7TH INTERNATIONAL WORKSHOP ON EMERGING PARALLEL DISTRIBUTED RUNTIME SYSTEMS AND MIDDLEWARE

MPI Communication Performance on AMD MI300A: Microbenchmarks and Applications

Goutham Kalikrishna Reddy Kuncham, Ohio State University; Siyuan Zhang, Ohio State University; Shoaib Mohammad, Ohio State University; Chen-Chun Chen, Ohio State University; Dhabaleswar K. Panda, Ohio State University

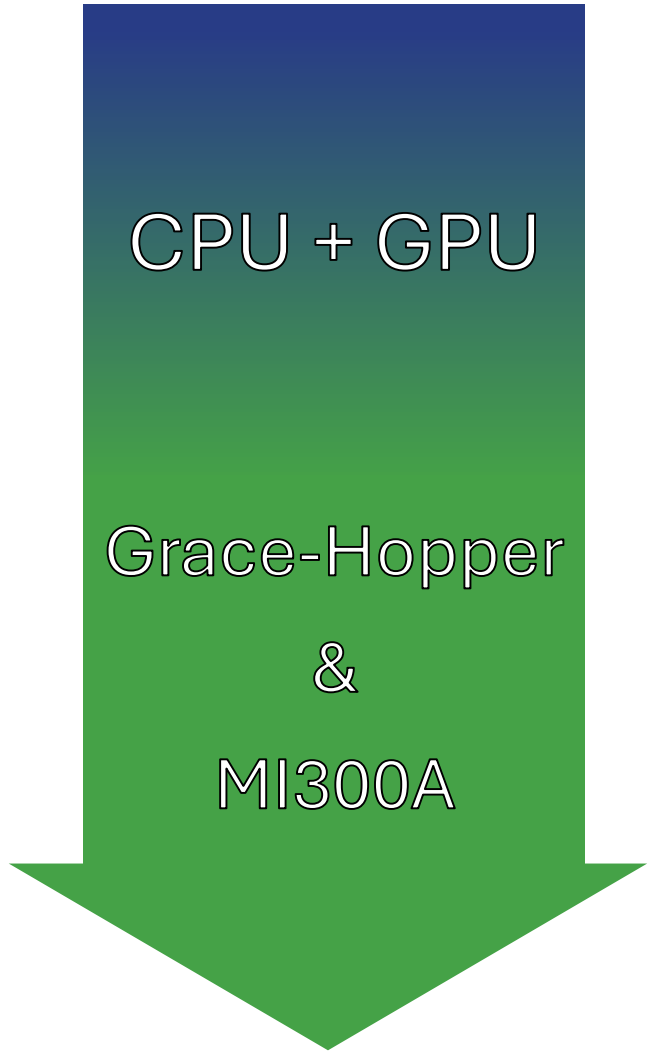
SC25 ST. LOUIS hpc ignites.

Outline

- Introduction
- Background
- Experimental Setup
- Performance Evaluation
- Conclusion & Future Work

HPC's Road of Heterogeneous Processor

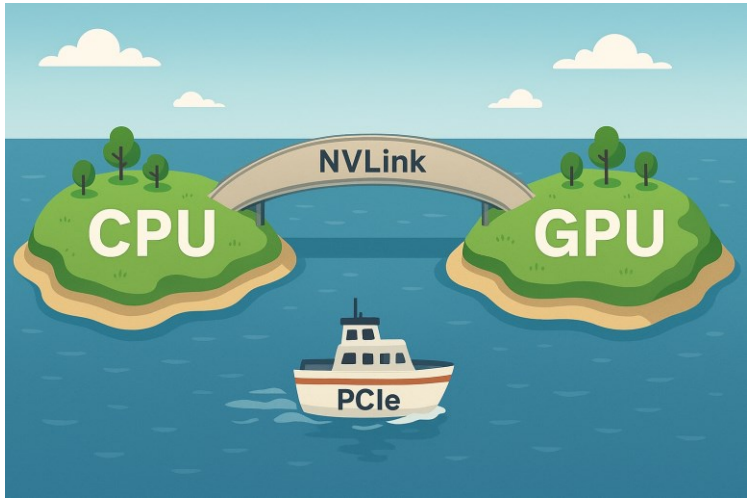
- **TSUBAME 1.2** (2008) – 1st GPGPU-powered Cluster
  - AMD Opteron Barcelona + NVIDIA Tesla T10
- **Titan** (2012) – #1 in TOP 500 GPGPU-powered Cluster
  - AMD Opteron 6274 + NVIDIA Tesla K20X
- **Alps** (2024) – 1st Grace-Hopper-powered Cluster
  - Grace 72 ARMv9 + Hopper H100 GPU (GH200)
- **El Capitan** (2025) - #1 in TOP 500
  - AMD MI300A
- **Green 500**
  - 4/10 Top 10 are using GH200 (#1, #2, #4, #8 @2025 June)
  - 2/10 Top 10 are using MI300A (#3, #9 @ 2025 June)



Outline

- Introduction
- Background
- Experimental Setup
- Performance Evaluation
- Conclusion & Future Work

Challenges: The "Two-Island" Problem



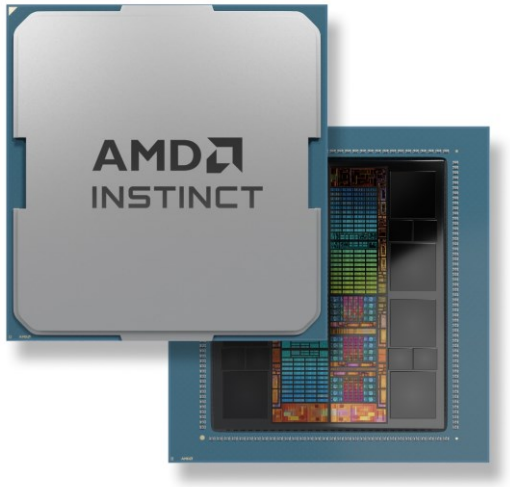
[1]

- **Separate Memory Pools**
  - CPU and GPU separate physical memory (DRAM ↔ HBM/GDDR).
- **The "Data Ferry": Expensive & Explicit Copies**
  - Data must be manually copied (`memcpy`) between host and device to be used.
  - This "ferry" trip consumes significant **time (latency)** and **energy**, reducing overall efficiency.
- **The "Bigger Bridge" Fallacy**
  - Faster interconnects (NVLink, Infinity Fabric) create a wider bridge, but don't solve the core issue.
  - The two memory islands still exist; copies and synchronization are still required.
- **The Burden on Software**
  - Developers and middleware must explicitly manage data location and traffic.
  - This results in higher code complexity, longer development time, and difficult performance trade-offs.

[1] Courtesy: Image generated by ChatGPT

AMD MI300A APU (Accelerated Processing Unit)

- First data-center APU MI300A integrating CPU and GPU cores within a single package
- MI300A combines x86 CPU cores, CDNA3 GPU compute units
- Features a unified, coherent pool of ultra-fast HBM3 memory accessible by both CPU and GPU cores.
- Eliminates explicit host-device data transfers
- Changes communication costs and middleware strategies

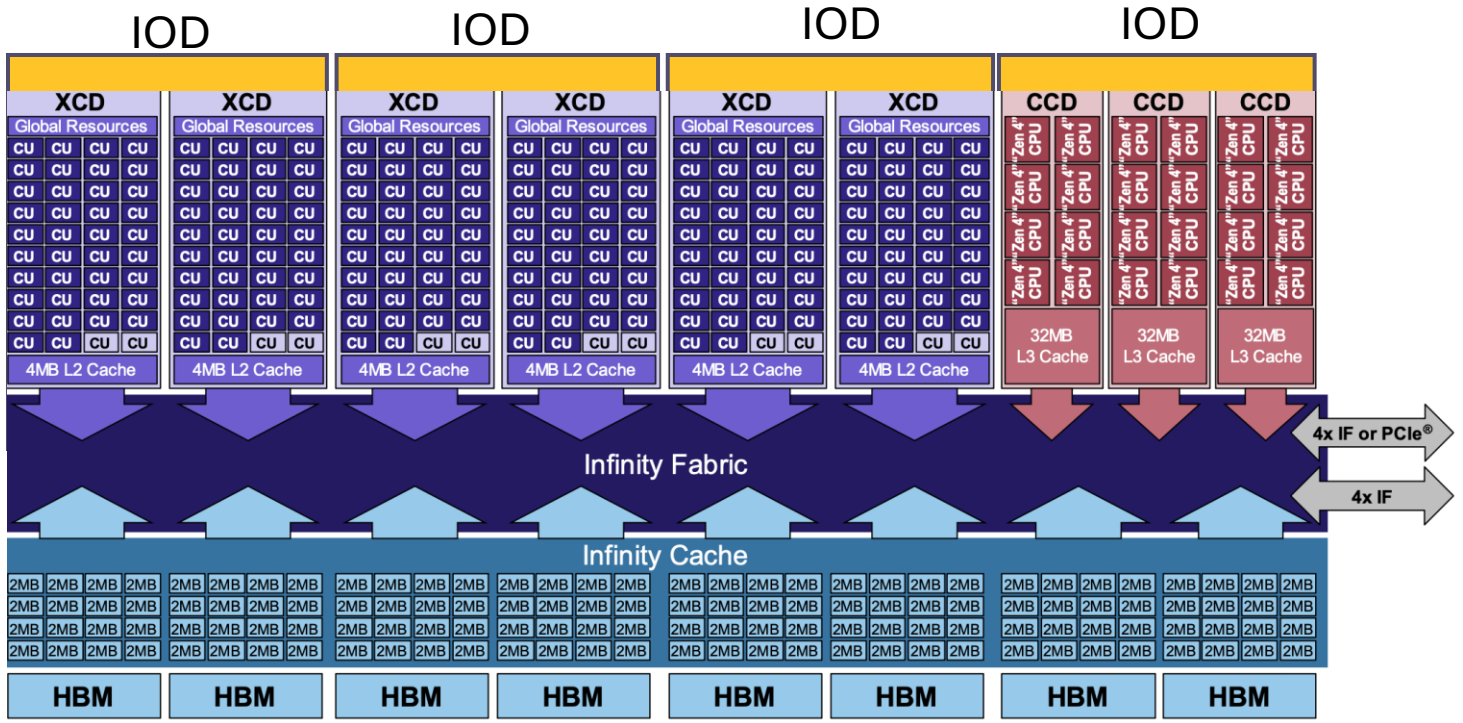


[1]

[1] Courtesy: AMD Instinct™ MI300A Accelerators from AMD website

MI300A Chiplet-Based Architecture Components

- **3** Core Complex Dies (CCDs), each with 8 high-performance Zen 4 CPU cores (24 total)
- **6** Accelerator Complex Dies (XCDs) delivering massive GPU compute (228 CDNA3 compute units)
- **4** I/O Dies (IODs) functioning as cached active interposers

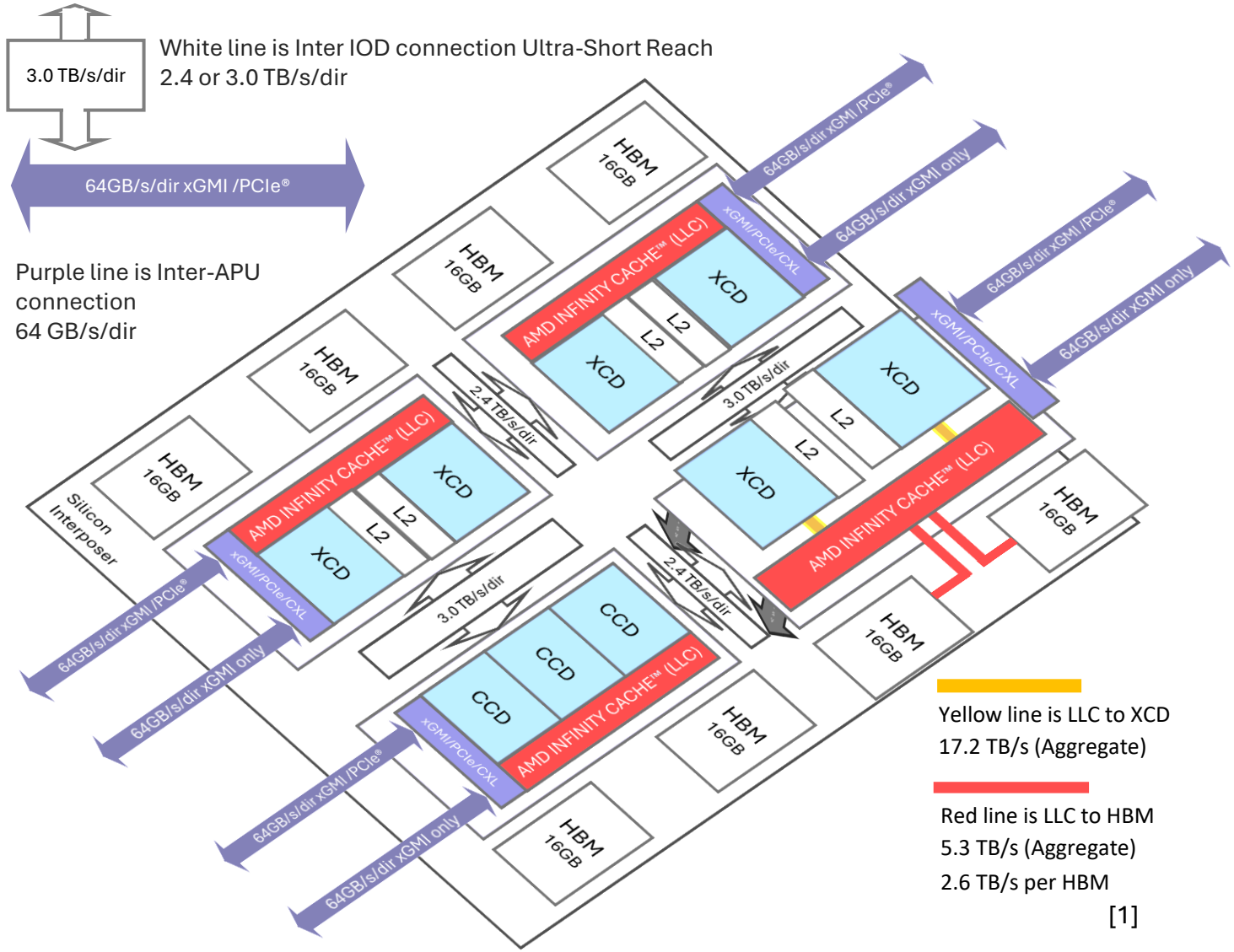


[1]

[1] Courtesy: A. Smith et al., "Realizing the AMD Exascale Heterogeneous Processor Vision : Industry Product," 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, 2024, pp. 876-889,

MI300A Chiplet-Based Architecture Components

- 256 MB Infinity Cache
- Peak theoretical bandwidth from the Infinity Cache is 17.2 TB/s
- 8 x 16GB HBM3 memory across 4 IODs (Total of 128GB)
- 5.3 TB/s peak theoretical HBM bandwidth

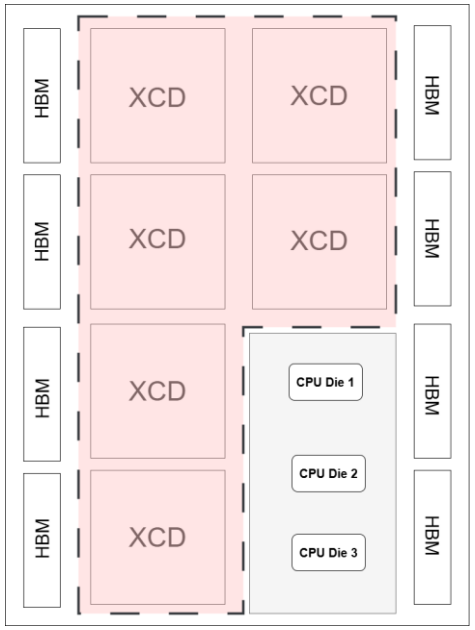


[1] Courtesy: A. Smith et al., "Realizing the AMD Exascale Heterogeneous Processor Vision : Industry Product," 2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, 2024, pp. 876-889,

MI300A Partition Modes

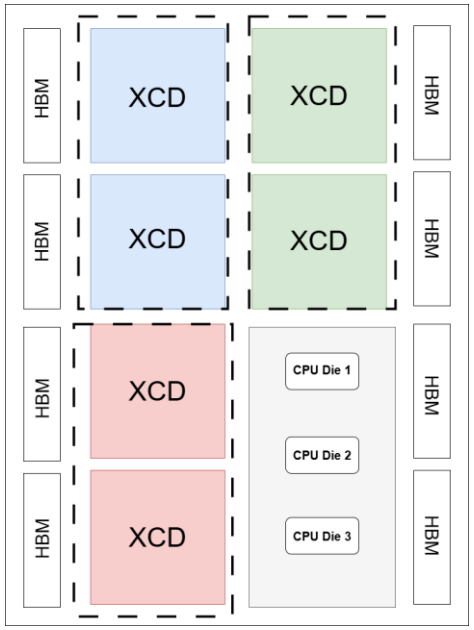
- SPX (Single Partition X-celerator):
  - Default Mode for MI300A platforms.
  - Combines all six XCDs into a single logical GPU device.
  - Best for unified, large-scale workloads
- TPX (Triple Partition X-celerator):
  - Splits the GPU into three partitions, each with 2 XCDs.
  - Ideal for batch scheduling and running multiple medium-size applications
- CPX (Core Partitioned X-celerator):
  - Treats each XCD as its own logical device, producing 6 independent GPU partitions.
  - Suitable for multi-user scenarios, task-level isolation, and fine-grained scheduling.

1 Logical GPU
228 CUs
128 GB HBM



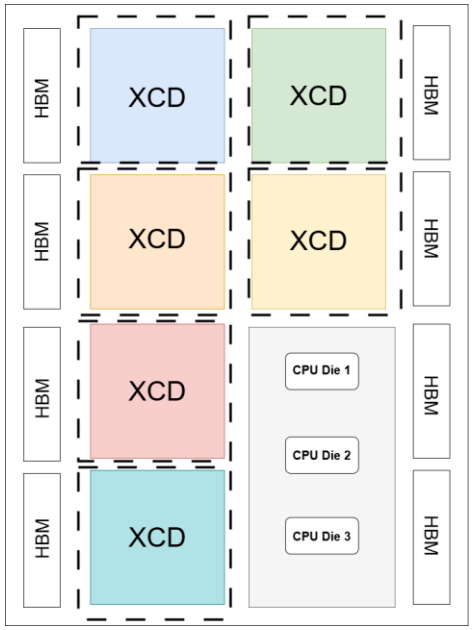
SPX

3 Logical GPUs
76 CUs
~ 32 GB HBM



TPX

6 Logical GPUs
38 CUs
~ 16 GB HBM



CPX

[1]

[1] Courtesy: https://instinct.docs.amd.com/projects/amdgpu-docs/en/latest/gpu-partitioning/mi300a/overview.html

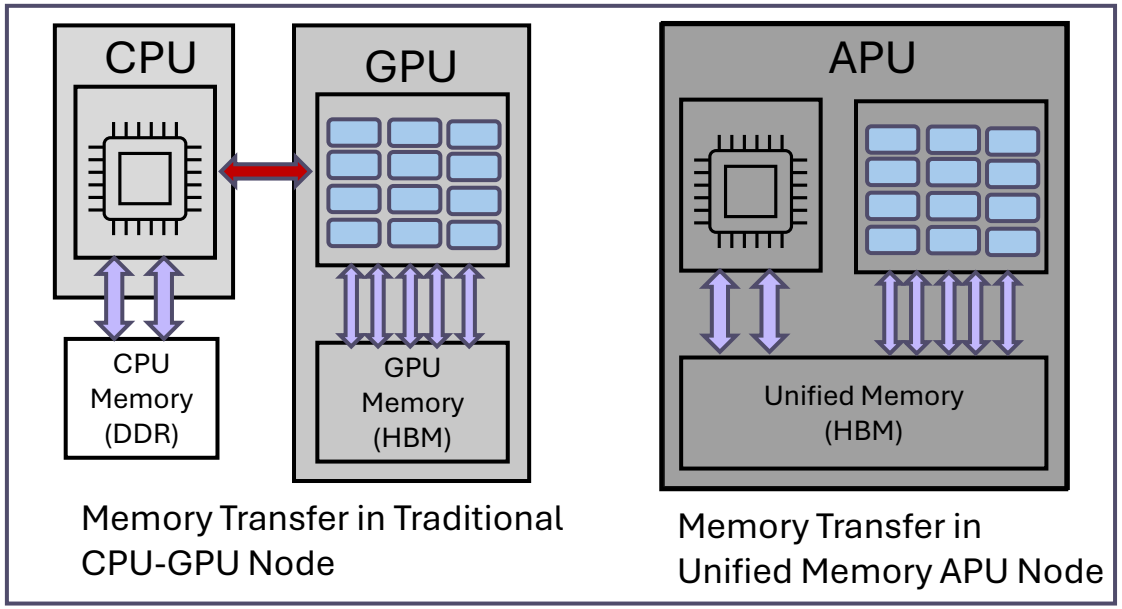
XNACK in MI300A APU Architecture

What is XNACK (Translation Not-Acknowledged)?

- XNACK enables the GPU to handle page faults by retrying memory accesses that initially fail due to missing pages, instead of terminating with an error.

Why XNACK matters in MI300A?

- Allows seamless access to host-allocated memory within the unified HBM3 memory pool shared by CPU and GPU, eliminating the need for explicit data migration.
- Without XNACK enabled on the MI300A APU, GPU accesses to host-allocated memory would incur costly page faults and fallback transfers, significantly degrading performance and increasing latency.



To Enable XNACK:

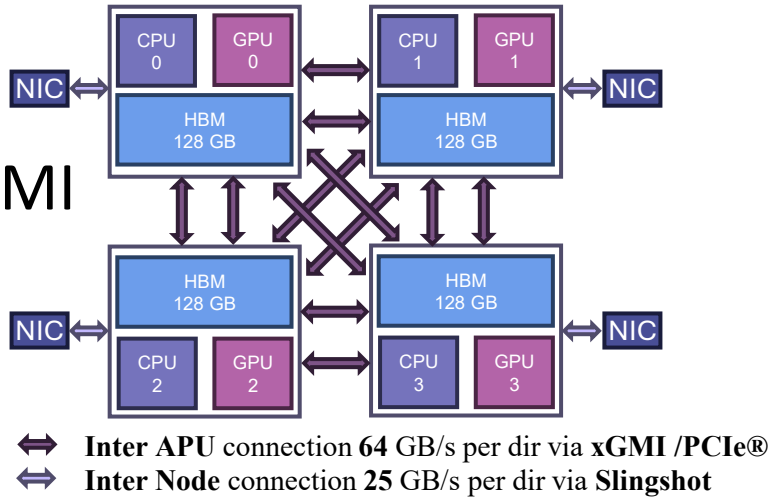
```
set HSA_XNACK=1
```

Outline

- Introduction
- Background
- Experimental Setup
- Performance Evaluation
- Conclusion & Future Work

SDSC COSMOS

- 42 nodes, each with 4 APUs total of 168 APUs
- Within the node, 4 APUs are interconnected by a fully connected network based on AMD's Infinity fabric and xGMI technology
- Each xGMI link is capable of 64 GB/s bandwidth in each direction.
- Interconnect delivers up to 256 GB/s (64 x 2 x 2) peer-to-peer bi-directional bandwidth and up to 768 GB/s (64 x 2 x 6) aggregate bi-directional bandwidth in a node.
- Every node is provisioned with four HPE Cray Slingshot-11 interconnects, offering an aggregate bidirectional bandwidth of 200 GB/s (equivalent to 25GB/s per link in each direction).
- All nodes in our experiments are in the SPX mode.



Software Details

MPI & Communication Libraries

- **Cray MPICH** 8.1.30
  - https://docs.nersc.gov/development/programming-models/mpi/cray-mpich/
- **MPICH** 4.3.1
  - https://github.com/pmodels/mpich/releases/tag/v4.3.1
- **MVAPICH-Plus** 4.1
  - https://mvapich.cse.ohio-state.edu
- **OpenMPI** 5.0.8
  - https://www.open-mpi.org

OSU Microbenchmarks 7.5

- https://mvapich.cse.ohio-state.edu/benchmarks/

OpenFOAM 2412 + GCC 12.3

- https://openfoam.org/

NanoGPT

- https://github.com/karpathy/nanoGPT

ROCm version 6.3.0

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-4.1 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 3,475 organizations in 93 countries (listed under the Users Tab of the MVAPICH page)**
- **More than 1.97 Million downloads from the OSU site directly**
- Empowering many TOP500 clusters (Jun'25 ranking)
  - 21st, 10,649,600-core (Sunway TaihuLight) at NSC, Wuxi, China
  - 67th, 448, 448 cores (Frontera) at TACC
  - 88th, 288,288 cores (Lassen) at LLNL
  - 109th, 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 67th ranked TACC Frontera system
- **Empowering Top500 systems for more than 20+ years**

Outline

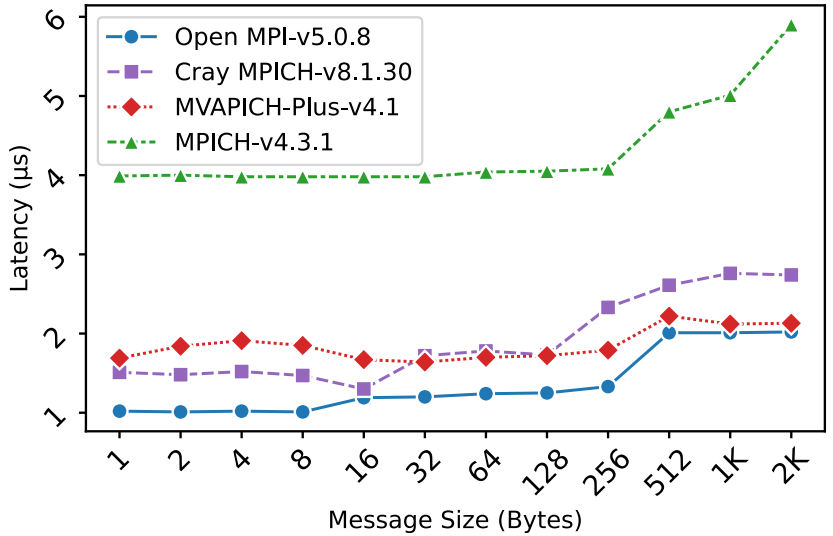
- Introduction
- Background
- Experimental Setup
- Performance Evaluation
- Conclusion & Future Work

Performance Evaluation

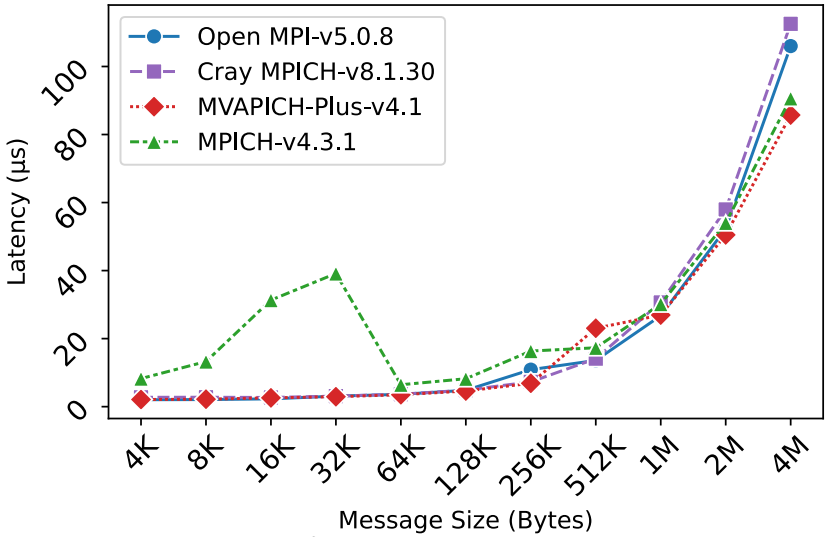
Point-to-Point

Point-to-Point Performance – Intra-Node CPU

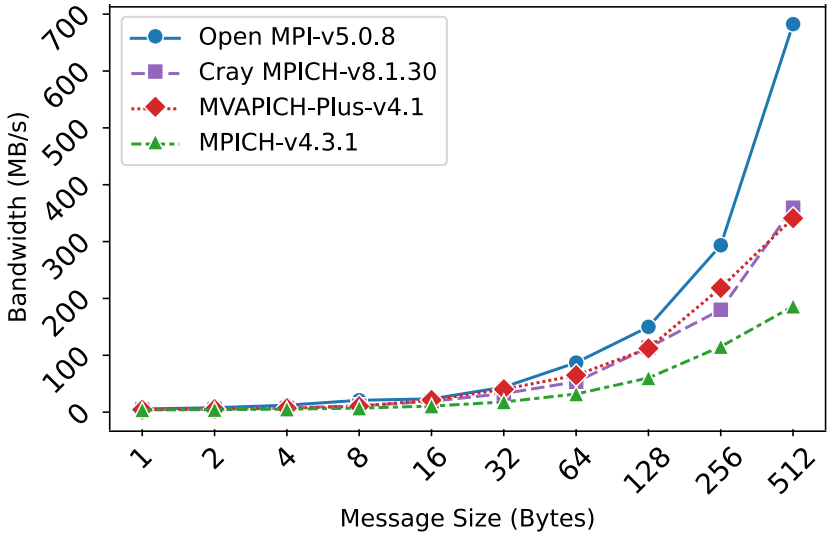
Latency (small messages)



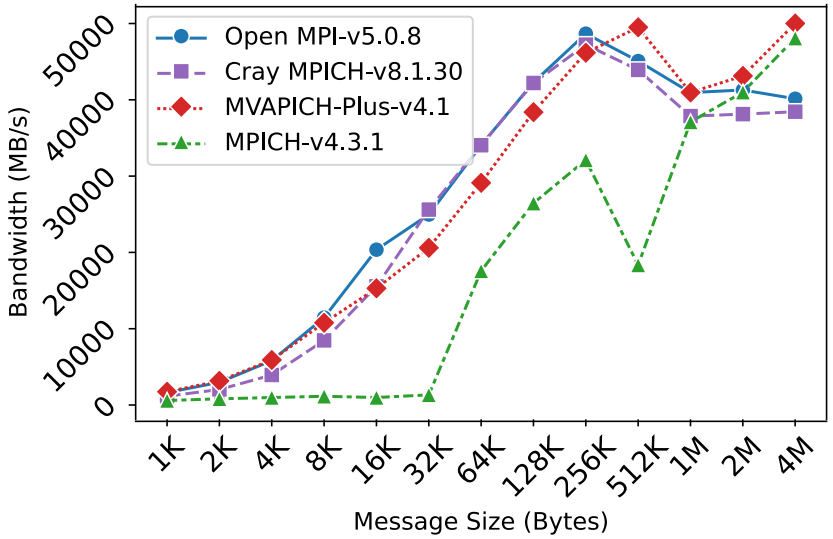
Latency (large messages)

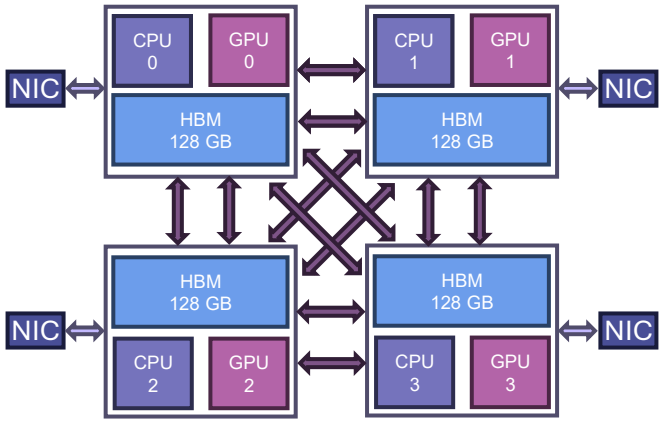


Bandwidth (small messages)



Bandwidth (large messages)





⟷ **Inter APU** connection **64** GB/s per dir per link via **xGMI/PCIe**
⟺ **Inter Node** connection **25** GB/s per dir via **Slingshot**

Peak Bandwidth achieved:

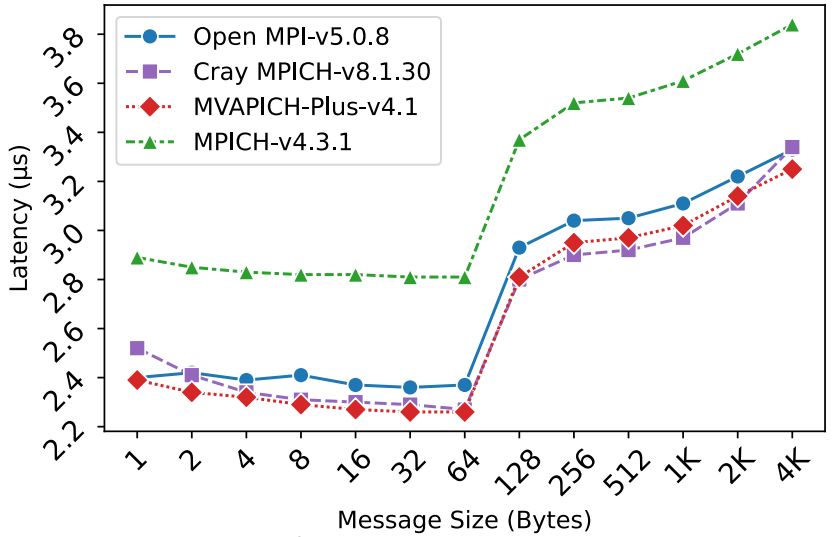
- **Open MPI** v5.0.8 47.26GB/s
- **Cray MPICH** v8.1.30 45.86GB/s
- **MVAPICH-Plus** v4.1 48.57GB/s
- **MPICH** v4.3.1 46.50GB/s

Latency at 1 Byte:

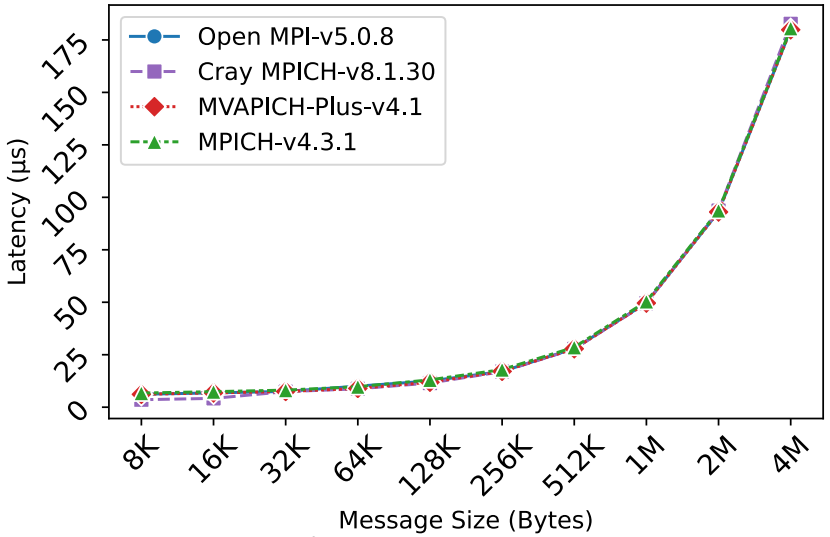
- **Open MPI** v5.0.8 1.02 µs
- **Cray MPICH** v8.1.30 1.51 µs
- **MVAPICH-Plus** v4.1 1.69 µs
- **MPICH** v4.3.1 3.99 µs

Point-to-Point Performance – Inter-Node CPU

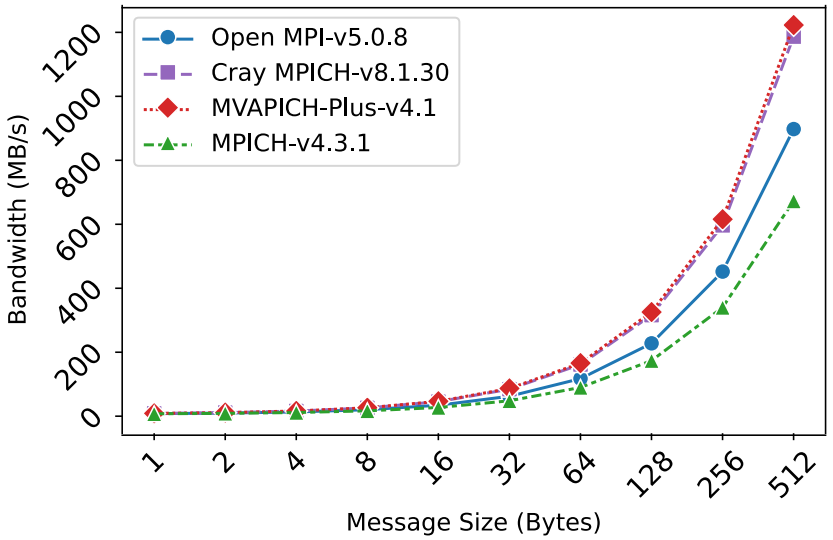
Latency (small messages)



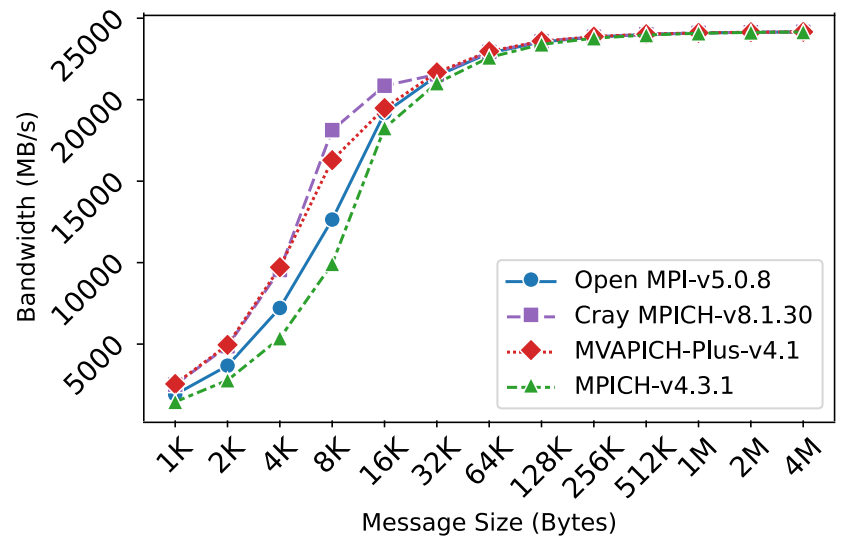
Latency (large messages)

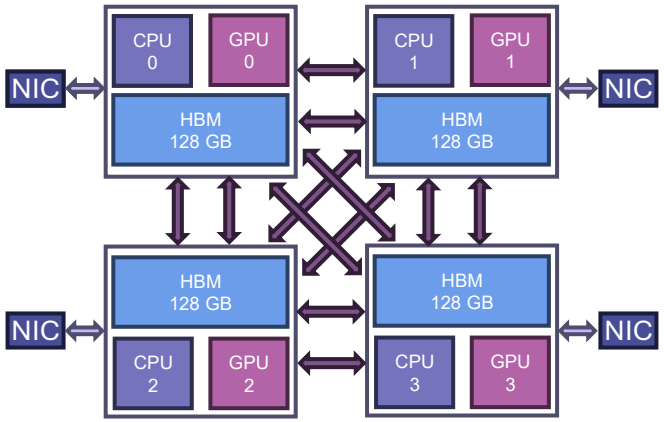


Bandwidth (small messages)



Bandwidth (large messages)





⟷ **Inter APU** connection **64** GB/s per dir per link via **xGMI/PCIe**
⟺ **Inter Node** connection **25** GB/s per dir via **Slingshot**

Peak Bandwidth achieved:

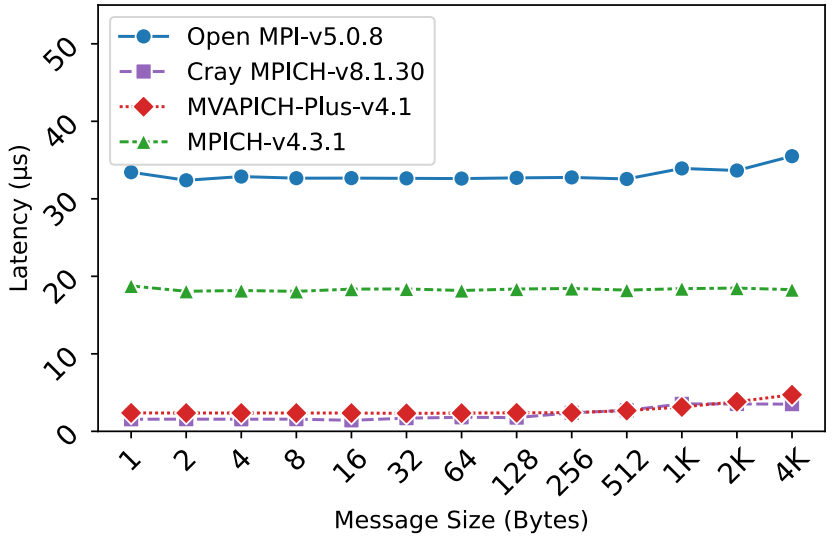
- **Open MPI** v5.0.8 23.472GB/s
- **Cray MPICH** v8.1.30 23.474GB/s
- **MVAPICH-Plus** v4.1 23.475GB/s
- **MPICH** v4.3.1 23.467GB/s

Latency at 1 Byte:

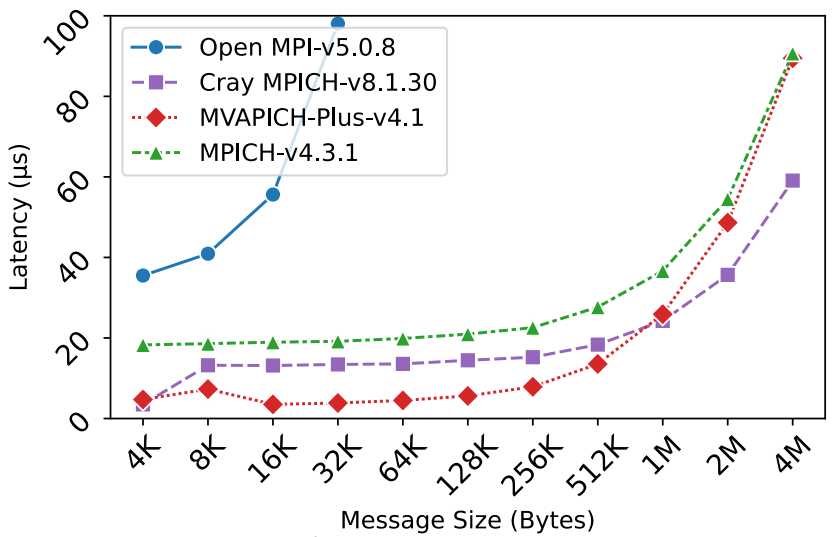
- **Open MPI** v5.0.8 2.40 µs
- **Cray MPICH** v8.1.30 2.52 µs
- **MVAPICH-Plus** v4.1 2.39 µs
- **MPICH** v4.3.1 2.89 µs

Point-to-Point Performance – Intra-Node GPU

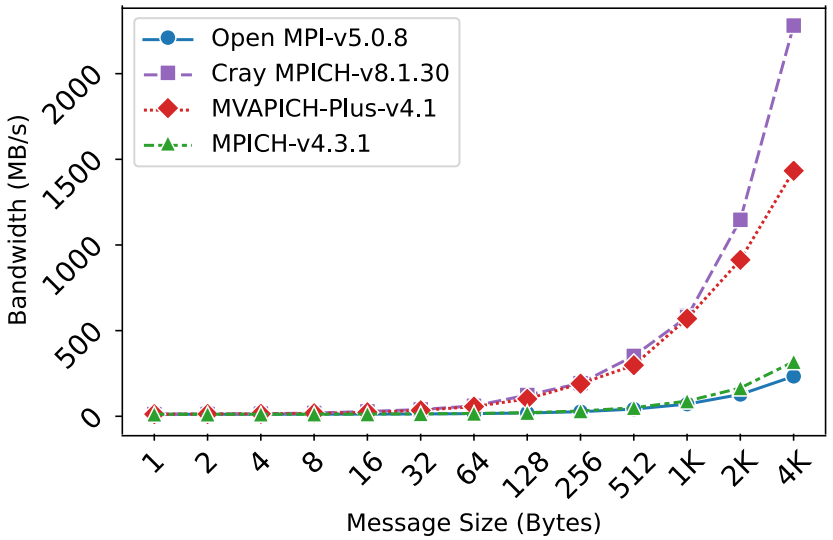
Latency (small messages)



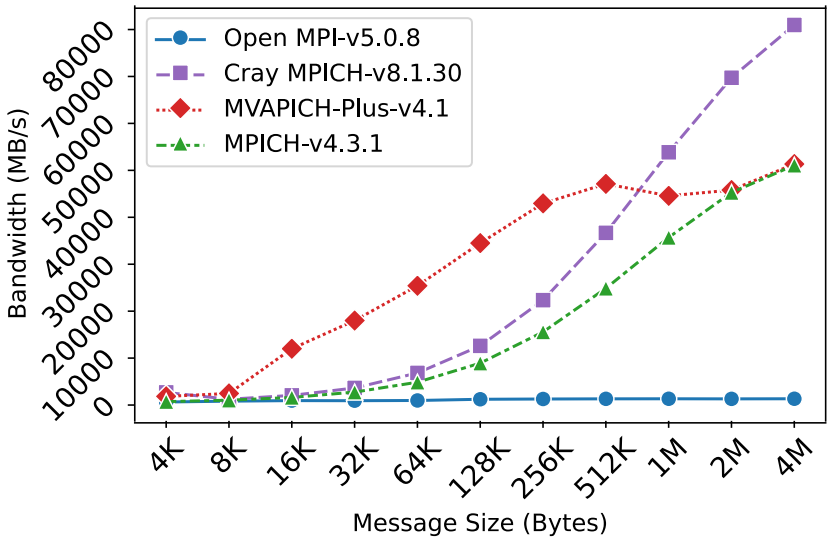
Latency (large messages)

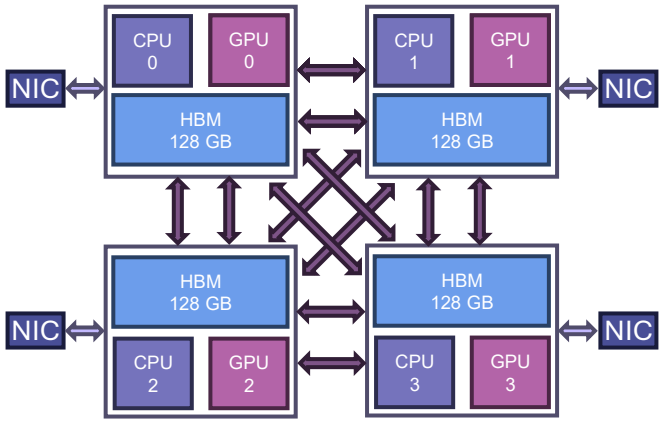


Bandwidth (small messages)



Bandwidth (large messages)





⟺ **Inter APU** connection **64** GB/s per dir per link via **xGMI/PCIe**
⟺ **Inter Node** connection **25** GB/s per dir via **Slingshot**

Peak Bandwidth achieved:

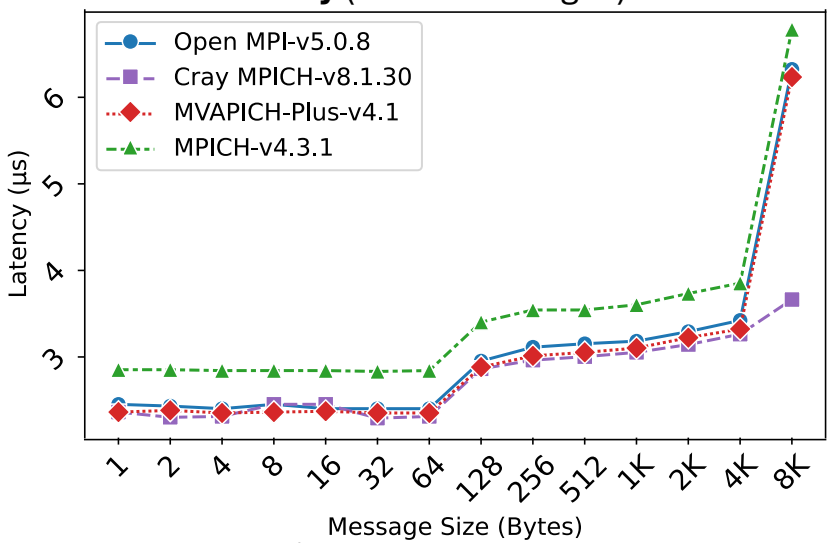
- **Open MPI** v5.0.8 0.89GB/s
- **Cray MPICH** v8.1.30 78.64GB/s
- **MVAPICH-Plus** v4.1 59.72GB/s
- **MPICH** v4.3.1 49.47GB/s

Latency at 1 Byte:

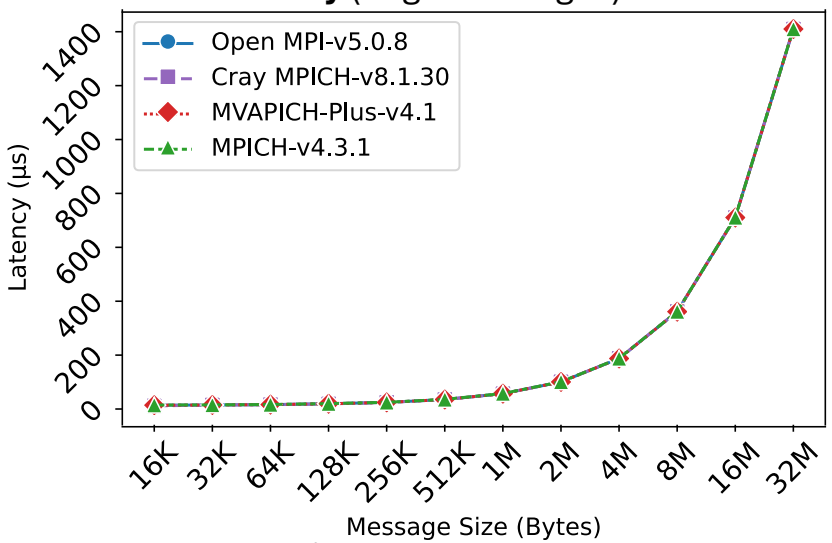
- **Open MPI** v5.0.8 33.43 µs
- **Cray MPICH** v8.1.30 1.56 µs
- **MVAPICH-Plus** v4.1 2.38 µs
- **MPICH** v4.3.1 18.76 µs

Point-to-Point Performance – Inter-Node GPU

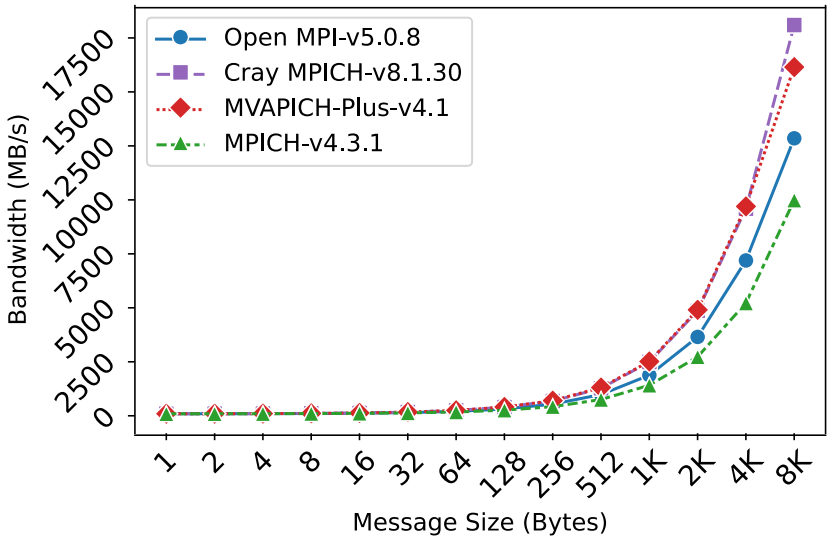
Latency (small messages)



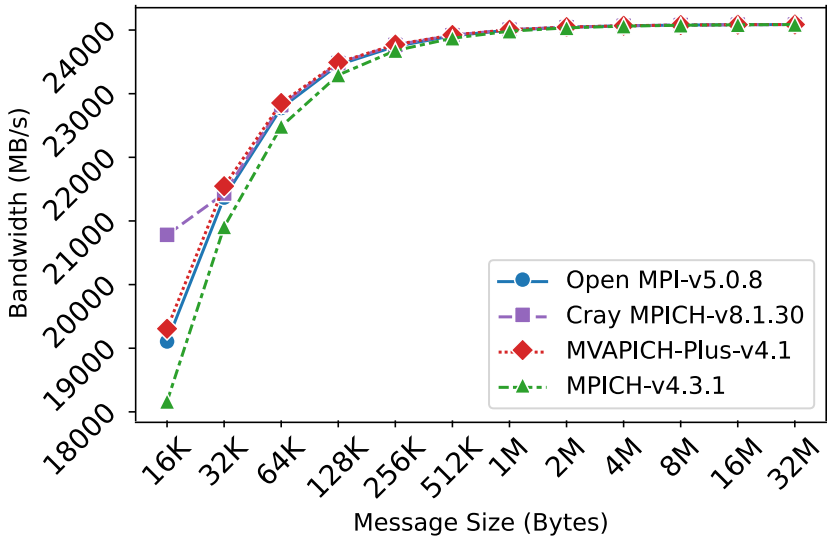
Latency (large messages)

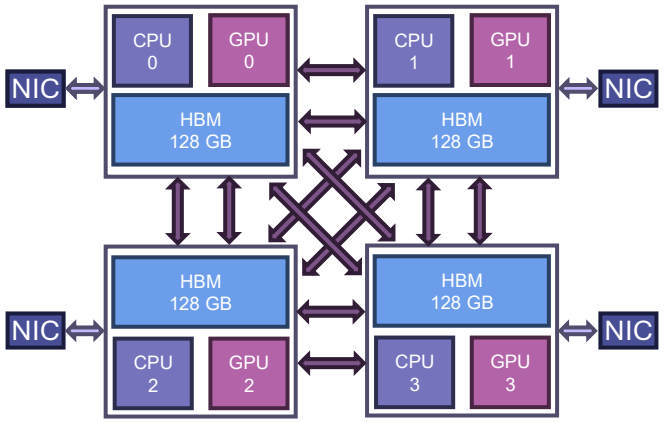


Bandwidth (small messages)



Bandwidth (large messages)





⟷ **Inter APU** connection **64** GB/s per dir per link via **xGMI/PCIe**
⟺ **Inter Node** connection **25** GB/s per dir via **Slingshot**

Peak Bandwidth achieved:

- **Open MPI** v5.0.8 23.49GB/s
- **Cray MPICH** v8.1.30 23.49GB/s
- **MVAPICH-Plus** v4.1 23.49GB/s
- **MPICH** v4.3.1 23.49GB/s

Latency at 1 Byte:

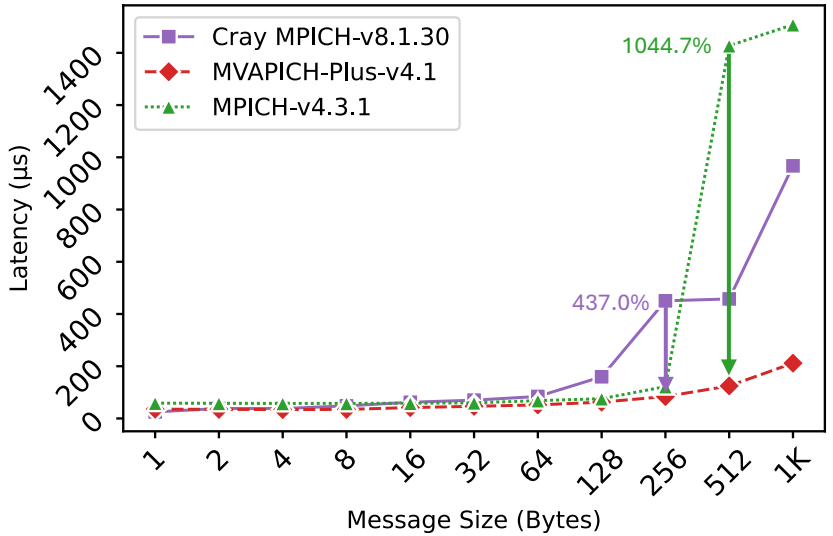
- **Open MPI** v5.0.8 2.43 µs
- **Cray MPICH** v8.1.30 2.34 µs
- **MVAPICH-Plus** v4.1 2.34 µs
- **MPICH** v4.3.1 2.83 µs

Performance Evaluation

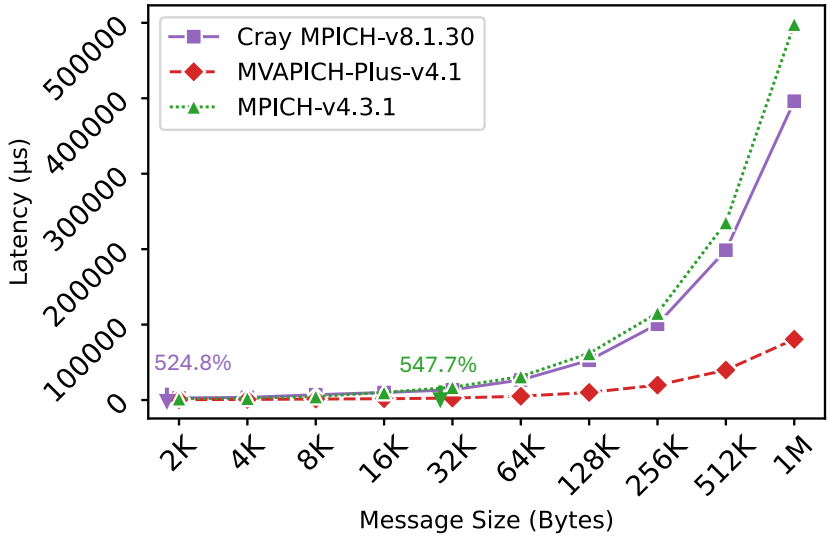
Collectives

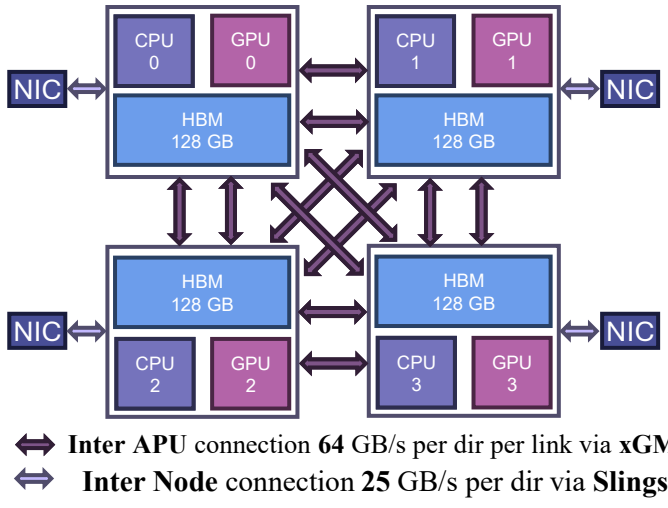
Collective Performance – 1 Node CPU (192PPN)

Allgather Latency (small messages)

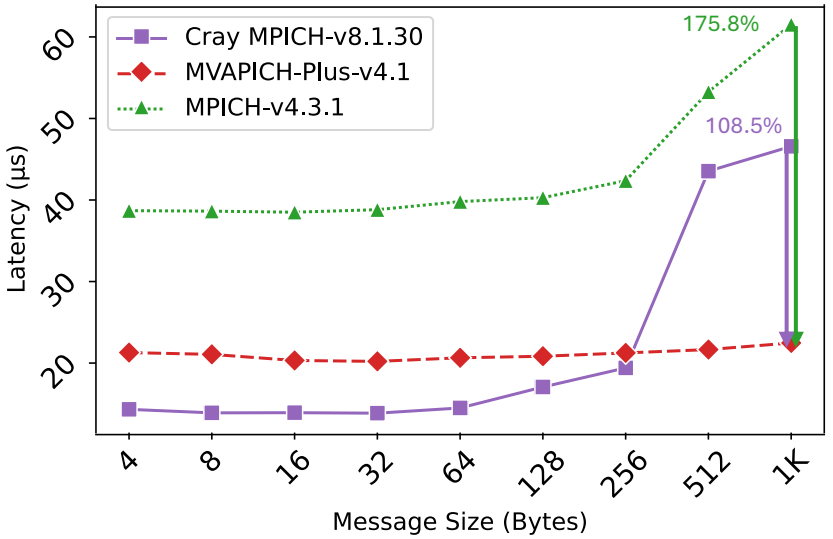


Allgather Latency (large messages)

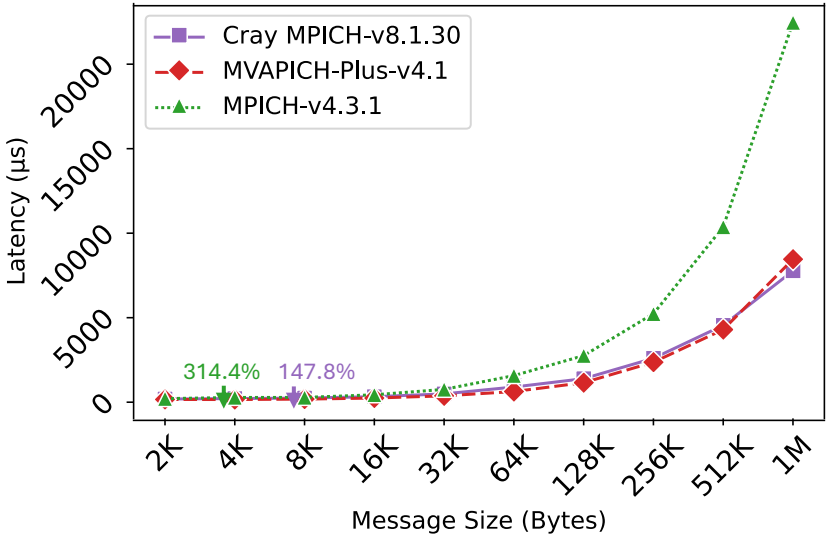




Allreduce Latency (small messages)



Allreduce Latency (large messages)

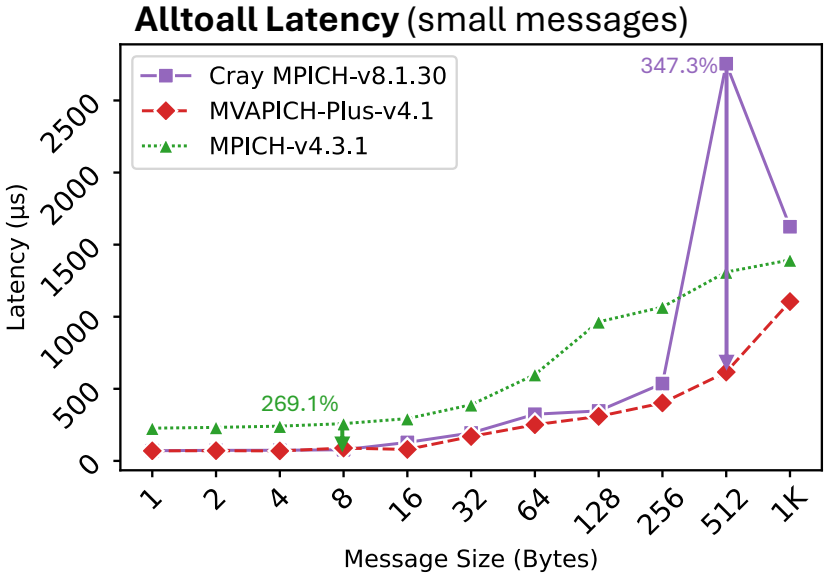


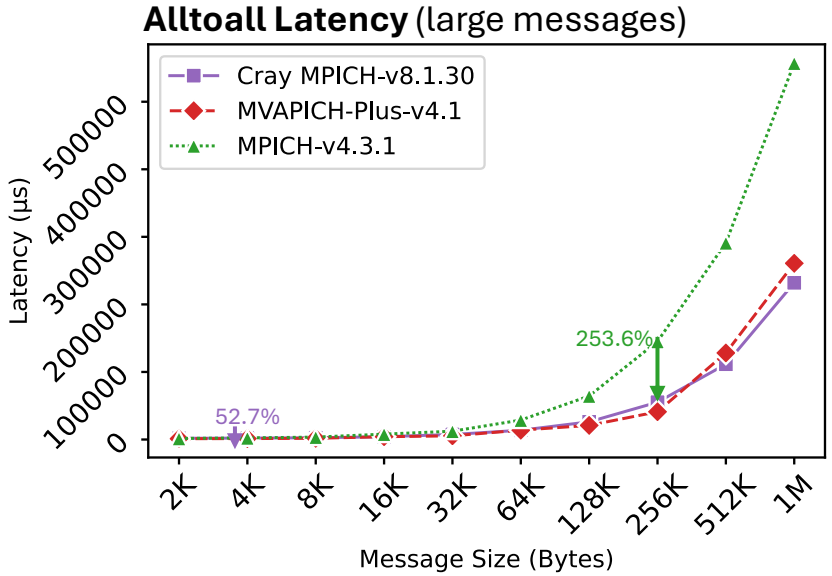
Allgather - MVAPICH-Plus outperforms MPICH by 547% at 32 KB and Cray MPICH by 524% at 2KB.

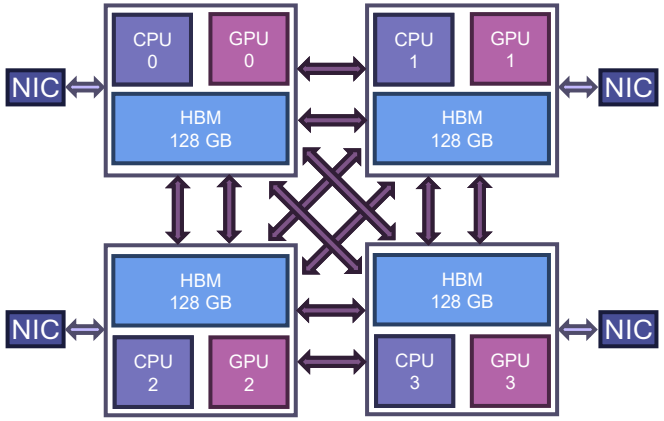
Allreduce - MVAPICH-Plus outperforms MPICH by 314% at 4KB and Cray MPICH by 147% at 8KB

*Open MPI numbers are omitted due to a segmentation fault in this system

Collective Performance – 1 Node CPU (192PPN)



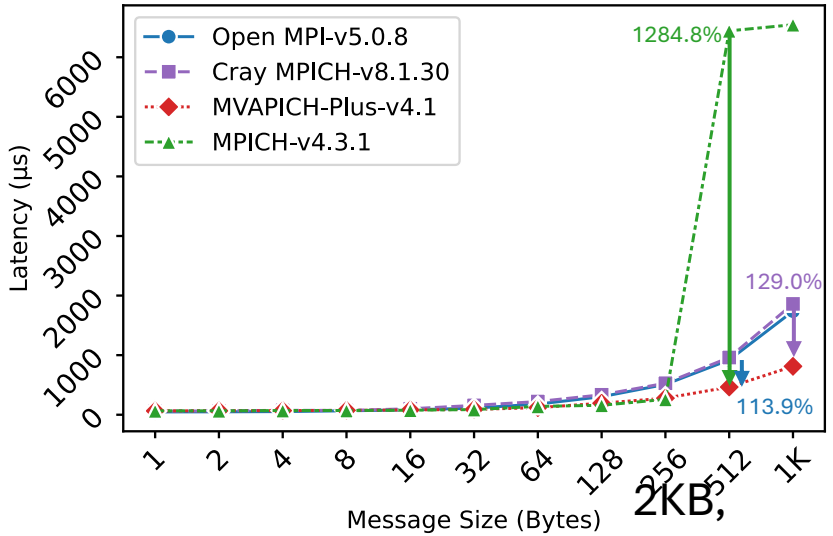




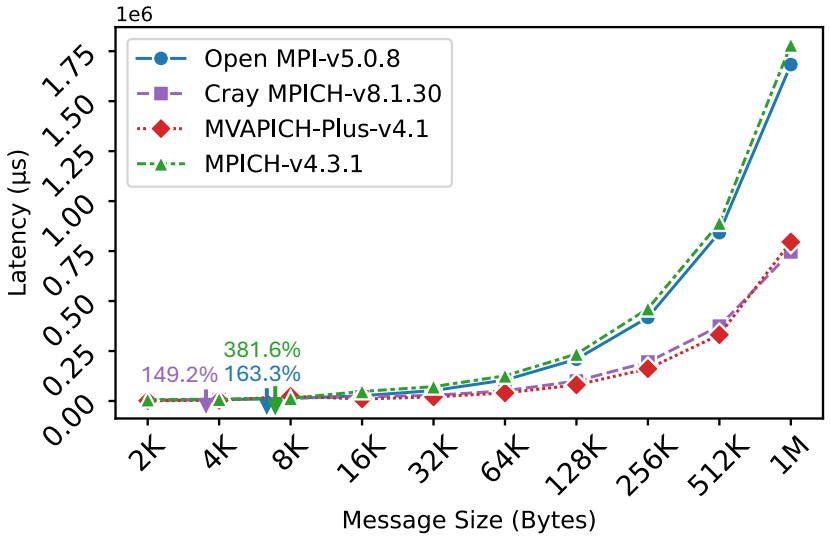
Alltoall - MVAPICH-Plus outperforms MPICH by 253% at 256 KB and Cray MPICH by 52% at 4KB.

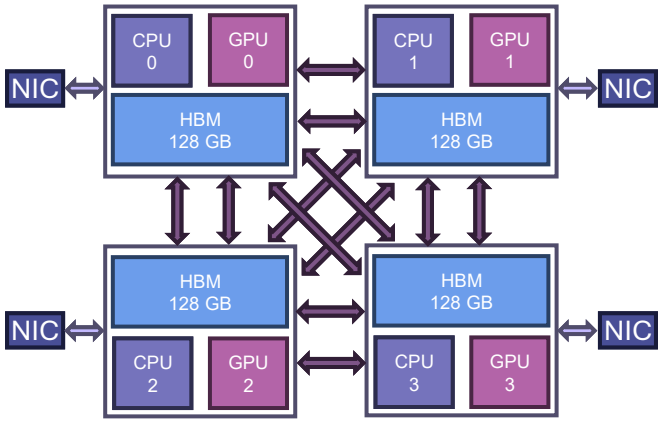
Collective Performance – 16 Nodes CPU (64PPN)

Allgather Latency (small messages)



Allgather Latency (large messages)

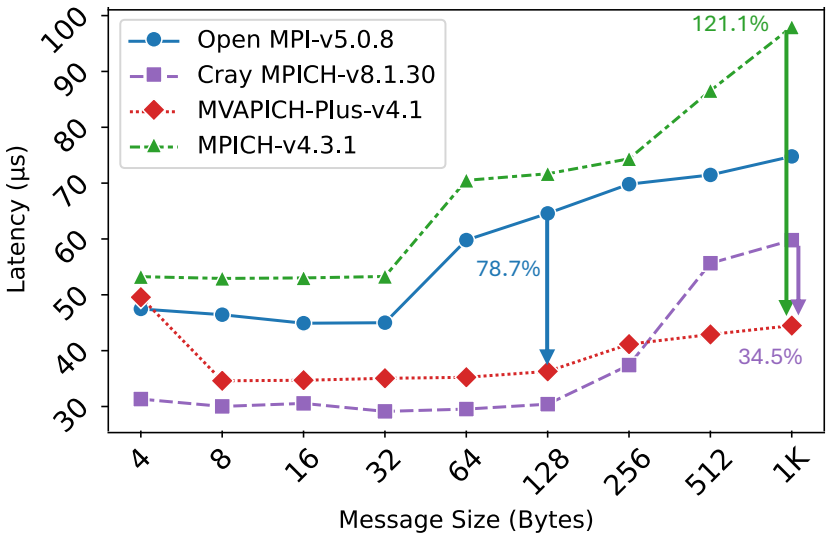




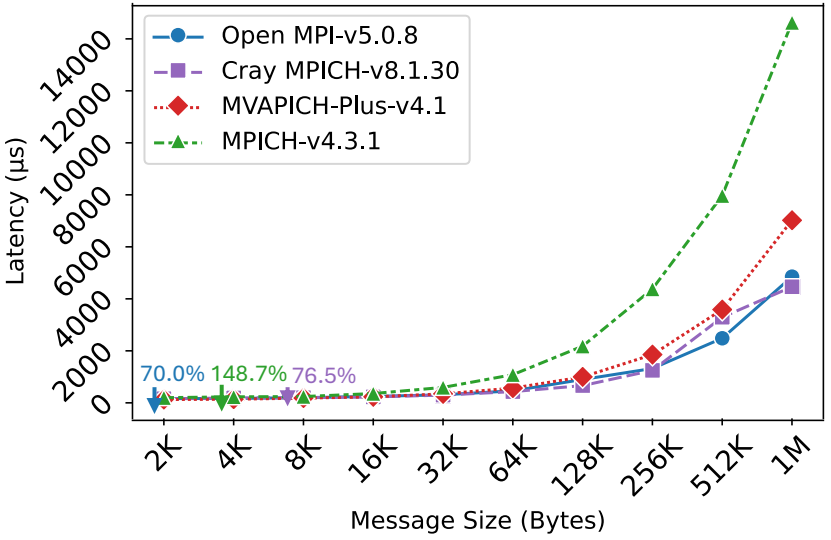
Inter APU connection **64** GB/s per dir per link via **xGMI/PCIe**

Inter Node connection **25** GB/s per dir via **Slingshot**

Allreduce Latency (small messages)



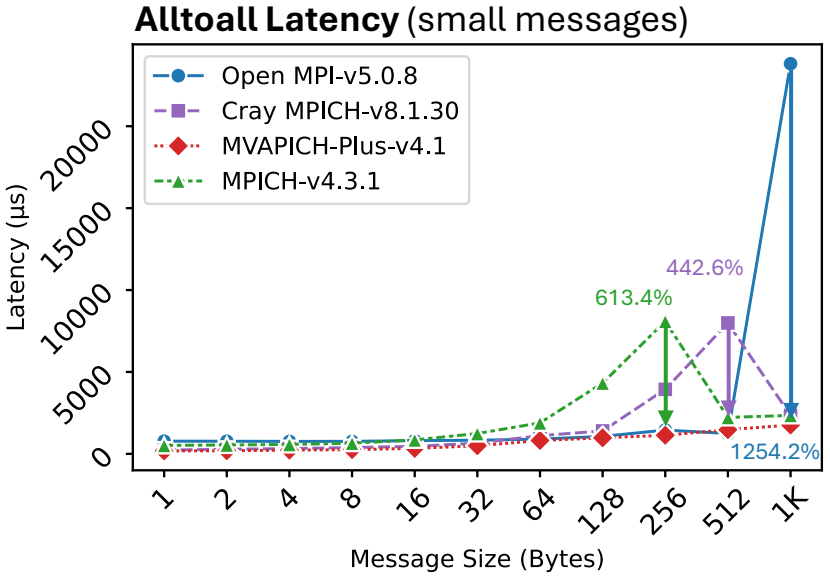
Allreduce Latency (large messages)

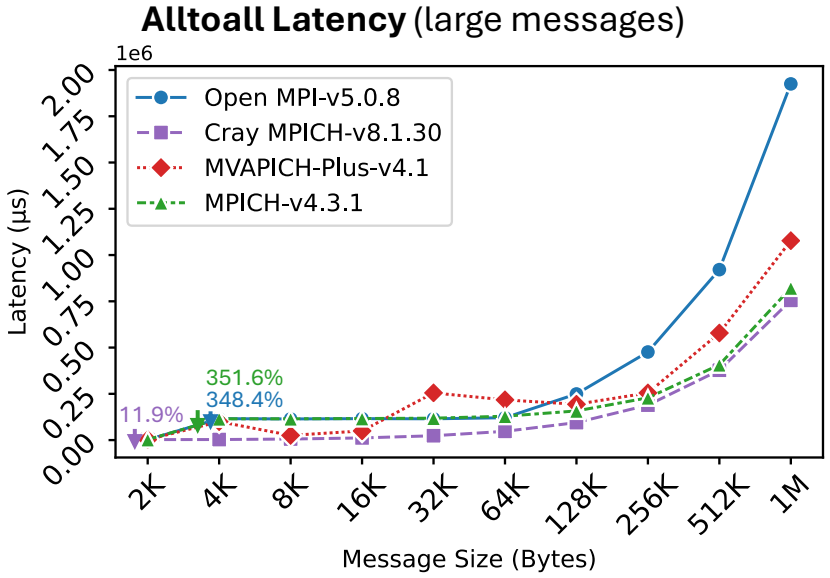


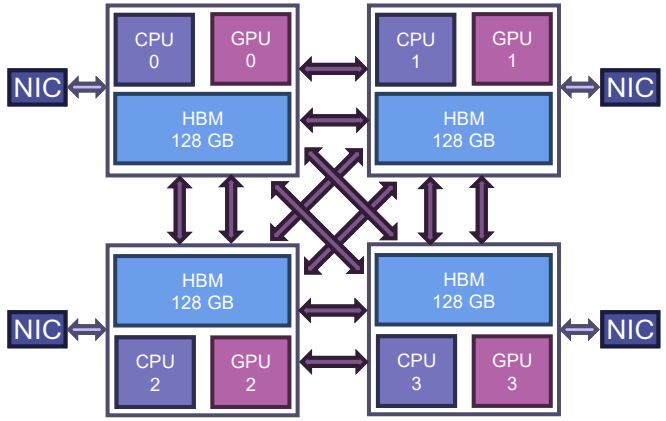
Allgather - MVAPICH-Plus outperforms MPICH by 381% at 8 KB, Cray MPICH by 149% at 2KB and Open MPI by 163% at 8KB

Allreduce - MVAPICH-Plus outperforms MPICH by 148% at 4 KB, Cray MPICH by 76% at 8KB, and Open MPI by 70% at 2KB

Collective Performance – 16 Nodes CPU (64PPN)



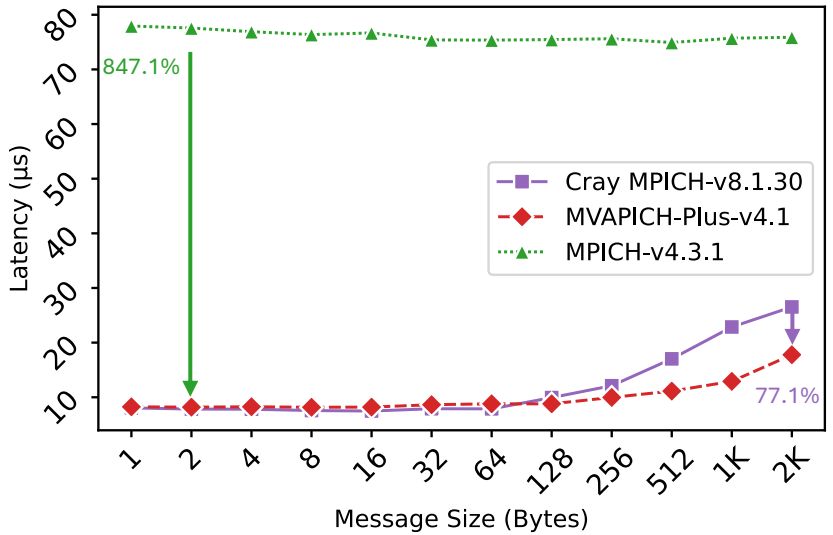




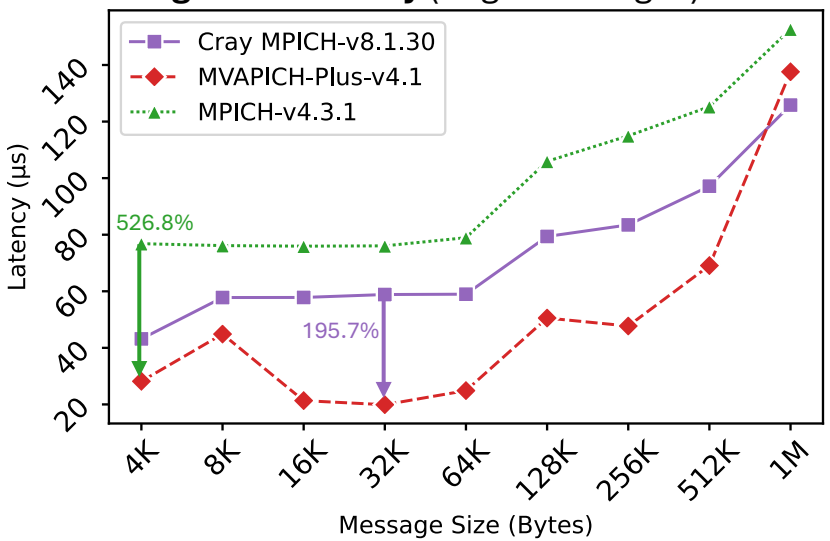
Alltoall - MVAPICH-Plus outperforms MPICH by 351% at 4 KB, Cray MPICH by 11% at 2KB, and Open MPI by 348% at 4KB

Collective Performance – 1 Nodes GPU (4APN)

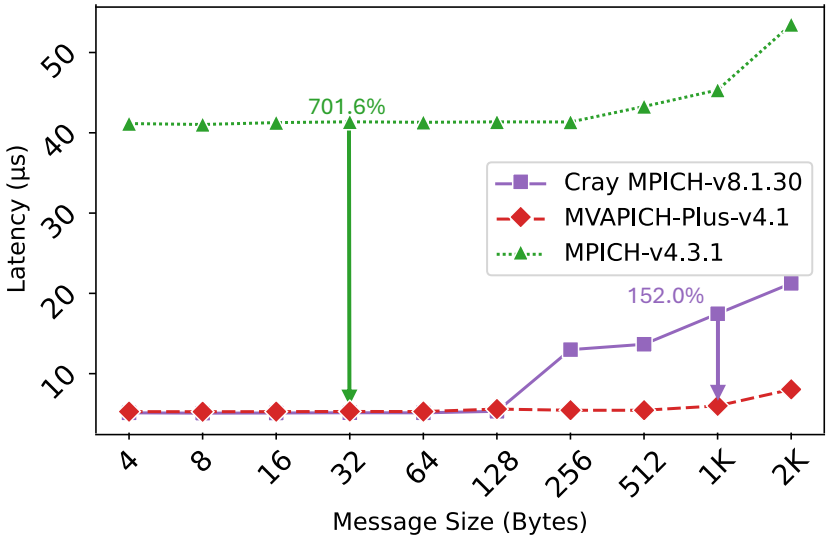
Allgather Latency (small messages)



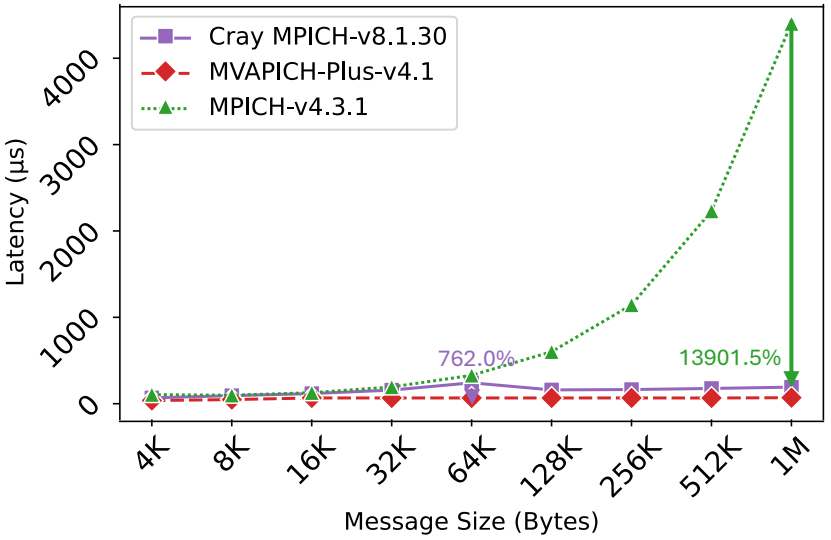
Allgather Latency (large messages)

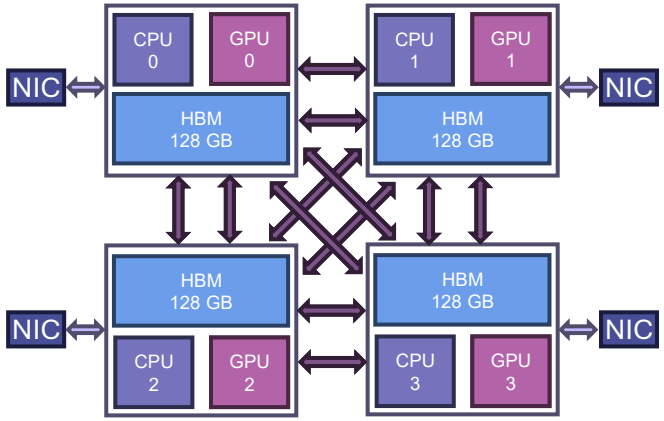


Allreduce Latency (small messages)



Allreduce Latency (large messages)





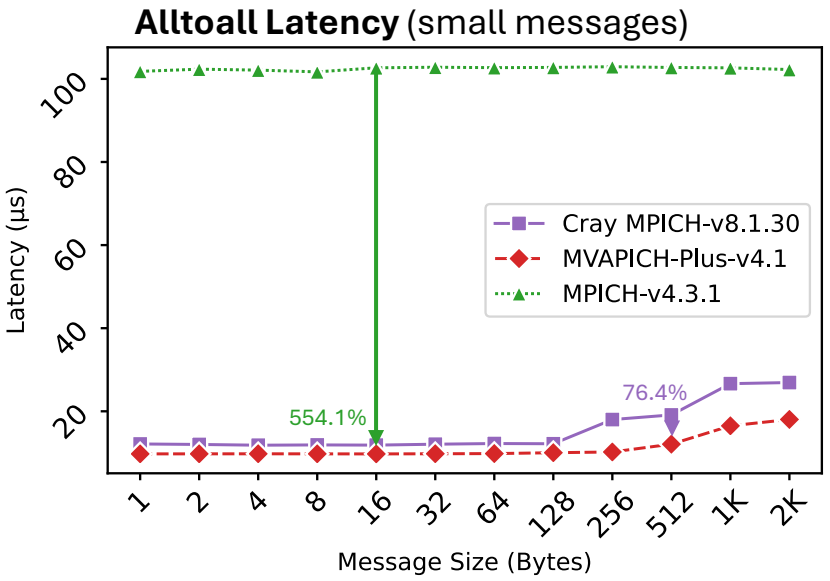
Inter APU connection **64** GB/s per dir per link via **xGMI/PCIe**

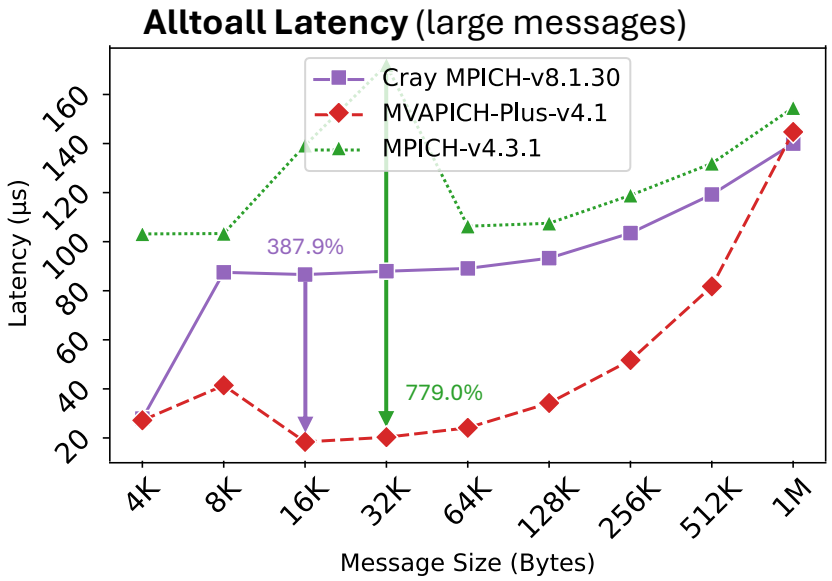
Inter Node connection **25** GB/s per dir via **Slingshot**

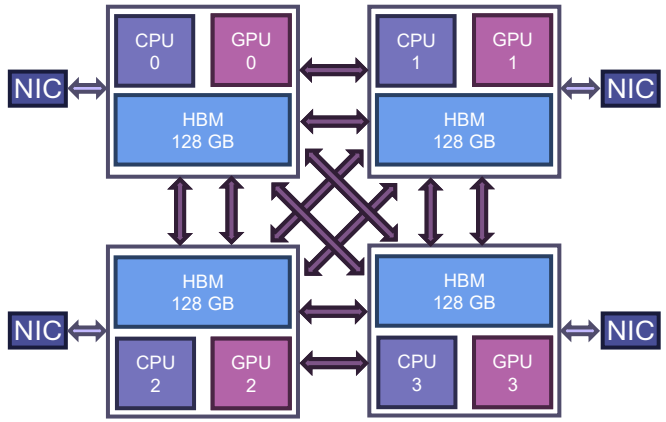
Allgather - MVAPICH-Plus outperforms MPICH by 526% at 4 KB, and Cray MPICH by 195% at 32KB.

Allreduce - MVAPICH-Plus outperforms MPICH by 13901% at 1 MB, and Cray MPICH by 762% at 64KB

Collective Performance – 1 Nodes GPU (4APN)







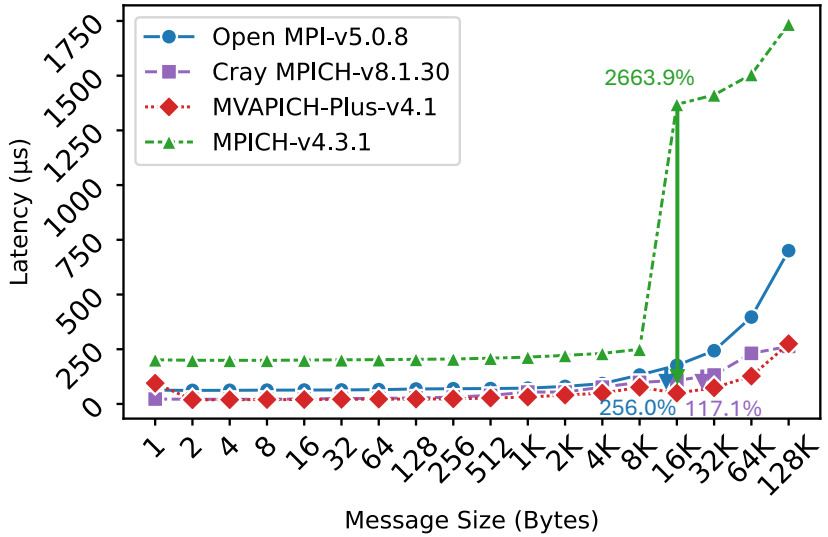
Inter APU connection **64** GB/s per dir per link via **xGMI/PCIe**

Inter Node connection **25** GB/s per dir via **Slingshot**

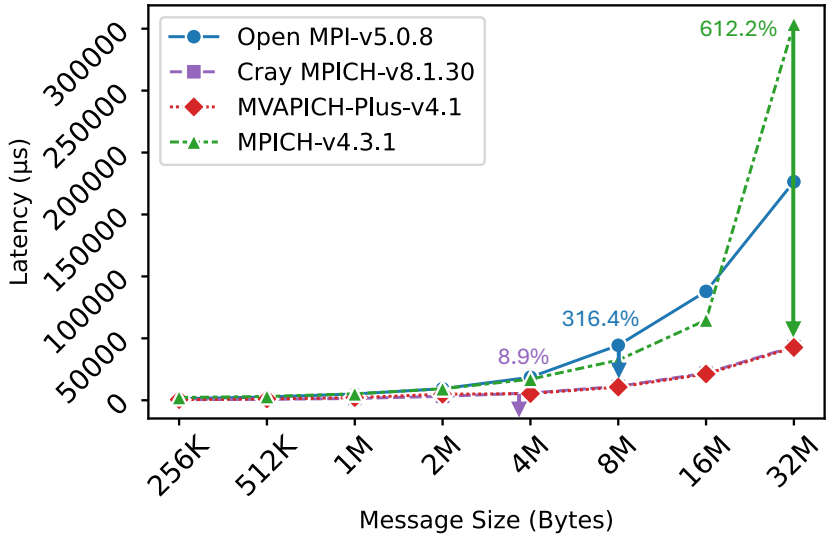
Alltoall - MVAPICH-Plus outperforms MPICH by 779% at 32 KB, and Cray MPICH by 387% at 16KB.

Collective Performance – 8 Nodes GPU (4APN)

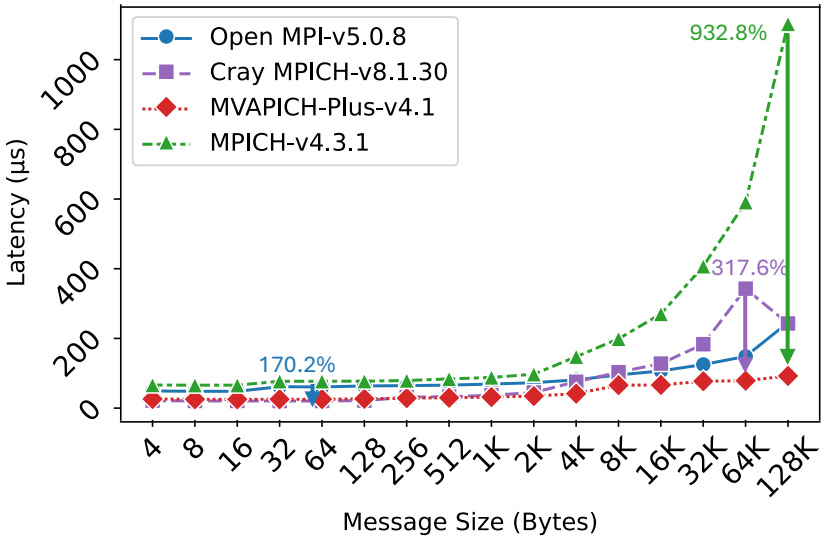
Allgather Latency (small messages)



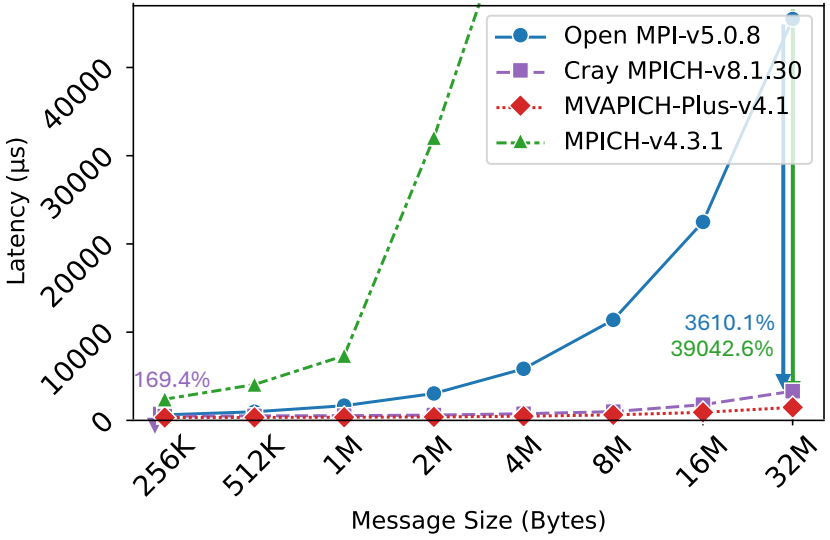
Allgather Latency (large messages)

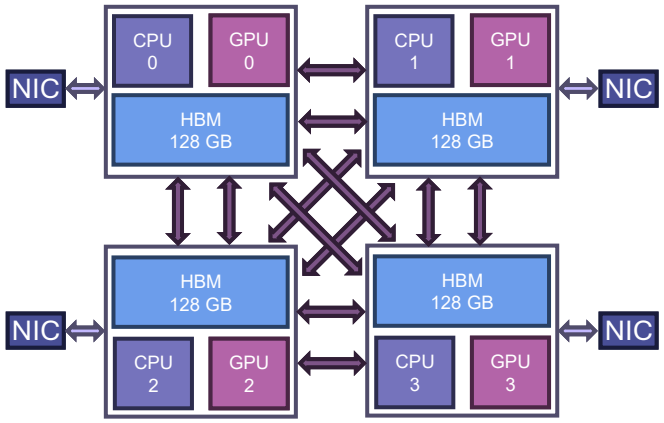


Allreduce Latency (small messages)



Allreduce Latency (large messages)



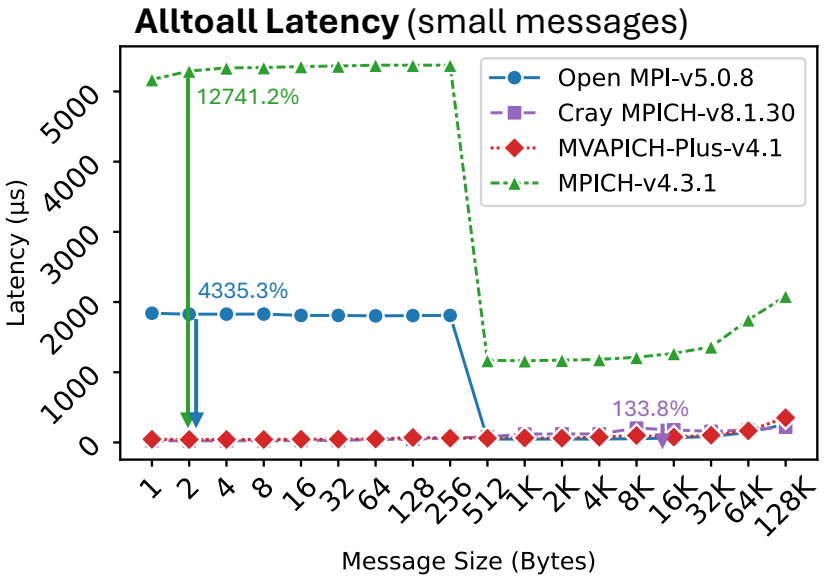


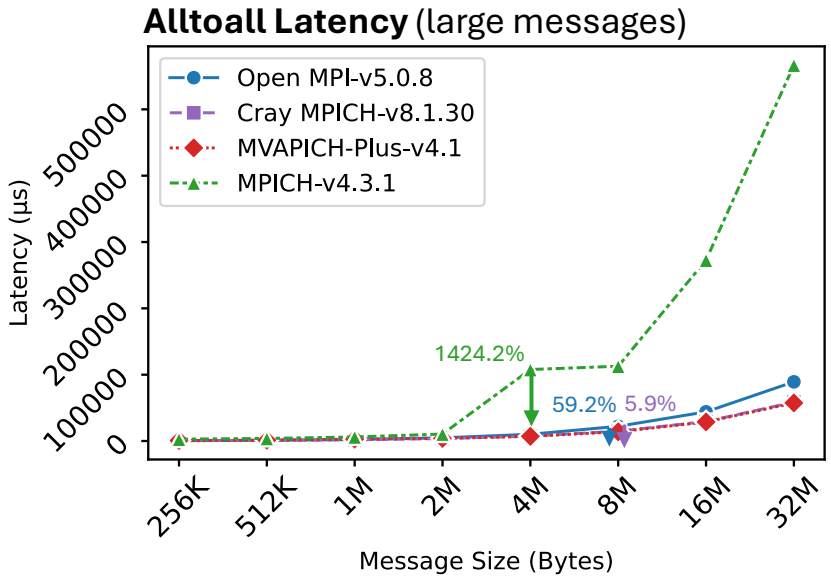
⟺ **Inter APU** connection **64** GB/s per dir per link via **xGMI/PCIe**
⟺ **Inter Node** connection **25** GB/s per dir via **Slingshot**

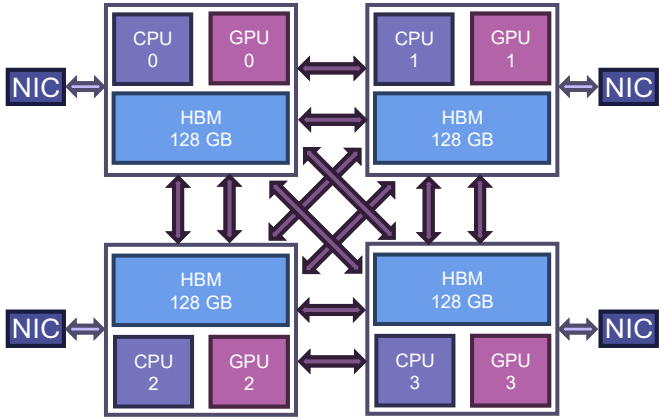
Allgather - MVAPICH-Plus outperforms MPICH by 612% at 1 MB, Cray MPICH by 117% at 32KB, and Open MPI by 316% at 8MB

Allreduce - MVAPICH-Plus outperforms MPICH by 39042% at 32 MB, Cray MPICH by 169% at 256KB, and Open MPI by 3610% at 32 MB

Collective Performance – 8 Nodes GPU (4APN)



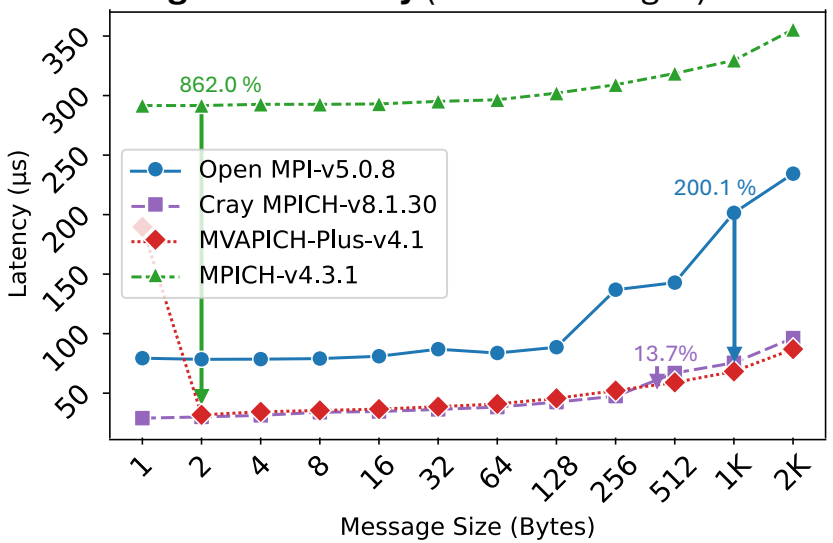




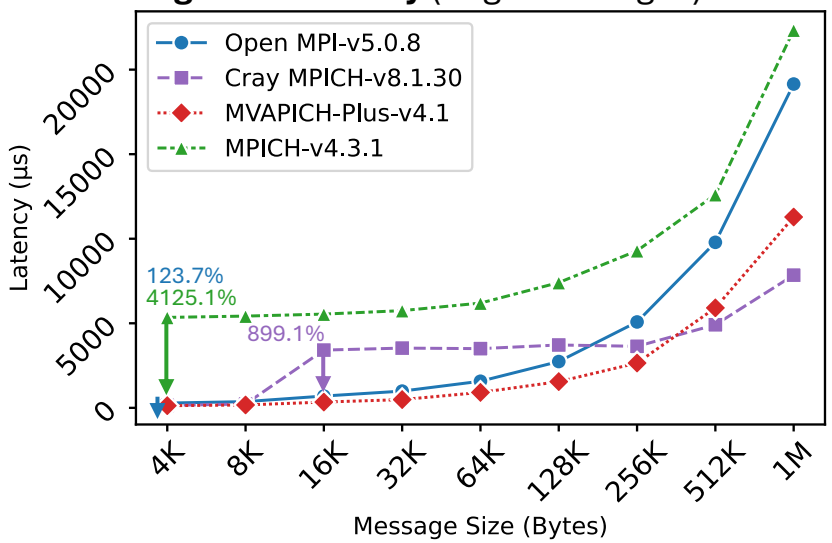
Alltoall - MVAPICH-Plus outperforms MPICH by 1424% at 4 MB, Cray MPICH by 133% at 16KB and Open MPI by 4335% at 2 B

Collective Performance – 32 Nodes GPU (4APN)

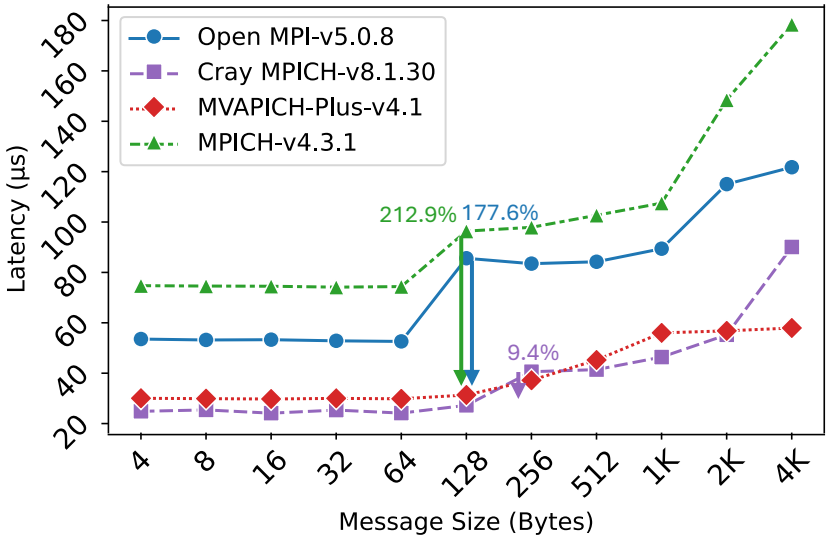
Allgather Latency (small messages)



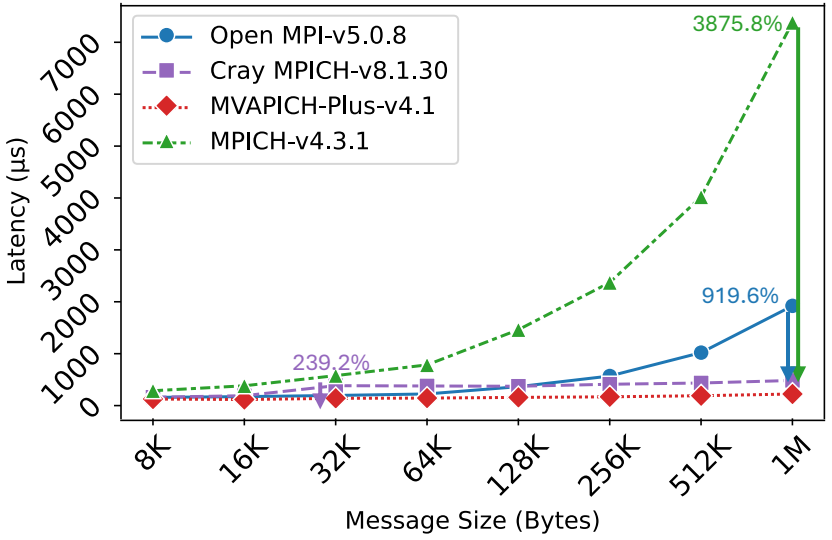
Allgather Latency (large messages)

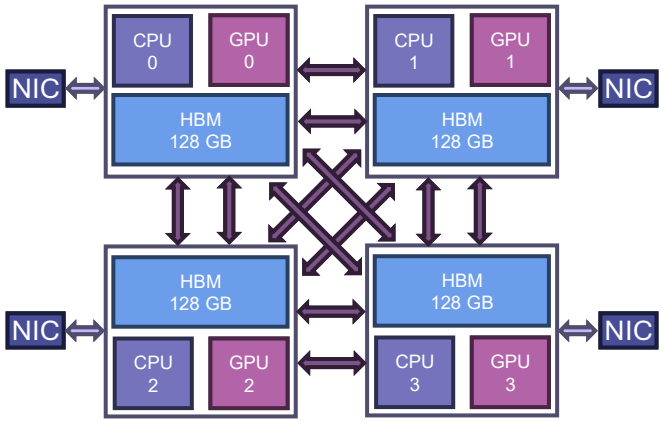


Allreduce Latency (small messages)



Allreduce Latency (large messages)



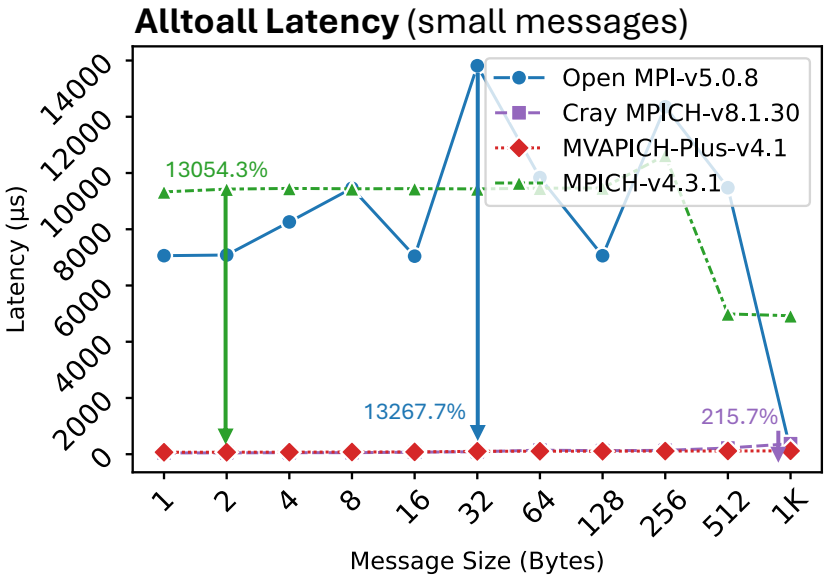


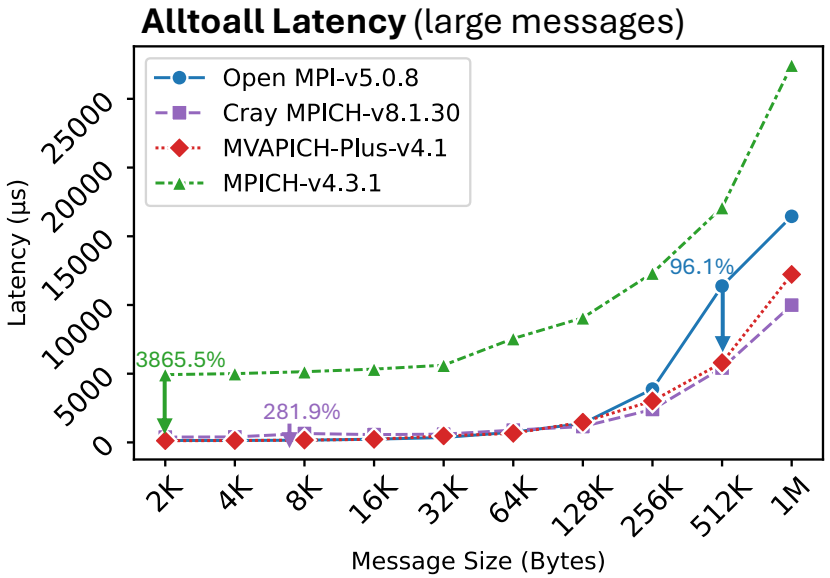
⟺ **Inter APU** connection **64** GB/s per dir per link via **xGMI/PCIe**
⟺ **Inter Node** connection **25** GB/s per dir via **Slingshot**

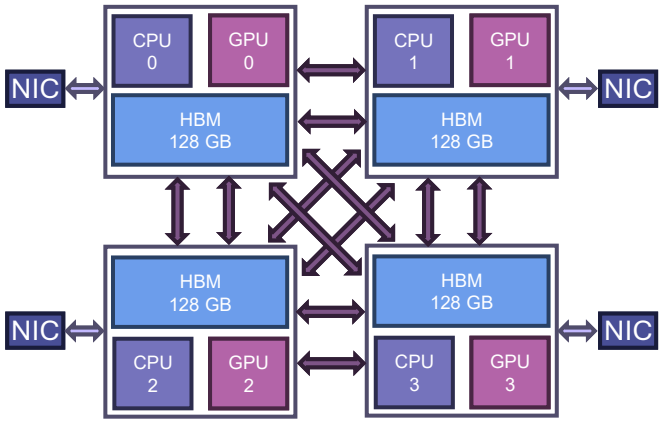
Allgather - MVAPICH-Plus outperforms MPICH by 4125% at 4 KB, Cray MPICH by 899% at 16 KB and Open MPI by 200% at 1KB

Allreduce - MVAPICH-Plus outperforms MPICH by 3875% at 1 MB, Cray MPICH by 239% at 32 KB, and Open MPI by 919% at 1MB

Collective Performance – 32 Nodes GPU (4APN)







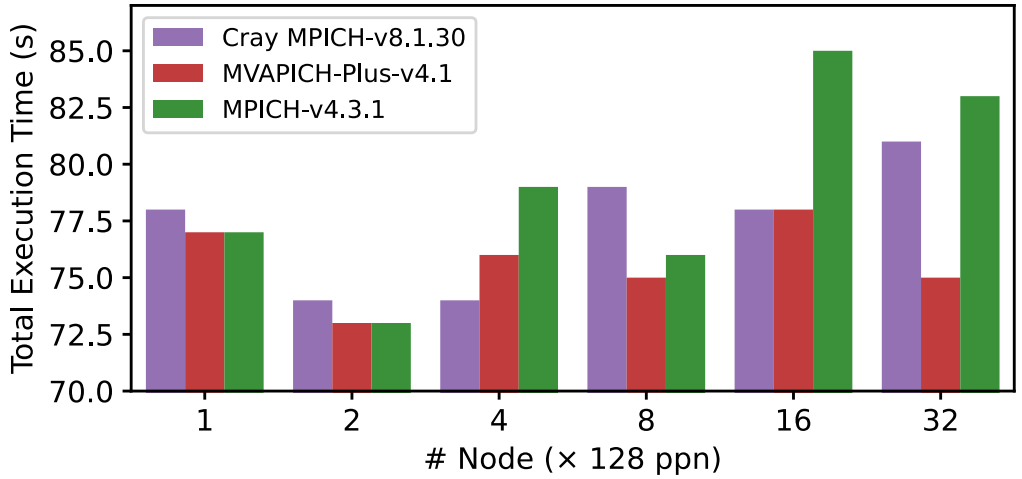
Alltoall - MVAPICH-Plus outperforms MPICH by 3865% at 2 KB, Cray MPICH by 281% at 8 KB and Open MPI by 96% at 512 KB

Performance Evaluation

Application

HPC Application: OpenFOAM (Weak Scaling)

OpenFOAM v 2412 + GCC 12.3
Cavity Test Case
Mesh Size defined as: #Nodes x 32 x 4

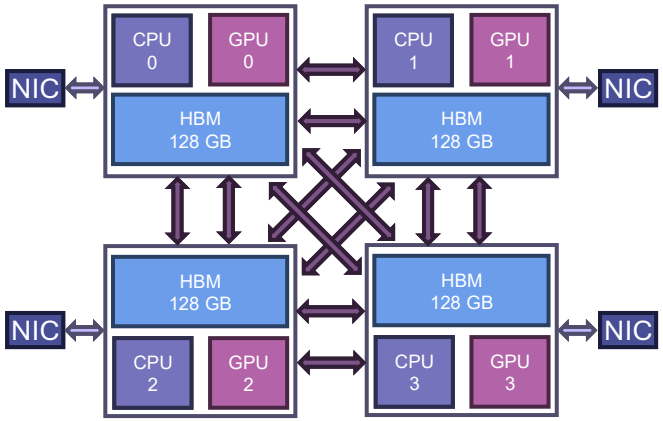


Performance at Small Scale

- All three MPI libraries perform similarly (**≤2%** difference)
- At **4** nodes:
  - Cray MPICH ≈ **2.6%** faster than MVAPICH-Plus
  - Cray MPICH ≈ **6.3%** faster than MPICH

Scalability at Larger Scale

- MVAPICH-Plus shows best scalability (**8–32** nodes)
- At **32** nodes:
  - MVAPICH-Plus outperforms MPICH by **9.6%**
  - MVAPICH-Plus outperforms Cray MPICH by **7.4%**



⟺ **Inter APU** connection **64** GB/s per dir per link via **xGMI/PCIe**
⟺ **Inter Node** connection **25** GB/s per dir via **Slingshot**

*Open MPI numbers are omitted due to a segmentation fault in this system

Deep Learning Application: nanoGPT (Strong Scaling)

- We evaluated the Distributed Data Parallel (DDP) training performance of **nanoGPT (~124M parameters)** on the **AMD MI300A cluster** using an internal PyTorch 2.7 fork optimized for MPI-based communication
- Each training step processed approximately **4 million tokens across all GPUs**.
- Gradient synchronization uses the **All-Reduce communication** pattern.

Performance Overview

- MVAPICH-Plus and Cray MPICH perform similarly, MVAPICH-Plus slightly faster
- Both significantly outperform MPICH and Open MPI

At 32 Nodes

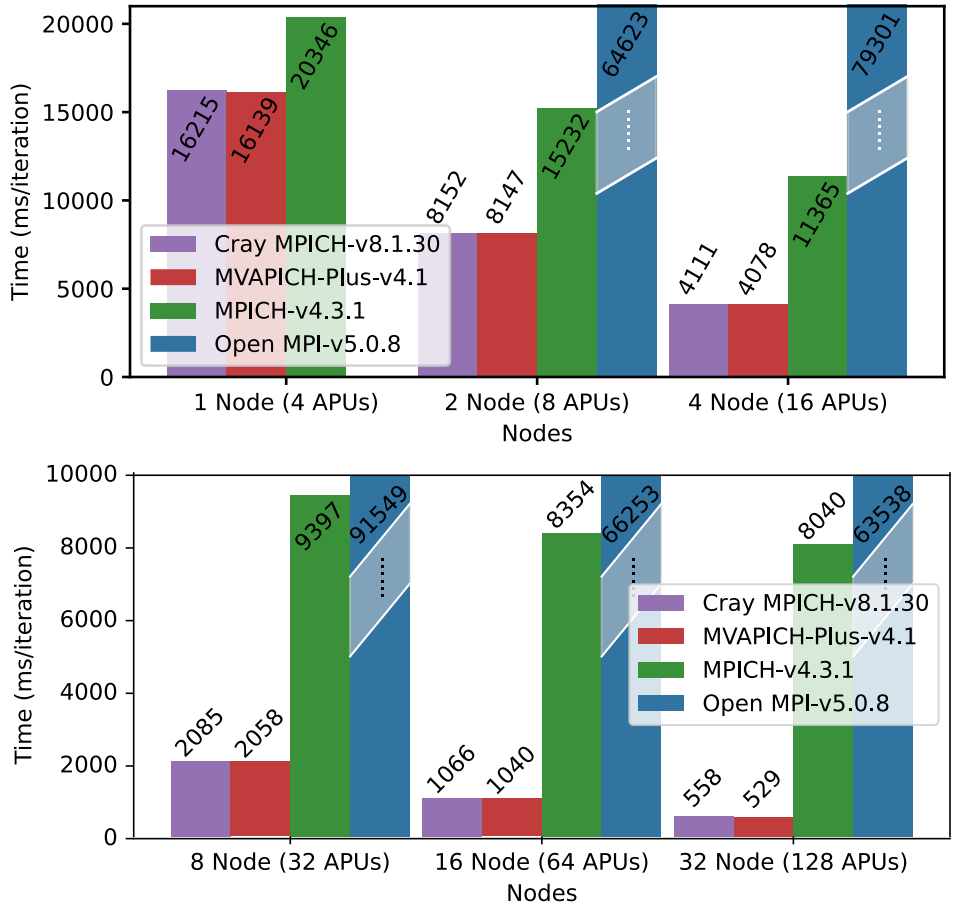
- MVAPICH-Plus & Cray MPICH achieve:
  - **93.42%** lower step time than MPICH
  - **99.17%** lower step time than Open MPI
- MPICH performs **87.35%** better than Open MPI

Scalability

- MPICH shows limited scaling beyond **8** nodes
- MVAPICH-Plus and Cray MPICH maintain efficiency at large scales

Stability

- Open MPI exhibited unstable training and runtime failures (single-node results omitted)



Outline

- Introduction
- Background
- Experimental Setup
- Performance Evaluation
- Conclusion & Future Work

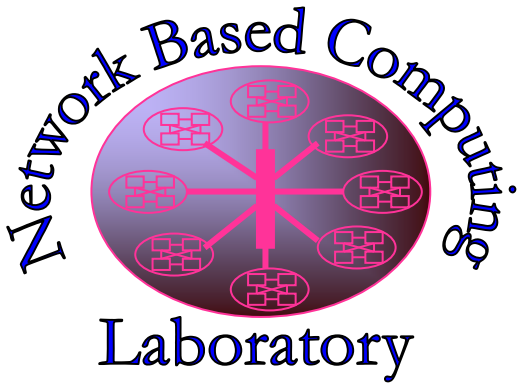
Conclusion & Future Work

- Demonstrated the performance characteristics and scaling behavior of MPI libraries on AMD MI300A APUs on the early access COSMOS system at SDSC.
- Unified HBM3 memory on MI300A eliminates traditional CPU-GPU memory transfer bottlenecks and simplifies programming.

Future Work

- Assess performance and programmability under different memory allocation strategies and GPU/CPU partitioning schemes.
- Explore optimization opportunities identified in the current results, aiming for improved scaling and reduced overhead.

Thank you!



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

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

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

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