MVAPICH
MPI, PGAS and Hybrid MPI+PGAS Library

Boosting the Performance of HPC Applications with MVAPICH and MVAPICH-Plus

A Tutorial at MUG'26

Presented by

Nathaniel Shineman and Benjamin Michalowicz

The MVAPICH Team

The Ohio State University

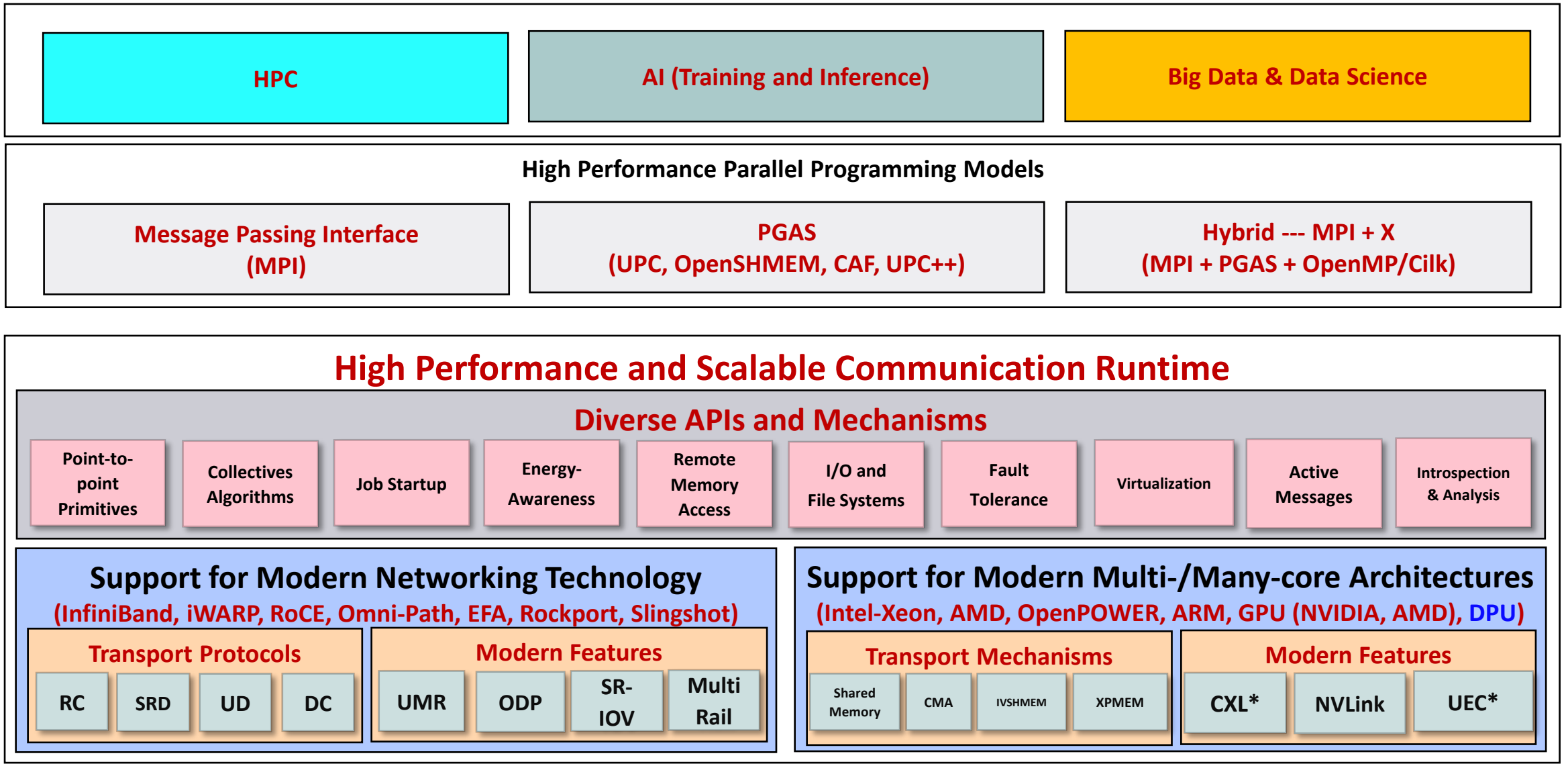
http://mvapich.cse.ohio-state.edu/

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-5.0 standard
- http://mvapich.cse.ohio-state.edu
- Additional optimized versions for different systems/environments:
  - MVAPICH-Plus (Unification of MVAPICH2-X and MVAPICH2-GDR), since 2023
  - 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 3,500 organizations in 94 countries (listed under the Users Tab of the MVAPICH page)**
- **More than 2.09 Million downloads from the OSU site directly**
- Empowering many TOP500 clusters (Jun '26 ranking)
  - 27th , 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 90th , 448, 448 cores (Frontera) at TACC
  - 298th, 42,336 cores (Cosmos) at SDSC
- Available with software stacks of many vendors and Linux Distros (RedHat, SuSE, OpenHPC, and Spack)
- Partner in the 90th ranked TACC Frontera system
- **Empowering Top500 systems for more than 20+ years**

MVAPICH Architecture (HPC, AI, Big Data, & Data Science)



* Upcoming

Updated User Guide and Resources

- https://mvapich-docs.readthedocs.io/en/latest/
- Latest RPM install instructions, CVAR names/aliases/etc.
- Runtime instructions, setting up environment variables, OMB installation and execution, FAQ

Contents

- CPU Process Mapping with MVAPICH
  - Summary of parallel launchers
- CH4 devices and their impact
  - UCX, OFI, enhanced libfabrics from OSU
- Dynamic Tuning
  - CPU, GPU dynamic tuning
- GPU-Awareness
  - NVIDA, AMD, Intel GPU support
- Application Case Study: High-Performance Linpack (HPL) on NVIDIA NVL72
- Future Plans

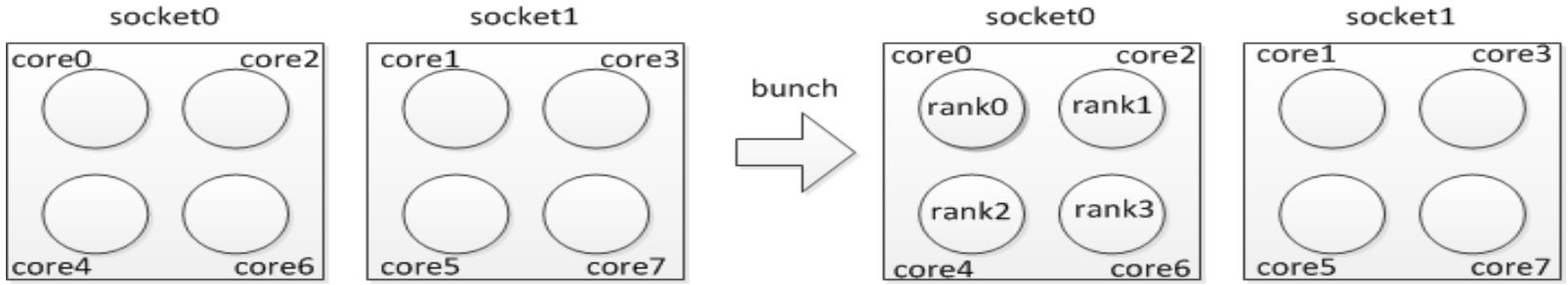
Common Parallel Launchers

- Hydra – included with MVAPICH/MVAPICH-Plus
  - mpiexec/mpirun binaries
  - Compatible with all commonly used resource managers
    - SLURM, PBS, Flux, etc.
    - Uses PMI1 interface standard – if on a system with multiple PMI versions/PMIx, use `MVP_PMI_VERSION=PMI1`
- Srun – included with the Slurm resource manager
  - Mutually exclusive with hydra and other launchers
    - Uses slurm specific PMI2
    - --with-pm=none --with-pmi=slurm

CPU Process Mapping

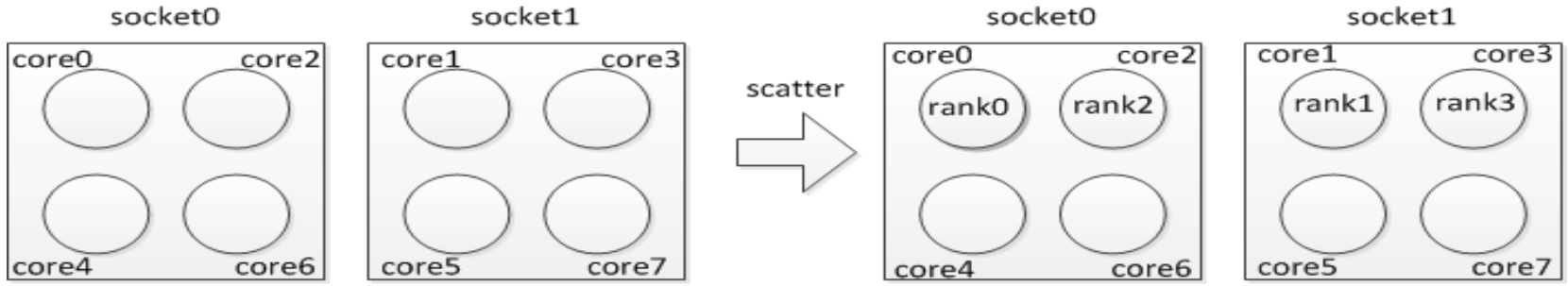
- Determines Process-to-Core mapping At Runtime via resource managers and parallel job launchers
- Supported by both Hydra and srun
  - Hydra: “mpiexec/mpirun … -bind-to <x> -map-by <y>” (and use of hwloc)
  - Srun: “srun --cpu-bind=<x>”
    - See slurm documentation for more details
- Common cases
  - “Bunch”
  - “Scatter”
  - “Spread”
  - Custom mappings
- To see process mappings at runtime:
  - Hydra: set HYDRA_TOPO_DEBUG=1
  - Srun: set “SLURM_CPU_BIND_VERBOSE=verbose” or "srun --cpu-bind=verbose[,options]"

Bunch Mapping



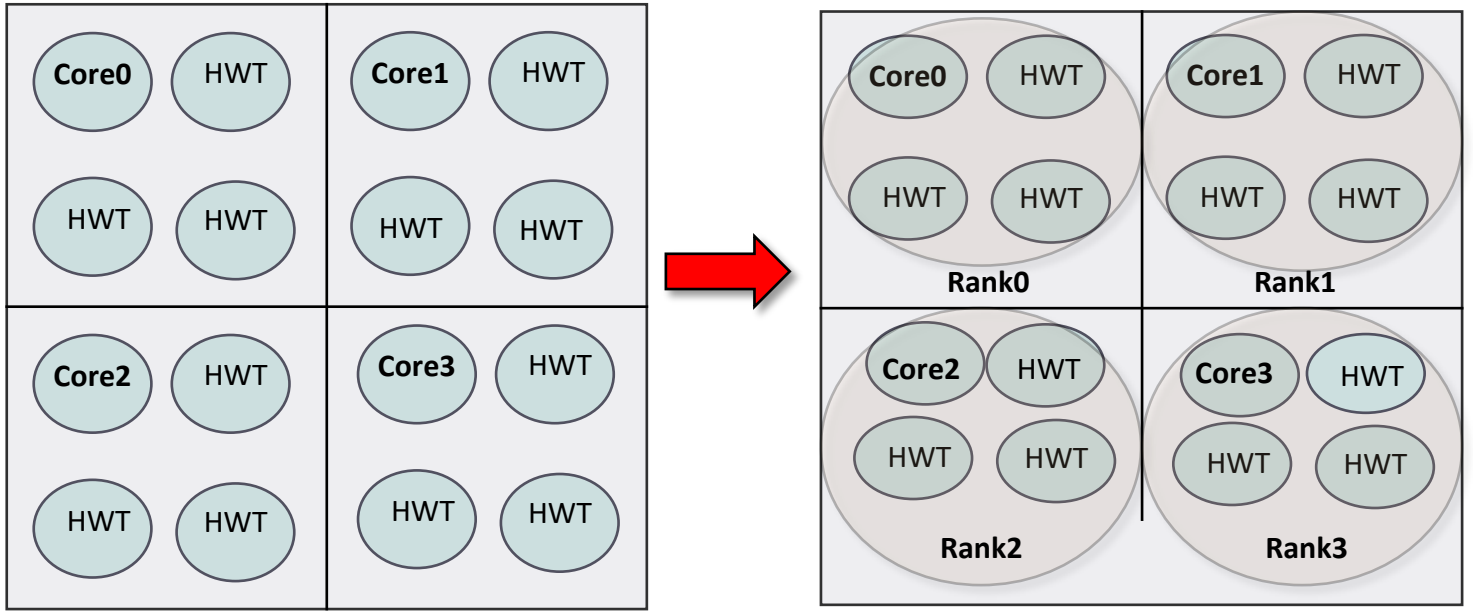
- Allocates ranks sequentially on cores
  - Optimal for near-neighbor communication
  - Can heavily leverage shared memory, CMA, XPMEM
  - Does not provide optimal resource distribution of memory, network devices, GPU, etc
  - Useful when fully subscribing nodes
- Launcher commands:
  - Hydra: mpiexec -np <num_tasks> … -map-by core -bind-to core
  - Srun: srun –n <num_tasks> … --cpu-bind=core --distribution=block:block

Scatter Mapping



- Performs a round-robin assignment per-socket
  - Prioritizes network device and memory access over intra-node transfers
  - Can be applied to other map-by resources, ie NUMA or L3 cache
  - Not optimal for applications bound by intra-node communication
  - Useful for sparse allocations with heavy network use and multiple network adapters
- Launcher commands:
  - Hydra: mpiexec -np <num_tasks> … -map-by socket -bind-to core
  - Srun: srun –n <num_tasks> … --cpu-bind=core --distribution=cyclic:cyclic

Spread Mapping



- Handles MPI+Threads workloads
  - Works with and without hyperthreading activated for a compute node
  - Best practice: divide total cores by requested ppn and map each process to N cores
- Launcher Commands:
  - Hydra: mpiexec --bind-to core --map-by core:<total_cores / ppn>
  - Srun: srun –n <num_tasks> … --cpus-per-task=<total_cores / ppn> --ntasks-per-node=<ppn> --cpu-bind=core

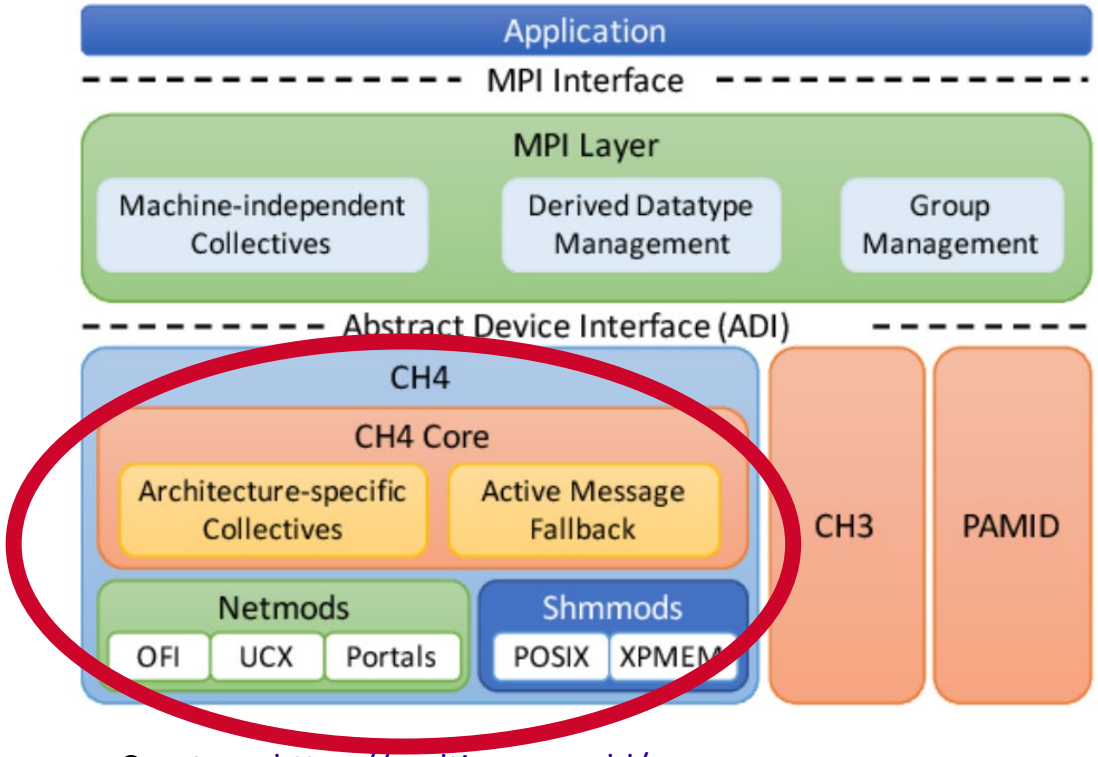
Custom Mappings

- Not all standard mappings work best for all applications
- Need fine-tuning/architecture/numa-aware process placement
- Hydra custom mappings:
  - Supports binding/mapping to NUMA, socket, core, hwthread, l1/2/3/4/5 cache, pci devices, GPUs, IB HCAs, eth devices, hfi, etc., and user-defined mappings
- SLURM/srun custom mappings:
  - See https://slurm.schedmd.com/mc_support.html for options
  - Additionally supports binding to memory regions with "--mem-bind=…"

Contents

CH4 Devices – Overview

- CH4 device
  - Replaced CH3 with MPICH 3.4 in 2021
- 2 components to CH4 device
  - Netmods (Network)
  - Shmmods (Shared Memory)
- MVAPICH Netmods:
  - UCX (Unified Communications X)
    - Recommended for IB/RoCE networks
  - OFI (Open Fabrics Interfaces/Libfabrics)
    - Recommended for all other networks
- Configure with
  - --with-device=ch4:ofi --with-device=ch4:ucx



Courtsey: https://multicore.world/wp-content/uploads/2024/02/multicoreworld24-brightwell.pdf

Using the UCX Netmod

- Configure UCX path with `--with-ucx=/path/to/ucx` or `embedded` options
- Common UCX environment variables:
  - UCX_RNDV_THRESH: transition point for eager/rendezvous message transfer protocols; provide a number in bytes
  - UCX_MAX_RNDV/EAGER_RAILS: number of network devices used per process for eager and rendezvous-based transfers
    (maximum supported is 4 (https://openucx.readthedocs.io/en/master/faq.html))
  - UCX_TLS: comma-separated list of transports used by UCX (rc, ud, shm, cma, xpmem, etc.)
  - UCX_NET_DEVICES: sets the device(s) UCX will use for message transports
  - UCX_LOG_LEVEL: logging information in various verbosities (trace, info, debug, error, fatal)

Using the OFI Netmod

- Configure UCX path with `--with-libfabric=/path/to/libfabric` or `embedded` options
- Common OFI environment variables:
  - FI_PROVIDER: forces a specific network provider to be used
    - MVAPICH will attempt to choose the highest performing provider that supports your network
    - Generally not necessary unless a particular network is required for testing
  - MPIR_CVAR_CH4_OFI_CAPABILITY_SETS_DEBUG: set to 1 to view the OFI provider being selected at runtime
  - MVP_CH4_OFI_ENABLE_HMEM: set to 1 to allow OFI provider to use GPU direct RDMA support
  - Highly provider dependent
  - More details at https://ofiwg.github.io/libfabric/main/man/

Contents

- CPU Process Mapping with MVAPICH
  - Summary of parallel launchers
- CH4 devices and their impact
  - UCX, OFI, enhanced libfabrics from OSU
- Dynamic Tuning
  - CPU, GPU dynamic tuning
- GPU-Awareness
  - NVIDA, AMD, Intel GPU support
- Application Case Study: High-Performance Linpack (HPL) on NVIDIA NVL72
- Future Plans

Dynamic Collective Tuning

- Available since MVAPICH-Plus 4.1
- Enables on-the-fly collective selection tuning
- Combines collective knowledge with real-time, adaptive, system-specific analysis
- Runs potential algorithms and collects runtime data to determine best collective algorithm at a given time

Controlling Dynamic Tuning with CVARs

Parameter	Significance	Default	Notes
MVP_ENABLE_DYNAMIC_TUNING	• Allows dynamic tuning to be activated or deactivated at runtime or within an application	1	• Can be disabled after warmups to prevent retuning
MVP_FORCE_STATIC_TUNING	• Force MVAPICH to use static tables	0	• Preferred method to completely disable dynamic tuning and all derived results
MVP_DYNAMIC_TUNE_THRESHOLD	• Controls when to retest algorithms	1000	• Eligible algorithms will be rerun after N collective calls of that size • Tracked per-comm • Match to OMB iterations for microbenchmark testing

Controlling Dynamic Tuning with CVARs

Parameter	Significance	Default	Notes
MVP_DYNAMIC_TUNE_TEST_COUNT	• Controls the number of times each algorithm is run during testing	5	• First iteration is treated as a warmup and is not timed • Requires N * available algorithm iterations to select
MVP_DYNAMIC_TUNE_LOCAL_EVAL_METRIC	• The metric used by each process to identify the "best" algorithm • Options: min, max, avg	avg	• Which algorithm is reported back to the tuning framework as the local preference • Reports algorithm with lowest {avg, min, max} latency
MVP_DYNAMIC_TUNE_SELECTION_METRIC	• The metric used to select the overall "best" algorithm • Options: min, max, avg	max	• Which metric will be minimized by the tuning framework • Max will minimize the maximum latency across all ranks for most consistent performance

Contents

- CPU Process Mapping with MVAPICH
  - Summary of parallel launchers
- CH4 devices and their impact
  - UCX, OFI, enhanced libfabrics from OSU
- Dynamic Tuning
  - CPU, GPU dynamic tuning
- GPU-Awareness
  - NVIDA, AMD, Intel GPU support
- Application Case Study: High-Performance Linpack (HPL) on NVIDIA NVL72
- Future Plans

MPI + CUDA - Naive

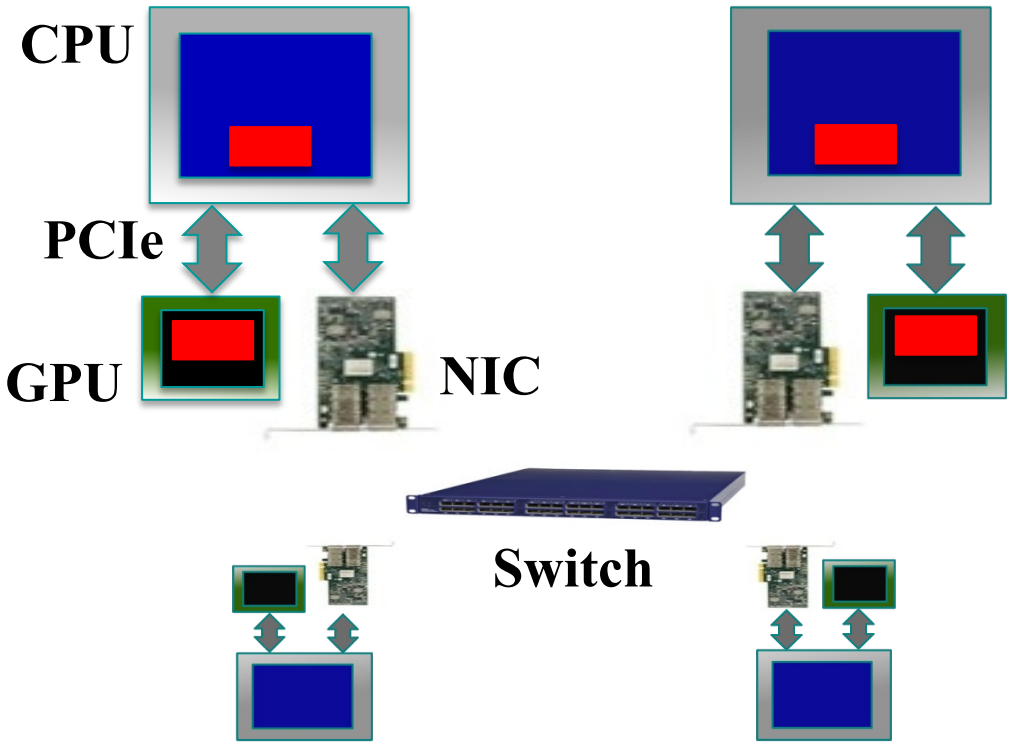
- Data movement in applications with standard MPI and CUDA interfaces

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

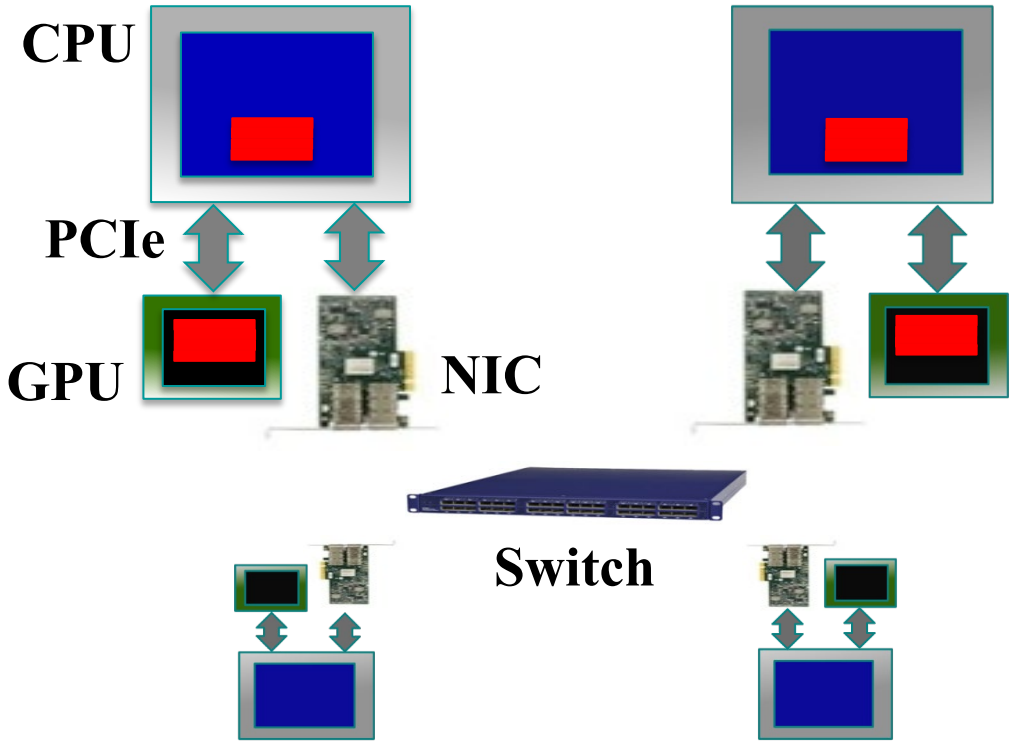
MPI + CUDA - Advanced

- 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 MPI Library: MVAPICH-Plus

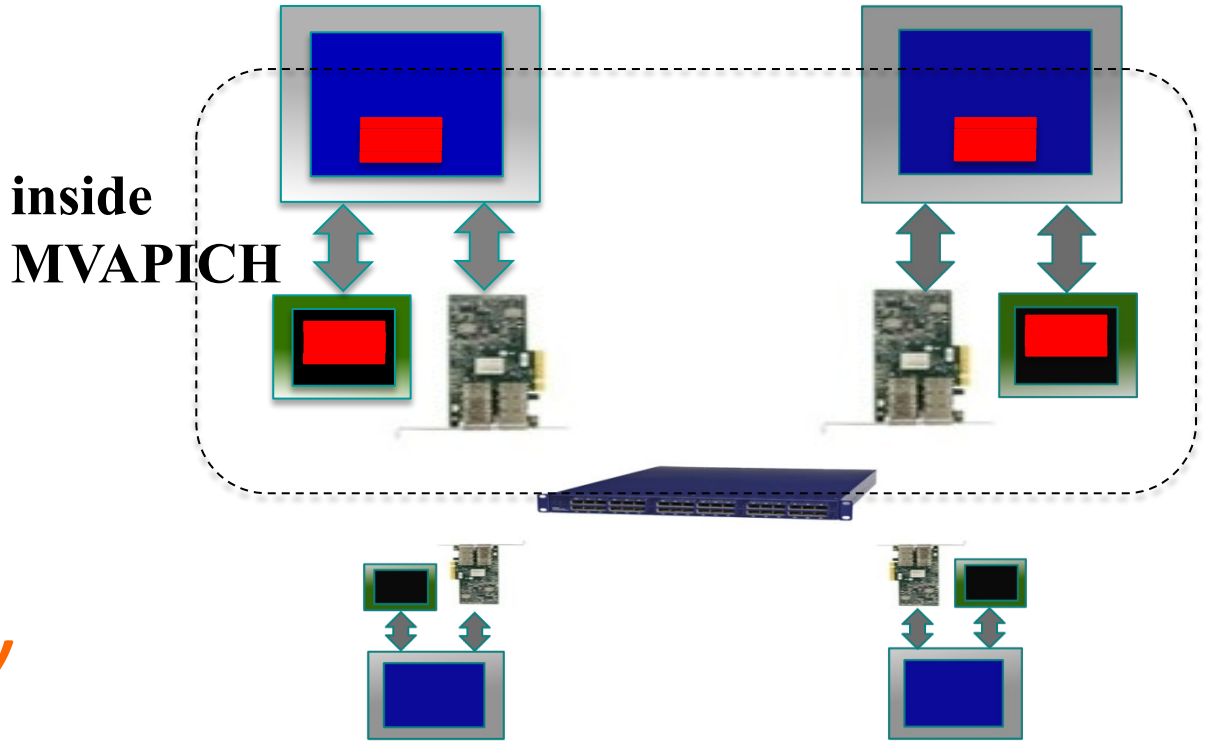
- Standard MPI interfaces used for unified data movement
- Takes advantage of Unified Virtual Addressing (>= CUDA 4.0, Similar Support since ROCm 4.5, since 2015 for Intel GPUs)
- Overlaps data movement from GPU with RDMA transfers

At Sender:

```
MPI_Send(s_devbuf, size, …);
```

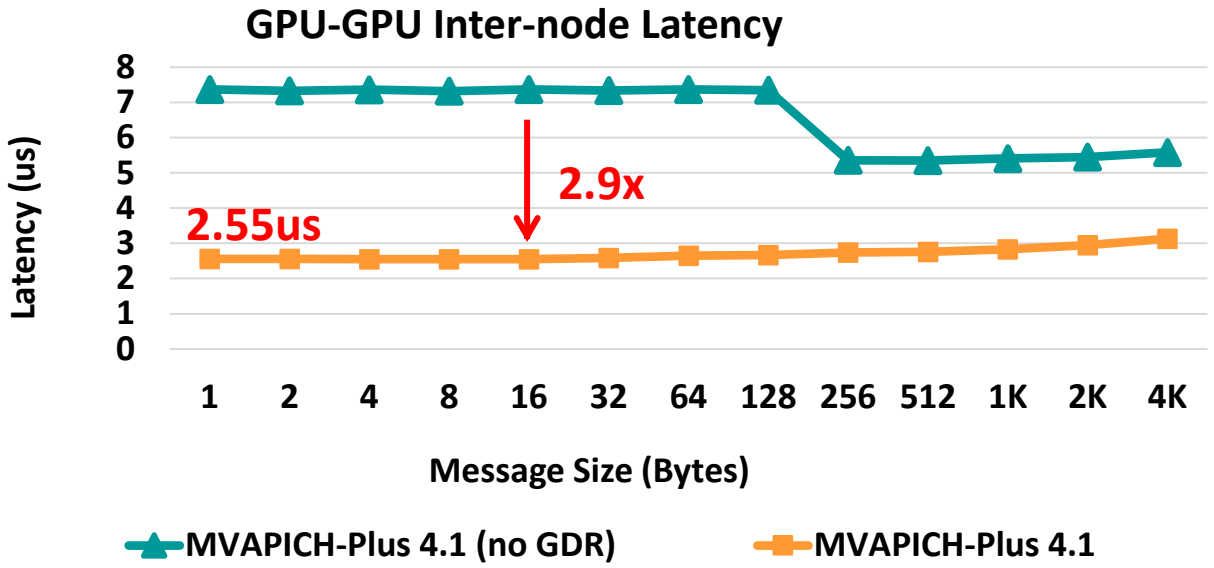
At Receiver:

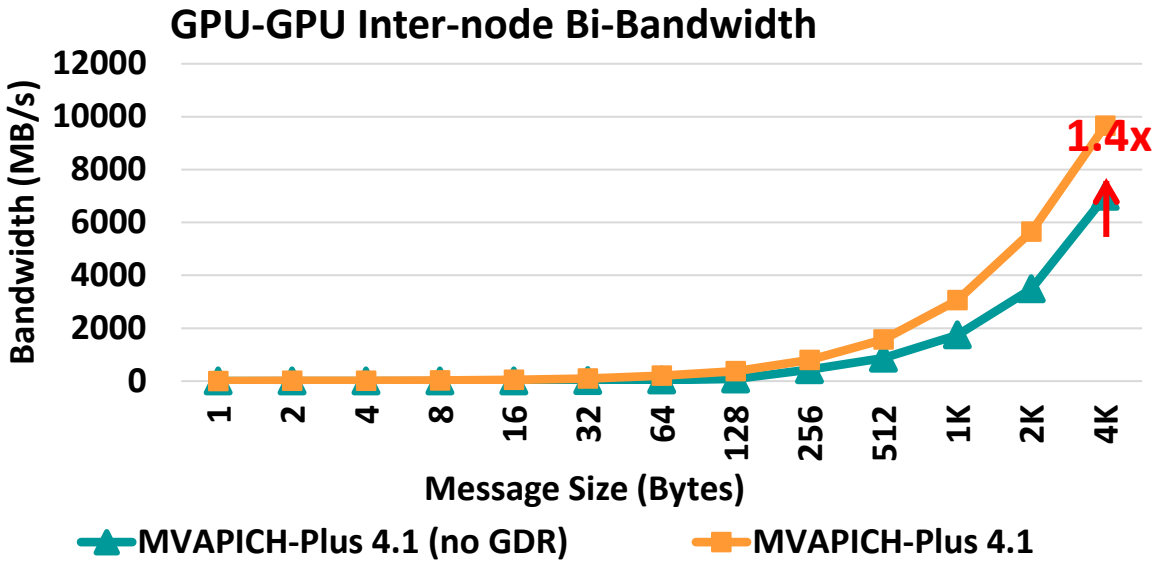
```
MPI_Recv(r_devbuf, size, …);
```

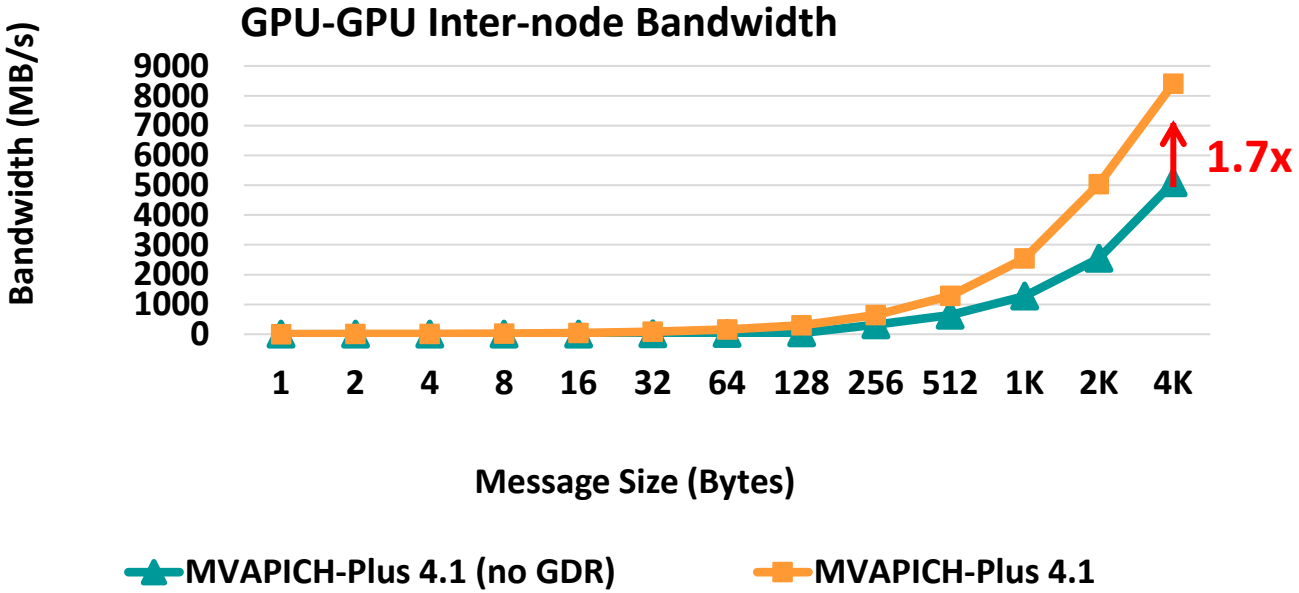


High Performance and High Productivity

Optimized MVAPICH-Plus Design



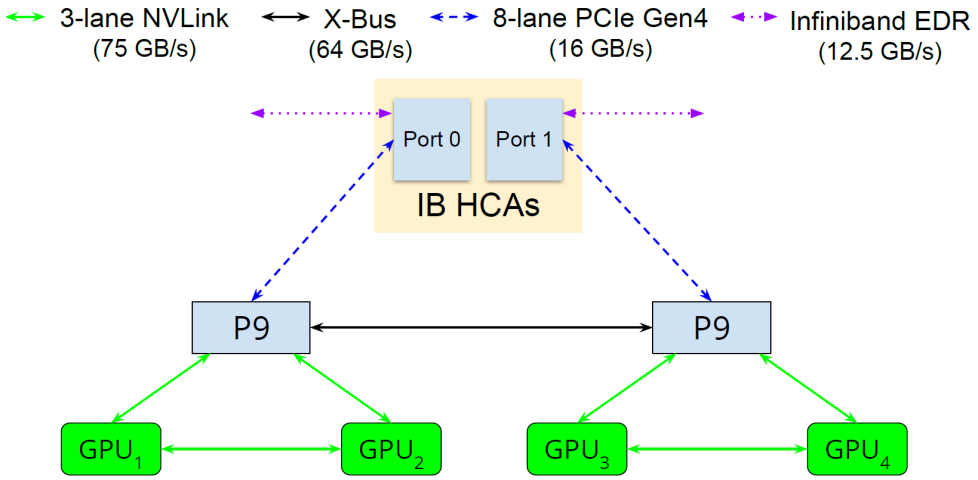




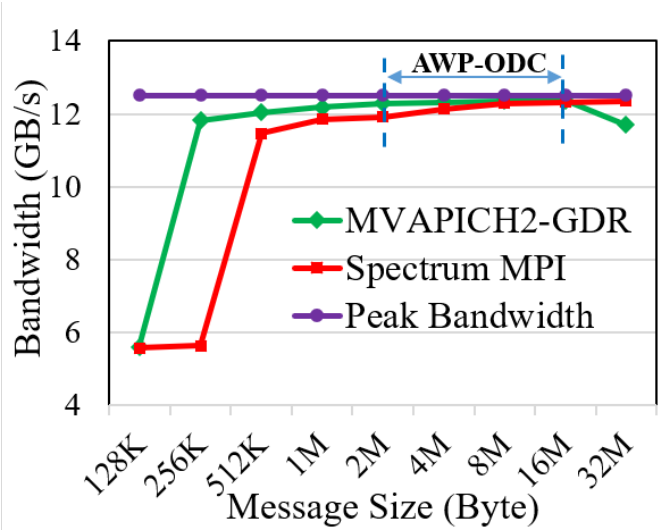
MVAPICH-Plus 4.1
GH200 Superchip (Grace @ 3.1 GHz) node - 72 cores
NVIDIA H100 GPU
Mellanox ConnectX-7 NDR400 HCA
CUDA 12.4
Mellanox OFED 25.01 with GPU-Direct-RDMA

MVAPICH-Plus: "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.

NEW: Compression Hints

- Supports communicator hints at creation or on update
  - mvp_cmp_enable_p2p
  - mvp_cmp_enable_allcoll
  - mvp_cmp_enable_allgather
  - mvp_cmp_enable_allreduce
  - mvp_cmp_enable_alltoall
  - mvp_cmp_enable_reduce_scatter
- Modify hints for allreduce compression

```
MPI_Info hints;
MPI_Info_create(&hints);
MPI_Info_set(hints, "mvp_cmp_enable_allreduce", "true");
MPI_Comm_set_info(MPI_COMM_WORLD, hints);
```

Tuning Pt2pt Compression Designs in MVAPICH-Plus

Parameter	Significance	Default	Notes
MVP_ENABLE_PT2PT_GPU_COMPRESSION	• Enable/disable compression for GPU transfers • Can be no, all, or percomm	percomm	• Recommended setting percomm allows for hints based implementation • Will have no effect without explicit hints
MVP_PT2PT_GPU_COMPRESSION_THRESHOLD	• Minimum message size to use pt2pt compression	1M Bytes	• Tune for your system/application

- **Refer to CVAR section of MVAPICH-Plus user guide for more information**
- https://mvapich-docs.readthedocs.io/en/mvapich-plus/cvar.html

Tuning Collective Compression Designs in MVAPICH-Plus

Parameter	Significance	Default	Notes
MVP_ENABLE_COMPRESSION	• Enable/disable compression for GPU transfers • Can be no, all, or percomm	percomm	• Recommended setting percomm allows for hints based implementation • Will have no effect without explicit hints
MVP_<coll>_GPU_COMPRESSION_THRESHOLD	• Minimum message size for using compression enabled collective	4M Bytes	• Supports alltoall, allgather, allreduce, reduce_scatter for coll • Only effective when compression enabled algorithm is in use • Compression algorithms included in tuning tables

- **Refer to CVAR section of MVAPICH-Plus user guide for more information**
- https://mvapich-docs.readthedocs.io/en/mvapich-plus/cvar.html

Contents

- CPU Process Mapping with MVAPICH
  - Summary of parallel launchers
- CH4 devices and their impact
  - UCX, OFI, enhanced libfabrics from OSU
- Dynamic Tuning
  - CPU, GPU dynamic tuning
- GPU-Awareness
  - NVIDA, AMD, Intel GPU support
- Application Case Study: High-Performance Linpack (HPL) on NVIDIA NVL72
- Future Plans

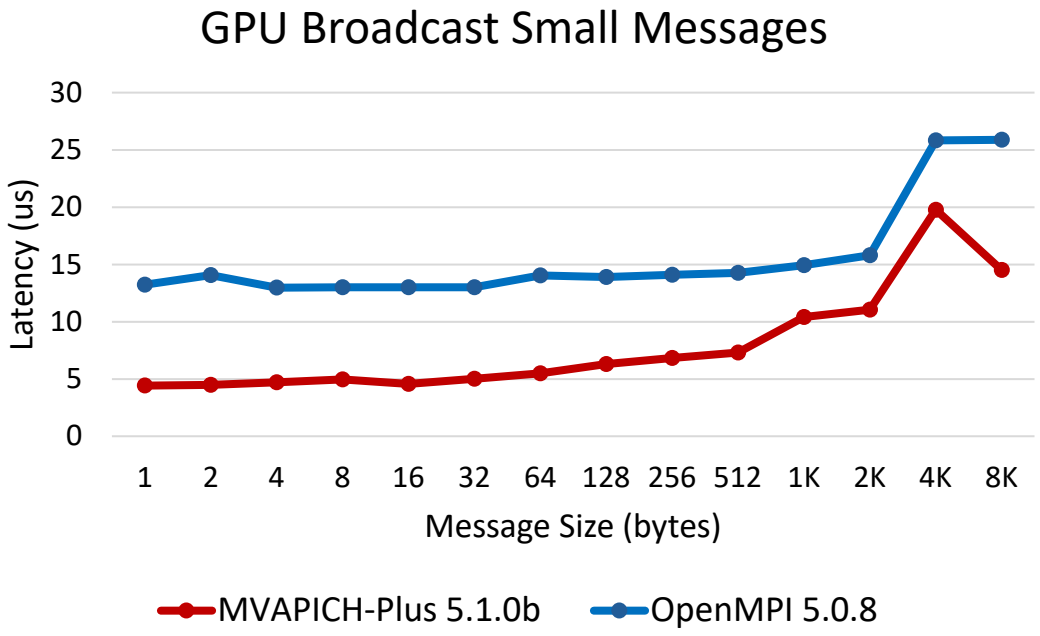
Motivation: Microbenchmarks are half the battle

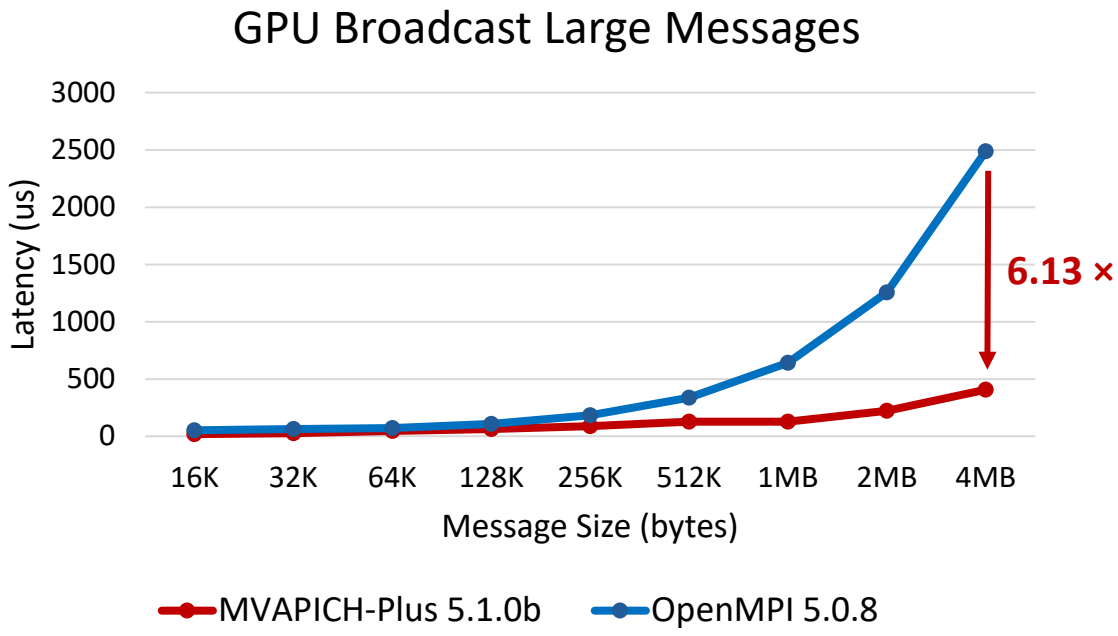
- Used for point-to-point and collective fine-tuning in isolation
  - How do fine-tuning point to point and collectives at the OMB level translate to application performance?
- Flagship benchmark: High-Performance Linpack (HPL)
  - LU Decomposition for solving Ax=b
- HPL is highly sensitive to broadcast

Application Fine-Tuning Considerations for HPL (Applicable to Other HPC and AI Applications)

- When to switch between Eager/Rendezvous?
- Optimal application-specific Broadcast algorithm choice?
  - Based on calculation of application message ranges
- When to leverage CPU<->GPU staging versus GPU IPC mechanisms?
- MPI Process Placement, transport selection type (RC/UD/CUDA_IPC/GDR_copy/etc.)?
- At what point do GPU-specific hardware (NVLink/InfinityFabric, etc.) get utilized?
- What process placement type is optimal for the application (is it latency-sensitive, bandwidth sensitive, etc.)?

Improved GPU Broadcast Performance Comparison (16 nodes/4 PPN)



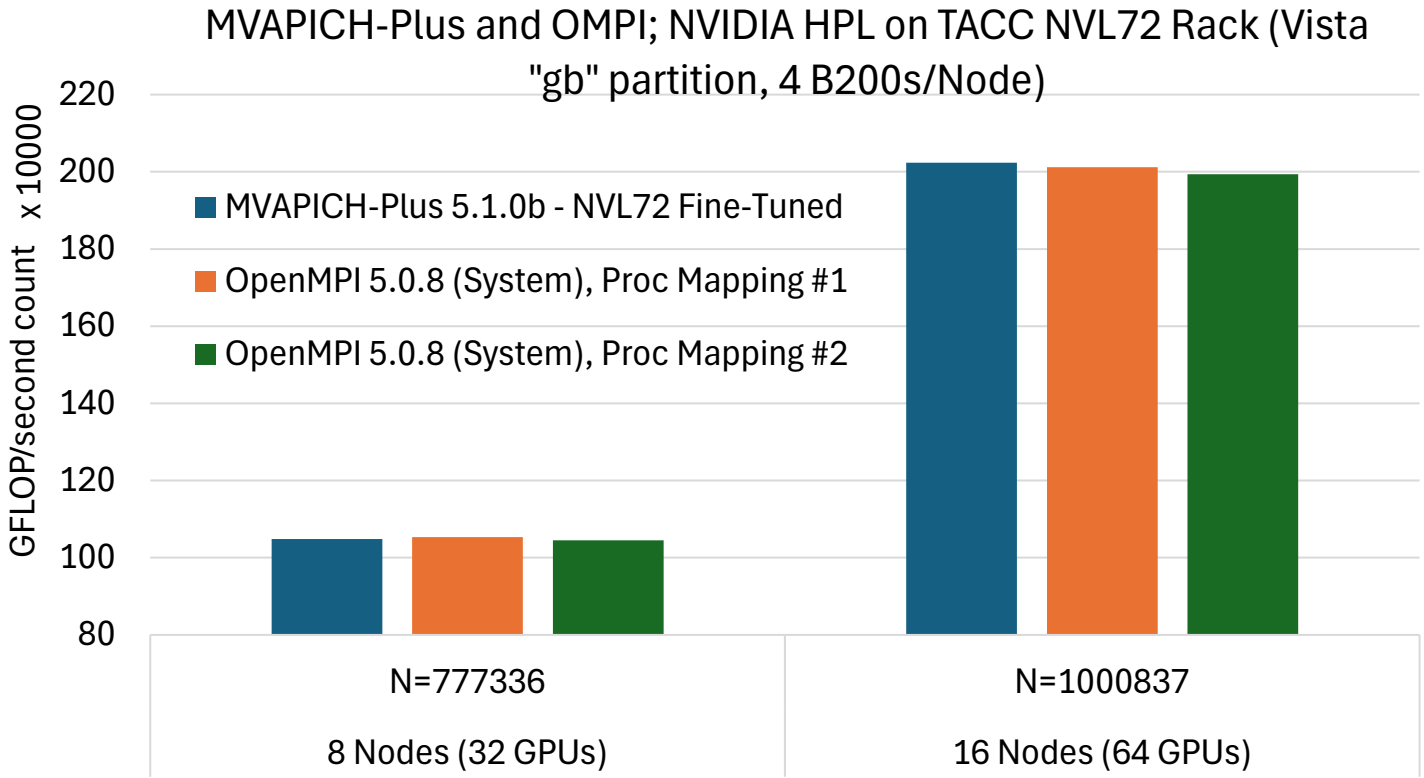


- For small message sizes, MVAPICH-Plus is 5-10 us faster than OpenMPI (40%)
- For large message sizes, the gap widens as message size increases, reaching up to 6.13x speedup at 4MB

Improved HPL 8 and 16 Node (4 B200/Node) Numbers

- OpenMPI Proc Mapping 1: Spread-based binding, per-NUMA mapping
- OpenMPI Proc Mapping 2: Bunch-based binding, per-NUMA mapping

- Up to 1.47% GFLOPS improvement at 16 Nodes
- Getting closer to R_{max} of NVL72 rack (2.073 TFLOPS for 16 Nodes)
- Further Enhancements to come



Contents

- CPU Process Mapping with MVAPICH
  - Summary of parallel launchers
- CH4 devices and their impact
  - UCX, OFI, enhanced libfabrics from OSU
- Dynamic Tuning
  - CPU, GPU dynamic tuning
- GPU-Awareness
  - NVIDA, AMD, Intel GPU support
- Application Case Study: High-Performance Linpack (HPL) on NVIDIA NVL72
- Future Plans

OSU Microbenchmarks (OMB) Version 8.0

- Latest 8.0rc release at https://mvapich.cse.ohio-state.edu/benchmarks/
- Total refactor of MPI benchmarks for easier contributions
  - Reduced files, less duplicated code, easier patches
- Enhanced runtimes to facilitate easier MPI library testing
  - Similar to IMB all-in-one tests
  - Run multiple tests from one MPI launch command
    - `mpiexec -np <np> … ./omb_coll [benchmark1[,benchmark2,...]]>
- Remains backwards compatible

MVAPICH 5.x.y enhancements

- Further shared-memory enhancements
- Enhanced GPU-aware collectives
- Integration of GPU-initiated communication
  - More details in Dr. Panda's Roadmap Talk on Tuesday

Funding Acknowledgments

Funding Support by

Equipment Support by

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

Current Students (Under/Graduate)

- N. Alnaasan (Ph.D.)
- Q. Anthony (Ph.D.)
- T. Chen (Ph.D.)
- S. Gumaste (Ph.D.)
- J. Hatef (Ph.D.)
- G. Kuncham (Ph.D.)
- S. Lee (Ph.D.)
- B. Michalowicz (Ph.D.)
- S. Mohammad (M.S.)
- T. Tran (Ph.D.)
- L. Xu (P.h.D.)
- S. Xu (Ph.D.)
- J. Wan (Ph.D.)
- J. Yao (Ph.D.)
- S. Zhang (Ph.D.)
- N. Klein (M.S.)

Current Research Specialist

- R. Motlagh

Current Software Engineers

- N. Shineman
- M. Lieber

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.)
- C.-C. Chen (Ph.D.)
- N. Contini (Ph.D.)
- N. Dandapanthula (M.S.)
- V. Dhanraj (M.S.)
- C.-H. Chu (Ph.D.)
- T. Gangadharappa (M.S.)
- K. Gopalakrishnan (M.S.)
- R. Gulhane (M.S.)
- J. Hashmi (Ph.D.)
- M. Han (M.S.)
- W. Huang (Ph.D.)
- A. Jain (Ph.D.)
- J. Jani (M.S.)
- W. Jiang (M.S.)
- J. Jose (Ph.D.)
- M. Kedia (M.S.)
- K. S. Khorassani (Ph.D.)
- S. Kini (M.S.)
- M. Koop (Ph.D.)
- P. Kousha (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.)
- J. Queiser (M.S.)
- K. Raj (M.S.)
- R. Rajachandrasekar (Ph.D.)
- B. Ramesh (Ph.D.)
- D. Shankar (Ph.D.)
- G. Santhanaraman (Ph.D.)
- N. Sarkauskas (B.S. and M.S)
- V. Sathu (M.S.)
- N. Senthil Kumar (M.S.)
- A. Singh (Ph.D.)
- J. Sridhar (M.S.)
- S. Srivastava (M.S.)
- H. Subramoni (Ph.D.)
- S. Sur (Ph.D.)
- K. K. Suresh (Ph.D.)
- K. Vaidyanathan (Ph.D.)
- A. Vishnu (Ph.D.)
- J. Wu (Ph.D.)
- W. Yu (Ph.D.)
- J. Zhang (Ph.D.)
- Q. Zhou (Ph.D.)
- N. Chmura (B.S.)

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 Research Scientists

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

Past Faculty

- H. Subramoni

Past Senior Research Associate

- J. Hashmi

Past Programmers

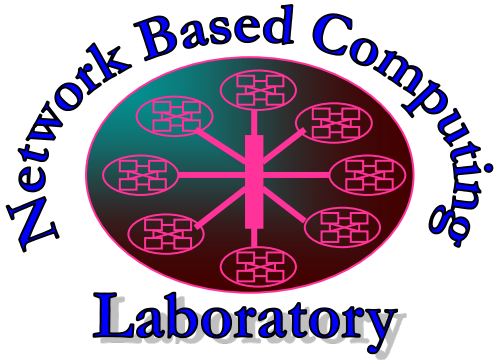
- A. Reifsteck
- D. Bureddy
- J. Perkins
- B. Seeds
- A. Guptha
- N. Pavuk

Past Research Specialist

- M. Arnold
- J. Smith

Thank You!

shineman.5@osu.edu, michalowicz.2@osu.edu



Follow us on

https://x.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 HPC-Accelerated AI Project

https://hpc-ai.engineering.osu.edu/