Khorassani KS, Chen CC, Ramesh B *et al*. High performance MPI over the Slingshot interconnect. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 38(1): 128-145 Jan. 2023. DOI: 10.1007/s11390-023-2907-5

High Performance MPI over the Slingshot Interconnect

Kawthar Shafie Khorassani, Chen-Chun Chen, Bharath Ramesh, Aamir Shafi
Hari Subramoni, *Member, ACM, IEEE*, and Dhabaleswar K. Panda, *Fellow, IEEE, Member, ACM*

Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, U.S.A.

E-mail: shafiekhorassani.1@osu.edu; chen.10252@osu.edu; ramesh.113@osu.edu; shafi.16@osu.edu; subramoni.1@osu.edu
panda@cse.ohio-state.edu

Received October 16, 2022; accepted January 5, 2023.

Abstract The Slingshot interconnect designed by HPE/Cray is becoming more relevant in high-performance computing with its deployment on the upcoming exascale systems. In particular, it is the interconnect empowering the first exascale and highest-ranked supercomputer in the world, Frontier. It offers various features such as adaptive routing, congestion control, and isolated workloads. The deployment of newer interconnects sparks interest related to performance, scalability, and any potential bottlenecks as they are critical elements contributing to the scalability across nodes on these systems. In this paper, we delve into the challenges the Slingshot interconnect poses with current state-of-the-art MPI (message passing interface) libraries. In particular, we look at the scalability performance when using Slingshot across nodes. We present a comprehensive evaluation using various MPI and communication libraries including Cray MPICH, OpenMPI + UCX, RCCL, and MVAPICH2 on CPUs and GPUs on the Spock system, an early access cluster deployed with Slingshot-10, AMD MI100 GPUs and AMD Epyc Rome CPUs to emulate the Frontier system. We also evaluate preliminary CPU-based support of MPI libraries on the Slingshot-11 interconnect.

Keywords AMD GPU, interconnect technology, MPI (message passing interface), Slingshot

1 Introduction

The Frontier Supercomputer[①] deployed at the Oakridge Leadership Computing Facility (OLCF), now leading the Top500[②] list of supercomputers in the world and officially recognized as the first exascale supercomputer, is empowered by the HPE Cray Slingshot interconnect. In preparation for the vast demands of exascale computing and moving to a Slingshot-based networking environment, it is important to have an understanding of the interconnect with respect to MPI (message passing interface) communication. MPI libraries have been heavily deployed and used on systems with an underlying InfiniBand interconnect connecting nodes. They have been optimized and extensively researched in this ecosystem. Now, with upcoming exascale systems[③] choosing to deploy the Slingshot interconnect[④] as the underlying connection between nodes, it is crucial to have an understanding of the interconnect technology and how it impacts or improves the performance of communication at scale.

In this paper, we provide an analysis of the performance of various MPI libraries on a system with pre-

Regular Paper

Special Issue in Honor of Professor Kai Hwang's 80th Birthday

A preliminary version of the paper was published in the Proceedings of PEARC 2022.

This work is supported in part by the U.S. National Science Foundation under Grant Nos. 1818253, 1854828, 1931537, and 2007991, and XRAC under Grant No. NCR-130002. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

[①]Frontier: ORNL's exascale supercomputer designed to deliver world-leading performance in 2021. https://www.olcf.ornl.gov/frontier, Dec. 2022.

[②]TOP 500 Supercomputer sites. http://www.top500.org, Dec. 2021.

[③]OLCF. Spock quick-start guide. https://docs.olcf.ornl.gov/systems/spock_quick_start_guide.html, Dec. 2022.

[④]HPE. HPE Slingshot interconnect. https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html, Dec. 2022.

liminary/experimental deployment of the Slingshot interconnect. As this is a new area that has seldom been researched and is going to become a critical component of future HPC deployment, it is important to have this kind of detailed information and analysis that could provide a better outlook on the needs for optimizations and enhancements on upcoming systems deployed with Slingshot networking. This drives future research and innovations while also providing scalable and competitive options in this ecosystem that compare or improve upon existing innovations in the current interconnect technology realms.

1.1 Motivation

Many of the top supercomputers utilize InfiniBand networking, with the deployment of the Mellanox InfiniBand interconnect to connect nodes across the network. This area has been heavily evaluated and analyzed over the years with various MPI libraries utilizing GPU-aware and CPU-based communication to scale out performance onto multiple nodes. This understanding of the limitations and advantages of the interconnect technology drove future directions in research over the years related to communication optimization and performance analysis. With the deployment of the Slingshot interconnect, it is just as important to develop an understanding of the advantages and features the interconnect introduces in order to motivate future approaches in the communication realm.

The underlying interconnect technology is a critical component in achieving high performance, low latency and high throughput, at scale on next-generation exascale systems. This drives the motivation to have a detailed analysis and understanding of the existing MPI libraries and the performance they are able to demonstrate at certain scales, various configurations, and for different communication operations. Through this work, we demonstrate a need for a thorough evaluation of communication over the newer Slingshot interconnect and its ecosystem in preparation for exascale systems in order to achieve the scalability and efficiency that is promised by the next generation of supercomputing.

1.2 Key Insights and Contributions

The performance of GPU-aware approaches to communication provided by the state-of-the-art communication libraries on the Slingshot interconnect have yet to be explored. There is a lack of thorough evaluation and analysis of performance comparing the different communication operations and detailing the demands for MPI at the application layer on a system with Slingshot interconnects. Additionally, the systems used in this study include AMD MI100 and AMD MI250X GPUs, which are also a snapshot of the type of system and ecosystem we can expect for the next-generation exascale systems. Through this work, we make the following contributions.

- Comprehensive evaluation of CPU-based communication using various communication libraries, including OpenMPI + UCX, MVAPICH2-X, and Cray MPICH on the Spock system with the Slingshot-10 interconnect, and AMD EPYC Rome CPUs for point-to-point and collective benchmarks.
- Comprehensive evaluation of GPU-aware communication using various communication libraries, including OpenMPI + UCX, MVAPICH2-GDR, Cray MPICH, and RCCL on the Spock system with the Slingshot-10 interconnect, AMD MI100 GPUs, and AMD EPYC Rome CPUs for point-to-point and collective benchmarks.
- Evaluation of preliminary CPU-based communication on the Slingshot-11 interconnect with MVAPICH2-3.0a and CrayMPICH on a system emulating the Frontier system with Slingshot-11 networking.
- Application-level evaluation using state-of-the-art communication libraries for rocHPCG and for the HeFFTe application using the rocfft backend for AMD GPUs and the fftw backend for CPUs.
- Delving into various challenges that the current Slingshot-10 interconnect brings about in terms of communication performance and what challenges to consider for future deployment of MPI libraries on systems with the upcoming Slingshot-11 interconnect such as support for the underlying Cray Fabric and adapter, in preparation for new exascale systems such as Frontier.

2 Background

This paper is an extended version of [1]. In [1], we presented our early experiences with Slingshot-10 and MPI libraries with a focus on GPU-aware MPI. In this paper, we have made the following enhancements:

1) We add CPU-based experiments and evaluations of MPI libraries including MVAPICH2-X, OpenMPI, and Cray MPICH on the Spock system using

OSU-Microbenchmarks. We evaluated intra-node and inter-node point-to-point performance on CPUs over the Slingshot-10 interconnect and between AMD Epyc Rome CPUs. We evaluated collectives performance on up to 512 CPUs (8 nodes, 64 PPN) for reduce, allreduce, gather, allgather, broadcast, and alltoall. These are detailed in Subsection 3.3.

2) We evaluate heFFTe using the fftw backend for CPU-based communication using MVAPICH2-X and OpenMPI on up to 512 CPUs on the Spock System for a problem size up to 512^3 with alltoall and alltoallv 3.7.

3) We add and evaluate the performance of MPI libraries over the Slingshot-11 interconnect and present a new system with Slingshot-11 networking used in our evaluations. No other work has been done on Slingshot-11 networking with MPI. This work details preliminary MPI support over the Slingshot-11 interconnect and demonstrates performance at the benchmark level and challenges to consider for future MPI deployment over upcoming exascale systems with Slingshot-11 networking. These are detailed further in Subsection 3.5 and Subsection 3.6.

2.1 State-of-the-Art Interconnect Technologies

Achieving high performance for complex HPC workloads that benefit from high levels of parallelism requires efficient and scalable network interconnects. Modern interconnects such as InfiniBand, RoCE, Omni-Path, and so on, were introduced into the market to address communication bottlenecks by achieving low latency and high throughput between nodes. In recent years, InfiniBand and high-speed Ethernet represent the gold standard for high-performance network interconnects. For instance, Summit@ORNL (ranked 4th on the June 2022 Top500 list[⑤]), uses Dual-rail Mellanox EDR InfiniBand as the underlying interconnect. Approximately 35% of supercomputers in the Top500 utilize InfiniBand networking (including Sierra@LLNL, Selene@NVIDIA, etc.), and about 48% deploy Gigabit Ethernet networking (including Perlmutter@NERSC, Polaris@ANL, etc.). The adoption rates for interconnects in upcoming exascale systems are rapidly changing due to the increased number of choices and evolving interconnect standards.

2.2 Slingshot Interconnect

HPE Slingshot[⑥] is a high-performance network designed by HPE Cray for upcoming exascale-era systems, and is based on Ethernet. It provides flexibility and capabilities to enable users to run a wide mix of workflows. The switches support a high-radix and up to 12.8 Tb/s bandwidth. While the latency of Ethernet networks is slightly worse when compared to InfiniBand systems in general, Ethernet networks claim the advantage of wider adoption across application domains. HPE Slingshot delivers low latency and high throughput for HPC workloads, and minimizes the number of switch hops in large networks (for instance, by employing the use of the Dragonfly[2] topology). The interconnect features adaptive routing techniques to help maintain the balanced traffic flows through fine-grained optimization. HPE Slingshot also introduces a fully automatic and hardware-implemented congestion control mechanism to minimize the impact of congestion when multiple workloads run at the same time. It is currently empowering the first official exascale supercomputer in the world, Frontier@OLCF, and in the work to be deployed on future exascale supercomputers as well, such as El Capitan@LLNL.

2.3 State-of-the-Art Communication Libraries

The Message Passing Interface (MPI) is a multiprocessing paradigm that enables communication among processes on parallel architectures. The communication primitives can be categorized as one-sided, point-to-point, and collective operations. One-sided communication indicates the use of only one process to move data to a remote process (without the remote process's involvement). Hence, it is also referred to as remote memory access (RMA). It decouples the process synchronization during data transfer. MPI_Put, MPI_Get, and MPI_Accumulate are well-known one-sided communication operations. The MPI standard also supports expressing point-to-point communication operations using two-sided semantics using MPI_Send, MPI_Recv, MPI_Isend, and MPI_Irecv. Collective communication operations defined by the MPI standard provide convenient abstractions for multiple processes/threads to efficiently communicate with one another. These operations can

[⑤]TOP 500 Supercomputer sites. http://www.top500.org, Dec. 2021.
[⑥]HPE. HPE Slingshot Interconnect. https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html, Dec. 2022.

involve computing operations (in reduction collectives such as MPI_Allreduce and MPI_Reduce) or just communication to represent common patterns such as a broadcast, scatter, gather, and others.

Aside from the MPI interface, there are other communication libraries that use and expose a different underlying API to transfer messages. For example, the NVIDIA Collective Communication Library (NCCL), provides optimized communication primitives for GPU-to-GPU communication within as well as across the node for NVIDIA GPUs. ROCm Communication Collectives Library (RCCL) is the communication library based on NCCL for AMD GPUs, providing primitives that enable GPU-to-GPU communication on AMD ROCm supported systems, similar to what NCCL achieves on systems with NVIDIA GPUs.

2.4 Limitations of State-of-the-Art Approaches

Existing MPI libraries provide support for various network features such as Omni-Path, RoCE, InfiniBand, and so on. The growth in the deployment of the Slingshot interconnect across upcoming systems adds the Slingshot interconnect to the growing list of features that MPI libraries will need to add functionality and optimizations for. HPE/Cray has designed the Slingshot interconnect in such a way to be ethernet compatible in order to provide ease of interoperability with existing systems. This enables a direct connection between the switches for Slingshot and ethernet networks and storage devices[7]. It also provides support for features such as adaptive routing, congestion control, and isolated workloads. These features provide several challenges and possibilities to explore and enhance state-of-the-art communication libraries. The limitations of current state-of-the-art approaches will be made more clear with the deployment of Slingshot-11. Current accessibility and deployment on early access Slingshot systems provide an ecosystem with Slingshot-10 interconnection amongst nodes. The second generation of Slingshot, Slingshot-11, is deployed over a Slingshot fabric and adapter, while the current deployment of Slingshot-10 is running over a Slingshot network with a Mellanox InfiniBand adapter. This second-generation deployment introduces additional challenges for communication libraries to develop functionality over the underlying adapter and fabrics. In Subsection 3.6, we present a preliminary evaluation of MPI libraries with support for Slingshot-11 on CPUs.

3 Evaluation and Analysis

In this section, we provide details of the Spock system (Fig.1) used for the experiments and evaluations and the software environment on this system. We also provide additional details specific to the MPI and communication libraries used in the evaluation. We include a detailed analysis of communication performance using various MPI libraries at the benchmark and application layers.

3.1 Spock—System and Software Details

The performance evaluation is done on the Spock system deployed at the Oakridge Leadership Computing Facility (OLCF)[8]. This is an early access system provided in preparation for the exascale system, Frontier. This preparation for the deployment of exascale systems allows for experiments and evaluations to be done in order to develop an understanding of what to expect in terms of communication library performance on the upcoming exascale systems, and the challenges in relation to communication on a system with Slingshot interconnects and the latest AMD GPUs.

The Spock cluster consists of 64-core AMD EPYC 7662 Rome CPUs, and four AMD MI100 GPUs with 32 GB HBM2 per node. The GPUs are connected within a node via Infinity Fabric and connected to the CPU via PCIe Gen4. The nodes are connected via the Slingshot-10 interconnect, providing 12.5 GB/s bandwidth across nodes. The latest version of ROCm deployed on the system is ROCm 5.0.2. This information is detailed in the Spock compute node presented in Fig.1. More details of the communication libraries and software stack versions used on this system for this evaluation are provided in Table 1[1].

3.1.1 MPI Libraries

Table 2[1] details the various MPI libraries used and configuration details specific to each of the libraries. The MVAPICH2-GDR library v2.3.7 was

[7]HPE. HPE Slingshot interconnect. https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html, Dec. 2022.
[8]OLCF. Oakridge National Laboratory: Leadership Computing Facility. https://www.olcf.ornl.gov, Dec. 2022.

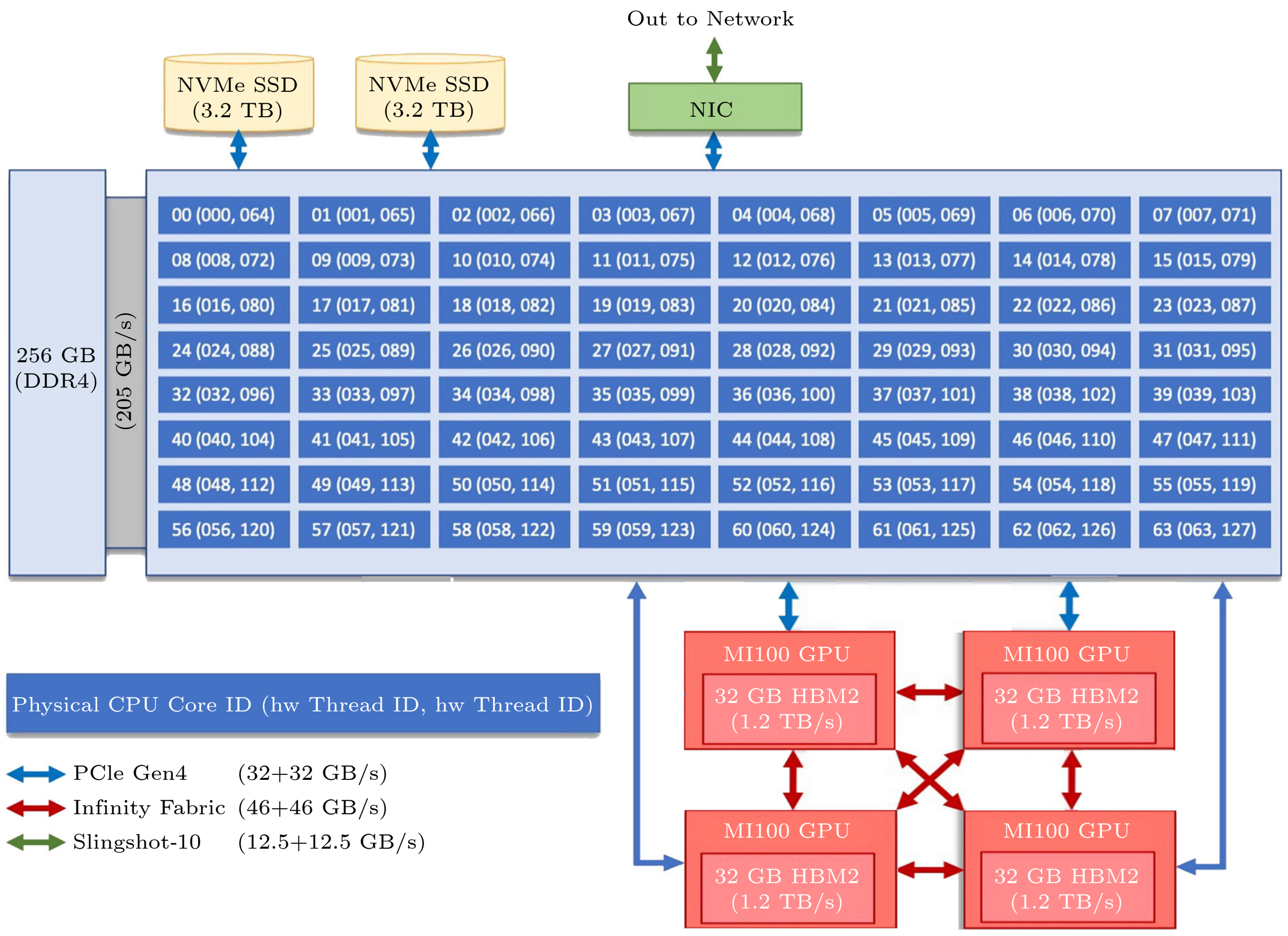


Fig.1. Spock compute node details[9]. hw: Hardware.

Table 1. System Details and Usage[1]

	Software	Version	Cite
MPI & communication libraries	Open MPI	4.1.4	[3]
	UCX	1.12.1	[10]
	Cray MPICH	8.1.14	[4]
	RCCL	5.0.2	[11]
	MVAPICH2-GDR	2.3.7	[5]
	MVAPICH2-X	2.3	[5]
	MVAPICH2	3.0a	[5]
Platform	ROCm	5.0.2	[12]
benchmarks &	OSU Micro-Benchmarks	5.9	[6]
applications	heFFTe	2.0	[13]

used for the evaluations done on GPUs (MVAPICH2-GDR optimized for GPU-aware communication). This library provides downloadable options from the site or through the user forum in order to execute on the system. Specific configuration was not required here. The MVAPICH2-GDR installation is linked to ROCm 5.0.2, the latest version of ROCm on the Spock system. OpenMPI version 4.1.4 and UCX version 1.12.1, the latest versions of the stack were used in the performance evaluation. The configuration details of UCX to link with ROCm and enable optimizations and the details for linking OpenMPI to this UCX installation are demonstrated in the table.

Cray MPICH 8.1.14 is the MPI library deployed on the Spock system by default. It requires a load of the existing module, adding ROCm into the path, and loading an additional module to detect the architecture. These modules are detailed in Table 2. Finally, the ROCm Collectives Communication Library (RCCL) was used as well in the evaluation of GPU-aware communication.

[9]OLCF. Spock quick-start guide. https://docs.olcf.ornl.gov/systems/spock_quick_start_guide.html, Dec. 2022.
[10]Unified communication x. http://www.openucx.org/, Dec. 2021.
[11]Rocm communication collectives library (rccl). https://github.com/ROCmSoftwarePlatform/rccl, Dec. 2021.
[12]Radeon open compute (rocm) platform. https://rocmdocs.amd.com, Dec. 2021.
[13]Highly efficient fft for exascale (heFFTe) library. https://github.com/af-ayala/heffte, Dec. 2021.

Table 2. MPI Libraries Configuration and Installation Details[1]

Communication & Library	Configuration & Installation Detail
MVAPICH2-GDR 2.3.7	MVAPICH2-GDR 2.3.7 + ROCm 5.0.2 for GPUs
MVAPICH2-X 2.3	MVAPICH2-X 2.3 + XPMEM
MVAPICH2 3.0a	--with-device=ch4:ofi --with-libfabric=<path-to-libfabric>
OpenMPI 4.1.4 + UCX 1.12.1	UCX: --with-rocm=<path-to-rocm> --without-knem --without-cuda --enable-optimizations OpenMPI: --with-ucx=<path-to-ucx> --without-verbs Run: -x UCX_RNDV_THRESH=128
Cray MPICH 8.1.14	module load craype-accel-amd-gfx908 module load cray-mpich/8.1.14 Run: MPICH_GPU_SUPPORT_ENABLED=1
RCCL 5.0.2	CXX=<path-to-rocm>/bin/hipcc

3.2 OSU Micro-Benchmarks

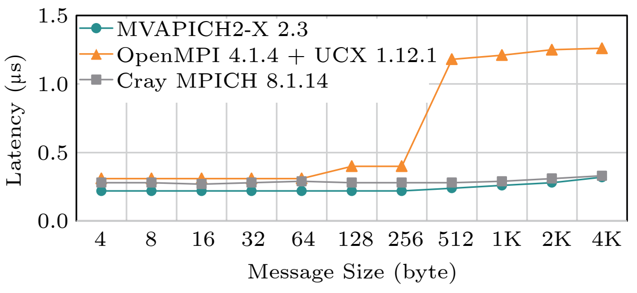
To compare the performance of various communication operations on the Spock cluster using different MPI libraries, we utilize the OSU Micro-Benchmarks (OMB) suite version 5.9. It reports intra- and inter-node point-to-point latency and bandwidth, and the performance of MPI collective operations at different message sizes.

3.3 Micro-Benchmark Evaluation on CPUs

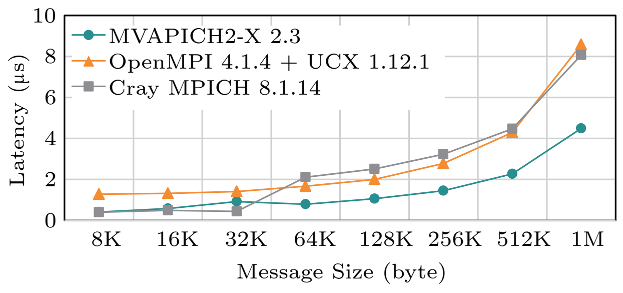
We first provide a detailed performance evaluation on CPUs for intra-node and inter-node point-to-point communication and for collectives communication on up to four nodes (due to the Spock system being an early access cluster, there are user limitations that instill a maximum allocation of four nodes).

3.3.1 Intra-Node Point-to-Point

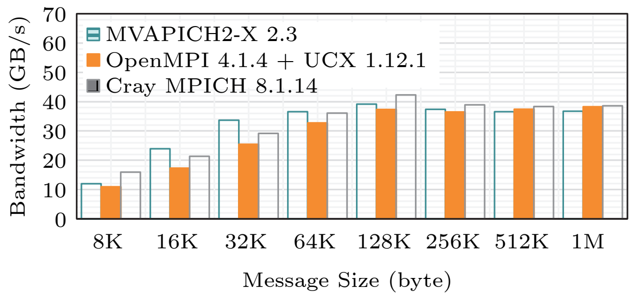
In Fig.2 we present an evaluation of intra-node point-to-point benchmark-level performance comparing MVAPICH2-X, OpenMPI + UCX, and Cray MPICH on AMD Epyc Rome CPUs. The evaluation was done between two CPUs within one node using the point-to-point benchmarking tests provided by the OSU-Microbenchmarks suite for latency (*osu_latency*), bandwidth (*osu_bw*), and bi-directional bandwidth (*osu_bibw*). In Fig.2(a), for small message intra-node latency, MVAPICH2-X, Cray MPICH, and OpenMPI + UCX achieve 0.22 µs, 0.31 µs, and 0.36 µs latency, respectively. Each of the libraries achieves their peak unidirectional bandwidth at a message size



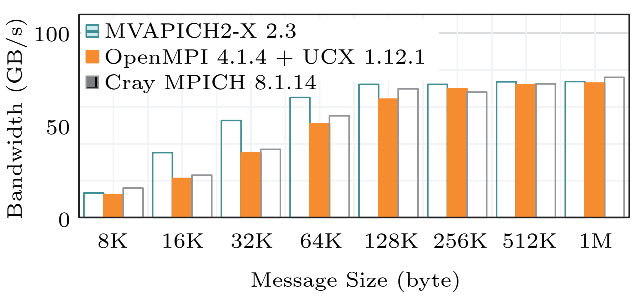
(a)



(b)



(c)



(d)

Fig.2. Intra-node point-to-point performance on AMD Epyc Rome CPUs on the Spock system. (a) Small message point-to-point latency. (b) Large message point-to-point latency. (c) Large message bandwidth. (d) Large message bi-directional bandwidth.

of 128 KB as demonstrated in Fig.2(c) with MVAPICH2-X achieving approximately 39 GB/s, OpenMPI + UCX at 37 GB/s, and Cray MPICH at 42 GB/s. The high bandwidth numbers can be attributed to the usage of XPMEM, and cooperative protocols[7] for the message exchange between processes. XPMEM enables a process to map the memory of another process into its virtual address space, thereby achieving direct load-store access in user-space. This outperforms kernel-based copy mechanisms such as cross memory attach (CMA). Co-operative rendezvous protocols employ the use of both sender and receiver CPU DMA engines to perform data transfers between two processes, effectively doubling the peak bandwidth that can be achieved by one CPU.

3.3.2 Inter-Node Point-to-Point

In Fig.3 we present an evaluation of inter-node point-to-point benchmark-level performance comparing MVAPICH2-X, OpenMPI + UCX, and Cray MPICH on AMD Epyc Rome CPUs. The evaluation is done between two CPUs on different nodes connected by the Slingshot interconnect for latency (*osu_latency*), bandwidth (*osu_bw*), and bi-directional bandwidth (*osu_bibw*). This configuration over the Slingshot-10 interconnect provides a 12.5 GB/s node injection bandwidth. All of the MPI libraries perform close to peak achievable bandwidth over the Slingshot interconnect as shown in Fig.3(c), where MVAPICH2-X, OpenMPI + UCX, and Cray MPICH all achieve approximately 12.2 GB/s bandwidth at 1 MB. For bi-directional bandwidth in Fig.3(d), all three libraries reach approximately 24 GB/s bi-directional bandwidth peak at 1 MB.

3.3.3 Collective Operations

We evaluate various collective operations including MPI_Reduce and MPI_Allreduce (Fig.4), MPI_Gather and MPI_Allgather (Fig.5), and MPI_Bcast and MPI_Alltoall (Fig.6) using the OSU Micro-Benchmarks suite. Various tests are included here, specific to each MPI operation. The performance evaluation demonstrates a comparison among the three different MPI libraries (MVAPICH2-X, OpenMPI + UCX, and Cray MPICH) on 512 AMD Rome CPUs (eight nodes, 64 CPUs per node). For small messages, MPI_Reduce, MVAPICH2-X, OpenMPI + UCX, and Cray MPICH achieve 0.8 µs, 2.05 µs, and 1.95 µs latency at 4 B, respectively. Large message MPI_Allreduce performance is depicted in

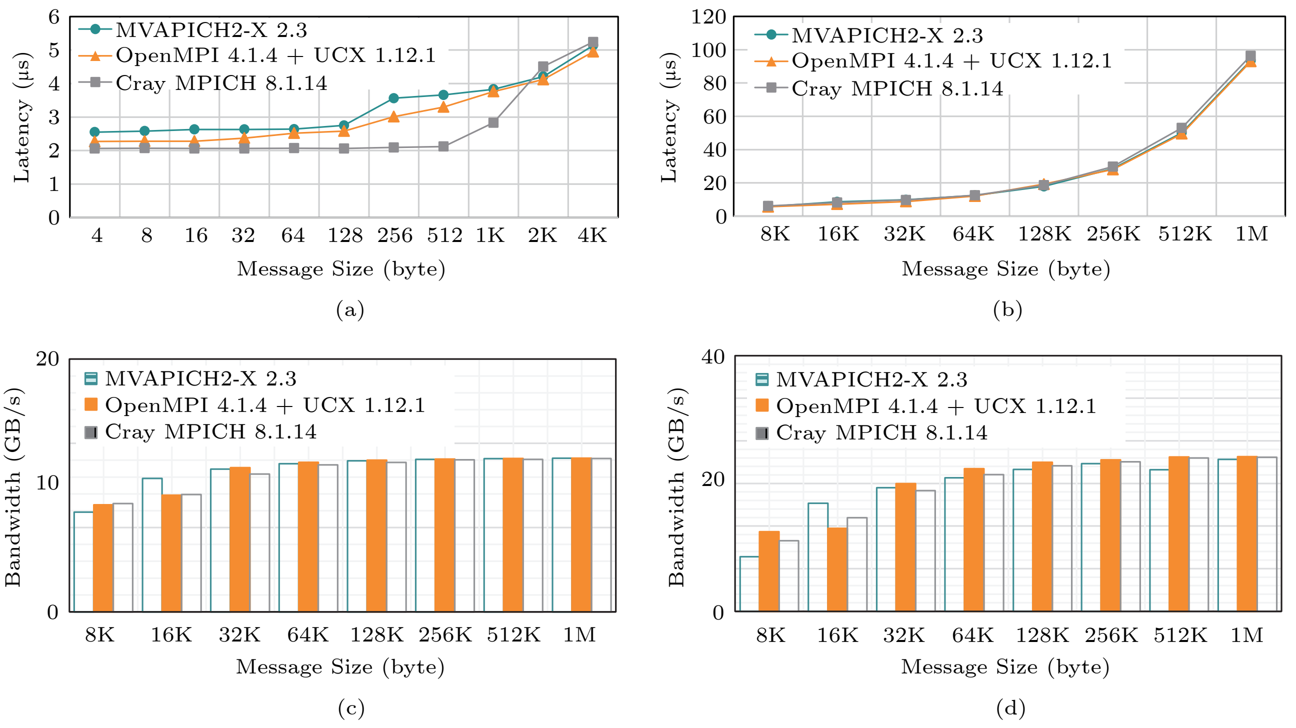


Fig.3. Inter-node point-to-point performance on AMD Epyc Rome CPUs on the Spock system over Slingshot-10. (a) Small message point-to-point latency. (b) Large message point-to-point latency. (c) Large message bandwidth. (d) Large message bi-directional bandwidth.

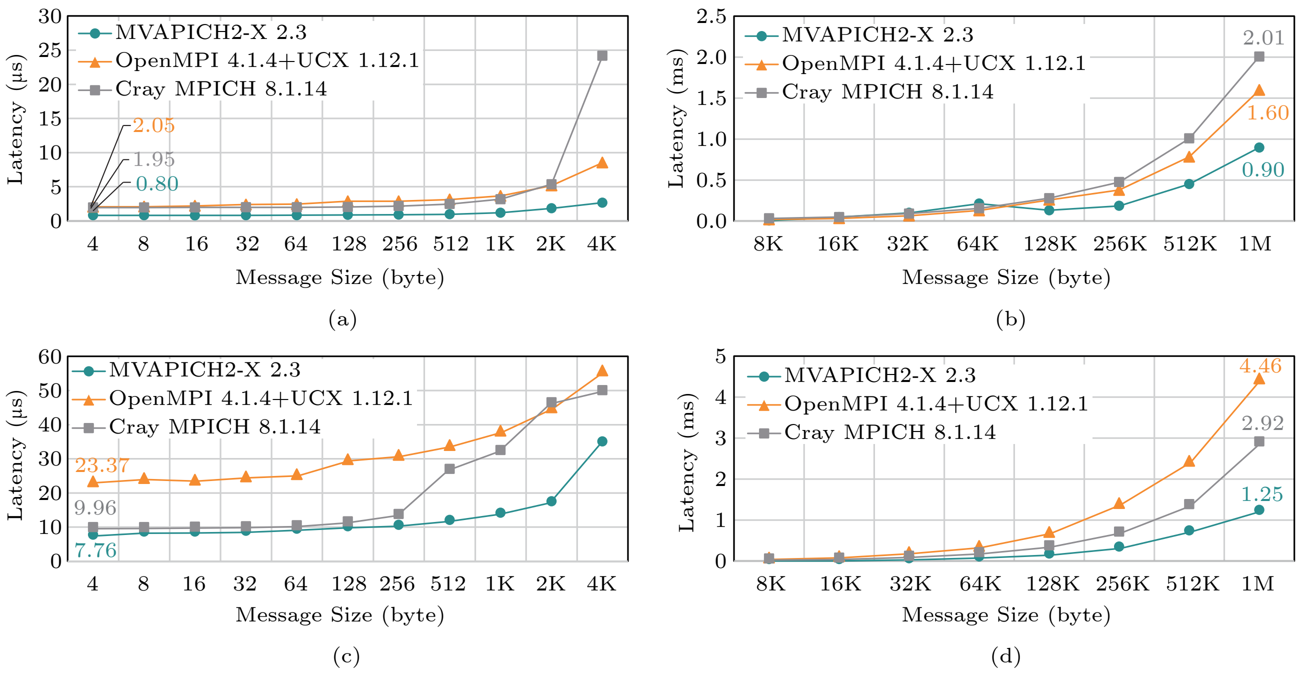


Fig.4. Performance of MPI collective MPI_Reduce and MPI_Allreduce operations on CPUs (512 CPUs—8 nodes & 64 PPN). (a) REDUCE—small message sizes. (b) REDUCE—large message sizes. (c) ALLREDUCE—small message sizes. (d) ALLREDUCE—large message sizes.

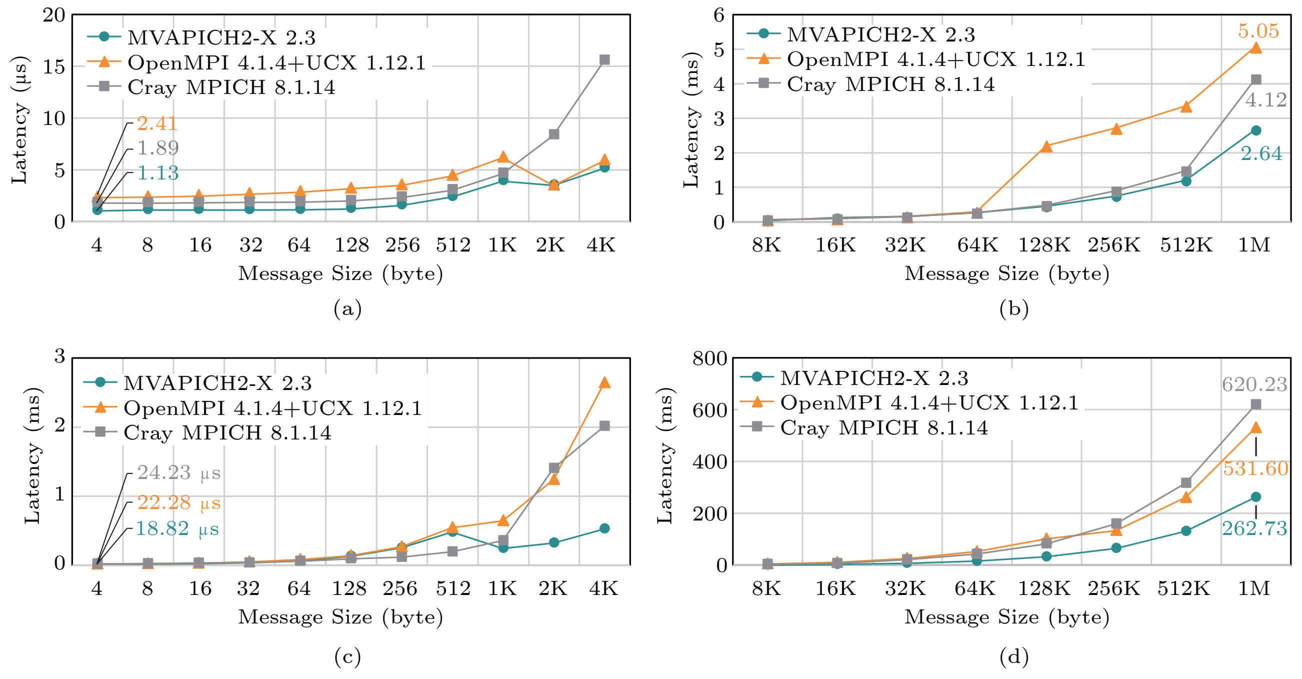


Fig.5. Performance of MPI collective MPI_Gather and MPI_Allgather operations on CPUs (512 CPUs—8 nodes & 64 PPN). (a) GATHER—small message sizes. (b) GATHER—large message sizes. (c) ALLGATHER—small message sizes. (d) ALLGATHER—large message sizes.

Fig.4(d), MVAPICH2-X achieves 1.25 ms latency, while OpenMPI + UCX achieves 4.46 ms, and Cray MPICH demonstrates 2.92 ms latency. We see similar alltoall performance between Cray MPICH and Open MPI + UCX in Figs.6(c) and 6(d).

3.4 Micro-Benchmark Evaluation on GPUs

In this subsection, we delve into the GPU-based evaluation utilizing GPU-aware MPI and communication libraries. We evaluate the point-to-point perfor-

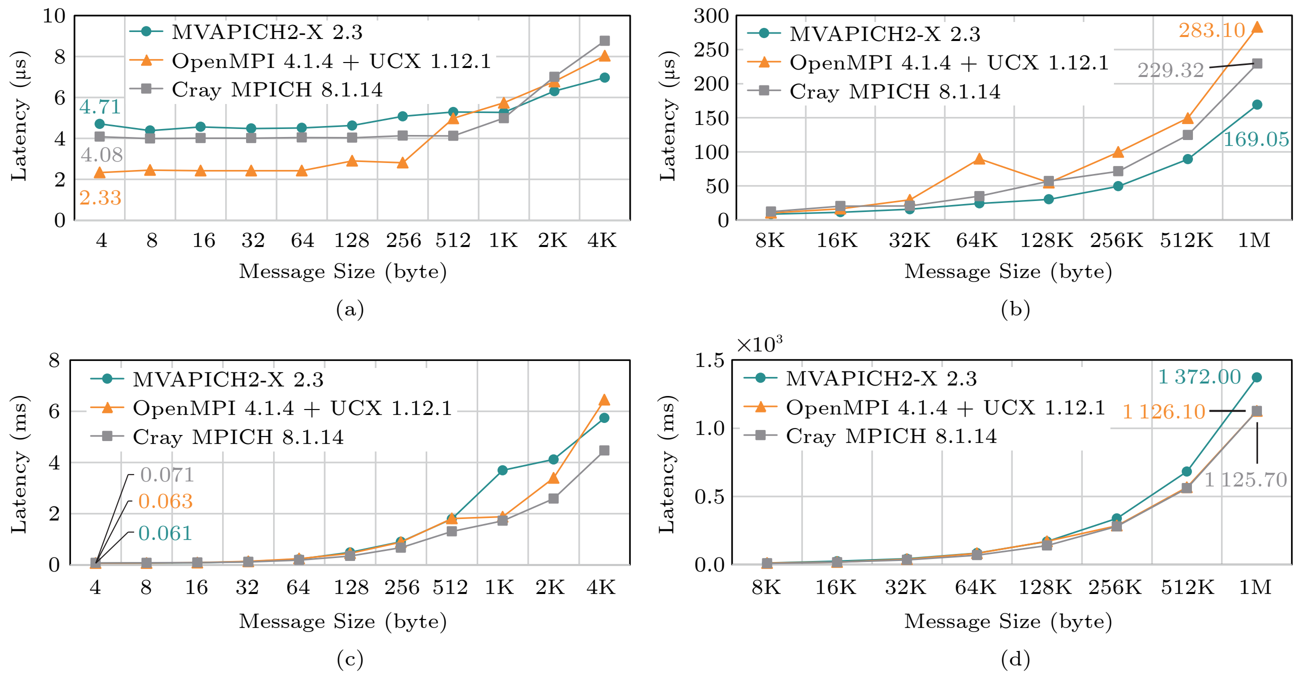


Fig.6. Performance of MPI collective MPI_Bcast and MPI_Alltoall operations on CPUs (512 CPUs—8 nodes & 64 PPN). (a) BCAST—small message sizes. (b) BCAST—large message sizes. (c) ALLTOALL—small message sizes. (d) ALLTOALL—large message sizes.

mance of the communication between two GPUs within the same node on the same socket, and two GPUs across nodes connected by the Slingshot-10 interconnect over the network. We also evaluate the performance of collective communication on the Spock system on up to 64 GPUs (16 nodes with four GPUs per node).

3.4.1 *Intra-Node Point-to-Point*

In Fig.7[1], we present an evaluation of intra-node point-to-point benchmark-level performance comparing MVAPICH2-GDR, OpenMPI + UCX, and Cray MPICH on AMD MI100 GPUs. The evaluation is done between two GPUs within one node for latency

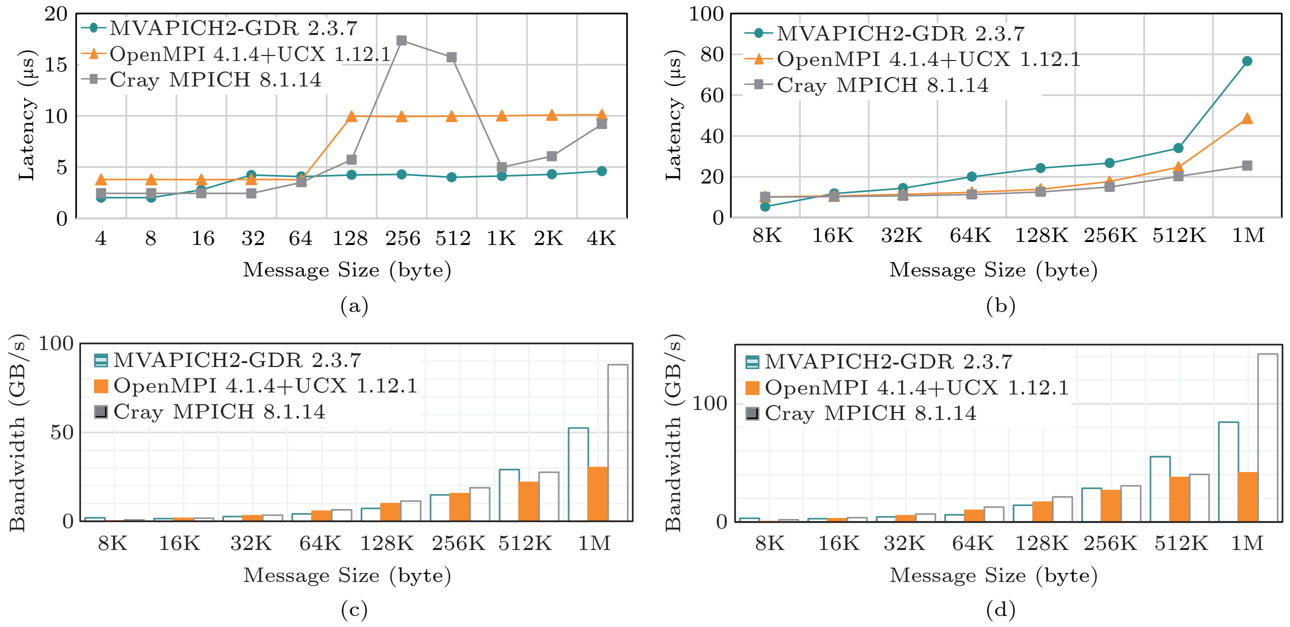


Fig.7. Intra-node point-to-point performance on GPUs on the Spock system over Infinity Fabric[1]. (a) Small message point-to-point latency. (b) Large message point-to-point latency. (c) Large message bandwidth. (d) Large message bi-directional bandwidth.

(*osu_latency*), bandwidth (*osu_bw*), and bi-directional bandwidth (*osu_bibw*). For small message latency shown in Fig.7(a), MVAPICH2-GDR, OpenMPI + UCX, and Cray MPICH achieve 2.01 µs, 3.79 µs, and 2.44 µs latency, respectively. This configuration involves two AMD MI100 GPUs within the same node, on the same socket, connected by Infinity Fabric. The trends in performance for intra-node communication between GPUs here reflect on protocols typically used for this configuration within MPI libraries such as: a GPU memory copy that utilizes the LargeBar feature of AMD GPUs and the ROCm driver for small message sizes, and ROCm IPC for larger message sizes[8]. The Infinity Fabric connection provides (46 + 46 GB/s) peak bandwidth. In Fig.7(c), MVAPICH2-GDR achieves a peak bandwidth at 1 MB of 52 GB/s, OpenMPI + UCX achieves 30 GB/s, and Cray MPICH achieves 88 GB/s.

3.4.2 Inter-Node Point-to-Point

In Fig.8[1] we present an evaluation of inter-node point-to-point benchmark-level performance comparing MVAPICH2-GDR, OpenMPI + UCX, and Cray MPICH on AMD MI100 GPUs. The evaluation is done between two GPUs on two different nodes connected by the Slingshot-10 interconnect for latency (*osu_latency*), bandwidth (*osu_bw*), and bi-directional bandwidth (*osu_bibw*). In Fig.8(a) and Fig.8(b), we see that MVAPICH2-GDR and Cray MPICH achieve 3.73 µs and 3.8 µs latency at 4 B and 115.94 µs and 154.09 µs at 1 MB, respectively. For small and medium message ranges, the performance here can be attributed to the use of the same underlying protocol with a GPU memory copy that utilizes the LargeBar feature of AMD GPUs and the ROCm driver that is discussed in Subsection 3.4.1. For larger message sizes, the MPI libraries rely on CPU-based staging mechanisms and GPU-Direct approaches to achieve near peak bandwidth performance. With this configuration over the Slingshot-10 interconnect, and 12.5 GB/s peak achievable bandwidth, MVAPICH2-GDR has peak uni-directional bandwidth performance at 32 KB with 11 GB/s performance, OpenMPI + UCX at 1 MB with 9.8 GB/s and Cray MPICH with 9.2 GB/s performance. In particular, we see lower bandwidth and bi-directional bandwidth for Cray MPICH in the message range between 8 KB and 512 KB as demonstrated in Figs.8(c) and 8(d).

3.4.3 Collective Operations

We evaluate various collective operations including MPI_Reduce and MPI_Allreduce (Fig.9), MPI_Gather and MPI_Allgather (Fig.10), and MPI_Bcast and MPI_Alltoall (Fig.11) using the OSU Micro-Benchmarks suite. Various tests are included here specific to each MPI operation. The per-

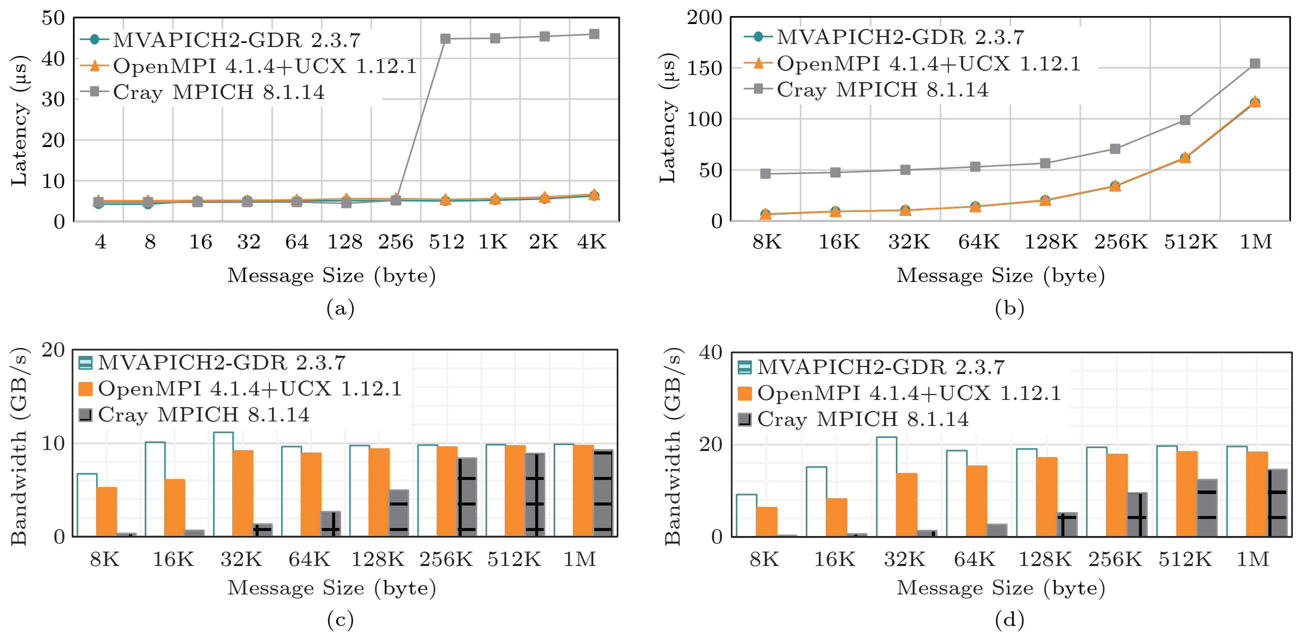


Fig.8. Inter-node point-to-point performance on GPUs on the Spock system over Slingshot-10[1]. (a) Small message point-to-point latency. (b) Large message point-to-point latency. (c) Large message bandwidth. (d) Large message bi-directional bandwidth.

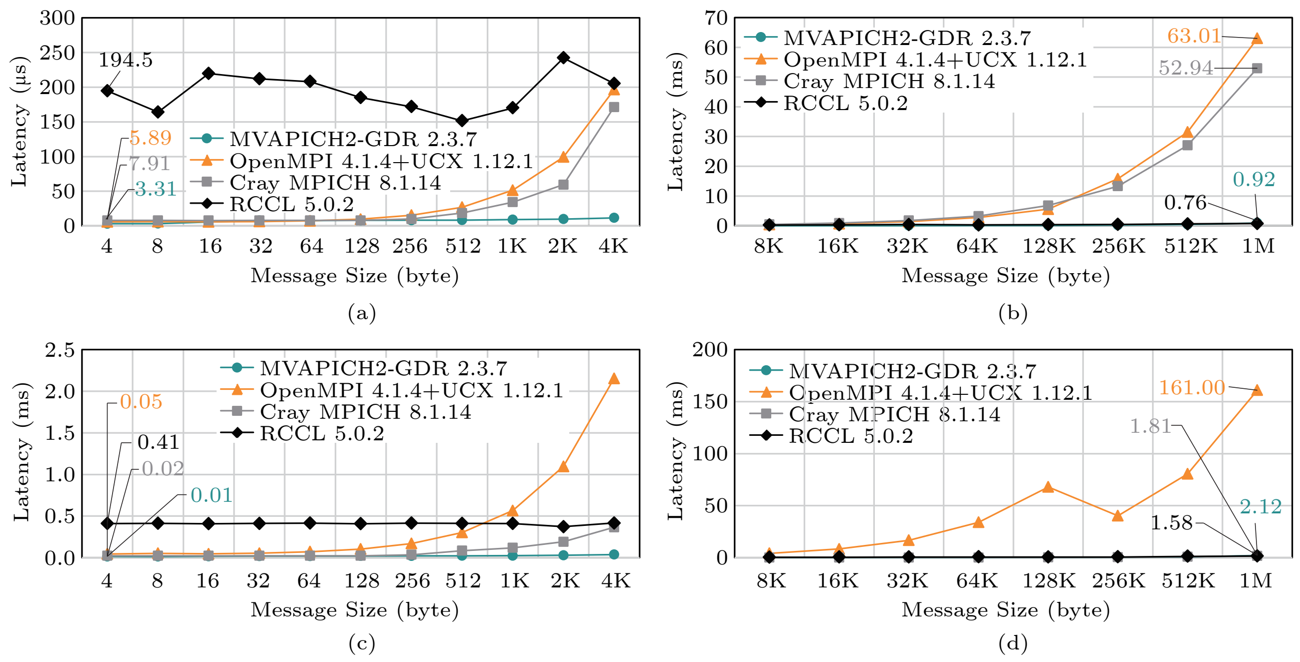


Fig.9. Performance of MPI collectives MPI_Reduce and MPI_Allreduce operations on 64 GPUs (64 GPUs—16 nodes & 4 PPN)[1]. (a) REDUCE—small message sizes. (b) REDUCE—large message sizes. (c) ALLREDUCE—small message sizes. (d) ALLREDUCE—large message sizes.

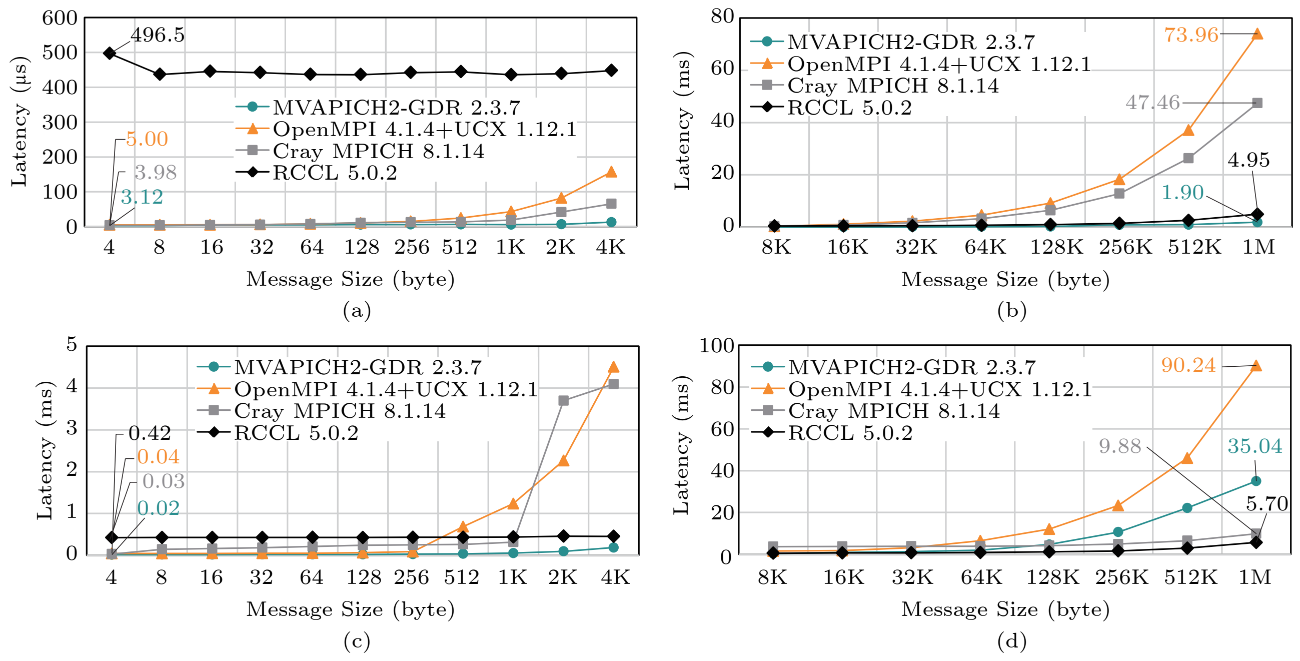


Fig.10. Performance of MPI collectives MPI_Gather and MPI_Allgather operations on 64 GPUs (64 GPUs—16 nodes & 4 PPN)[1]. (a) GATHER—small message sizes. (b) GATHER—large message sizes. (c) ALLGATHER—small message sizes. (d) ALLGATHER—large message sizes.

formance evaluation demonstrates a comparison between four different communication libraries (MVAPICH2-GDR, OpenMPI + UCX, Cray MPICH, and RCCL) on 64 AMD MI100 GPUs (16 nodes, 4 GPUs per node). In Figs.9–11[1], one particular trend we noticed is that RCCL performance is typically not optimal for smaller message sizes between 4 B and 4 KB, but RCCL performs well for large message allgather, and alltoall. For large message allreduce latency performance, MVAPICH2-GDR achieves 1.4 ms, OpenMPI + UCX achieves 160 ms, Cray MPICH demonstrates 1.8 ms, while RCCL performs at 1.5 ms. In

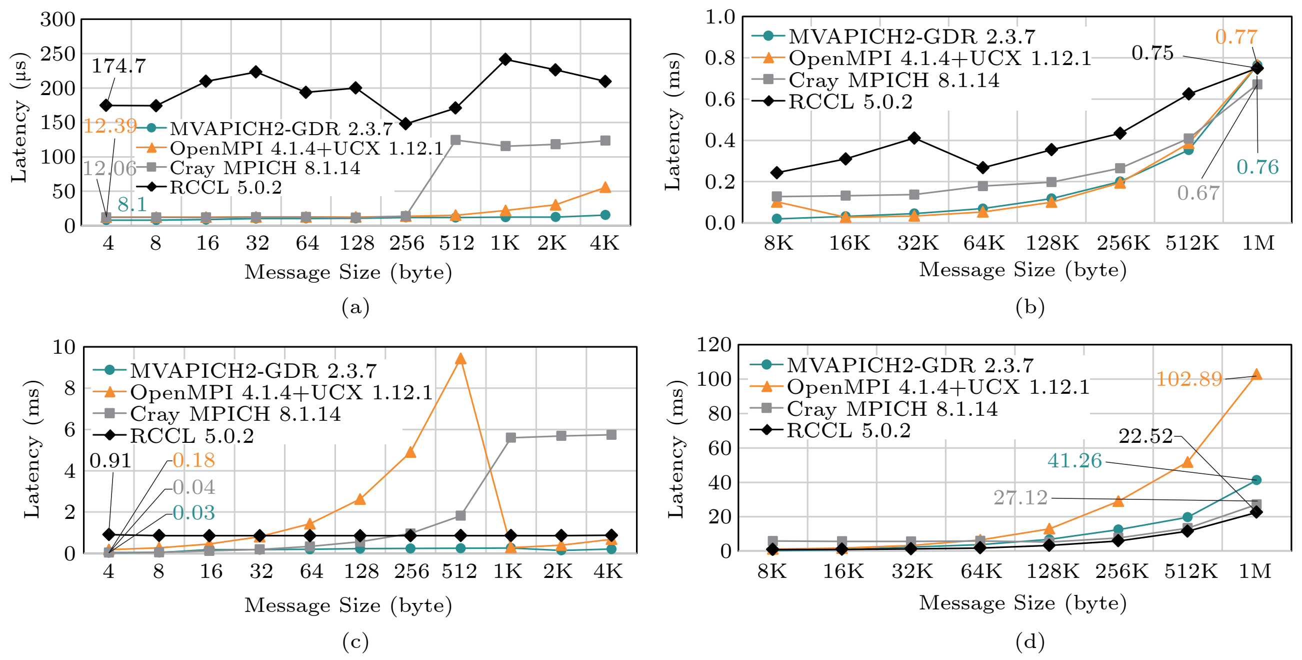


Fig.11. Performance of MPI collectives MPI_Bcast and MPI_Alltoall operations on 64 GPUs (64 GPUs—16 nodes & 4 PPN)[1]. (a) BCAST—small message sizes. (b) BCAST—large message sizes. (c) ALLTOALL—small message sizes. (d) ALLTOALL—large message sizes.

Fig.11(a), we demonstrate small message broadcast performance for each of the libraries with MVAPICH2-GDR at 8.1 μs, OpenMPI + UCX at 12.39 μs, Cray MPICH at 12.06 μs, and RCCL with 174.7 μs at 4 bytes.

We demonstrate the importance of efficient alltoall collective operation performance in Subsection 3.7 with the HeFFTe application which is heavily reliant on MPI_Alltoall or MPI_Alltoallv communication. In Fig.11(c), we evaluate the performance of small message GPU-aware alltoall performance for MVAPICH2-GDR at 27.09 μs, OpenMPI + UCX at 182.42 μs, Cray MPICH at 40.21 μs, and RCCL at 909.4 μs at 4 bytes.

Overall, the performance discrepancies presented here for different libraries can be a result of various components including, but not limited to: protocol changes, lack of tuning specific to a system or architecture, or underutilization of interconnect/link bandwidth. Through this evaluation, we highlight various areas that need to be optimized or accounted for in terms of communication performance. The difference between the peak achievable performance for MPI libraries compared to the available link bandwidth (provided by Infinity Fabric between GPUs and the Slingshot-10 network between nodes) demonstrates the importance of link utilization to take advantage of the vast performance made possible by these interconnects.

3.5 Slingshot-11 Networking System—System and Software Details

We utilize a system that emulates the Frontier system with Slingshot-11 networking to evaluate Slingshot-11 support of MPI libraries on CPUs. In these experiments, we evaluate MVAPICH2-3.0a with added support for the Slingshot-11 fabric and adapter, configured with the appropriate parameters demonstrated in Table 2[1].

3.6 Slingshot-11 Interconnect Evaluation

In Fig.12, we evaluate intra-node point-to-point performance between two AMD CPUs on the same node using MVAPICH2-3.0a and Cray MPICH 8.1.18 (the latest version of the library available on the system). Within a node, we demonstrate approximately 38 GB/s peak bandwidth and 0.53 μs minimum latency with MVAPICH2-3.0a and 18 GB/s peak bandwidth and 0.46 μs latency with Cray MPICH. The vast difference in peak achievable bandwidth between the two libraries can be attributed to the use of XPMEM (cross-partition memory) in MVAPICH2-X for large message sizes, allowing mapping the memory of one process into another process' virtual address space.

In Fig.13, we evaluate inter-node point-to-point performance between two AMD CPUs on different

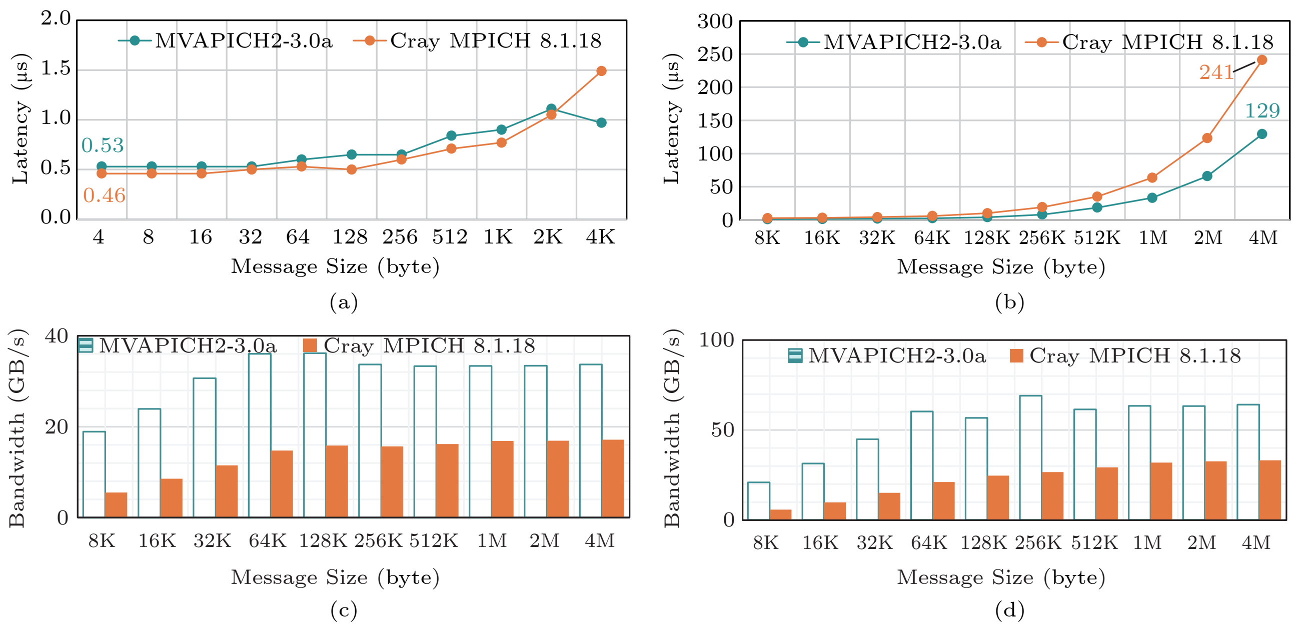


Fig.12. Intra-node point-to-point performance on CPUs on a system with Slingshot-11 networking across nodes. (a) Small message point-to-point latency. (b) Large message point-to-point latency. (c) Large message bandwidth. (d) Large message bi-directional bandwidth.

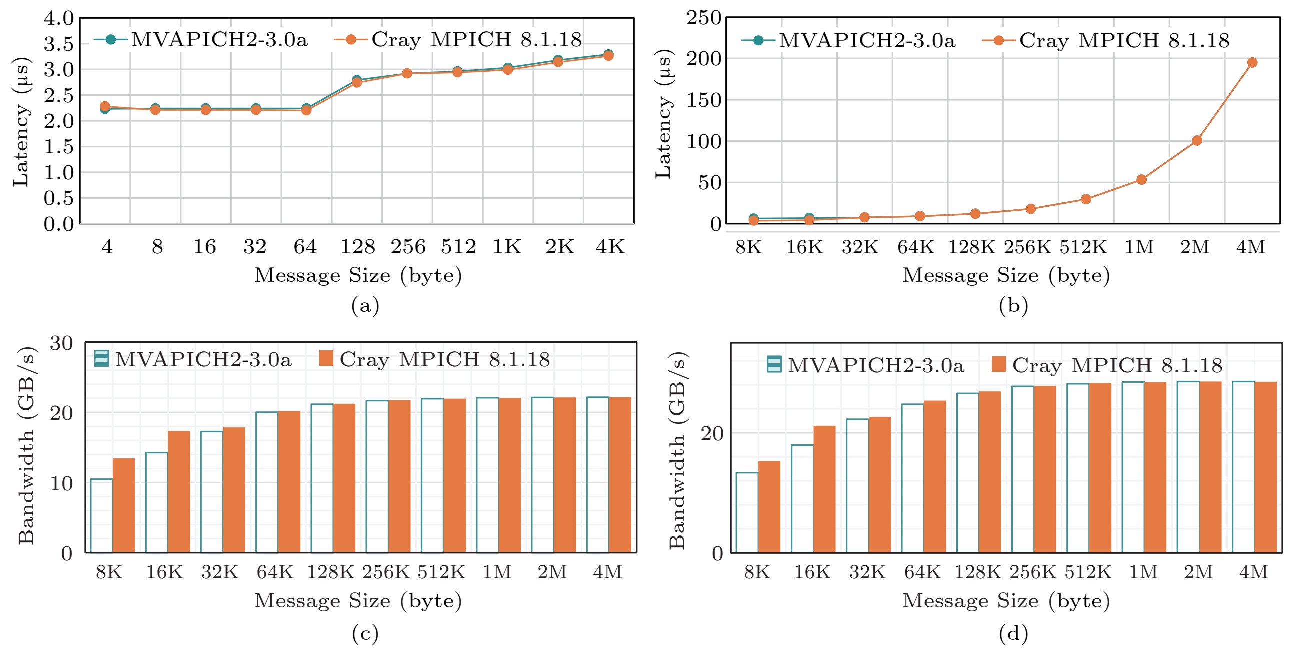


Fig.13. Inter-node point-to-point performance on CPUs on a system with Slingshot-11 networking across nodes. (a) Small message point-to-point latency. (b) Large message point-to-point latency. (c) Large message bandwidth. (d) Large message bi-directional bandwidth.

nodes over the Slingshot-11 network. Both libraries demonstrated here provide similar minimum latency at approximately 2.2 μs, and peak bandwidth of around 24 GB/s. A factor to be addressed by MPI libraries that is revealed through this evaluation is noted in the performance of bi-directional bandwidth across the network. While in Fig.12(d), we see that the peak achievable bi-directional bandwidth for intra-node communication is close to double the uni-directional bandwidth presented in Fig.12(c), and this is not the case for inter-node performance over Slingshot-11. In Fig.13(d), the peak bi-directional bandwidth is approximately 28.5 GB/s compared with a peak unidirectional peak bandwidth of 24 GB/s. In order to fully utilize the high bandwidth made possible through the Slingshot-11 interconnect, this performance discrepancy is one of the areas where performance can be addressed at the MPI layer to get closer to the peak ideal bandwidth.

3.7 Application-Level Evaluation

In this subsection, we evaluate the various MPI libraries at the application level. We use the heFFTe (highly efficient FFTs for Exascale)[14] application detailed below to demonstrate GPU-aware and CPU-based MPI libraries' performance. In this case, the datatype required by heFFTe is not supported by RCCL and therefore RCCL is not included in the evaluation below. Due to compilation issues at the application layer with Cray MPICH and cmake, Cray MPICH is also emitted from this evaluation. We use the rocHPCG application as well to compare the GPU-aware MPI libraries.

3.7.1 heFFTe

The heFFTe application is a highly efficient Fast Fourier transform (FFT) library for exascale systems. It uses GPU-aware MPI for communication and is provided as an open-source application. It provides the GPU kernel implementation with efficient scalability on large-scale clusters for 2-D and 3-D FFT libraries. Based on FFTMPI and SWFFT libraries, it presents so-called pencil-to-pencil methodology to compute 3-D FFT.

We evaluate the performance of the heFFTe application as a measure of GFlops/s with different problem sizes. The application can be run with either an alltoall-based or alltoallv-based problem. When running heFFTe on GPUs using GPU-aware MPI libraries, we utilize the rocFFT backend provided for the heFFTe benchmarks with support for ROCm. We demonstrate the performance of heFFTe on GPUs in Figs.14(c) and 14(d) for alltoall with 65 GFlops/s and alltoallv with 187 GFlops/s using MVAPICH2-GDR for a problem size of 512^3. This is in contrast to 3.17 GFlops/s and 3.28 GFlops with OpenMPI + UCX for alltoall and alltoallv, respectively, for the same problem size.

We utilize the fftw backend provided by heFFTe to run with CPU-based MPI libraries and demonstrate an evaluation using MVAPICH2-X and OpenMPI on the Spock system on 512 CPUs in Fig.15. With an alltoall problem and 512^3 problem size, we see 12.99 GFlops/s and 12.92 GFlops/s with MVAPICH2-X and OpenMPI, respectively. With an alltoallv problem and 256^3 problem size, we demonstrate 146 GFlops/s and 131 GFlops/s with MVAPICH2-X and OpenMPI, respectively.

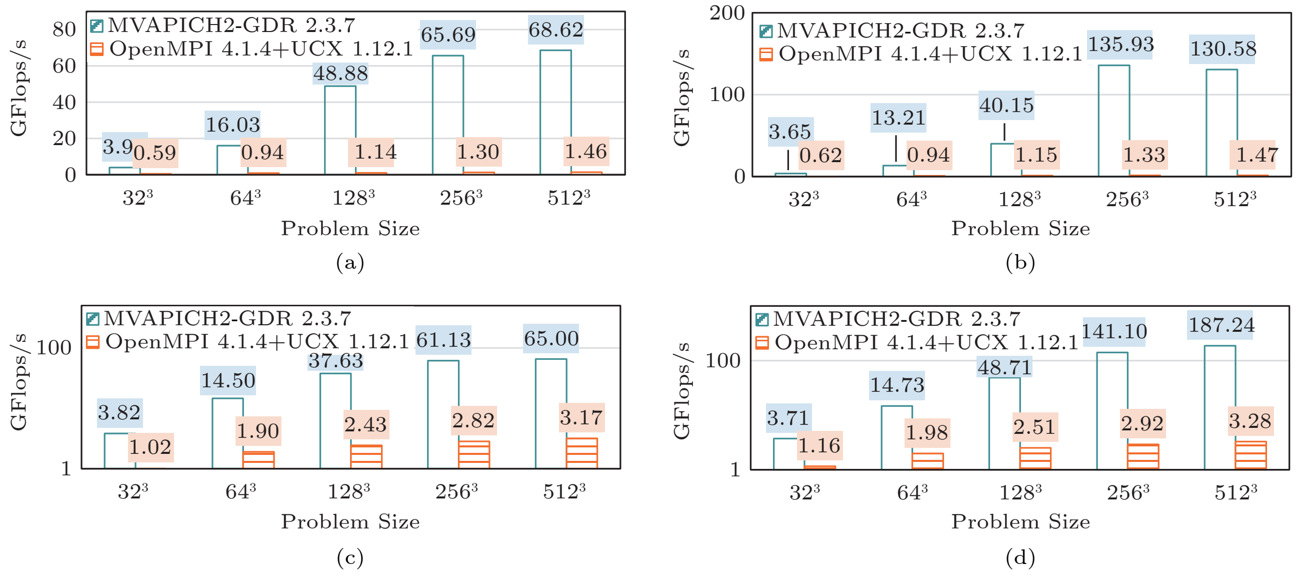
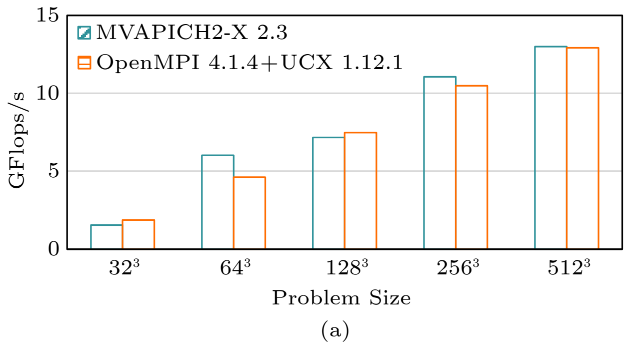


Fig.14. Performance of the heFFTe application using the rocfft backend for different problem sizes on 16 GPUs (4 nodes, 4 GPUs per node), and 32 GPUs (8 nodes, 4 GPUs per node)[1]. Two different communication methods are shown including MPI_Alltoall [-a2a] ((c)) and MPI_Alltoallv [-a2av] ((d)) using various MPI libraries including MVAPICH2-GDR, and OpenMPI + UCX. (a) heFFTe—16 GPUs (alltoall). (b) heFFTe—16 GPUs (alltoallv). (c) heFFTe—32 GPUs (alltoall). (d) heFFTe—32 GPUs (alltoallv).

[14]https://icl.utk.edu/files/publications/2020/icl-utk-1388-2020.pdf, Jan. 2023.



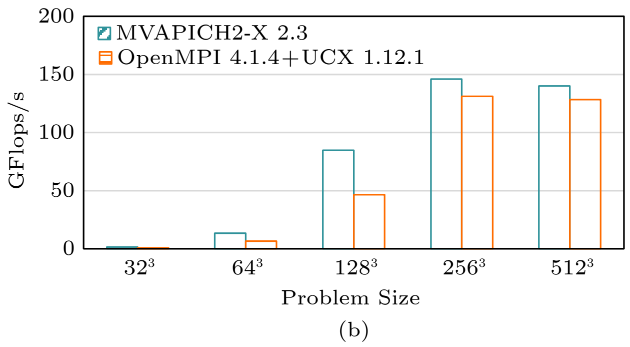


Fig.15. Performance of heFFTe application using the fftw backend for different problem sizes on 512 CPUs (8 nodes, 64 cores per node). Two different communication methods are shown including MPI_Alltoall [-a2a] ((a)) and MPI_Alltoallv [-a2av] ((b)) using various MPI libraries including MVAPICH2-X, and OpenMPI + UCX. (a) heFFTe—512 CPUs (alltoall). (b) heFFTe—512 CPUs (alltoallv).

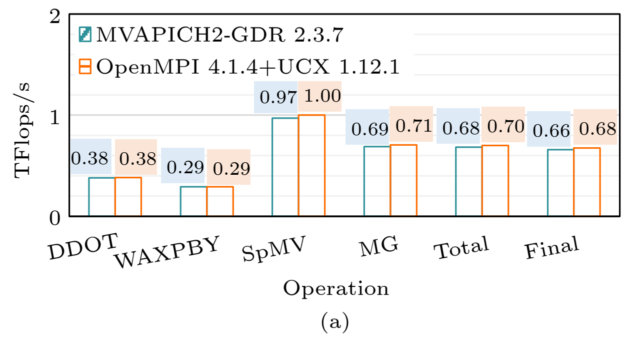
3.7.2 rocHPCG

rocHPCG[15] is a ROCm runtime benchmark based on the High-Performance Conjugate Gradients (HPCG) application for AMD GPUs. The HPCG benchmark is used as a metric for the Top500 systems since it simulates the computational and data-access patterns of a variety of scientific applications, and communication patterns, including MPI point-to-point and collective operations and OpenMP supports. rocHPCG consists of different sub-operation metrics, including global dot product (DDOT), vector update (WAXPBY), sparse matrix-vector multiplication (SpMV), multigrid preconditioner (MG), etc. We demonstrate the performance of each phase separately in the evaluation done in Fig.16 comparing MVAPICH2-GDR with OpenMPI + UCX.

4 Related Work

De Sensi *et al.*[9] proposed early research investigating Slingshot for large-scale computing systems. They described Slingshot as the next-generation large-scale system and summarized the key features as the following: high-radix Ethernet switches, adaptive routing, congestion control, and QoS management. They evaluated the system performance using Slingshot with both individual and concurrent workloads to close the real HPC system usage. They found less congestion on Slingshot and the control algorithm is effective for most HPC and data center applications. Also, a lower impact on performance from allocation policies was reported. Furthermore, Slingshot guarantees the bandwidth for jobs in different traffic classes.

The details of HPE Cray MPI are described in OLCF[16], including the latest implementation overview, HPE Cray MPI tuning and placement, GPU support, and its GPU-NIC asynchronous features. It also delves into the current support status with AMD and NVIDIA GPUs, including intra-node IPC and inter-node RDMA. Moreover, it introduces the GPU-NIC Async proposals, which decouples



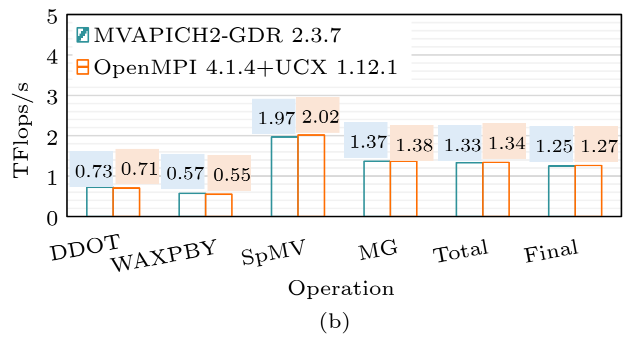


Fig.16. Performance of rocHPCG on 8 GPUs and 16 GPUs[1]. (a) rocHPCG—8 GPUs (2 nodes, 4 PPN). (b) rocHPCG—16 GPUs (4 nodes, 4 PPN).

[15]rocHPCG. https://github.com/ROCmSoftwarePlatform/rocHPCG, Dec. 2021.

[16]OLCF. HPE Cray MPI– Spock Workshop, 2021. https://www.olcf.ornl.gov/wp-content/uploads/2021/04/HPE-Cray-MPI-Update-nfr-presented.pdf, Dec. 2022.

CPU-GPU control and data paths to reduce the CPU-GPU synchronization frequency and overheads.

Melesse Vergara *et al.*[10] elaborated on their experience of porting the current kernels of main applications to a novel platform with AMD GPUs and HPE/Cray programming environment. They ported GENASIS, Minisweep, and Sparkler to the HIP-based kernel and compared the performance. The experience of porting applications from CUDA-based to HIP-based kernel proved that the porting procedure is easy, but there could be limitations, such as OpenMP support. In addition, additional tuning is required for fully utilizing the computing power on AMD GPUs. This work provides good examples for users to further port other kernel applications using HIP on AMD GPUs. Khorassani *et al.*[8] proposed an early research and designed a ROCm-aware MPI Library for the upcoming exascale systems, such as Frontier and El Capitan. They focused on Radeon Open Compute (ROCm) platform that adopts AMD GPUs. They utilized the ROCm features such as PeerDirect, ROCm-IPC, and large-BAR mapped memory to design a ROCm-aware MPI. An abstract communication layer with CUDA or ROCm backend allowed portability for the MPI runtime.

5 Conclusions

Next-generation exascale systems, and the first exascale and leading supercomputer in the world, Frontier, are equipped with nodes connected by the HPE Cray Slingshot interconnect. This interconnect technology is relatively new in the High-Performance Computing realm and is seldom evaluated at the communication layer. In this work, we delved into a comprehensive CPU-based and GPU-aware evaluation and analysis of various state-of-the-art MPI libraries including MVAPICH2, OpenMPI+UCX, Cray MPICH, and RCCL on a system, Spock, equipped with the Slingshot-10 interconnect to connect nodes over the network and with AMD MI100 GPUs and AMD Epyc Rome CPUs. We demonstrated the performance of various point-to-point communication operations for latency and bandwidth and various collective operations on AMD Rome CPUs and GPU-aware communication on AMD MI100 GPUs. The Slingshot-10 interconnect provides a 12.5 GB/s node injection bandwidth. MVAPICH2-X, OpenMPI+UCX, and Cray MPICH all achieve approximately 12.2 GB/s bandwidth at 1 MB for point-to-point operations between 2 CPUs across nodes. For inter-node GPU-aware communication with two GPUs across two nodes, MVAPICH2-GDR has peak uni-directional bandwidth performance at 32 KB with 11 GB/s performance, OpenMPI + UCX at 1 MB with 9.8 GB/s and Cray MPICH with 9.2 GB/s performance.

We then demonstrated preliminary support of MPI libraries over the Slingshot-11 interconnect, with both MVAPICH2-3.0a and Cray MPICH providing minimum latency at approximately 2.2 μs, and the peak bandwidth of around 24 GB/s. We also extended this evaluation using the heFFTe library and rocHPCG, demonstrating GPU-aware performance of heFFTe with alltoall, achieving 65 GFlops/s with MVAPICH2-GDR in contrast to 3.17 GFlops/s for OpenMPI + UCX for a problem size of 512^3.

In the future, we plan to extend this evaluation to cover additional applications with high demand for efficient communication performance, evaluate at a larger scale on a larger number of nodes based on system access, and ensure that state-of-the-art MPI and communication libraries provide the functionality, support, and efficiency that is to be expected with the growing demand and the roll-out of Slingshot-11 networking.

Acknowledgements We would like to thank Dr. Sameer Shende (University of Oregon) and Dr. Olga Pearce (Lawrence Livermore National Laboratory) for providing access to the HPC systems used in the paper. We would also like to thank Nat Shineman for his development work on the MVAPICH project to provide basic support and features over the underlying interconnect with MVAPICH.

References

[1] Khorassani K S, Chen C C, Ramesh B, Shafi A, Subramoni H, Panda D. High performance MPI over the Slingshot interconnect: Early experiences. In *Proc. the 2022 Practice and Experience in Advanced Research Computing*, Jul. 2022. DOI: 10.1145/3491418.3530773.

[2] Kim J, Dally W J, Scott S, Abts D. Technology-driven, highly-scalable dragonfly topology. In *Proc. the 2008 International Symposium on Computer Architecture*, Jun. 2008, pp.77–88. DOI: 10.1109/ISCA.2008.19.

[3] Gabriel E, Fagg G E, Bosilca G, Angskun T, Dongarra J J, Squyres J M, Sahay V, Kambadur P, Barrett B, Lumsdaine A, Castain R H, Daniel D J, Graham R L, Woodall T S. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proc. the 11th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*,

Sept. 2004, pp.97–104. DOI: 10.1007/978-3-540-30218-6_19.

[4] Thakur R, Rabenseifner R, Gropp W. Optimization of collective communication operations in MPICH. *International Journal of High Performance Computing Applications*, 2005, 19(1): 49–66. DOI: 10.1177/1094342005051521.

[5] Panda D K, Subramoni H, Chu C H, Bayatpour M. The MVAPICH project: Transforming research into high-performance MPI library for HPC community. *Journal of Computational Science*, 2021, 52: 101208. DOI: 10.1016/j.jocs.2020.101208.

[6] Bureddy D, Wang H, Venkatesh A, Potluri S, Panda D K. OMB-GPU: A micro-benchmark suite for evaluating MPI libraries on GPU clusters. In *Proc. the 19th European Conference on Recent Advances in the Message Passing Interface*, Sept. 2012, pp.110–120. DOI: 10.1007/978-3-642-33518-1_16.

[7] Chakraborty S, Bayatpour M, Hashmi J, Subramoni H, Panda D K. Cooperative rendezvous protocols for improved performance and overlap. In *Proc. the 2018 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2018, pp.361–373. DOI: 10.1109/SC.2018.00031.

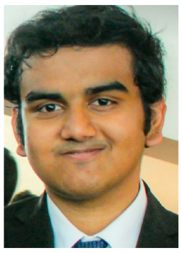
[8] Khorassani K S, Hashmi J, Chu C H, Chen C C, Subramoni H, Panda D K. Designing a ROCm-aware MPI library for AMD GPUs: Early experiences. In *Proc. the 36th International Conference on High Performance Computing*, Jun. 24–Jul. 2, 2021, pp.118–136. DOI: 10.1007/978-3-030-78713-4_7.

[9] De Sensi D, Di Girolamo S, McMahon K H, Roweth D, Hoefler T. An in-depth analysis of the Slingshot interconnect. In *Proc. the 2020 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2020. DOI: 10.1109/SC41405.2020.00039.

[10] Melesse Vergara V G, Budiardja R D, Joubert W. Early experiences evaluating the HPE/Cray ecosystem for AMD GPUs. U.S. Department of Energy, 2021. https://cug.org/proceedings/cug2021_proceedings/includes/files/pap108s2-file2.pdf, Jan. 2023.

Kawthar Shafie Khorassani is a Ph.D. student in the Department of Computer Science and Engineering at The Ohio State University, Columbus. She got her Bachelor's degree in mathematics and computer science at Wayne State University in Detroit, MI. She currently works in the Network Based Computing Laboratory on the MVAPICH2-GDR project. Her research interests lie in high-performance computing (HPC), and in GPU communication and computation.

Chen-Chun Chen is a Ph.D. student in the Department of Computer Science and Engineering at The Ohio State University, Columbus. He currently works on the MVAPICH2-GDR project. His research interests include high-performance computing (HPC) and GPU communication. He is a member of the Network Based Computing Laboratory (NBCL).


Bharath Ramesh is a Ph.D. student in the Department of Computer Science and Engineering at The Ohio State University, Columbus. His research interests include high-performance computing, architecture-aware communication, network/hardware-based offloading, and topology-aware collective algorithms. He is a graduate research associate in the Network-Based Computing Laboratory (NBCL), working on the MVAPICH2 (High-Performance MPI over InfiniBand, iWARP, and RoCE) library.

Aamir Shafi is currently a research scientist at The Ohio State University, Columbus, where he is involved in the High-Performance Big Data and Deep Learning projects. Dr. Shafi was a Fulbright Visiting Scholar at MIT where he worked on the award-winning Cilk technology. His research interests include architecting robust communication libraries and tools for HPC systems with emphasis on machine and deep learning applications. More details about Dr. Shafi are available from https://people.engineering.osu.edu/people/shafi.16.

Hari Subramoni is an assistant professor in the Department of Computer Science and Engineering at The Ohio State University, Columbus. His current research interests include high-performance interconnects and protocols, parallel computer architecture, network-based computing, exascale computing, network topology aware computing, QoS, poweraware LAN-WAN communication, fault tolerance, virtualization, deep learning, big data, and cloud computing. He has published over 120 papers in international journals and conferences related to these research areas. Recently, Dr. Subramoni is doing research and working on the design and development of MVAPICH2, MVAPICH2-GDR, and MVAPICH2-X software packages. He is a member of ACM and IEEE. More details about Dr. Subramoni are available from: https://web.cse.ohio-state.edu/~subramoni.1/.

Dhabaleswar K. Panda is a professor of computer science and engineering and University Distinguished Scholar at The Ohio State University, Columbus. His research interests include parallel computer architecture, high-performance networking, exascale computing, Big Data, Deep Learning, programming models, accelerators, high-performance file systems and storage, virtualization, and cloud computing. He has published over 500 papers in major journals and international conferences related to these research areas. Dr. Panda and his research group members have been doing extensive research on modern networking technologies including InfiniBand, High-Speed Ethernet, RDMA over Converged Enhanced Ethernet (RoCE), OmniPath, and EFA. Dr. Panda and his team have been actively working on high-performance MPI and PGAS libraries (http://mvapich.cse.ohio-state.edu), and Deep Learning libraries (http://hidl.cse.ohio-state.edu), and Big Data libraries (http://hibd.cse.ohio-state.edu). Dr. Panda serves as the director of the NSF-funded ICICLE AI Institute (icicle.ai). More details are available at http://www.cse.ohio-state.edu/~panda.