MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Latest version of the slides available at

http://cse.osu.edu/~subramon/mug20-mvapich2-tutorial.pdf

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

A Tutorial at MUG'20

by

The MVAPICH Team

The Ohio State University

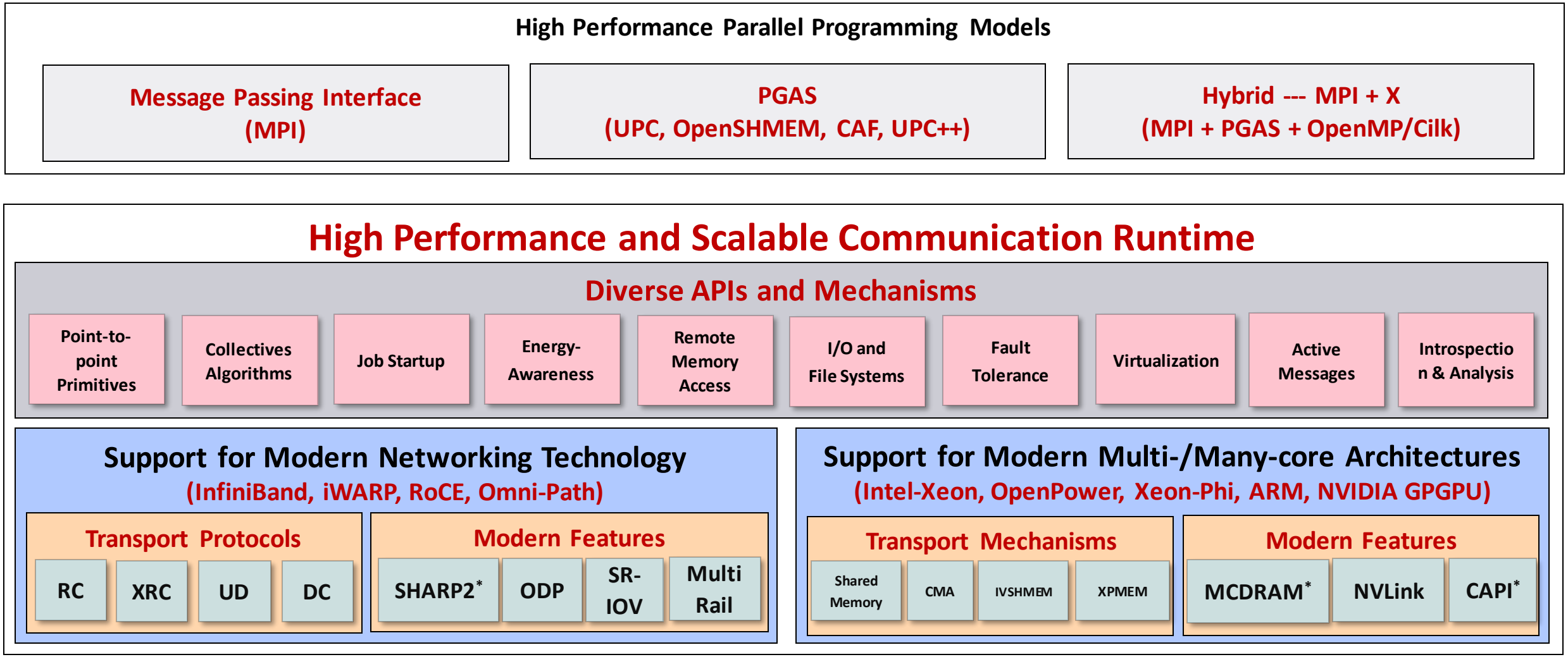
http://mvapich.cse.ohio-state.edu/

Overview of the MVAPICH2 Project

- High Performance open-source MPI Library
- Support for multiple interconnects
  - InfiniBand, Omni-Path, Ethernet/iWARP, RDMA over Converged Ethernet (RoCE), and AWS EFA
- Support for multiple platforms
  - x86, OpenPOWER, ARM, Xeon-Phi, GPGPUs (NVIDIA and AMD (upcoming))
- Started in 2001, first open-source version demonstrated at SC '02
- Supports the latest MPI-3.1 standard
- http://mvapich.cse.ohio-state.edu
- Additional optimized versions for different systems/environments:
  - MVAPICH2-X (Advanced MPI + PGAS), since 2011
  - MVAPICH2-GDR with support for NVIDIA GPGPUs, since 2014
  - MVAPICH2-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 3,100 organizations in 89 countries**
- **More than 820,000 (> 0.8 million) downloads from the OSU site directly**
- Empowering many TOP500 clusters (June '20 ranking)
  - 4th, 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 8th, 448, 448 cores (Frontera) at TACC
  - 12th, 391,680 cores (ABCI) in Japan
  - 18th, 570,020 cores (Nurion) in South Korea and many others
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 8th ranked TACC Frontera system
- **Empowering Top500 systems for more than 15 years**

Architecture of MVAPICH2 Software Family (for HPC and DL)



* Upcoming

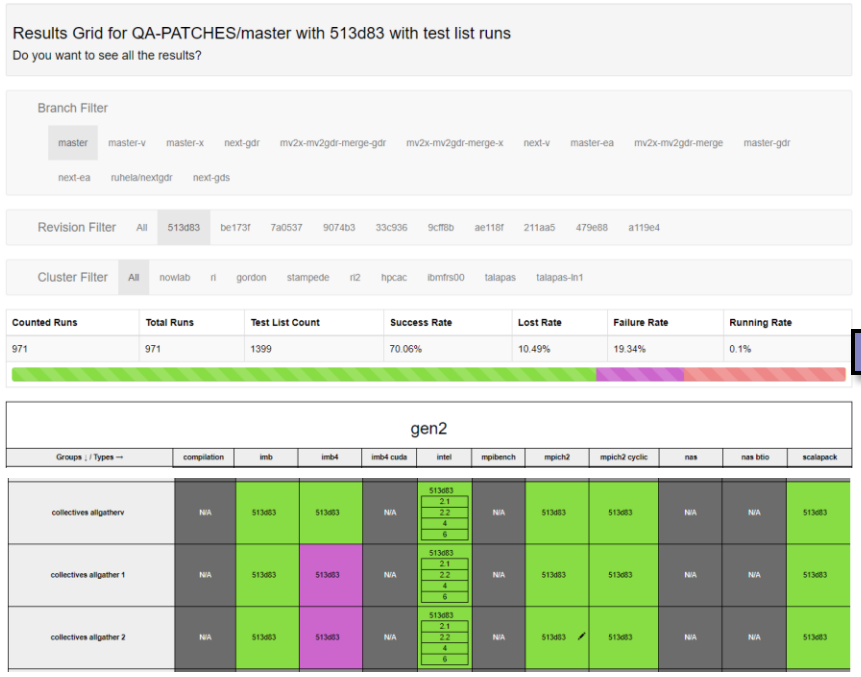
Production Quality Software Design, Development and Release

- 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

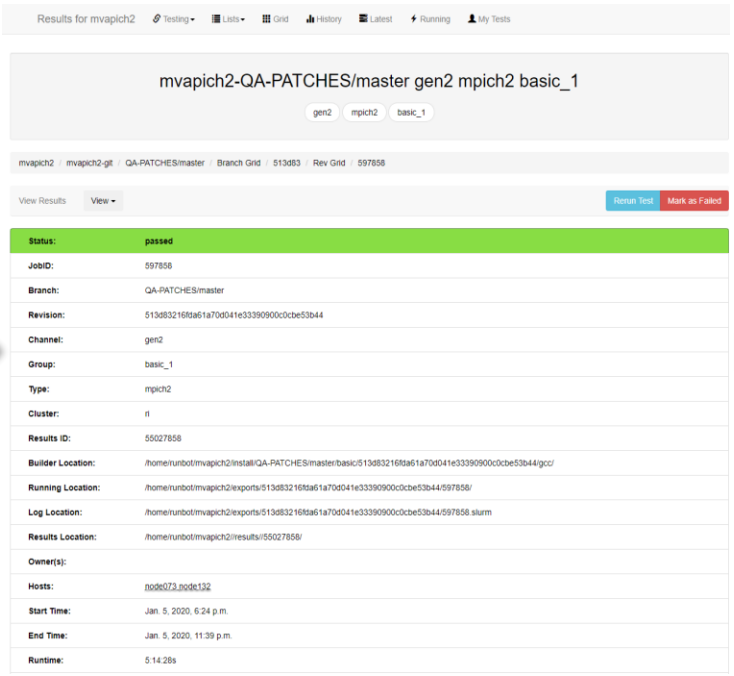
Automated Procedure for Testing Functionality

- 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 different combinations of *environment variables* meant to trigger different communication paths in MVAPICH2

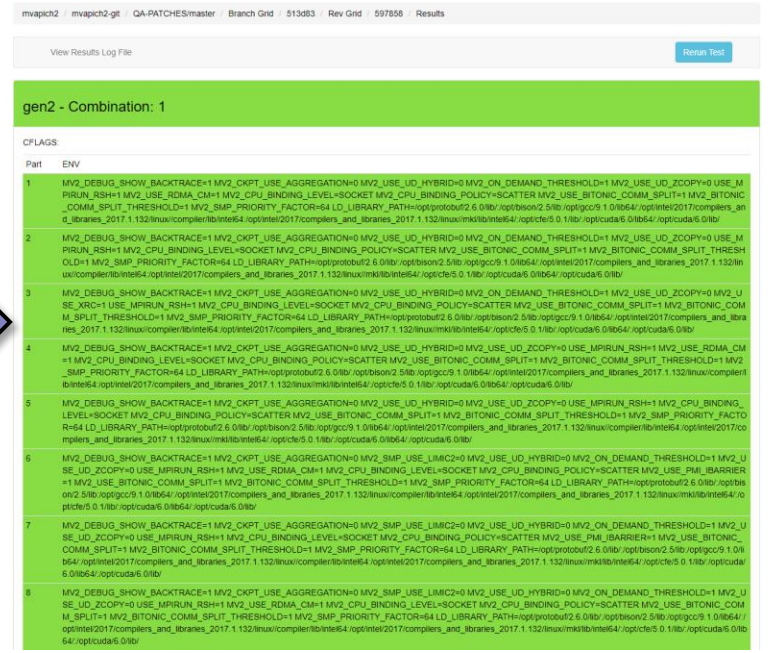
Summary of all tests for one commit



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 3642.22 -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 %

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

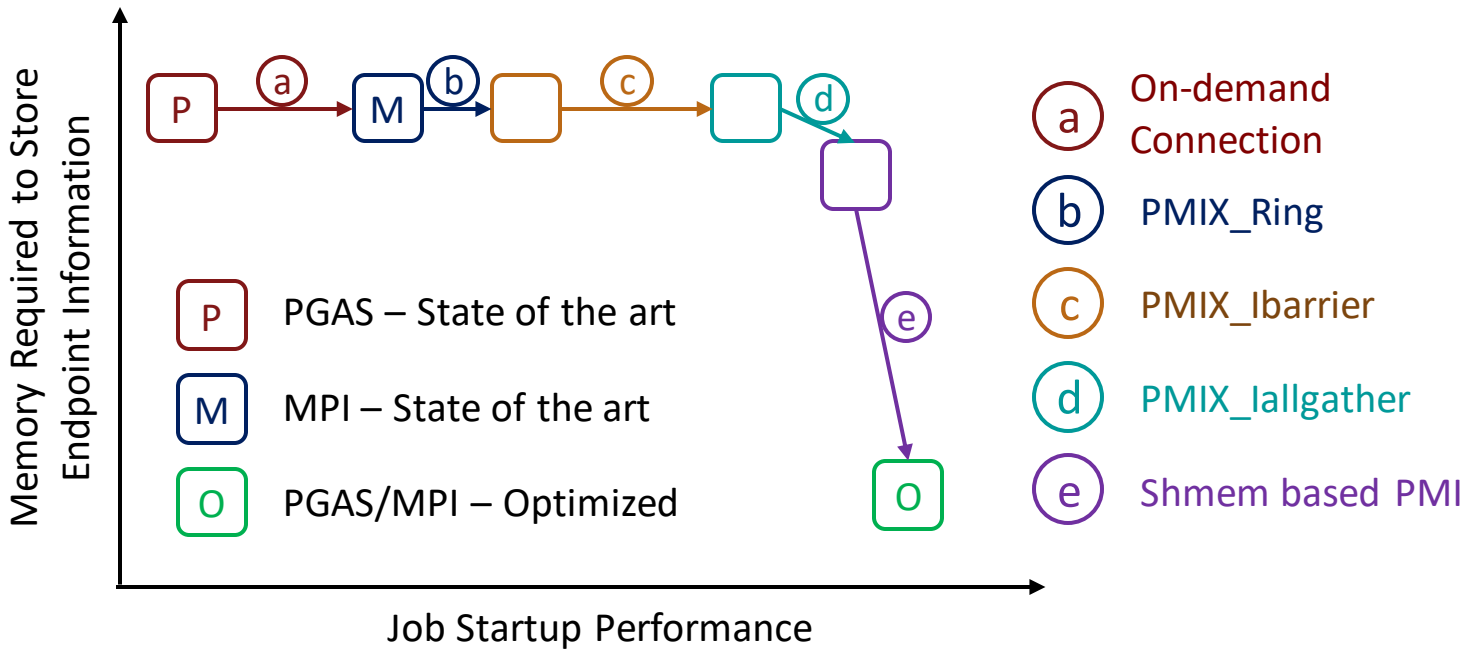
MVAPICH2 Software Family

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 MVAPICH2 Features

- Job start-up
- Transport Type Selection
- Process Mapping and Point-to-point Intra-node Protocols
- Collectives

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(processes per node)
- 1GB Memory saved per node with 1M processes and 16 processes per node

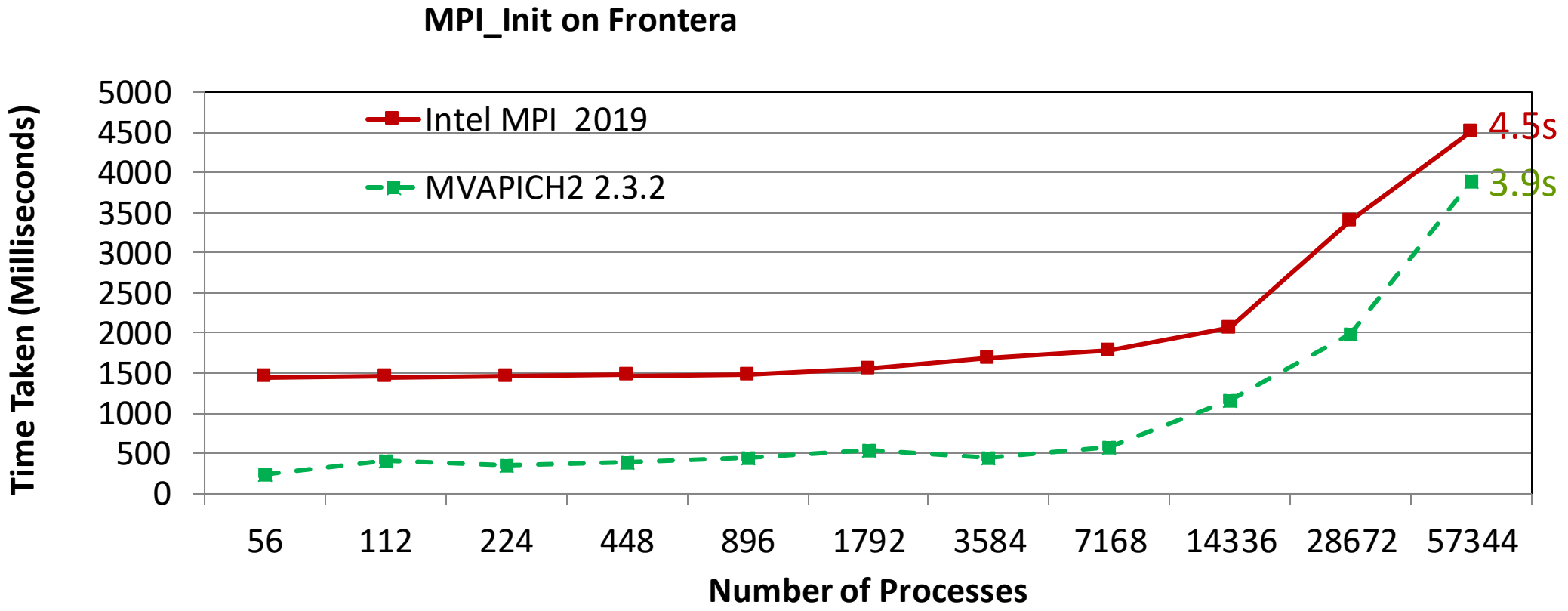
(a) **On-demand Connection Management for OpenSHMEM and OpenSHMEM+MPI.** S. Chakraborty, H. Subramoni, J. Perkins, A. A. Awan, and D K Panda, 20th International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS '15)

(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 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 since MVAPICH2-2.3.2

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

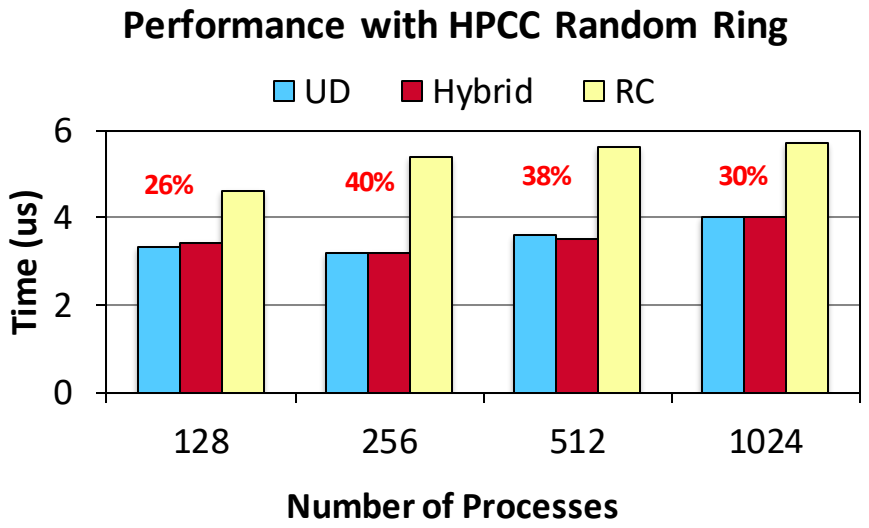
Using SLURM as launcher

- **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

Transport Protocol Selection in MVAPICH2

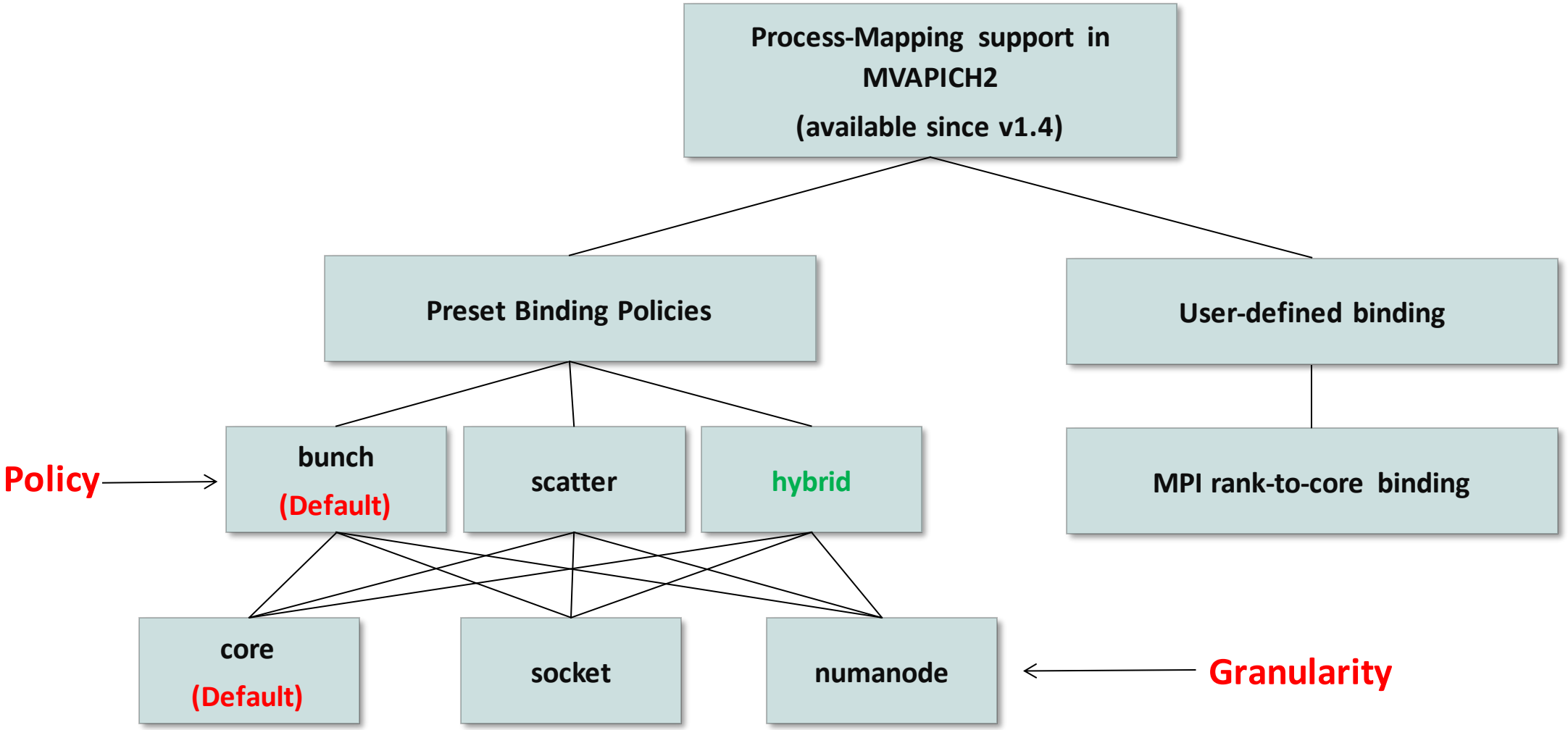


- Both UD and RC/XRC have benefits
  - Hybrid for the best of both
- Enabled by configuring MVAPICH2 with the –enable-hybrid
- Available since MVAPICH2 1.7 as integrated interface

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**

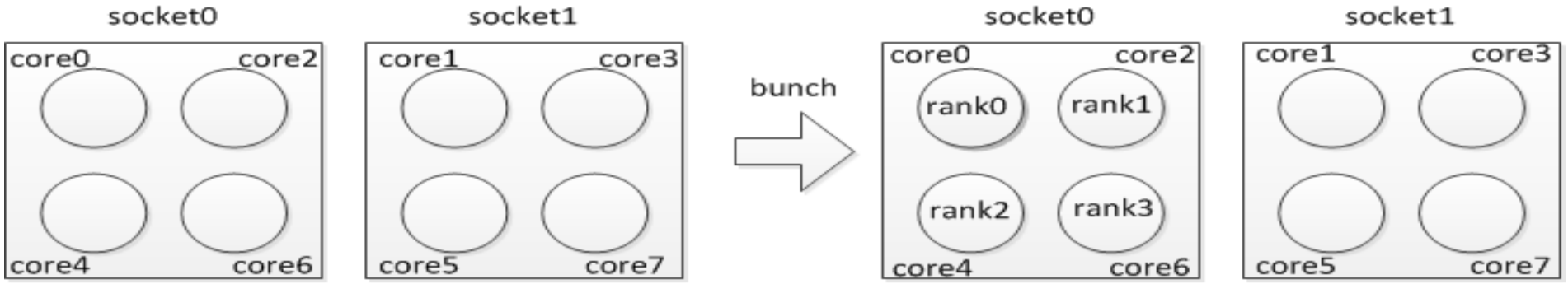
Process Mapping support in MVAPICH2



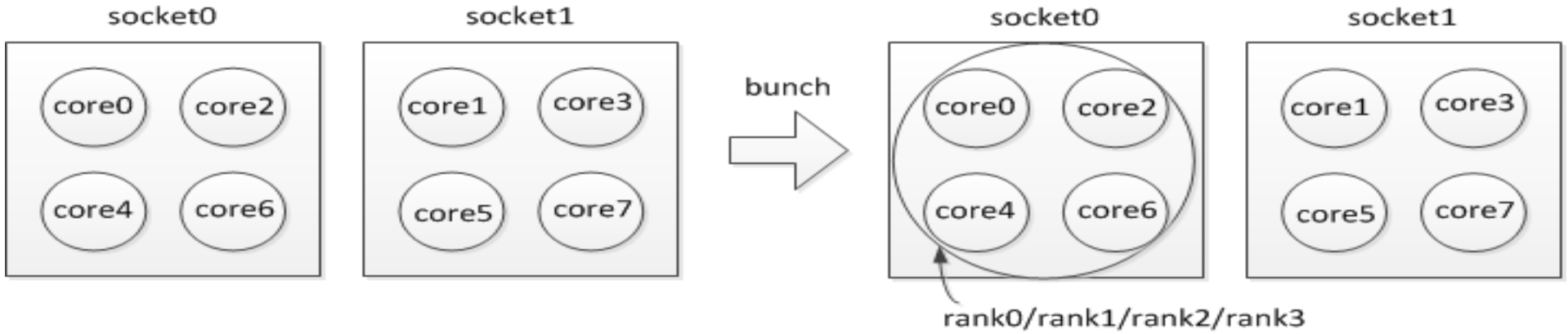
MVAPICH2 detects processor architecture at job-launch

Preset Process-binding Policies – Bunch

- "Core" level "Bunch" mapping (Default)
  - MV2_CPU_BINDING_POLICY=bunch

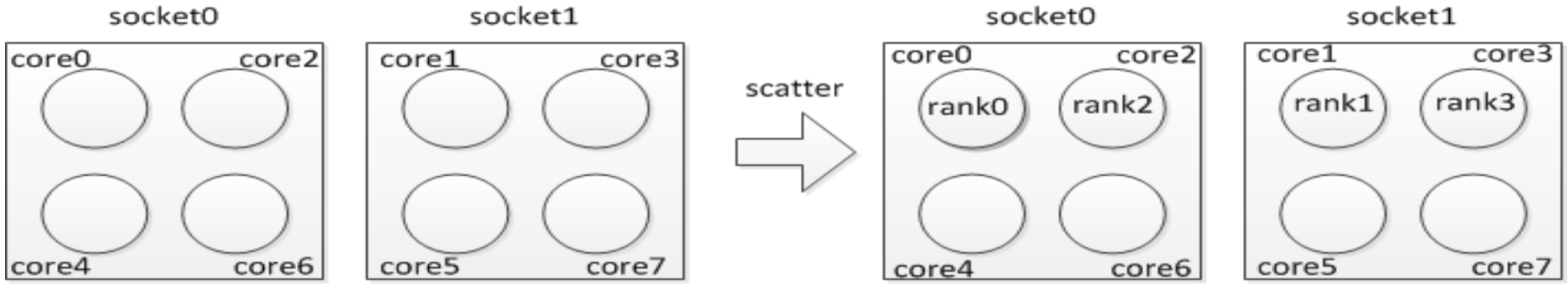


- "Socket/Numanode" level "Bunch" mapping
  - MV2_CPU_BINDING_LEVEL=socket MV2_CPU_BINDING_POLICY=bunch

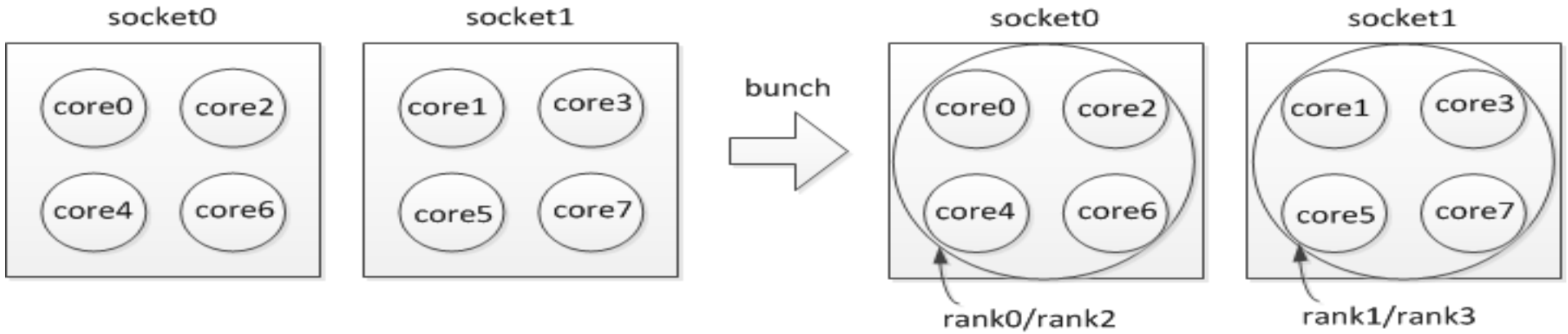


Preset Process-binding Policies – Scatter

- "Core" level "Scatter" mapping
  - MV2_CPU_BINDING_POLICY=scatter



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

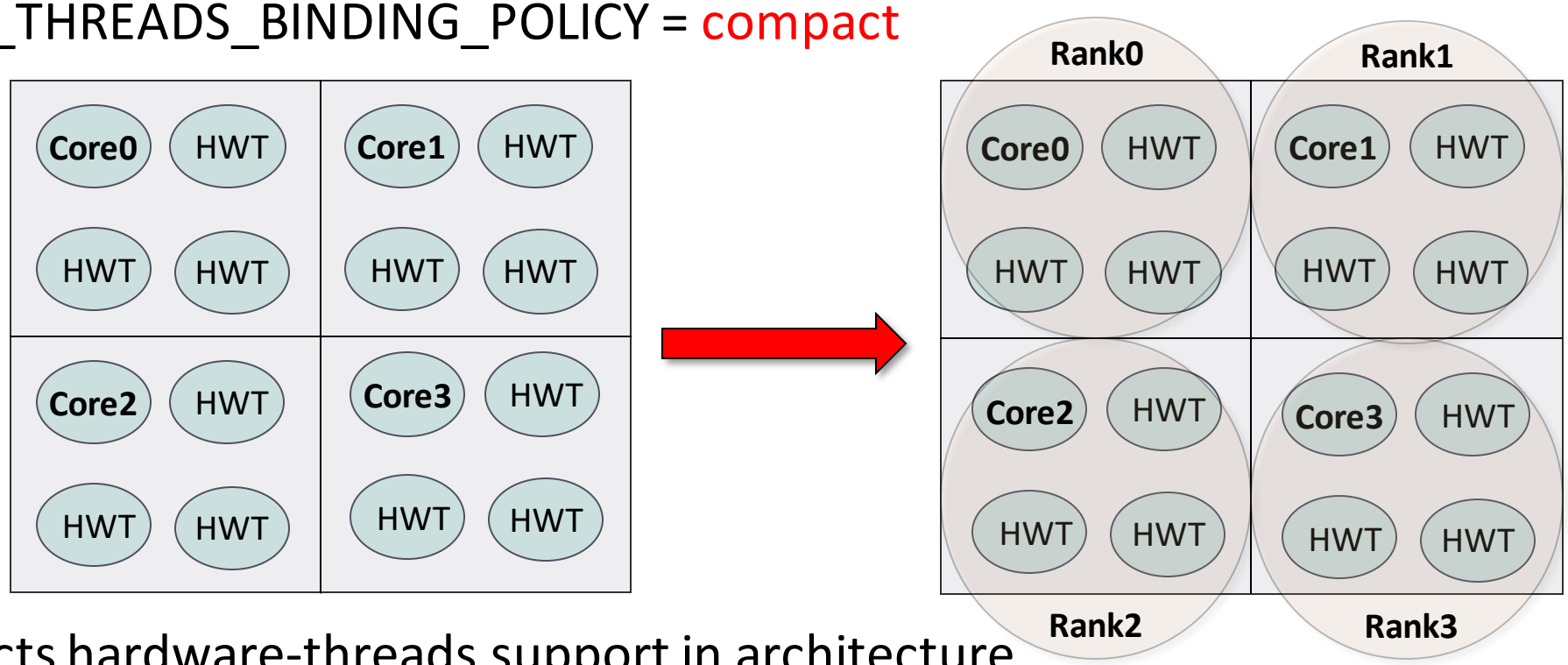


Process and thread binding policies in hybrid MPI+Threads

- 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)

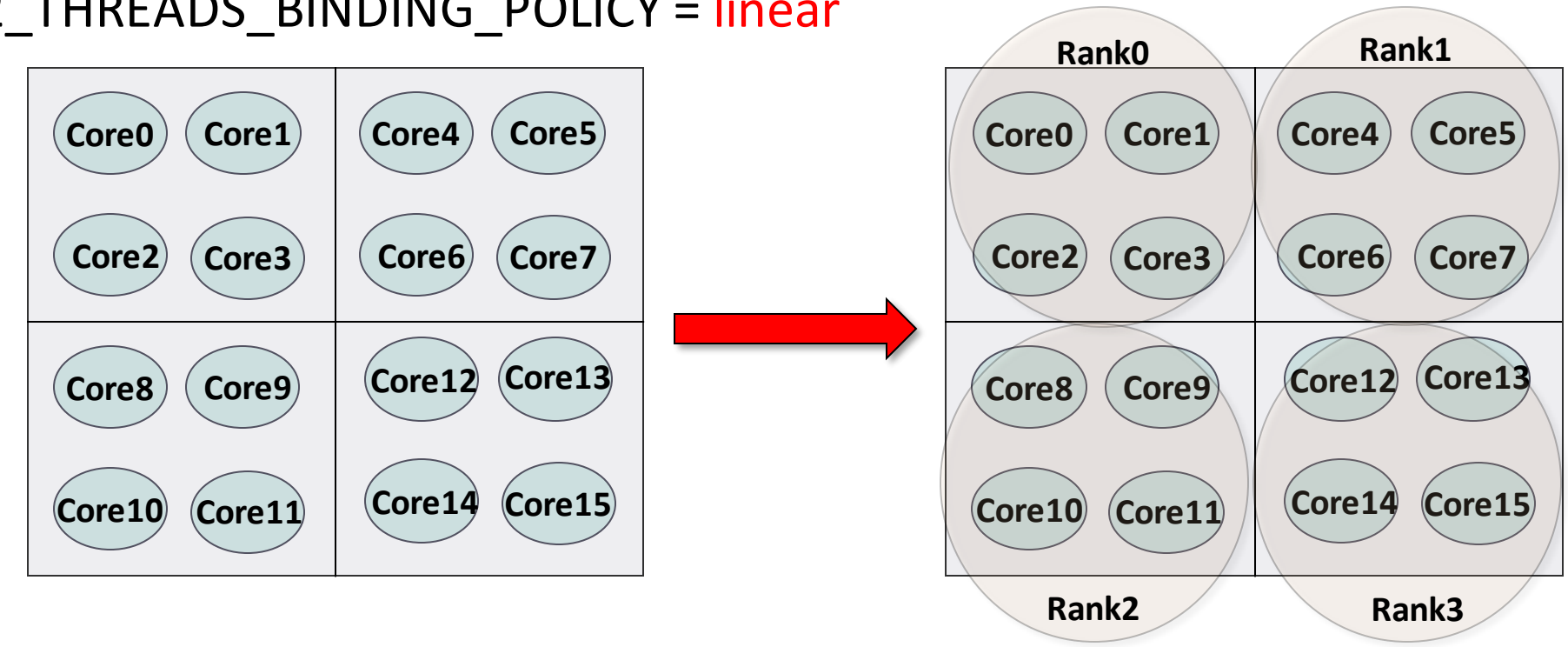
- MPI Processes = 4, OpenMP Threads per Process = 4
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_THREADS_PER_PROCESS = 4
- MV2_THREADS_BINDING_POLICY = compact



- 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

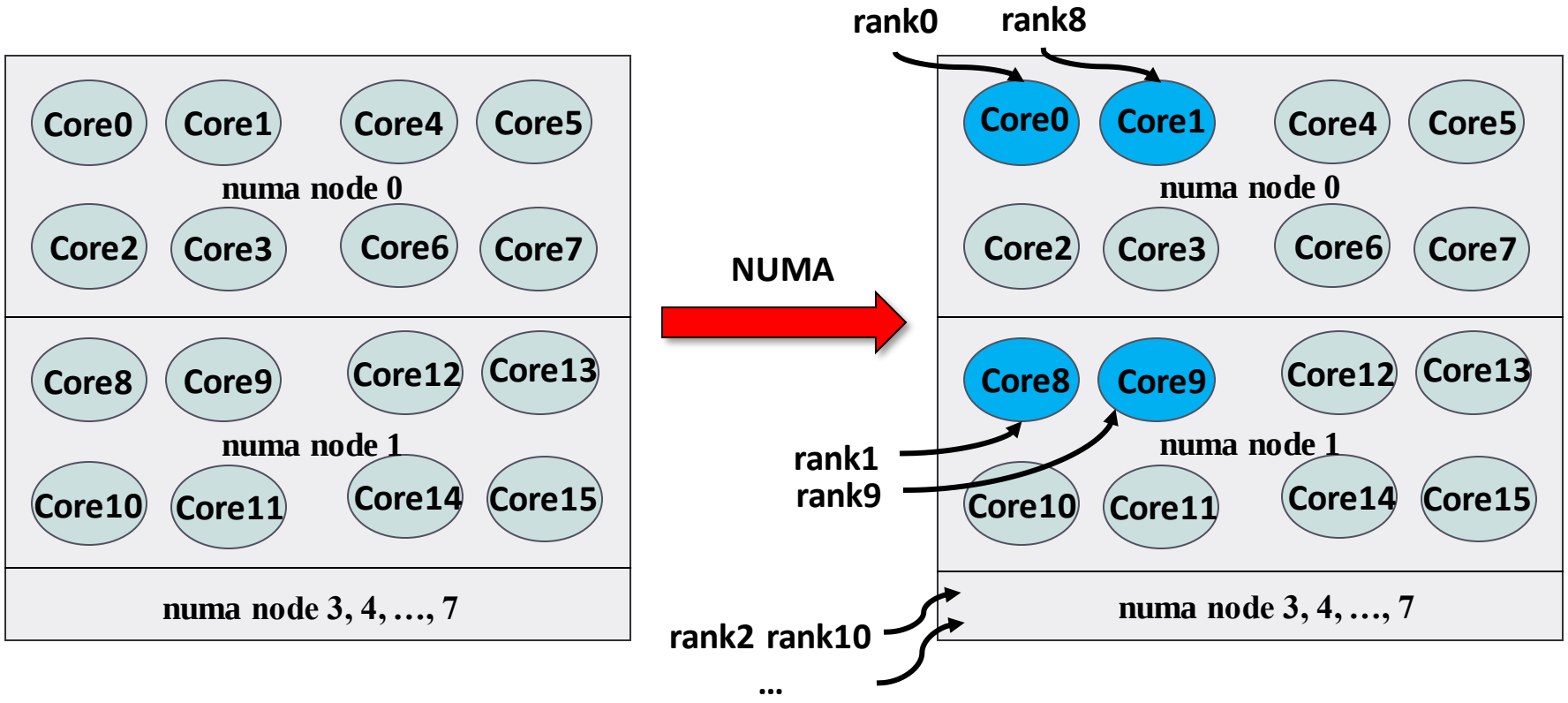
- MPI Processes = 4, OpenMP Threads per Process = 4
- MV2_CPU_BINDING_POLICY = hybrid
- MV2_THREADS_PER_PROCESS = 4
- MV2_THREADS_BINDING_POLICY = linear



- 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



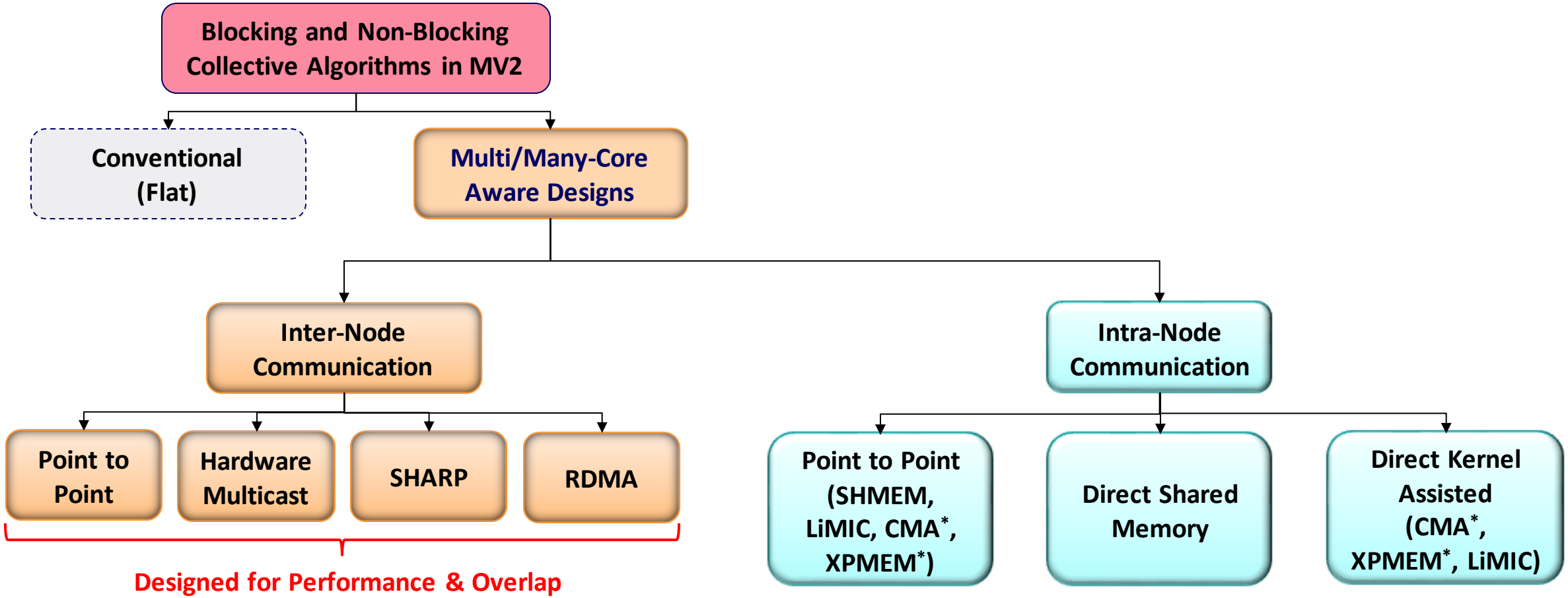
User-Defined Process Mapping

- 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**

Collective Communication in MVAPICH2



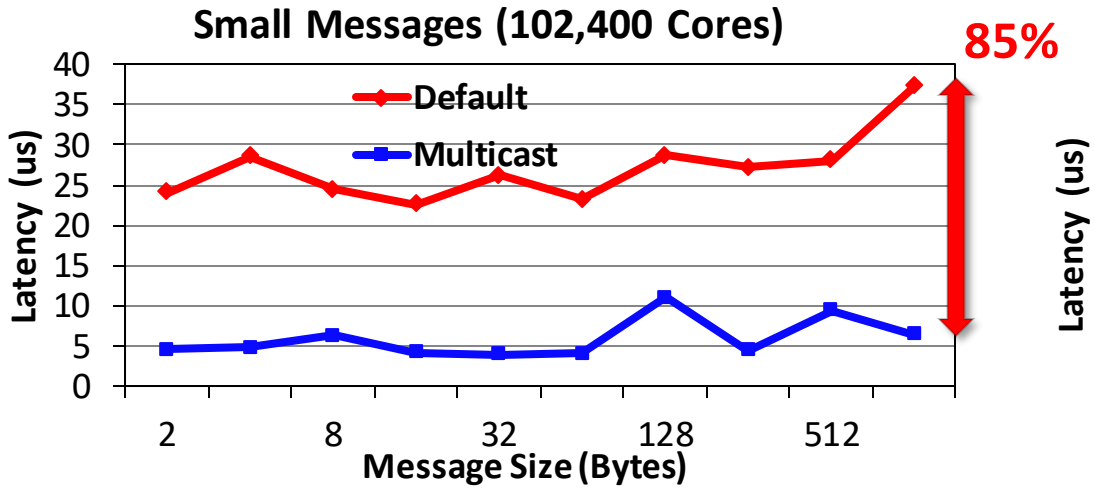
Run-time flags:

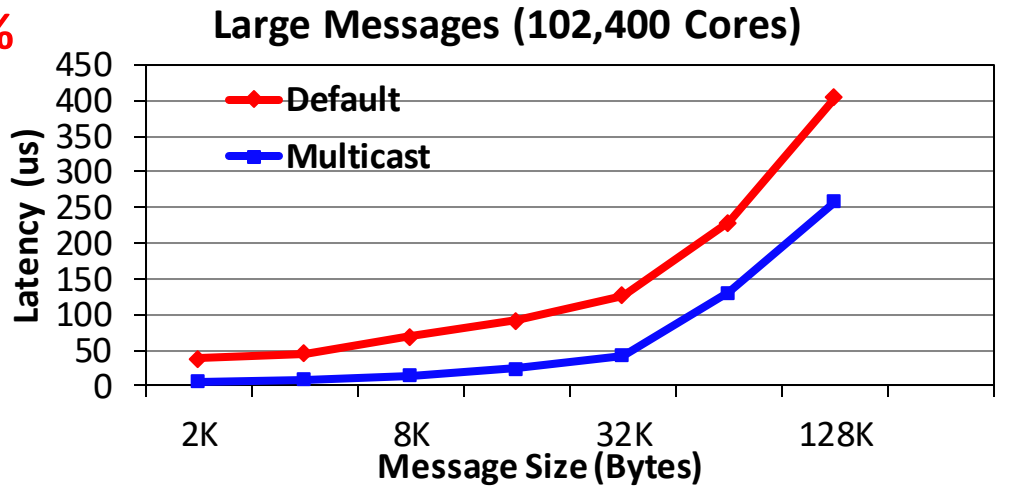
All shared-memory based collectives : MV2_USE_SHMEM_COLL (Default: ON)

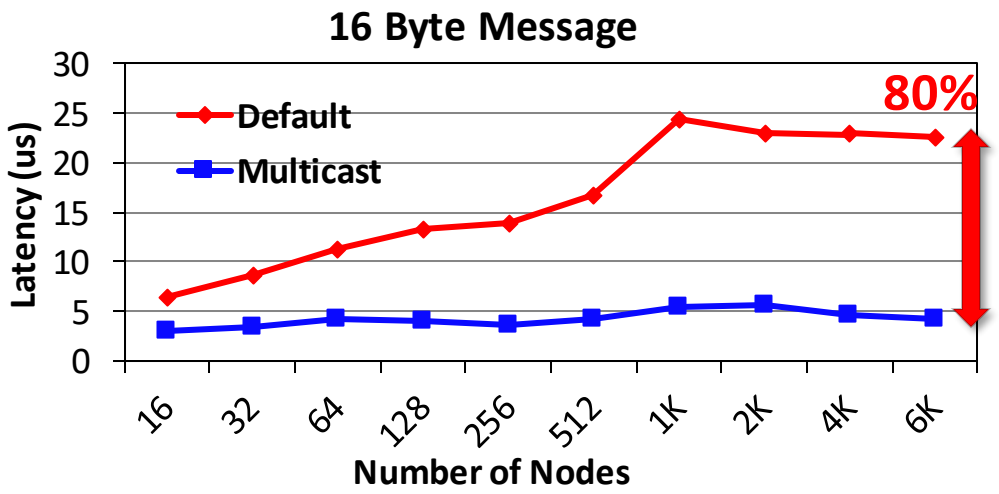
Hardware Mcast-based collectives : MV2_USE_MCAST (Default : OFF)

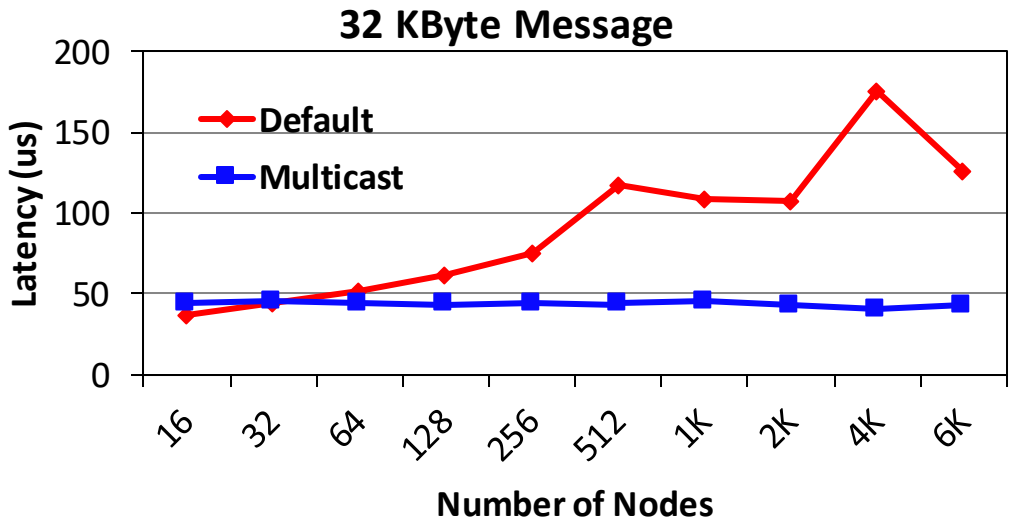
CMA and XPMEM-based collectives are in MVAPICH2-X

Hardware Multicast-aware MPI_Bcast on TACC Stampede



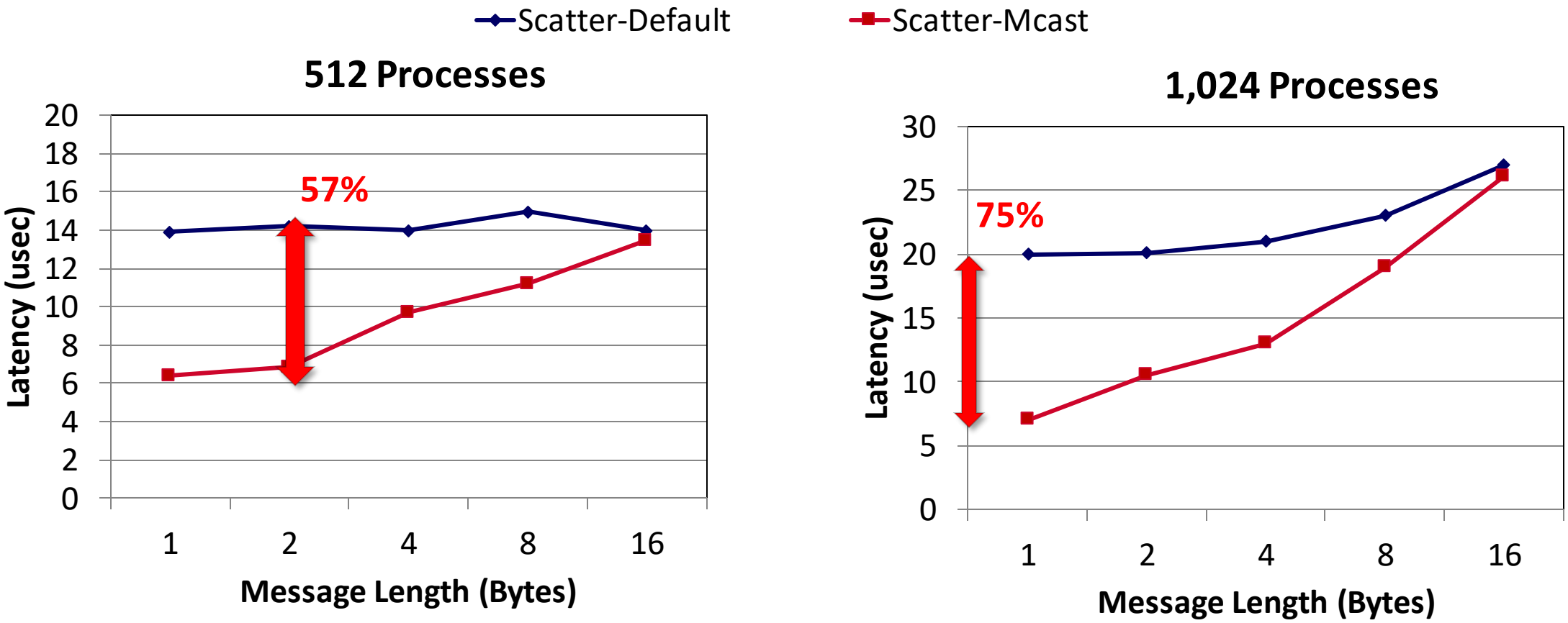






- MCAST-based designs improve latency of MPI_Bcast by up to **85%**
- Use MV2_USE_MCAST=1 to enable MCAST-based designs

MPI_Scatter - Benefits of using Hardware-Mcast



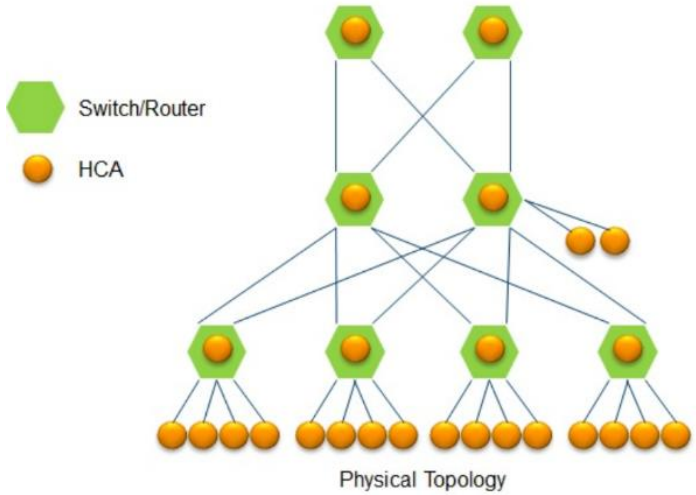
- Enabling MCAST-based designs for MPI_Scatter improves small message up to **75%**

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

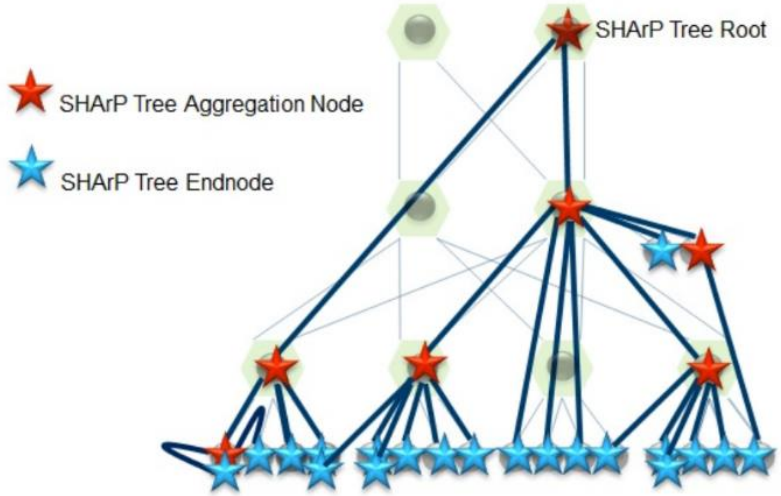
Offloading with Scalable Hierarchical Aggregation Protocol (SHArP)

- 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 *

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



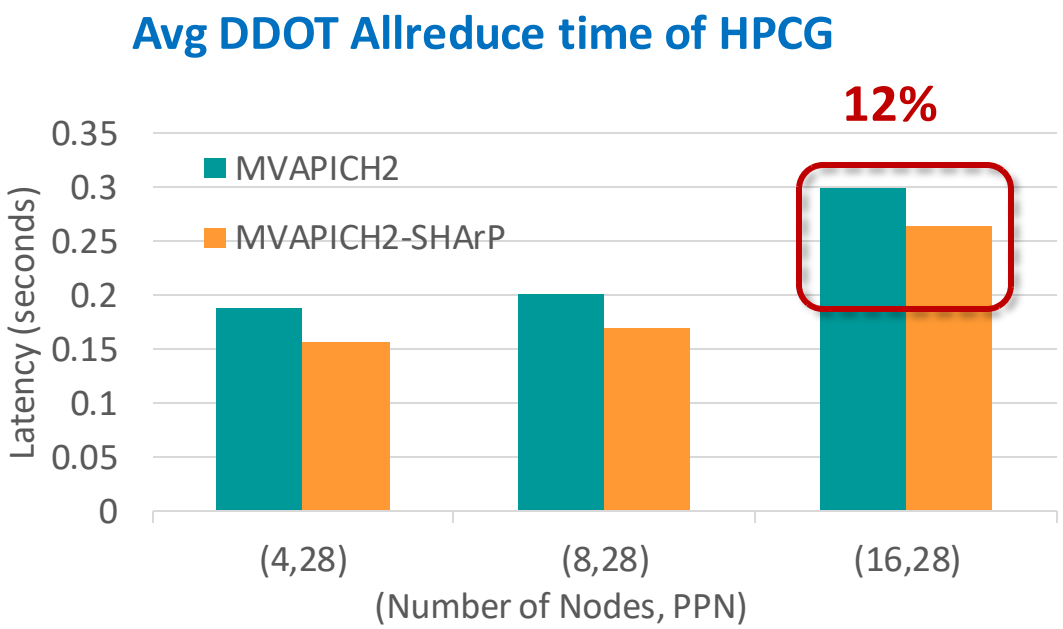
Physical Network Topology*



Logical SHArP Tree*

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

Benefits of SHARP Allreduce at Application Level



More details in the talk "Impact of SHARP and Adaptive Routing on Applications on Frontera" by John Cazes (TACC) on Wednesday (08/26/2020) from 12:00 PM - 12:30 PM EDT

SHARP support available since MVAPICH2 2.3a

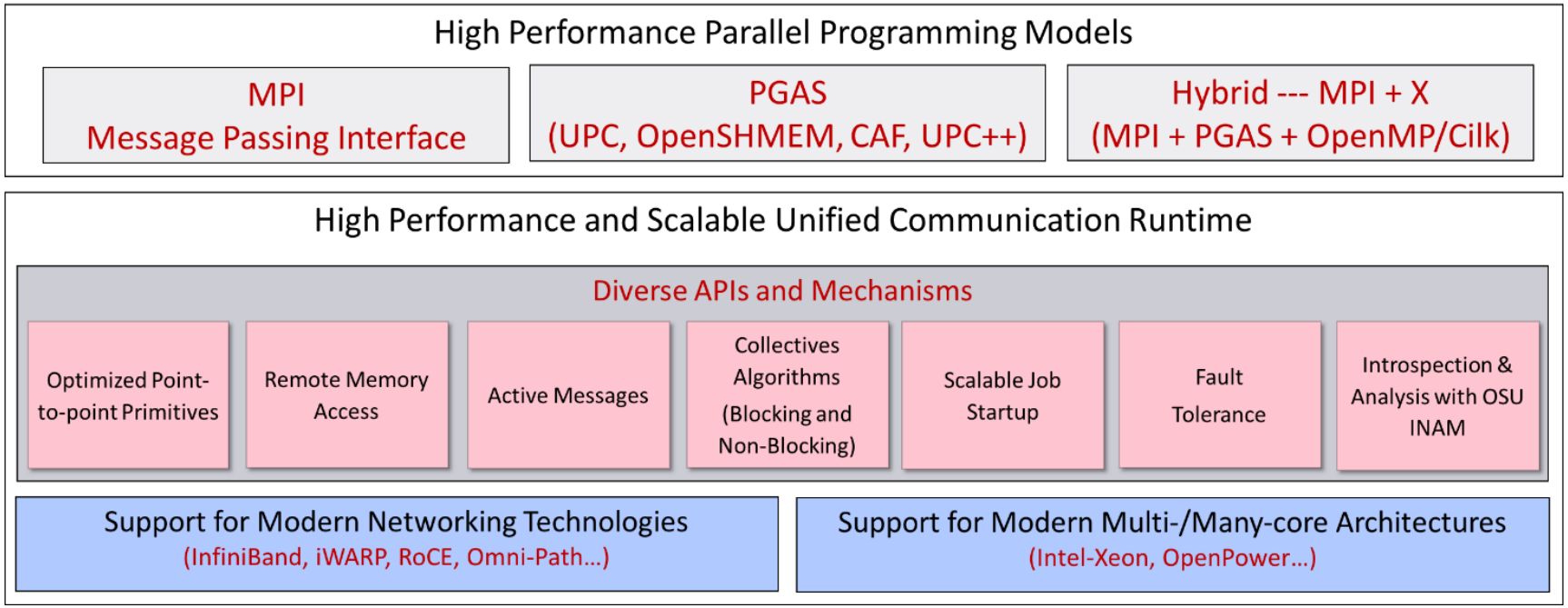
Parameter	Description	Default
MV2_ENABLE_SHARP=1	Enables SHARP-based collectives	Disabled
--enable-sharp	Configure flag to enable SHARP	Disabled

- 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

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 2.3

- **Released on 06/08/2020**
- Major Features and Enhancements
  - **MPI Features**
    - Based on MVAPICH2 2.3.4
      - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interfaces
    - Enhanced point-to-point and collective tunings for AMD EPYC, Catalyst@EPCC, Mayer@Sandia, Auzre@Microsoft, AWS, and Frontera@TACC
  - **MPI (Advanced) Features**
    - Optimized support for large message MPI_Allreduce and MPI_Reduce
      - OFA-IB-CH3 and OFA-IB-RoCE interfaces
    - Improved performance for communication using DC transport
      - OFA-IB-CH3 interface
    - Enhanced support for AWS EFA adapter and SRD transport protocol
      - OFA-IB-CH3 interface
    - Enhanced point-to-point and collective tuning for AWS EFA adapter and SRD transport protocol
      - OFA-IB-CH3 interface
    - Add multiple MPI_T PVARs and CVARs for point-to-point and collective operations
    - Tuning for MPI collective operations for Intel Broadwell, Intel CascadeLake, Azure HB (AMD EPYC), and Azure HC (Intel Skylake) systems
      - OFA-IB-CH3, OFA-IB-RoCE, PSM-CH3, and PSM2-CH3 interface
  - **Support for OSU InfiniBand Network Analysis and Management (OSU INAM) Tool v0.9.6**
  - **Unified Runtime Features**
    - Based on MVAPICH2 2.3.4 (OFA-IB-CH3 interface). All the runtime features enabled by default in OFA-IB-CH3 and OFA-IB-RoCE interface of MVAPICH2 2.3.4 are available in MVAPICH2-X 2.3

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*+				✓

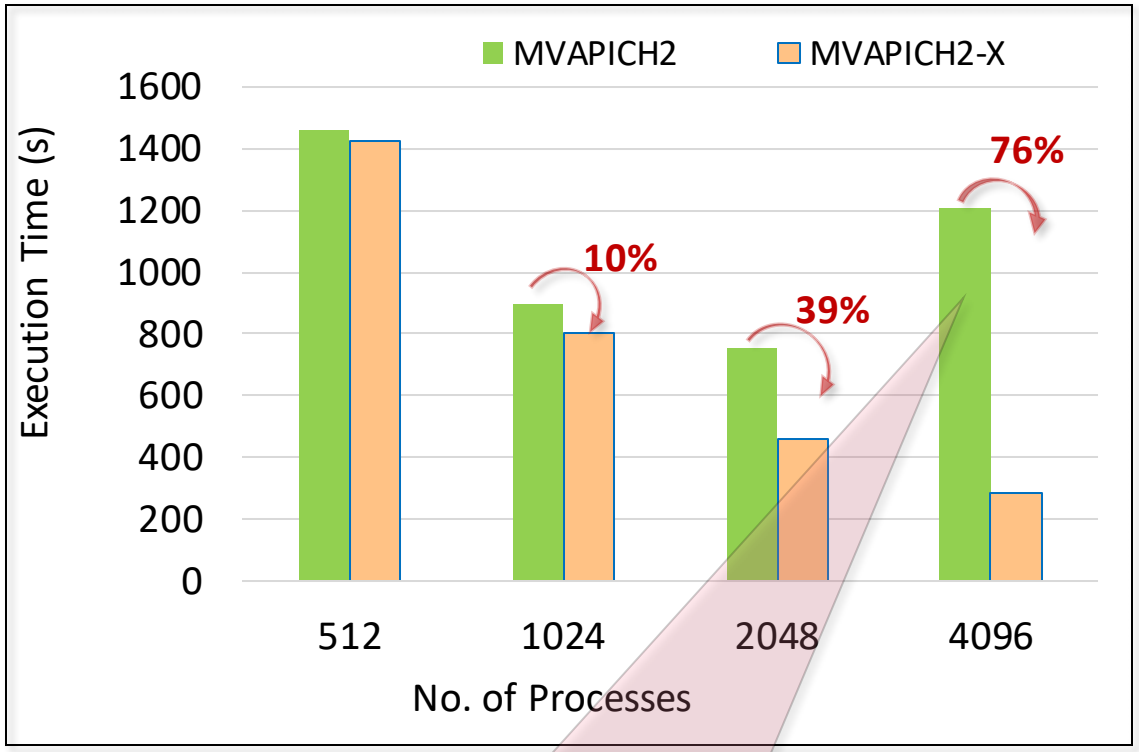
- *** indicates disabled by default at runtime. Must use appropriate environment variable in MVAPICH2-X user guide to enable it.**
- **+ indicates features only tested with InfiniBand network**

Overview of MVAPICH2-X Features

- Direct Connect (DC) Transport
  - Available from MVAPICH2-X 2.3rc1 onwards
- 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

Impact of DC Transport Protocol on Neuron

Neuron with YuEtAl2012



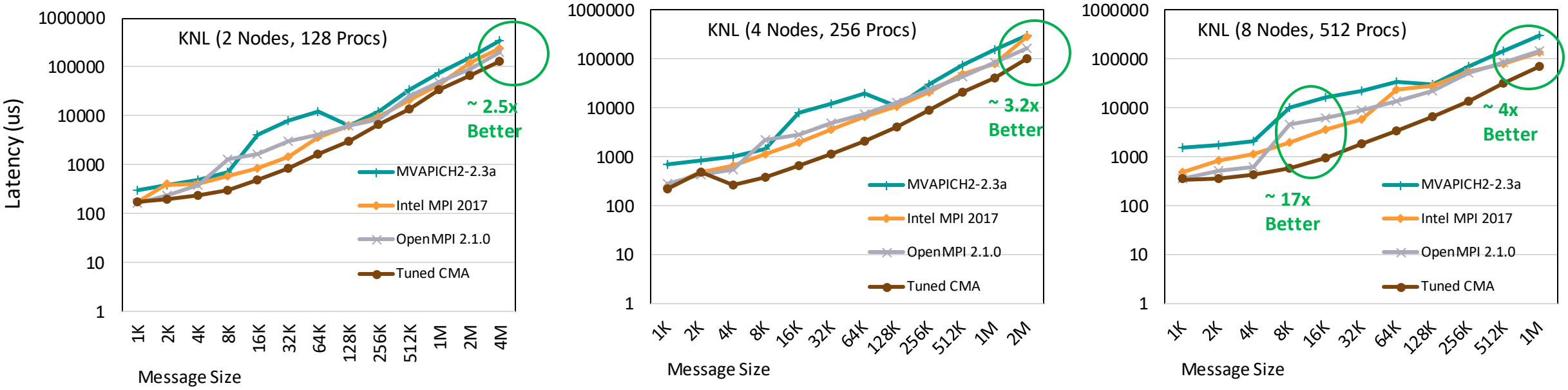
Overhead of RC protocol for connection establishment and communication

- 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

More details in talk "Building Brain Circuits: Experiences with shuffling terabytes of data over MPI" by Matthias Wolf, Blue Brain Project, EPFL, Switzerland on Wednesday (08/26/2020) from 1:30 PM - 2:00 PM EDT.

Optimized CMA-based Collectives for Large Messages



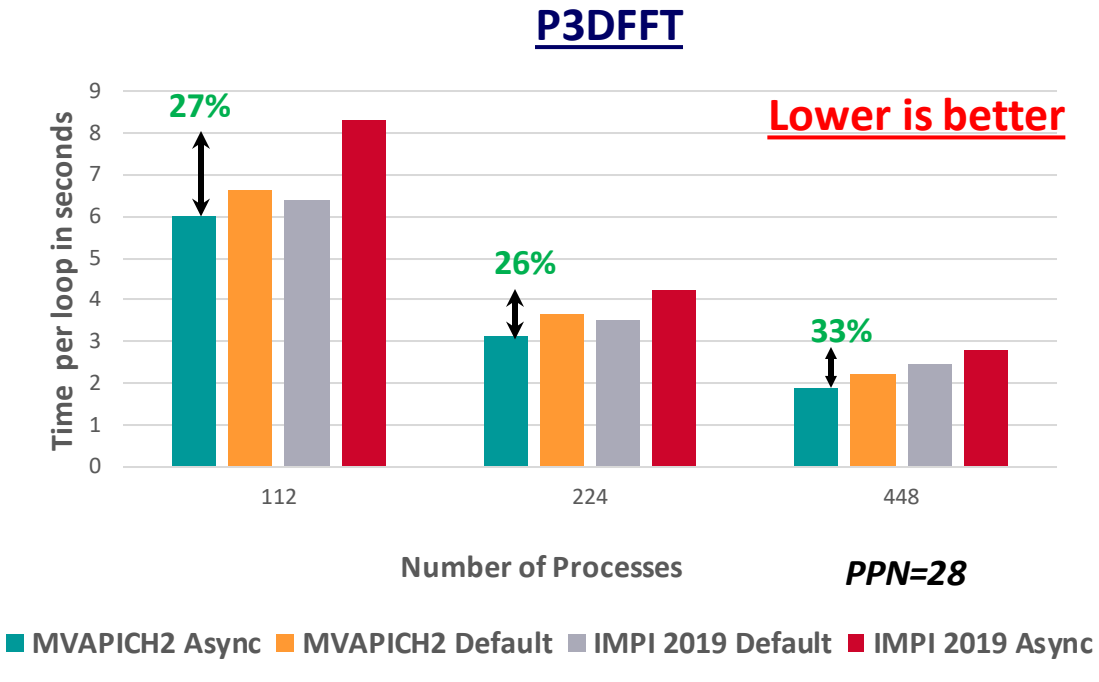
Performance of MPI_Gather on KNL nodes (64PPN)

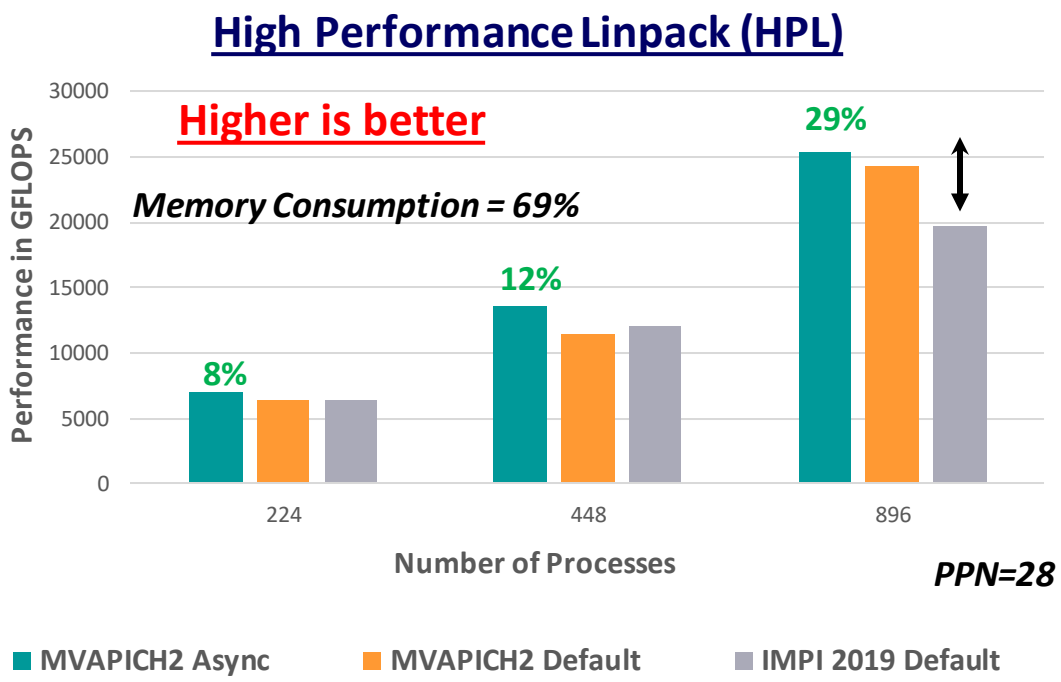
- Significant improvement over existing implementation for Scatter/Gather with 1MB messages (up to 4x on KNL, 2x on Broadwell, 14x on OpenPower)
- New two-level algorithms for better scalability
- Improved performance for other collectives (Bcast, Allgather, and Alltoall)

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

Benefits of the New Asynchronous Progress Design: Broadwell + InfiniBand





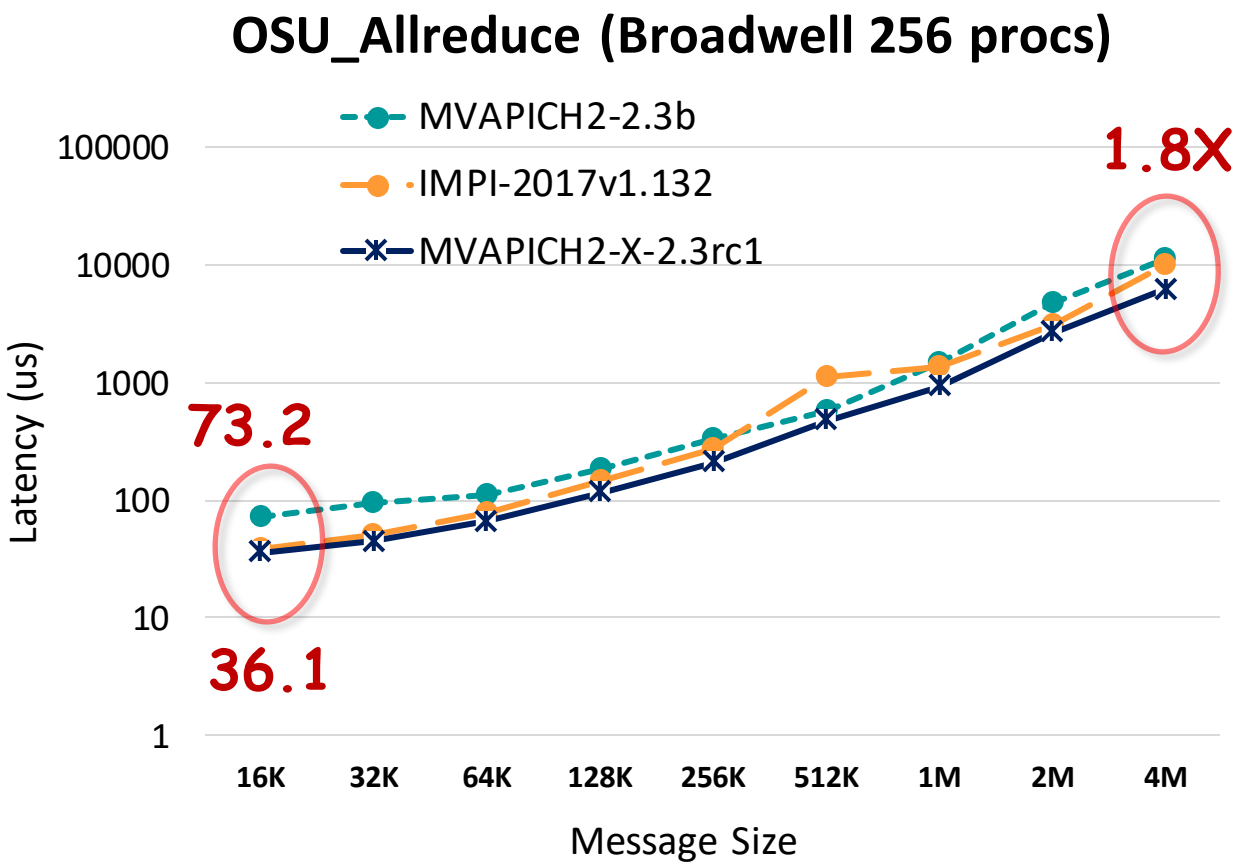
Up to 33% performance improvement in P3DFFT application with 448 processes

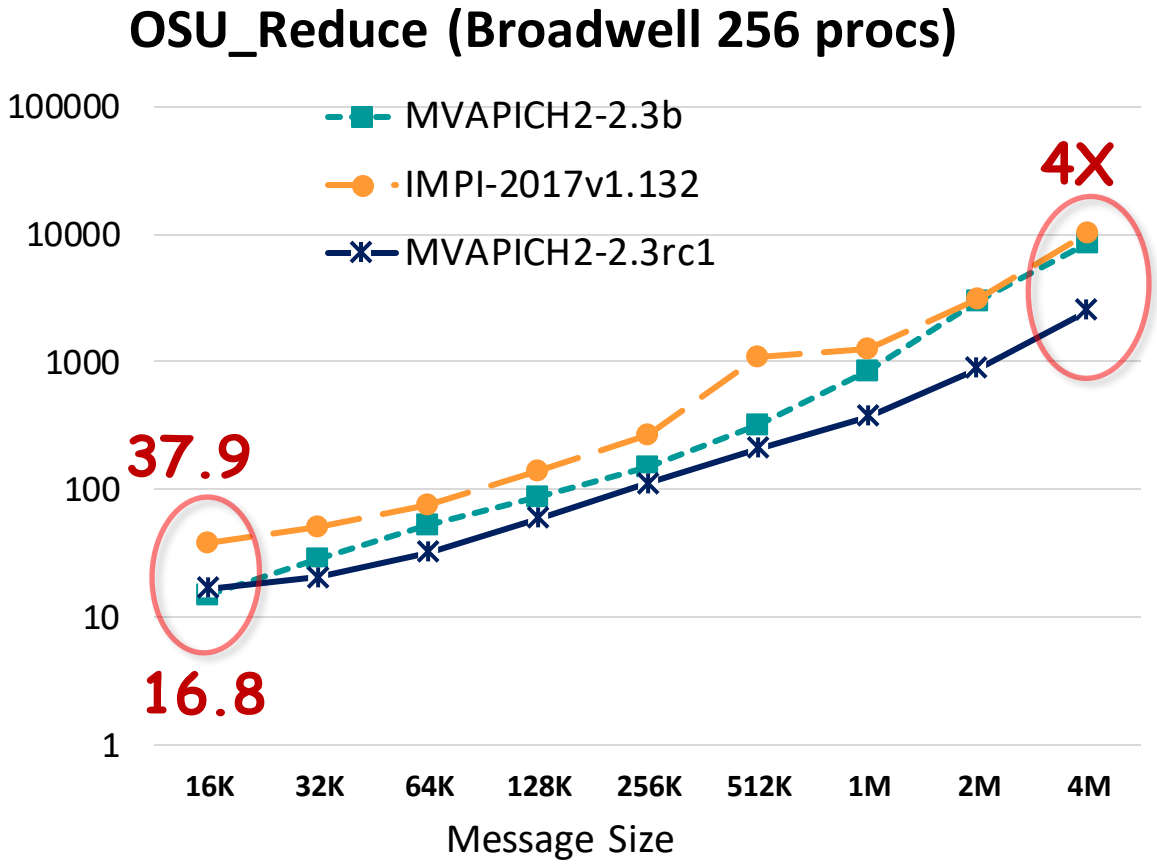
Up to 29% performance improvement in HPL application with 896 processes

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

Shared Address Space (XPMEM)-based Collectives Design



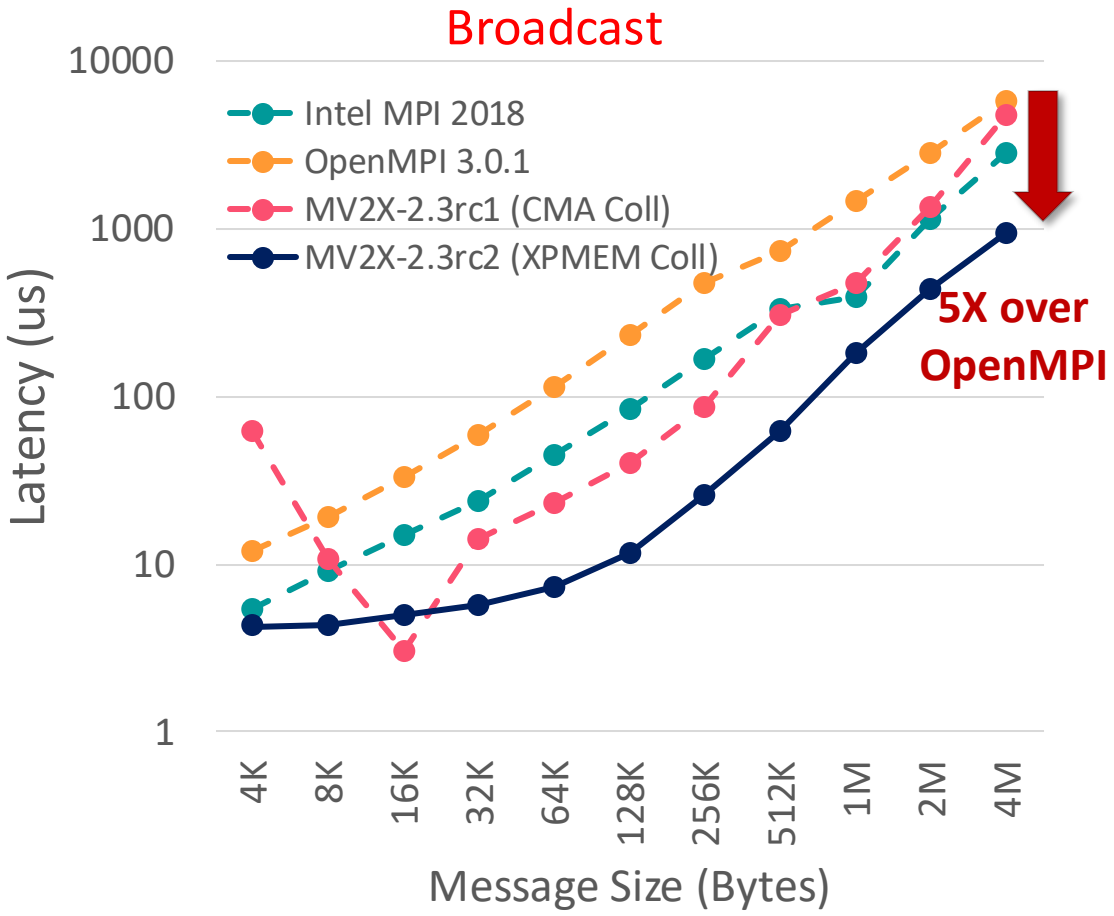


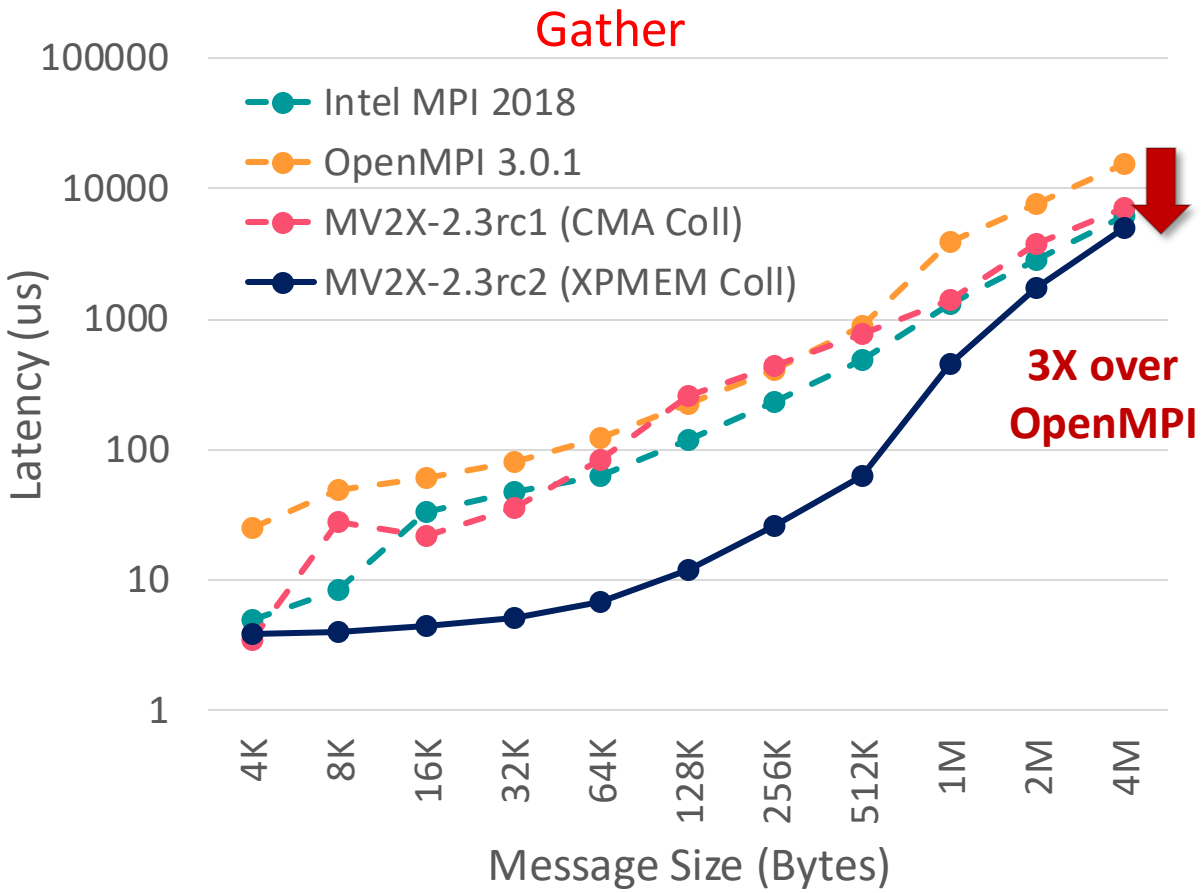
- "*Shared Address Space*"-based true *zero-copy* Reduction collective designs in MVAPICH2
- Offloaded computation/communication to peers ranks in reduction collective operation
- Up to **4X** improvement for 4MB Reduce and up to **1.8X** improvement for 4M AllReduce

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

Performance of Non-Reduction Collectives with XPMEM

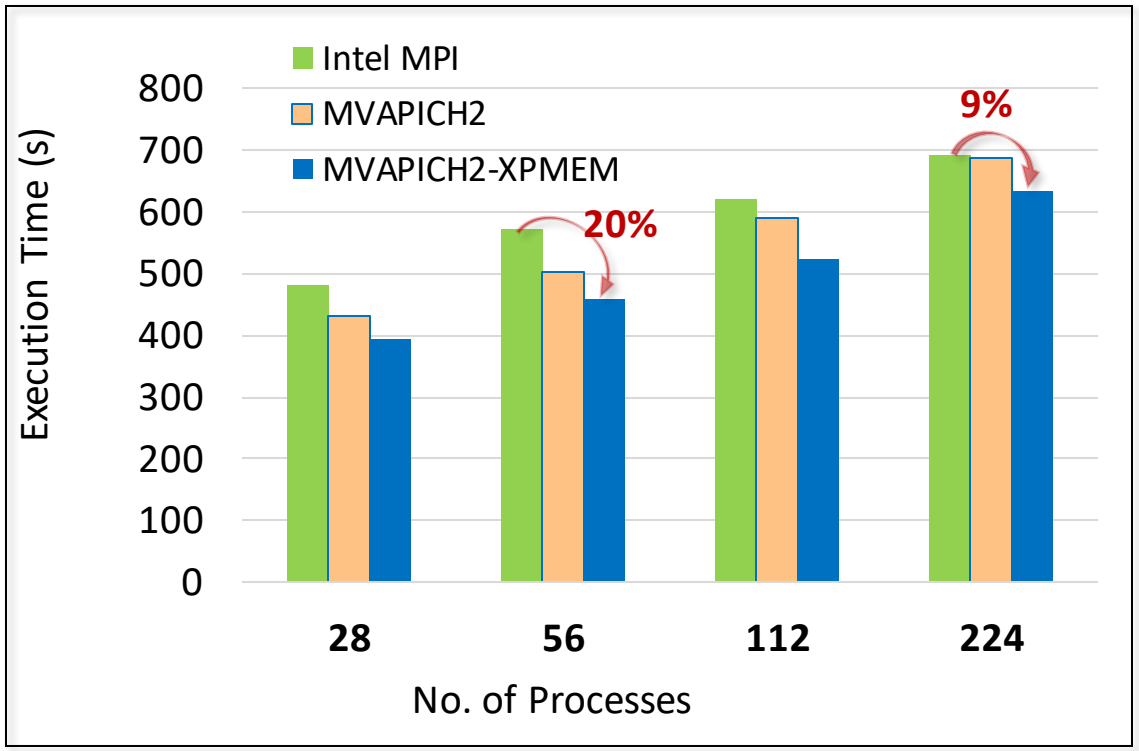




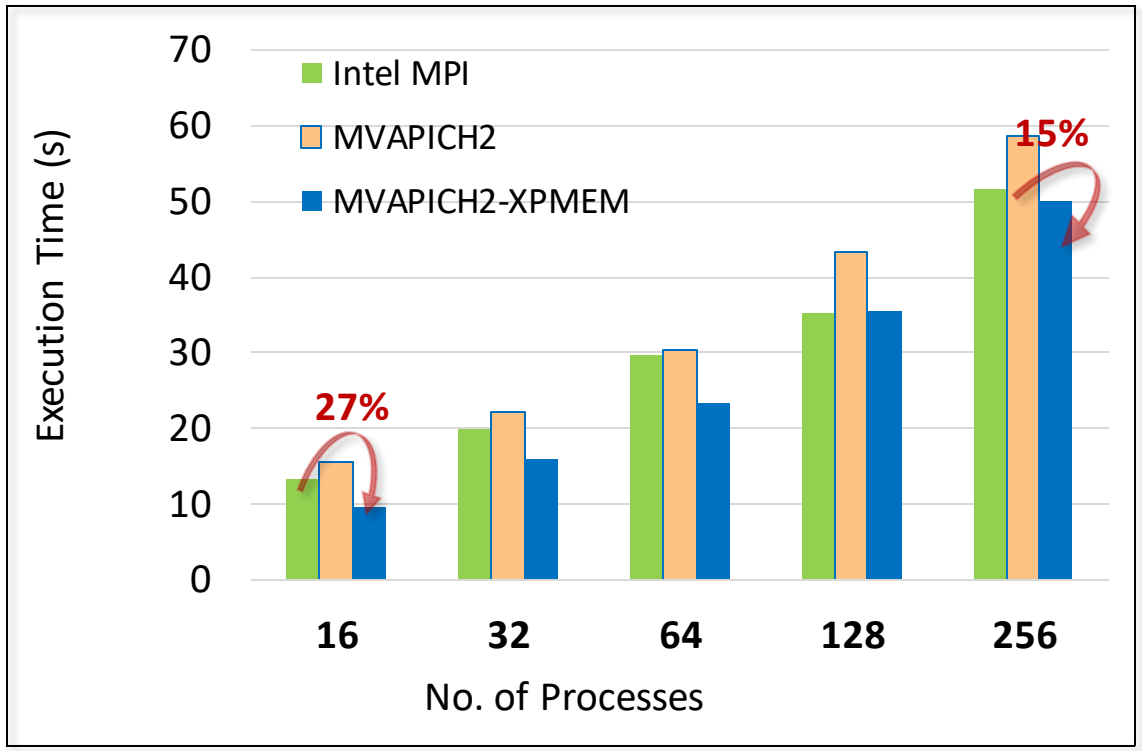
- **28 MPI Processes** on single dual-socket Broadwell E5-2680v4, 2x14 core processor
- Used osu_bcast from OSU Microbenchmarks v5.5

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

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

MPI + CUDA - Naive

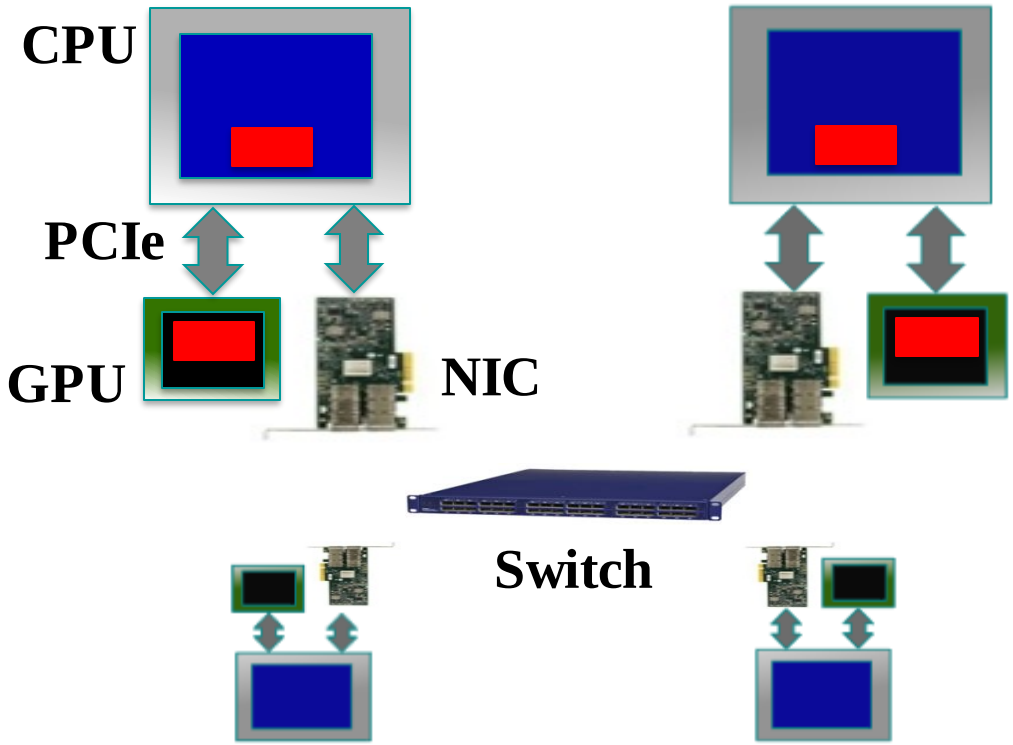
- Data movement in applications with standard MPI and CUDA interfaces

At Sender:

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

At Receiver:

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



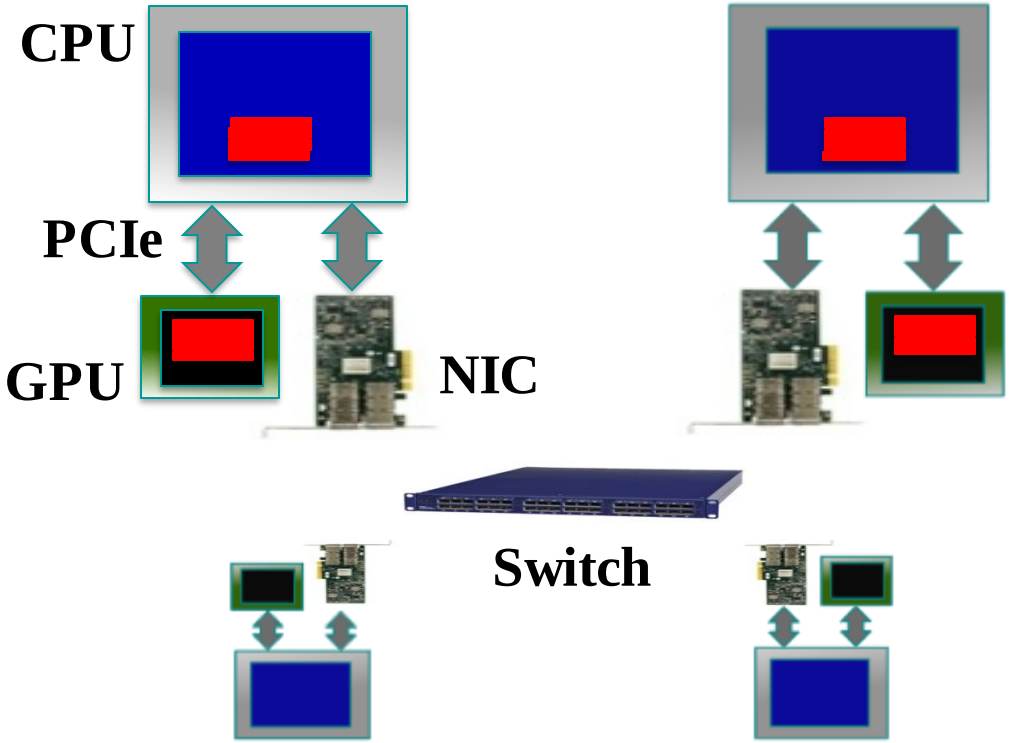
High Productivity and Low Performance

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 *
      blksz, …);
for (j = 0; j < pipeline_len; j++) {
       while (result != cudaSucess) {
            result = cudaStreamQuery(…);
             if(j > 0) MPI_Test(…);
       }
       MPI_Isend(s_hostbuf + j * block_sz, blksz . . .);
}
MPI_Waitall();

<<Similar at receiver>>
```



Low Productivity and High Performance

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

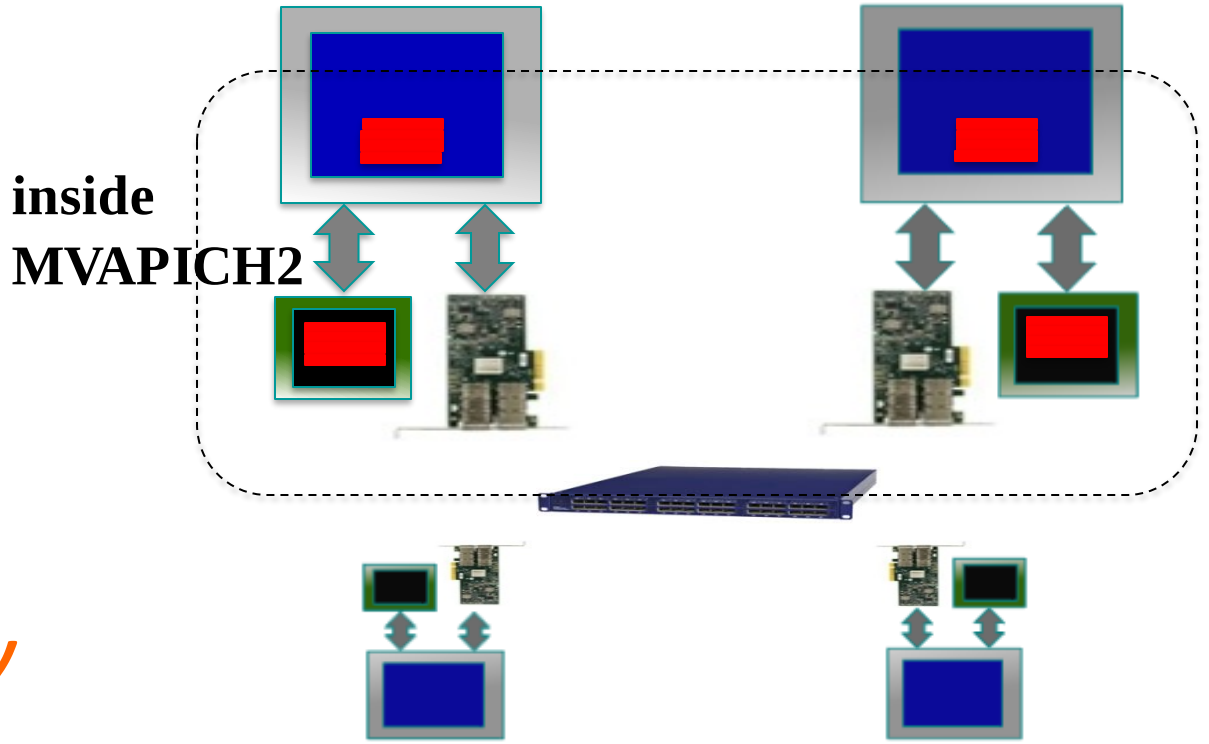
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

At Receiver:

```
MPI_Recv(r_devbuf, size, …);
```



High Performance and High Productivity

CUDA-Aware MPI: MVAPICH2-GDR 1.8-2.3.4 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 2.3.4 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 (== <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 2.3.4

- Released on 06/04/2020
- Major Features and Enhancements
  - Based on MVAPICH2 2.3.4
  - Enhanced MPI_Allreduce performance on DGX-2 systems
  - Enhanced MPI_Allreduce performance on POWER9 systems
  - Reduced the CUDA interception overhead for non-CUDA symbols
  - Enhanced performance for point-to-point and collective operations on Frontera's RTX nodes
  - Add new runtime variable 'MV2_SUPPORT_DL' to replace 'MV2_SUPPORT_TENSOR_FLOW'
  - Added compilation and runtime methods for checking CUDA support
  - Enhanced GDR output for runtime variable MV2_SHOW_ENV_INFO
  - Tested with Horovod and common DL Frameworks (TensorFlow, PyTorch, and MXNet)
  - Tested with PyTorch Distributed
  - Support for CUDA 10.1
  - Support for PGI 20.x
  - 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

Tuning GDRCOPY Designs in MVAPICH2-GDR

Parameter	Significance	Default	Notes
MV2_USE_GDRCOPY	• Enable / Disable GDRCOPY-based designs	1 (Enabled)	• Always enable
MV2_GDRCOPY_LIMIT	• Controls messages size until which GDRCOPY is used	8 KByte	• Tune for your system • 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 • 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**

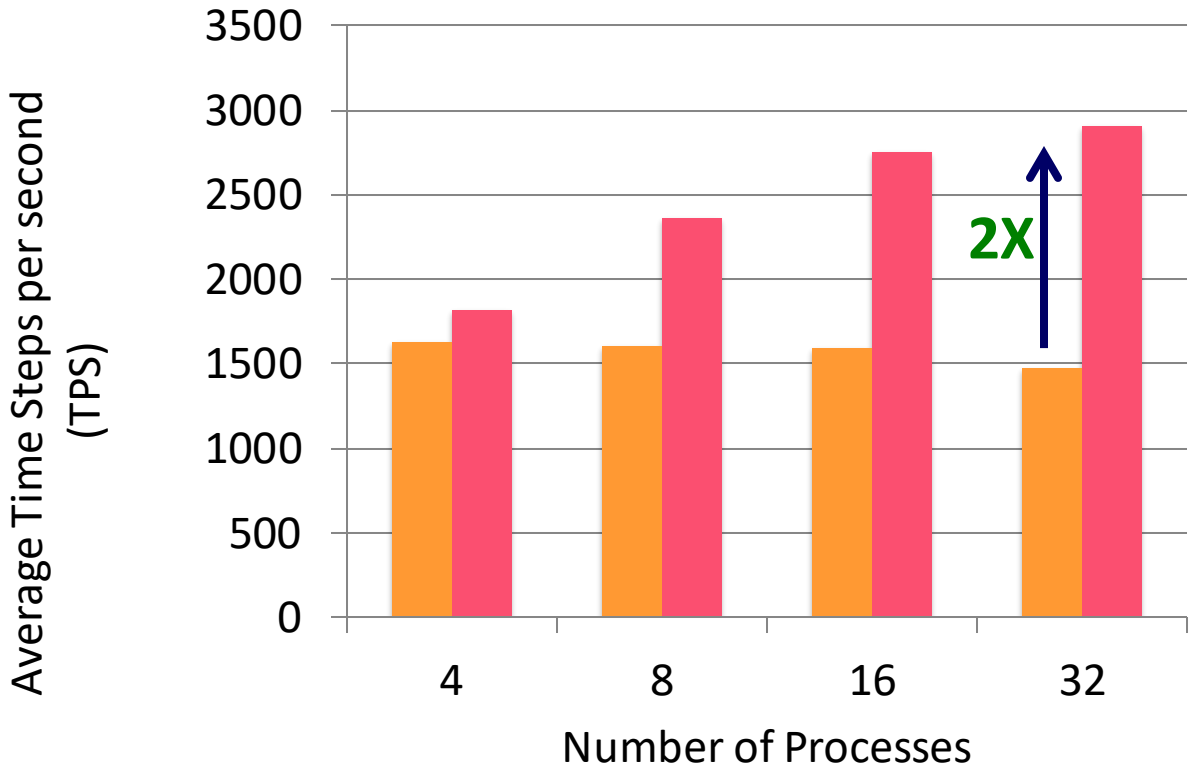
Tuning GPUDirect RDMA (GDR) Designs in MVAPICH2-GDR

Parameter	Significance	Default	Notes
MV2_USE_GPUDIRECT	• Enable / Disable GDR-based designs	1 (Enabled)	• Always enable
MV2_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
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 • GPU type, HCA type and configuration

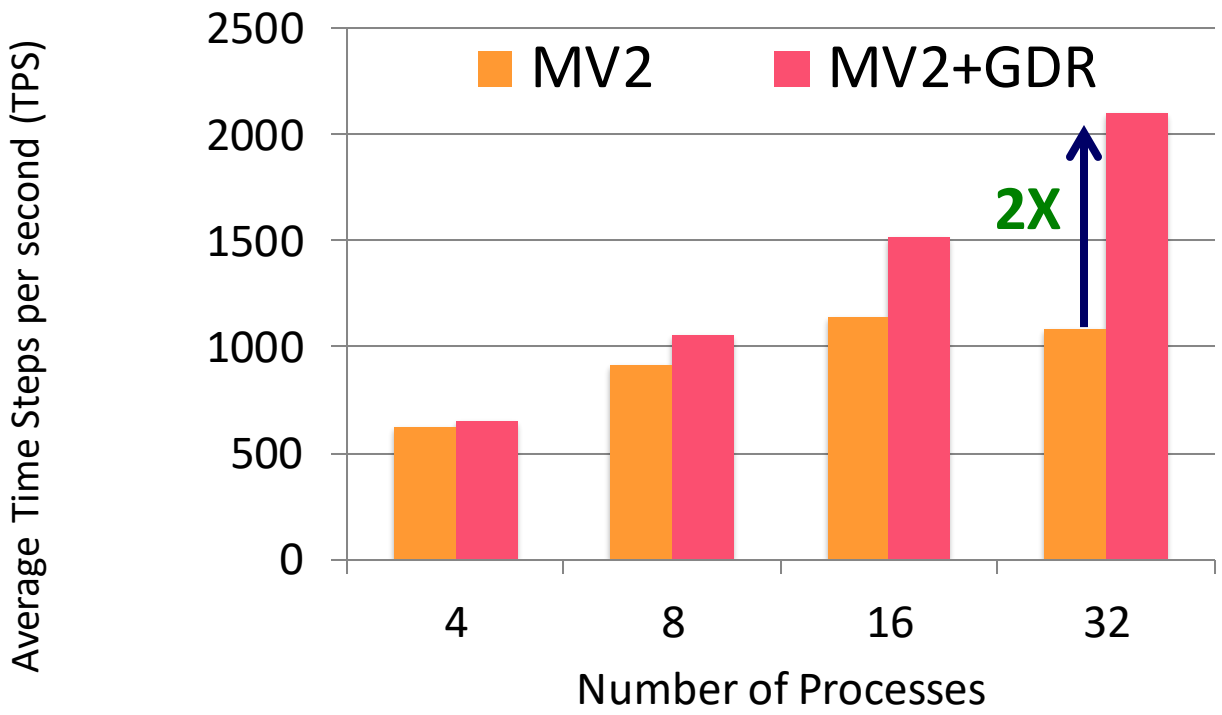
- **Refer to Tuning and Usage Parameters section of MVAPICH2-GDR user guide for more information**
- **http://mvapich.cse.ohio-state.edu/userguide/gdr/#_tuning_and_usage_parameters**

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

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.

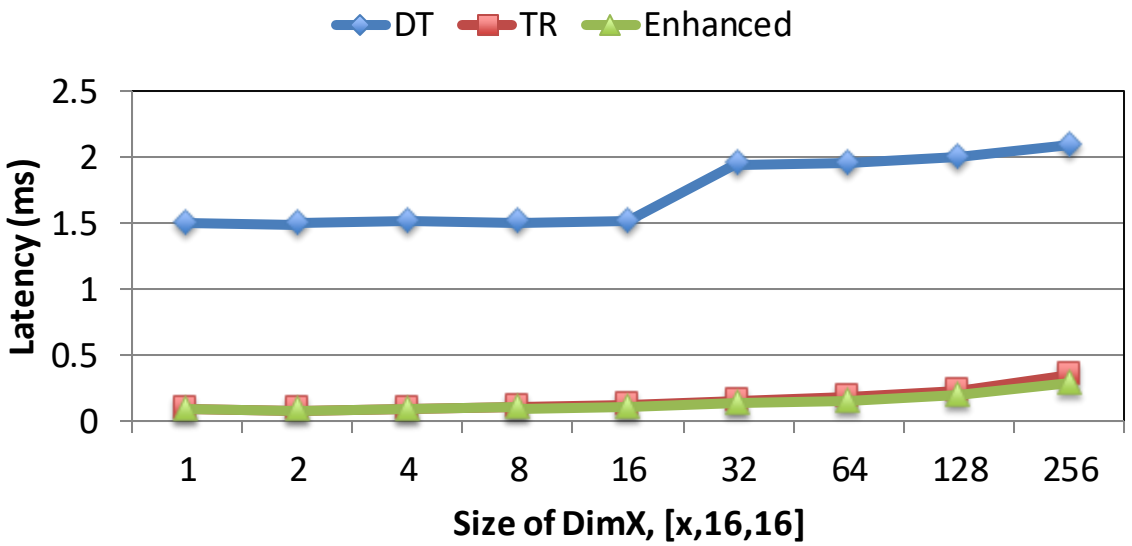
MPI Datatype Processing (Computation Optimization)

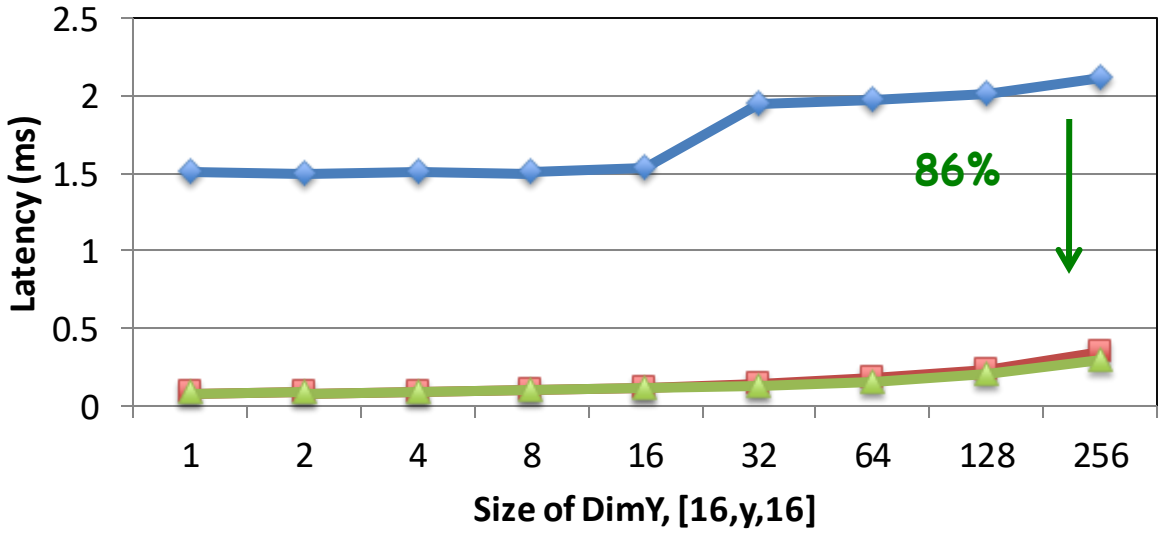
- Comprehensive support
  - Targeted kernels for regular datatypes - vector, subarray, indexed_block
  - Generic kernels for all other irregular datatypes
- Separate non-blocking stream for kernels launched by MPI library
  - Avoids stream conflicts with application kernels
- Flexible set of parameters 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

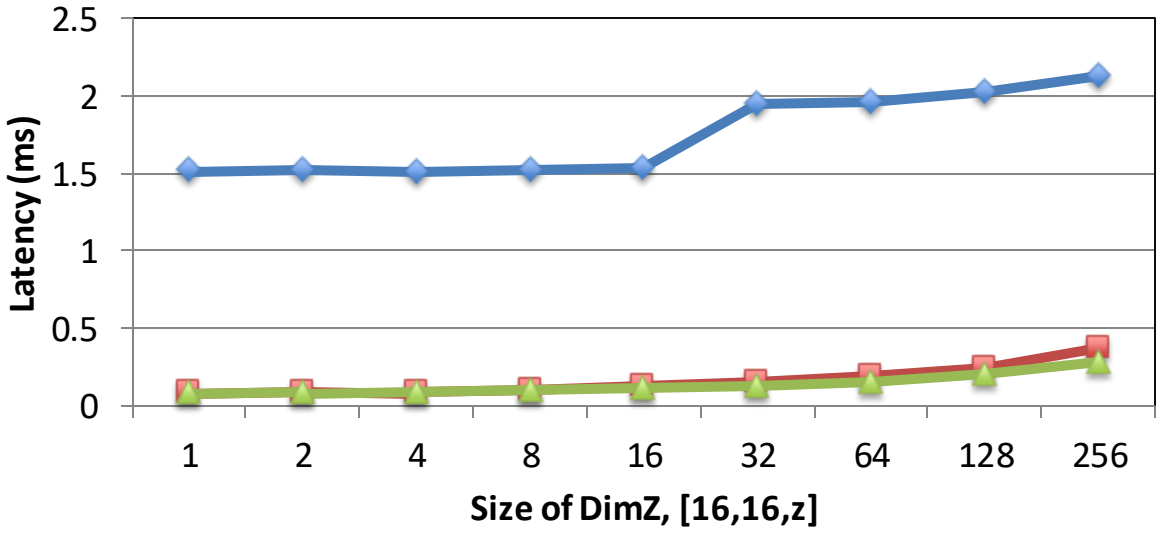
Performance of Stencil3D (3D subarray)

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







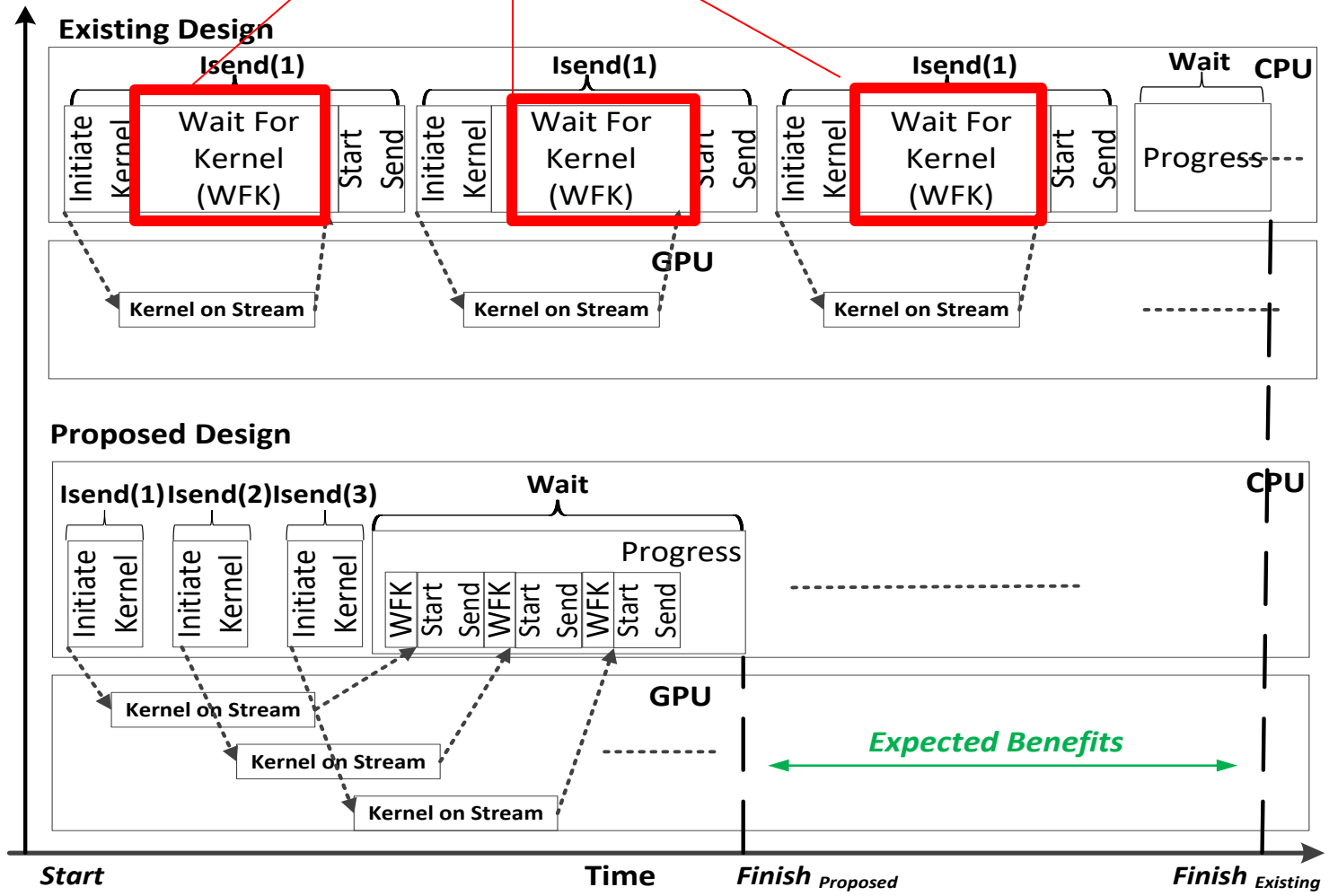
MPI Datatype Processing (Communication Optimization)

Common Scenario

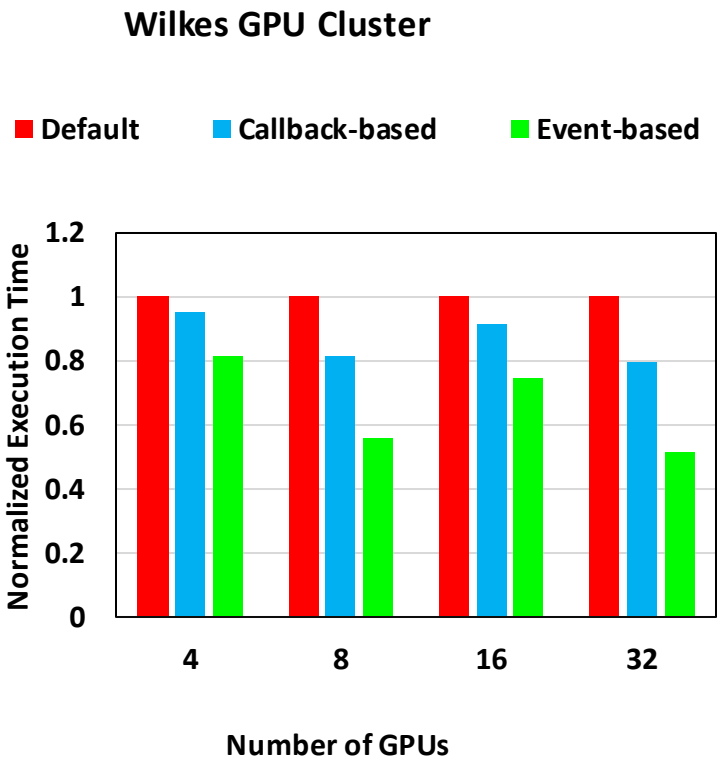
```
MPI_Isend (A,.. Datatype,…)
MPI_Isend (B,.. Datatype,…)
MPI_Isend (C,.. Datatype,…)
MPI_Isend (D,.. Datatype,…)
…

MPI_Waitall (…);
```

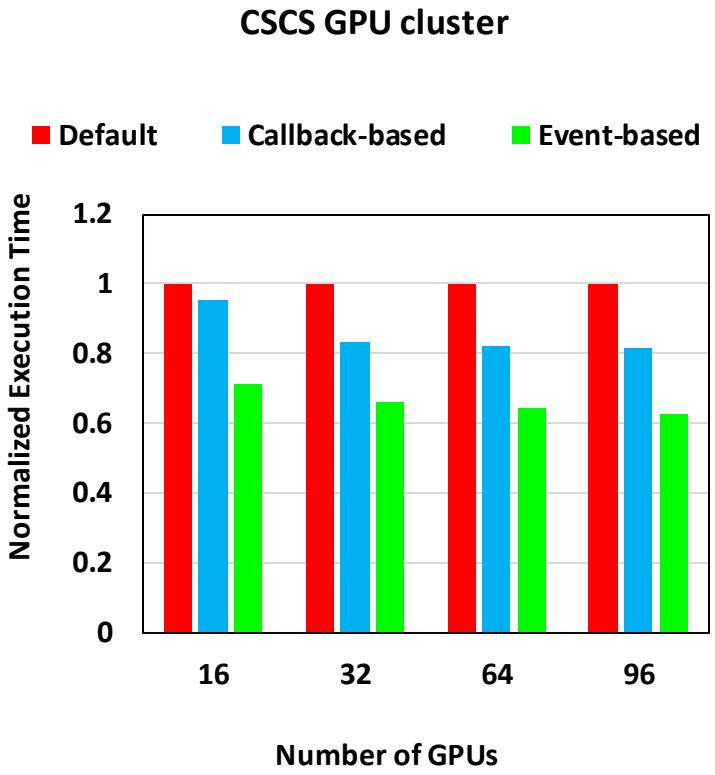
*A, B…contain non-contiguous MPI Datatype

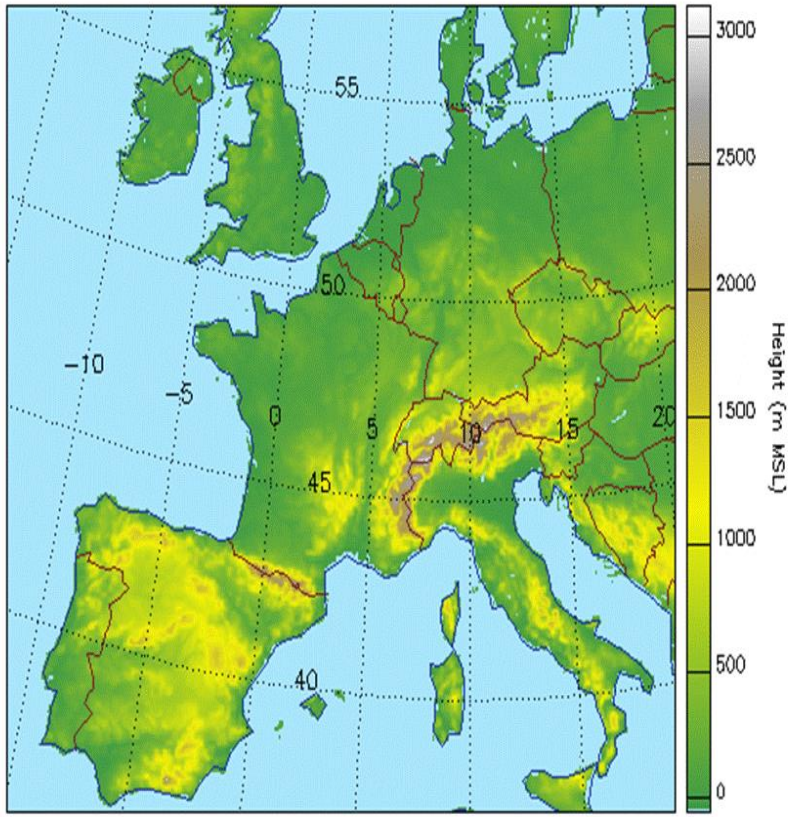
Waste of computing resources on CPU and GPU



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







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

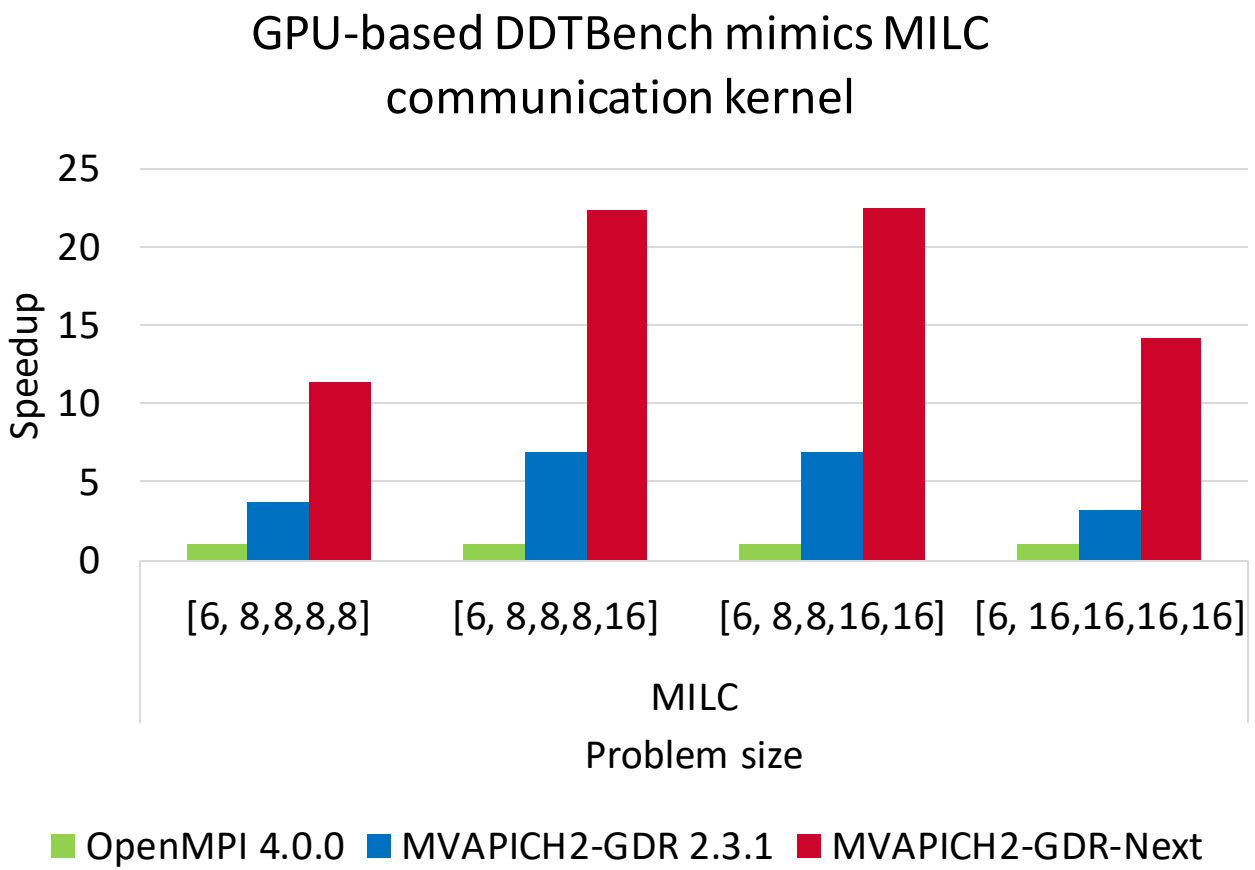
Cosmo model: http://www2.cosmo-model.org/content/tasks/operational/meteoSwiss/

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

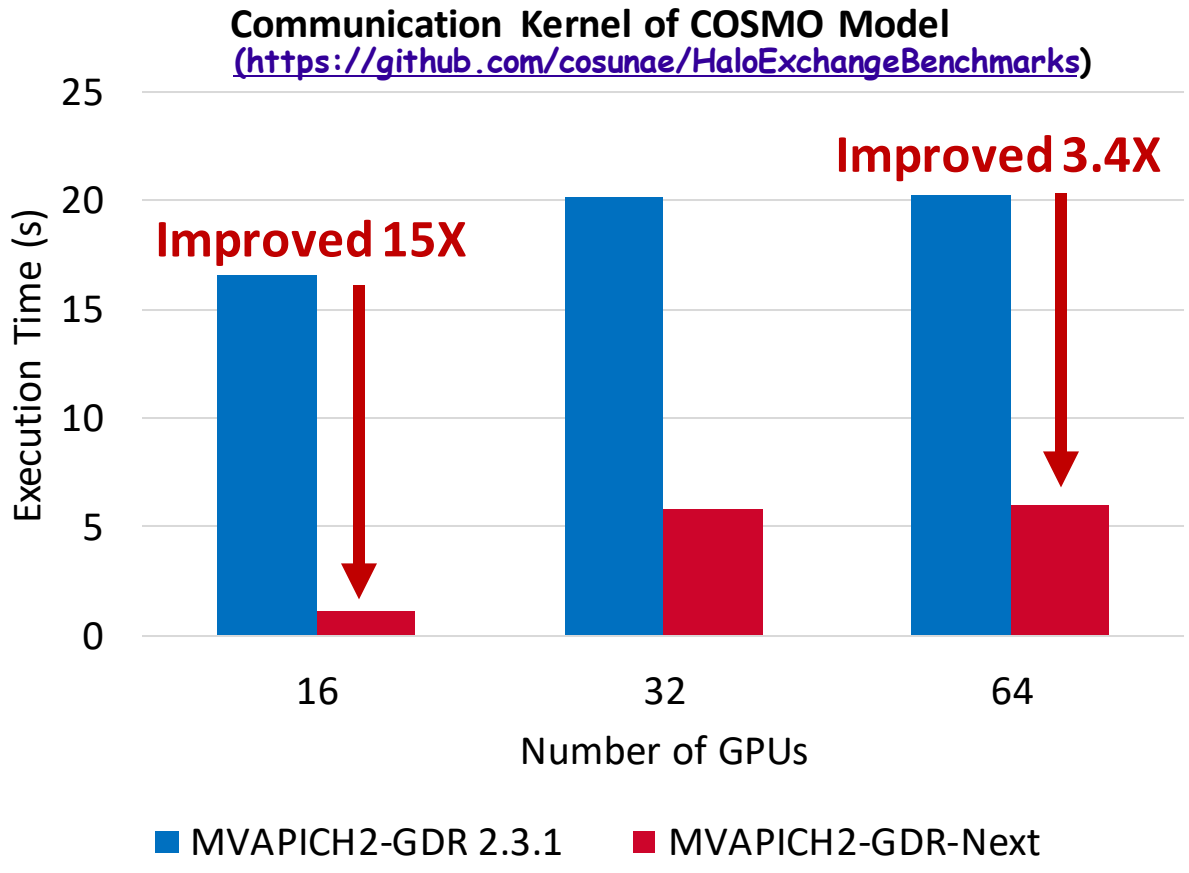
MVAPICH2-GDR: Enhanced Derived Datatype

- **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**



Platform: Nvidia DGX-2 system

(NVIDIA Volta GPUs connected with NVSwitch), CUDA 9.2

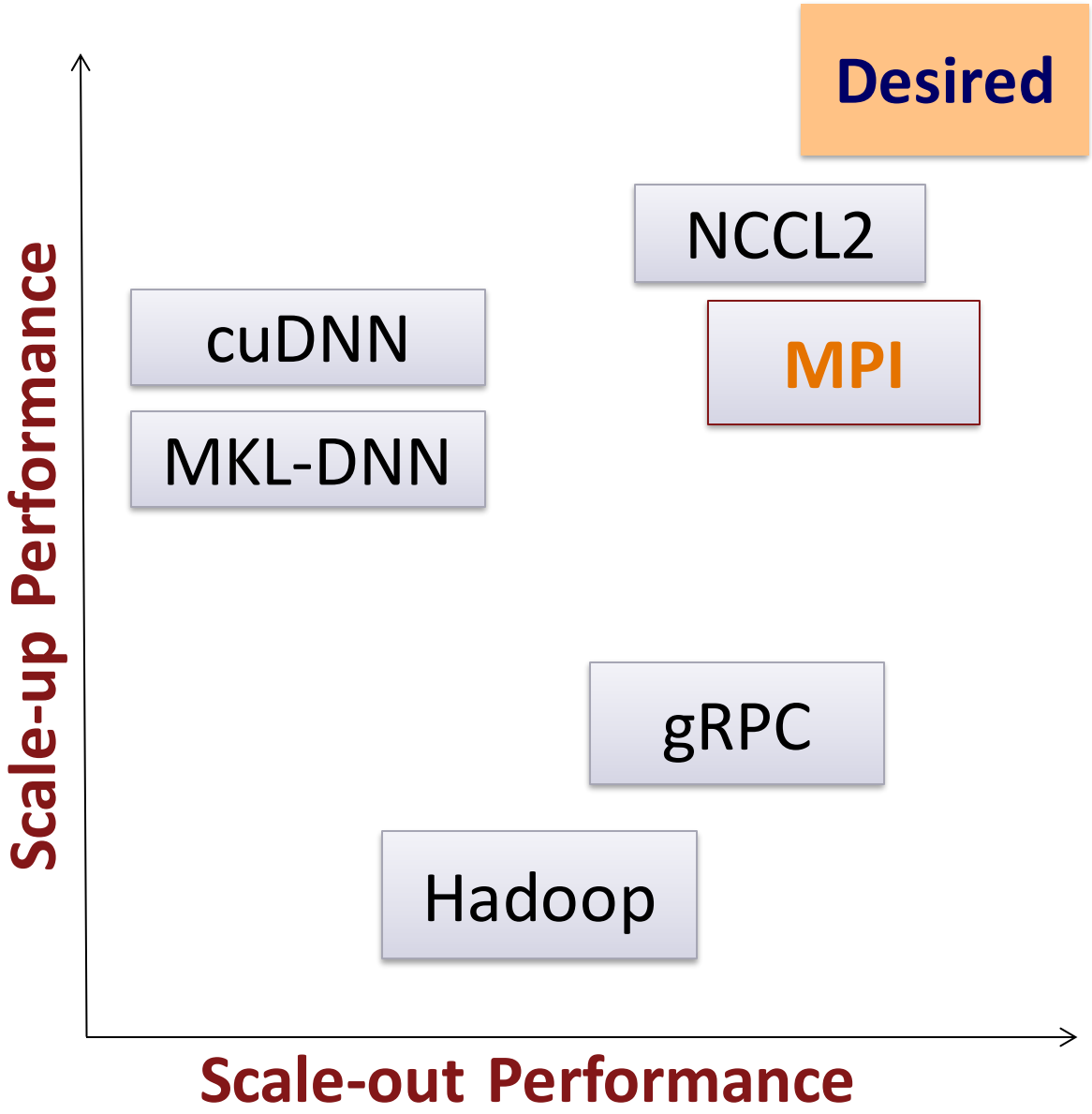


Platform: Cray CS-Storm

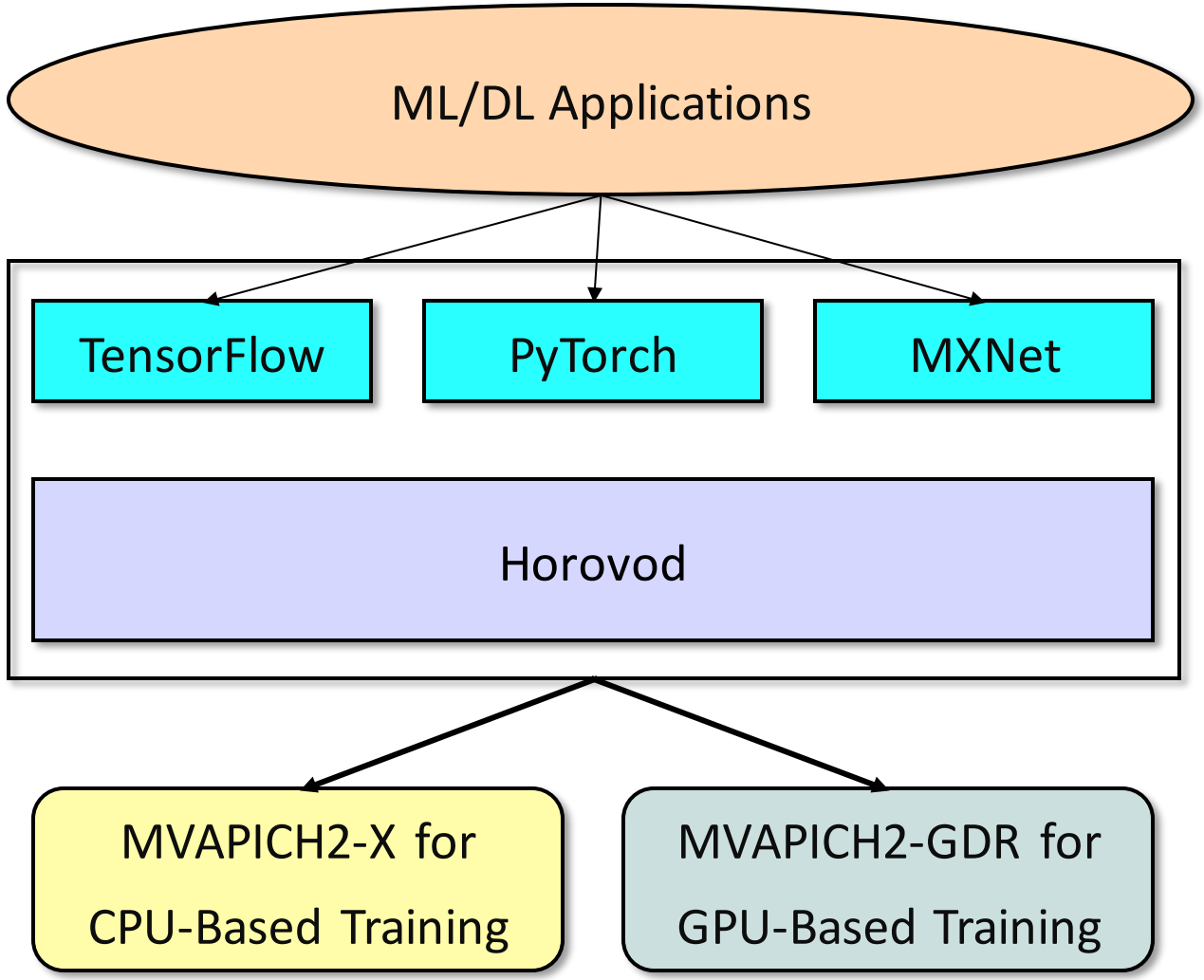
(16 NVIDIA Tesla K80 GPUs per node), CUDA 8.0

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



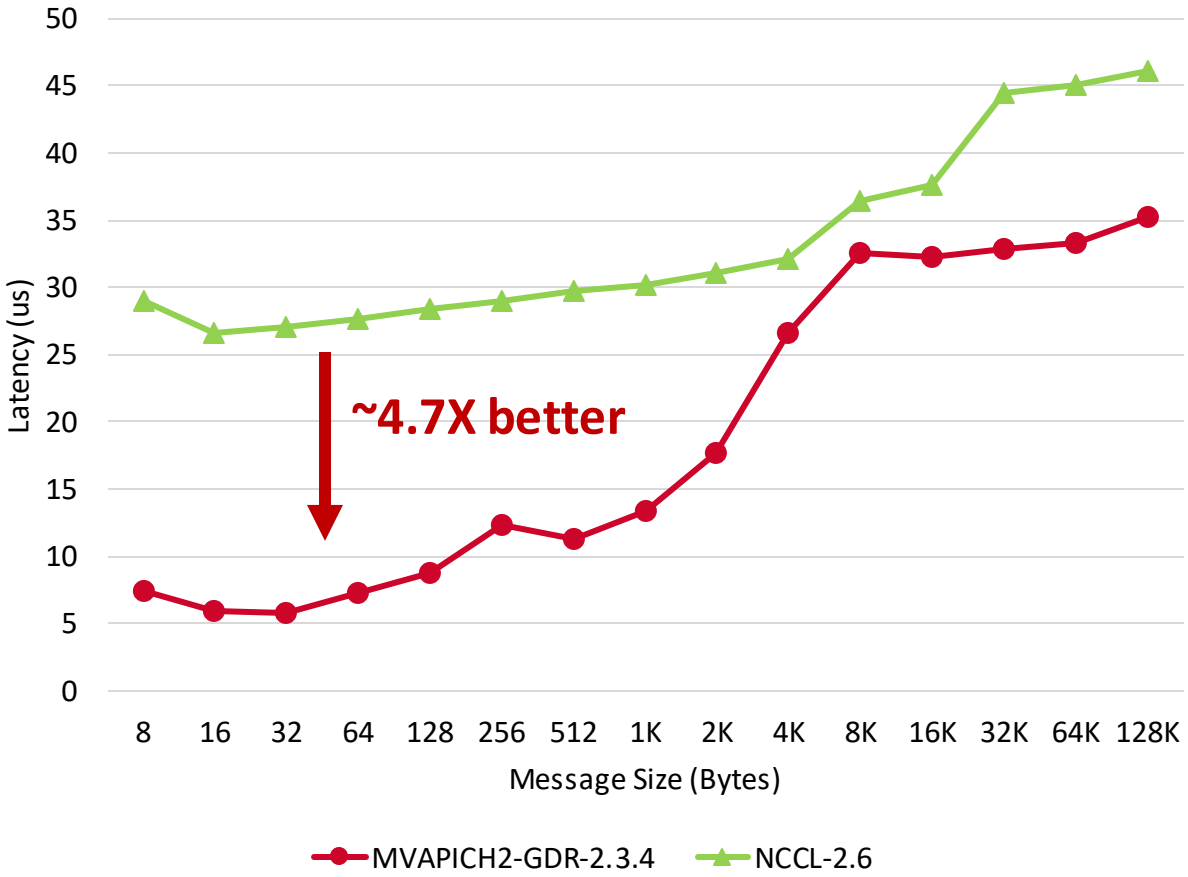
MVAPICH2 (MPI)-driven Infrastructure for ML/DL Training

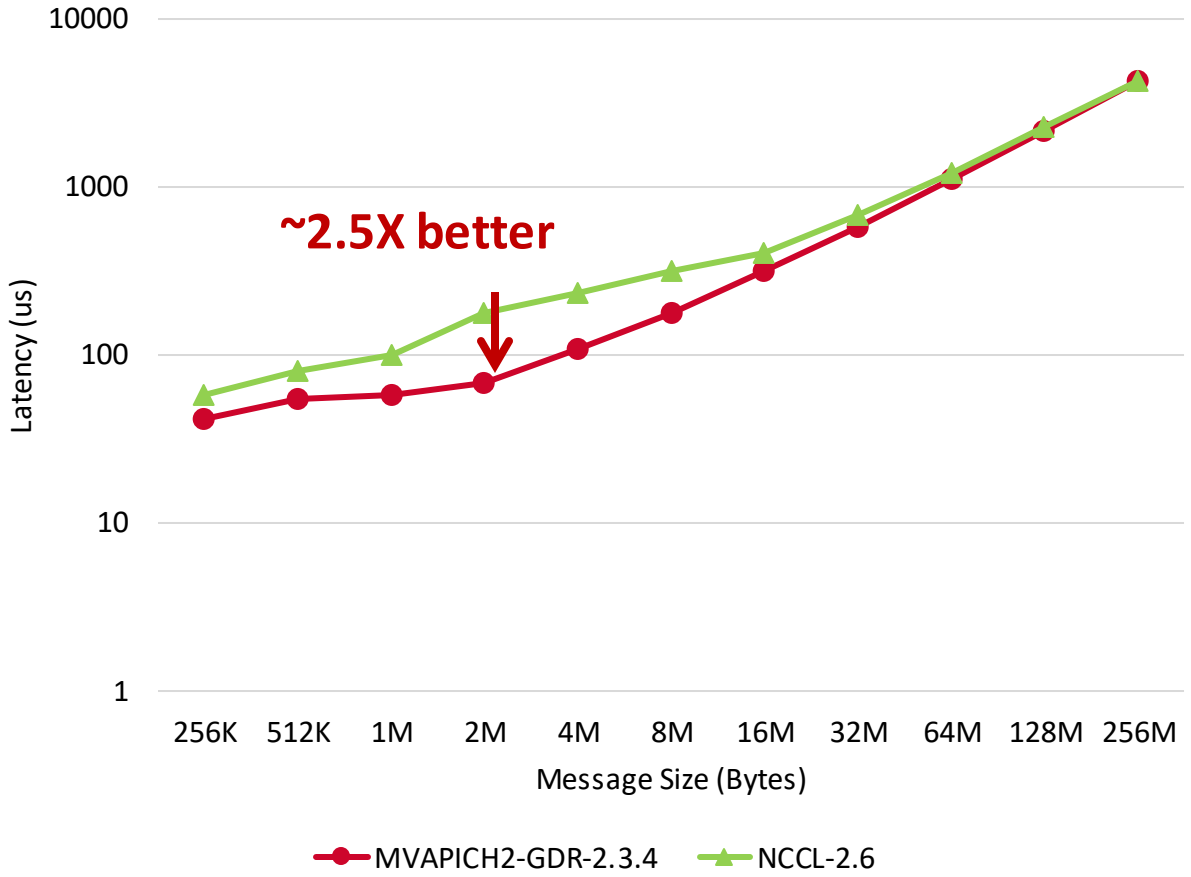


MVAPICH2-GDR vs. NCCL2 – Allreduce Operation (DGX-2)

- Optimized designs in 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 10.1



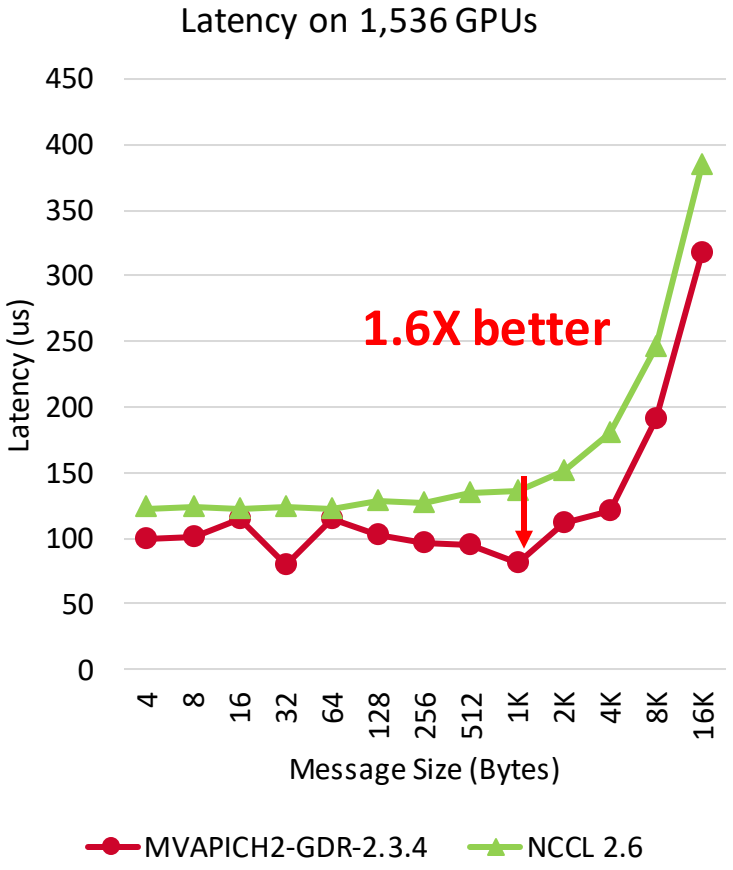


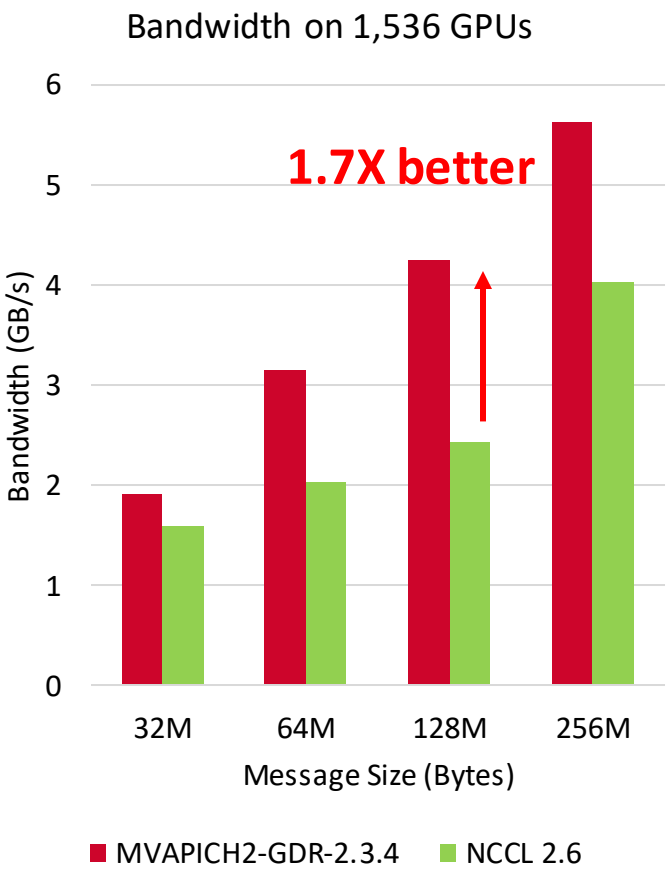
C.-H. Chu, P. Kousha, A. Awan, K. S. Khorassani, H. Subramoni and D. K. Panda, "NV-Group: Link-Efficient Reductions for Distributed Deep Learning on Modern Dense GPU Systems, " ICS-2020, June-July 2020.

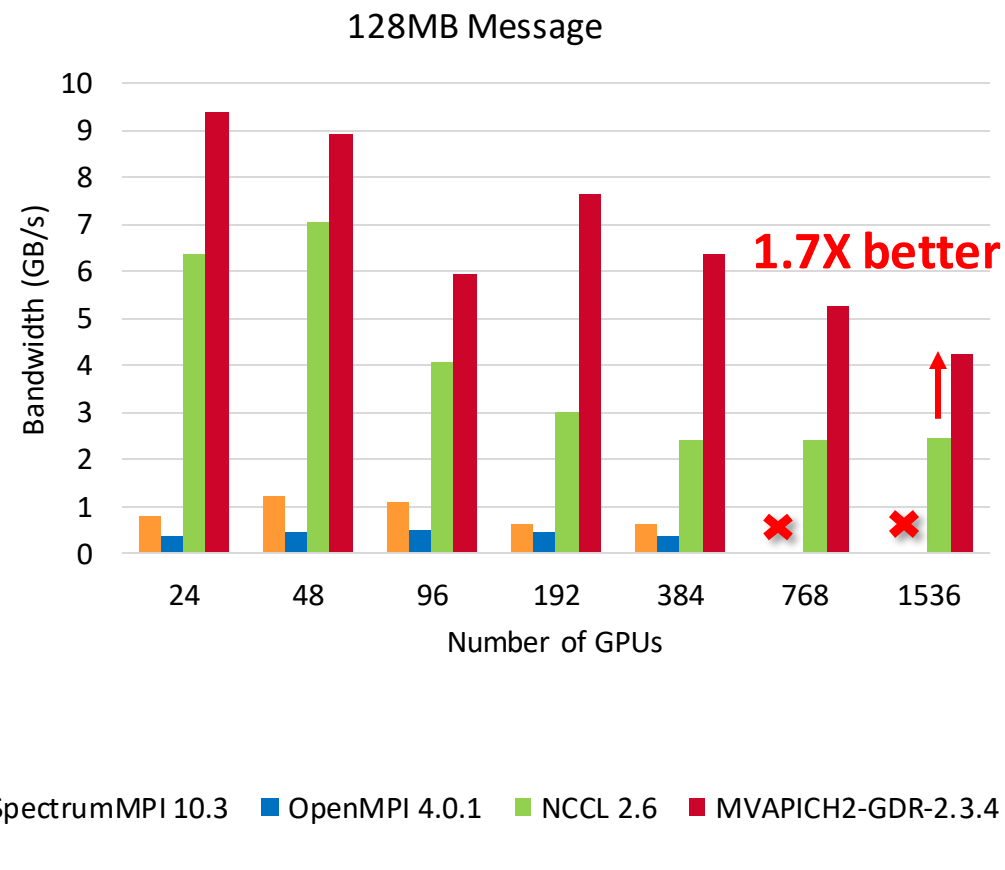
MVAPICH2-GDR: MPI_Allreduce at Scale (ORNL Summit)

- Optimized designs in 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



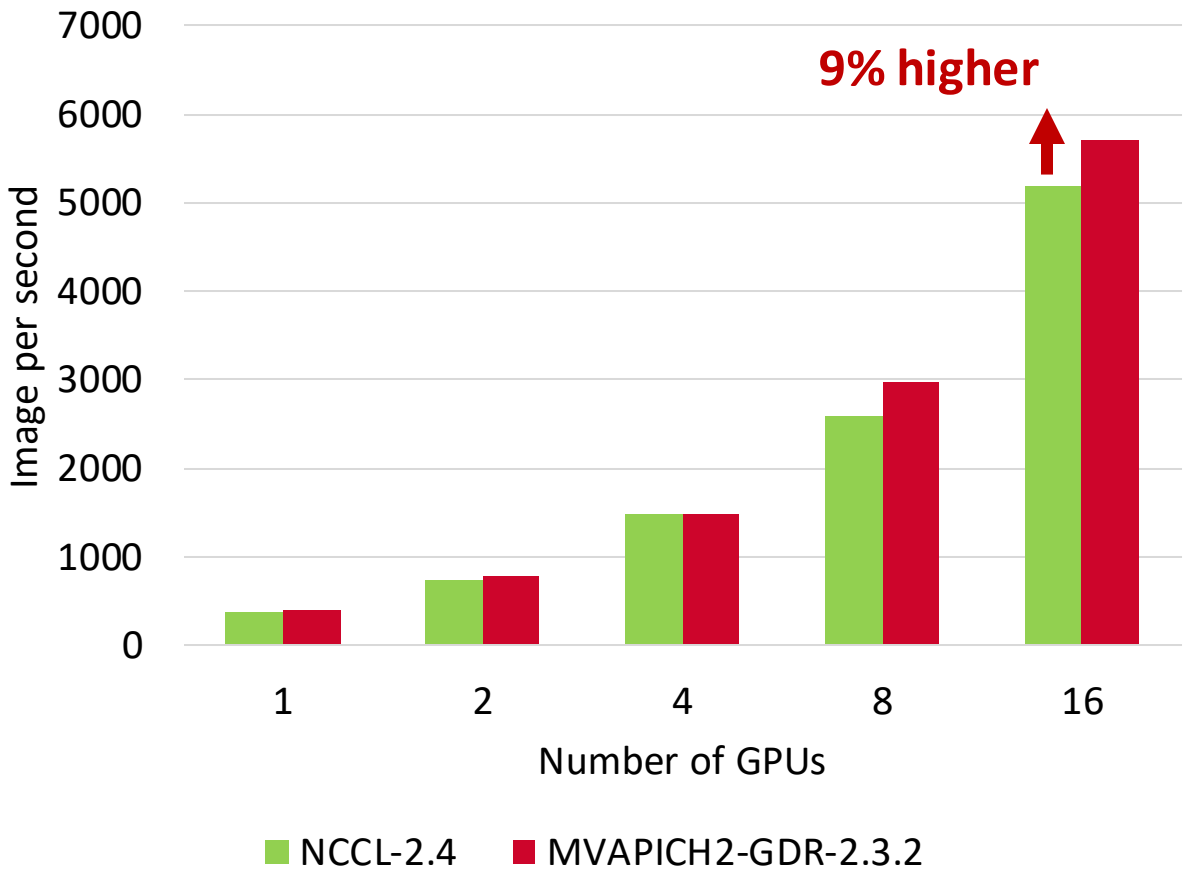




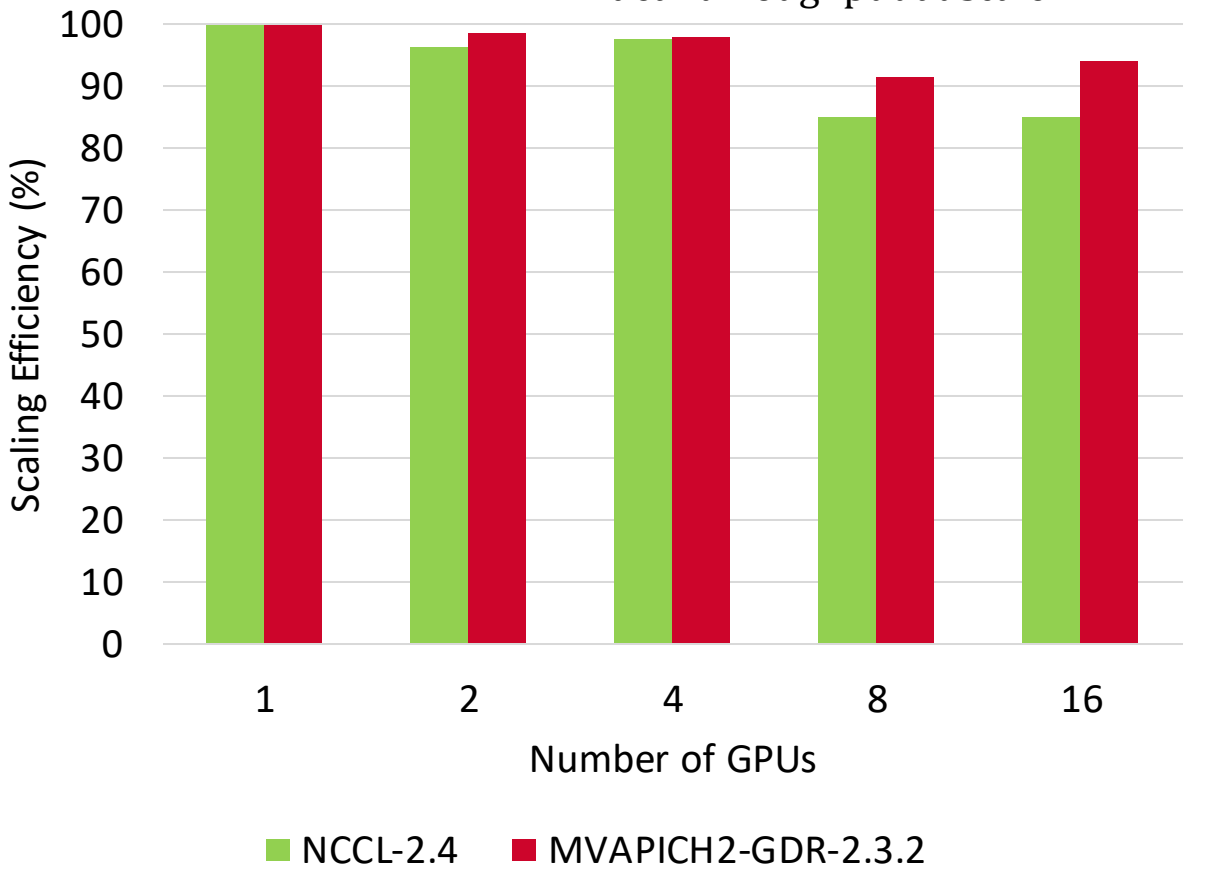
C.-H. Chu, P. Kousha, A. Awan, K. S. Khorassani, H. Subramoni and D. K. Panda, "NV-Group: Link-Efficient Reductions for Distributed Deep Learning on Modern Dense GPU Systems, " ICS-2020, June-July 2020.

Distributed Training with TensorFlow and MVAPICH2-GDR

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



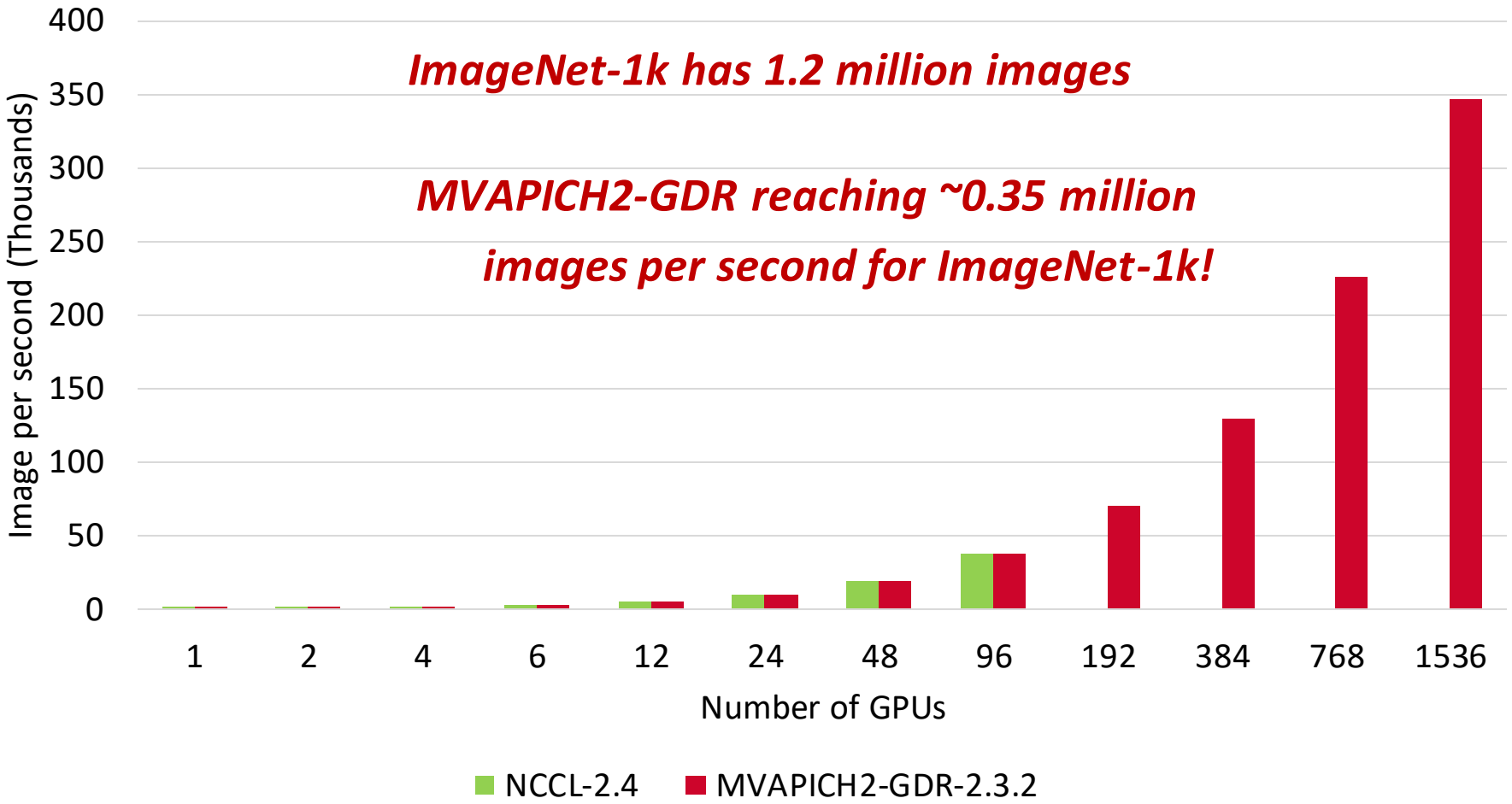
$$\text{Scaling Efficiency} = \frac{\text{Actual throughput}}{\text{Ideal throughput at scale}} \times 100\%$$



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 x 90 = 332 seconds = **5.5 minutes!**

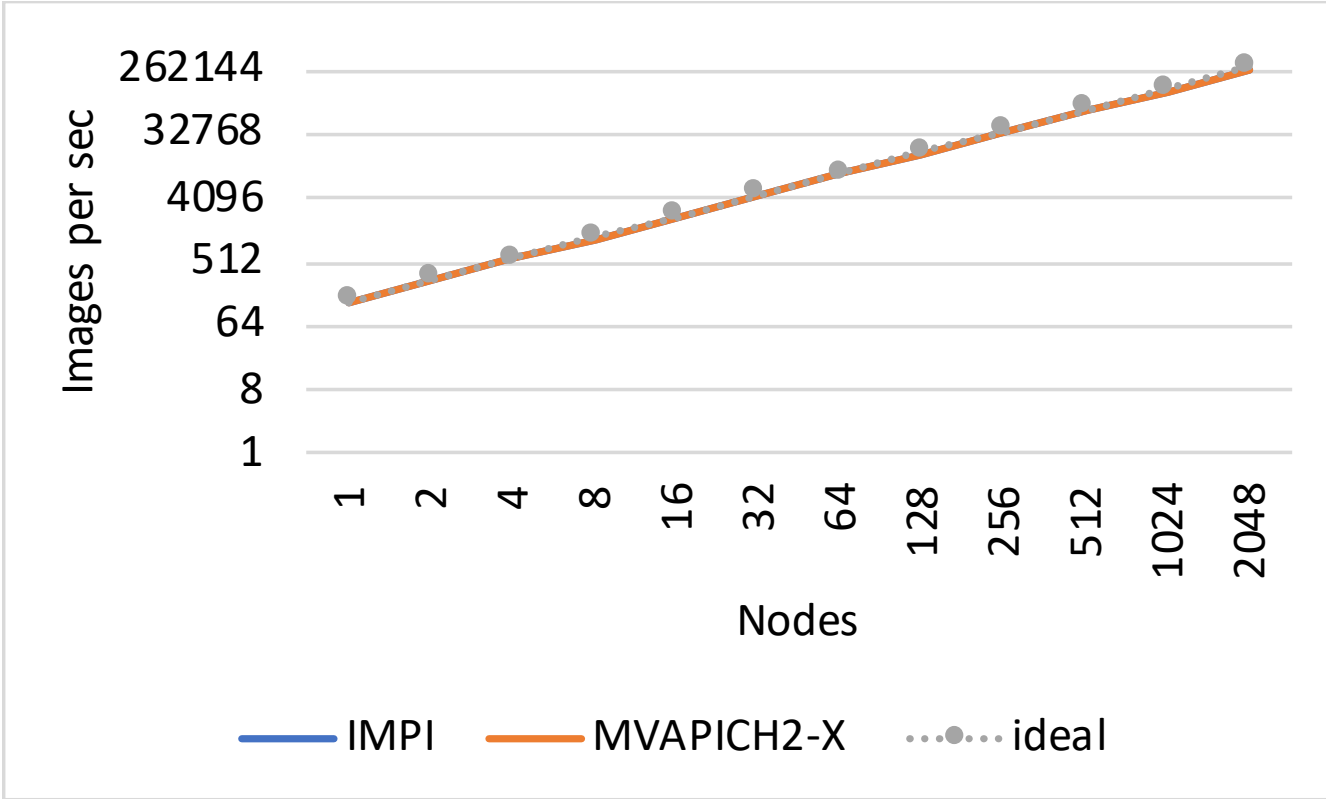


*We observed errors for NCCL2 beyond 96 GPUs

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

Distributed TensorFlow on TACC Frontera (2048 CPU nodes)

- Scaled TensorFlow to 2048 nodes on Frontera using MVAPICH2 and IntelMPI
- MVAPICH2 and IntelMPI give similar performance for DNN training
- Report a peak of 260,000 images/sec on 2048 nodes
- On 2048 nodes, ResNet-50 can be trained in 7 minutes!

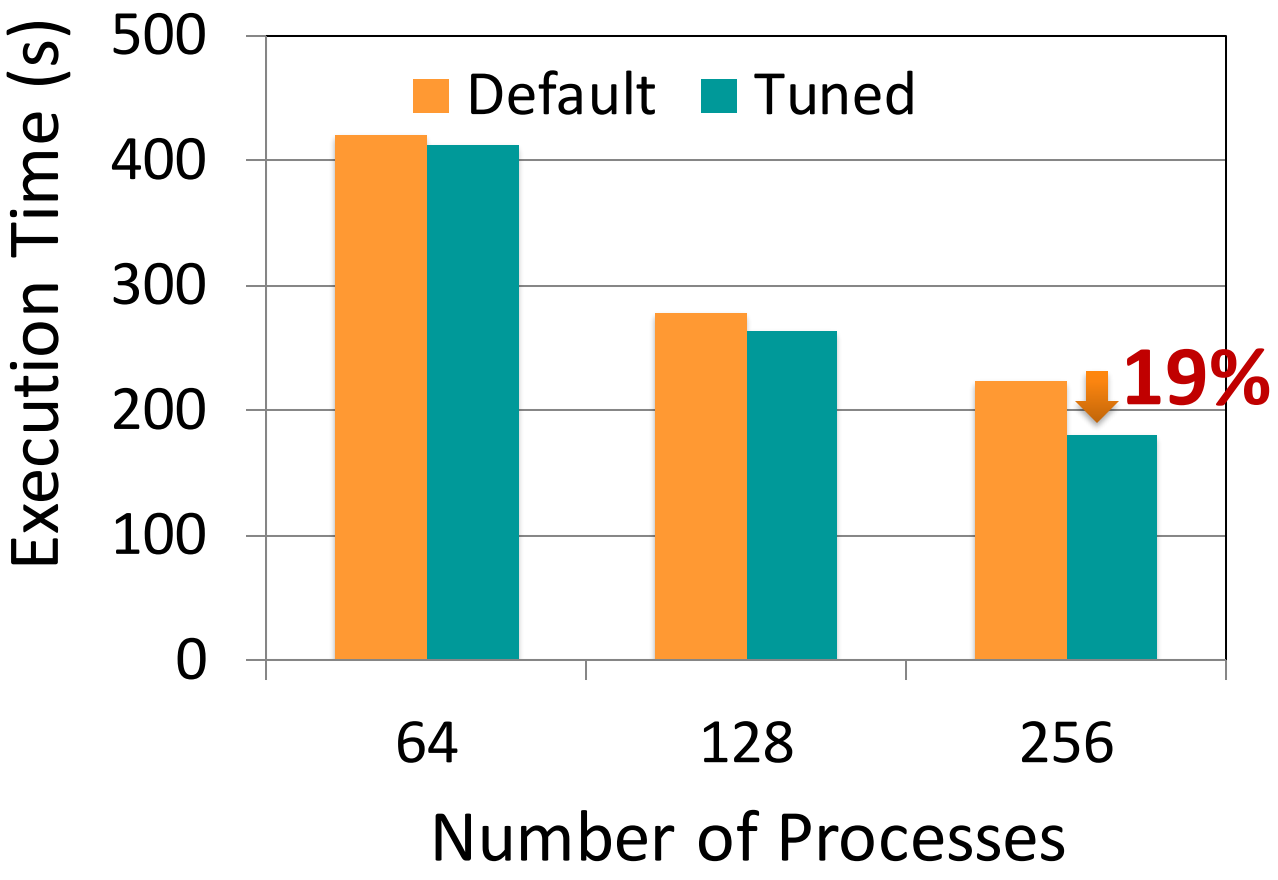


A. Jain, A. A. Awan, H. Subramoni, DK Panda, "Scaling TensorFlow, PyTorch, and MXNet using MVAPICH2 for High-Performance Deep Learning on Frontera", DLS '19 (SC '19 Workshop).

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.

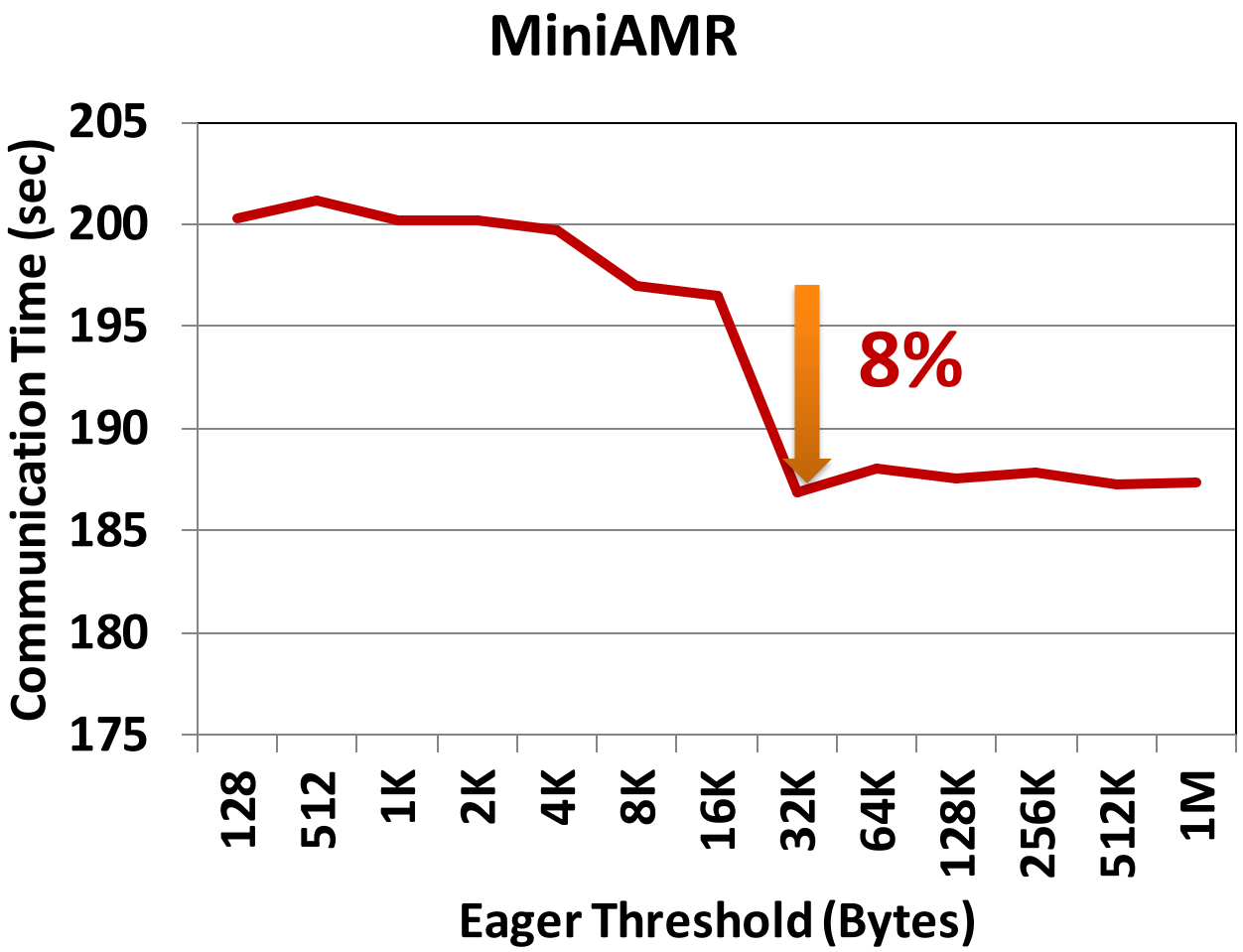
Amber: Impact of Tuning Eager Threshold



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

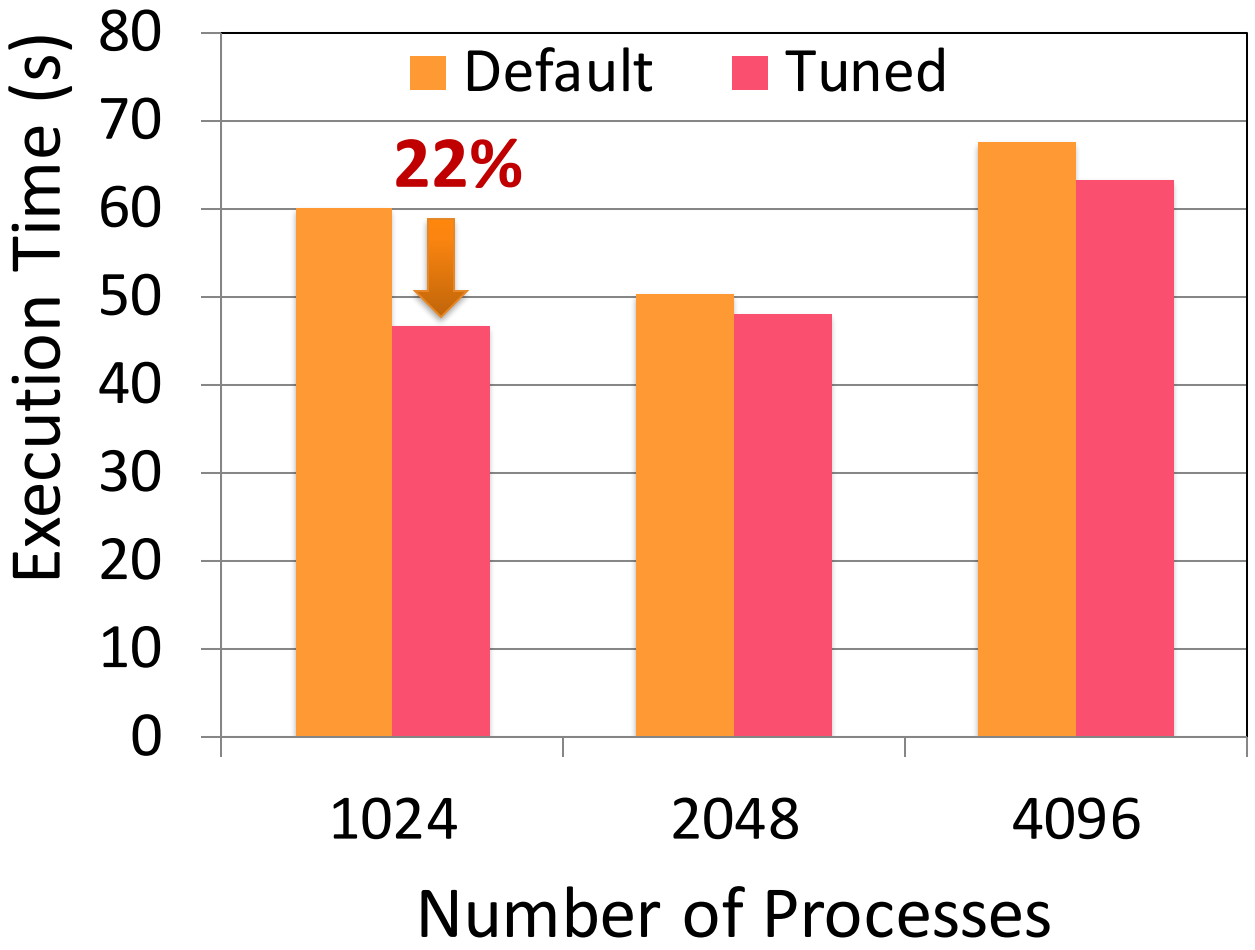
MiniAMR: Impact of Tuning Eager Threshold



- 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

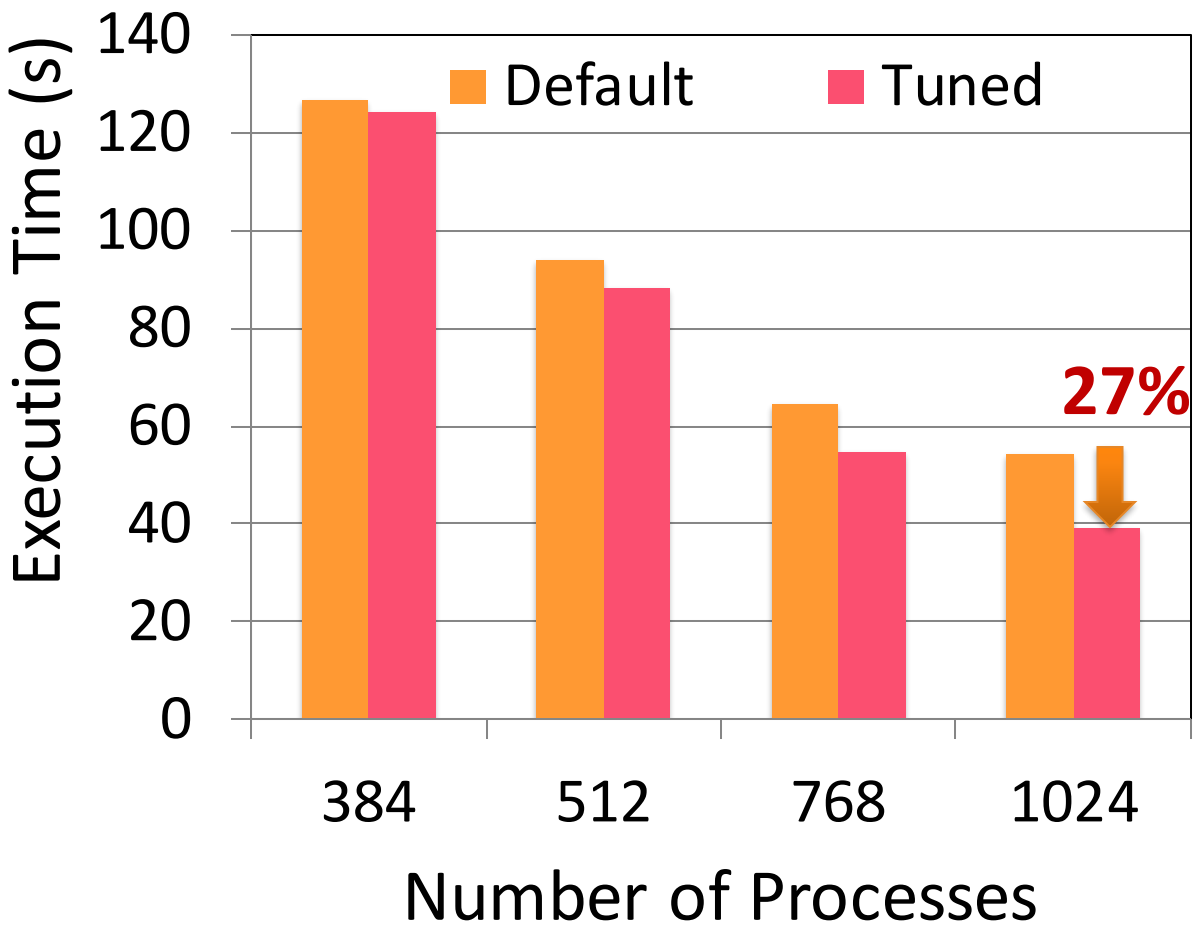
SMG2000: Impact of Tuning Transport Protocol



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

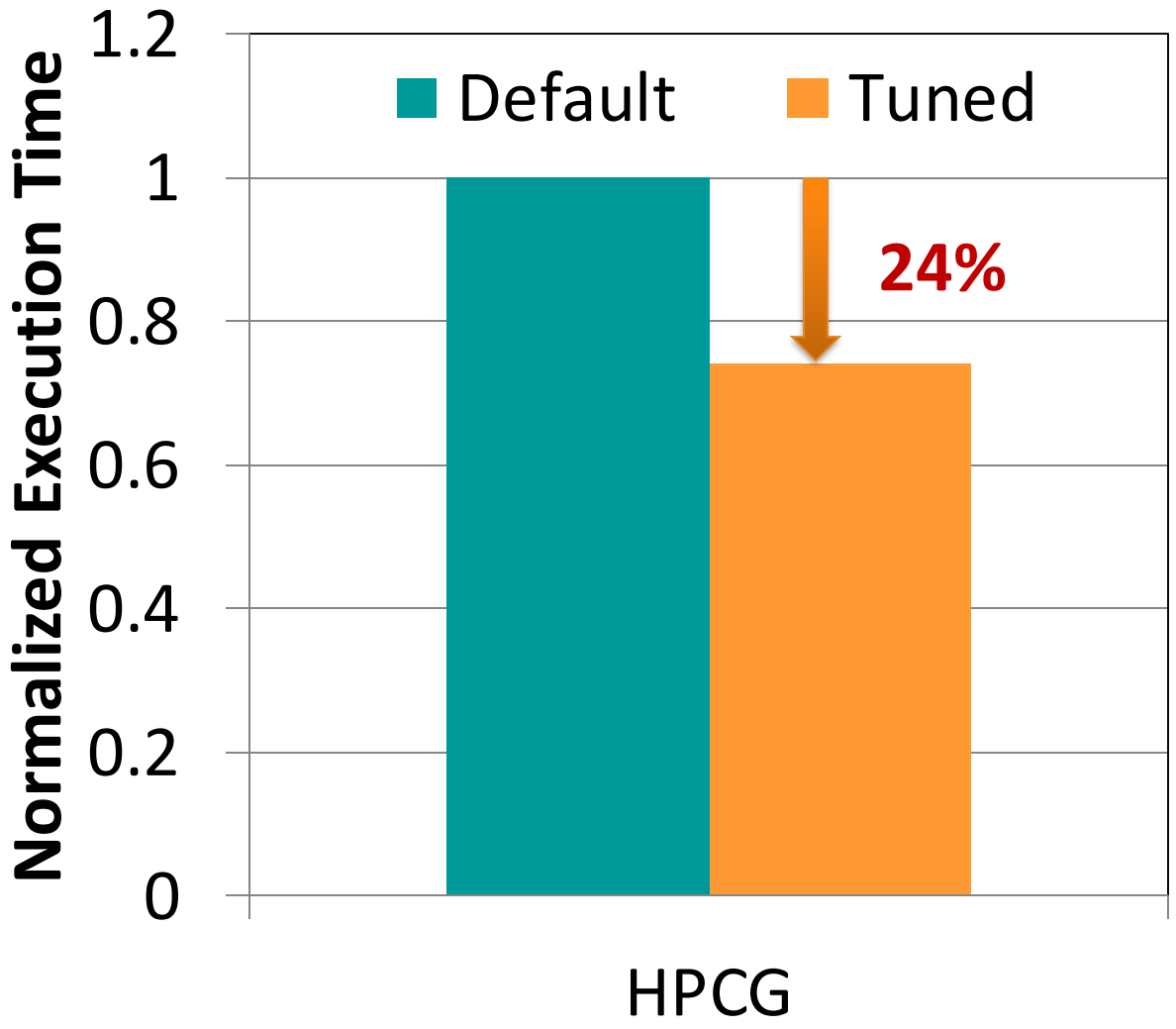
Neuron: Impact of Tuning Transport Protocol



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.

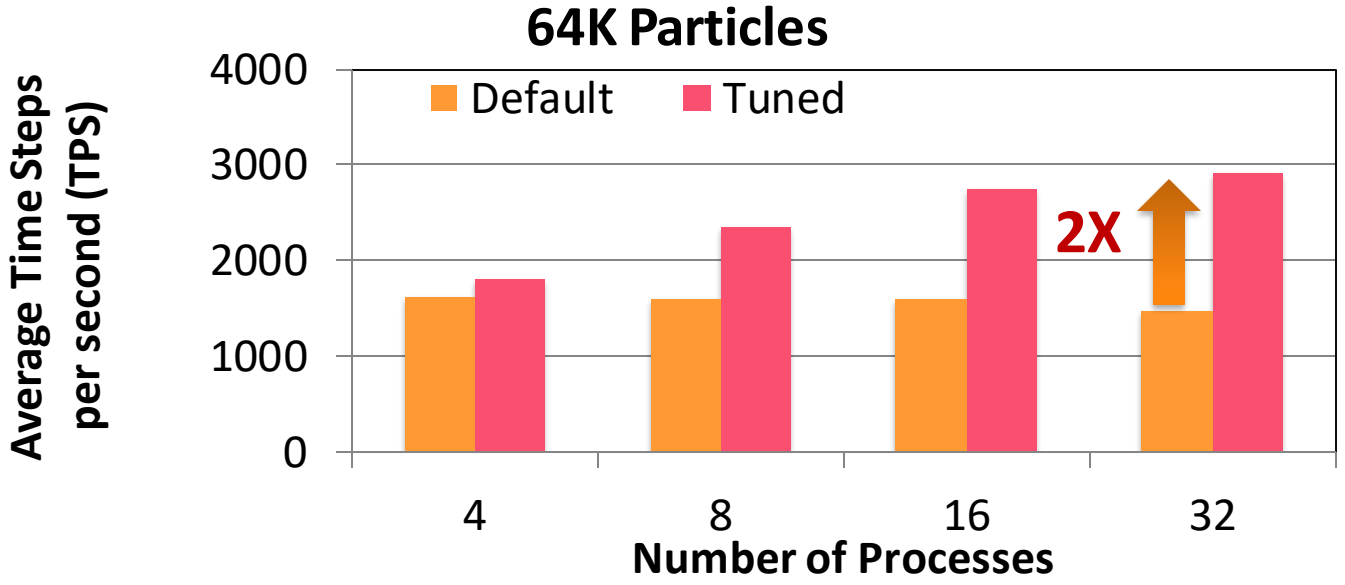
HPCG: Impact of Collective Tuning for MPI+OpenMP Programming Model

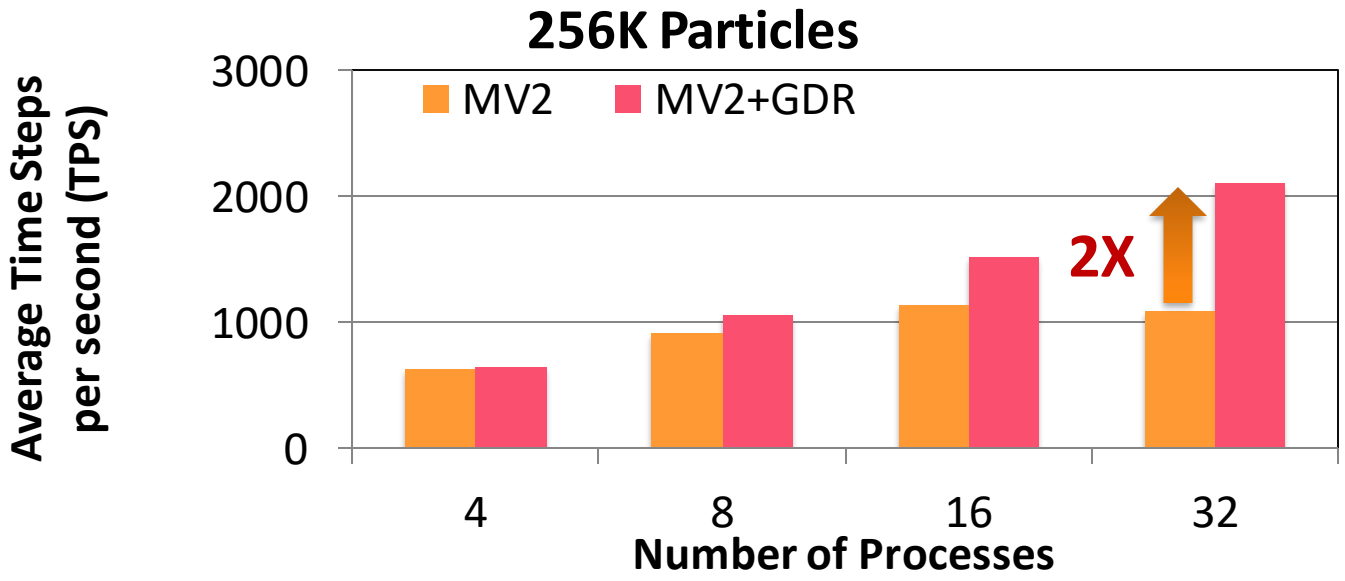


- Partial subscription nature of hybrid MPI+OpenMP programming requires a new level of collective tuning
  - For PPN=2 (Processes Per Node), the tuned version of MPI_Reduce shows 51% improvement on 2,048 cores
- 24% improvement on 512 cores
  - 8 OpenMP threads per MPI processes
- Library Version: MVAPICH2 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

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

HOOMD-blue: Impact of GPUDirect RDMA Based Tuning



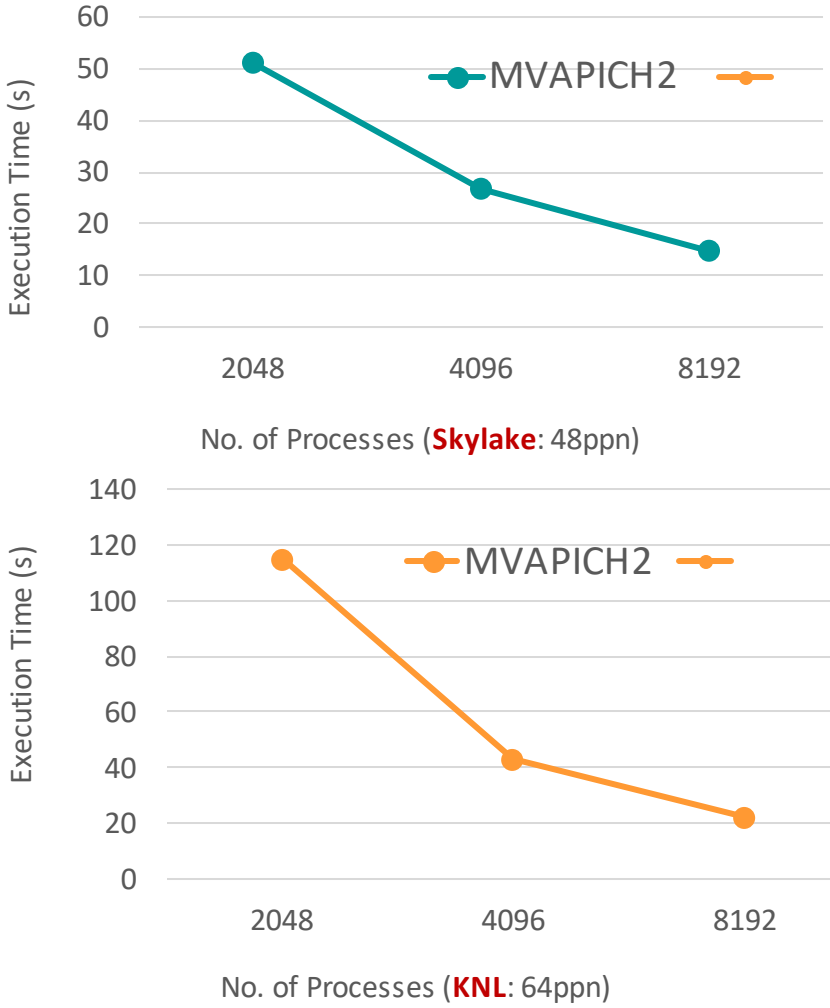


Data Submitted by Khaled Hamidouche @ OSU

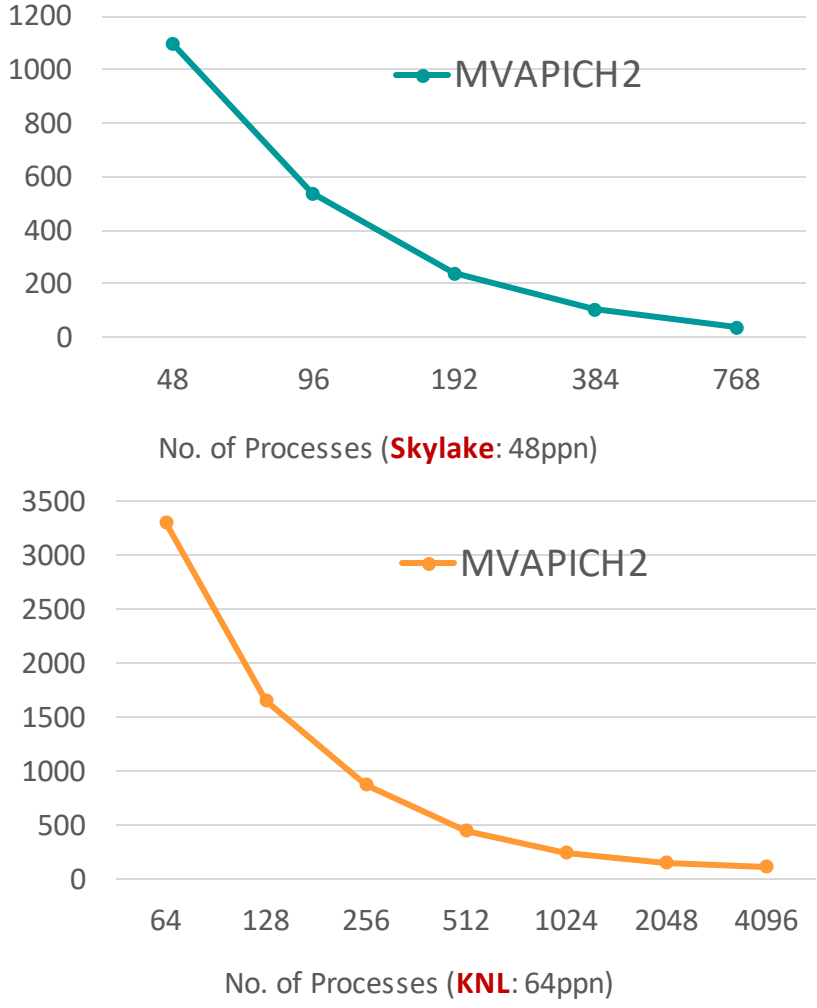
- HOOMD-blue is a Molecular Dynamics simulation using a custom force field.
- GPUDirect specific features selection and tuning significantly benefit the HOOMD-blue application. We observe a factor of 2X improvement on 32 GPU nodes, with both 64K and 256K particles
- Library Version: MVAPICH2-GDR 2.2
- 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

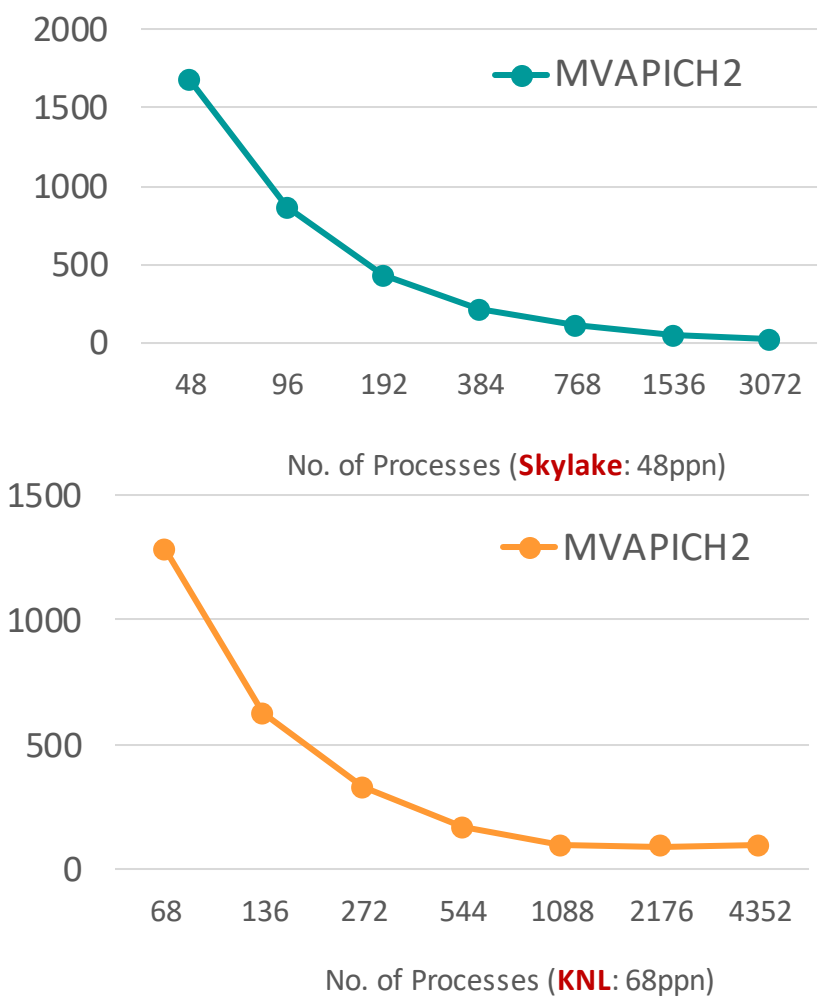
MiniFE (1300x1300x1300 ~ 910 GB)



NEURON (YuEtAl2012)



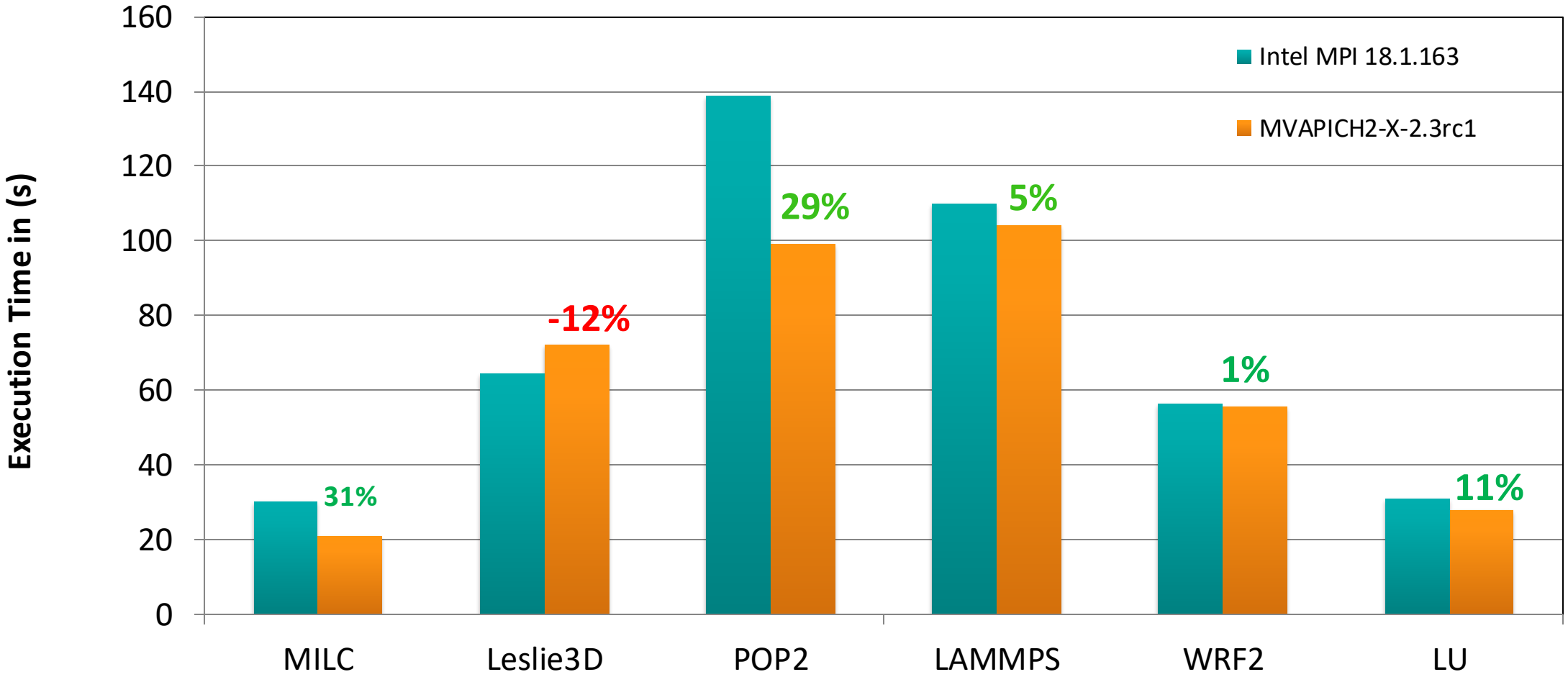
Cloverleaf (bm64) MPI+OpenMP, NUM_OMP_THREADS = 2



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*
  - Bluefield based offload*
- Enhanced communication schemes for upcoming architectures
  - ROCm*
  - 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

For More Details on other MVAPICH2 Libraries/Features

- MPI_T Support
  - More details in the talk "Performance Engineering using MVAPICH and TAU" by Sameer Shende (Paratools/UO) on Tuesday (08/25/2020) from 3:30 PM - 4:00 PM EDT
- MVAPICH2-Azure
  - More details in the talk "MVAPICH2 on Microsoft Azure HPC" by Jithin Jose (Microsoft, Azure) on Tuesday (08/25/2020) from 1:30 PM - 2:00 PM EDT
- MVAPICH2-AWS
  - More details in the talk "Scaling Message Passing on Amazon Web Services with Elastic Fabric Adapter" by Raghunath Rajachandrasekar (AWS) on Tuesday (08/25/2020) from 1:00 PM - 1:30 PM EDT
- Support for SHARP
  - More details in the talk "Impact of SHARP and Adaptive Routing on Applications on Frontera" by John Cazes (TACC) on Wednesday (08/26/2020) from 12:00 PM - 12:30 PM EDT

Funding Acknowledgments

Funding Support by

Equipment Support by

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

Current Students (Graduate)

- Q. Anthony (Ph.D.)
- M. Bayatpour (Ph.D.)
- C.-H. Chu (Ph.D.)
- A. Jain (Ph.D.)
- M. Kedia (M.S.)
- K. S. Khorassani (Ph.D.)
- P. Kousha (Ph.D.)
- N. S. Kumar (M.S.)
- B. Ramesh (Ph.D.)
- K. K. Suresh (Ph.D.)
- N. Sarkauskas (Ph.D.)
- S. Srivastava (M.S.)
- S. Xu (Ph.D.)
- Q. Zhou (Ph.D.)

Current Research Scientists

- A. Shafi
- H. Subramoni

Current Senior Research Associate

- J. Hashmi

Current Software Engineer

- A. Reifsteck

Current Post-docs

- M. S. Ghazimeersaeed
- K. Manian

Current Research Specialist

- J. Smith

Past Students

- A. Awan (Ph.D.)
- 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. Chakraborthy (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- J. Hashmi (Ph.D.)
- 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.)
- K. Raj (M.S.)
- R. Rajachandrasekar (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- N. Sarkauskas (B.S.)
- A. Singh (Ph.D.)
- J. Sridhar (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
- H.-W. Jin
- J. Lin
- M. Luo
- E. Mancini
- S. Marcarelli
- A. Ruhela
- J. Vienne
- H. Wang

Past Research Scientists

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

Past Programmers

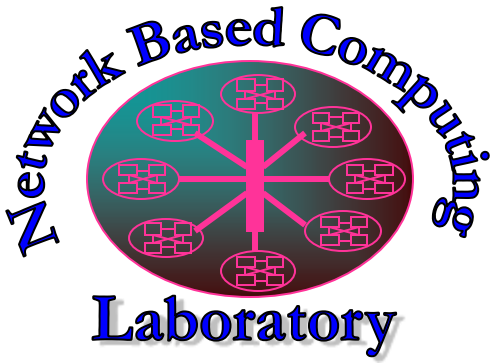
- D. Bureddy
- J. Perkins

Past Research Specialist

- M. Arnold

Thank You!

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

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

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