High Performance MPI over the Slingshot Interconnect: Early Experiences

Kawthar Shafie Khorassani
The Ohio State University
Columbus, USA
shafiekhorassani.1@osu.edu

Chen-Chun Chen
The Ohio State University
Columbus, USA
chen.10252@osu.edu

Bharath Ramesh
The Ohio State University
Columbus, USA
ramesh.113@osu.edu

Aamir Shafi
The Ohio State University
Columbus, USA
shafi.16@osu.edu

Hari Subramoni
The Ohio State University
Columbus, USA
subramon@cse.ohio-state.edu

Dhabaleswar K. Panda
The Ohio State University
Columbus, USA
panda@cse.ohio-state.edu

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 raises questions about performance, scalability, and any potential bottlenecks as they are a critical element contributing to the scalability across nodes on these systems. In this paper, we will delve into the challenges the slingshot interconnect poses with current state-of-the-art MPI 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-GDR on GPUs on the Spock system, an early access cluster deployed with Slingshot and AMD MI100 GPUs, to emulate the Frontier system.

KEYWORDS

Slingshot, AMD GPUs, Interconnect Technology, MPI.

ACM Reference Format:
Kawthar Shafie Khorassani, Chen-Chun Chen, Bharath Ramesh, Aamir Shafi, Hari Subramoni, and Dhabaleswar K. Panda. 2022. High Performance MPI over the Slingshot Interconnect: Early Experiences. In *Practice and Experience in Advanced Research Computing (PEARC '22), July 10–14, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3491418.3530773

1 INTRODUCTION

The Frontier Supercomputer [7] deployed at the Oakridge Leadership Computing Facility (OLCF), now leading the Top500 [5] 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 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 choosing to deploy the Slingshot interconnect 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 [11], [16].

In this paper, we provide an analysis of the performance of various MPI libraries on a system with preliminary/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 these systems. 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 [5] 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

PEARC '22, July 10–14, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9161-0/22/07…$15.00
https://doi.org/10.1145/3491418.3530773

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 system used in this study includes AMD MI100 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 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.
- Application-level evaluation using state-of-the-art communication libraries for rocHPCG and for the heFFTe application using the rocfft backend for AMD GPUs.
- Discuss the 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, in preparation for new exascale systems such as Frontier.

2 BACKGROUND

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, etc., 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 an increased number of choices and evolving interconnect standards.

2.2 Slingshot Interconnect

HPE Slingshot [11] 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 12.8Tb/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 [12] 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 works 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 multi-processing 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's 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 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, etc. With the expected growth in deployment of the Slingshot Interconnect across upcoming systems, this will be added to the growing list of features

that MPI libraries will need to add functionality and optimizations for. HPE 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 [11]. 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.

3 EVALUATION AND ANALYSIS

In this section, we provide details of the Spock system (Figure 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 System and Software Details

The performance evaluation is done on the Spock system deployed at the Oakridge Leadership Computing Facility(OLCF) [15]. This is an early access system provided in preparation for the exascale system, Frontier [7]. 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.

Table 1: Spock System Details and Usage

	Software	Version	Reference
MPI & Communication Libraries	Open MPI	4.1.4	[10]
	UCX	1.12.1	[6]
	Cray MPICH	8.1.14	[19]
	RCCL	5.0.2	[4]
	MVAPICH2-GDR	2.3.7	[17]
Platform	ROCm	5.0.2	[2]
Benchmarks & Applications	OSU Micro-benchmarks	5.9	[8]
	heFFTe	2.0	[1]

The Spock cluster consists of 64-core AMD EPYC 7662 Rome CPUs, and 4 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 Figure 1. More details of the communication libraries and software stack versions used on this system for this evaluation are provided in Table 1.

3.1.1 MPI Libraries —. Table 2 details the various MPI libraries used and configuration details specific to each of the libraries. The MVAPICH2-GDR library v2.3.7 was 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 required 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 the table below. Finally, the ROCm Collectives Communication Library (RCCL) was used as well in the evaluation of GPU-aware communication.

Table 2: MPI Libraries Configuration and Installation Details

Communication Libraries	Configuration & Installation Details
MVAPICH2-GDR 2.3.7	MVAPICH2-GDR 2.3.7 + ROCm 5.0.2 for GPUs *Run:* MV2_USE_ROCM=1
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 GPUs

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

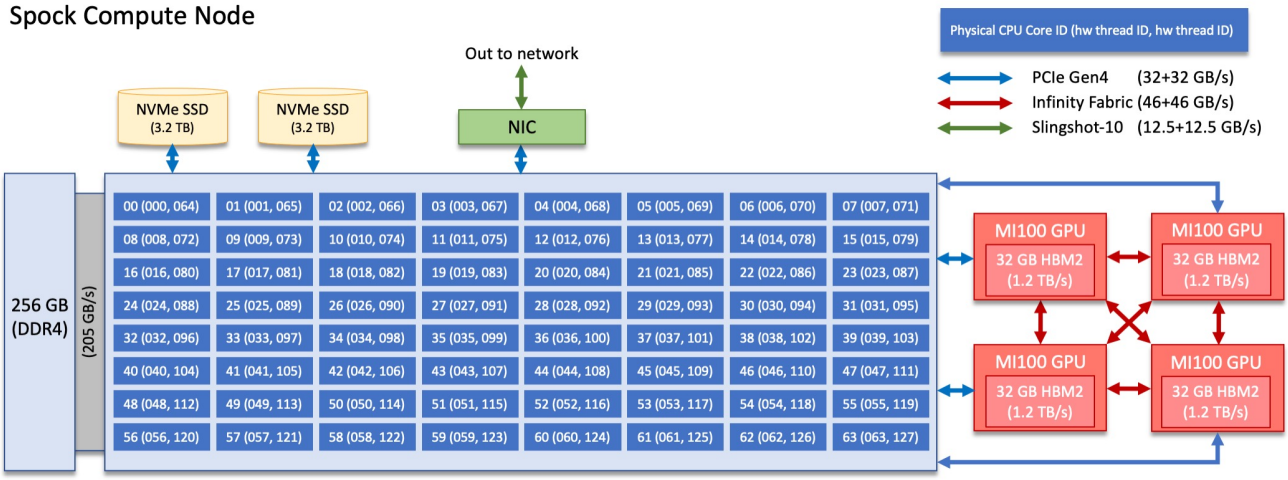
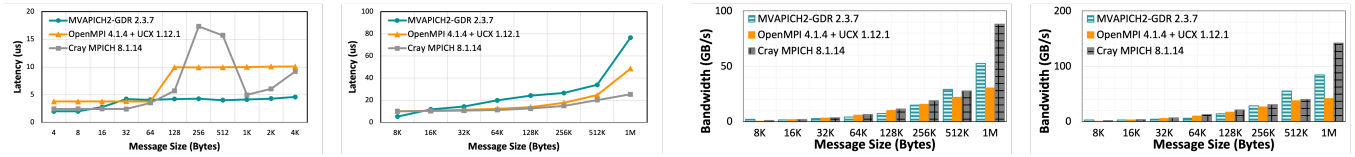
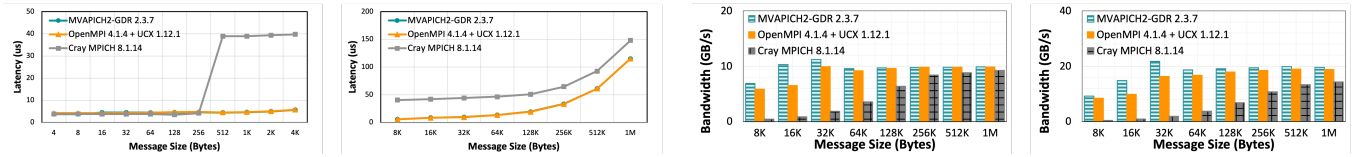


Figure 1: Spock Compute Node Details (Courtesy [16])



(a) Small Message Point-to-Point Latency (b) Large Message Point-to-Point Latency (c) Large Message Bandwidth (d) Large Message Bi-Directional Bandwidth

Figure 2: Intra-Node Point-to-Point Performance on GPUs over Infinity Fabric



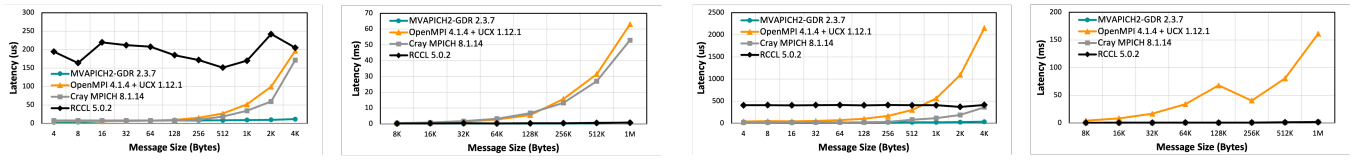
(a) Small Message Point-to-Point Latency (b) Large Message Point-to-Point Latency (c) Large Message Bandwidth (d) Large Message Bi-Directional Bandwidth

Figure 3: Inter-Node Point-to-Point Performance on GPUs over Slingshot-10 Interconnect

point-to-point performance of 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 4 GPUs per node).

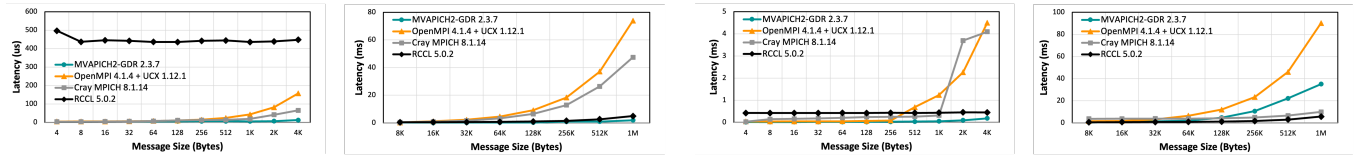
3.3.1 Intra-Node Point-to-Point —. In Figure 2, we present an evaluation of intra-node point-to-point benchmark-level performance comparing MVAPICH2-GDR, OpenMPI + UCX, and Cray MPICH on AMD MI100 GPUs. The evaluation is done between two GPUs within one node for latency (*osu_latency*), bandwidth (*osu_bw*), and bi-directional bandwidth (*osu_bibw*). For small message latency shown in Figure 2(a), MVAPICH2-GDR, OpenMPI + UCX, and Cray MPICH achieve 2.01 us, 3.79 us, and 2.44 us 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 reflects 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 [18]. The Infinity Fabric connection provides (46 + 46 GB/s) peak bandwidth. In Figure 2(c), MVAPICH2-GDR achieves a peak bandwidth at 1MB of 52 GB/s, OpenMPI + UCX achieving 30 GB/s, and Cray MPICH at 88 GB/s.

3.3.2 Inter-Node Point-to-Point —. In Figure 3 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 Figure 3(a) and Figure 3(b), we see that MVAPICH2-GDR and Cray MPICH achieve 3.73 us and 3.8 us latency at 4B and 115.26 us and 148.08 us at 1MB, respectively. With this configuration over the Slingshot-10 interconnect, with 12.5GB/s peak achievable bandwidth, MVAPICH2-GDR has peak



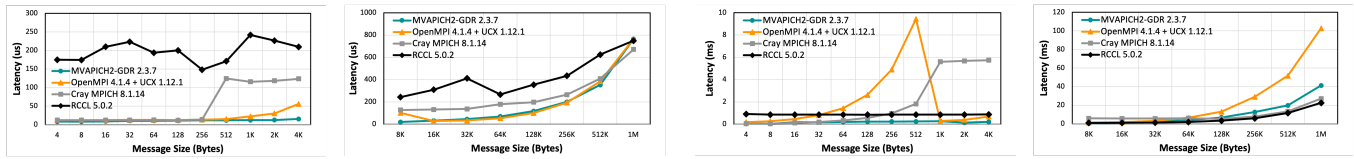
(a) REDUCE - Small Message Sizes (b) REDUCE - Large Message Sizes (c) ALLREDUCE - Small Message Sizes (d) ALREDUCE - Large Message Sizes

Figure 4: Performance of MPI Collectives MPI_Reduce and MPI_Allreduce Operations on 64 GPUs (16 Nodes, 4 GPUs Per Node)



(a) GATHER - Small Message Sizes (b) GATHER - Large Message Sizes (c) ALLGATHER - Small Message Sizes (d) ALLGATHER - Large Message Sizes

Figure 5: Performance of MPI Collectives MPI_Gather and MPI_Allgather Operations on 64 GPUs (16 Nodes, 4 GPUs Per Node)



(a) BCAST - Small Message Sizes (b) BCAST - Large Message Sizes (c) ALLTOALL - Small Message Sizes (d) ALLTOALL - Large Message Sizes

Figure 6: Performance of MPI Collectives MPI_Bcast and MPI_Alltoall Operations on 64 GPUs (16 Nodes, 4 GPUs Per Node)

uni-directional bandwidth performance at 32KB with 11 GB/s performance, OpenMPI + UCX at 1MB 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 8KB and 512 KB as demonstrated in Figures 3(c) and 3(d).

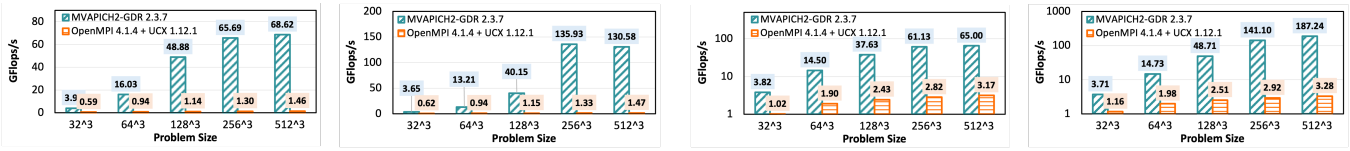
3.3.3 Collective Operations —. We evaluate various collective operations including MPI_Reduce, MPI_Allreduce (Figure 4), MPI_Gather, MPI_Allgather (Figure 5), MPI_Bcast, and MPI_Alltoall (Figure 6) using the OSU-Micro-benchmarks suite. Various tests are included here specific to each MPI operation. The performance 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 Figures 4, 5, and 6, one particular trend we noticed is that RCCL performance is typically not optimal for smaller message sizes between 4B-4KB, but 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 Figure 6(a), we demonstrate small message broadcast performance for each of the libraries with MVAPICH2-GDR at 8.1 us, OpenMPI + UCX at 12.39 us, Cray MPICH at 12.06 us, and RCCL with 174.7 us at 4 Bytes.

We demonstrate the importance of efficient Alltoall collective operation performance in Section 3.4 with the heFFTe application which is heavily reliant on MPI_Alltoall or MPI_Alltoallv communication. In figure 6(c), we evaluate the performance of small message GPU-aware Alltoall performance for MVAPICH2-GDR at 27.09 us, OpenMPI + UCX at 182.42 us, Cray MPICH at 40.21 us, and RCCL at 909.4 us 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. In particular, the difference between the peak achievable performance for MPI libraries compared to the available link bandwidth presented by Infinity Fabric between GPUs and the Slingshot-10 network between nodes demonstrates the importance of link utilization and taking advantage of the vast performance made possible by these interconnects.

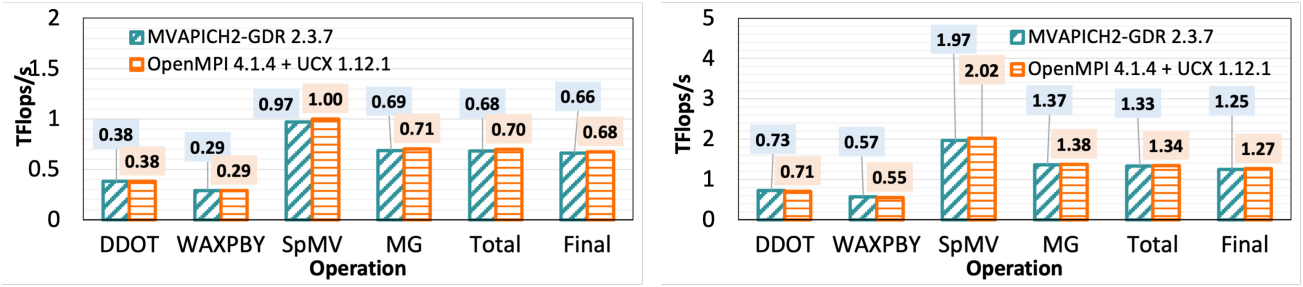
3.4 Application-Level Evaluation

In this section, we evaluate the various MPI libraries at the application level. We use the heFFTE application detailed below to demonstrate GPU-aware 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 CrayMPICH and cmake, CrayMPICH is also emitted from this evaluation. We use



(a) heFFTe - 16 GPUs (alltoall) (b) heFFTe - 16 GPUs (alltoallv) (c) heFFTe - 32 GPUs (alltoall) (d) heFFTe - 32 GPUs (alltoallv)

Figure 7: Performance of 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). Two different communication methods are shown including MPI_Alltoall [-a2a] (7(c)) and MPI_Alltoallv [-a2av] (7(d)) using various MPI libraries including MVAPICH2-GDR, and OpenMPI + UCX.



(a) rocHPCG - 8 GPUs (2 nodes, 4 ppn) (b) rocHPCG - 16 GPUs (4 nodes, 4ppn)

Figure 8: Performance of rocHPCG on 8 GPUs and 16 GPUs

the rocHPCG application as well to compare the GPU-aware MPI libraries.

3.4.1 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 Figures 7(c) and 7(d) for alltoall with 65 GFlops/s and alltoallv with 187 GFlops/s using MVAPICH2-GDR for a problem size of 512^3, in contrast to 3.17 GFlops/s and 3.28 GFlops with OpenMPI + UCX for altoall and alltoallv, respectively, for the same problem size.

3.4.2 rocHPCG —. rocHPCG [3] is a ROCm runtime benchmark based on the High-Performance Conjugate Gradients (HPCG) application for AMD GPUs. 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 Figure 8 comparing MVAPICH2-GDR performance with OpenMPI + UCX.

4 RELATED WORK

The HPE Cray Slingshot Interconnect will be deployed on the upcoming exascale systems. 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 [14], 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 introduced the GPU-NIC Async proposals, which decouples CPU-GPU control and data paths to reduce the CPU-GPU synchronization frequency and overheads.

Melesse Vergara et. al [13] 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. Plus, additional tuning is required for fully utilizing the computing power on AMD GPUs. This work provided good examples for users to further port other kernel applications using HIP on AMD GPUs. Shafie Khorassani et. al [18] 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 adaptability for the MPI runtime.

5 CONCLUSION

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 evaluation and analysis of various state-of-the-art MPI libraries including MVAPICH2-GDR, 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. We demonstrate 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. Due to the limitations of access to systems with the Slingshot interconnect arising from its relatively new introduction, and limited accessibility of early access systems that emulate the expected ecosystem of upcoming exascale systems, our evaluation is based on our early experiences with the system and with Slingshot-10 interconnect technology. 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 rollout of Slingshot-11 networking.

ACKNOWLEDGMENTS

We thank Dr. Sameer Shende (University of Oregon) for providing access to the Spock system. This research is supported in part by NSF grants #1818253, #1854828, #1931537, #2007991, and XRAC grant #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.

REFERENCES

[1] 2021. Highly Efficient FFT for Exascale (HeFFTe) library. https://github.com/af-ayala/heffte. Accessed: June 13, 2022.
[2] 2021. Radeon Open Compute (ROCm) Platform. https://rocmdocs.amd.com. Accessed: June 13, 2022.
[3] 2021. rocHPCG. https://github.com/ROCmSoftwarePlatform/rocHPCG. Accessed: June 13, 2022.
[4] 2021. ROCm Communication Collectives Library (RCCL). https://github.com/ROCmSoftwarePlatform/rccl. Accessed: June 13, 2022.
[5] 2021. TOP 500 Supercomputer Sites. http://www.top500.org.
[6] 2021. Unified Communication X. http://www.openucx.org/. Accessed: June 13, 2022.
[7] 2022. Frontier: ORNL's exascale supercomputer designed to deliver world-leading performance in 2021. https://www.olcf.ornl.gov/frontier/. Accessed: June 13, 2022.
[8] D. Bureddy, H. Wang, A. Venkatesh, S. Potluri, and D. K. Panda. 2012. OMB-GPU: A Micro-benchmark Suite for Evaluating MPI Libraries on GPU Clusters. In *Proceedings of the 19th European Conference on Recent Advances in the Message Passing Interface* (Vienna, Austria) *(EuroMPI'12)*. 110–120.
[9] Daniele De Sensi, Salvatore Di Girolamo, Kim H. McMahon, Duncan Roweth, and Torsten Hoefler. 2020. An In-Depth Analysis of the Slingshot Interconnect. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–14. https://doi.org/10.1109/SC41405.2020.00039
[10] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. 2004. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*. Budapest, Hungary, 97–104.
[11] HPE. 2022. HPE SLINGSHOT INTERCONNECT. https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.html. Accessed: June 13, 2022.
[12] John Kim, Wiliam J. Dally, Steve Scott, and Dennis Abts. 2008. Technology-Driven, Highly-Scalable Dragonfly Topology. In *2008 International Symposium on Computer Architecture*. 77–88. https://doi.org/10.1109/ISCA.2008.19
[13] Veronica Melesse Vergara, Reuben Budiardja, and Wayne Joubert. 2021. Early Experiences Evaluating the HPE/Cray Ecosystem for AMD GPUs. (7 2021). https://www.osti.gov/biblio/1817474
[14] OLCF. 2021. HPE CRAY MPI – SPOCK WORKSHOP. https://www.olcf.ornl.gov/wp-content/uploads/2021/04/HPE-Cray-MPI-Update-nfr-presented.pdf. Accessed: June 13, 2022.
[15] OLCF. 2022. Oakridge National Laboratory: Leadership Computing Facility. https://www.olcf.ornl.gov. Accessed: June 13, 2022.
[16] OLCF. 2022. Spock Quick-Start Guide. https://docs.olcf.ornl.gov/systems/spock_quick_start_guide.html. Accessed: June 13, 2022.
[17] Dhabaleswar Kumar Panda, Hari Subramoni, Ching-Hsiang Chu, and Mohammadreza Bayatpour. 2020. The MVAPICH project: Transforming research into high-performance MPI library for HPC community. *Journal of Computational Science* (2020), 101208. https://doi.org/10.1016/j.jocs.2020.101208
[18] Kawthar Shafie Khorassani, Jahanzeb Hashmi, Ching-Hsiang Chu, Chen-Chun Chen, Hari Subramoni, and Dhabaleswar K. Panda. 2021. Designing a ROCm-Aware MPI Library for AMD GPUs: Early Experiences. In *High Performance Computing*, Bradford L. Chamberlain, Ana-Lucia Varbanescu, Hatem Ltaief, and Piotr Luszczek (Eds.). Springer International Publishing, Cham, 118–136.
[19] Rajeev Thakur, Rolf Rabenseifner, and William Gropp. 2005. Optimization of Collective Communication Operations in MPICH. *The International Journal of High Performance Computing Applications* 19, 1 (2005), 49–66. https://doi.org/10.1177/1094342005051521 arXiv:https://doi.org/10.1177/1094342005051521