MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Boosting the Performance of HPC Applications with MVAPICH

A Tutorial at MUG'24

Presented by

Hari Subramoni and Nathaniel Shineman

The MVAPICH Team

The Ohio State University

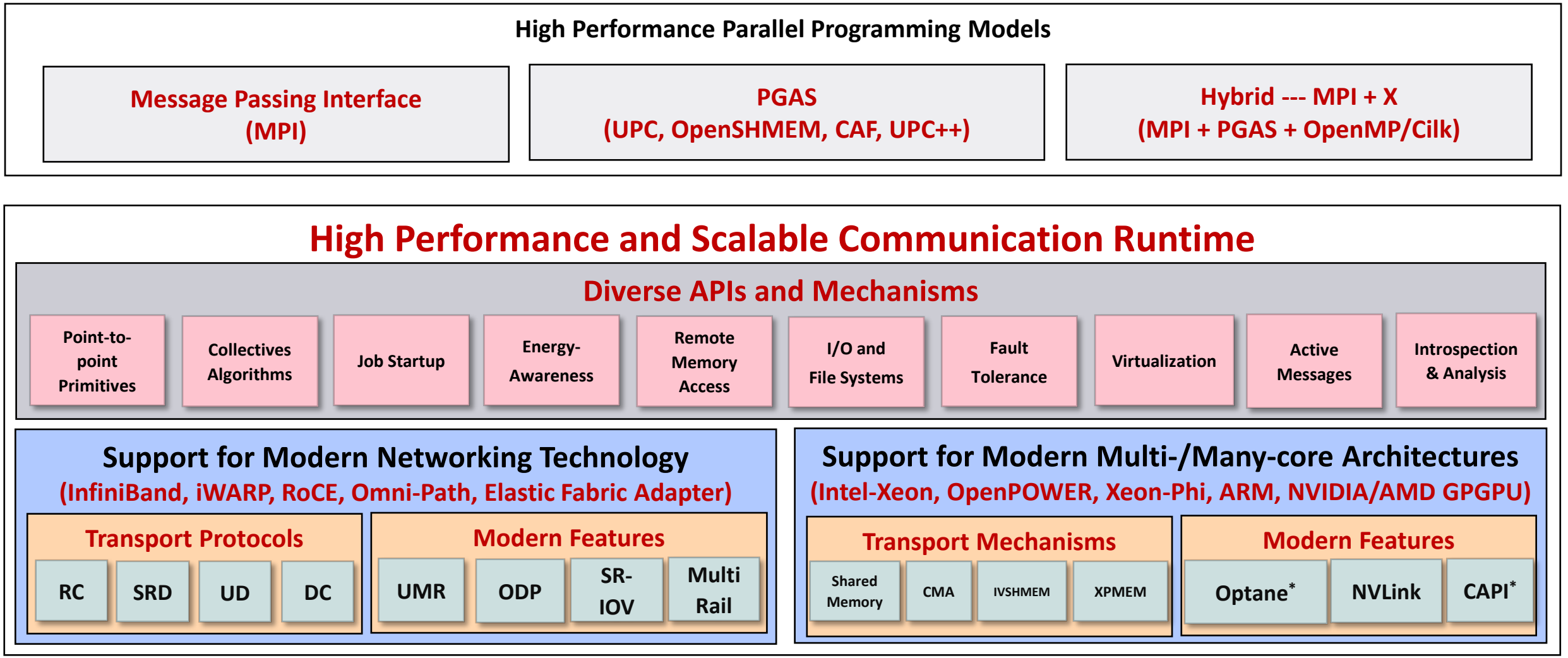
http://mvapich.cse.ohio-state.edu/

Overview of the MVAPICH Project

- High Performance open-source MPI Library
- Support for multiple interconnects
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), AWS EFA, OPX, Broadcom RoCE, Intel Ethernet, Rockport Networks, Slingshot 10/11
- Support for multiple platforms
  - x86, OpenPOWER, ARM, Xeon-Phi, GPGPUs (NVIDIA and AMD)
- Started in 2001, first open-source version demonstrated at SC '02
- Supports the latest MPI-3.1 standard
- http://mvapich.cse.ohio-state.edu
- Additional optimized versions for different systems/environments:
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA (since 2014) and AMD (since 2020) GPUs
  - MVAPICH2-MIC with support for Intel Xeon-Phi, since 2014
  - MVAPICH2-Virt with virtualization support, since 2015
  - MVAPICH2-EA with support for Energy-Awareness, since 2015
  - MVAPICH2-Azure for Azure HPC IB instances, since 2019
  - MVAPICH2-X-AWS for AWS HPC+EFA instances, since 2019
- Tools:
  - OSU MPI Micro-Benchmarks (OMB), since 2003
  - OSU InfiniBand Network Analysis and Monitoring (INAM), since 2015

- **Used by more than 3,400 organizations in 92 countries**
- **More than 1.81 Million downloads from the OSU site directly**
- Empowering many TOP500 clusters (June '24 ranking)
  - 13th , 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 33rd, 448, 448 cores (Frontera) at TACC
  - 57th, 288,288 cores (Lassen) at LLNL
  - and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 29th ranked TACC Frontera system
- **Empowering Top500 systems for more than 18 years**

Architecture of MVAPICH Software Family for HPC and DL/ML



* Upcoming

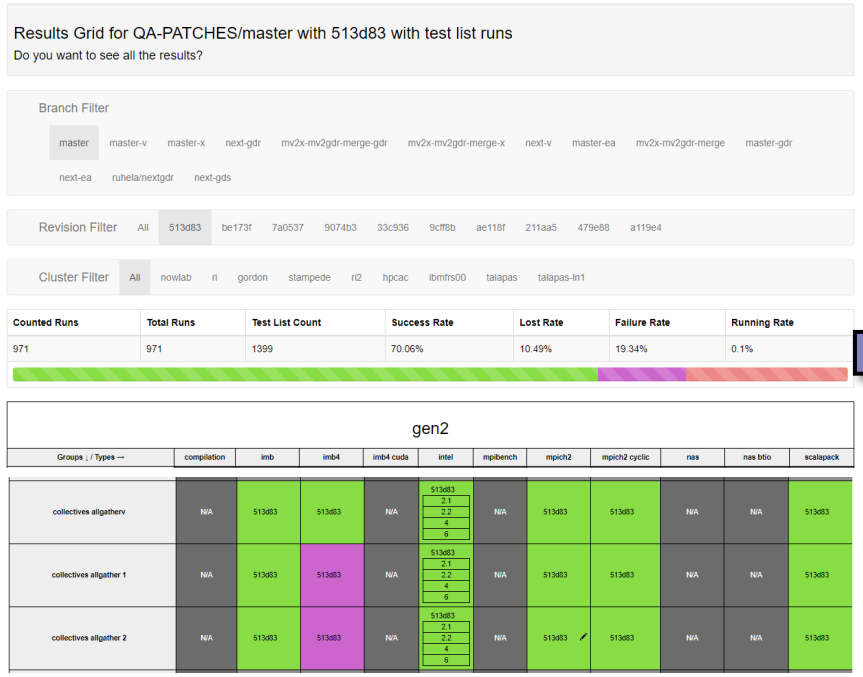
Production Quality Software Design, Development and Release

- Rigorous Q&A procedure before making a release
  - Exhaustive unit testing
  - Various test procedures on diverse range of platforms and interconnects
  - Test 19 different benchmarks and applications including, but not limited to
    - OMB, IMB, MPICH Test Suite, Intel Test Suite, NAS, Scalapak, and SPEC
  - Spend about 18,000 core hours per commit
  - Performance regression and tuning
  - Applications-based evaluation
  - Evaluation on large-scale systems
- All versions (alpha, beta, RC1 and RC2) go through the above testing

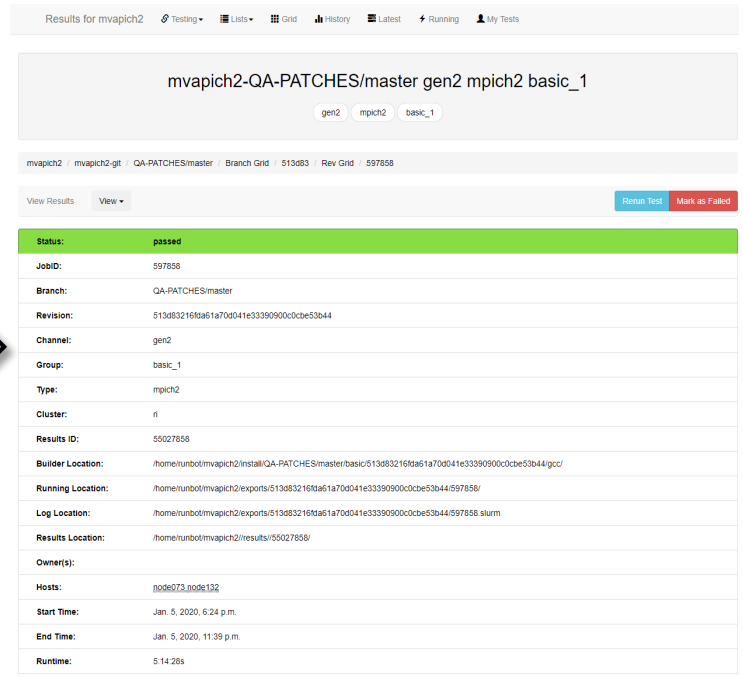
Automated Procedure for Testing Functionality

- Test OMB, IMB, MPICH Test Suite, Intel Test Suite, NAS, Scalapak, and SPEC
- Tests done for each build done build “buildbot”
- Test done for various combinations of *environment variables* meant to trigger different communication paths in MVAPICH

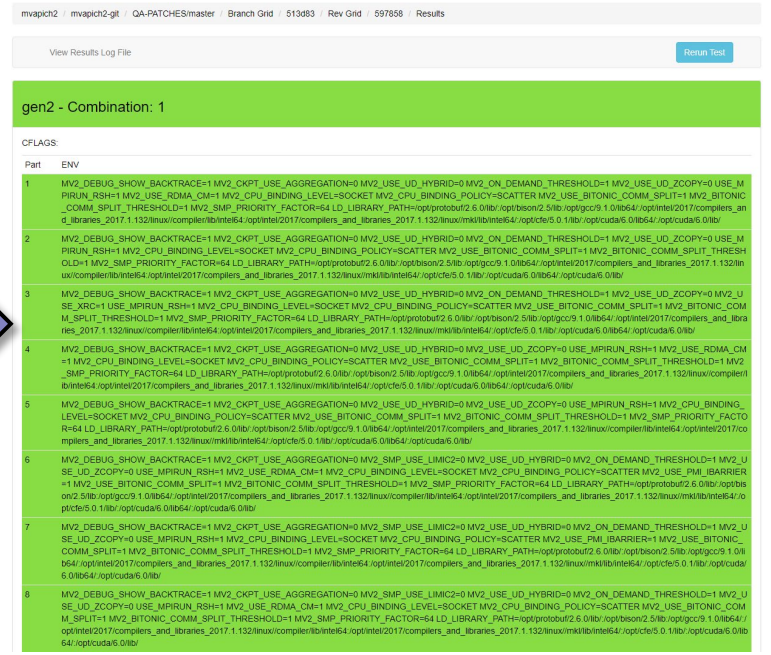
Summary of all tests for one commit



Summary of an individual test



Details of individual combinations in one test



Scripts to Determine Performance Regression

- Automated method to identify performance regression between different commits
- Tests different MPI primitives
  - Point-to-point; Collectives; RMA
- Works with different
  - Job Launchers/Schedulers
    - SLURM, PBS/Torque, JSM
  - Works with different interconnects
- Works on multiple HPC systems
- Works on CPU-based and GPU-based systems

Performance regression of mvapich2-2.3rc2-x-3e5551 and mvapich2-masterx-2950c8 on FRONTERA (cascadelake architecture) Thu Aug 15 09:23:48 CDT 2019

OLD_TUNEVAR=

NEW_TUNEVAR=

Legend

Dark Green : Performance of mvapich2-masterx-2950c8 is more than 5 % better than mvapich2-2.3rc2-x-3e5551

Light Green : Performance of mvapich2-masterx-2950c8 is less than 5 % better than mvapich2-2.3rc2-x-3e5551

Grey : Performance of mvapich2-masterx-2950c8 is same as mvapich2-2.3rc2-x-3e5551

Light Red : Performance of mvapich2-masterx-2950c8 is less than 5 % worse compared to mvapich2-2.3rc2-x-3e5551

Dark Red : Performance of mvapich2-masterx-2950c8 is more than 5 % worse compared to mvapich2-2.3rc2-x-3e5551

Inter-node

	1	2	4	8	16	32	64	128	256	512	1K	2K	4K	8K
getbw	4.12 4.24 2 %	8.13 8.52 4 %	16.36 17.15 4 %	32.57 34.35 5 %	65.19 68.65 5 %	133.07 137.11 2 %	254.43 268.90 5 %	512.64 529.51 3 %	1013.21 1047.48 3 %	1959.28 2034.04 3 %	3806.50 3888.42 2 %	6495.99 6540.39 0 %	7935.33 8009.98 0 %	10118.37 10075.08 0 %
putbibw	7.32 7.18 -1 %	14.79 14.52 -1 %	29.56 29.05 -1 %	58.98 58.02 -1 %	117.73 116.10 -1 %	219.70 221.33 0 %	438.70 442.21 0 %	862.87 852.53 -1 %	1821.56 1795.21 -1 %	3505.96 3539.99 0 %	6728.97 6714.72 0 %	11697.93 11634.75 0 %	15297.26 15186.84 0 %	18873.10 18919.87 0 %
putbw	4.22 4.32 2 %	8.41 8.64 2 %	17.11 17.43 1 %	34.13 34.84 2 %	68.41 69.74 1 %	127.38 132.60 3 %	253.67 265.58 4 %	494.66 509.46 2 %	1051.14 1080.49 2 %	1987.26 2079.78 4 %	3801.76 3978.30 4 %	6357.54 6438.37 1 %	8422.30 8478.65 0 %	9972.75 10031.73 0 %
acclat	2.30 2.30 0 %	2.30 2.30 0 %	2.30 2.30 0 %	2.31 2.31 0 %	2.47 2.50 -1 %	2.51 2.53 0 %	2.91 2.93 0 %	2.97 2.99 0 %	3.09 3.11 0 %	3.33 3.35 0 %	3.82 3.85 0 %	4.79 4.81 0 %	7.03 6.96 0 %	10.76 10.79 0 %
getlat	1.96 1.96 0 %	1.97 1.96 0 %	1.96 1.95 0 %	1.97 1.95 1 %	1.97 1.95 1 %	1.97 1.95 1 %	1.97 1.95 1 %	2.01 2.01 0 %	2.07 2.07 0 %	2.11 2.13 0 %	2.23 2.24 0 %	2.51 2.50 0 %	3.07 3.12 -1 %	3.79 3.76 0 %
putlat	1.59 1.59 0 %	1.58 1.59 0 %	1.59 1.58 0 %	1.59 1.58 0 %	1.59 1.59 0 %	1.63 1.63 0 %	1.62 1.62 0 %	1.64 1.64 0 %	1.98 1.97 0 %	2.03 2.02 0 %	2.12 2.11 0 %	2.33 2.32 0 %	2.85 2.87 0 %	3.58 3.62 -1 %
lat	1.09 1.10 0 %	1.12 1.12 0 %	1.12 1.12 0 %	1.12 1.12 0 %	1.12 1.12 0 %	1.16 1.16 0 %	1.16 1.16 0 %	1.18 1.18 0 %	1.23 1.23 0 %	1.64 1.63 0 %	1.73 1.72 0 %	1.89 1.90 0 %	2.27 2.26 0 %	3.30 3.16 4 %
bibw	0.01 6.02 99 %	8.84 12.41 28 %	17.67 24.90 29 %	35.26 50.43 30 %	69.95 102.34 31 %	139.10 209.77 33 %	272.80 441.25 38 %	527.61 906.82 41 %	1014.76 1649.39 38 %	1906.89 2984.43 36 %	3512.06 5576.41 37 %	5983.62 7606.62 21 %	8808.00 2974.96 -196 %	12102.21 11577.63 -4 %
bw	4.20 5.62 25 %	8.65 11.37 23 %	18.55 22.70 18 %	37.12 45.17 17 %	70.05 88.18 20 %	140.56 169.93 17 %	277.59 361.68 23 %	535.34 759.80 29 %	992.70 1471.40 32 %	1834.85 2513.06 26 %	3403.25 3894.56 12 %	5539.04 5642.22 1 %	7134.90 7440.25 4 %	9246.59 9152.05 -1 %
mbw_mr	4643958.72 5611922.55 17 %	4652819.64 5672494.10 17 %	4660101.58 5656039.25 17 %	4651994.45 5642174.30 17 %	4408316.86 5511754.90 20 %	4386219.34 5312786.53 17 %	4343210.34 5624117.68 22 %	4181690.38 5945467.49 29 %	3961422.35 5710061.67 30 %	3582486.69 4904271.86 26 %	3358575.31 3815154.75 11 %	2709562.74 2757052.57 1 %	1751091.52 1816130.44 3 %	1128687.30 1117241.92 -1 %

128 process collective tests

	1	2	4	8	16	32	64	128	256	512	1K	2K	4K	8K	16K	32K	64K	128K	256K	512K	1M
osu_allgather	21.83 17.34 20 %	23.72 16.16 31 %	22.11 17.28 21 %	24.40 17.87 26 %	28.29 19.89 29 %	38.99 21.65 44 %	28.70 26.24 8 %	47.85 43.90 8 %	66.45 63.03 5 %	105.64 102.18 3 %	914.13 177.28 80 %	2330.43 276.96 88 %	726.91 502.60 30 %	1197.84 1272.69 -6 %	2460.79 2555.89 -3 %	3996.50 3247.50 18 %	6274.14 5516.63 12 %	13385.82 10428.17 22 %	20382.71 20449.20 0 %	43029.04 43000.04 0 %	102436.87 88526.72 13 %
osu_allgatherv	25.66 21.91 14 %	27.61 22.04 20 %	25.41 19.47 23 %	27.68 20.73 25 %	31.34 23.41 25 %	40.98 25.06 38 %	78.11 29.32 62 %	133.69 50.28 62 %	207.27 133.06 35 %	505.08 233.32 53 %	529.16 239.41 54 %	558.27 284.32 49 %	689.62 525.36 23 %	1176.09 1254.16 -6 %	2461.61 2524.56 -2 %	3898.46 3167.98 18 %	6573.11 5468.24 16 %	11721.43 10026.13 14 %	21588.64 20060.80 7 %	43244.00 43166.45 0 %	102924.40 87885.28 14 %
osu_allreduce	36.82 19.08 48 %	32.90 22.28 32 %	32.66 19.37 40 %	33.15 19.37 41 %	35.42 21.29 39 %	36.90 27.06 26 %	41.44 22.96 44 %	51.83 29.01 44 %	69.81 37.88 45 %	107.22 60.36 43 %	48.50 44.66 7 %	51.63 48.42 6 %	86.36 84.79 1 %	120.66 109.31 9 %	184.02 172.09 6 %	296.64 267.65 9 %	521.36 469.93 9 %	1996.51 967.96 51 %	3221.70 2649.74 17 %	6665.53 6305.41 5 %	9984.70 8972.08 10 %

Deployment Solutions: RPM and Debian Deployments

- Provide customized RPMs for different system requirements
  - ARM, Power8, Power9, x86 (Intel and AMD)
  - Different versions of Compilers (ICC, PGI, GCC, XLC, ARM), CUDA, OFED/Intel IFS

Deploy to Azure

MVAPICH2-X 2.3 Library and User Guide

- The MVAPICH2-X 2.3 library is distributed under the BSD License.
- OSU MVAPICH2-X 2.3 (06/10/20), ABI compatible with MPICH-3.2.1.
  - CHANGELOG for MVAPICH2-X 2.3
  - Patch to add PMI Extensions with SLURM 15
  - Patch to add PMI Extensions with SLURM 16
  - Patch to add PMI Extensions with SLURM 17
- MVAPICH2-X User Guide: A detailed user guide with instructions to install MVAPICH2-X and execute MPI/UPC/UPC++/OpenSHMEM/CAF/Hybrid programs is availab (HTML, PDF)
- **Installation using Spack:** A detailed user guide with instructions to install MVAPICH2-X using Spack is available here.
- **Installation Guide**
  - These tarballs contain the MVAPICH2-X software for Redhat and Debian based systems combined together in one combined package.
  - Running the install.sh script in the tarball will install the libraries.
  - These RPMs are relocatable and advanced users may skip the install.sh script to directly use alternate commands to install the desired RPMs.
- **Which RPM should I Install?**
  - InfiniBand / RoCE System

Features for InifiniBand (OFA-IB-CH3) and ROCE (OFA-RoCE-CH3)	Basic	Basic-XPMEM	Advanced	Advanced-XPMEM
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	✓	✓	✓	✓
Cooperative Rendezvous Protocols	✓	✓	✓	✓
Optimized Asynchronous Progress*	✓	✓	✓	✓
InfiniBand Hardware Multicast-based MPI_Bcast*+	✓	✓	✓	✓

MVAPICH2-GDR 2.3.6 Library

- The MVAPICH2-GDR library is distributed under the BSD License.
- OSU MVAPICH2-GDR 2.3.6 (8/12/2021), ABI compatible with MPICH-3.2.1.
  - CHANGELOG for MVAPICH2-GDR 2.3.6.
- MVAPICH2-GDR User Guide: A detailed user guide with instructions to build, install MVAPICH2-GDR and execute MPI programs over GPU buffers is available.
- **Installation using Spack:** A detailed user guide with instructions to install MVAPICH2-GDR using Spack is available here.
- These RPMs contain the MVAPICH2-GDR software on the corresponding distro. **Please note that the RHEL RPMs are compatible with CentOS as well. For Debian/Ubuntu users, please follow the instructions in the install section in the userguide.**

OpenPOWER RPMs

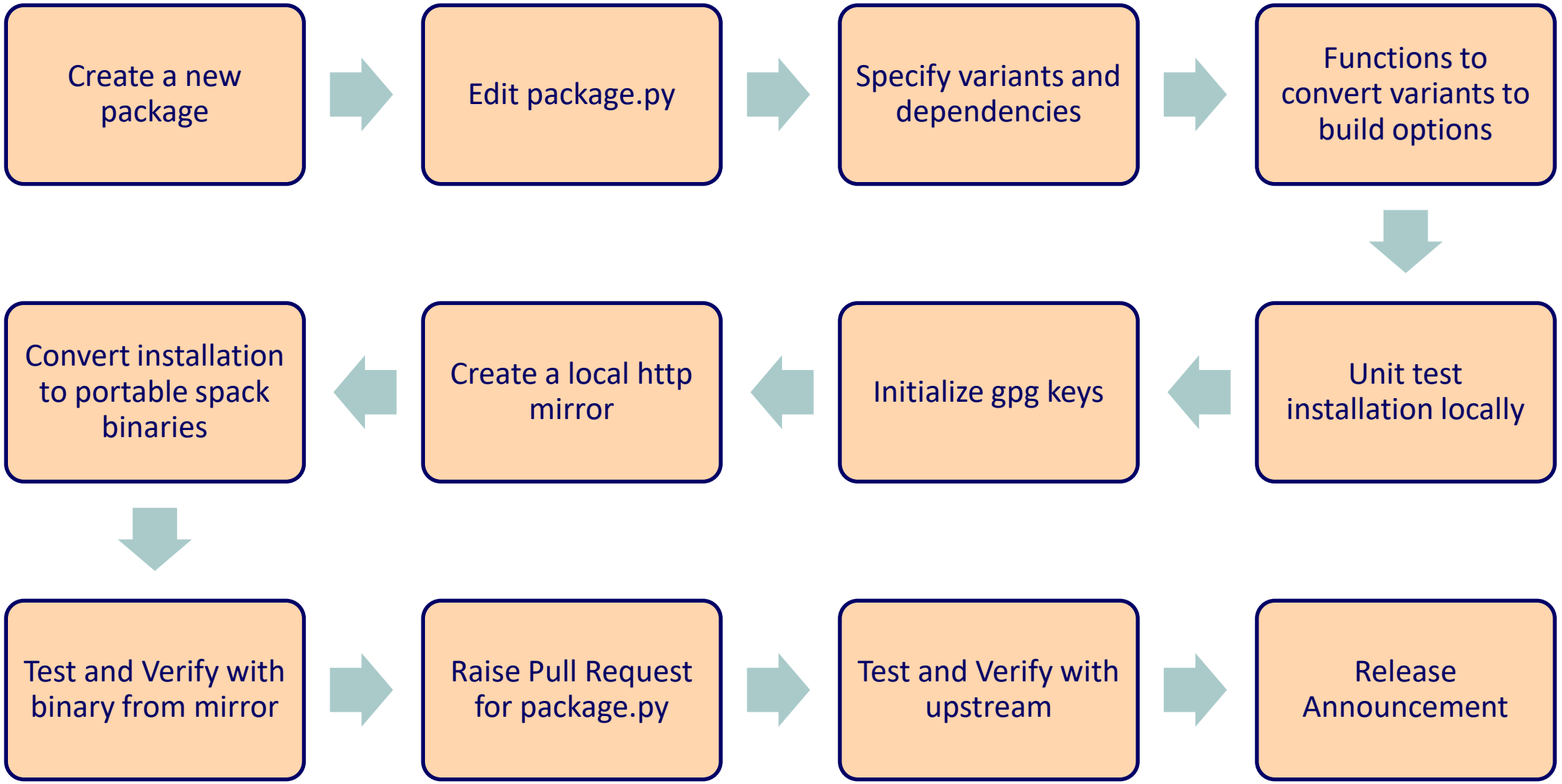
	GNU 4.9.3	GNU 4.9.3 (w/ jsrun)	GNU 7.3.1	GNU 7.3.1 (w/ jsrun)	GNU 8.3.1	GNU 8.3.1 (w/ jsrun)	XLC 16.01	XLC 16.01 (w/ jsrun)
MLNX-OFED 4.7(Lassen/Sierra)	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.0] [CUDA 11.1] [CUDA 11.2]

	GNU 4.8.5	GNU 4.8.5 (w/jsrun)	GNU 7.4.0	GNU 7.4.0 (w/jsrun)	XLC 16.01	XLC 16.01 (w/jsrun)
MLNX-OFED 4.7(Summit)	[CUDA 10.2] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.2]	[CUDA 10.2] [CUDA 11.2]	[CUDA 10.1] [CUDA 11.2]	[CUDA 10.1] [CUDA 11.2]

x86 RPMs

CUDA

Deployment Solutions: Spack Workflow



Deployment Solutions: Installation and Setup MVAPICH from Spack

Install Spack

$ git clone https://github.com/spack/spack.git

$ source ~/spack/share/spack/setup-env.sh

Installing MVAPICH (From Source)

$ spack info mvapich

$ spack install mvapich@3.0a %gcc@8.3.0 (or other available compiler on the system)

$ spack find -l -v -p mvapich

Deployment Solutions: MVAPICH2-X or MVAPICH2-GDR

Add the required mirrors

$ spack mirror add MVAPICHx http://mvapich.cse.ohio-state.edu/download/mvapich/spack-mirror/MVAPICHx

$ spack mirror add MVAPICH2-GDR http://mvapich.cse.ohio-state.edu/download/mvapich/spack-mirror/MVAPICH2-GDR

Trust the public key used to sign the packages

$ wget http://mvapich.cse.ohio-state.edu/download/mvapich/spack-mirror/MVAPICHx/build_cache/public.key

$ spack gpg trust public.key

Deployment Solutions: MVAPICH2-X or MVAPICH2-GDR from Spack

List the available binaries in the mirror

```
$ spack buildcache list -L -v -a
```

Install MVAPICH2-X and MVAPICH2-GDR

```
$ spack install MVAPICHx@2.3%gcc@4.8.5 distribution=mofed4.6 feature=advanced-xpmem
pmi_version=pmi1 process_managers=mpirun target=x86_64
```

```
$ spack install MVAPICH2-GDR@2.3.3~core_direct+mcast~openacc distribution=mofed4.5
pmi_version=pmi1 process_managers=mpirun ^cuda@9.2.88 target=x86_64
```

Supported CUDA Versions

- ^cuda@9.2.88, ^cuda@10.1.243, ^cuda@10.2.89

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

MVAPICH Software Family

Requirements	Library
MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)	MVAPICH
Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis), PGAS (OpenSHMEM, UPC, UPC++, and CAF), MPI+PGAS (OpenSHMEM, UPC, UPC++, and CAF) with IB and RoCE (v1/v2)	MVAPICH2-X
Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications	MVAPICH2-GDR
Advanced MPI with unified MVAPICH2-GDR and MVAPICH2-X features for HPC, DL, ML, Big Data and Data Science applications	MVAPICH-PLUS

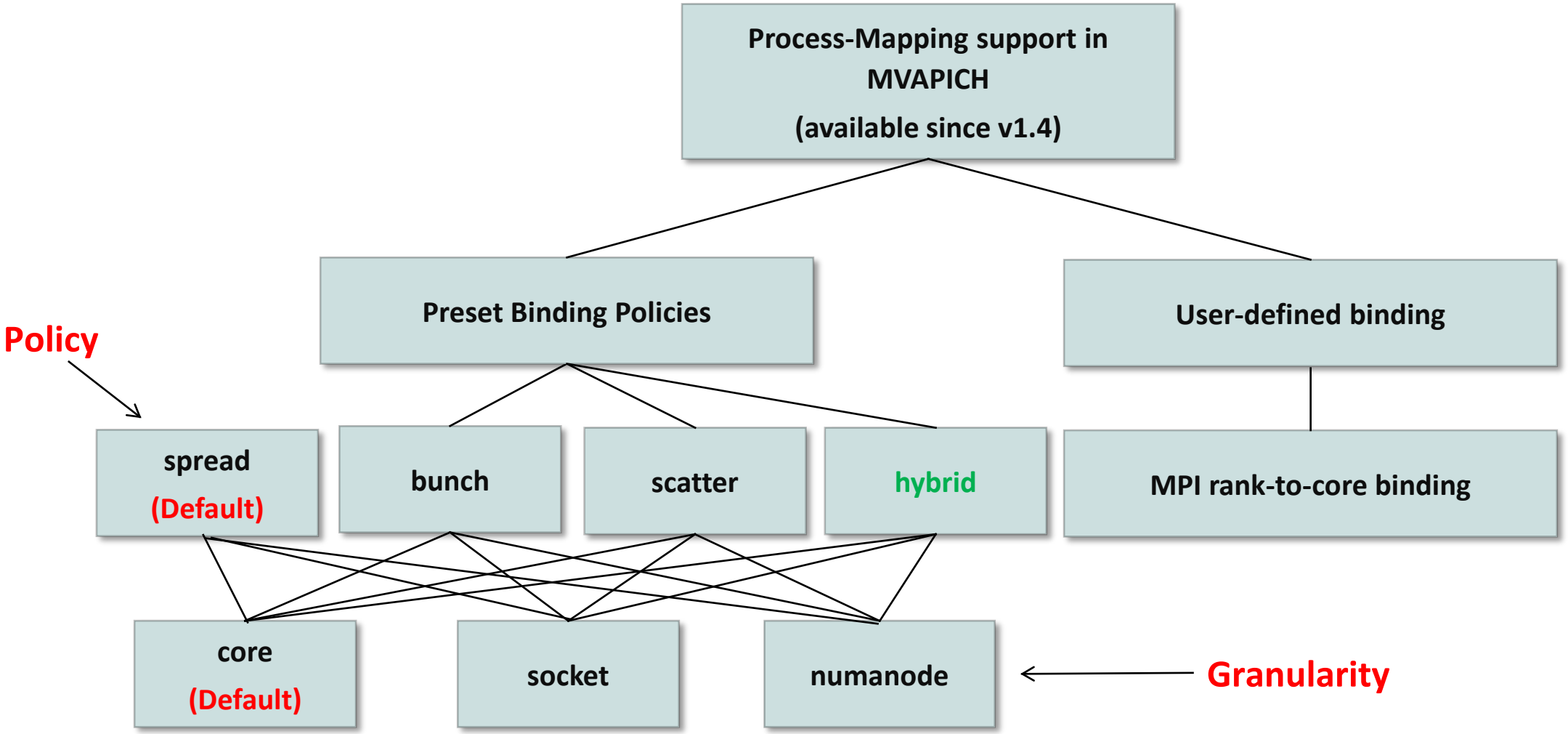
MVAPICH 3.0

- **Full version released 02/22/24**
- Major Features and Enhancements
  - Based on MPICH 3.4.3
  - Support for OFI and UCX network libraries through MPICH netmod interface
  - MVAPICH enhanced collective designs
  - Support for Cray Slingshot 11 network
  - Improved CVAR interface for consistency and performance
  - Support for SHARP

Overview of MVAPICH Features

- Process Mapping and Point-to-point Intra-node Protocols
- Enhanced Collective Communication Designs
- MVAPICH 3.0 New Features

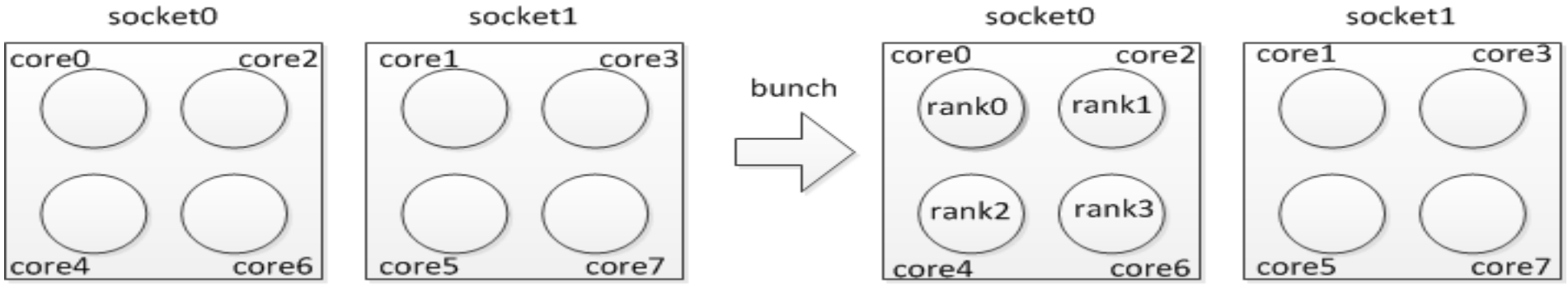
Process Mapping support in MVAPICH



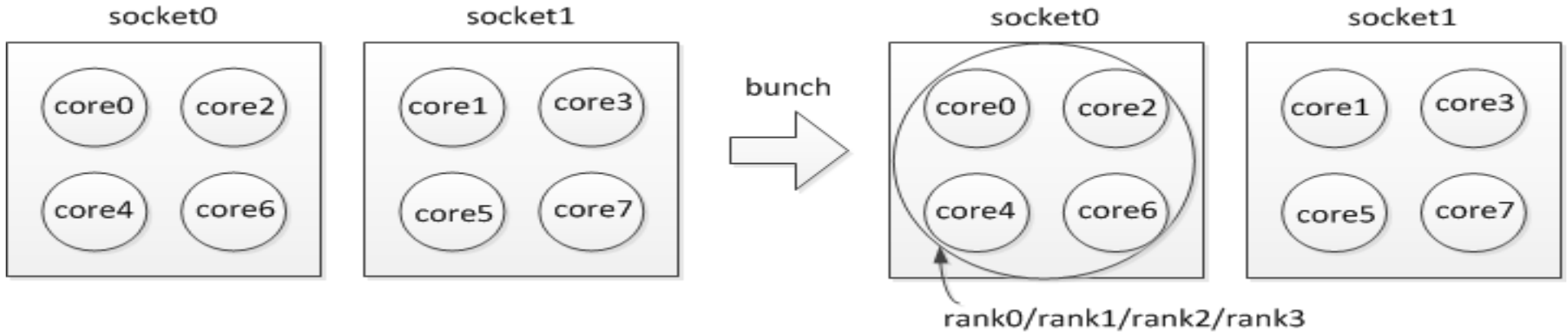
MVAPICH detects processor architecture at job-launch

Preset Process-binding Policies – Bunch

- "Core" level "Bunch" mapping
  - MVP_CPU_BINDING_POLICY=bunch

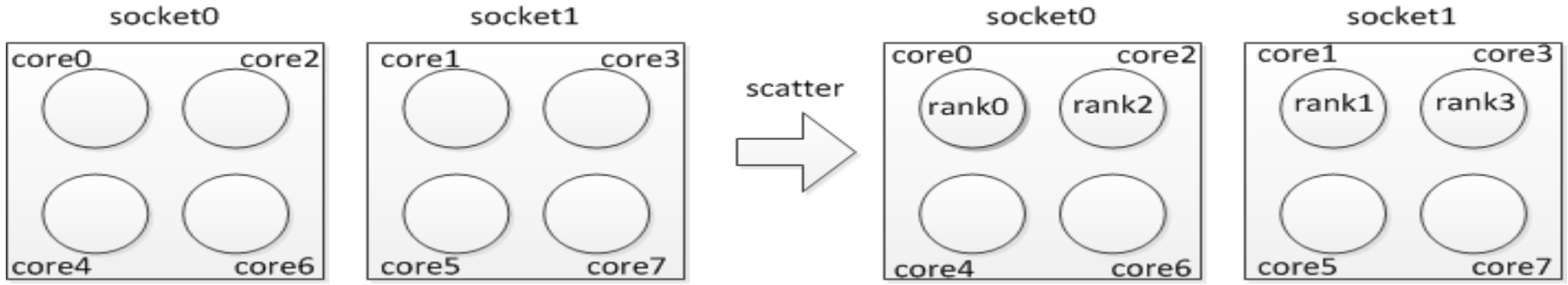


- "Socket/Numanode" level "Bunch" mapping
  - MVP_CPU_BINDING_LEVEL=socket MVP_CPU_BINDING_POLICY=bunch

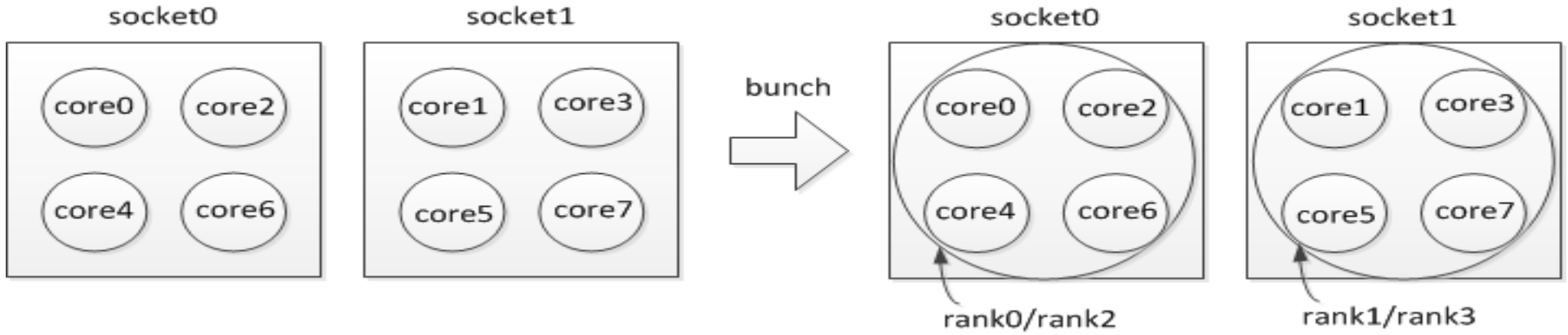


Preset Process-binding Policies – Scatter

- "Core" level "Scatter" mapping
  - MVP_CPU_BINDING_POLICY=scatter



- "Socket/Numanode" level "Scatter" mapping
  - MVP_CPU_BINDING_LEVEL=socket MVP_CPU_BINDING_POLICY=scatter

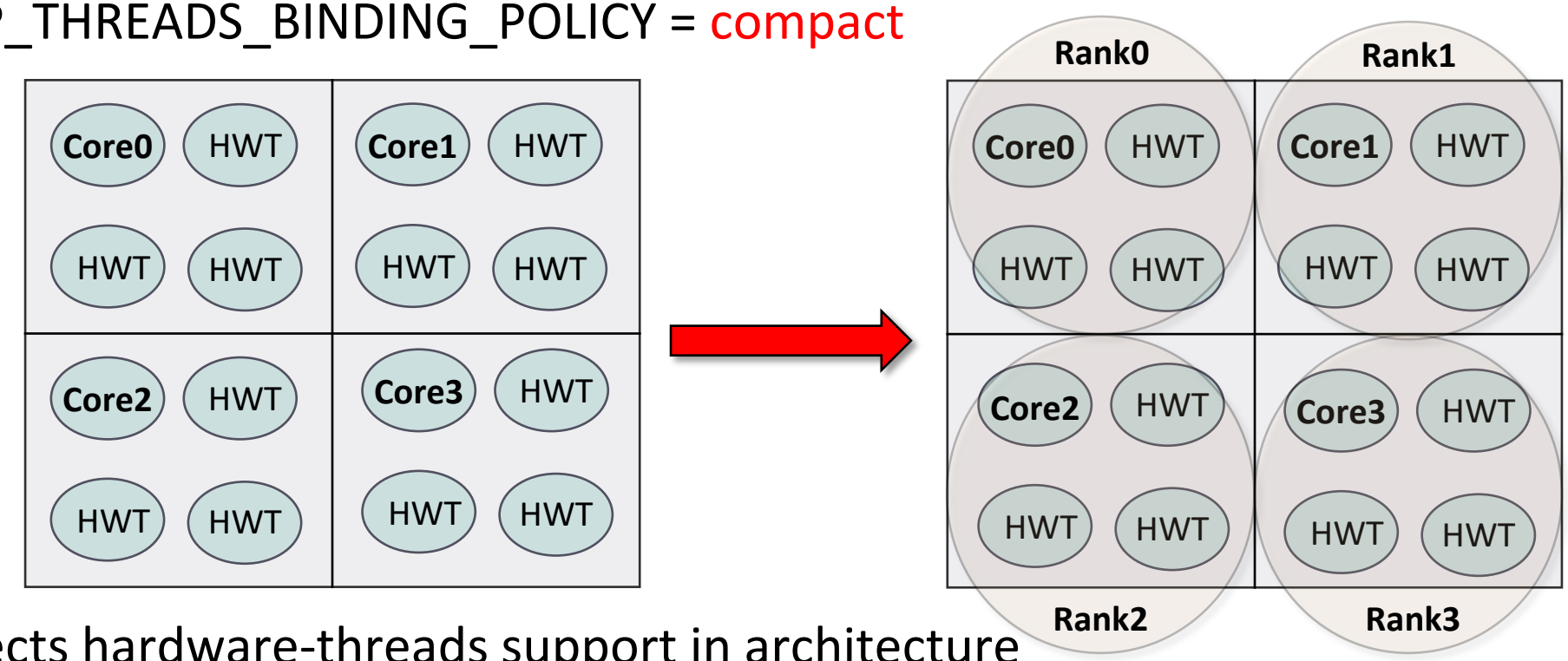


Process and thread binding policies in hybrid MPI+Threads

- A new process binding policy – "hybrid"
  - MVP_CPU_BINDING_POLICY = hybrid
- A new environment variable for co-locating Threads with MPI Processes
  - MVP_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
  - MVP_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)

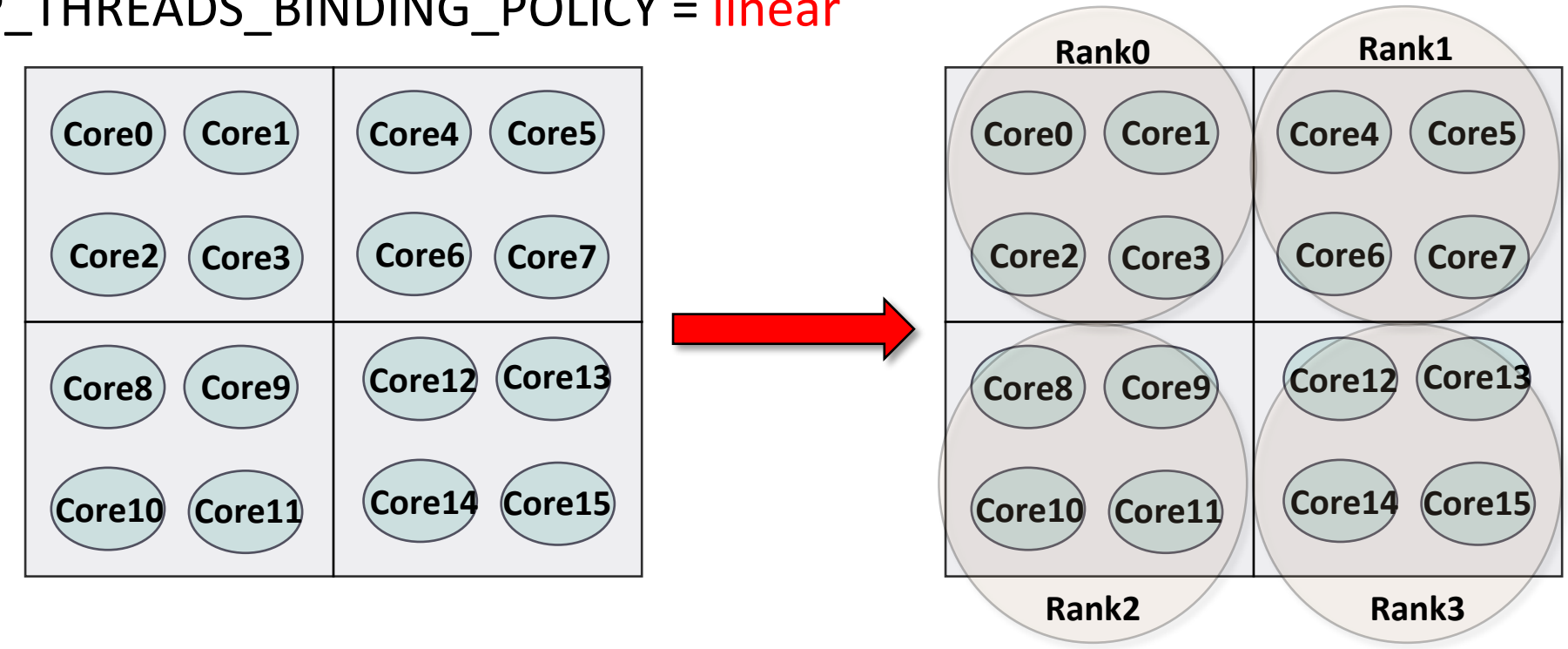
- MPI Processes = 4, OpenMP Threads per Process = 4
- MVP_CPU_BINDING_POLICY = hybrid
- MVP_THREADS_PER_PROCESS = 4
- MVP_THREADS_BINDING_POLICY = compact



- 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

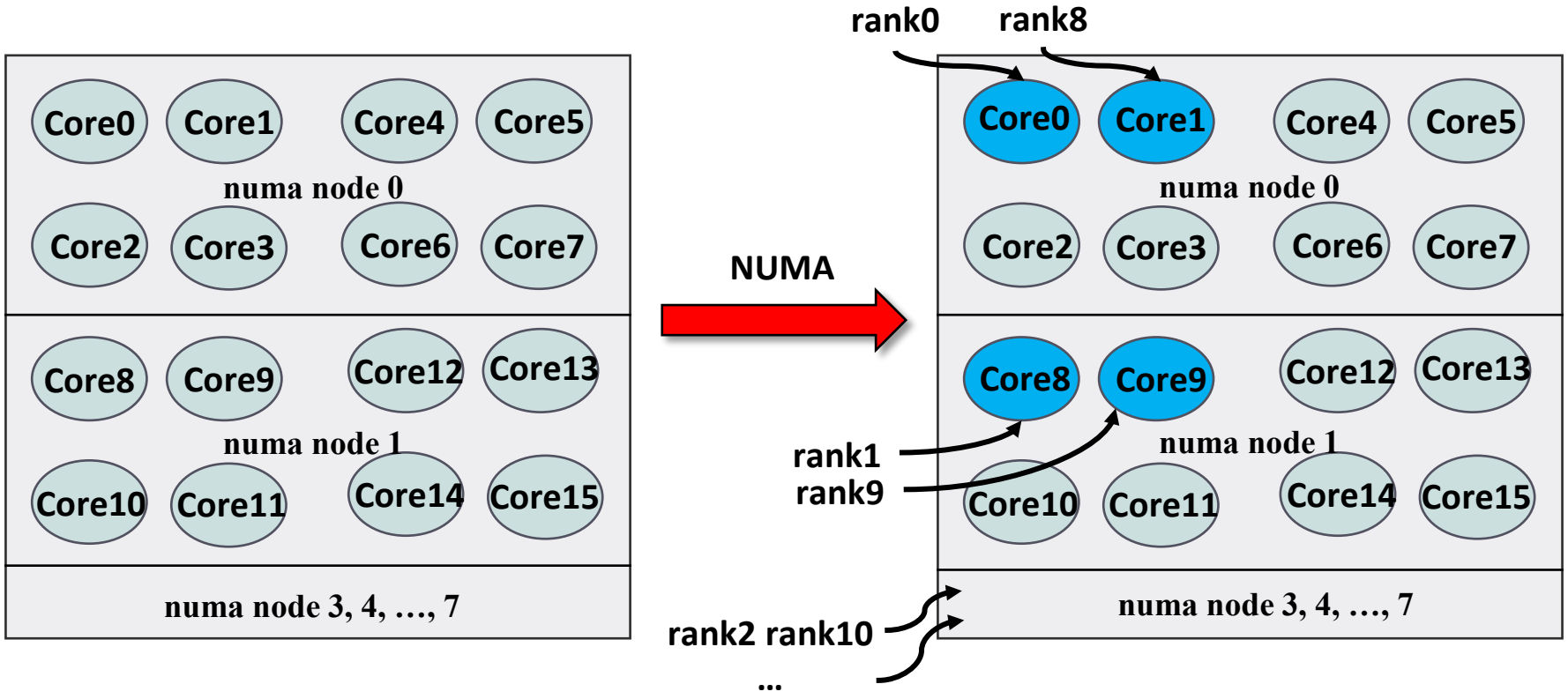
- MPI Processes = 4, OpenMP Threads per Process = 4
- MVP_CPU_BINDING_POLICY = hybrid
- MVP_THREADS_PER_PROCESS = 4
- MVP_THREADS_BINDING_POLICY = linear



- 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
- MVP_CPU_BINDING_POLICY = hybrid
- MVP_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 MVP_CPU_MAPPING=0:1:4:5 ./a.out
- Use ',' or '-' to bind to a set of cores:
  - $mpirun_rsh -np 64 -hostfile hosts MVP_CPU_MAPPING=0,2-4:1:5:6 ./a.out
- Is process binding working as expected?
  - **MVP_SHOW_CPU_BINDING=1**
    - Display CPU binding information
    - Launcher independent
    - Example
      - MVP_SHOW_CPU_BINDING=1 MVP_CPU_BINDING_POLICY=scatter

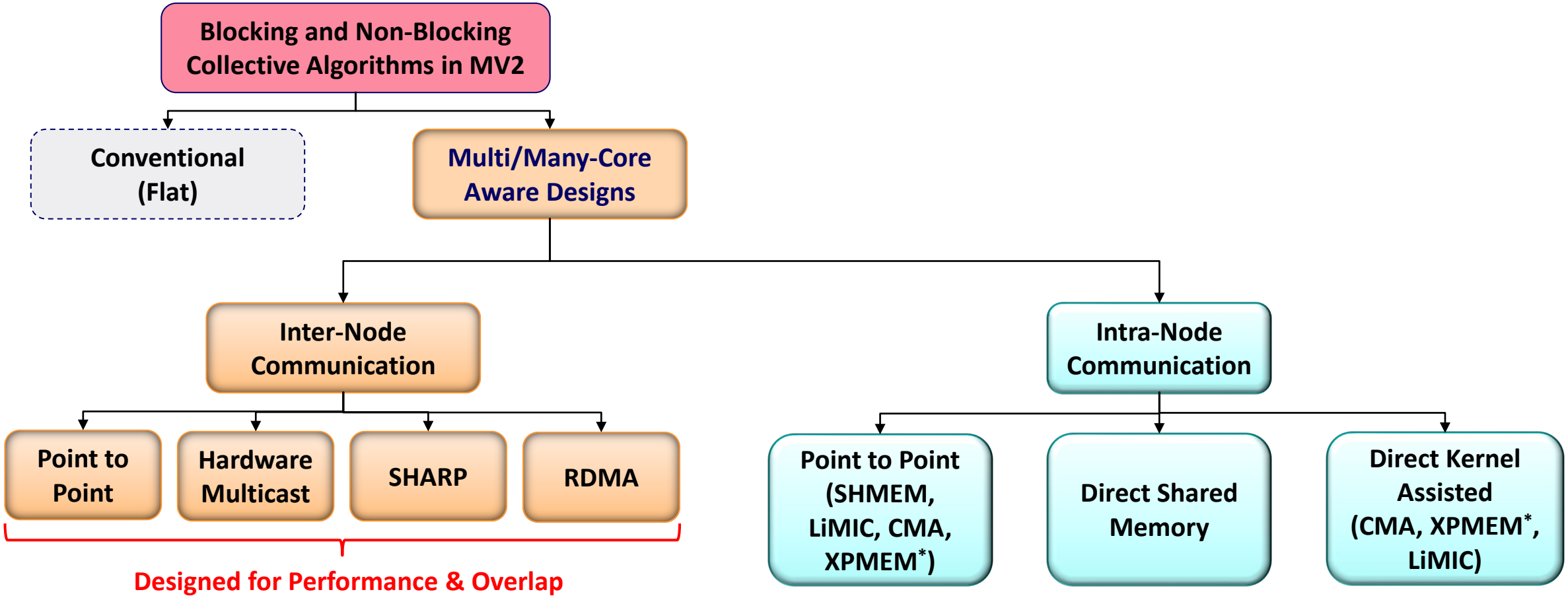
        ----------CPU AFFINITY----------

        RANK:0 CPU_SET: 0

        RANK:1 CPU_SET: 8

- **Refer to Running with Efficient CPU (Core) Mapping section of MVAPICH user guide for more information**
- **https://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-userguide.pdf#page=52**

Collective Communication in MVAPICH



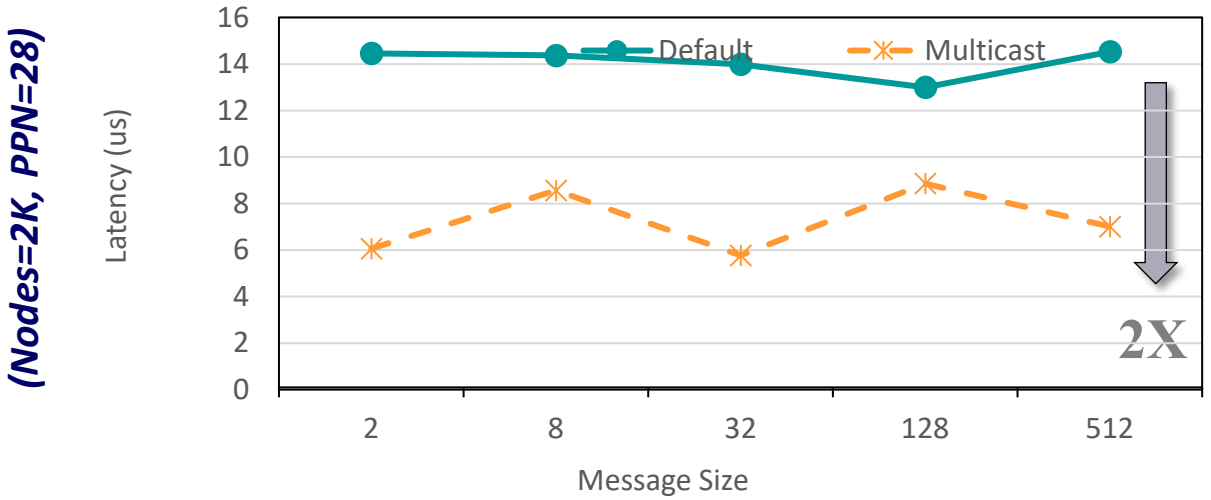
Run-time flags:

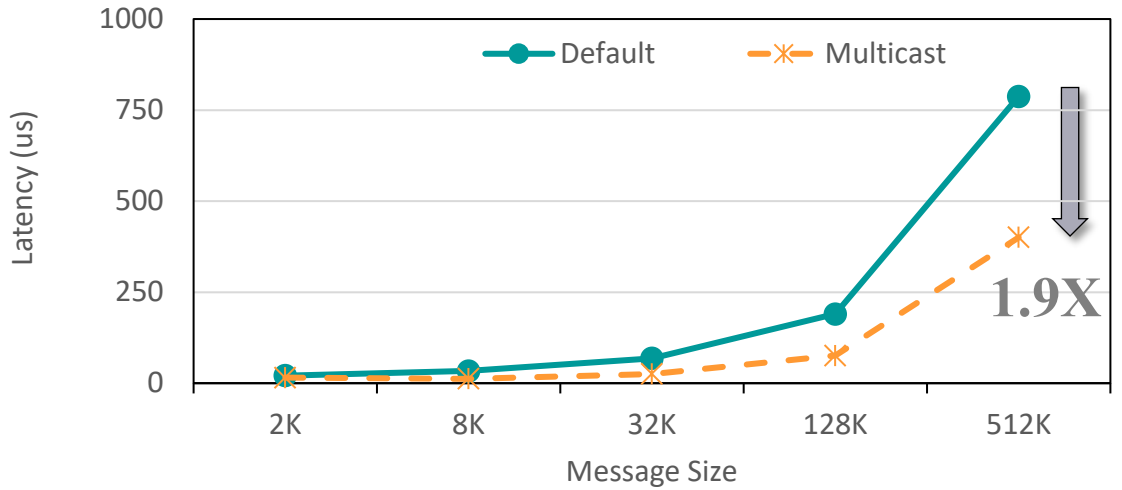
All shared-memory based collectives : MVP_USE_SHMEM_COLL (Default: ON)

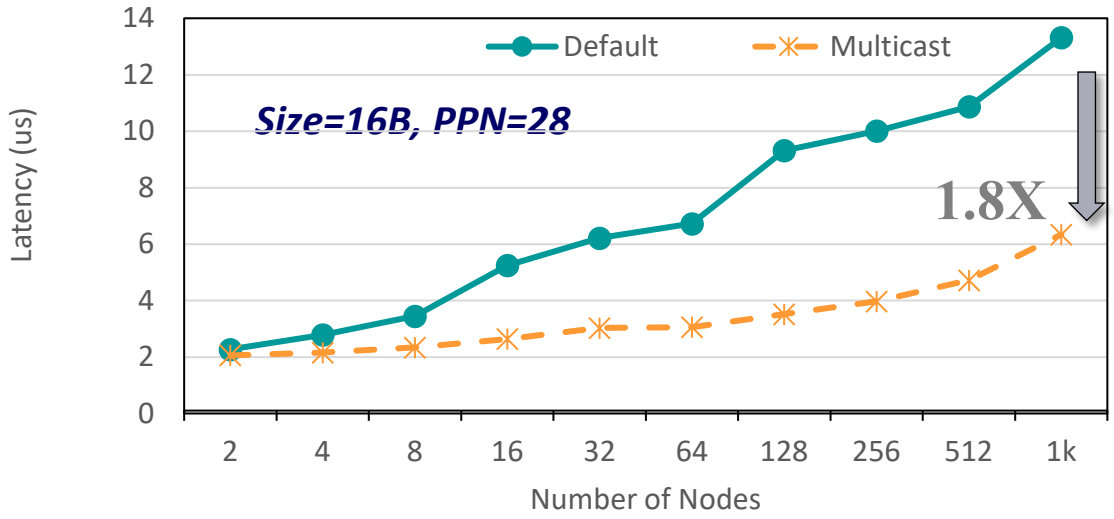
Hardware Mcast-based collectives : MVP_USE_MCAST (Default : OFF)

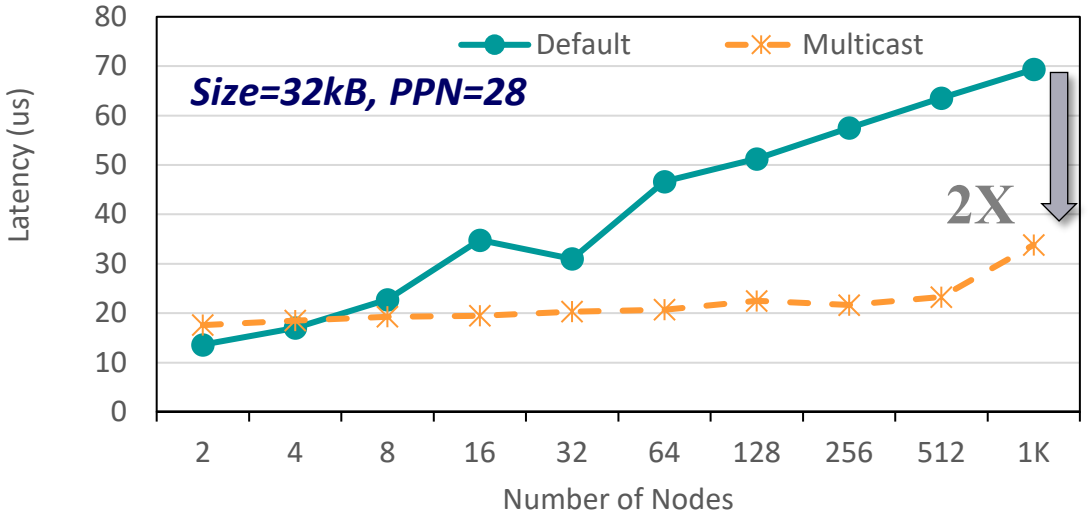
XPMEM-based collectives are available in MVAPICH2-X

Hardware Multicast-aware MPI_Bcast on TACC Frontera



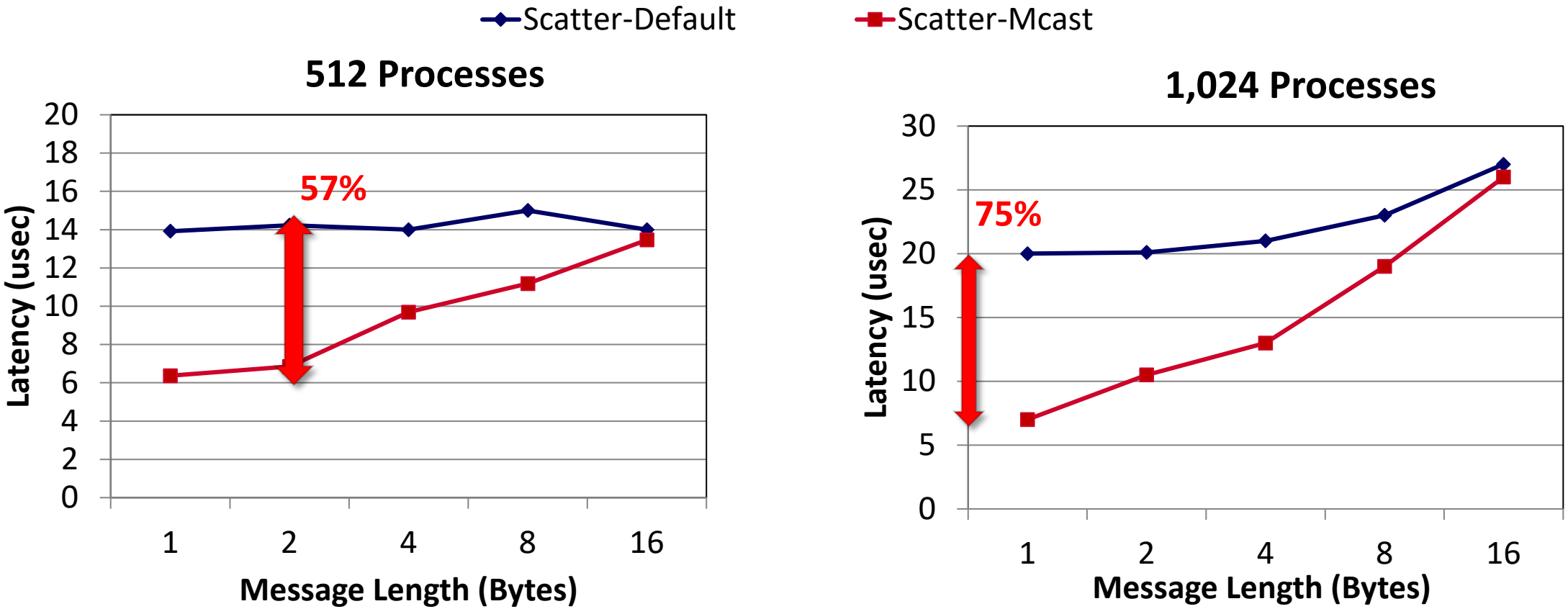






- MCAST-based designs improve latency of MPI_Bcast by up to **2X at 2,048 nodes**
- Use MVP_USE_MCAST=1 to enable MCAST-based designs

MPI_Scatter - Benefits of using Hardware-Mcast



- Enabling MCAST-based designs for MPI_Scatter improves small message up to **75%**

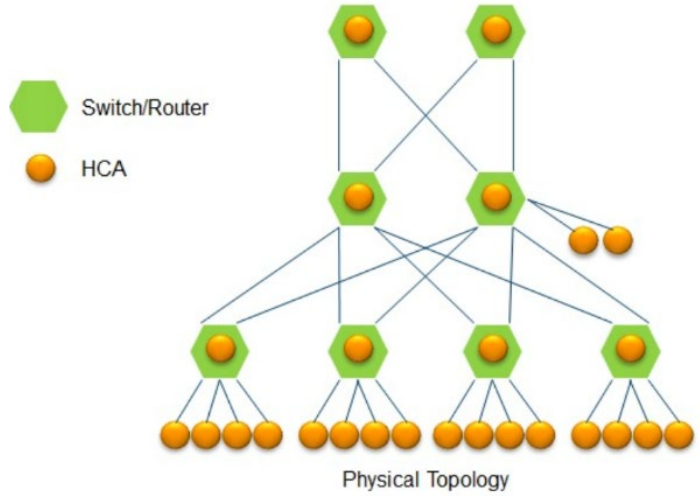
Parameter	Description	Default
MVP_USE_MCAST = 1	Enables hardware Multicast features	Disabled
--enable-mcast	Configure flag to enable	Enabled

- **Refer to Running Collectives with Hardware based Multicast support section of MVAPICH user guide for more information**
- **https://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-userguide.pdf#page=58**

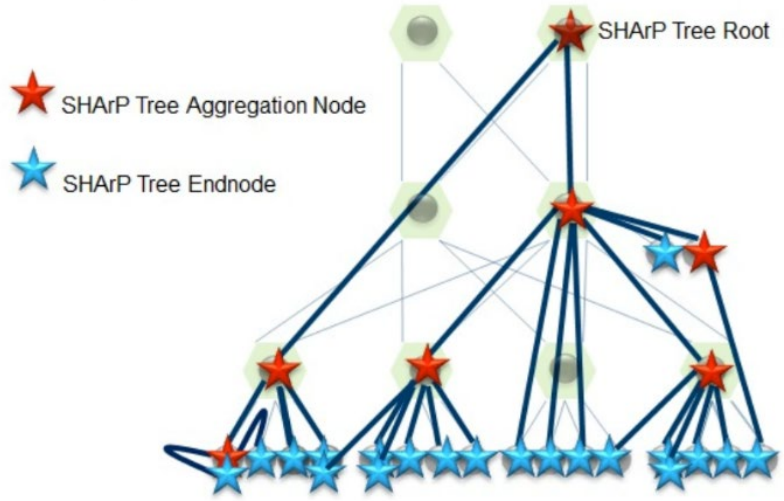
Offloading with Scalable Hierarchical Aggregation Protocol (SHArP)

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

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



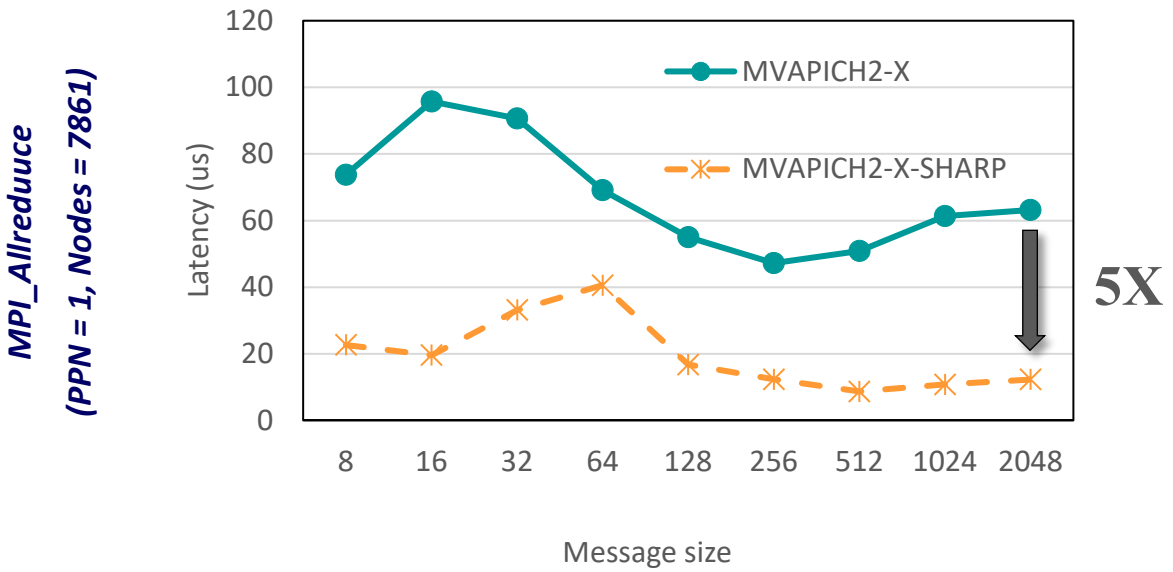
Physical Network Topology*

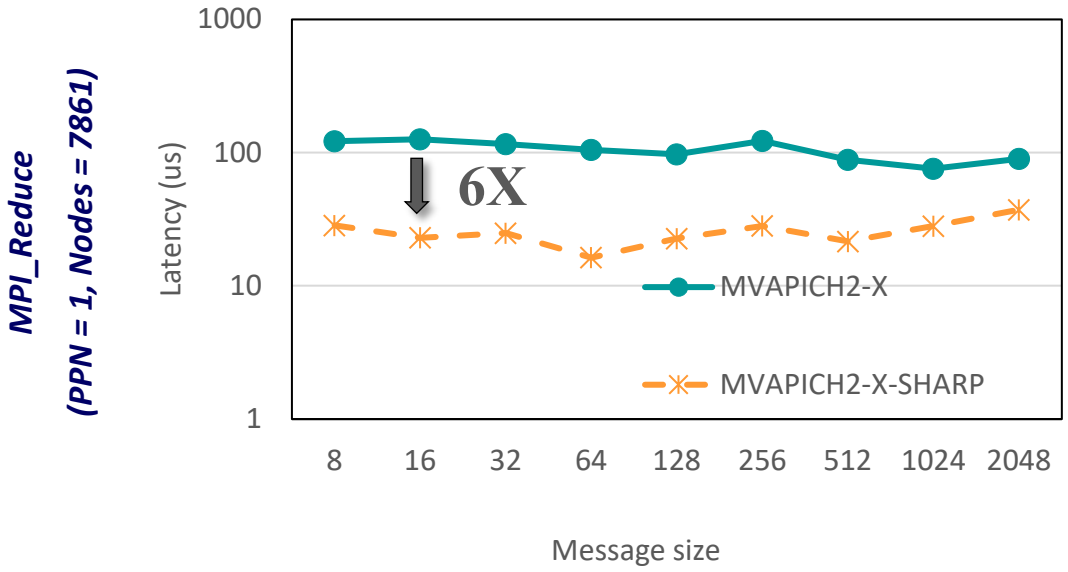


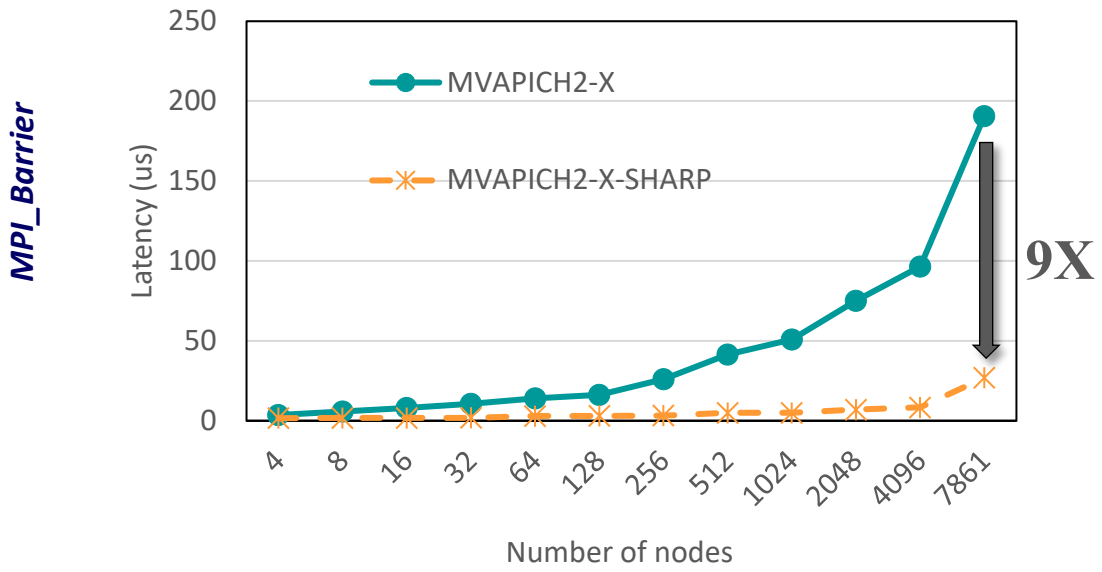
Logical SHArP Tree*

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

Performance of Blocking Collectives with In-Network Computing







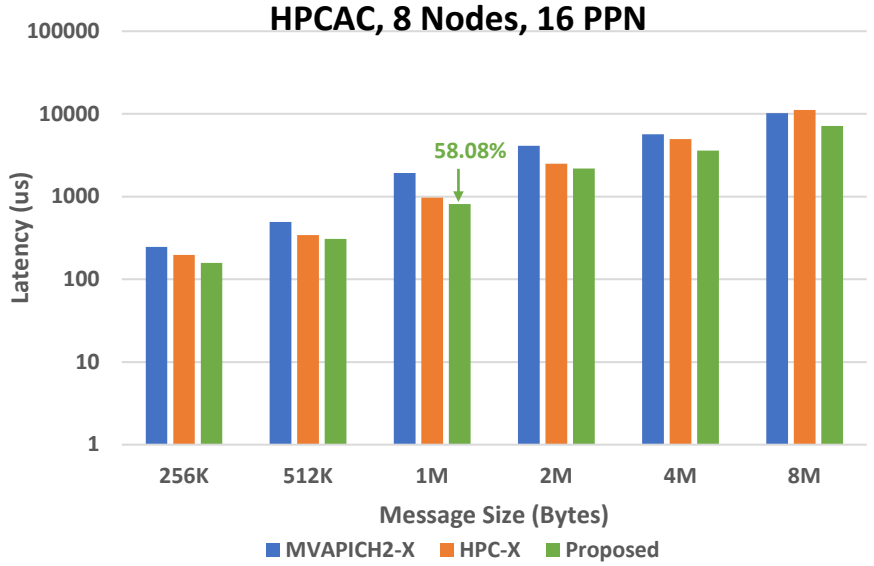
Optimized SHARP designs in MVAPICH2-X

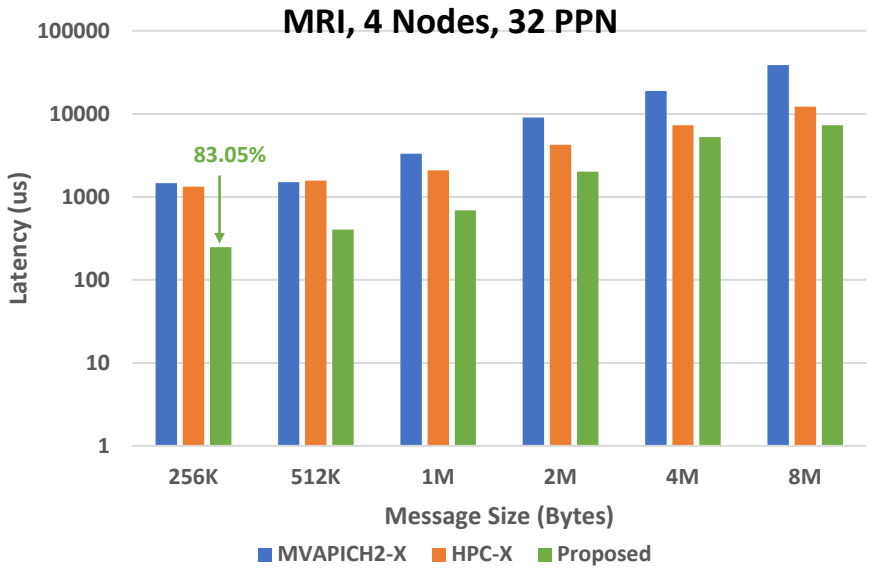
Up to 9X performance improvement with SHARP over MVAPICH2-X default for 1ppn MPI_Barrier, ***6X*** for 1ppn MPI_Reduce and ***5X*** for 1ppn MPI_Allreduce

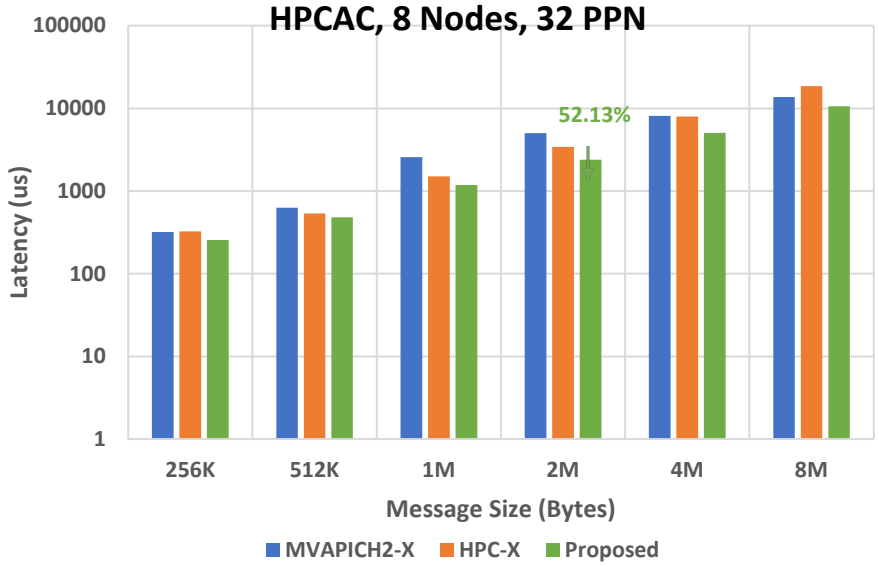
B. Ramesh , K. Suresh , N. Sarkauskas , M. Bayatpour , J. Hashmi , H. Subramoni , and D. K. Panda, Scalable MPI Collectives using SHARP: Large Scale Performance Evaluation on the TACC Frontera System, ExaMPI2020 - Workshop on Exascale MPI 2020, Nov 2020.

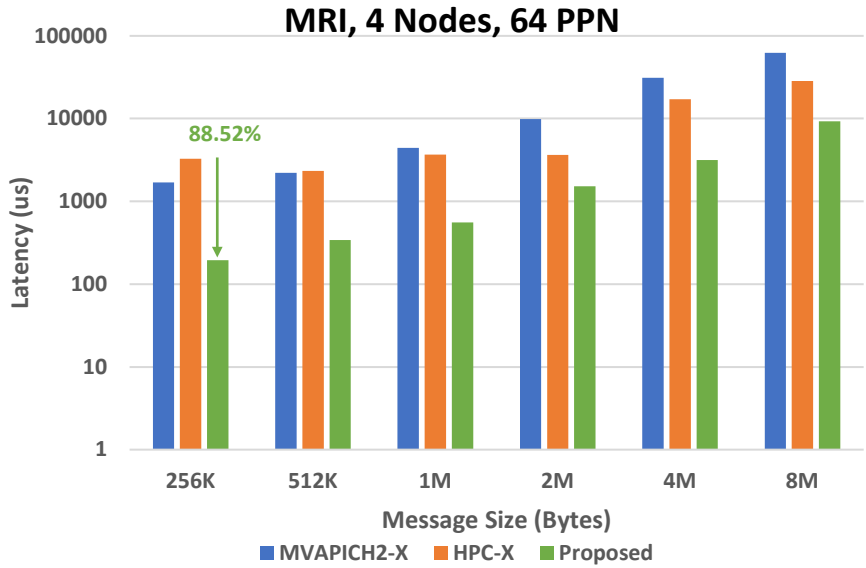
Optimized Runtime Parameters: MVP_ENABLE_SHARP = 1

Performance of Reduction Collectives with Streaming Aggregation



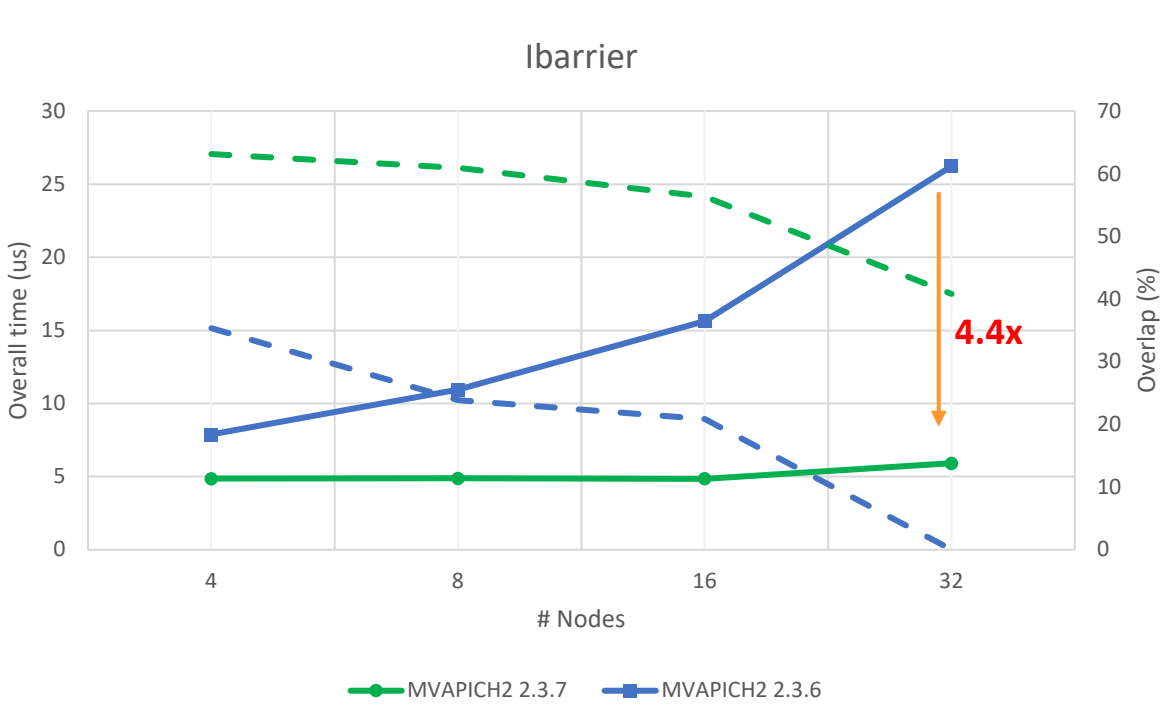


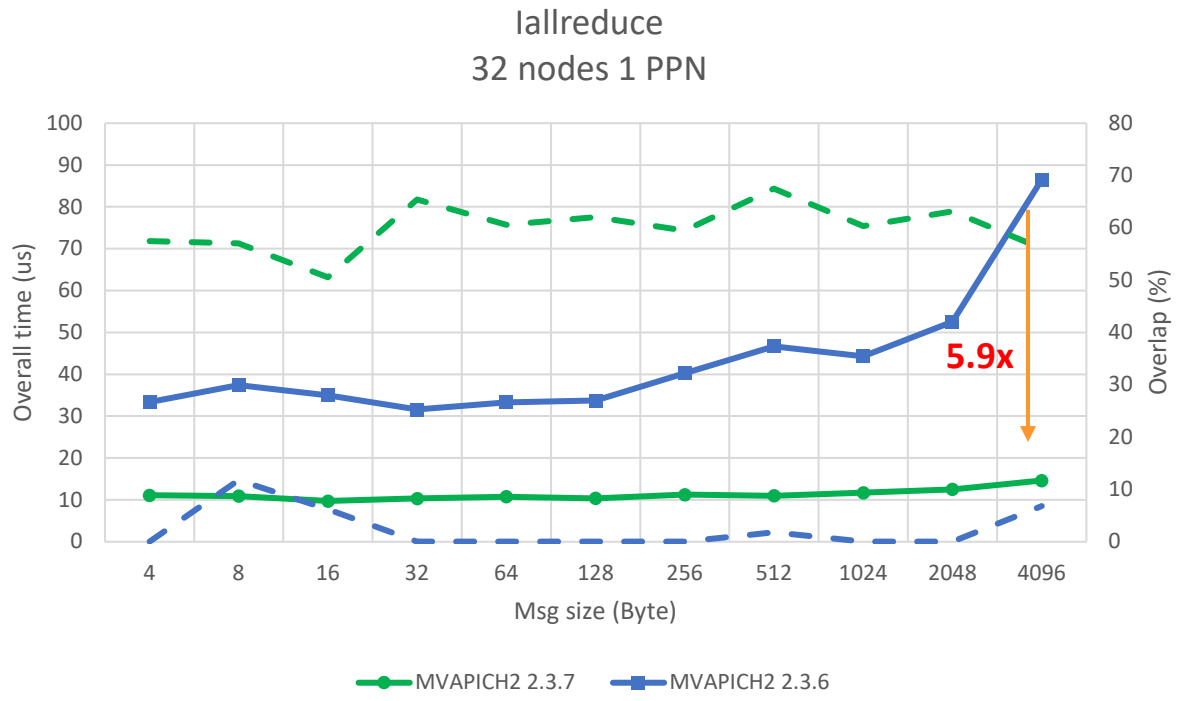




B. Ramesh, G. Kuncham, K. Suresh, R. Vaidya, N. Alnaasan, M. Abduljabbar, A. Shafi, D. Panda, Designing In-network Computing Aware Reduction Collectives in MPI, Hot Interconnects 2023, Aug 2023.

Non-blocking Collectives Support with In-Network Computing



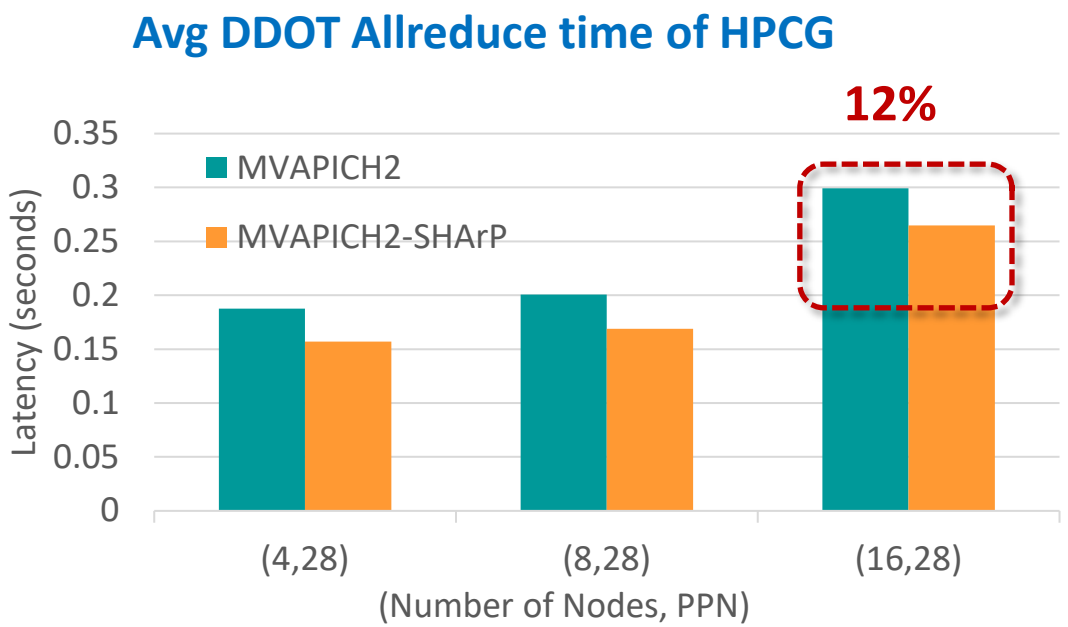


- With SHARP:
  - Flat scaling in terms of overall time
  - High overlap between computation and communication

Available with MVAPICH 2.3.7

Platform: Dual-socket Intel(R) Xeon(R) Platinum 8280 CPU @ 2.70GHz nodes equipped with Mellanox InfiniBand, HDR-100 Interconnect

Benefits of SHARP Allreduce at Application Level



SHARP support available since MVAPICH 2.3a

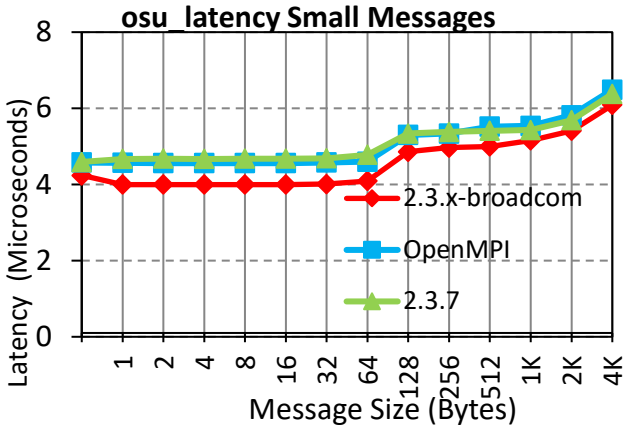
More details in the talk "Benefits of Streaming Aggregation with SHARPv2 in MVAPICH", Bharath Ramesh, The Ohio State University MUG 2020

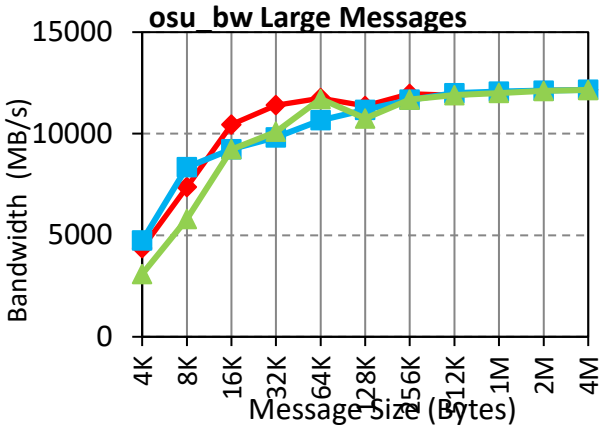
Parameter	Description	Default
MVP_ENABLE_SHARP=1	Enables SHARP-based collectives	Disabled
--enable-sharp	Configure flag to enable SHARP	Disabled

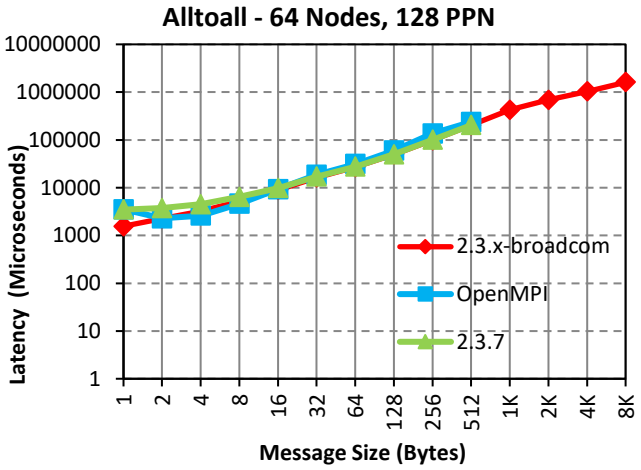
- Refer to Running Collectives with Hardware based SHARP support section of MVAPICH user guide for more information
- http://mvapich.cse.ohio-state.edu/static/media/mvapich/MVAPICH-userguide.html#x1-1050006.27

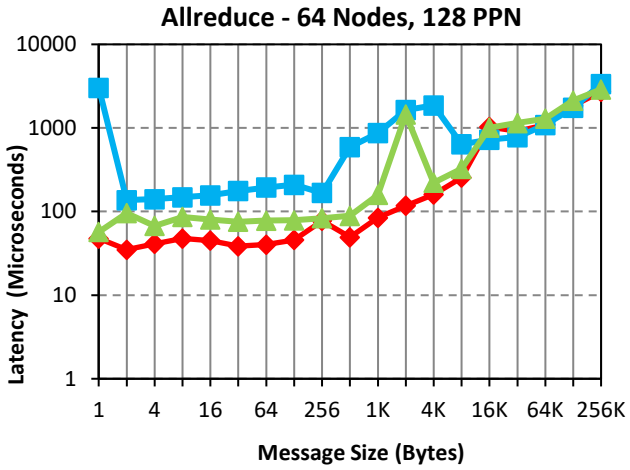
Performance Evaluation – Micro-benchmarks on Broadcom RoCE

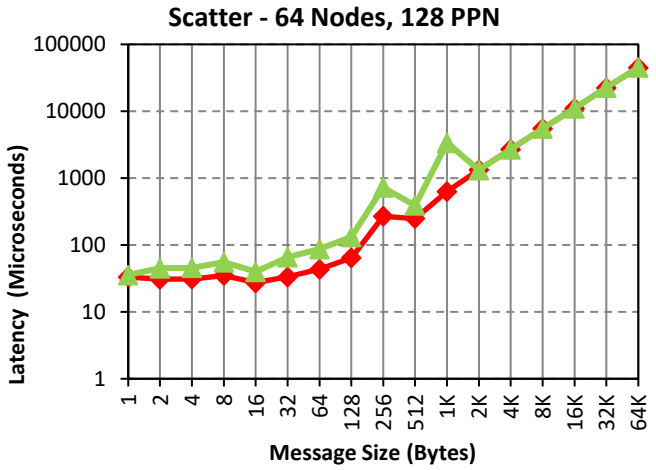
- Experimental results from Dell Bluebonnet
- Up to 20% reduction in small message point-to-point latency
- From 0.1x to 2x increase in bandwidth
- Up to 12.4x lower MPI_Allreduce latency
- Up to 5x lower MPI_Scatter latency



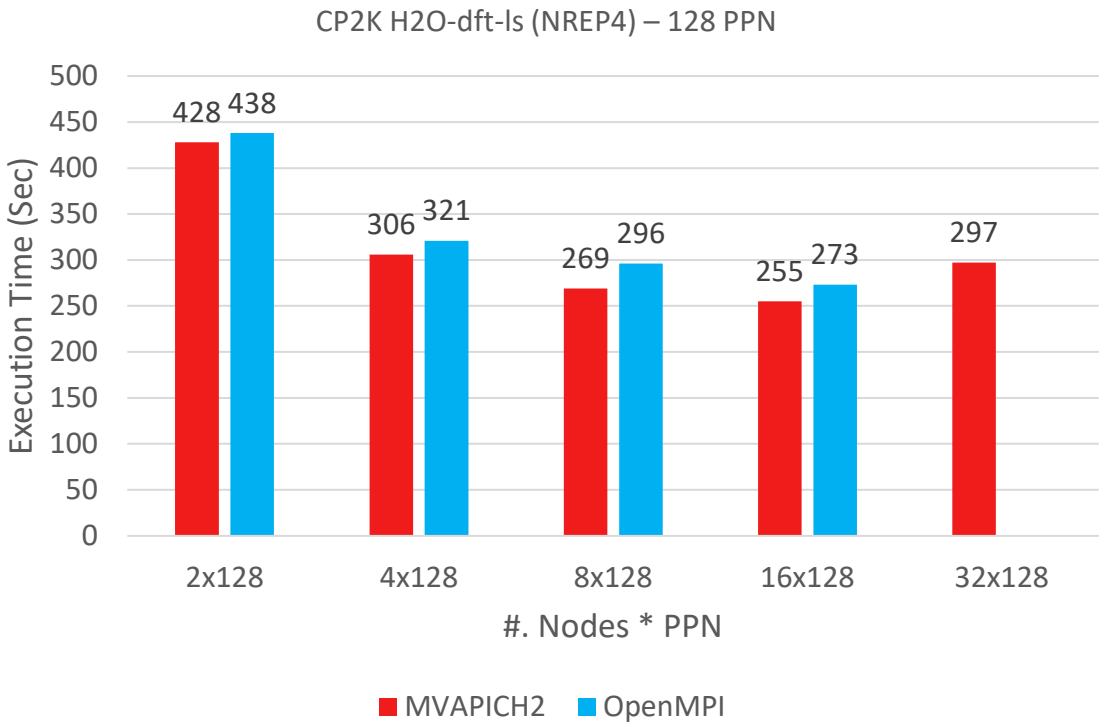


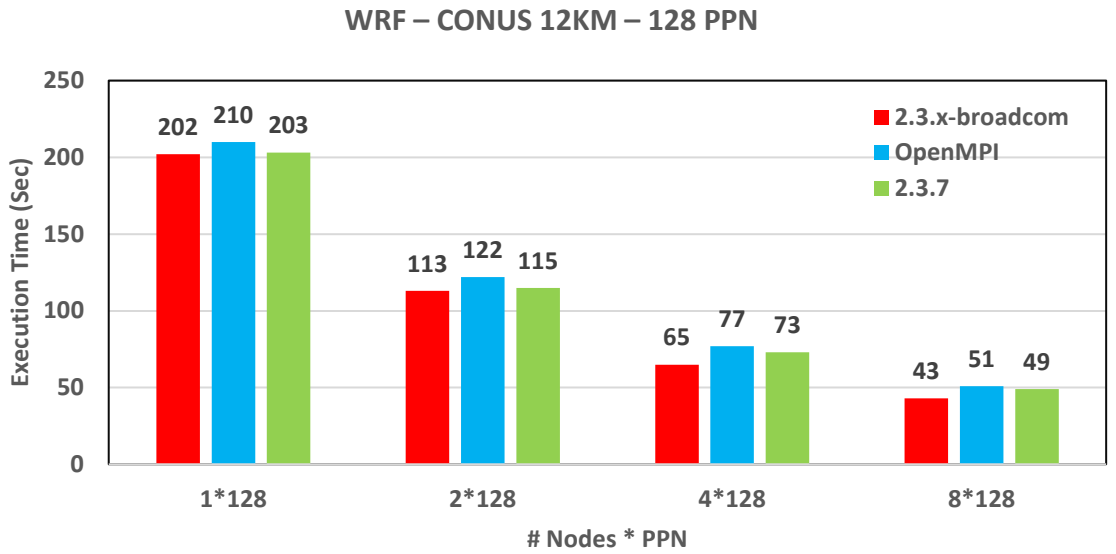


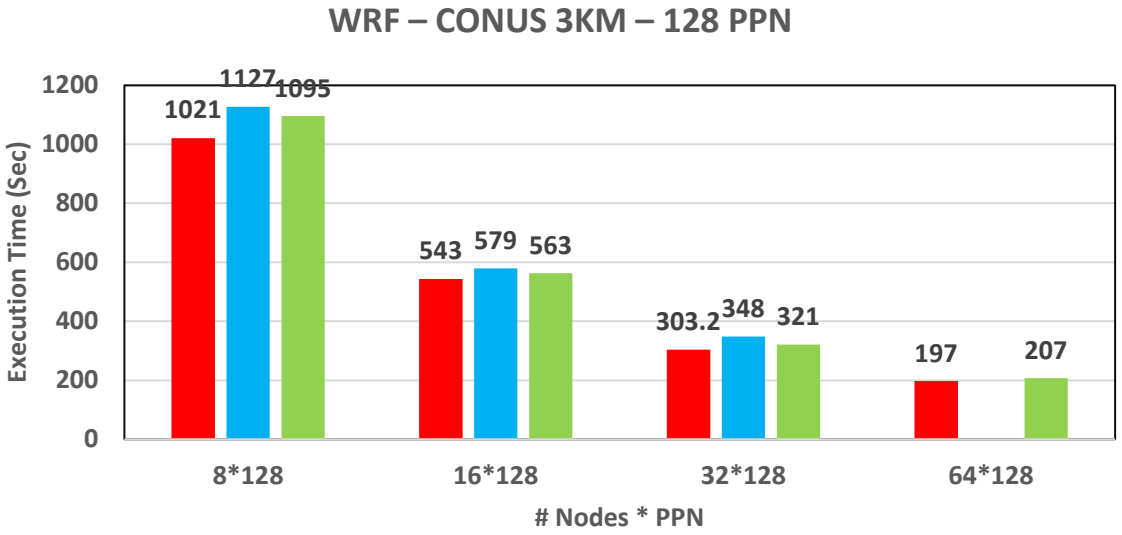




Performance Evaluation – Applications on Broadcom RoCE







- Reduce up to 45% execution time of CP2K H2O-dft-ls (NREP4)
- Reduce up to 7% execution time of WRF CONUS 3KM

MVAPICH 3.0 - OFI and UCX Support

- Support a broad range of interconnects with widely used libraries
  - Configure with `--with-device=ch4:ofi` or `--with-device=ch4:ucx`
- Runtime provider selection via CVARs
  - `MPIR_CVAR_OFI_USE_PROVIDER=<prov>`
- System default, embedded, or custom installation of OFI/UCX
  - Configure with `--with-libfabric=embedded` or `--with-libfabric=<path>`
  - Configure with `--with-ucx=embedded` or `--with-ucx=<path>`
- Enhanced MVAPICH collective designs available on all supported networks

Upcoming/Planned Features

- Enhanced MVAPICH OFI provider with unified design
- Enhanced intra-node designs
- Enhanced pt2pt support for IB/RoCE systems
- Enhanced launcher

MVAPICH Software Family

Requirements	Library
MPI with Support for InfiniBand, Omni-Path, Ethernet/iWARP and, RoCE (v1/v2)	MVAPICH
Advanced MPI features/support (UMR, ODP, DC, Core-Direct, SHArP, XPMEM), OSU INAM (InfiniBand Network Monitoring and Analysis), PGAS (OpenSHMEM, UPC, UPC++, and CAF), MPI+PGAS (OpenSHMEM, UPC, UPC++, and CAF) with IB and RoCE (v1/v2)	MVAPICH2-X
Optimized MPI for clusters with NVIDIA GPUs and for GPU-enabled Deep Learning Applications	MVAPICH2-GDR
Advanced MPI with unified MVAPICH2-GDR and MVAPICH2-X features for HPC, DL, ML, Big Data and Data Science applications	MVAPICH-PLUS

GPU-Aware (CUDA-Aware) MPI Library: MVAPICH-GPU

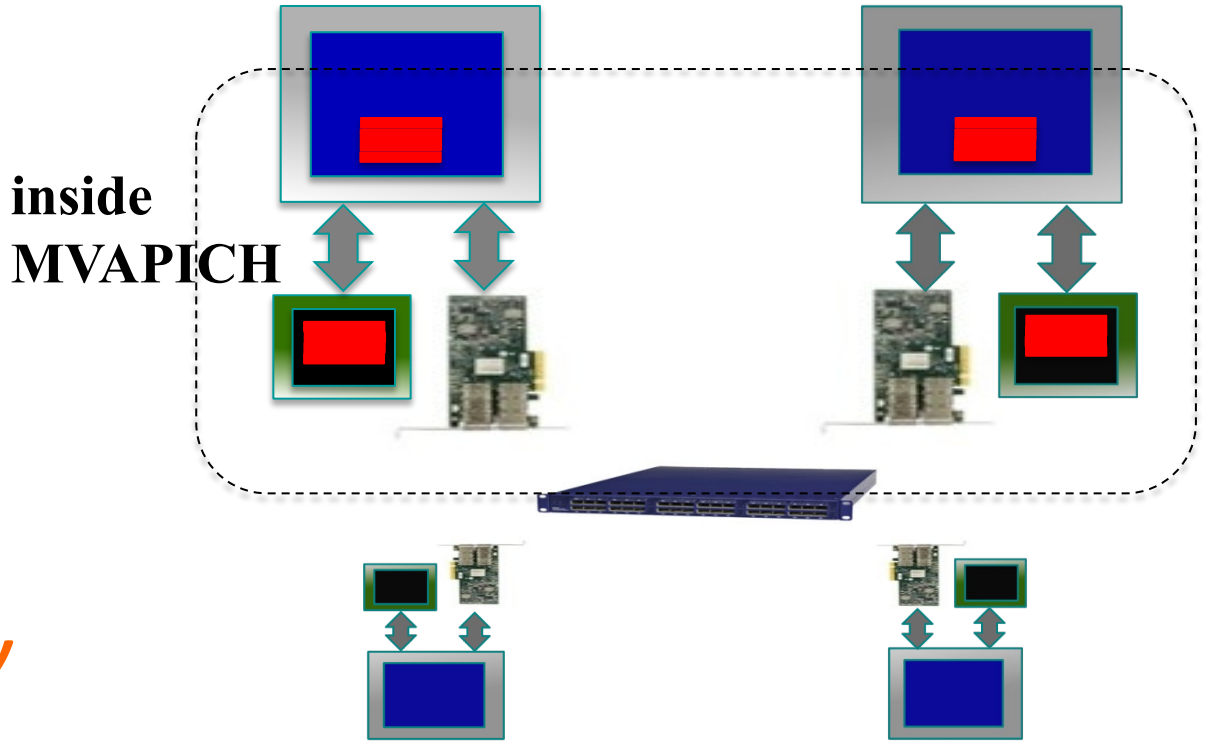
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

At Receiver:

```
MPI_Recv(r_devbuf, size, …);
```



High Performance and High Productivity

GPU-Aware MPI: MVAPICH2-GDR 1.8-2.3.7 Releases

- GPU-aware MPI:
  - CUDA-aware MPI: Support for MPI communication from NVIDIA GPU device memory
  - ROCm-aware MPI: Support for MPI communication between AMD GPUs (from MVAPICH2-GDR 2.3.5+)
- 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 2.3.7 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.gnu4.8.5-2.3.7-1.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

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 2.3.7

- Released on 05/27/2022
- Major Features and Enhancements
  - Based on MVAPICH 2.3.7
  - Enhanced performance for GPU-aware MPI_Alltoall and MPI_Alltoallv
  - Added automatic rebinding of processes to cores based on GPU NUMA domain
    - This is enabled by setting the env MVP_GPU_AUTO_REBIND=1
  - Added NCCL communication substrate for various non-blocking MPI collectives
    - MPI_Iallreduce, MPI_Ireduce, MPI_Iallgather, MPI_Iallgatherv, MPI_Ialltoall, MPI_Ialltoallv, MPI_Iscatter, MPI_Iscatterv, MPI_Igather, MPI_Igatherv, and MPI_Ibcast
  - Enhanced point-to-point and collective tuning for AMD Milan processors with NVIDIA A100 and AMD Mi100 GPUs
  - Enhanced point-to-point and collective tuning for NVIDIA DGX-A100 systems
  - Added support for Cray Slingshot-10 interconnect
  - Added support for 'on-the-fly' compression of point-to-point messages used for GPU-to-GPU communication
    - Applicable to NVIDIA GPUs
  - NCCL communication substrate for various MPI collectives
    - Support for hybrid communication protocols using NCCL-based, CUDA-based, and IB verbs-based primitives
    - MPI_Allreduce, MPI_Reduce, MPI_Allgather, MPI_Allgatherv, MPI_Alltoall, MPI_Alltoallv, MPI_Scatter, MPI_Scatterv, MPI_Gather, MPI_Gatherv, and MPI_Bcast
  - Full support for NVIDIA DGX, NVIDIA DGX-2 V-100, and NVIDIA DGX-2 A-100 systems
  - Enhanced architecture detection, process placement and HCA selection
  - Enhanced intra-node and inter-node point-to-point tuning
  - Enhanced collective tuning
  - Introduced architecture detection, point-to-point tuning and collective tuning for ThetaGPU @ANL
  - Enhanced point-to-point and collective tuning for NVIDIA GPUs on Frontera @TACC, Lassen @LLNL, and Sierra @LLNL
  - Enhanced point-to-point and collective tuning for Mi50 and Mi60 AMD GPUs on Corona @LLNL
  - Added several new MPI_T PVARs
  - Added support for CUDA 11.3
  - Added support for ROCm 4.1
  - Enhanced output for runtime variable MVP_SHOW_ENV_INFO
  - Tested with Horovod and common DL Frameworks
    - TensorFlow, PyTorch, and MXNet
  - Tested with MPI4Dask 0.2
    - MPI4Dask is a custom Dask Distributed package with MPI support
  - Tested with MPI4cuML 0.1
    - MPI4cuML is a custom cuML package with MPI support

Tuning GDRCOPY Designs in MVAPICH2-GDR

Parameter	Significance	Default	Notes
MVP_ENABLE_GPU_GDRCOPY	• Enable / Disable GDRCOPY-based designs	1 (Enabled)	• Always enable
MVP_GPU_GDRCOPY_RANGE	• Controls messages size until which GDRCOPY is used	0 - 8 KByte	• Tune for your system • GPU type, host architecture. Impacts the eager performance
MVP_GDRCOPY_LIB	• Path to the GDRCOPY library	Unset	• Set if gdrcopy is not available on standard system paths
MVP_GPU_GDRCOPY_D2H_LIMIT	• Controls messages size until which GDRCOPY is used at sender	16 Bytes	• Tune for your systems • 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**

Tuning Loopback Designs in MVAPICH2-GDR

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

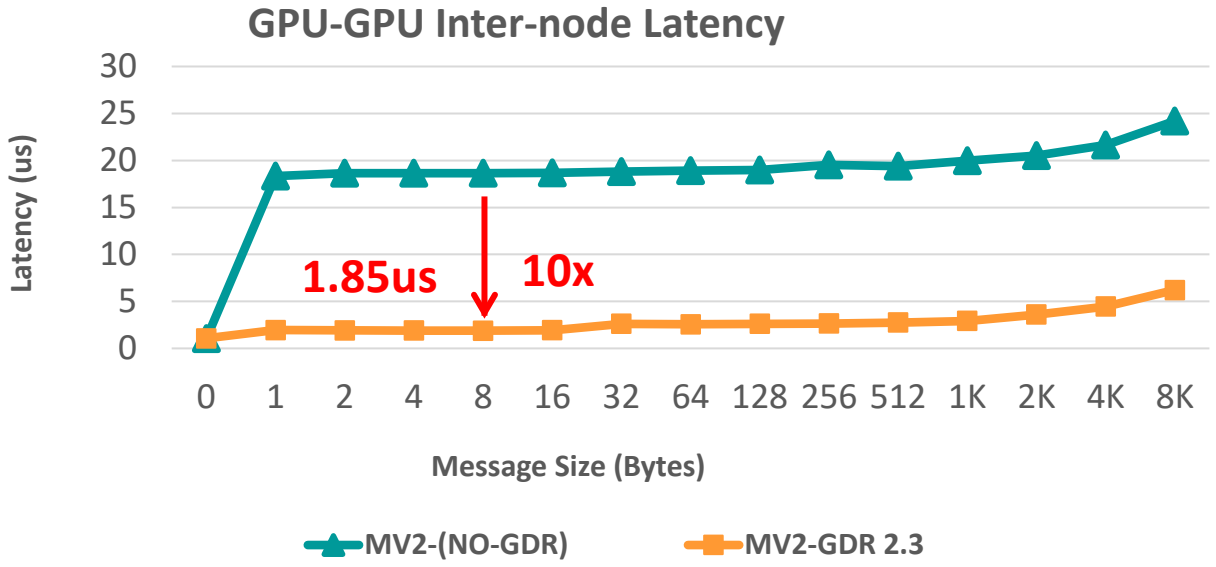
- **Refer to Tuning and Usage Parameters section of MVAPICH2-GDR user guide for more information**
- **http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters**

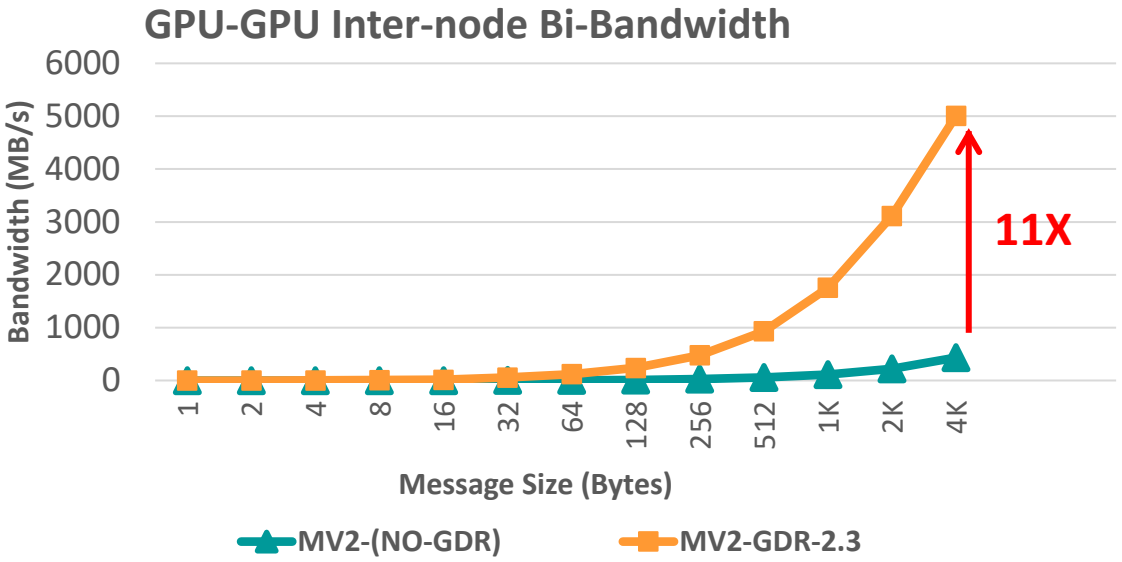
Tuning GPUDirect RDMA (GDR) Designs in MVAPICH2-GDR

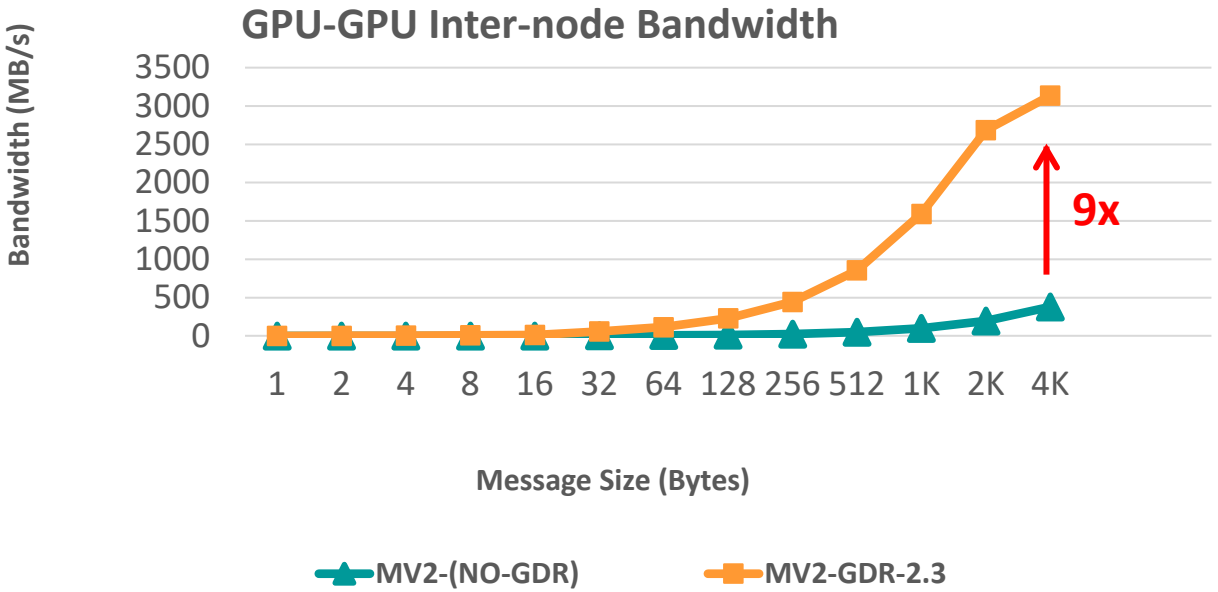
Parameter	Significance	Default	Notes
MVP_USE_GPUDIRECT	• Enable / Disable GDR-based designs	1 (Enabled)	• Always enable
MVP_GPUDIRECT_LIMIT	• Controls messages size until which GPUDirect RDMA is used	8 KByte	• Tune for your system • GPU type, host architecture and CUDA version: impact pipelining overheads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

MVAPICH2-GDR with CUDA-aware MPI Support



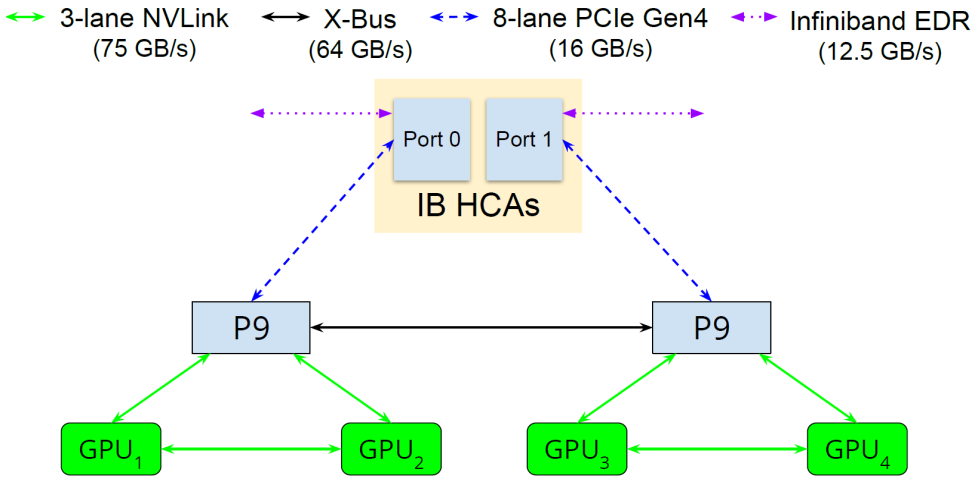




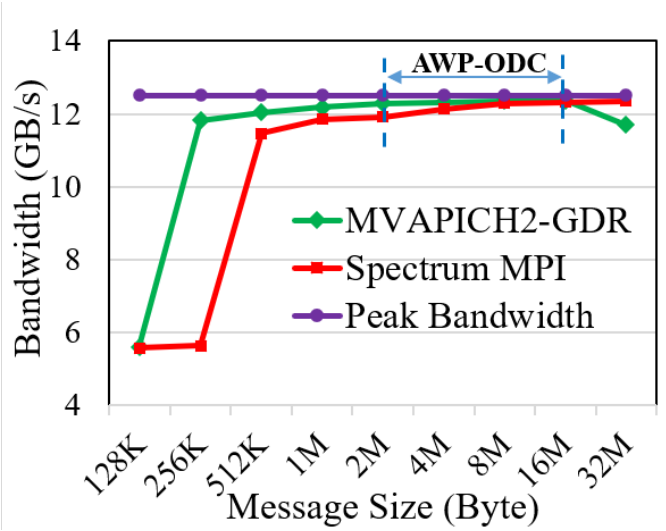
MVAPICH2-GDR-2.3.7
Intel Haswell (E5-2687W @ 3.10 GHz) node - 20 cores
NVIDIA Volta V100 GPU
Mellanox Connect-X4 EDR HCA
CUDA 9.0
Mellanox OFED 4.0 with GPU-Direct-RDMA

MVAPICH2-GDR: "On-the-fly" Compression – Motivation

- For HPC and data science applications on modern GPU clusters
  - With larger problem sizes, applications exchange **orders of magnitude more data** on the network
  - Leads to significant **increase in communication times** for these applications on larger scale (AWP-ODC)
  - On modern HPC systems, there is **disparity** between intra-node and inter-node GPU communication bandwidths that prevents efficient scaling of applications on larger GPU systems
  - CUDA-Aware MPI libraries **saturate the bandwidth** of IB network
  - **Compression** can reduce the data size and lower the pressure on network with limited bandwidth



(a) Disparity between intra-node and inter-node GPU communication on Sierra OpenPOWER supercomputer [1]



(b) Saturated bandwidth at large message size

[1] K. S. Khorassani, C.-H. Chu, H. Subramoni, and D. K. Panda, "Performance Evaluation of MPI Libraries on GPU-enabled OpenPOWER Architectures: Early Experiences", in International Workshop on Open-POWER for HPC (IWOPH 19) at the 2019 ISC High Performance Conference, 2018.

Install Supporting Libraries for “On-the-fly” Compression Support

- MPC
  - Installation (Built-in with MVAPICH2-GDR)
  - Runtime parameters

    MVP_ENABLE_GPU=1 MVP_ENABLE_COMPRESSION=1 MVP_COMPRESSION_ALGORITHM=MPC
- ZFP
  - Bundled with MVAPICH since 3.0
  - Runtime parameters

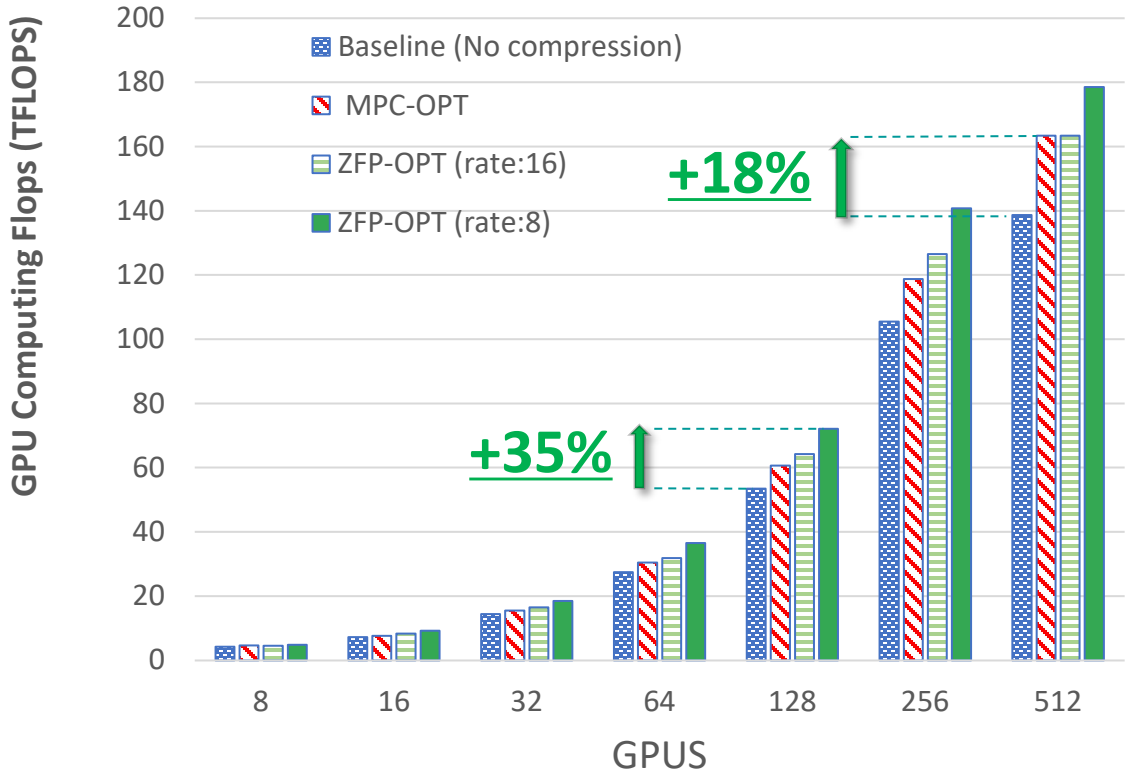
    MVP_ENABLE_GPU=1 MVP_ENABLE_COMPRESSION=1 MVP_COMPRESSION_ALGORITHM=ZFP

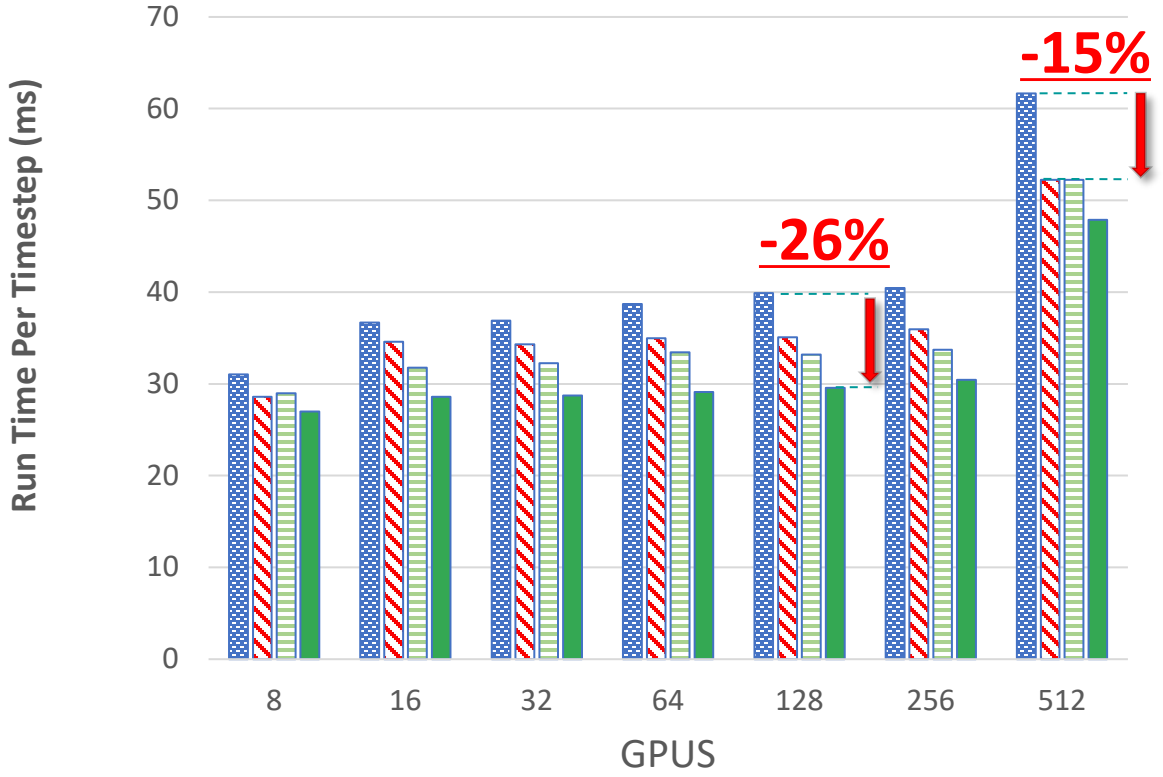
Tuning "On-the-fly" Compression Support in MVAPICH2-GDR

Parameter	Default value	Notes
MVP_ENABLE_COMPRESSION	0	Enable compression
MVP_COMPRESSION_ALGORITHM	MPC	MPC: Use MPC compression ZFP: Use ZFP compression
MVP_COMPRESSION_THRESHOLD	524288	Threshold of using compression for inter-node pt2pt GPU communication
MVP_COMPRESSION_THRESHOLD_INTRA	524288	Threshold of using compression for intra-node pt2pt GPU communication
MVP_COMPRESSION_DIMENSION	Depends on compression library	Dimensionality of input data [1, 31]: For MPC compression 1: For ZFP compression
MVP_COMPRESSION_ZFP_RATE	8	Compressed bits/value [1, 32]: For ZFP compression only

"On-the-fly" Compression Support in MVAPICH2-GDR

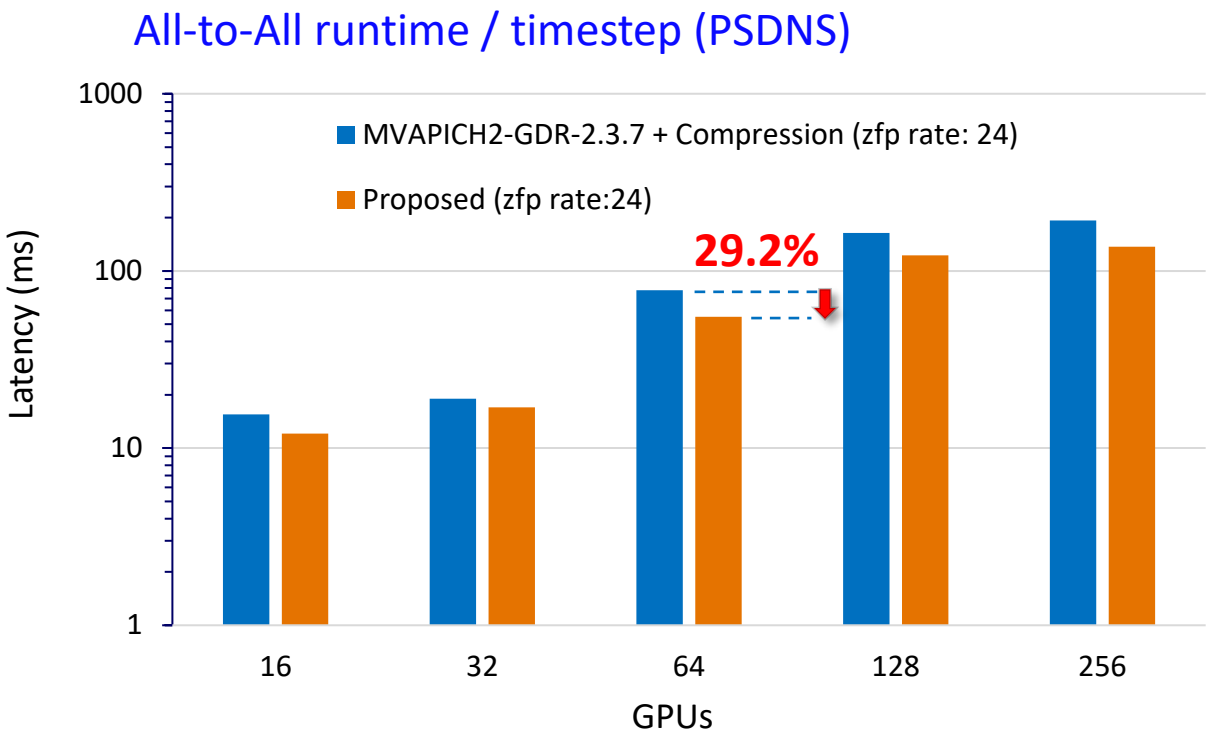
- Weak-Scaling of HPC application **AWP-ODC** on Lassen cluster (V100 nodes)
- MPC-OPT achieves up to **+18%** GPU computing flops, **-15%** runtime per timestep
- ZFP-OPT achieves up to **+35%** GPU computing flops, **-26%** runtime per timestep

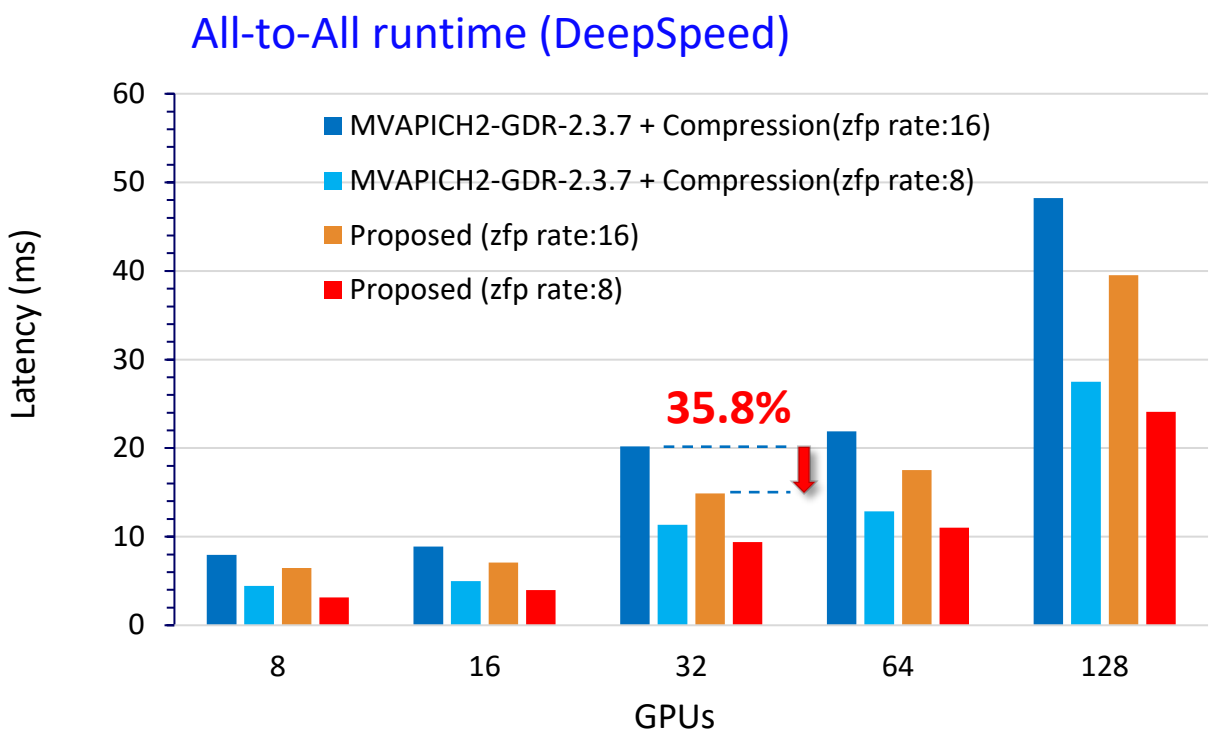




Q. Zhou, C. Chu, N. Senthil Kumar, P. Kousha, M. Ghazimirsaeed, H. Subramoni, and D.K. Panda, Designing High-Performance MPI Libraries with On-the-fly Compression for Modern GPU Clusters, 35th IEEE International Parallel & Distributed Processing Symposium (IPDPS), May 2021. [Best Paper Finalist]

Performance of All-to-All with Online Compression





- Improvement compared to MVAPICH2-GDR-2.3.7 with Point-to-Point compression
  - 3D-FFT: Reduce All-to-All runtime by up to 29.2% with ZFP(rate: 24) on 64 GPUs
  - DeepSpeed benchmark: Reduce All-to-All runtime by up to 35.8% with ZFP(rate: 16) on 32 GPUs

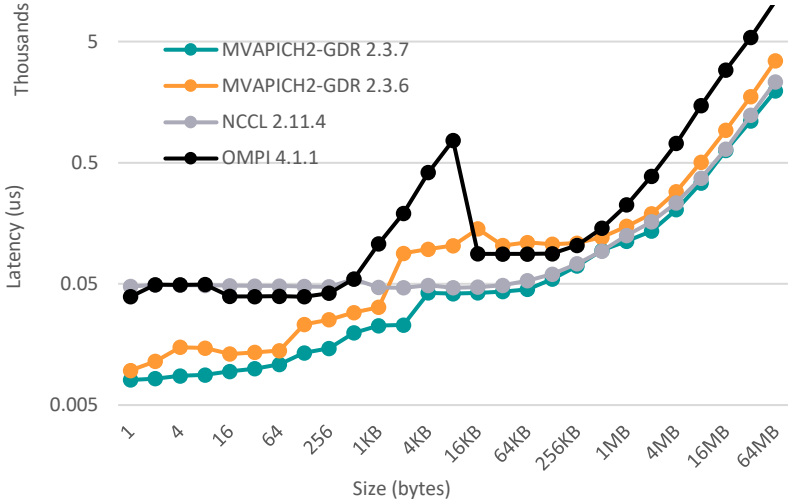
Q. Zhou, P. Kousha, Q. Anthony, K. Khorassani, A. Shafi, H. Subramoni, and D.K. Panda, "Accelerating MPI All-to-All Communication with Online Compression on Modern GPU Clusters", ISC '22.

Available in MVAPICH2-GDR 2.3.7

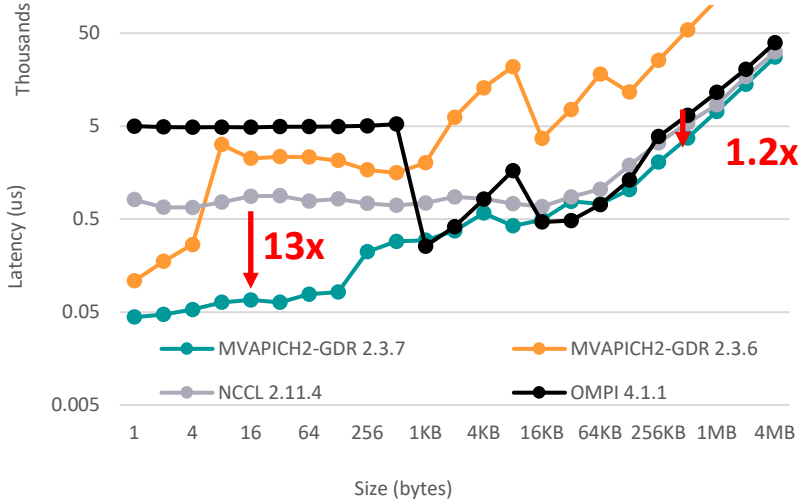
Highly Efficient Optimized Alltoall(v) Communication

Propose an optimized Alltoall(v) design to overlap inter (sendrecv-based) and intra-node (IPC-based) communication.

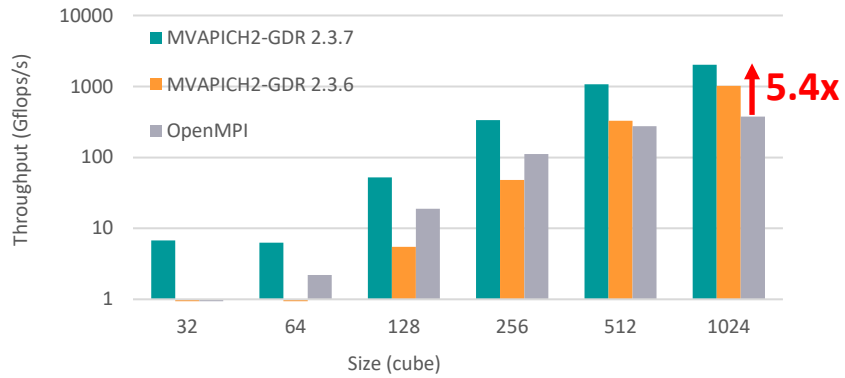
Alltoall latency on 1 node (8 GPUs)



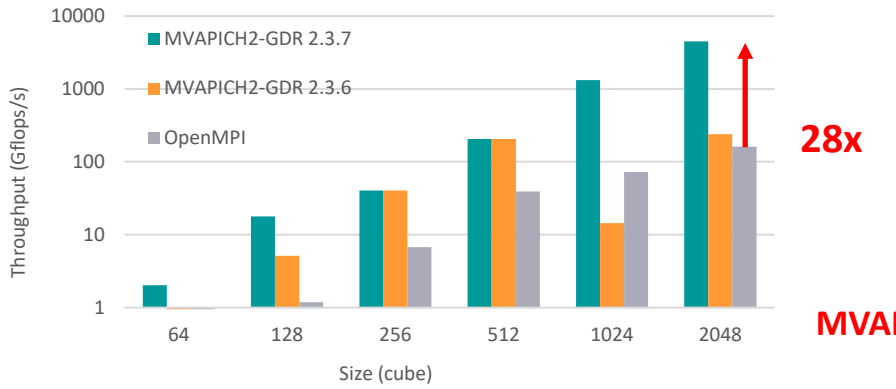
Alltoall latency on 16 nodes (128 GPUs)



heFFTe throughput (alltoallv) on 1 node (8 GPUs)



heFFTe throughput (alltoallv) on 16 node (128 GPUs)



Available in MVAPICH2-GDR v2.3.7

C.-C. Chen, K. Shafie Khorassani, Q. Anthony, A. Shafi, H. Subramoni and D. Panda, "Highly Efficient Alltoall and Alltoallv Communication Algorithms for GPU Systems", HCW '22

MVAPICH-Plus 3.0

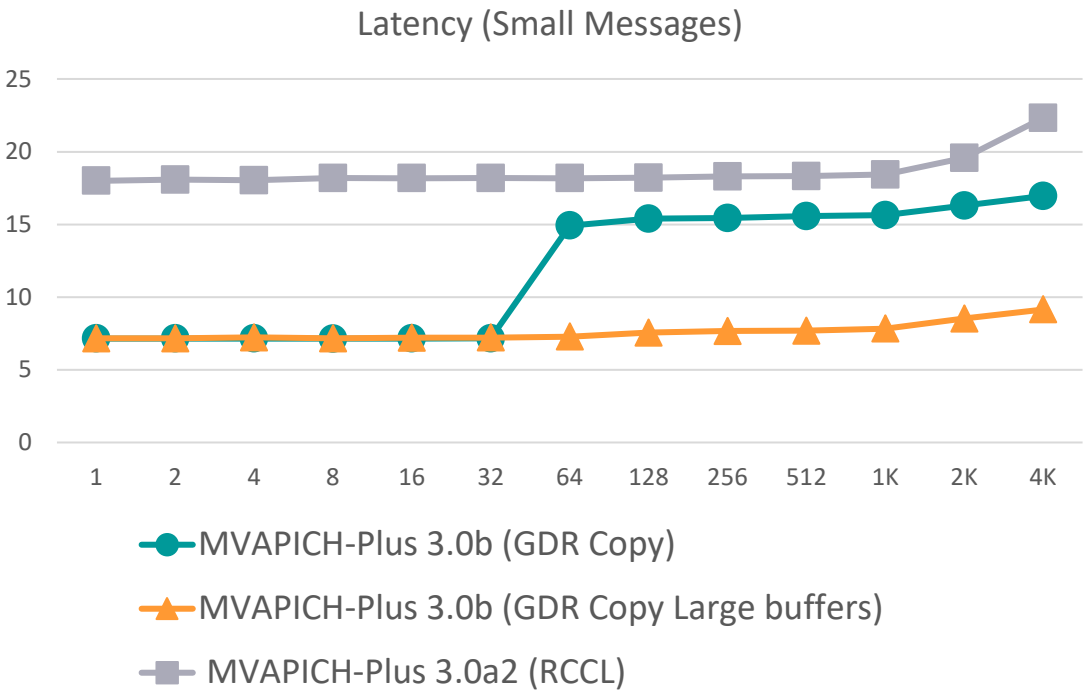
- Latest Release 03/08/2024
- Major Features and Enhancements
  - Based on MVAPICH 3.0
  - Supports all networks types supported by MVAPICH
  - Support for AMD and NVIDIA GPUs
  - Support for MVAPICH enhanced GPU Collectives
  - Support for enhanced GPU pt2pt operations
    - GDRCOPY and IPC
- Upcoming:
  - Combined features of MVAPICH2-X and MVAPICH2-GDR
  - Enhanced designs to leverage next generation Exascale systems
    - Frontier, El Capitan

MVAPICH-Plus 4.0

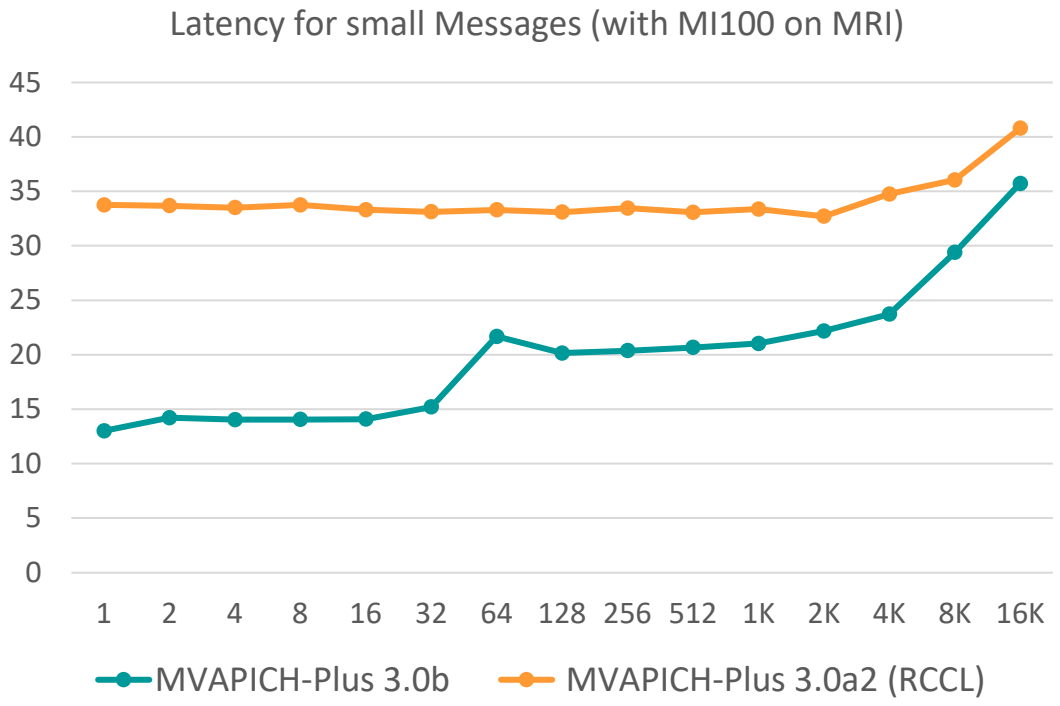
- Latest Beta Release 08/16/2024
- Major Features and Enhancements
  - Based on MPICH 4.2 with support for MPI 4.1
  - Supports all networks types supported by MVAPICH
    - Includes new OFI provider for IB systems, "mverbs;ofi_ucr"
  - Support for AMD, NVIDIA, and Intel GPUs
  - Support for MVAPICH enhanced GPU Collectives
  - Support for on-the-fly collective compression on NVIDIA GPUs
  - Support for enhanced GPU pt2pt operations with support for GDRCOPY and IPC
- Upcoming:
  - Combined features of MVAPICH2-X and MVAPICH2-GDR
  - Enhanced designs to leverage next generation Exascale systems
    - Frontier, El Capitan
  - AMD compression
  - Kernel based collective operations

MVAPICH-PLUS with GDRCopy Design on AMD GPUs

Latency:



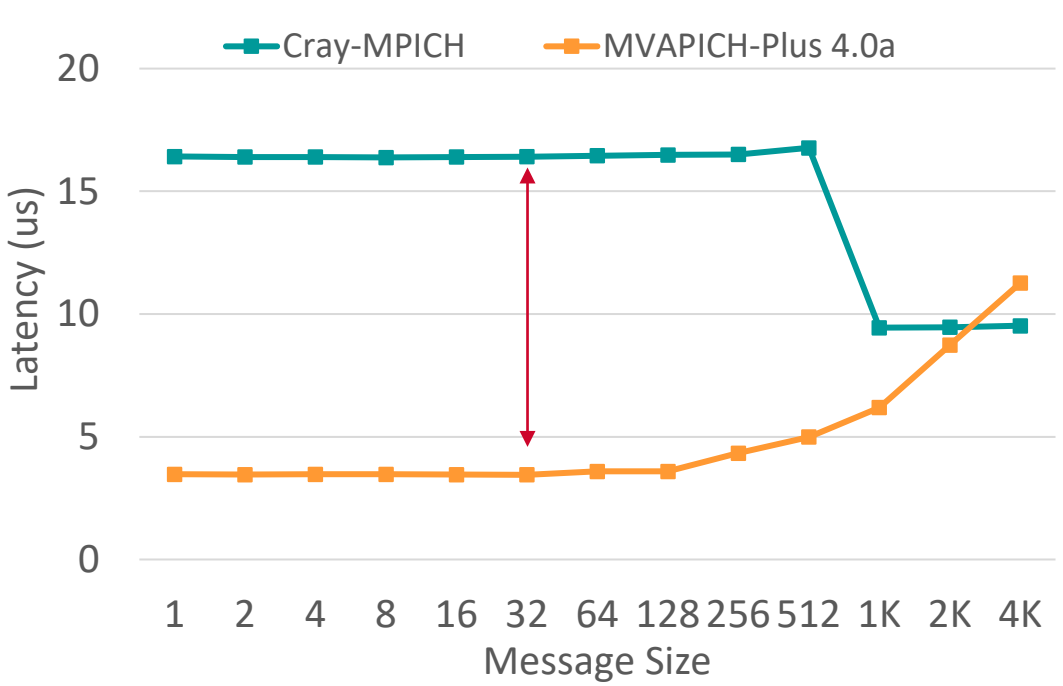
TIOGA - AMD MI100 GPUs



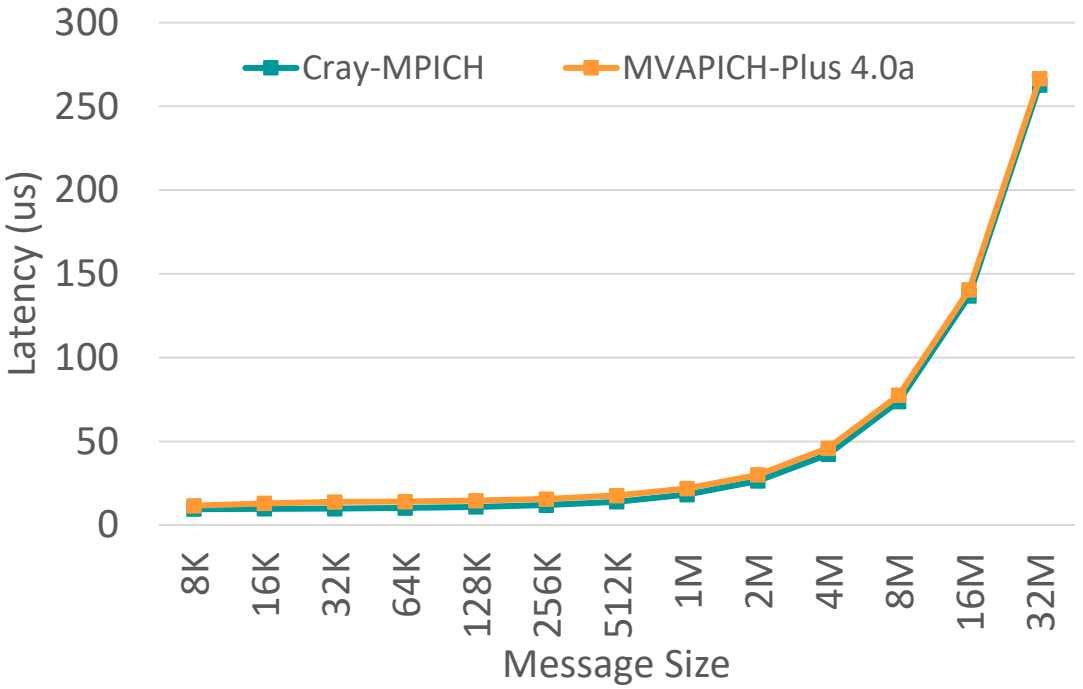
AMD GPUs on MRI

MVAPICH-PLUS GPU on Isambard (G-H + Slingshot) – Intranode PT2PT

Latency Intranode :

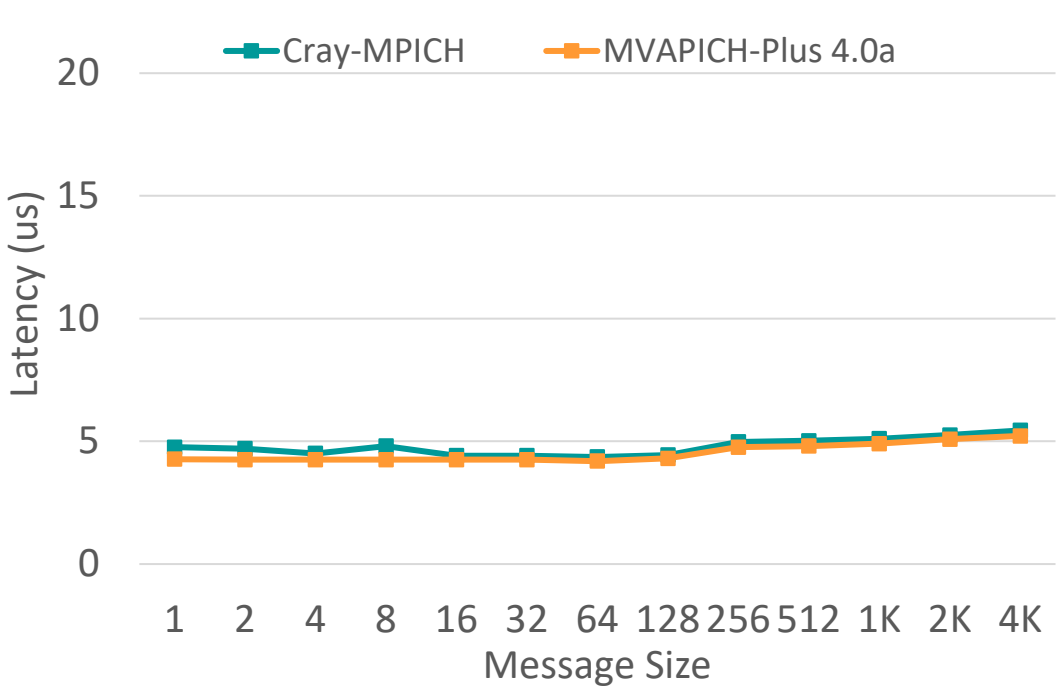


1 node, 2 GPN – Small Message (osu_latency)



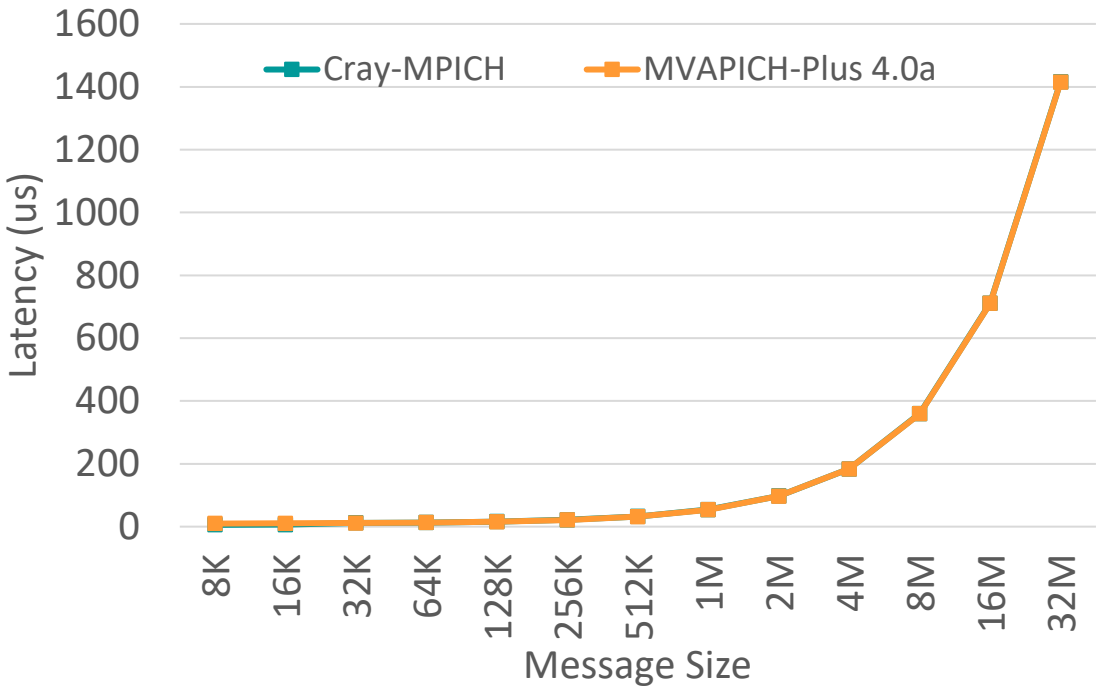
1 nodes, 2 GPN – Large Message (osu_latency)

MVAPICH-PLUS GPU on Isambard (G-H + Slingshot) – Internode PT2PT



2 nodes, 1 GPN – Small Message

(osu_latency)

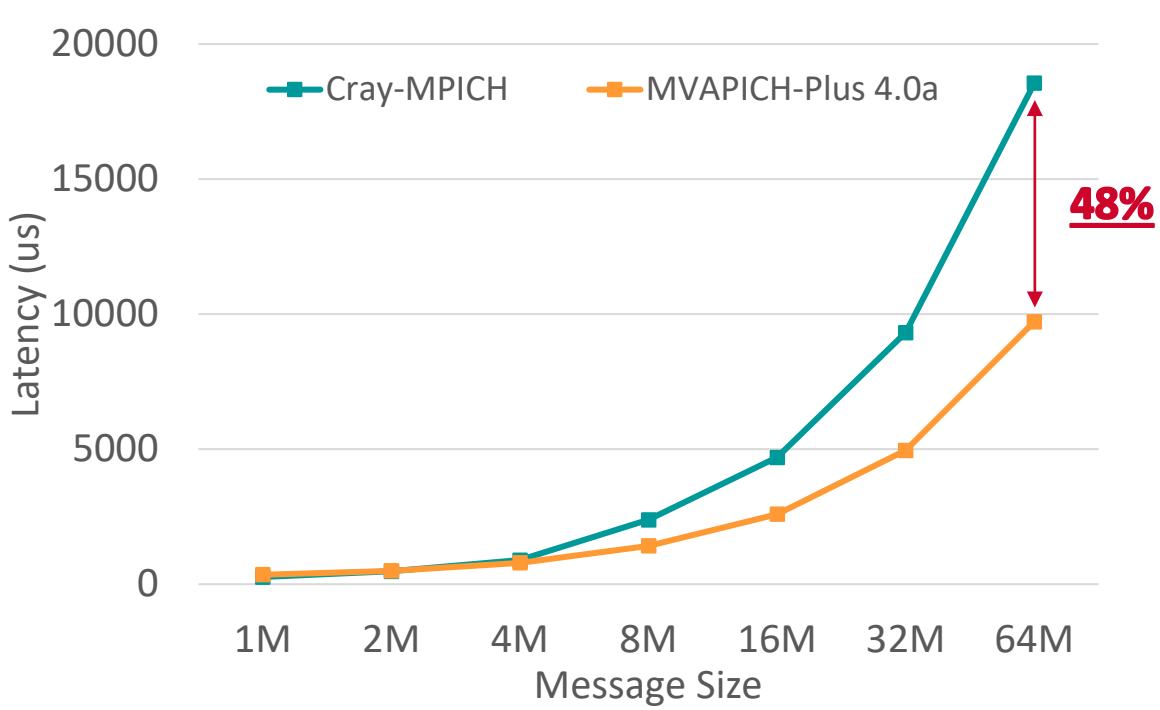


2 nodes, 1 GPN – Large Message

(osu_latency)

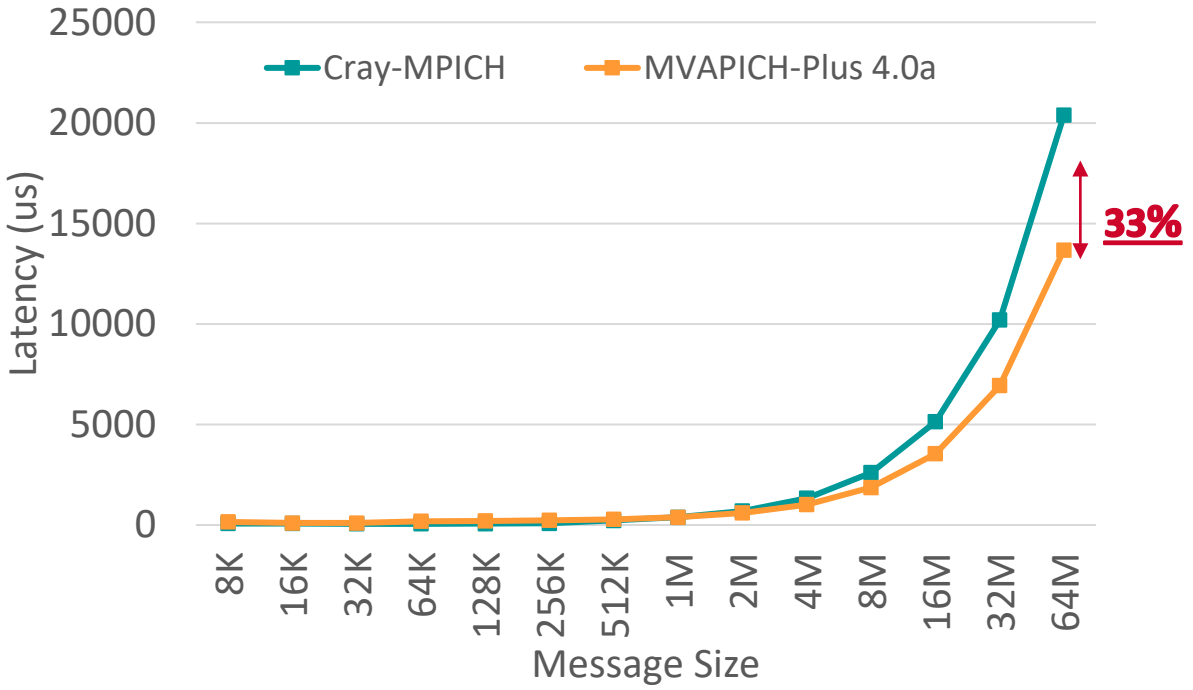
MVAPICH-PLUS GPU Collectives on Isambard (G-H + Slingshot)

Allgather



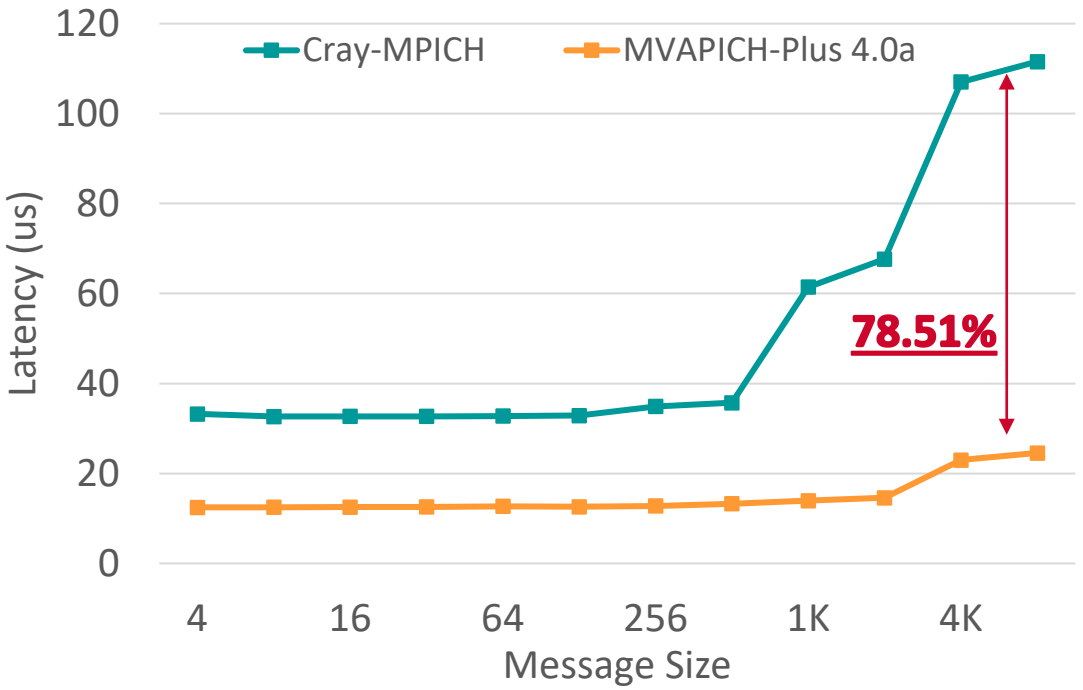
2 nodes, 4 GPN – Large Message

Alltoall

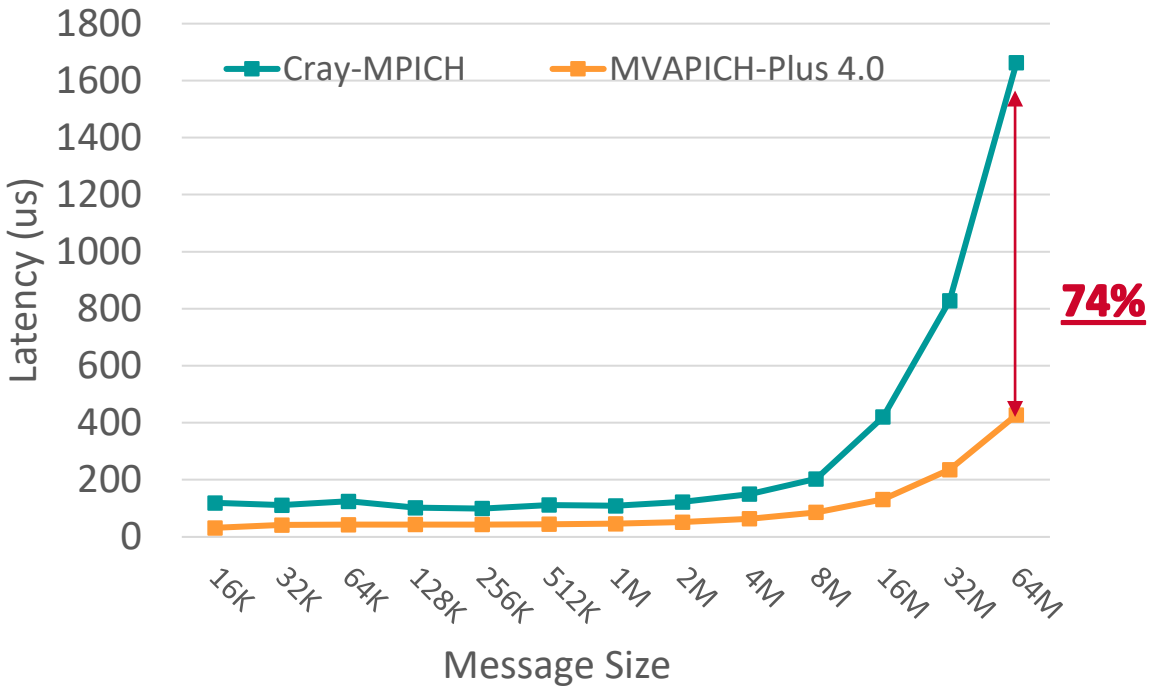


2 nodes, 4 GPN – Large Message

MVAPICH-PLUS GPU Optimized on Isambard – Allreduce on 1 Node



1 node, 4 GPN – Small Message



1 node, 4 GPN – Large Message

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/
- 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.

Case Study: GROMACS – Impact of Tuning Transport Protocol Experimental Setup

- **Platform:**
  - **Broadcom RoCEv2 Thor Adapter**
  - **64 Nodes x 2 x AMD EPYC 7713 64-Core Processor**
- **Application:**
  - **GROMACS v2022.3**
  - **Dataset: 3000k-atoms dataset**
- **Raw run lines:**
  - **OpenMPI 4.1.3 + UCX 1.14**

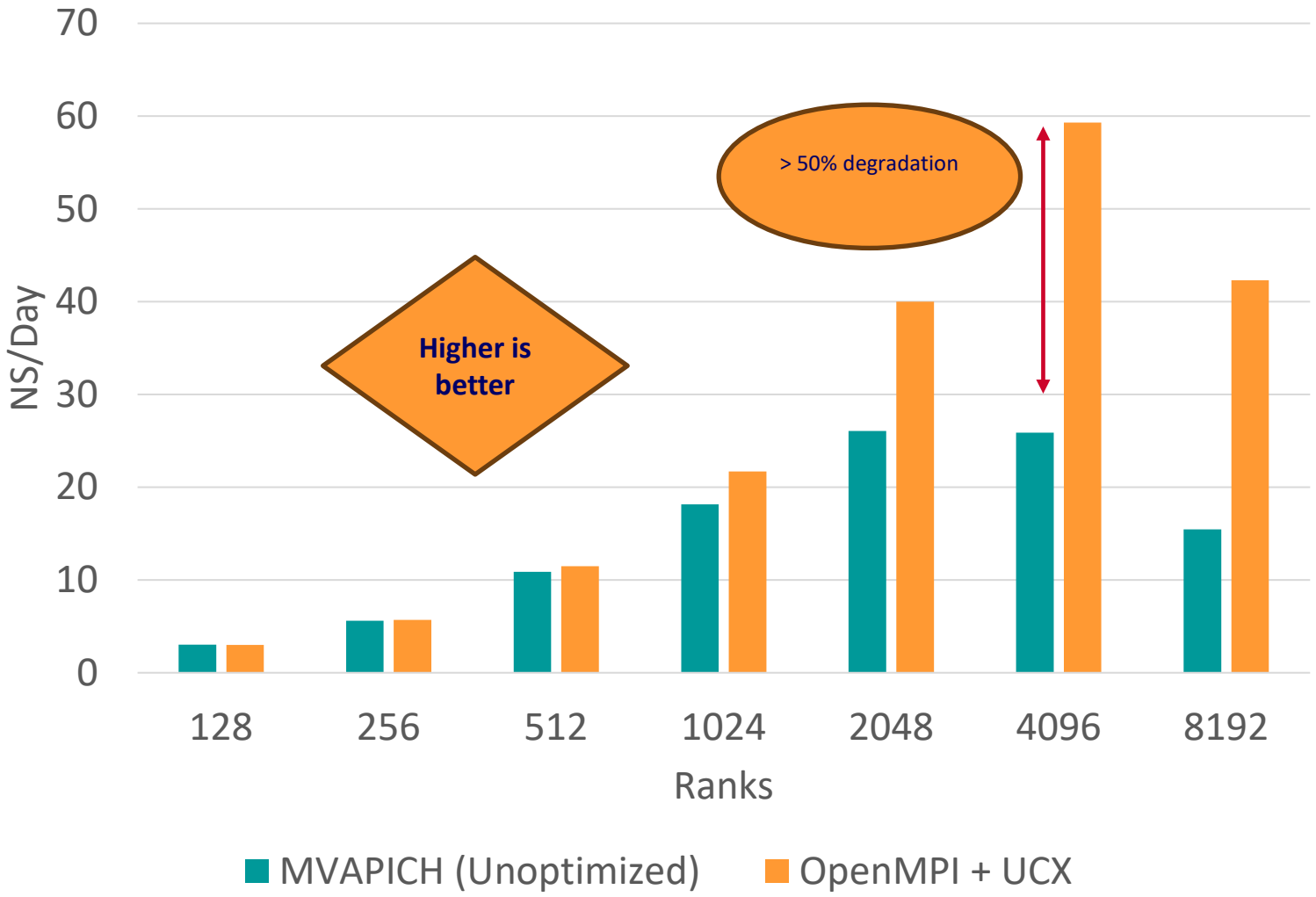
```
export OMPI_PARAM="--mca pml ucx --mca osc ucx --mca spml ucx --mca btl
^vader,tcp,openib,uct -x UCX_NET_DEVICES=bnxt_re0:1 -x UCX_IB_GID_INDEX=3 -x
UCX_TLS=self,sm,rc_v”
mpirun --hostfile $HOSTFILE $OMPI_PARAM gmx_mpi mdrun -ntomp 1 -s $GROMACS_BENCHMARK -deffnm
md -nsteps 10000
```

  - **MVAPICH2-2.3.7-Broadcom**

```
mpirun_rsh --export-all -np $NP -ppn $PPN run_mpi gmx_mpi mdrun -ntomp 1 -s
$GROMACS_BENCHMARK -deffnm md -nsteps 10000
```

Case Study: GROMACS – Impact of Tuning Transport Protocol First experiment – Unoptimized version

- Strong scaling the GROMACS application performance
- We are measuring the nanoseconds of simulated time per day
  - Higher is better
- Degradation observed beyond 1K MPI processes
- This is the unoptimized MVAPICH2-2.3.7 version
- Need to use TAU to see what MPI calls are causing the degradation

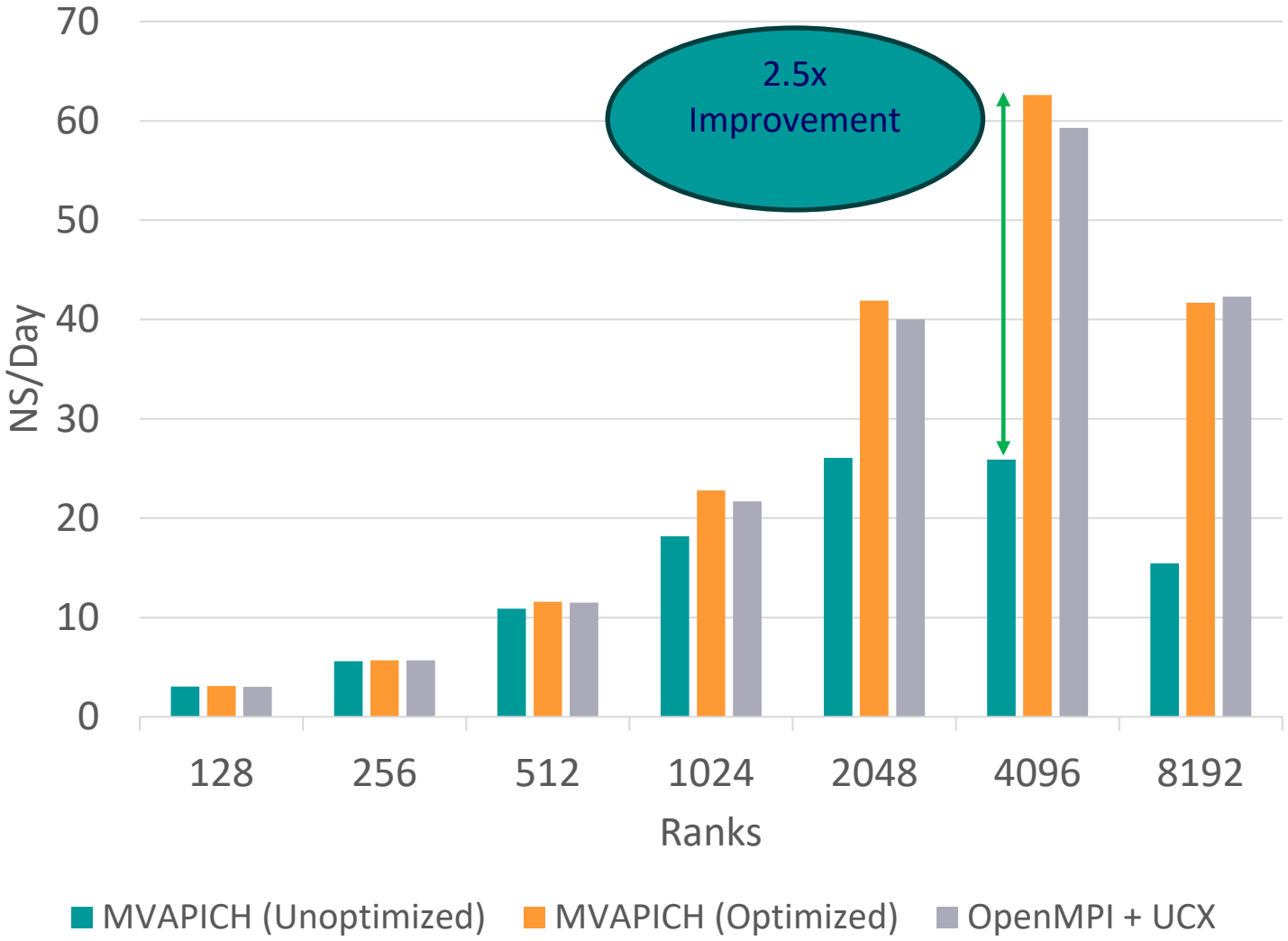


Case Study: GROMACS – Performance Engineering with TAU Diagnosis and workaround found

- Investigate UD communication (read progress poll)
- Use RC to get the desired lead in performance
- Gains:
  - 2.5x improvement over MVAPICH baseline
  - 15% compared to OpenMPI default RC
- Update the following parameter for GROMACS runs

`MV2_HYBRID_ENABLE_THRESHOLD = 8192`

this will enable UD-hybrid communication after the 8192 threshold*



*For more details check user-guide:
https://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-userguide.html#:~:text=use%20any%20HugePages.-,11.110,-MV2_HYBRID_ENABLE_THRESHOLD

Case Study: 3D Stencil – Impact of Tuning Eager Threshold Experimental Setup

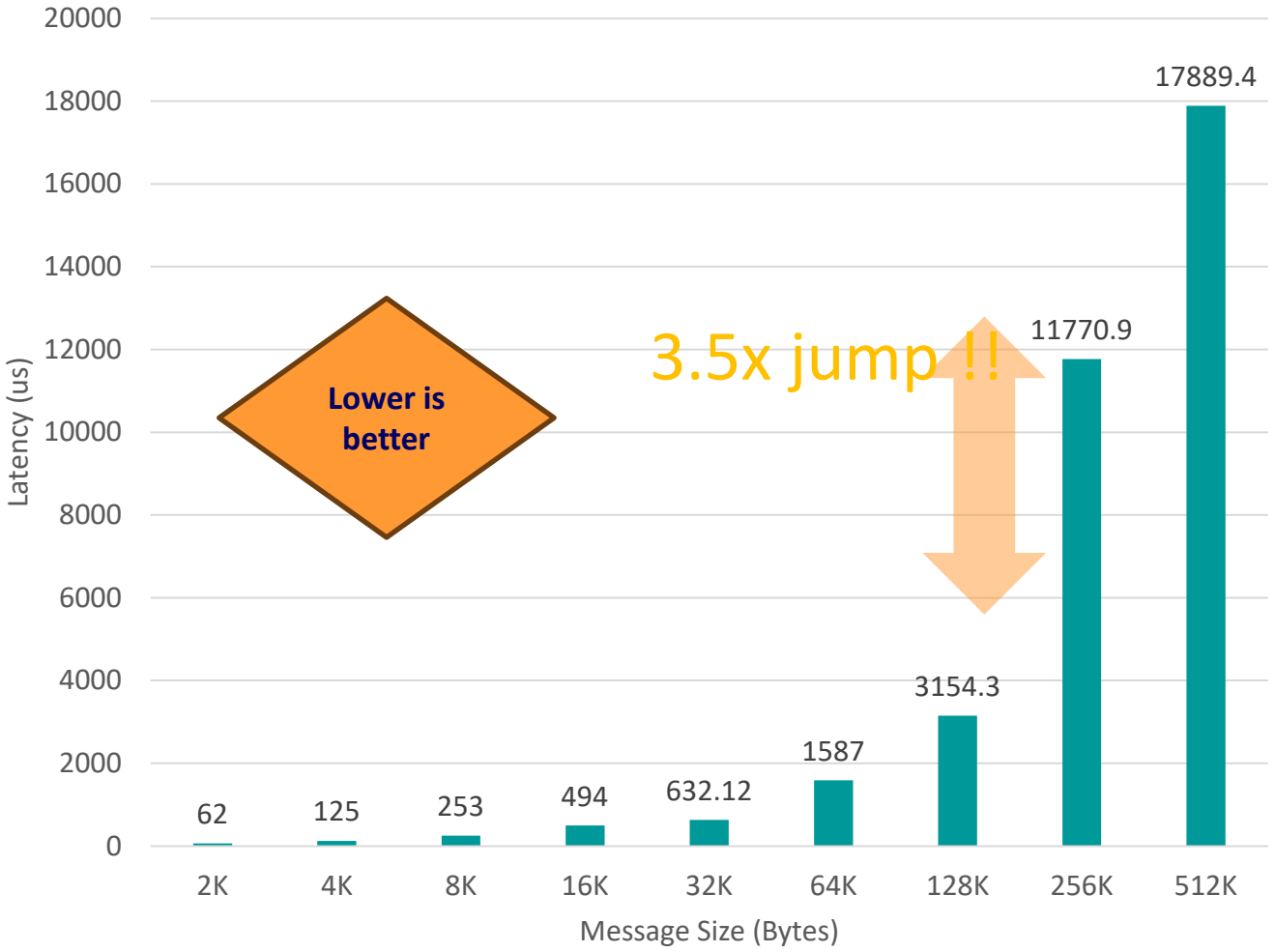
- **Platform:**
  - **Broadcom RoCEv2 Thor Adapter**
  - **64 Nodes x 2 x AMD EPYC 7713 64-Core Processor**
- **Application:**
  - **3D Stencil HPC Benchmark**
  - **Dataset: 3000k-atoms dataset**
- **Raw run lines:**

  - **MVAPICH2-2.3.7-Broadcom**

```
mpirun_rsh -np $NP -ppn $PPN ./3Dstencil_overlap 8 8 8 1000
```

Case Study: 3D Stencil – Impact of Tuning Eager Threshold First experiment – Unoptimized version

- Execution time tests on 2 Nodes x 128 PPN (512 ranks)
- We are measuring the latency
  - Lower is better
- Degradation observed at 256K message
- This is the unoptimized MVAPICH2-2.3.7 version
- Need to use TAU to see
  - what MPI calls are causing the degradation
  - What is the dominant communication pattern

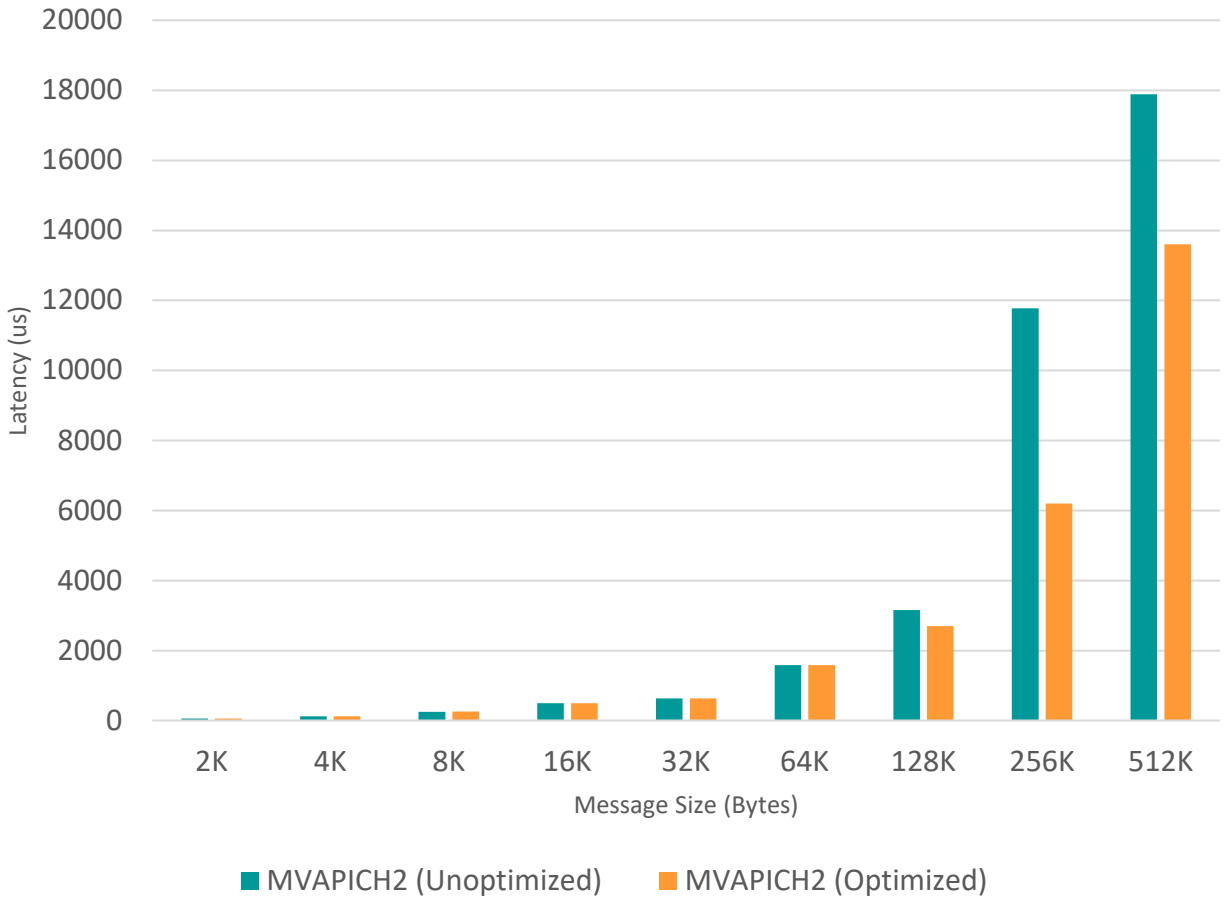


Case Study: 3D Stencil – Impact of Tuning Eager Threshold Diagnosis and workaround found

- Diagnosis: more time is spent in inter-node pt-to-pt Rendezvous communication
- Solution: Use pt-to-pt eager communication
- Gains:
  - 2x reduction in latency
- Update the following parameter for the 3D Stencil runs

`MV2_IBA_EAGER_THRESHOLD = 524288`

this will enable inter-node eager communication until the specified message size*



*For more details check user-guide:
https://mvapich.cse.ohio-state.edu/static/media/mvapich/mvapich2-userguide.html#:~:text=for%20the%20job.-,12.5,-MV2_IBA_EAGER_THRESHOLD

MVAPICH – Future Roadmap and Plans for Exascale

- Performance and Memory scalability toward 1M-10M cores
- Hybrid programming (MPI + OpenSHMEM, MPI + UPC, MPI + CAF …)
  - MPI + Task*
- Enhanced Optimization for GPUs and FPGAs*
- Taking advantage of advanced features of Mellanox InfiniBand
  - Tag Matching*
  - Adapter Memory*
- Enhanced communication schemes for upcoming architectures
  - NVLINK*
  - InfiniyFabric*
  - Bluefield-3*
- 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 MVAPICH Releases

Funding Acknowledgments

Funding Support by

Equipment Support by

Acknowledgments to all the Heroes (Past/Current Students and Staffs)

Current Students (Graduate)

- N. Alnaasan (Ph.D.)
- Q. Anthony (Ph.D.)
- C.-C. Chun (Ph.D.)
- N. Contini (Ph.D.)
- T. Chen (Ph.D.)
- J. Hatef (Ph.D.)
- H-R. Huang (Ph.D)
- P. Kousha (Ph.D.)
- S. Lee (Ph.D.)
- B. Michalowicz (Ph.D.)
- A. Potlapally (Ph. D.)
- B. Ramesh (Ph.D.)
- K. K. Suresh (Ph.D.)
- A. H. Tu (Ph.D.)
- S. Xu (Ph.D.)
- Q. Zhou (Ph.D.)
- K. Al Attar (Ph.D.)
- L. Xu (Ph.D.)
- G. Kuncham (Ph.D.)
- J. Yao (Ph.D.)
- M. Han (M.S.)

Current Faculty

- H. Subramoni

Current Research Scientists

- A. Shafi

Past Students (Undergrads)

- J. Sulewski
- T. Chen

Current Software Engineers

- N. Pavuk
- N. Shineman
- M. Lieber
- A. Gupta

Current Research Specialist

- R. Motlagh

Past Students

- A. Awan (Ph.D.)
- A. Augustine (M.S.)
- P. Balaji (Ph.D.)
- M. Bayatpour (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. Chakraborthy (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- C.-H. Chu (Ph.D.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- A. Guptha (M.S.)
- J. Hashmi (Ph.D.)
- W. Huang (Ph.D.)
- A. Jain (Ph.D.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- K. S. Khorassani (Ph.D.)
- M. Kedia (M.S.)
- 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.)
- K. Raj (M.S.)
- R. Rajachandrasekar (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- N. Sarkauskas (B.S. and M.S)
- N. Senthil Kumar (M.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Srivastava (M.S.)
- S. Sur (Ph.D.)
- H. Subramoni (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)

Past Post-Docs

- D. Banerjee
- X. Besseron
- M. S. Ghazimirsaeed
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- K. Manian
- S. Marcarelli
- A. Ruhela
- J. Vienne
- H. Wang

Past Senior Research Associate

- J. Hashmi

Past Programmers

- A. Reifsteck
- D. Bureddy
- J. Perkins
- B. Seeds

Past Research Specialist

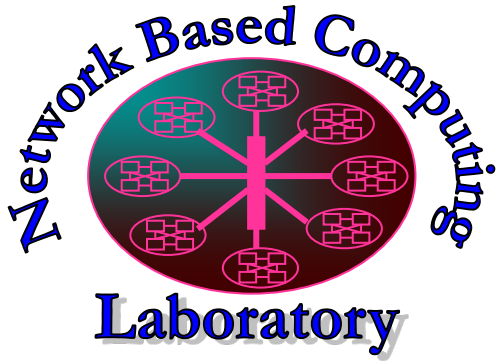
- M. Arnold
- J. Smith

Past Research Scientists

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

Thank You!

panda@cse.ohio-state.edu, subramon@cse.ohio-state.edu



Follow us on

https://twitter.com/mvapich

Network-Based Computing Laboratory

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

The High-Performance MPI/PGAS Project

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

The High-Performance Big Data Project

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

The High-Performance Deep Learning Project

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