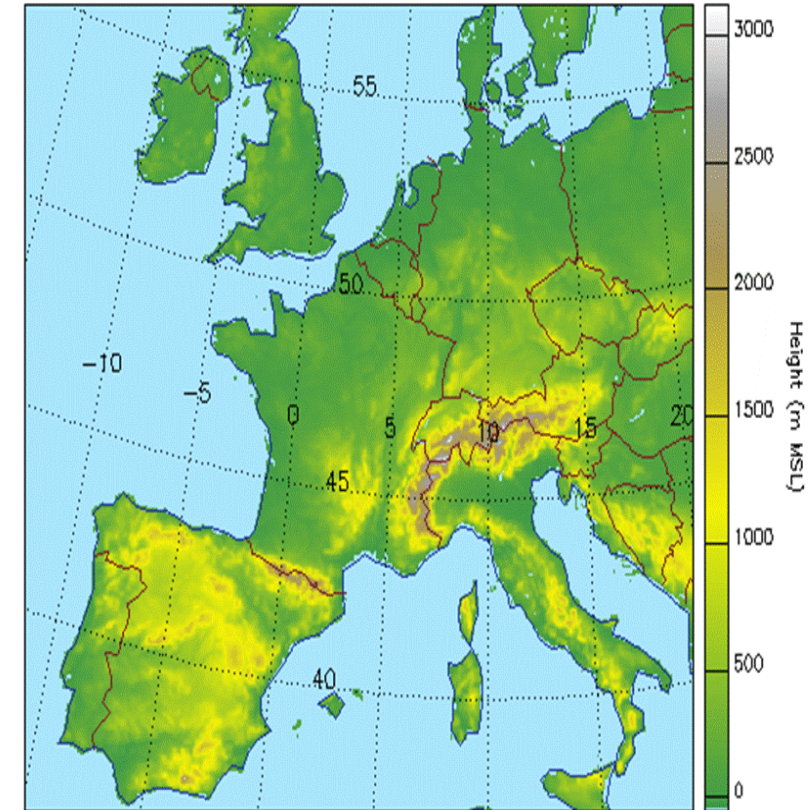
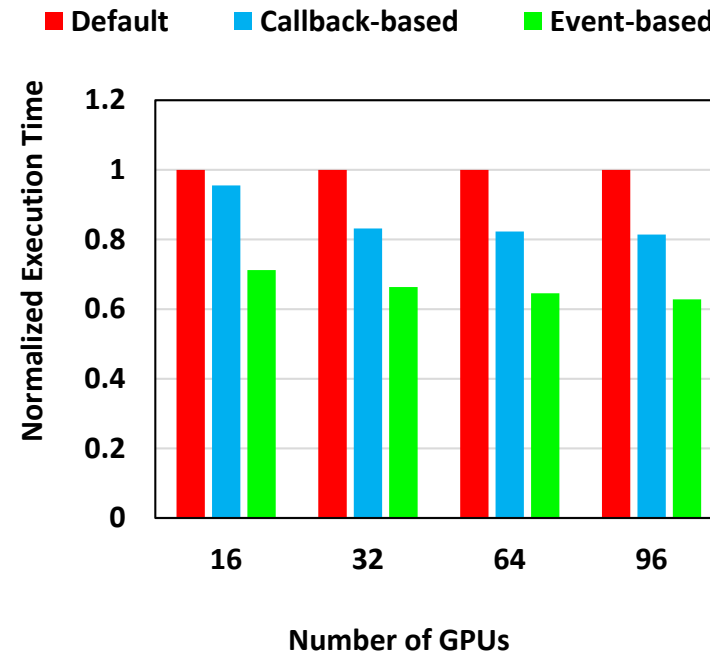


# Application-Level Evaluation (Cosmo) and Weather Forecasting in Switzerland

Wilkes GPU Cluster



CSCS GPU cluster



- 2X improvement on 32 GPUs nodes
- 30% improvement on 96 GPU nodes (8 GPUs/node)

Cosmo model: <http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/>

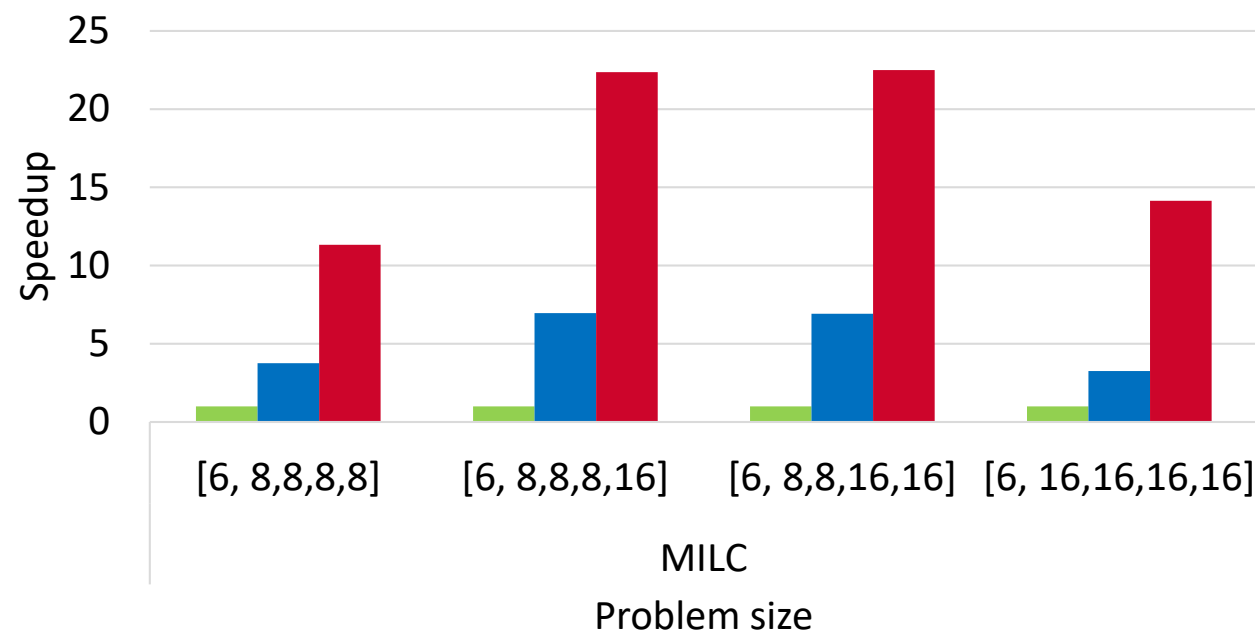
**On-going collaboration with CSCS and MeteoSwiss (Switzerland) in co-designing MV2-GDR and Cosmo Application**

C. Chu, K. Hamidouche, A. Venkatesh, D. Banerjee, H. Subramoni, and D. K. Panda, Exploiting Maximal Overlap for Non-Contiguous Data Movement Processing on Modern GPU-enabled Systems, IPDPS'16

# MVAPICH2-GDR: Enhanced Derived Datatype

- Kernel-based and GDRCOPY-based one-shot packing for inter-socket and inter-node communication
- Zero-copy (packing-free) for GPUs with peer-to-peer direct access over PCIe/NVLink

GPU-based DDTBench mimics MILC communication kernel

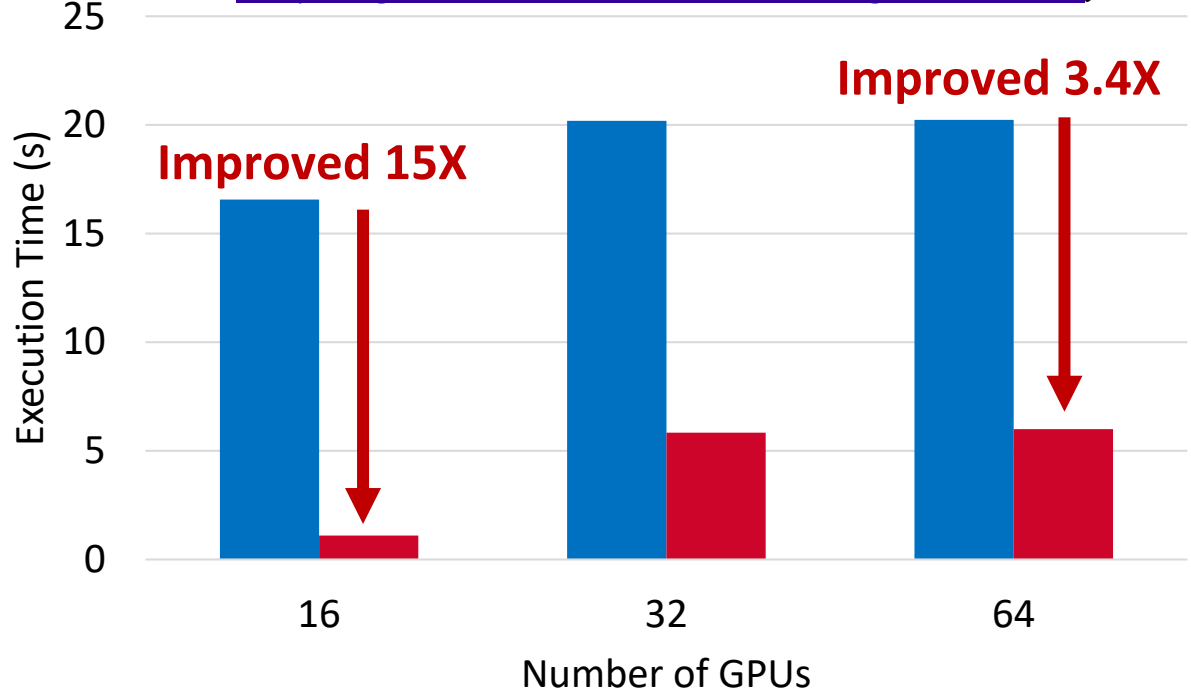


OpenMPI 4.0.0   MVAPICH2-GDR 2.3.1   MVAPICH2-GDR-Next

Platform: Nvidia DGX-2 system

(NVIDIA Volta GPUs connected with NVSwitch), CUDA 9.2

Communication Kernel of COSMO Model  
<https://github.com/cosunae/HaloExchangeBenchmarks>



MVAPICH2-GDR 2.3.1   MVAPICH2-GDR-Next

Platform: Cray CS-Storm

(16 NVIDIA Tesla K80 GPUs per node), CUDA 8.0



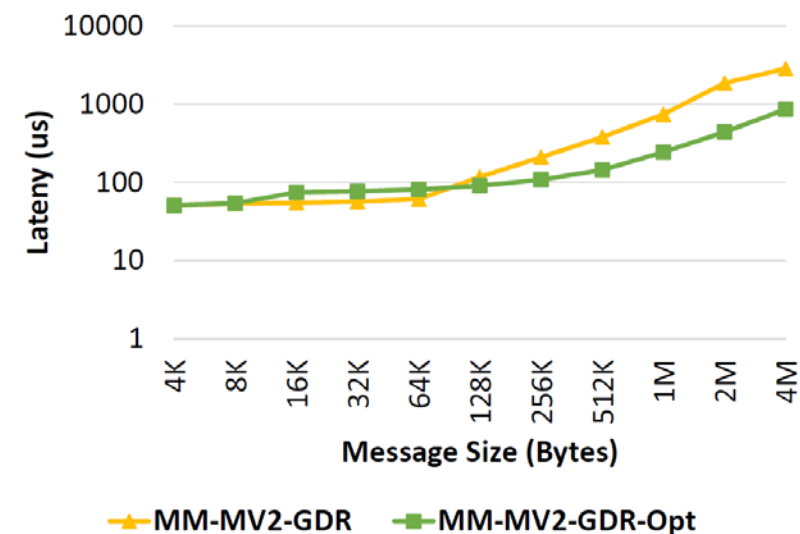
# Overview of MVAPICH2-GDR Features

- Support for Efficient Small Message Communication with GPUDirect RDMA
- Multi-stream Communication for IPC
- CMA- based Intra-node Host-to-Host Communication Support
- MPI Datatype Support
- **Support for Managed Memory**
- Optimized Support for Deep Learning

# Enhanced Support for Intra-node Unified Memory

- CUDA Unified Memory(UM) => no memory pin down
  - No IPC support for intra-node communication
  - No GDR support for Inter-node communication
- Initial and basic support in MVAPICH2-GDR
  - For both intra- and inter-nodes use “pipeline through” host memory
- Enhance intra-node UM to use IPC
  - Double buffering pair-wise IPC-based scheme
  - Brings IPC performance to UM
  - High performance and high productivity
- Available since MVAPICH2-GDR 2.2RC1

On K80 with MV2-GDR



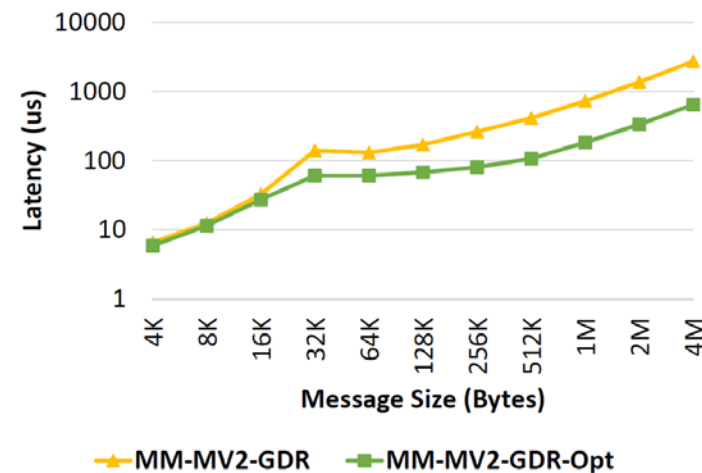
K. Hamidouche, A. Awan, A. Venkatesh, and D. K Panda, CUDA M3: Designing Efficient CUDA Managed Memory-aware MPI by Exploiting GDR and IPC, HiPC '16

# Characterizing Unified Memory aware MPI on modern GPUs

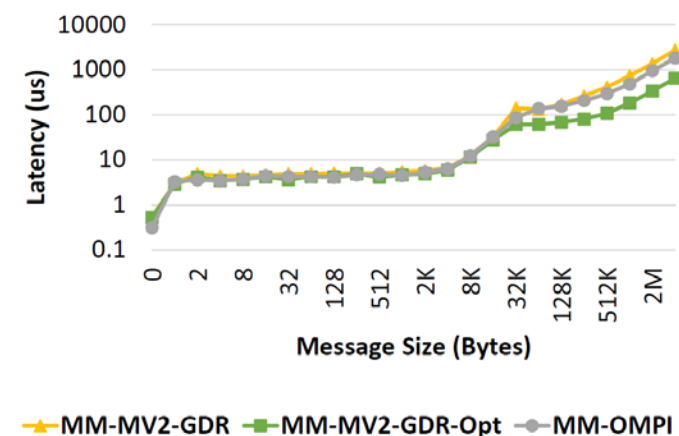
- Improved UM support in Pascal & Volta GPUs through:
  - Advanced GPU page fault engines
  - *cudaMemPrefetch* and *cudaMemAdvise* APIs provide more control for UM data placement
- Are the UM designs developed during Kepler era still valid?
- Carried out an in-depth characterization
- Our characterization studies show:
  - The UM designs from Kepler era are still valid
  - They are 4.2X and 2.8X better in latency compared to MVAPICH2-GDR and Open MPI

K. V. Manian, A. Awan, A. Ruhela, C. Chu, H. Subramoni and D. K Panda, Characterizing CUDA Unified Memory (UM)-Aware MPI Designs on Modern GPU Architectures, GPGPU '19 Workshop, in conjunction with ASPLOS '19, April '19

On V100 with MV2-GDR



On V100 with MV2-GDR and OMPI

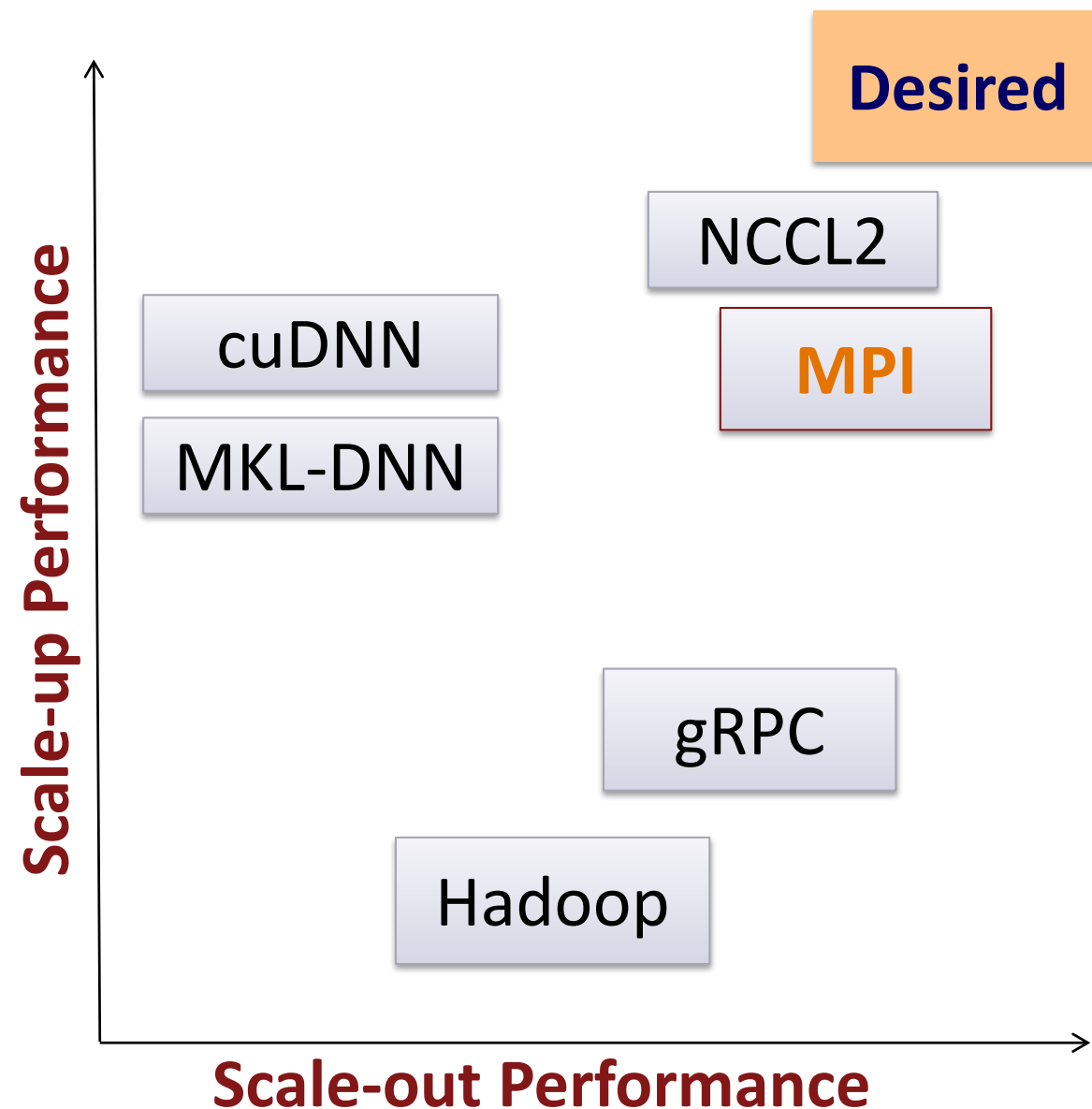


# Overview of MVAPICH2-GDR Features

- Support for Efficient Small Message Communication with GPUDirect RDMA
- Multi-stream Communication for IPC
- CMA- based Intra-node Host-to-Host Communication Support
- MPI Datatype Support
- Support for Managed Memory
- **Optimized Support for Deep Learning**

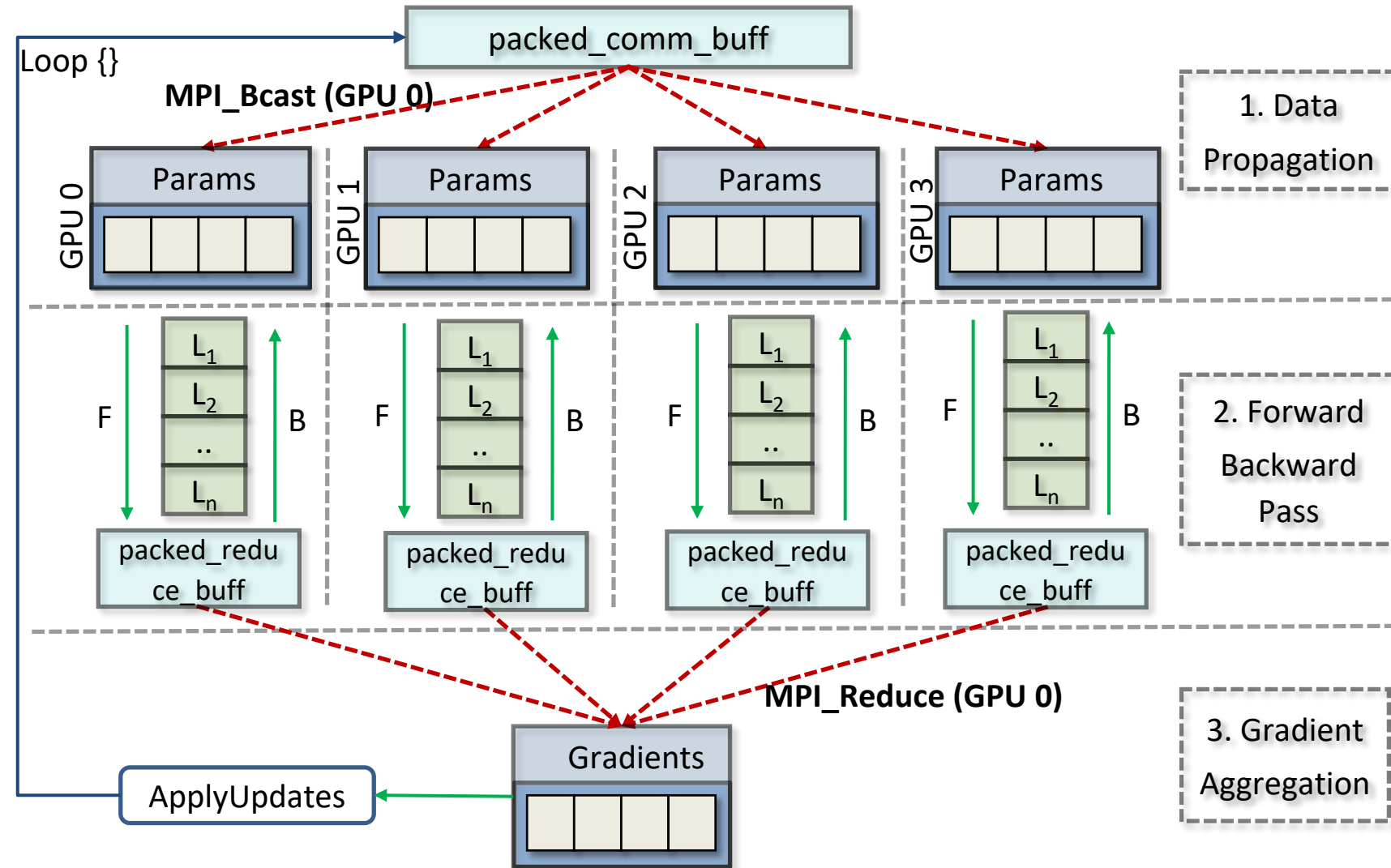
# Deep Learning: New Challenges for Runtimes

- **Scale-up:** Intra-node Communication
  - Many improvements like:
    - NVIDIA cuDNN, cuBLAS, NCCL, etc.
    - CUDA 9 Co-operative Groups
- **Scale-out:** Inter-node Communication
  - DL Frameworks – most are optimized for single-node only
  - Distributed (Parallel) Training is an emerging trend
    - **OSU-Caffe – MPI-based**
    - Microsoft CNTK – MPI/NCCL2
    - Google TensorFlow – gRPC-based/MPI/NCCL2
    - Facebook Caffe2 – Hybrid (NCCL2/Gloo/MPI)
    - PyTorch



# Data Parallel Deep Learning and MPI Collectives

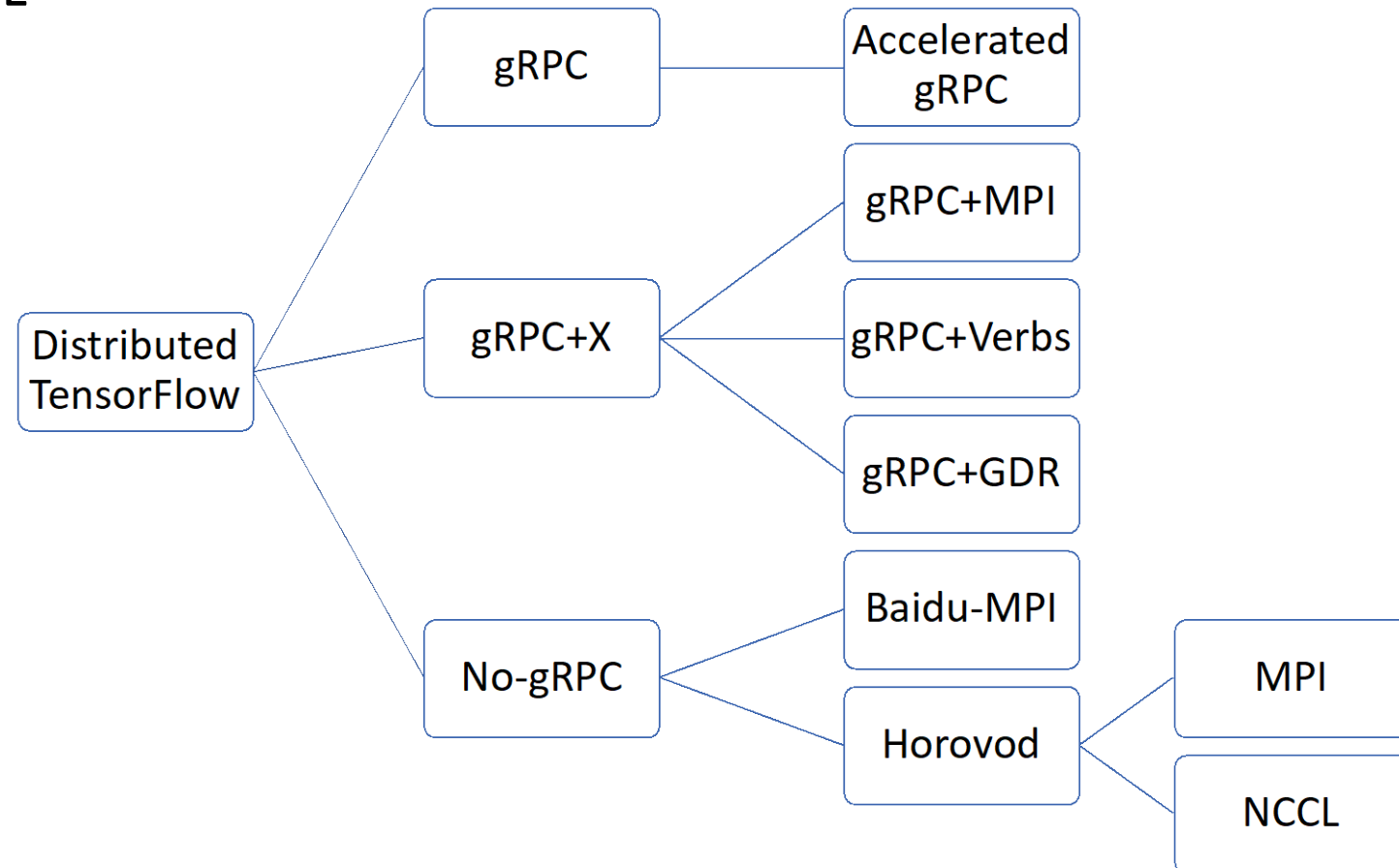
- Major **MPI Collectives** involved in Designing distributed frameworks
- **MPI\_Bcast** – required for DNN parameter exchange
- **MPI\_Reduce** – needed for gradient accumulation from multiple solvers
- **MPI\_Allreduce** – use just one Allreduce instead of Reduce and Broadcast



A. A. Awan, K. Hamidouche, J. M. Hashmi, and D. K. Panda, S-Caffe: Co-designing MPI Runtimes and Caffe for Scalable Deep Learning on Modern GPU Clusters. In *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '17)*

# Distributed Training using TensorFlow (TF)

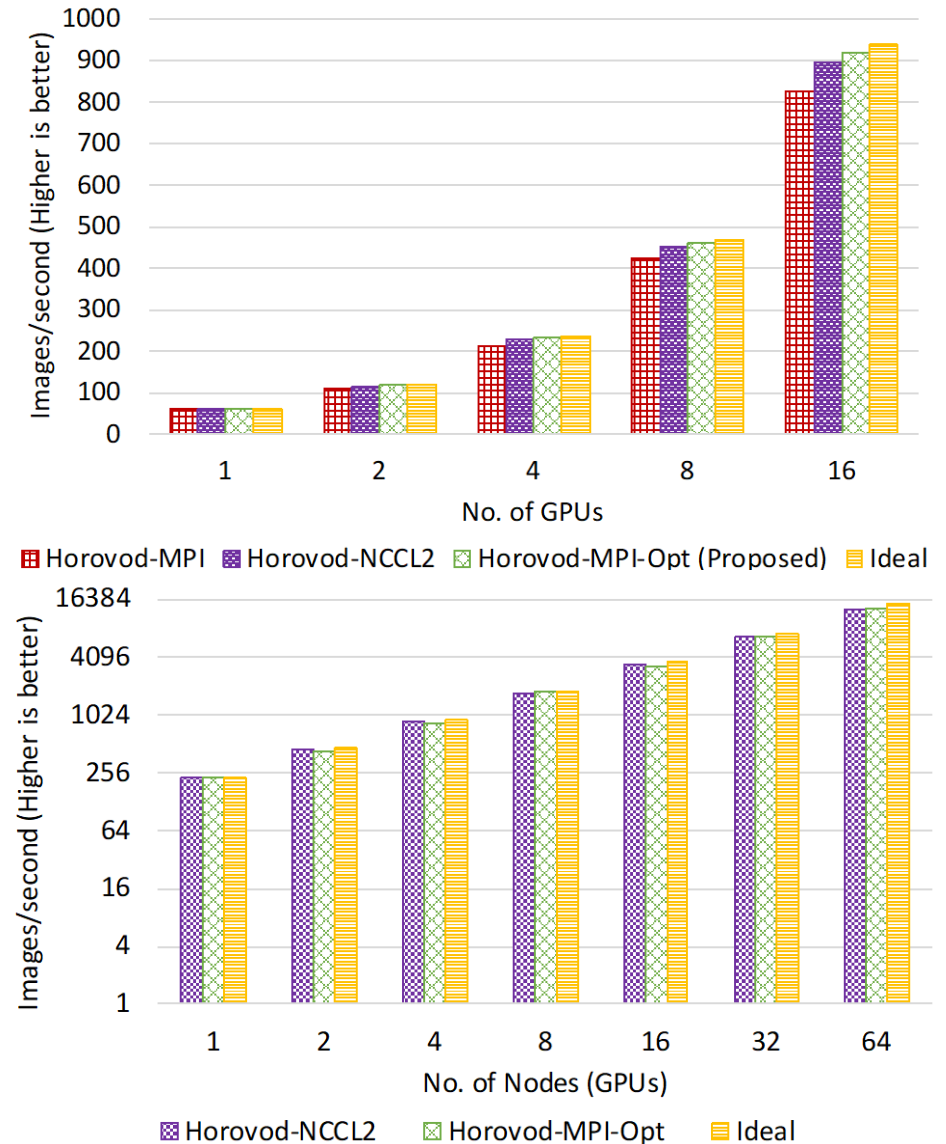
- TensorFlow is the most popular DL framework
- gRPC is the official distributed training runtime
  - Many problems for HPC use-cases
- Community efforts - Baidu and Uber's Horovod have added MPI support to TF across nodes
- Need to understand several options currently available →



Awan et al., "Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation", CCGrid '19. <https://arxiv.org/abs/1810.11112>

# Scalable TensorFlow using Horovod, MPI, and NCCL

- Efficient Allreduce is crucial for Horovod's overall training performance
  - Both MPI and NCCL designs are available
- We have evaluated Horovod extensively and compared across a wide range of designs using gRPC and gRPC extensions
- MVAPICH2-GDR achieved up to **90%** scaling efficiency for ResNet-50 Training on 64 Pascal GPUs

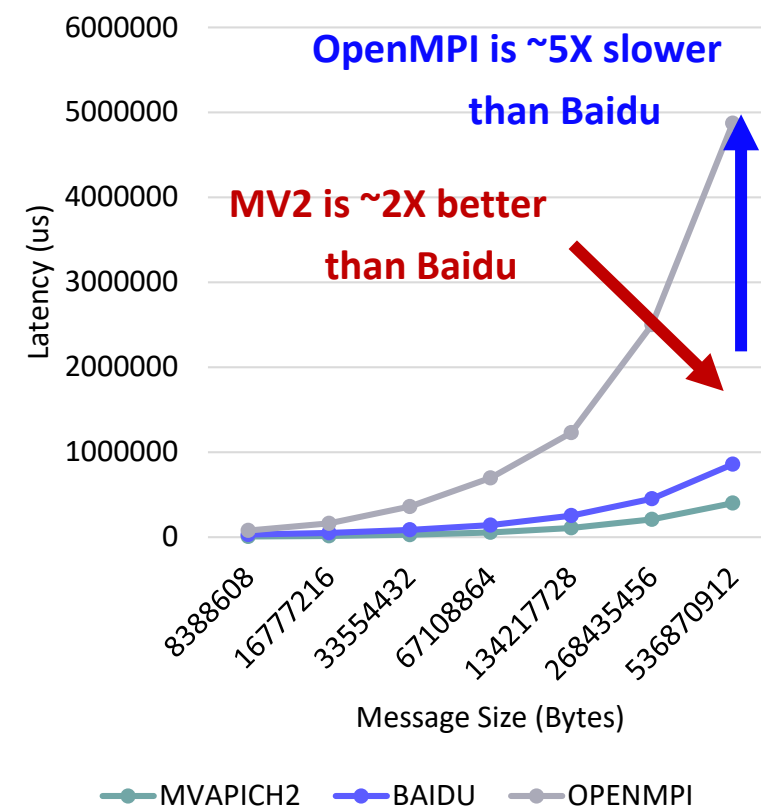
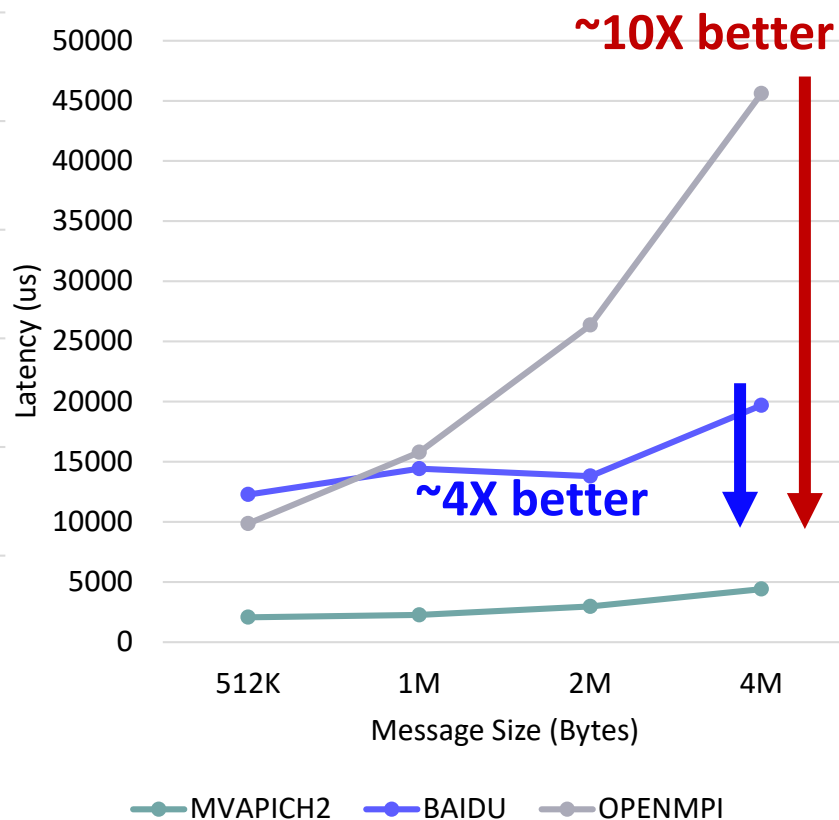
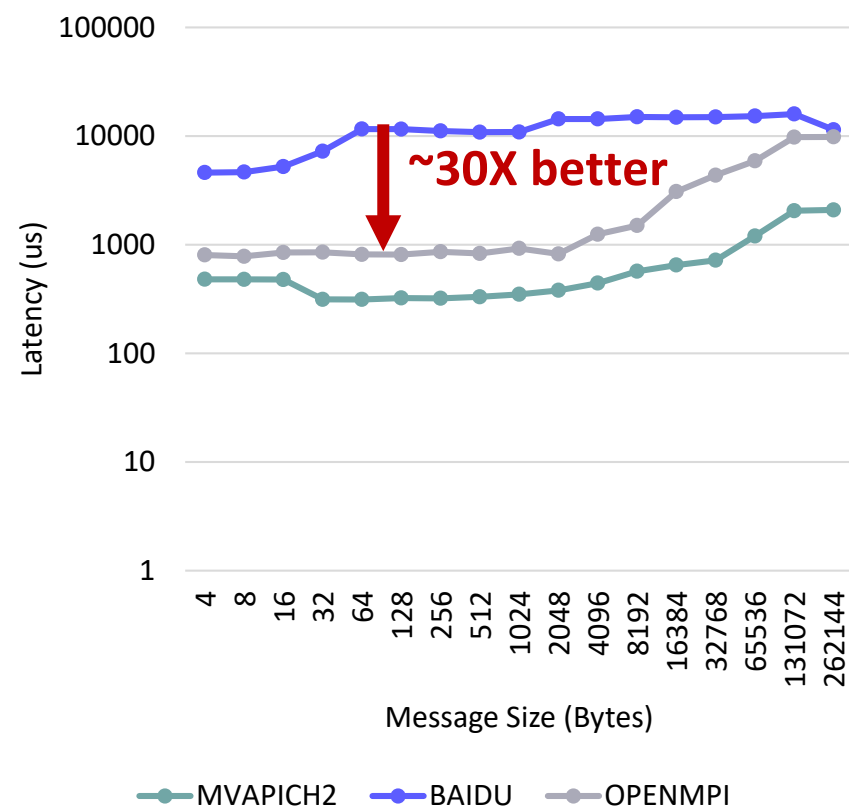


Awan et al., "Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation", CCGrid '19.  
<https://arxiv.org/abs/1810.11112>



# MVAPICH2-GDR: Allreduce Comparison with Baidu and OpenMPI

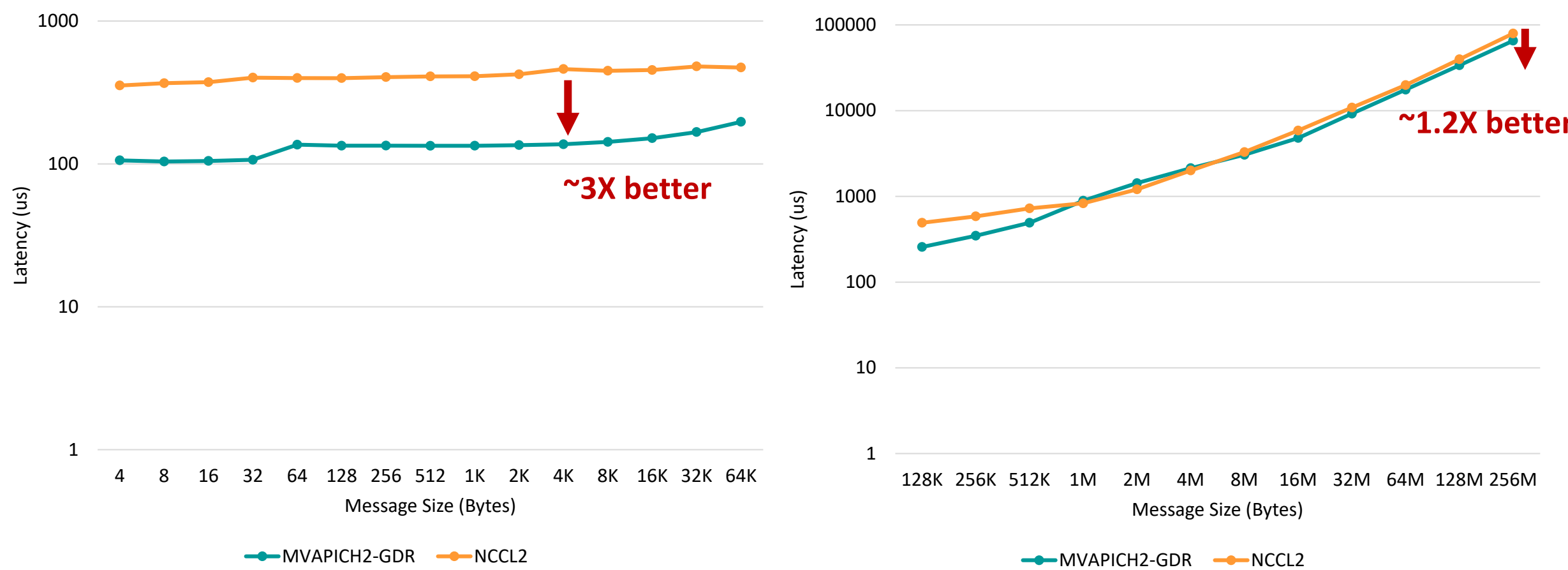
- 16 GPUs (4 nodes) MVAPICH2-GDR vs. Baidu-Allreduce and OpenMPI 3.0



*\*Available since MVAPICH2-GDR 2.3a*

# MVAPICH2-GDR vs. NCCL2 – Allreduce Operation

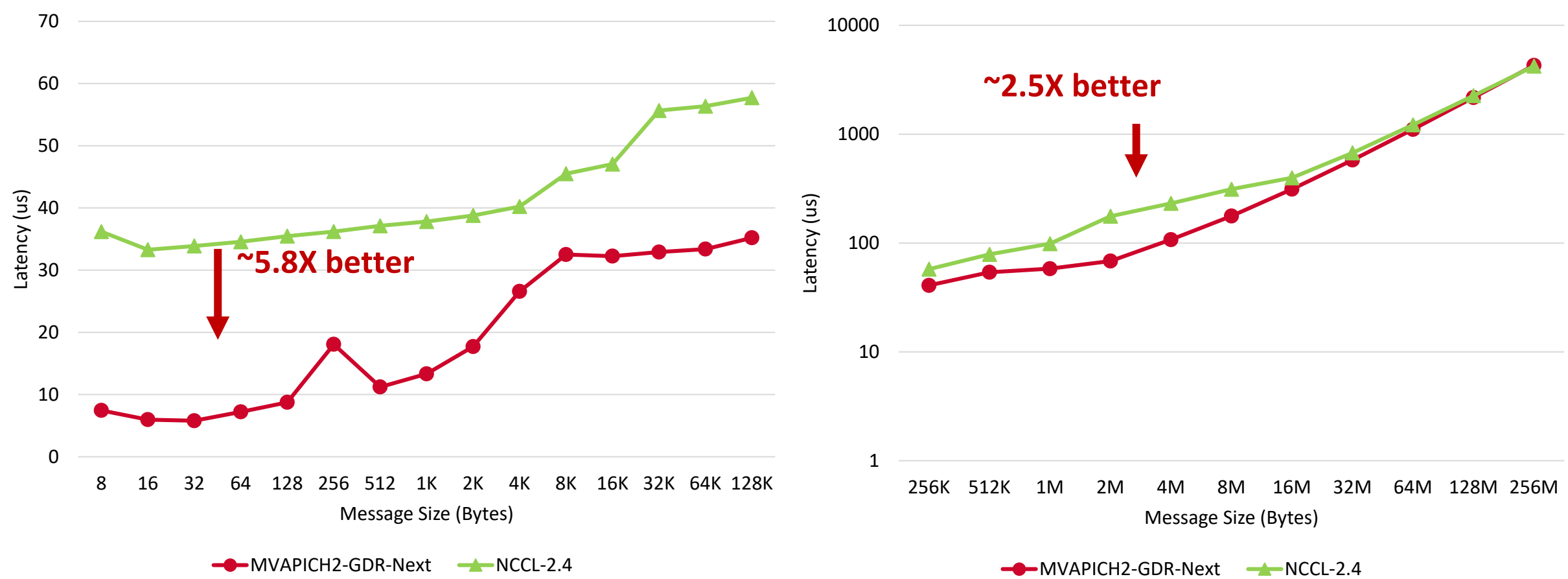
- Optimized designs in MVAPICH2-GDR 2.3 offer better/comparable performance for most cases
- MPI\_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 16 GPUs



Platform: Intel Xeon (Broadwell) nodes equipped with a dual-socket CPU, 1 K-80 GPUs, and EDR InfiniBand Inter-connect

# MVAPICH2-GDR vs. NCCL2 – Allreduce Operation (DGX-2)

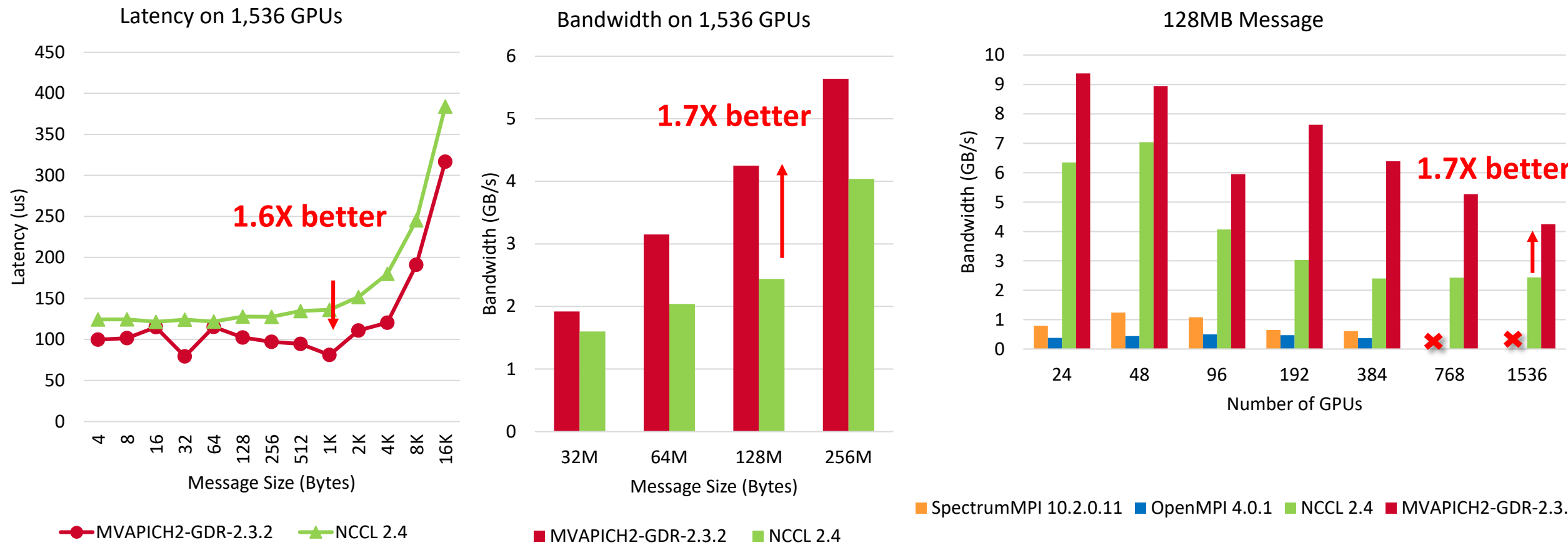
- Optimized designs in upcoming MVAPICH2-GDR offer better/comparable performance for most cases
- MPI\_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) on 1 DGX-2 node (16 Volta GPUs)



Platform: Nvidia DGX-2 system (16 Nvidia Volta GPUs connected with NVSwitch), CUDA 9.2

# MVAPICH2-GDR: Enhanced MPI\_Allreduce at Scale

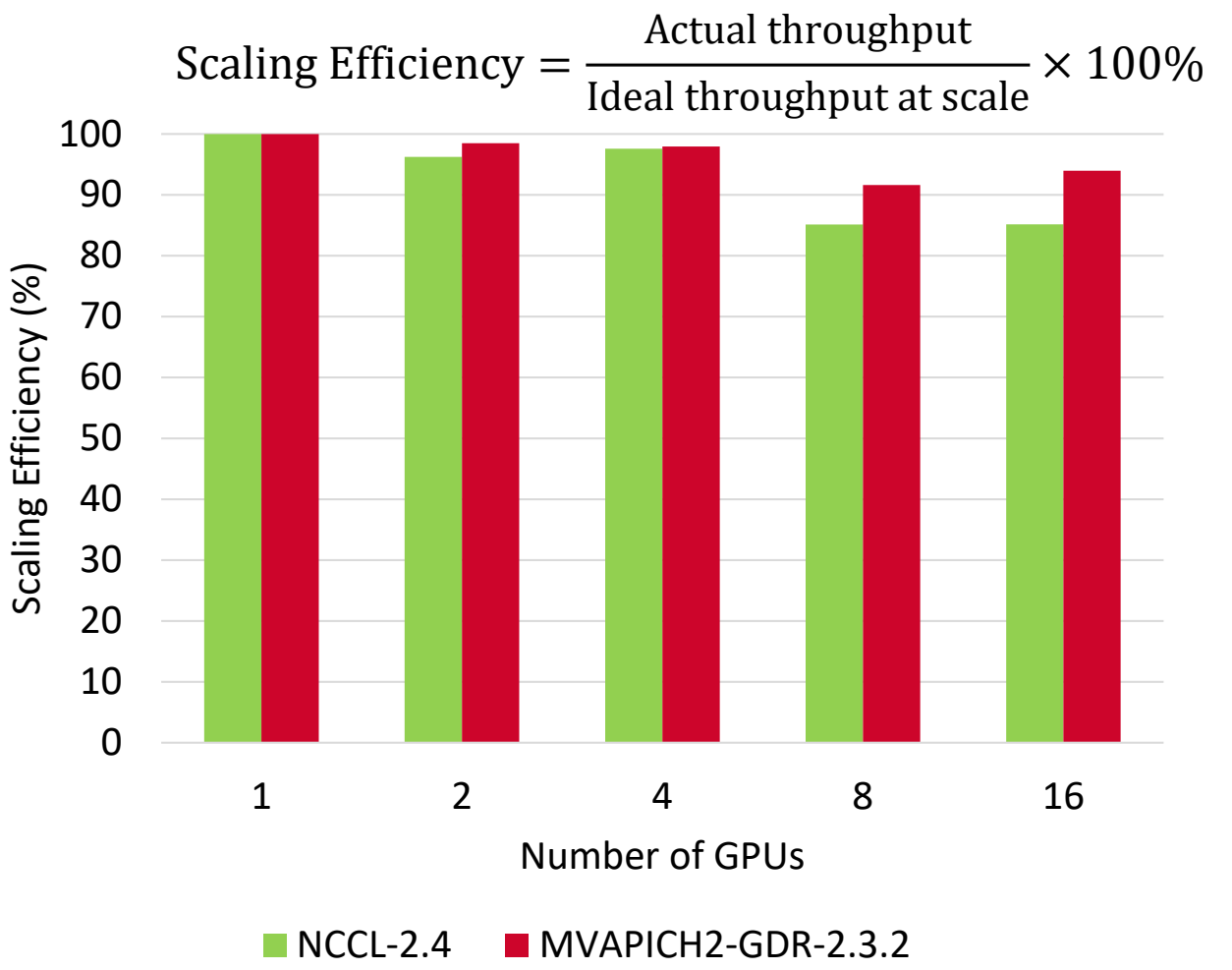
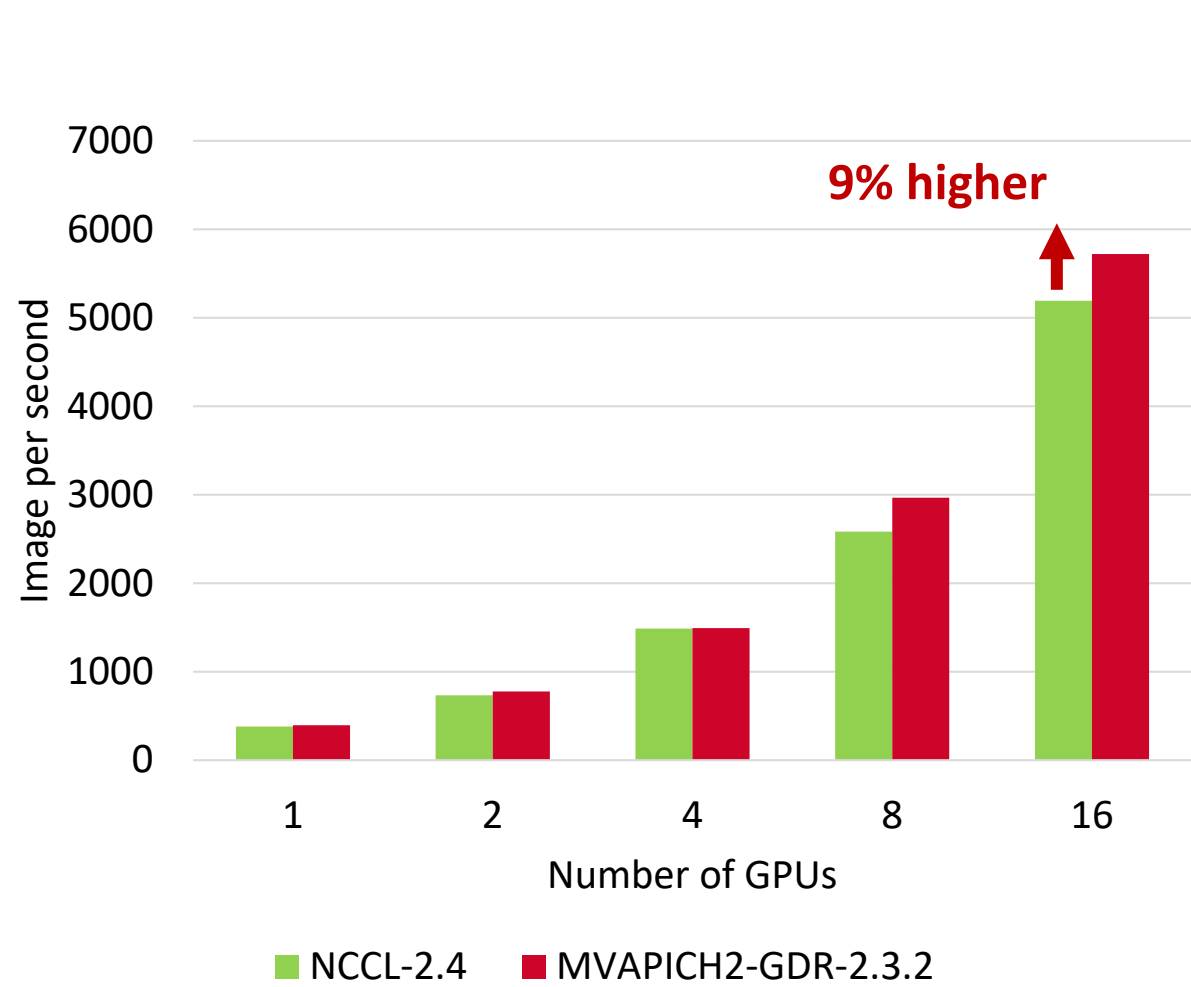
- Optimized designs in upcoming MVAPICH2-GDR offer better performance for most cases
- MPI\_Allreduce (MVAPICH2-GDR) vs. ncclAllreduce (NCCL2) up to 1,536 GPUs



Platform: Dual-socket IBM POWER9 CPU, 6 NVIDIA Volta V100 GPUs, and 2-port InfiniBand EDR Interconnect

# Distributed Training with TensorFlow and MVAPICH2-GDR

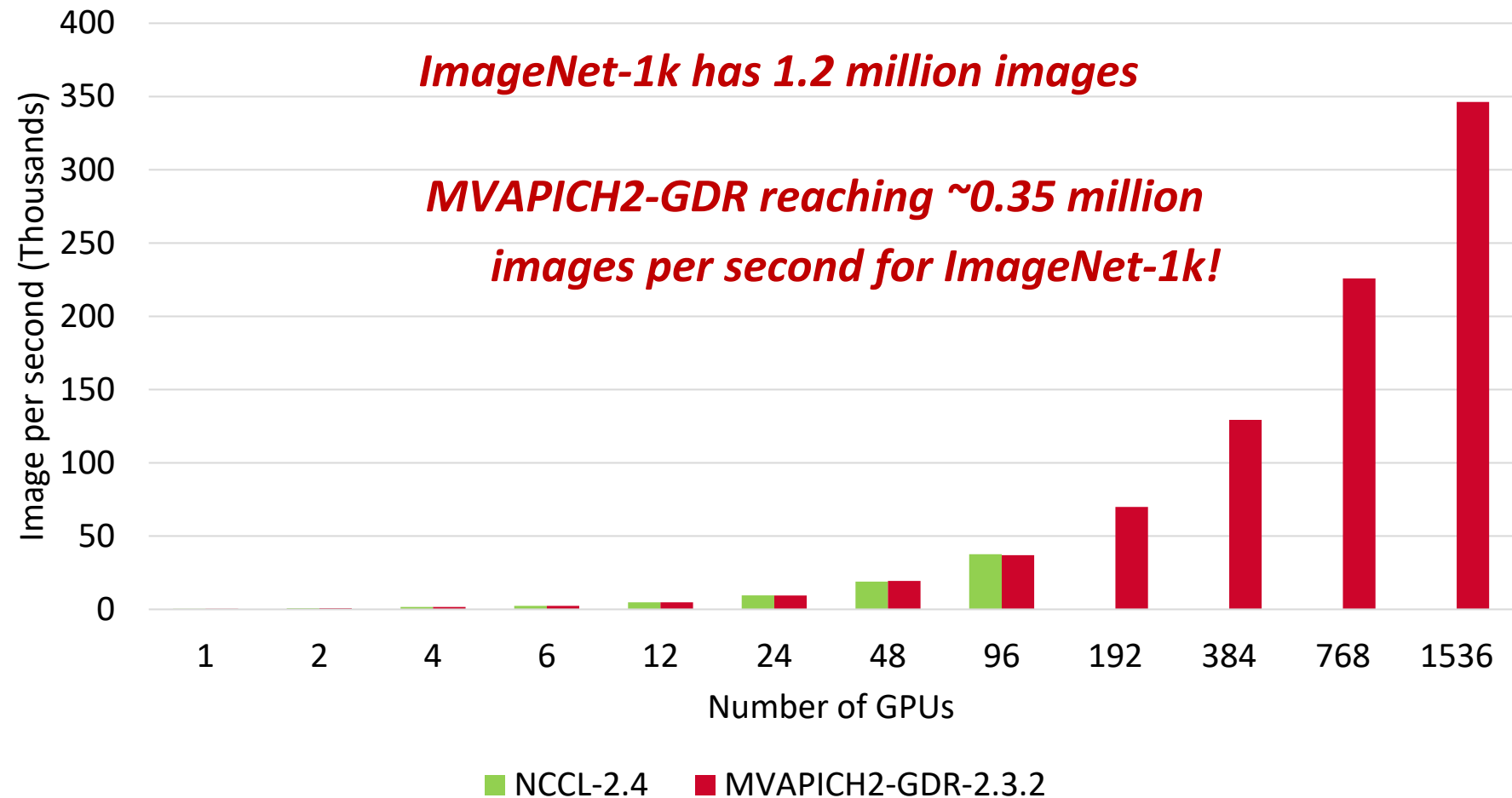
- ResNet-50 Training using TensorFlow benchmark on 1 DGX-2 node (16 Volta GPUs)



*Platform: Nvidia DGX-2 system (16 Nvidia Volta GPUs connected with NVSwitch), CUDA 9.2*

# Distributed Training with TensorFlow and MVAPICH2-GDR

- ResNet-50 Training using TensorFlow benchmark on SUMMIT -- 1536 Volta GPUs!
- 1,281,167 (1.2 mil.) images
- Time/epoch = 3.6 seconds
- Total Time (90 epochs) =  $3.6 \times 90 = 332$  seconds = **5.5 minutes!**



\*We observed errors for NCCL2 beyond 96 GPUs

*Platform: The Summit Supercomputer (#1 on Top500.org) – 6 NVIDIA Volta GPUs per node connected with NVLink, CUDA 9.2*

# MVAPICH2 Software Family

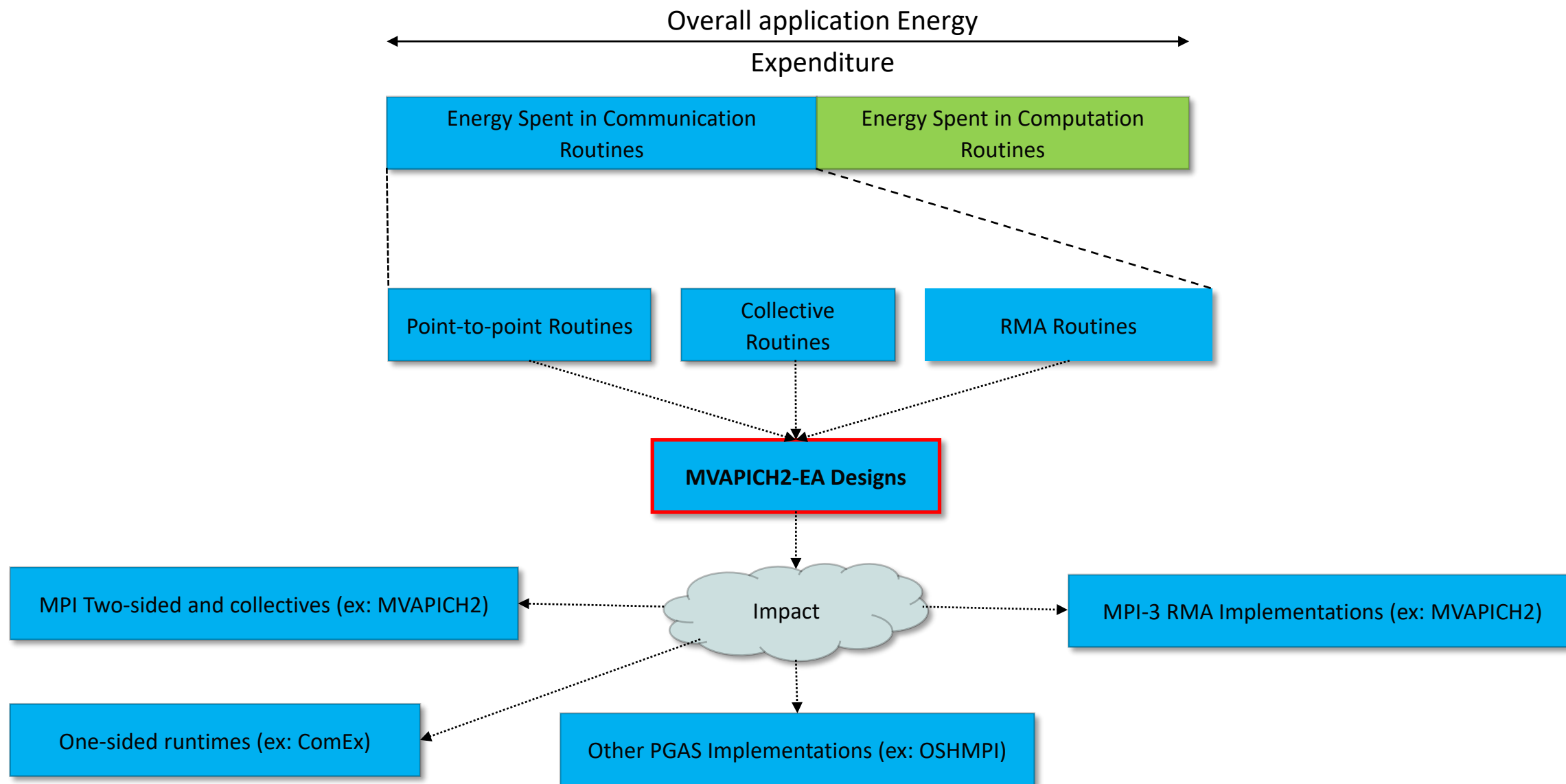
| Requirements                                                                                                                    | Library        |
|---------------------------------------------------------------------------------------------------------------------------------|----------------|
| MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                                    | MVAPICH2       |
| Optimized Support for Microsoft Azure Platform with InfiniBand                                                                  | MVAPICH2-Azure |
| Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis), | MVAPICH2-X     |
| Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric Adapter (EFA)                                      | MVAPICH2-X-AWS |
| Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications                                      | MVAPICH2-GDR   |
| Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                       | MVAPICH2-EA    |
| MPI Energy Monitoring Tool                                                                                                      | OEMT           |
| InfiniBand Network Analysis and Monitoring                                                                                      | OSU INAM       |
| Microbenchmarks for Measuring MPI and PGAS Performance                                                                          | OMB            |

# Energy-Aware MVAPICH2 & OSU Energy Management Tool (OEMT)

- MVAPICH2-EA 2.1 (Energy-Aware)
  - A white-box approach
  - New Energy-Efficient communication protocols for pt-pt and collective operations
  - Intelligently apply the appropriate Energy saving techniques
  - Application oblivious energy saving
- OEMT
  - A library utility to measure energy consumption for MPI applications
  - Works with all MPI runtimes
  - PRELOAD option for precompiled applications
  - Does not require ROOT permission:
    - A safe kernel module to read only a subset of MSRs

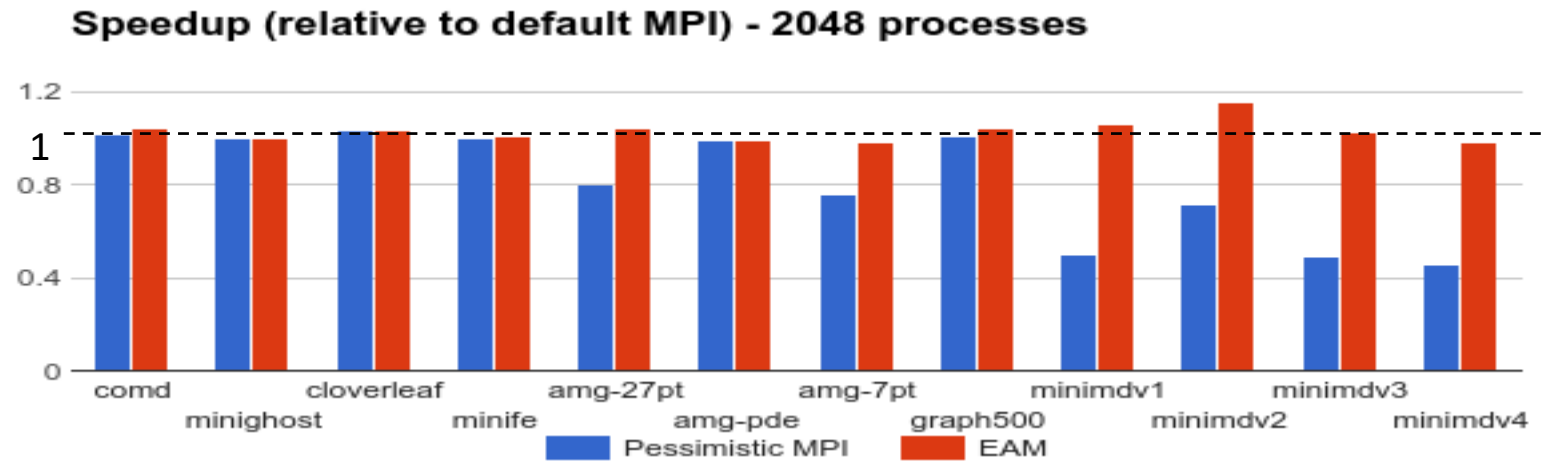
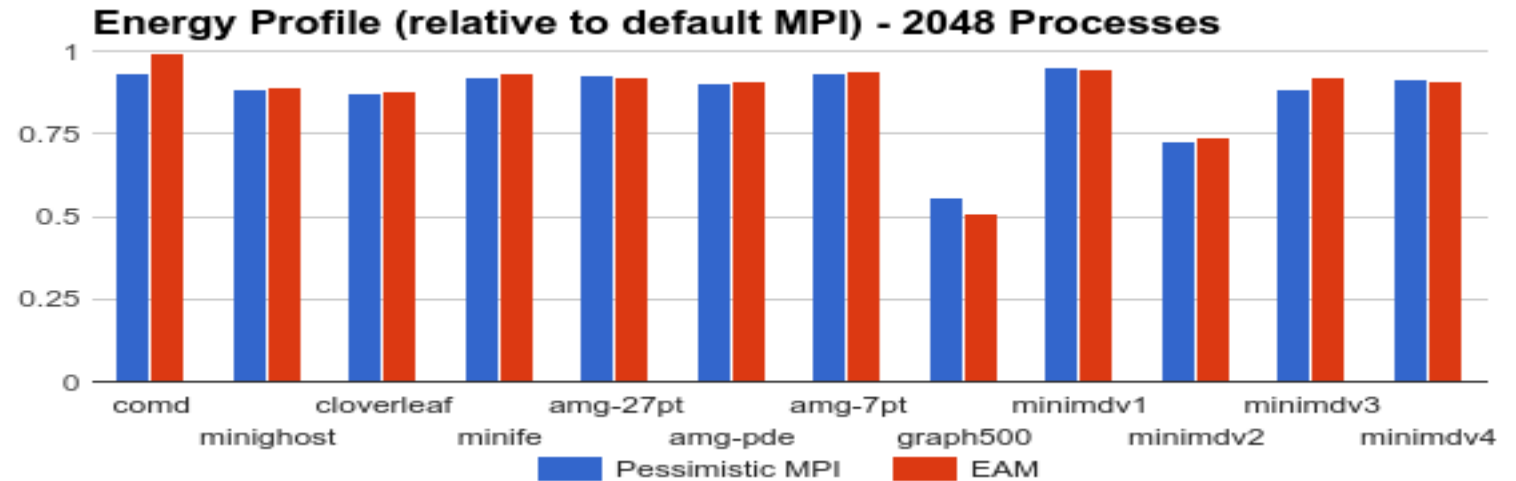


# Designing Energy-Aware (EA) MPI Runtime



# MVAPICH2-EA: Application Oblivious Energy-Aware-MPI (EAM)

- An energy efficient runtime that provides energy savings without application knowledge
- Uses automatically and transparently the best energy lever
- Provides guarantees on maximum degradation with 5-41% savings at  $\leq 5\%$  degradation
- Pessimistic MPI applies energy reduction lever to each MPI call



A Case for Application-Oblivious Energy-Efficient MPI Runtime A. Venkatesh, A. Vishnu, K. Hamidouche, N. Tallent, D.

K. Panda, D. Kerbyson, and A. Hoise, Supercomputing '15, Nov 2015 [*Best Student Paper Finalist*]

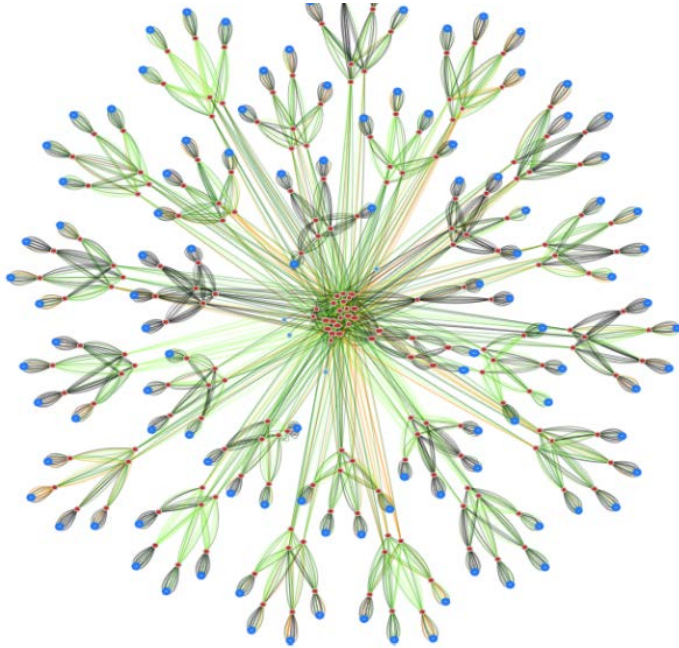
# MVAPICH2 Software Family

| Requirements                                                                                                                    | Library        |
|---------------------------------------------------------------------------------------------------------------------------------|----------------|
| MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                                    | MVAPICH2       |
| Optimized Support for Microsoft Azure Platform with InfiniBand                                                                  | MVAPICH2-Azure |
| Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis), | MVAPICH2-X     |
| Advanced MPI features (SRD and XPMEM) with support for Amazon Elastic Fabric Adapter (EFA)                                      | MVAPICH2-X-AWS |
| Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications                                      | MVAPICH2-GDR   |
| Energy-aware MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)                                       | MVAPICH2-EA    |
| MPI Energy Monitoring Tool                                                                                                      | OEMT           |
| InfiniBand Network Analysis and Monitoring                                                                                      | OSU INAM       |
| Microbenchmarks for Measuring MPI and PGAS Performance                                                                          | OMB            |

# Overview of OSU INAM

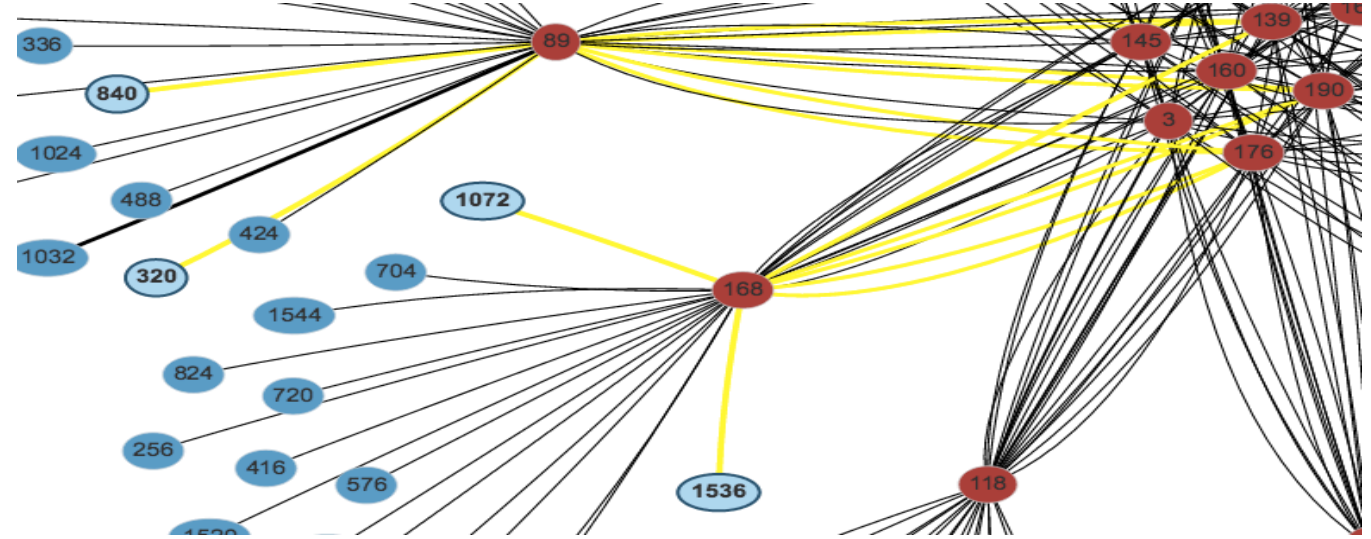
- A network monitoring and analysis tool that is capable of analyzing traffic on the InfiniBand network with inputs from the MPI runtime
  - <http://mvapich.cse.ohio-state.edu/tools/osu-inam/>
- Monitors IB clusters in real time by querying various subnet management entities and gathering input from the MPI runtimes
- Capability to analyze and profile **node-level, job-level and process-level activities** for MPI communication
  - Point-to-Point, Collectives and RMA
- Ability to filter data based on type of counters using “drop down” list
- Remotely monitor various metrics of MPI processes at user specified granularity
- "Job Page" to display jobs in ascending/descending order of various performance metrics in conjunction with MVAPICH2-X
- Visualize the data transfer happening in a **“live” or “historical”** fashion for entire network, job or set of nodes
- **OSU INAM 0.9.4 released on 11/10/2018**
  - Enhanced performance for fabric discovery using optimized OpenMP-based multi-threaded designs
  - Ability to gather InfiniBand performance counters at sub-second granularity for very large (>2000 nodes) clusters
  - Redesign database layout to reduce database size
  - Enhanced fault tolerance for database operations
    - Thanks to Trey Dockendorf @ OSC for the feedback
  - OpenMP-based multi-threaded designs to handle database purge, read, and insert operations simultaneously
  - Improved database purging time by using bulk deletes
  - Tune database timeouts to handle very long database operations
  - Improved debugging support by introducing several debugging levels

# OSU INAM Features



Comet@SDSC --- Clustered View

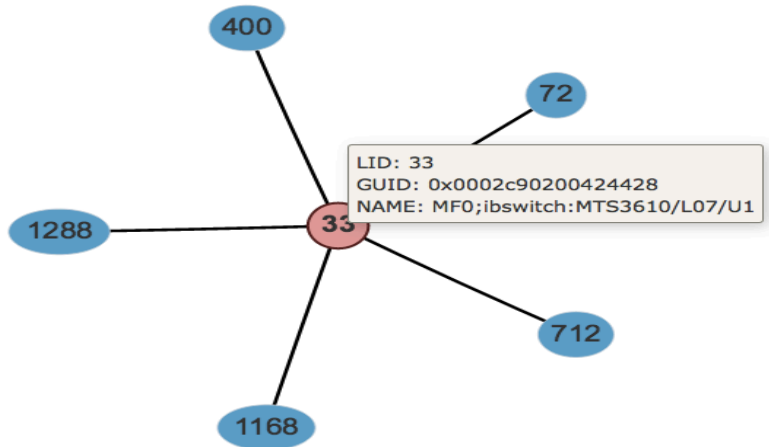
(1,879 nodes, 212 switches, 4,377 network links)



Finding Routes Between Nodes

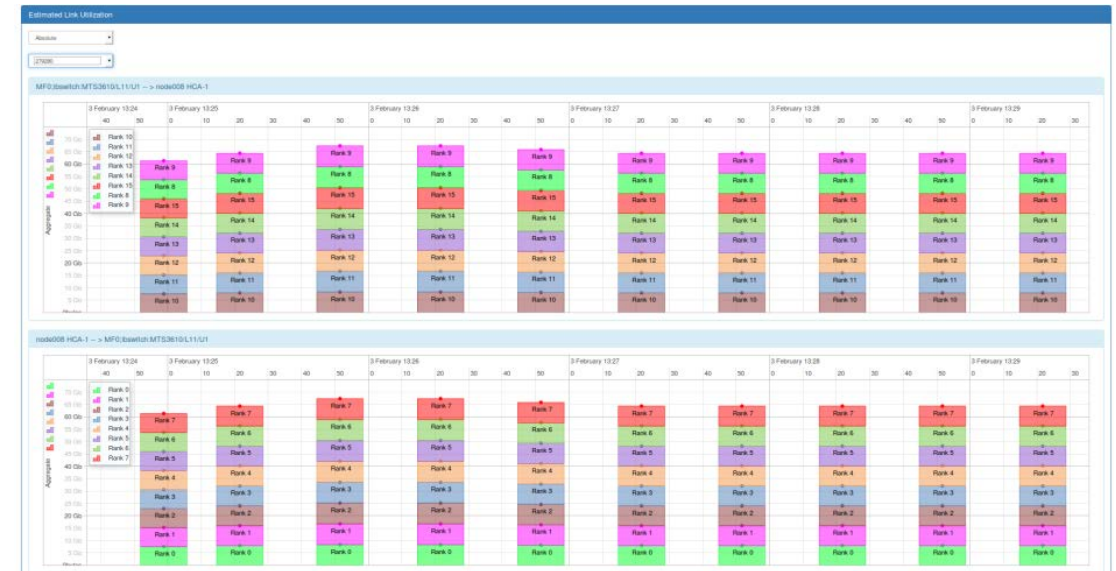
- Show network topology of large clusters
- Visualize traffic pattern on different links
- Quickly identify congested links/links in error state
- See the history unfold – play back historical state of the network

# OSU INAM Features (Cont.)



Visualizing a Job (5 Nodes)

- Job level view
  - Show different network metrics (load, error, etc.) for any live job
  - Play back historical data for completed jobs to identify bottlenecks
- Node level view - details per process or per node
  - CPU utilization for each rank/node
  - Bytes sent/received for MPI operations (pt-to-pt, collective, RMA)
  - Network metrics (e.g. XmitDiscard, RcvError) per rank/node



Estimated Process Level Link Utilization

- Estimated Link Utilization view
  - Classify data flowing over a network link at different granularity in conjunction with MVAPICH2-X 2.2rc1
    - Job level and
    - Process level

More Details in Tutorial/Demo

Session Tomorrow

# OSU Microbenchmarks

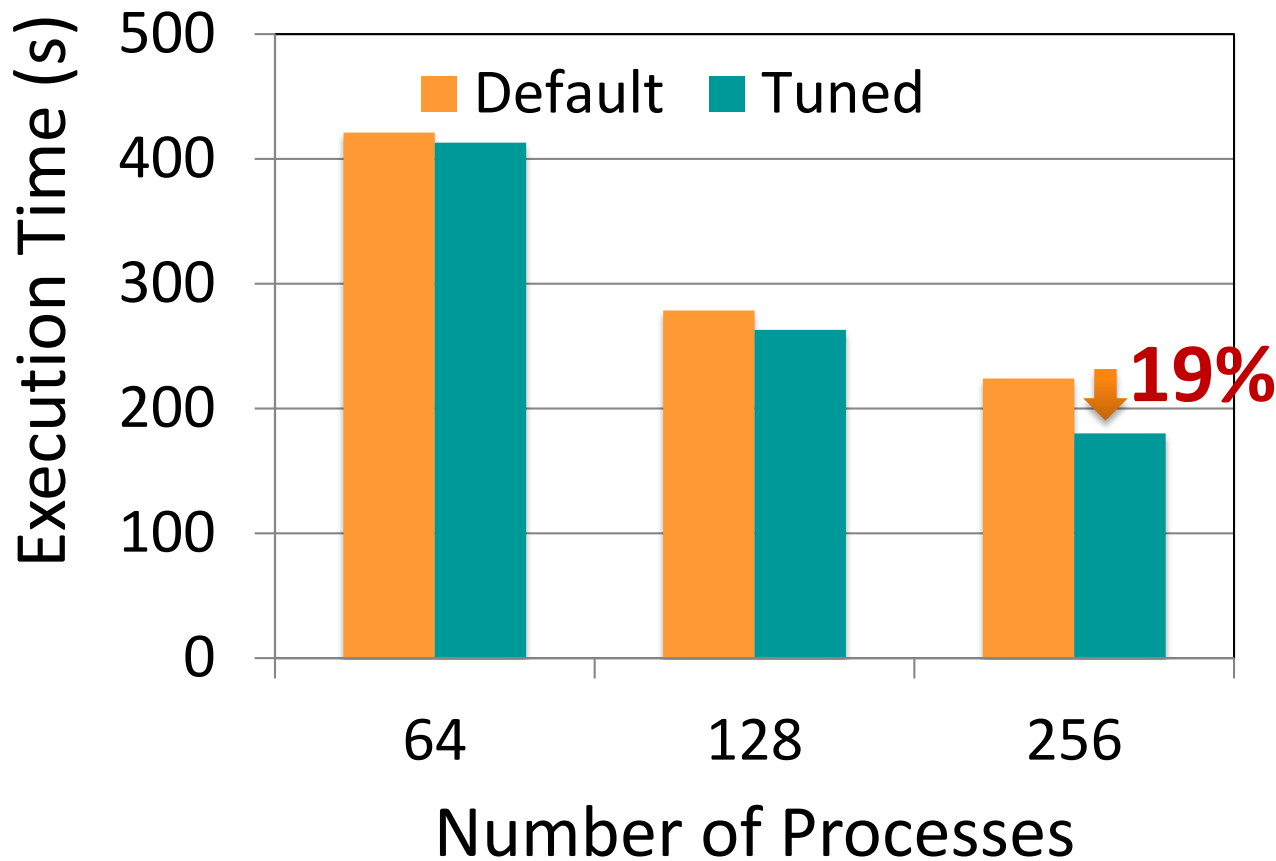
- Available since 2004
- Suite of microbenchmarks to study communication performance of various programming models
- Benchmarks available for the following programming models
  - Message Passing Interface (MPI)
  - Partitioned Global Address Space (PGAS)
    - Unified Parallel C (UPC)
    - Unified Parallel C++ (UPC++)
    - OpenSHMEM
- Benchmarks available for multiple accelerator based architectures
  - Compute Unified Device Architecture (CUDA)
  - OpenACC Application Program Interface
- Part of various national resource procurement suites like NERSC-8 / Trinity Benchmarks
- Continuing to add support for newer primitives and features
- Please visit the following link for more information
  - <http://mvapich.cse.ohio-state.edu/benchmarks/>

# Applications-Level Tuning: Compilation of Best Practices

- MPI runtime has many parameters
- Tuning a set of parameters can help you to extract higher performance
- Compiled a list of such contributions through the MVAPICH Website
  - [http://mvapich.cse.ohio-state.edu/best\\_practices/](http://mvapich.cse.ohio-state.edu/best_practices/)
- Initial list of applications
  - Amber
  - HoomDBlue
  - HPCG
  - Lulesh
  - MILC
  - Neuron
  - SMG2000
  - Cloverleaf
  - SPEC (LAMMPS, POP2, TERA\_TF, WRF2)
- Soliciting additional contributions, send your results to mvapich-help at cse.ohio-state.edu.
- We will link these results with credits to you.



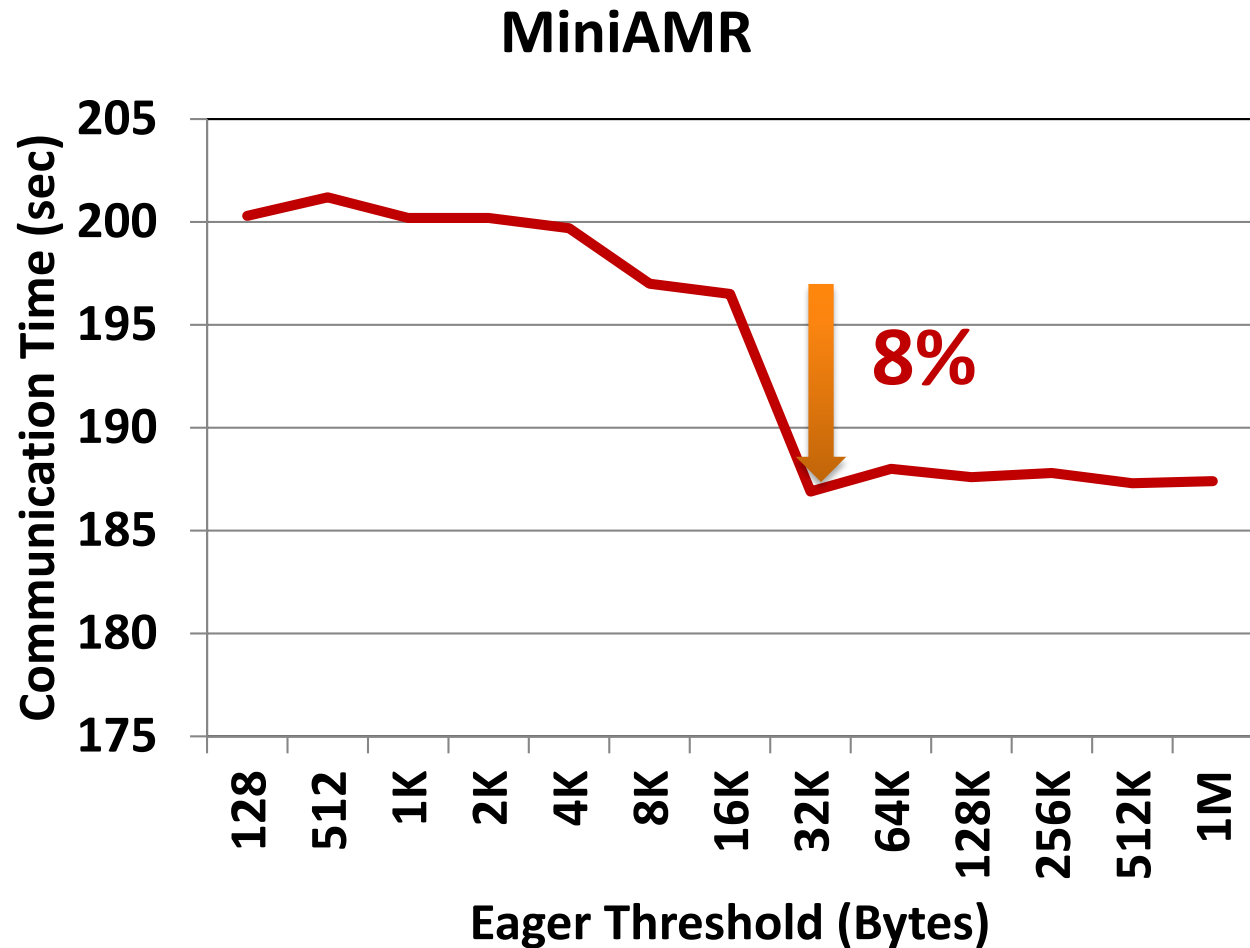
# Amber: Impact of Tuning Eager Threshold



Data Submitted by: Dong Ju Choi @ UCSD

- Tuning the Eager threshold has a significant impact on application performance by avoiding the synchronization of rendezvous protocol and thus yielding better communication computation overlap
- 19% improvement in overall execution time at 256 processes
- Library Version: MVAPICH2 2.2
- MVAPICH Flags used
  - `MV2_IBA_EAGER_THRESHOLD=131072`
  - `MV2_VBUF_TOTAL_SIZE=131072`
- Input files used
  - Small: [MDIN](#)
  - Large: [PMTOP](#)

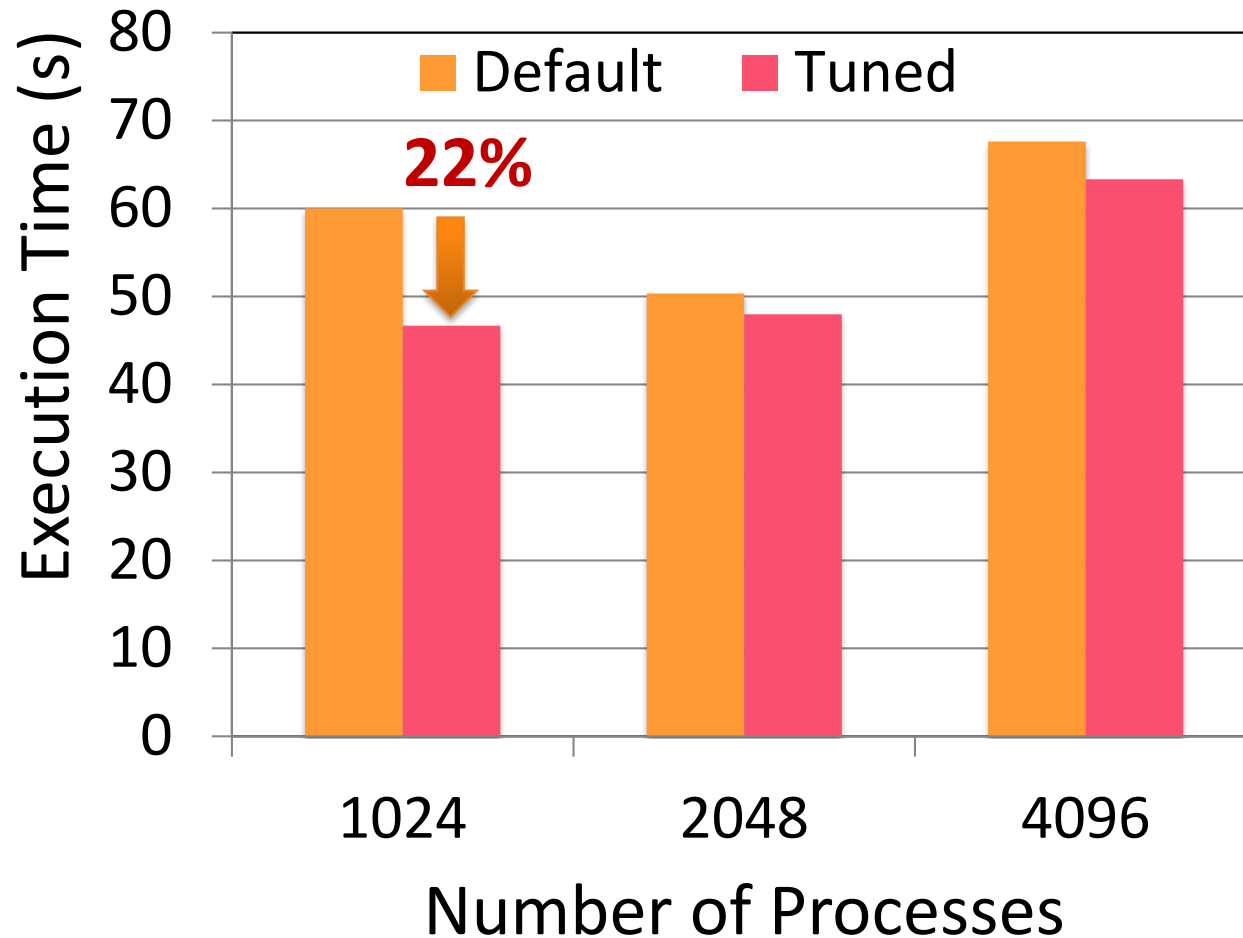
# MiniAMR: Impact of Tuning Eager Threshold



- Tuning the Eager threshold has a significant impact on application performance by avoiding the synchronization of rendezvous protocol and thus yielding better communication computation overlap
- 8% percent reduction in total communication time
- Library Version: MVAPICH2 2.2
- MVAPICH Flags used
  - `MV2_IBA_EAGER_THRESHOLD=32768`
  - `MV2_VBUF_TOTAL_SIZE=32768`

Data Submitted by Karen Tomko @ OSC and Dong Ju Choi @ UCSD

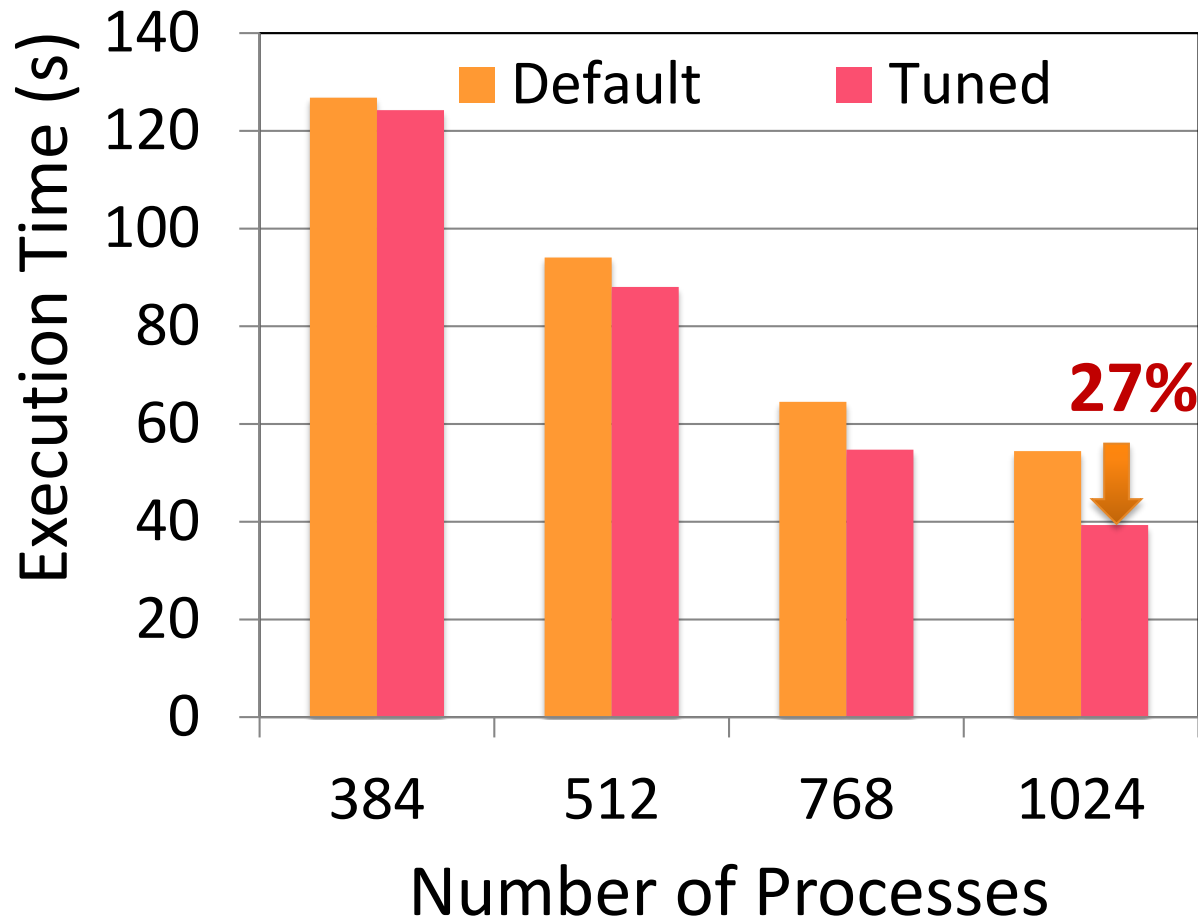
# SMG2000: Impact of Tuning Transport Protocol



Data Submitted by Jerome Vienne @ TACC

- UD-based transport protocol selection benefits the SMG2000 application
- 22% and 6% on 1,024 and 4,096 cores, respectively
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - `MV2_USE_ONLY_UD=1`
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

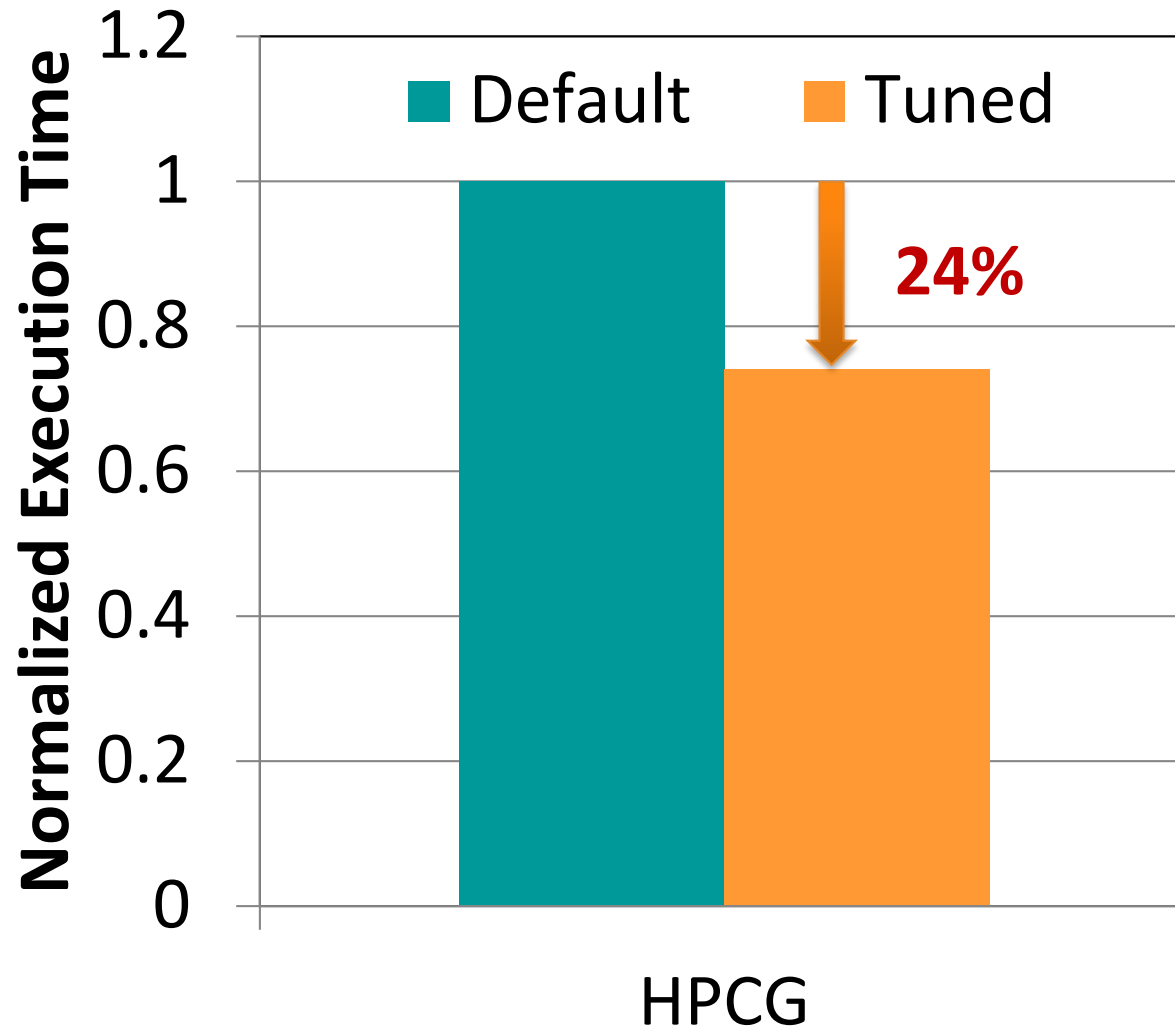
# Neuron: Impact of Tuning Transport Protocol



Data Submitted by Mahidhar Tatineni @ SDSC

- UD-based transport protocol selection benefits the SMG2000 application
- 15% and 27% improvement is seen for 768 and 1,024 processes respectively
- Library Version: MVAPICH2 2.2
- MVAPICH Flags used
  - `MV2_USE_ONLY_UD=1`
- Input File
  - [YuEtAl2012](#)
- System Details
  - Comet@SDSC
  - Haswell nodes with dual 12-cores socket per node and Mellanox FDR (56 Gbps) network.

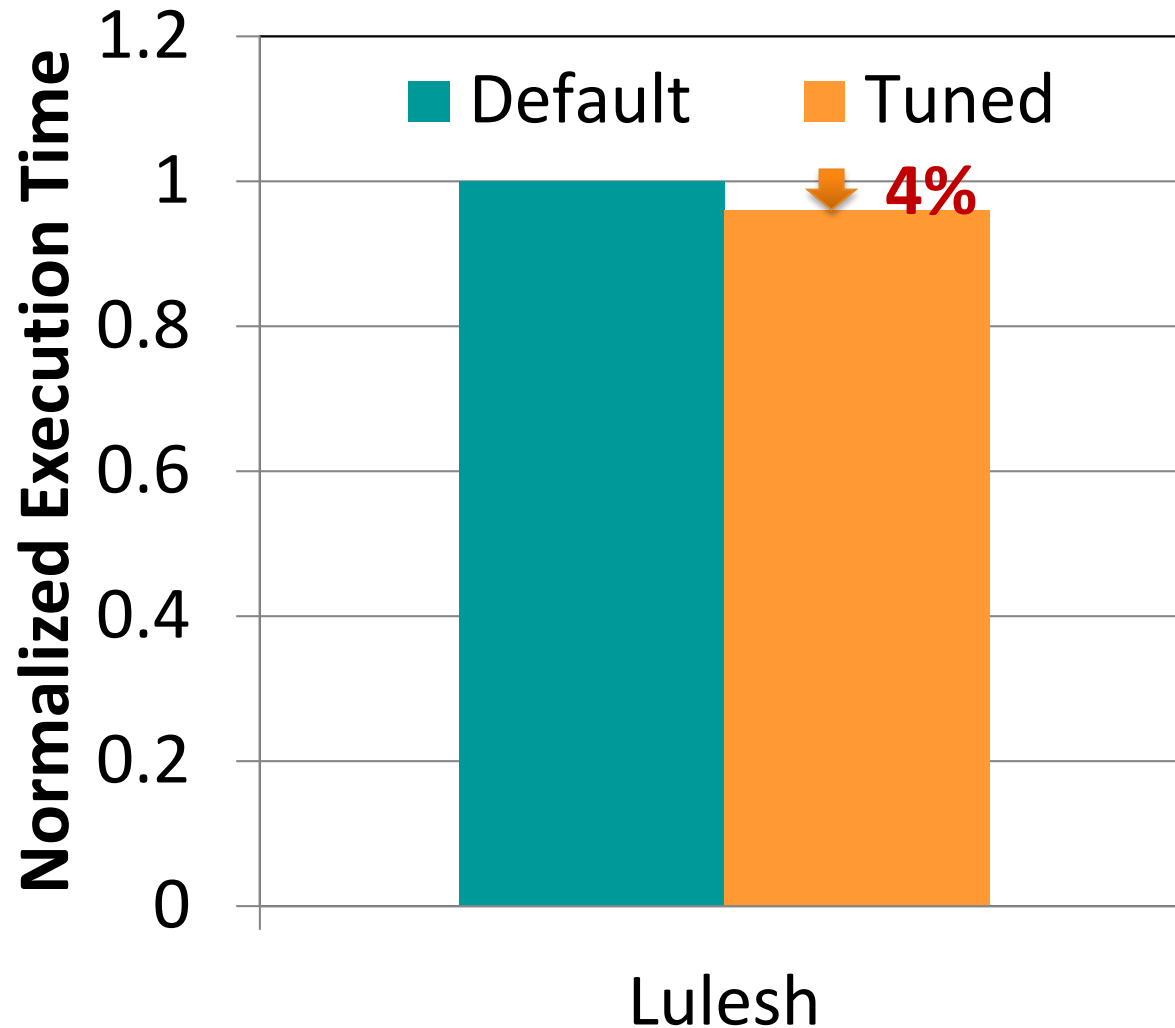
# HPCG: Impact of Collective Tuning for MPI+OpenMP Programming Model



Data Submitted by Jerome Vienne and Carlos Rosales-Fernandez @ TACC

- Partial subscription nature of hybrid MPI+OpenMP programming requires a new level of collective tuning
  - For PPN=2 (Processes Per Node), the tuned version of MPI\_Reduce shows 51% improvement on 2,048 cores
- **24% improvement on 512 cores**
  - 8 OpenMP threads per MPI processes
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - The tuning parameters for hybrid MPI+OpenMP programming models is on by default from MVAPICH2-2.1 onward
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

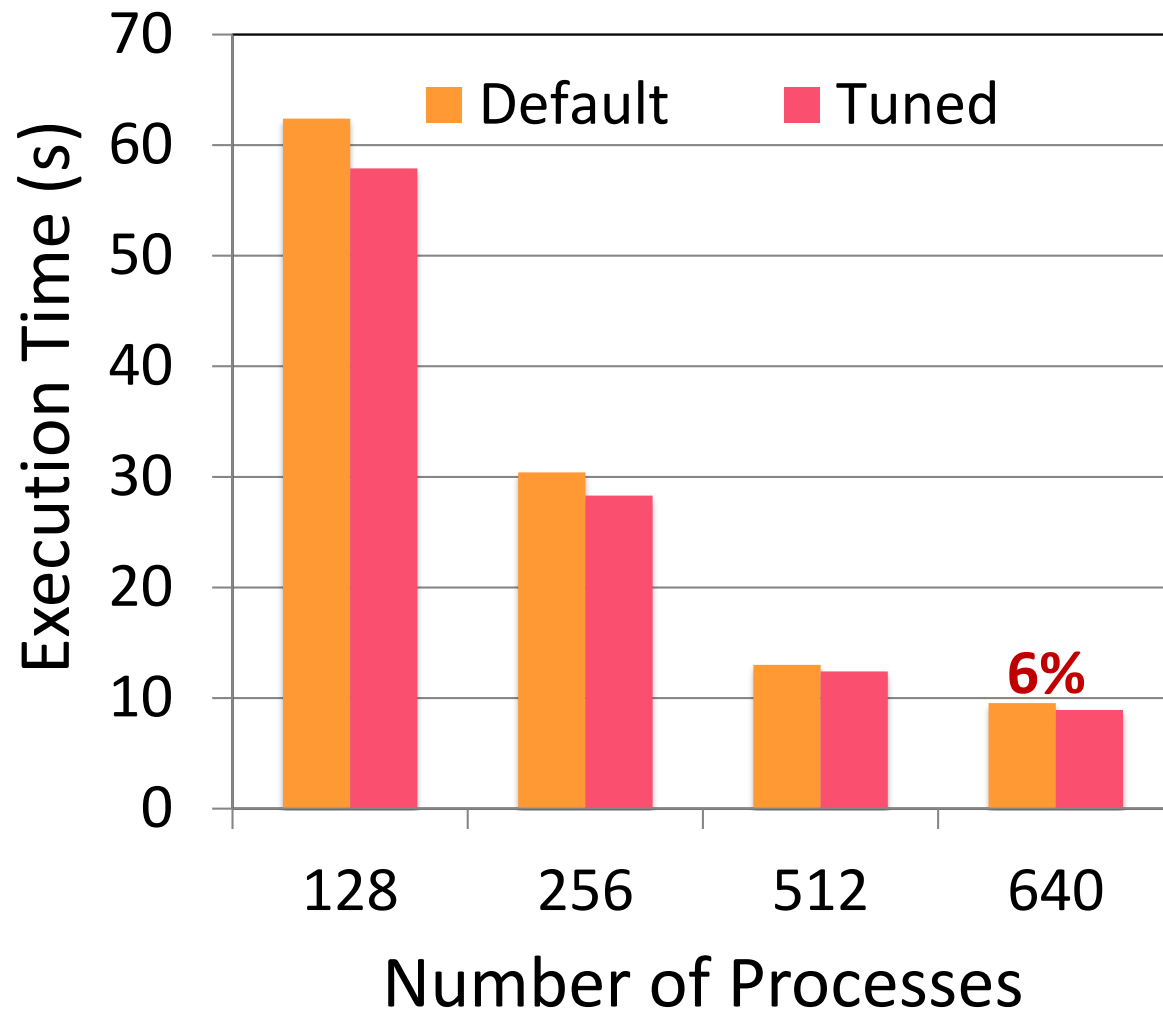
# LULESH: Impact of Collective Tuning for MPI+OpenMP Programming Model



- Partial subscription nature of hybrid MPI+OpenMP programming requires a new level of collective tuning
  - For PPN=2 (Processes Per Node), the tuned version of MPI\_Reduce shows 51% improvement on 2,048 cores
- 4% improvement on 512 cores
  - 8 OpenMP threads per MPI processes
- Library Version: MVAPICH2 2.1
- MVAPICH Flags used
  - The tuning parameters for hybrid MPI+OpenMP programming models is on by default from MVAPICH2-2.1 onward
- System Details
  - Stampede@ TACC
  - Sandybridge architecture with dual 8-cores nodes and ConnectX-3 FDR network

Data Submitted by Jerome Vienne and Carlos Rosales-Fernandez @ TACC

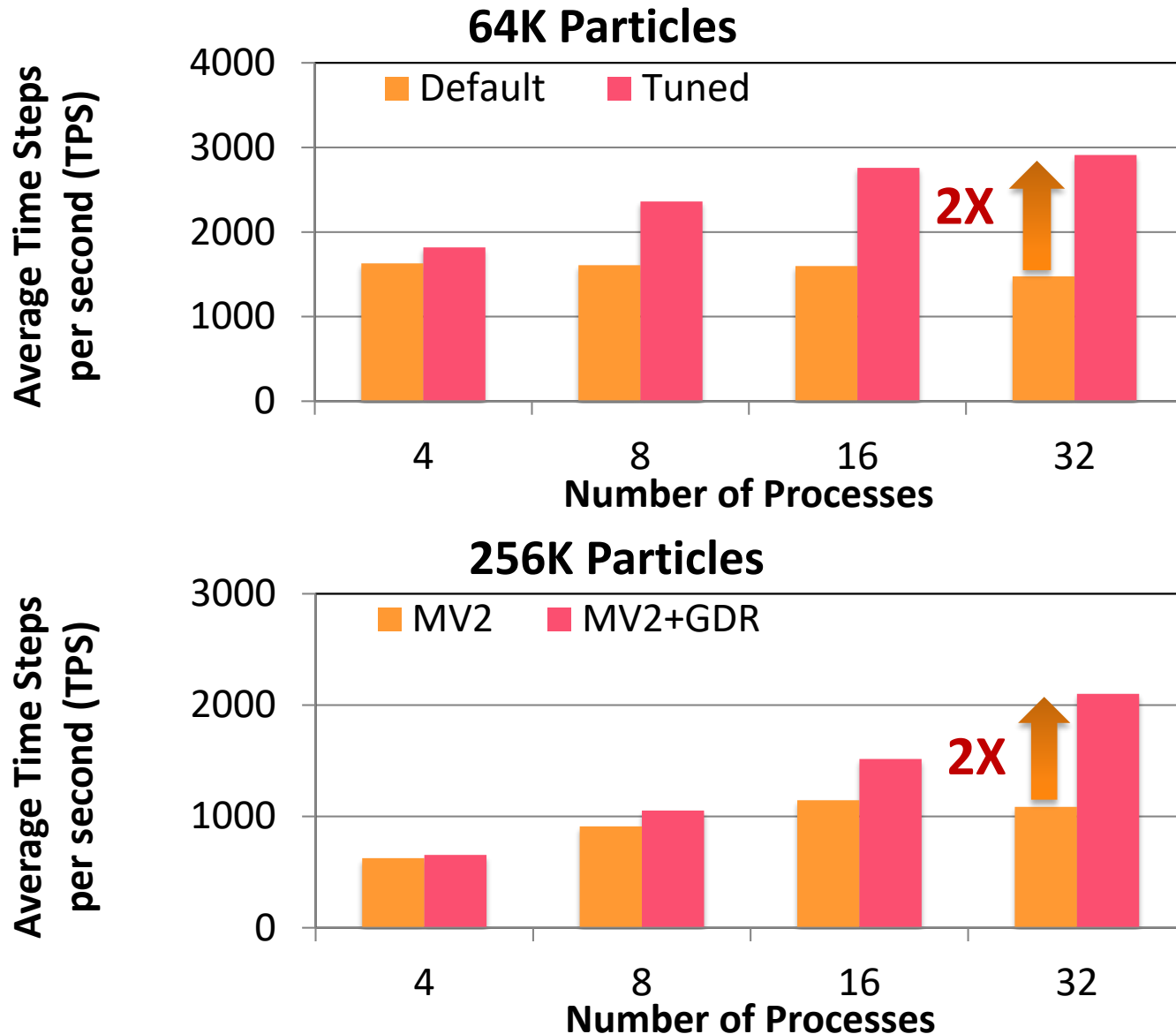
# MILC: Impact of User-mode Memory Registration (UMR) based tuning



Data Submitted by Mingzhe Li @ OSU

- Non-contiguous data processing is very common on HPC applications. MVAPICH2 offers efficient designs for MPI Datatype support using novel hardware features such as UMR
- UMR-based protocol selection benefits the MILC application.
  - 4% and 6% improvement in execution time at 512 and 640 processors, respectively
- Library Version: MVAPICH2-X 2.2
- MVAPICH Flags used
  - `MV2_USE_UMR=1`
- System Details
  - The experimental cluster consists of 32 Ivy Bridge Compute nodes interconnected by Mellanox FDR.
  - The Intel Ivy Bridge processors consist of Xeon dual ten-core sockets operating at 2.80GHz with 32GB RAM and Mellanox OFED version 3.2-1.0.1.1.

# HOOMD-blue: Impact of GPUDirect RDMA Based Tuning

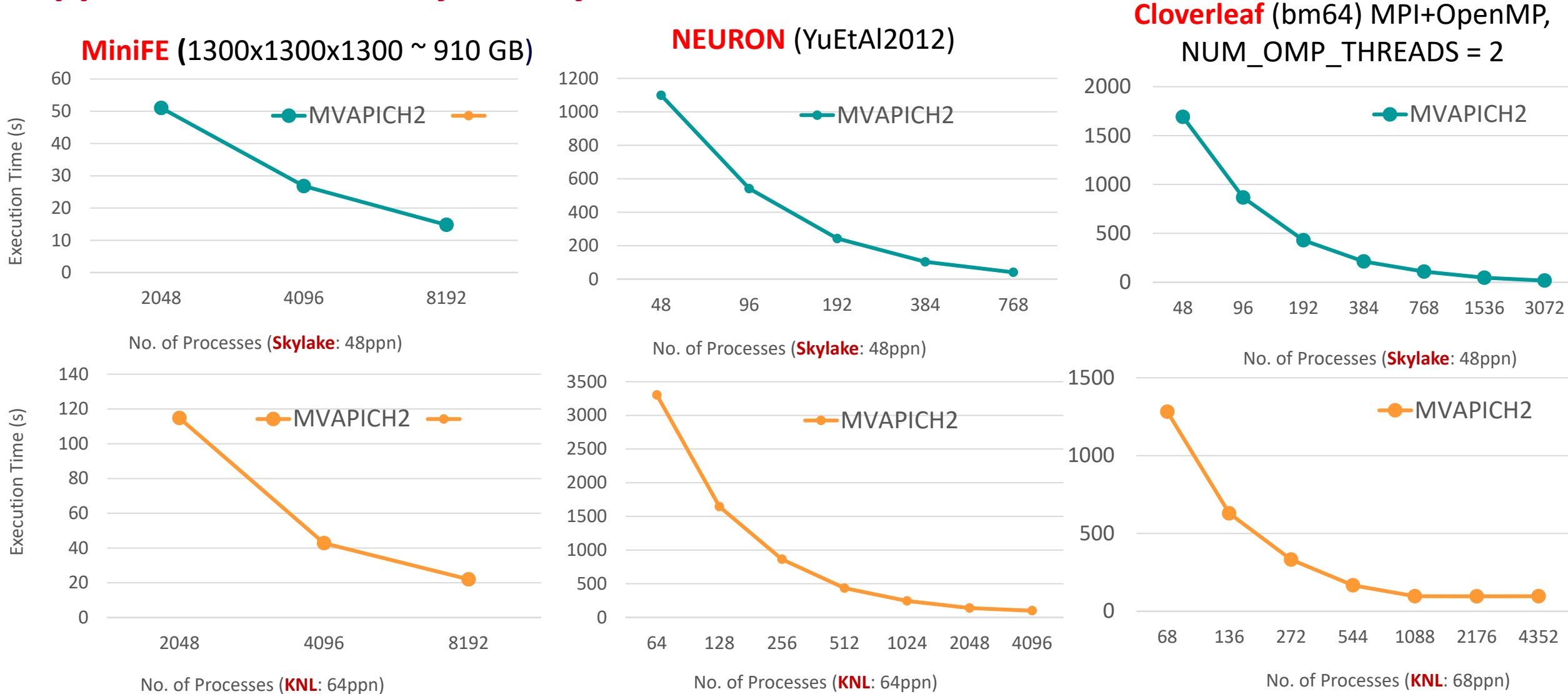


Data Submitted by Khaled Hamidouche @ OSU

- HOOMD-blue is a Molecular Dynamics simulation using a custom force field.
- GPUDirect specific features selection and tuning significantly benefit the HOOMD-blue application. We observe a factor of 2X improvement on 32 GPU nodes, with both 64K and 256K particles
- Library Version: MVAPICH2-GDR 2.2
- MVAPICH-GDR Flags used
  - `MV2_USE_CUDA=1`
  - `MV2_USE_GPUDIRECT=1`
  - `MV2_GPUDIRECT_GDRCOPY=1`
- System Details
  - Wilkes@Cambridge
  - 128 Ivybridge nodes, each node is a dual 6-cores socket with Mellanox FDR



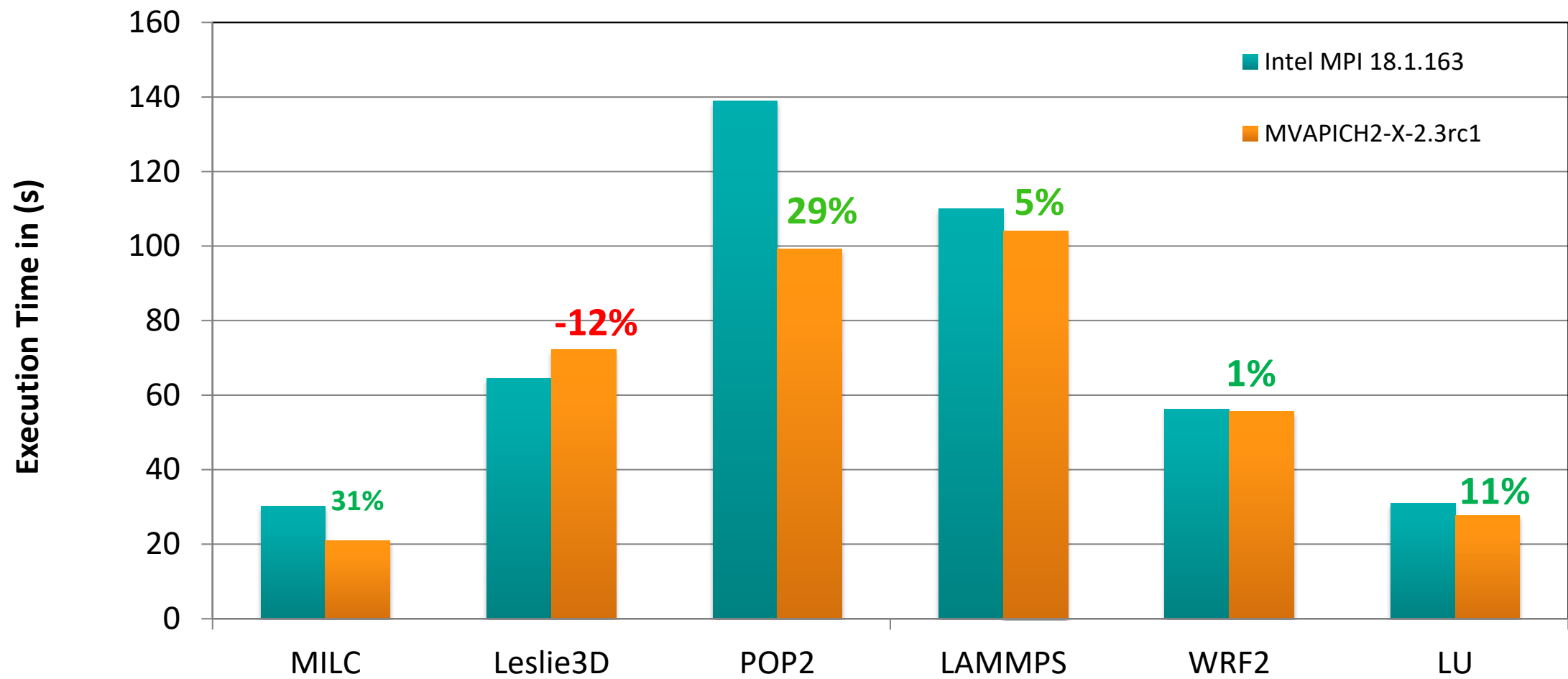
# Application Scalability on Skylake and KNL with Omni-Path



Courtesy: Mahidhar Tatineni @SDSC, Dong Ju (DJ) Choi@SDSC, and Samuel Khuvis@OSC ---- Testbed: TACC Stampede2 using MVAPICH2-2.3b

Runtime parameters: MV2\_SMPI\_LENGTH\_QUEUE=524288 PSM2\_MQ\_RNDV\_SHM\_THRESH=128K PSM2\_MQ\_RNDV\_HFI\_THRESH=128K

# SPEC MPI 2007 Benchmarks: Broadwell + InfiniBand



**MVAPICH2-X outperforms Intel MPI by up to 31%**

Configuration: 448 processes on 16 Intel E5-2680v4 (Broadwell) nodes having 28 PPN and interconnected with 100Gbps Mellanox MT4115 EDR ConnectX-4 HCA

# MVAPICH2 – Plans for Exascale

- Performance and Memory scalability toward 1-10M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF ...)
  - MPI + Task\*
- Enhanced Optimization for GPU Support and Accelerators
- Taking advantage of advanced features of Mellanox InfiniBand
  - Tag Matching\*
  - Adapter Memory\*
- Enhanced communication schemes for upcoming architectures
  - Intel Optane\*
  - BlueField\*
  - CAPI\*
- Extended topology-aware collectives
- Extended Energy-aware designs and Virtualization Support
- Extended Support for MPI Tools Interface (as in MPI 3.0)
- Extended FT support
- Support for \* features will be available in future MVAPICH2 Releases

# Commercial Support for MVAPICH2, HiBD, and HiDL Libraries

- Supported through X-ScaleSolutions (<http://x-scalesolutions.com>)
- Benefits:
  - Help and guidance with installation of the library
  - Platform-specific optimizations and tuning
  - Timely support for operational issues encountered with the library
  - Web portal interface to submit issues and tracking their progress
  - Advanced debugging techniques
  - Application-specific optimizations and tuning
  - Obtaining guidelines on best practices
  - Periodic information on major fixes and updates
  - Information on major releases
  - Help with upgrading to the latest release
  - Flexible Service Level Agreements
- Support provided to Lawrence Livermore National Laboratory (LLNL) for the last two years



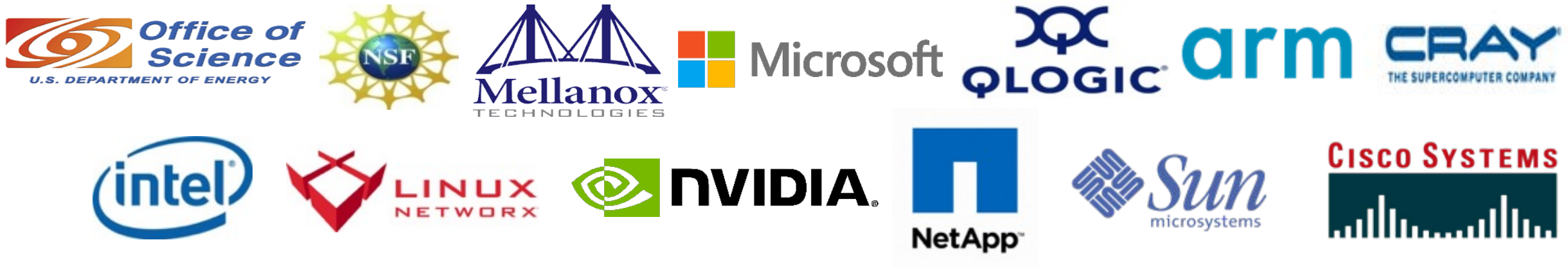
# Silver ISV Member for the OpenPOWER Consortium

- Recently joined the OpenPOWER Consortium as a silver ISV member
- Provides flexibility:
  - To have MVAPICH2, HiDL and HiBD libraries getting integrated into the OpenPOWER software stack
  - A part of the OpenPOWER ecosystem
  - Can participate with different vendors for bidding, installation and deployment process



# Funding Acknowledgments

## *Funding Support by*



## *Equipment Support by*



# Personnel Acknowledgments

## Current Students (Graduate)

- A. Awan (Ph.D.)
- M. Bayatpour (Ph.D.)
- C.-H. Chu (Ph.D.)
- J. Hashmi (Ph.D.)
- A. Jain (Ph.D.)
- K. S. Kandadi (M.S.)
- Kamal Raj (M.S.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- A. Quentin (Ph.D.)
- B. Ramesh (M. S.)
- S. Xu (M.S.)
- Q. Zhou (Ph.D.)

## Past Students

- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- R. Biswas (M.S.)
- S. Bhagvat (M.S.)
- A. Bhat (M.S.)
- D. Buntinas (Ph.D.)
- L. Chai (Ph.D.)
- B. Chandrasekharan (M.S.)
- S. Chakraborty (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- W. Huang (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- K. Kulkarni (M.S.)
- R. Kumar (M.S.)
- S. Krishnamoorthy (M.S.)
- K. Kandalla (Ph.D.)
- M. Li (Ph.D.)
- P. Lai (M.S.)
- J. Liu (Ph.D.)
- M. Luo (Ph.D.)
- A. Mamidala (Ph.D.)
- G. Marsh (M.S.)
- V. Meshram (M.S.)
- A. Moody (M.S.)
- S. Naravula (Ph.D.)
- R. Noronha (Ph.D.)
- X. Ouyang (Ph.D.)
- S. Pai (M.S.)
- S. Potluri (Ph.D.)

## Past Post-Docs

- D. Banerjee
- X. Besseron
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- S. Marcarelli
- J. Vienne
- H. Wang

## Current Research Scientist

- H. Subramoni

## Current Students (Undergraduate)

- V. Gangal (B.S.)
- N. Sarkauskas (B.S.)

## Current Post-doc

- M. S. Ghazimeersaeed
- A. Ruhela
- K. Manian

## Current Research Specialist

- J. Smith

## Past Research Scientist

- K. Hamidouche
- S. Sur
- X. Lu

## Past Programmers

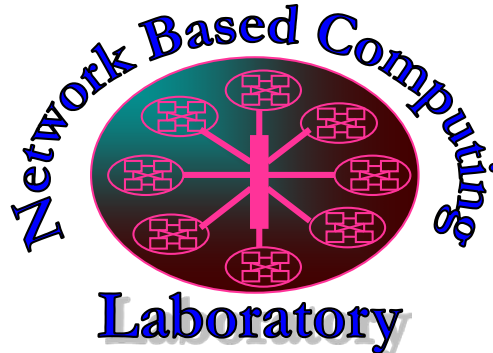
- D. Bureddy
- J. Perkins

## Past Research Specialist

- M. Arnold

# Thank You!

[panda@cse.ohio-state.edu](mailto:panda@cse.ohio-state.edu)



Network-Based Computing Laboratory

<http://nowlab.cse.ohio-state.edu/>



The High-Performance MPI/PGAS Project

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

Follow us on Twitter: @mvapich



High-Performance  
Big Data

The High-Performance Big Data Project

<http://hibd.cse.ohio-state.edu/>



The High-Performance Deep Learning Project

<http://hidl.cse.ohio-state.edu/>