Performance Evaluation of MPI Libraries on GPU-enabled OpenPOWER Architectures: Early Experiences

Kawthar Shafie Khorassani, Ching-Hsiang Chu, Hari Subramoni, and Dhabaleswar K (DK) Panda

Department of Computer Science and Engineering, The Ohio State University, Columbus OH 43210, USA
{shafiekhorassani.1, chu.368}@osu.edu, {subramon, panda}@cse.ohio-state.edu

Abstract. The advent of Graphics Processing Unit (GPU)-enabled OpenPOWER architectures are empowering the advancement of various High-Performance Computing (HPC) applications from dynamic modular simulation to deep learning training. GPU-aware Message Passing Interface (MPI) is one of the most efficient libraries used to exploit the computing power on GPU-enabled HPC systems at scale. However, there is a lack of thorough performance evaluations for GPU-aware MPI libraries to provide insights into the varying costs and benefits of using each one on GPU-enabled OpenPOWER systems. In this paper, we provide a detailed performance evaluation and analysis of point-to-point communication using various GPU-aware MPI libraries including SpectrumMPI, OpenMPI+UCX, and MVAPICH2-GDR on OpenPOWER GPU-enabled systems. We demonstrate that all three MPI libraries deliver approximately 95% of achievable bandwidth for NVLink communication between two GPUs on the same socket. For inter-node communication where the InfiniBand network dominates the peak bandwidth, MVAPICH2-GDR and SpectrumMPI attain approximately 99% achievable bandwidth, while OpenMPI delivers close to 95%. This evaluation is useful to determine which MPI library can provide the highest performance enhancement.

Keywords: OpenPOWER · MPI · GPU · NVLink · RDMA.

1 Introduction

With an increasing demand for higher computing power for end applications, the adoption of GPU is becoming more prevalent in the HPC community [25]. This is an obvious trend in the recent Top500 supercomputer list [4], where 126 out of 500 supercomputers, i.e., 25.2%, are equipped with NVIDIA GPUs (8.4% higher than the previous year). This is further demonstrated by the fact that #1 Summit and #2 Sierra (as of Nov '18) are adopting a GPU-enabled OpenPOWER architecture with high-speed interconnects, including NVIDIA NVLink [12, 21] and InfiniBand networks [22].

On the software side, MPI is a standard programming model for developing parallel applications in the HPC community. The MPI standard provides high-level primitives for application developers to hide the complexity of handling data movement through various interconnects under different configurations, e.g., non-uniform memory access (NUMA) effect. Compute Unified Device Architecture (CUDA), which is an extension of C/C++, is the parallel computing platform to harness GPU's high-bandwidth memory (HBM) and massive parallelism. To efficiently perform parallel computing tasks on multiple GPU nodes, the concept of CUDA-aware MPI [28] has been introduced and widely adopted in many production MPI libraries. It is worth noting that an intelligent CUDA-aware MPI implementation, which leverages the cutting-edge hardware technologies, can *transparently* provide significant performance improvement to the end applications [23, 24]. As a result, the use of MPI for parallel applications significantly increases productivity and improves performance.

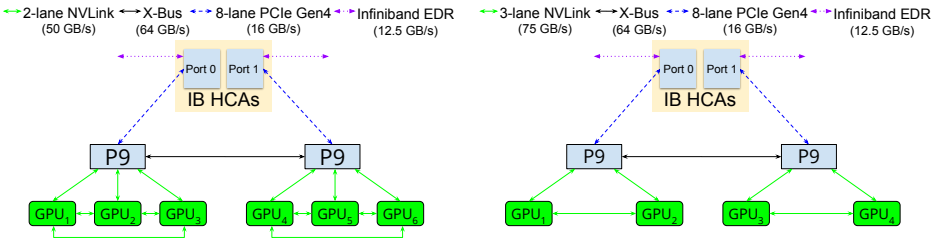
The advent of the GPU-enabled OpenPOWER systems not only brings new opportunities, but also introduces additional challenges due to the variety of interconnects. Many studies have been presented to provide the performance evaluation on an OpenPOWER system from different angles [6, 19, 9, 27]. However, the performance of CUDA-aware MPI libraries on GPU-enabled OpenPOWER architectures remains ambiguous due to the lack of thorough and comprehensive performance evaluations. The broad research question is: **Can the state-of-the-art MPI libraries fully leverage the various interconnects on GPU-enabled OpenPOWER architectures?** To the best of our knowledge, this is the first work to provide a comprehensive evaluation of multiple CUDA-aware MPI libraries and make the following contributions:

- Evaluate the state-of-the-art CUDA-aware MPI libraries in a systematic manner on Sierra-like and Summit OpenPOWER Systems
- Present comprehensive evaluation results with various configurations to understand the achievable performance of MPI libraries through different interconnects
- Provide insight into the expected performance of MPI libraries for the end applications

We elaborate on the importance and motivation of having a comprehensive evaluation of MPI performance on GPU-enabled OpenPOWER systems in Section 2. Section 3 describes the necessary background knowledge related to this work. The experimental setup is thoroughly detailed, including information on the hardware and software used in Section 4. To conclude, we present the evaluation results with various configurations and provide a thorough analysis of how the results compare to each other based on the context in sections 5 and 6.

2 Motivation

With advancement in technology and an ever more prevalent interest in parallel computing, it has become increasingly vital to optimize communication



(a) Summit - #1 Supercomputer (b) Sierra - #2 Supercomputer

Fig. 1. Hardware configuration of cutting-edge OpenPOWER GPU-enabled systems

libraries such as MPI. Optimizing MPI communication entails being able to handle upgraded performance requirements in an increasingly efficient manner. On a GPU-enabled OpenPOWER system, as exhibited in Figure 1, there are four primary interconnects being deployed: 1) NVLink between CPU and GPU and between GPUs, 2) X-Bus between two IBM POWER9 (P9) processes, i.e., NUMA nodes, 3) Peripheral Component Interconnect express (PCIe) between CPU to Mellanox InfiniBand EDR Host Channel Adapter (HCA), and 4) InfiniBand networks between multiple OpenPOWER nodes. This architecture with such powerful interconnects undeniably entails a performance boost [27]. On the other hand, it also poses a great challenge to MPI library developers to optimize communication protocols for the different data paths that can be selected between a pair of processes. The data path(s) selected varies in different use cases to provide the best performance.

Although previous studies provide a detailed evaluation of the achievable performance of various interconnects on the OpenPOWER systems [27, 19], it is unclear whether the state-of-the-art CUDA-aware MPI can achieve the peak performance that underlying interconnects provide when moving data within GPUs, between GPUs, between CPU and GPU and between GPU nodes. Table 1 presents the theoretical and achievable peak bandwidth of various interconnects on OpenPOWER systems (refer to Sections 4 and 5 for experimental configurations). Based on the table, the achievable bandwidths range from 85.43% to 94.56% of theoretical peak bandwidths due to the overhead of hardware, firmware and software protocols, and other factors like cache effect. It is critical to understand how much overhead the CUDA-aware MPI implementations have when using various interconnects and how it would reflect to the end applications. Through generating a comprehensive evaluation and analysis, we can develop knowledge about the MPI libraries that can obtain the highest bandwidth closest to the theoretical peak bandwidth of all interconnects in an OpenPOWER GPU-enabled system. By understanding the performance, restrictions, and drawbacks of different CUDA-aware MPI implementations, we generate a thorough analysis of the factors that need to be considered in future adjustments to the communication libraries and end applications.

Table 1. Theoretical and Achievable Peak Bandwidth of Data Movement over Interconnects in a Sierra-like and Summit OpenPOWER Systems

		Sierra-Like System		Summit			
	GPU HBM2	3-lane NVLink2 CPU-GPU	3-lane NVLink2 GPU-GPU	2-lane NVLink2 CPU-GPU	2-lane NVLink2 GPU-GPU	X-Bus	InfiniBand EDR
Theoretical Peak Bandwidth (Uni-directional)	900 GB/s	75 GB/s	75 GB/s	50 GB/s	50 GB/s	64 GB/s	12.5 GB/s
Achievable Peak Bandwidth (Uni-directional)	768.91 GB/s	68.78 GB/s	70.56 GB/s	45.9 GB/s	47 GB/s	58.01 GB/s	11.82 GB/s
Fraction of Peak	85.43%	91.7%	91.81%	91.8%	94%	90.64%	94.56%

3 Background

In this section, we describe background information related to the content of the analysis and evaluation performed.

3.1 GPU and NVIDIA GPUDIRECT Technology

General-purpose GPU has been widely adopted in the HPC community due to its high bandwidth memory and ultra-high throughput of computing. The cutting-edge NVIDIA Tesla Volta V100 GPU has 16GB HBM2 with theoretically 900 GB/s bandwidth [21] for 80 streaming multiprocessors and 4 copy engines. GPUDirect technology provided by NVIDIA enables faster handling of compute-intensive applications through the various features it provides. These features range from peer-to-peer memory access and transfers between GPUs to improving bandwidth and reducing latency through remote direct memory access (RDMA). GPUDIRECT RDMA allows the third-party devices, e.g., Mellanox InfiniBand HCA, to access GPU memory without intervention from the CPU, reducing the overhead of additional memory copies. Direct memory access enables copying data between the memory of GPUs on the same PCIe bus or NVLink. Through GPUDirect technology, a pinned buffer shared by the GPU and third party devices eliminates the need to copy memory multiple times in CUDA host memory[20].

3.2 Message Passing Interface

MPI is a programming paradigm often used in parallel applications that provides a mechanism for processes to communicate with each other. This communication can happen in different forms: point-to-point, one-sided, or collective. These various communication patterns involve a different number of processes communicating with each other and various restrictions on synchronization between the processes involved. The ability to apply the MPI standard to heterogeneous systems has evolved with support for inter-process and intra-process communication through the CUDA-aware feature. CUDA-aware MPI enables communication between the host and the GPU, further optimizing applications by introducing this level of GPU support. Some modern MPI libraries that provide support for CUDA-aware MPI include SpectrumMPI, OpenMPI (with and without UCX), and MVAPICH2.

SpectrumMPI, provided by IBM, is a default CUDA-aware MPI library deployed on many OpenPOWER systems including Summit and Sierra, as previously mentioned. This library takes advantage of various optimizing schemes such as GPUDirect RDMA and CUDA Inter-process communication (IPC) to enhance the efficiency of GPU based communication.

OpenMPI is a CUDA-aware library implementation of MPI with similar support for GPU based point-to-point and collective communication as noted above [13, 17]. **Unified Communication X (UCX)** is an open-source communication framework for HPC applications and also supports CUDA-aware point-to-point communication [5]. It was developed as a result of a collaboration between academia, government, and industry and presents an optimized communication layer for MPI. It is often recommended to build and use OpenMPI with UCX support for GPU-based communication.

MVAPICH2 is an MPI library implementation with support for Infiniband, Omni-Path, Ethernet/iWarp, and RoCE. Various versions of the library include additional features for a more specific application of the library [2]. MVAPICH2-GDR is optimized with features to support clusters with NVIDIA GPUs and is used for GPU-enabled HPC and deep learning applications [10, 11, 7]. It exploits the advantages of GPUDirect RDMA to optimize data movement between nodes and to offload communication between GPUs on clusters with NVIDIA GPUs [23, 24]. It is also enhanced with support for OpenPOWER with NVLink interconnect, CUDA-aware managed memory, and MPI-3 one-sided communication, among many features.

4 Experimental Setup

In this section, we elaborate on the hardware and software environment used in our evaluation. We also describe the methods used to evaluate MPI libraries in various configurations. The experiments were conducted on a GPU-enabled OpenPOWER system similar to the one presented in Figure 1(b). Each node is equipped with two NUMA nodes, where each one has a 22-core IBM POWER9 and 2 NVIDIA Volta GPUs. Each NUMA node has 128GB system memory, and each GPU has 16GB HBM. The nodes run Red Hat Enterprise Linux Server release 7.5 with a kernel version of 4.14.0-49.18.1. Mellanox OFED 4.3, NVIDIA driver version 418.39 and CUDA toolkit 9.2.148 are used on all nodes.

In this paper, we first present the achievable peak bandwidth of interconnects based on benchmarks using the low-level primitives. Next, we conduct a performance evaluation of CUDA-aware MPI libraries to perform the data movement through desired interconnects between MPI processes.

Evaluating Achievable Native Performance of Interconnects To build a proper baseline, i.e., upper bound, we use different software tools to obtain the achievable bandwidth of various interconnects as shown in Table 1. Each tool performs the corresponding data movement 1,000 times, and we report the average in this paper. The following points summarize the details of the tests:

1. NVLink between CPU and GPU: We used a modified *bandwidthTest* test from NVIDIA CUDA sample to obtain the bandwidth of NVLink between CPU and GPU by performing multiple *cudaMemcpyAsync* back-to-back between system memory to GPU memory (i.e., with copy type *cudaMemcpyDeviceToHost* and *cudaMemcpyHostToDevice*).
2. GPU HBM2 and NVLink between GPUs: Similarly, we used the *simpleIPC* test from NVIDIA CUDA sample to fork two CPU processes to perform data transfer within one GPU and between GPUs to measure the bandwidth of HBM2 and NVLink, respectively, (i.e., with copy type *cudaMemcpyDeviceToDevice* with CUDA P2P feature enabled).
3. X-Bus: STREAM benchmark [15] is used to measure the bandwidth of accessing system memory from the CPU. We manually bind the memory and CPU on different NUMA nodes by using *numactl* tool to measure the achievable bandwidth of X-Bus.
4. InfiniBand: We use an *ib_read_bw* test in InfiniBand Verbs Performance Tests [1] to measure the bandwidth of the IB network by moving data between two physical nodes. To have a comprehensive and fair analysis, we measured two data movement paths: 1) from system memory to remote system memory, and 2) from one GPU to remote GPU memory using GPUDirect RDMA technology.

Evaluating MPI-Level Performance The OSU Micro-benchmark (OMB) suite is a benchmark used to evaluate the performance of various MPI libraries. Bureddy et al. [8] extend OMB to support evaluating point-to-point, multi-pair, and collective communication on GPU clusters. This extended version of OMB includes latency, bandwidth, and bidirectional bandwidth benchmarks for point-to-point communication. Each of these tests takes two parameters to indicate the location of the buffers being passed into the communication at different processes. The buffer can either be on the device, i.e., GPU, or on the host. The various configurations of the buffer locations at each rank determine whether the benchmark is evaluating inter-node or intra-node communication on the host, on the device, or between the host and the device.

In this work, we use OMB v5.6.1 and focus on the evaluation of point-to-point communication and report the most representative metrics including latency, uni-directional bandwidth, and bi-directional bandwidth. The latency test is performed in a ping-pong manner by using MPI_Send and MPI_Recv primitives. In the uni-directional bandwidth configuration, a set number of back-to-back messages are sent from the sender by calling MPI_Isend. The sender then waits for a reply from the receiver after receiving all the messages by calling MPI_Waitall. To receive the data, it uses MPI_Irecv on the receiver process. The bi-directional bandwidth benchmark is different than the bandwidth benchmark in that it measures the maximum overall bandwidth between the two processes. It does this through sending back-to-back messages from both the sender and the receiver and waiting for both processes to send a reply only after obtaining all the messages.

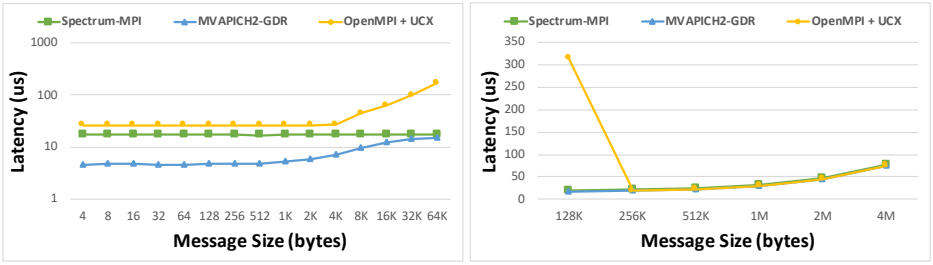
To evaluate data movement on different interconnects, we used the environment variable *CUDA_VISIBLE_DEVICES* to force MPI processes to select the desired GPU(s). This can be summarized as follows:

1. (Section 5.1) Evaluating NVLink between GPUs (NVLink GPU-GPU): This is a common case where each CPU process uses different physical GPUs, and NVLink is available between GPUs. We use *CUDA_VISIBLE_DEVICES=0,1* to make only two GPUs, which are connected by the NVLink, visible to MPI processes and processes can bind to different GPUs.
2. (Section 5.2) Evaluating HBM2 (GPU HBM2): Resulting from Multi-Process Service (MPS) capabilities in NVIDIA GPUs, multiple processes can be using the same GPU concurrently. This proves to be a benefit in scenarios where a single process cannot fully utilize the GPU compute capacity. To evaluate this scenario, we set the environment variable *CUDA_VISIBLE_DEVICES=0* to make only one GPU visible to all processes; this results in all MPI processes using the same GPU through MPS transparently.
3. (Section 5.3) Evaluating NVLink between CPU and GPU (NVLink CPU-GPU): In a heterogeneous system, applications may exploit both CPU and GPU to maximize the parallelism. This is where a CPU process may require transfer of data from system memory to a GPU owned by another CPU process. In this case, we place two MPI processes on the same NUMA node; thus, a NVLink is available between CPU and GPU.
4. (Section 5.4) Evaluating X-Bus and NVLink (X-Bus): In a multi-GPU system, communication between GPUs across NUMA node is inevitable. To evaluate this scenario, we use *CUDA_VISIBLE_DEVICES=0,3* to make two GPUs, which physically reside on different NUMA nodes, visible to the MPI processes.
5. (Section 5.5) Evaluating GPU transfer across nodes via InfiniBand network (Infiniband): MPI processes are launched on different nodes. By default, each MPI process selects the first discovered GPU in the same socket (GPU 0 in our case).

In this paper, we evaluated the following three CUDA-aware MPI libraries: 1) SpectrumMPI 10.3.0.01, which is the default library installed on the OpenPOWER system (labeled *SpectrumMPI*) [14], 2) OpenMPI 4.0.1 + UCX 1.6 (labeled *OpenMPI+UCX*) [3], and 3) MVAPICH2-GDR 2.3.2 pre-release version (labeled *MVAPICH2-GDR*) [2]. We ran the experiments with settings recommended by the user guides provided by the MPI libraries.

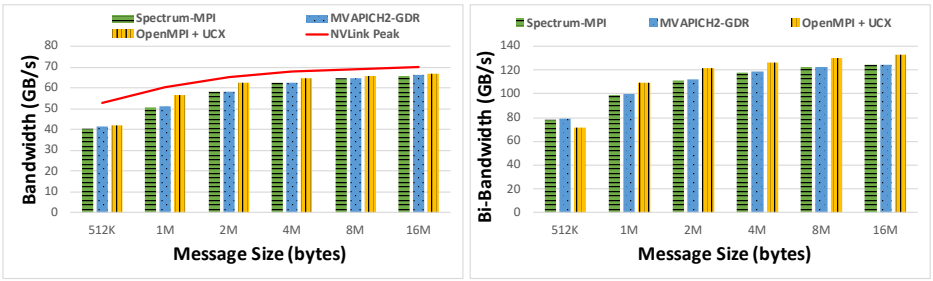
5 Evaluation and Analysis

In this section, we present the experimental results based on the environmental setup described in Section 4 and provide an analysis of these results. There are five primary scenarios presented: 1) Communication through NVLink between GPUs, 2) Communication through GPU HBM2, 3) Communication through



(a) Small Message Size Latency (b) Large Message Size Latency

Fig. 2. Latency comparison of MPI libraries on moving data between GPUs on the same socket (i.e., via NVLink interconnect) on a Sierra-like system



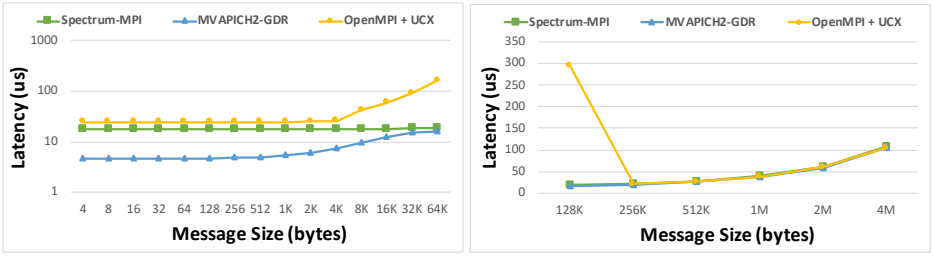
(a) Uni-Bandwidth (b) Bi-Bandwidth

Fig. 3. Bandwidth comparison of MPI libraries on moving data between GPUs on the same socket (i.e., via NVLink interconnect) on a Sierra-like system

NVLink between CPU and GPU, 4) Communication through NVLink and X-Bus, and 5) Communication through InfiniBand Network. Note that we only present the most representative results to avoid repetition.

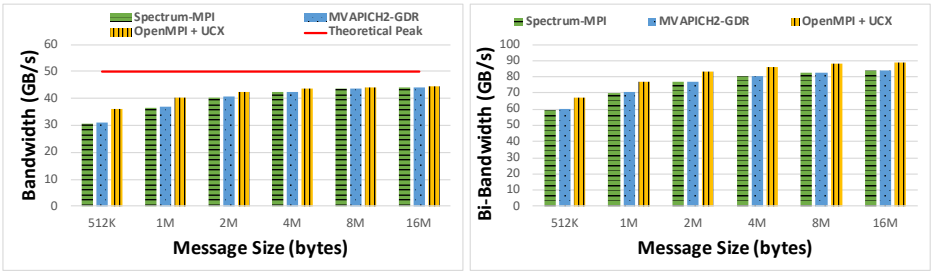
5.1 Communication through NVLink between GPUs

One MPI process using one single GPU is the most common configuration for MPI+CUDA applications. Here, we bind MPI processes to two GPUs with NVLink connection on the same socket to evaluate if MPI libraries can efficiently utilize the NVLink. Figure 2 shows a comparison of the MPI libraries through communication via the NVLink interconnect. MVAPICH2 outperforms SpectrumMPI by a factor of up to 4 for message sizes up to 32KB for latency and outperforms OpenMPI by a factor of up to 5. Both SpectrumMPI and MVAPICH2 libraries generally depict the same range for bandwidth and bi-bandwidth. These bandwidth numbers are close to the theoretical peak bandwidth of NVLink as shown in Figure 3. OpenMPI is relatively within a similar range for uni-bandwidth but slightly outperforms the other libraries for bi-bandwidth, whereas latency is higher than both comparing libraries from message sizes between 4 and 126KB. Similar trends are also observed on the Summit system for latency in Figure 4. In contrast to the 75 GB/s theoretical peak bandwidth on a Sierra-like OpenPOWER System, the theoretical peak



(a) Small Message Size Latency (b) Large Message Size Latency

Fig. 4. Latency comparison of MPI libraries on moving data between GPUs on the same socket (i.e., via NVLink interconnect) on Summit system



(a) Uni-Bandwidth (b) Bi-Bandwidth

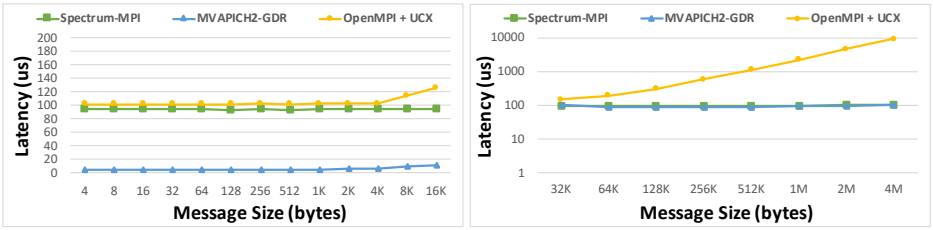
Fig. 5. Bandwidth comparison of MPI libraries on moving data between GPUs on the same socket (i.e., via NVLink interconnect) on Summit system

for uni-directional bandwidth on the Summit system is 50 GB/s as depicted in Figure 5.

5.2 Communication through GPU HBM2

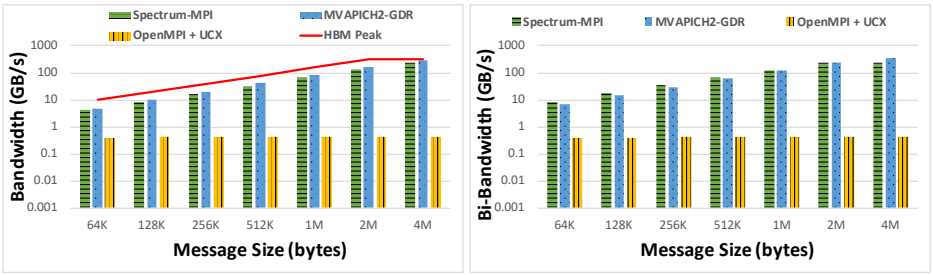
To evaluate the performance of MPI libraries when moving data within a GPU, i.e., through GPU's HBM2, we map two MPI processes into the same GPU. Figure 6 depicts a comparison of latency obtained from SpectrumMPI, OpenMPI, and MVAPICH2-GDR for moving data within a single GPU. MVAPICH2-GDR drastically outperforms SpectrumMPI for message sizes between 1 byte and 16KB then depicts similar latency behavior for larger message sizes. The latency provided by OpenMPI is also drastically higher than MVAPICH2 and SpectrumMPI for large message sizes. The same communication pattern is also compared for bandwidth and bidirectional bandwidth in Figure 7. MVAPICH2 outperforms SpectrumMPI for both bi-directional bandwidth and uni-directional bandwidth, while OpenMPI does not have comparable bandwidth numbers for any of the large message sizes. After profiling further to determine the cause of this performance, we found that OpenMPI does not use CUDA IPC for intra-node, intra-GPU communication, unlike MVAPICH2-GDR and SpectrumMPI. However, all MPI libraries can only achieve about half of the peak bandwidth provided by the GPU HBM2, which is 768.91 GB/s. We suspect this is due to the limitation of GPU's MPS feature when sharing a single GPU with multiple

processes, i.e., the bandwidth of GPU memory is also shared. Additionally, the bidirectional bandwidth is only slightly higher and sometimes lower than the uni-directional bandwidth because it already reaches the peak in one direction.



(a) Small Message Size Latency (b) Large Message Size Latency

Fig. 6. Latency comparison of MPI libraries on moving data within one single GPU on a Sierra-like system



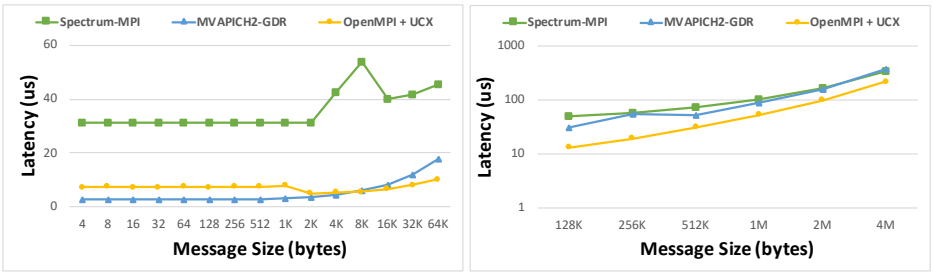
(a) Uni-Bandwidth (b) Bi-Bandwidth

Fig. 7. Bandwidth comparison of MPI libraries on moving data within one single GPU on a Sierra-like system

5.3 Communication through NVLink between CPU and GPU

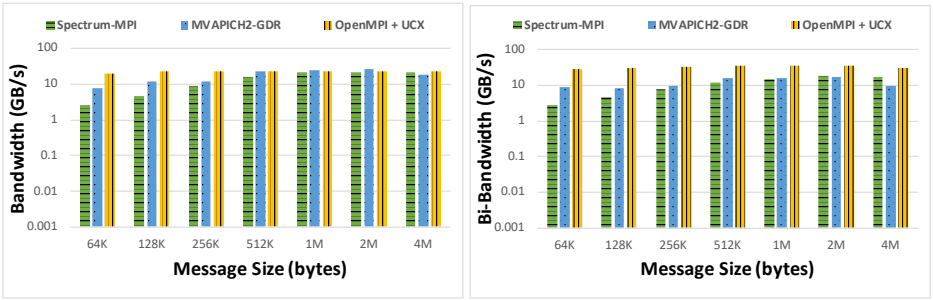
In a hybrid application, it is common to utilize the computing power of both CPU cores and GPUs. Therefore, data transfer from system memory to another process's GPU memory may be required. To evaluate such a scenario, we perform the tests on two MPI processes, where one process uses a communication buffer on system memory, and the communication buffer of another process is on GPU memory (i.e., Host-to-Device communication). Figure 8 provides a latency comparison of Host-to-Device communication between the MPI libraries. For small message sizes, MVAPICH-GDR and OpenMPI perform in a similar range while SpectrumMPI is up to 10× higher. SpectrumMPI also depicts a jump in latency between 2KB and 8KB message sizes.

As shown in Figure 9, the achievable uni-directional peak bandwidth of SpectrumMPI, OpenMPI, and MVAPICH2-GDR are 21.74 GB/s, 23.63 GB/s, and



(a) Latency Small Message Sizes (b) Latency Large Message Sizes

Fig. 8. Latency comparison of MPI libraries on moving data from the Host to the Device on a Sierra-like system



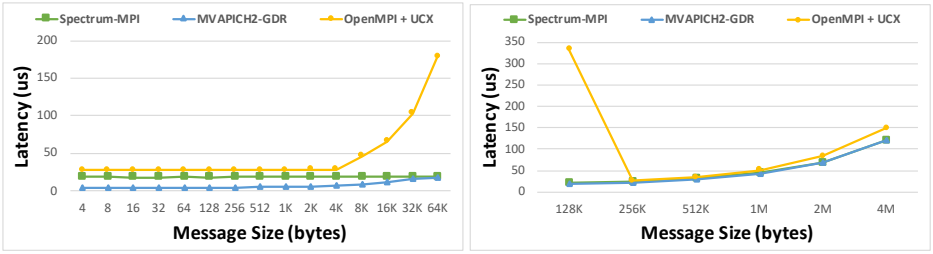
(a) Uni-Bandwidth (b) Bi-Bandwidth

Fig. 9. Bandwidth comparison of MPI libraries on moving data from the Host to the Device on a Sierra-like system. The peak achievable uni-directional bandwidth in this case is approximately 68.78 GB/s.

26.84 GB/s, respectively. Clearly, none of the MPI libraries efficiently utilize the NVLink between CPU and GPU as they are only able to attain between 31% to 39% achievable peak bandwidth. MVAPICH2-GDR may push the data through InfiniBand HCA as their peak uni- and bi-bandwidth are close to the peak bandwidth 8-lane PCIe Gen4 provides. Based on these results, we can conclude that the current CUDA-aware MPI libraries do not have good locality support to efficiently move data between CPU and GPU. Similar results can be expected for Device-to-Host communication, which is not shown to avoid repetition.

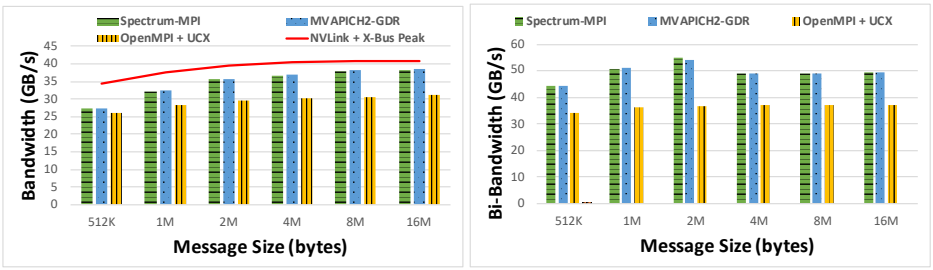
5.4 Communication through NVLink and X-Bus

Similar to Section 5.1, communication can happen between GPUs without direct connection via NVLink. On a GPU-enabled OpenPOWER system, the data needs to be moved not just through NVLink between CPU and GPU but also through X-Bus between the NUMA nodes. Figure 10 displays a latency comparison where the data is being moved between GPUs on different sockets. Compared to Figure 2, we can see that latency is similar for small messages but significantly higher for large messages. This indicates that the X-bus becomes the performance bottleneck when moving data across sockets. Figure 11 indicates that SpectrumMPI and MVAPICH2-GDR can achieve comparable uni-bandwidth



(a) Small Message Size Latency (b) Large Message Size Latency

Fig. 10. Latency comparison of MPI libraries on moving data between GPUs on different sockets on a Sierra-like system



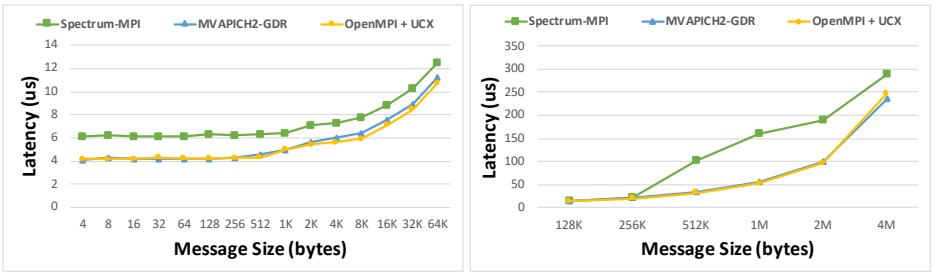
(a) Uni-Bandwidth (b) Bi-Bandwidth

Fig. 11. Bandwidth comparison of MPI libraries on moving data between GPUs on different sockets on a Sierra-like system

and bi-bandwidth, where OpenMPI is 25% lower than the peak bandwidth that can be attained. However, the peak bandwidth MPI libraries can achieve is only around 80% of achievable bandwidth of X-bus as exhibited in Table 1. We observed that it is because these MPI libraries rely on moving GPU-resident data using CUDA driver or runtime APIs. In this case, it is using CUDA IPC feature since peer-to-peer access is possible across the socket due to the Address Translation Service provided by NVLink2 technology. However, CUDA IPC can only achieve around 41 GB/s, which is only 64% of theoretical peak bandwidth, when data needs to be moved across NUMA-node. Taking this into account, SpectrumMPI and MVAPICH2-GDR are both achieving within a range of 80% to 95% of achievable peak bandwidth over the various message sizes shown. Nevertheless, the bidirectional bandwidth shown in Figure 11(b) indicates that MPI libraries are not able to fully utilize the bi-directional X-bus.

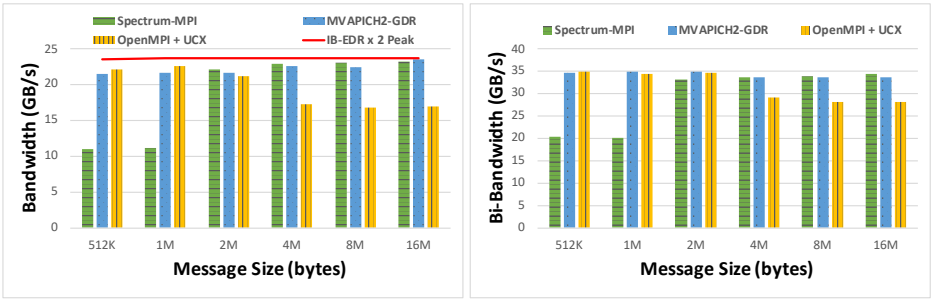
5.5 Communication through InfiniBand Network

Moving GPU-resident data across nodes is common for HPC applications in order to achieve higher performance at scale. It is critical to understand if MPI libraries can saturate the bandwidth provided by InfiniBand networks. Here, we conducted latency, uni- and bi-bandwidth tests between two GPU nodes. Figure 12 provides a latency comparison of inter-node communication between the MPI libraries. As can be seen, MVAPICH2-GDR and OpenMPI provide the



(a) Small Message Size Latency (b) Large Message Size Latency

Fig. 12. Latency comparison of MPI libraries on moving data between GPU Nodes on a Sierra-like system



(a) Uni-Bandwidth (b) Bi-Bandwidth

Fig. 13. Bandwidth comparison of MPI libraries on moving data between GPU nodes on a Sierra-like system

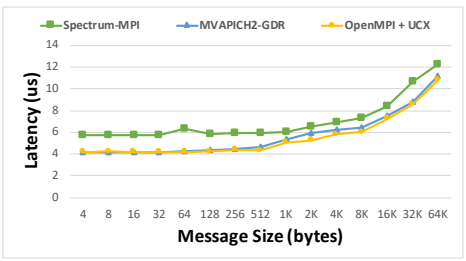
lowest latency across all message sizes. The latency of SpectrumMPI increases at 256KB after steadily maintaining latency about 30% higher than that of OpenMPI and MVAPICH2-GDR.

In terms of bandwidth as shown in Figure 13(a), it indicates that the libraries are leveraging both IB EDR adapters, i.e., so-called multi-rail support, efficiently in the system to achieve almost twice peak bandwidth of a single IB EDR, i.e., single-rail. The bandwidth and bi-bandwidth shown in Figure 13(b) reveals a performance degradation issue in OpenMPI when the message size is larger than 2MB. Spectrum-MPI achieves approximately 50% of the bandwidth achieved by OpenMPI and MVAPICH2-GDR until 1MB message size.

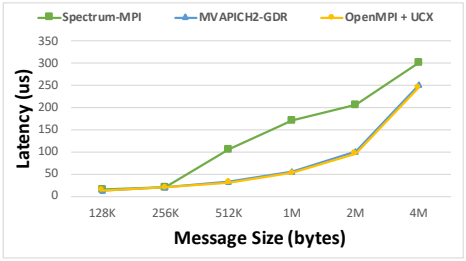
By default, multi-rail support is not enabled for GPU resident data when using OpenMPI+UCX, therefore, HCA selection is crucial at run-time in order to achieve comparable performance to that of MVAPICH2-GDR and SpectrumMPI.

6 Discussion

Based on the experiments presented above, Table 2 summarizes the peak bandwidth the CUDA-aware MPI libraries can achieve over various interconnects on

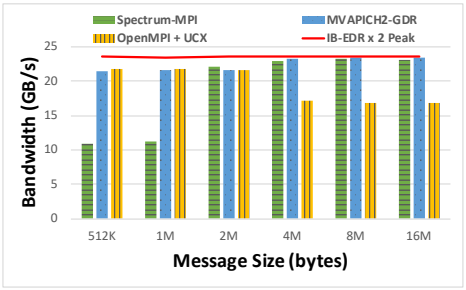


(a) Small Message Size Latency

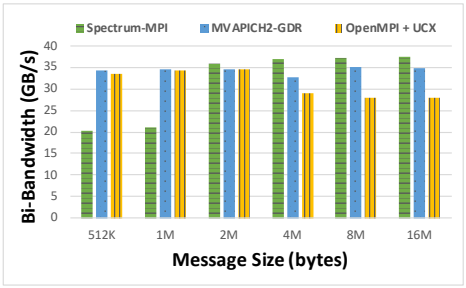


(b) Large Message Size Latency

Fig. 14. Latency comparison of MPI libraries on moving data between GPU Nodes on Summit system



(a) Uni-Bandwidth



(b) Bi-Bandwidth

Fig. 15. Bandwidth comparison of MPI libraries on moving data between GPU nodes on Summit system

a GPU-enabled OpenPOWER system. It is worth noting that the achievable peak bandwidths of HBM2 and X-Bus are further reduced due to the overhead or limitation of the CUDA driver when inter-process communication is involved, e.g., MPS and CUDA IPC.

As show in Table 2 SpectrumMPI, OpenMPI+UCX, and MVAPICH2-GDR provide 99.20%, 95.40%, and 99.70% achievable peak bandwidth, respectively, for inter-node communication by utilizing the achievable bandwidth of two IB EDR adapters. This can significantly improve the performance of HPC applications at scale. All three libraries achieve in the range of 31% to 39% of the NVLink which is available to be used when moving data between system and GPU memory. This is an open performance issue for all CUDA-aware MPI libraries. Finally, both SpectrumMPI and MVAPICH2-GDR outperform OpenMPI + UCX when the communication is limited by HMB2 or when the communication is through NVLink and X-Bus.

Through the results and analysis presented in Section 5, we highlight the several limitations of the existing MPI libraries on the OpenPOWER architectures as follows:

1. Limited bandwidth of HBM2 when sharing a memory
2. Host-to-Device and Device-to-Host communications are not efficiently utilizing NVLink
3. MPI Libraries are not able to fully utilize the bi-directional X-bus

Table 2. Summary of achievable peak bandwidth of MPI libraries and fraction of peak over Interconnects on a Sierra-like GPU-enabled OpenPOWER System

	GPU HBM2	3-lane NVLink2 CPU-GPU	3-lane NVLink2 GPU-GPU	X-Bus	InfiniBand EDR×2
SpectrumMPI	329 GB/s	21.74 GB/s	67.14 GB/s	39.16 GB/s	23.45 GB/s
	(36.55%)	(31.61%)	(95.20%)	(94.60%)	(99.20%)
OpenMPI+UCX	0.457 GB/s	23.63 GB/s	67.22 GB/s	31.77 GB/s	22.55 GB/s
	(0.05%)	(34.35%)	(95.40%)	(76.73%)	(95.40%)
MVAPICH2-GDR	390.88 GB/s	26.84 GB/s	67.15 GB/s	39.28 GB/s	23.56 GB/s
	(43.43%)	(39.02%)	(95.30%)	(94.97%)	(99.70%)

4. Multi-rail support is a pertinent feature for high-performance GPU-to-GPU communication

It is worth noting that these limitations may become the performance bottleneck of collective operations as many collectives are implemented based on the point-to-point primitives. To address these limitations, the CUDA-aware MPI libraries need to be further optimized with new features and designs under different communication patterns.

7 Related Work

In [18], Mojumder et al. generate a performance analysis of training Deep Neural Networks with peer-to-peer data transfer and with the NVIDIA Collective Communications library on a DGX-1 system. This analysis was used to identify any bottlenecks in the system architecture and to conclude that various factors such as the neural network architecture, and the GPU-to-GPU communication method can heavily influence performance. In the work done by Acun et al. [9], a parallel molecular dynamics application referred to as Nanoscale Molecular Dynamics (NAMD) is optimized for enhancement in performance on the IBM Newell platform (Power9 processors and NVIDIA Volta V100 GPUs). Various approaches were incorporated to achieve improved performance including improving GPU offload efficiency, load balancing, and vectorizing NAMD routines on Power9.

In [16], Li et al. evaluate the various modern GPU interconnects including: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect, on various systems to develop an analysis of their impact on multi-GPU application performance. They determine that GPU communication efficiency is heavily influenced by selecting a correct GPU combination. Talle et al. [26] contrast the performance of a PCIe based GPU interconnect with NVIDIA's NVLink interconnect to determine the performance impact each can entail. They use NVIDIA DGX-1 and Cirrascale GX8 to develop this comparison, leading them to conclude that DGX-1 is attributed with higher performance due to the additional links and higher per-link bandwidth associated with the NVLinks.

8 Conclusion

With enhancing GPU-enabled OpenPOWER architectures, HPC applications are achieving higher performance through exploiting the various features that enable such efficiency. CUDA-aware MPI is the communication standard that is able to exploit these features. In order to optimize HPC applications, take advantage of the enhanced support provided, and achieve peak performance, we need detailed evaluation of the most efficient communication libraries to use.

In this paper, the following three MPI libraries: SpectrumMPI, OpenMPI, and MVAPICH2-GDR were evaluated based on the latency, uni-directional bandwidth, and bi-directional bandwidth of point-to-point communication. The evaluation results show that all three MPI libraries deliver approximately 95% of achievable bandwidth for NVLink communication between two GPUs on the same socket. Most notably, for inter-node communication where the InfiniBand network determines the peak bandwidth, MVAPICH2-GDR and SpectrumMPI attain approximately 99% achievable bandwidth, while OpenMPI delivers close to 95%. Through our evaluation of these MPI libraries on GPU-enabled OpenPOWER architectures, we witnessed varying performance associated with each library based on the interconnects selected.

Finally, we have identified the following performance limitations for the state-of-the-art CUDA-aware MPI libraries: 1) communication between the CPU and GPU is not utilizing NVLink efficiently, 2) the bi-directional X-bus is not fully utilized by the MPI libraries, and 3) bandwidth is limited when MPI processes are sharing the GPU. In the future, we plan to conduct a comprehensive evaluation to include MPI collectives for GPU-resident data on OpenPOWER systems and their impact on applications.

References

1. Infiniband Verbs Performance Tests, `https://github.com/linux-rdma/perftest`, Accessed: August 8, 2019
2. MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE, `http://mvapich.cse.ohio-state.edu/features/`
3. Open MPI: Open Source High Performance Computing, `https://www.open-mpi.org`
4. TOP 500 Supercomputer Sites. http://www.top500.org
5. Unified Communication X. http://www.openucx.org/, Accessed: August 8, 2019
6. Ashworth, M., Meng, J., Novakovic, V., Siso, S.: Early application performance at the Hartree Centre with the OpenPOWER architecture. In: International Conference on High Performance Computing. pp. 173–187. Springer (2016)
7. Awan, A.A., Bédorf, J., Chu, C.H., Subramoni, H., Panda, D.K.: Scalable Distributed DNN Training using TensorFlow and CUDA-Aware MPI: Characterization, Designs, and Performance Evaluation. In: The 19th Annual IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing (CCGRID 2019) (2019)

8. 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: Proceedings of the 19th European Conference on Recent Advances in the Message Passing Interface (EuroMPI). pp. 110–120 (2012)
9. C. Pearson, I. Chung, Z.S.W.H., Xiong, J.: NUMA-aware Data-transfer Measurements for Power/NVLink Multi-GPU Systems,. In: International Workshop on OpenPOWER for HPC (IWOPH18) at the 2018 ISC High Performance Conference (2018)
10. Chu, C.H., Hamidouche, K., Venkatesh, A., Banerjee, D.S., Subramoni, H., Panda, D.K.: Exploiting Maximal Overlap for Non-Contiguous Data Movement Processing on Modern GPU-Enabled Systems. In: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 983–992 (May 2016)
11. Chu, C.H., Lu, X., Awan, A.A., Subramoni, H., Hashmi, J., Elton, B., Panda, D.K.: Efficient and Scalable Multi-Source Streaming Broadcast on GPU Clusters for Deep Learning. In: 46th International Conference on Parallel Processing (ICPP-2017) (Aug 2017)
12. Foley, D., Danskin, J.: Ultra-Performance Pascal GPU and NVLink Interconnect. IEEE Micro **37**(2), 7–17 (Mar 2017). https://doi.org/10.1109/MM.2017.37, https://doi.org/10.1109/MM.2017.37
13. 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: Proceedings, 11th European PVM/MPI Users' Group Meeting. pp. 97–104. Budapest, Hungary (September 2004)
14. IBM: IBM Spectrum MPI version 10.3 (2019), https://www.ibm.com/support/knowledgecenter/en/SSZTET_10.3/navigation/welcome.html, Accessed: August 8, 2019
15. John D. McCalpin: STREAM: Sustainable Memory Bandwidth in High Performance Computers (2019), https://www.cs.virginia.edu/stream/, Accessed: August 8, 2019
16. Li, A., Song, S.L., Chen, J., Li, J., Liu, X., Tallent, N.R., Barker, K.J.: Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect. CoRR **abs/1903.04611** (2019), http://arxiv.org/abs/1903.04611
17. Luo, X., Wu, W., Bosilca, G., Patinyasakdikul, T., Wang, L., Dongarra, J.: ADAPT: An Event-based Adaptive Collective Communication Framework. In: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing. pp. 118–130. HPDC '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3208040.3208054
18. Mojumder, S.A., Louis, M.S., Sun, Y., Ziabari, A.K., Abelln, J.L., Kim, J., Kaeli, D., Joshi, A.: Profiling DNN Workloads on a Volta-based DGX-1 System. In: 2018 IEEE International Symposium on Workload Characterization (IISWC). pp. 122–133 (Sep 2018). https://doi.org/10.1109/IISWC.2018.8573521
19. Moreno, R., Arias, E., Navarro, A., Tapiador, F.J.: How good is the OpenPOWER architecture for high-performance CPU-oriented weather forecasting applications? The Journal of Supercomputing (Apr 2019). https://doi.org/10.1007/s11227-019-02844-3, https://doi.org/10.1007/s11227-019-02844-3
20. NVIDIA: NVIDIA GPUDirect, https://developer.nvidia.com/gpudirect, Accessed: August 8, 2019
21. NVIDIA: NVIDIA TESLA V100 GPU ARCHITECTURE (2019), https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf, Accessed: August 8, 2019

22. Pfister, G.F.: An Introduction to the Infiniband Architecture. High Performance Mass Storage and Parallel I/O **42**, 617–632 (2001)
23. Potluri, S., Hamidouche, K., Venkatesh, A., Bureddy, D., Panda, D.K.: Efficient Inter-node MPI Communication Using GPUDirect RDMA for InfiniBand Clusters With NVIDIA GPUs. In: Parallel Processing (ICPP), 2013 42nd International Conference on. pp. 80–89. IEEE (2013)
24. Shi, R., Potluri, S., Hamidouche, K., Perkins, J., Li, M., Rossetti, D., Panda, D.K.: Designing Efficient Small Message Transfer Mechanism for Inter-node MPI Communication on InfiniBand GPU Clusters. In: 2014 21st International Conference on High Performance Computing (HiPC). pp. 1–10 (Dec 2014)
25. Stone, J.E., Hynninen, A.P., Phillips, J.C., Schulten, K.: Early Experiences Porting the NAMD and VMD Molecular Simulation and Analysis Software to GPU-Accelerated OpenPOWER Platforms. In: Taufer, M., Mohr, B., Kunkel, J.M. (eds.) High Performance Computing. pp. 188–206. Springer International Publishing, Cham (2016)
26. Tallent, N.R., Gawande, N.A., Siegel, C., Vishnu, A., Hoisie, A.: Evaluating On-Node GPU Interconnects for Deep Learning Workloads. In: Jarvis, S., Wright, S., Hammond, S. (eds.) High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation. pp. 3–21. Springer International Publishing, Cham (2018)
27. Vazhkudai, S.S., de Supinski, B.R., Bland, A.S., Geist, A., Sexton, J., Kahle, J., Zimmer, C.J., Atchley, S., Oral, S., Maxwell, D.E., Larrea, V.G.V., Bertsch, A., Goldstone, R., Joubert, W., Chambreau, C., Appelhans, D., Blackmore, R., Casses, B., Chochia, G., Davison, G., Ezell, M.A., Gooding, T., Gonsiorowski, E., Grinberg, L., Hanson, B., Hartner, B., Karlin, I., Leininger, M.L., Leverman, D., Marroquin, C., Moody, A., Ohmacht, M., Pankajakshan, R., Pizzano, F., Rogers, J.H., Rosenburg, B., Schmidt, D., Shankar, M., Wang, F., Watson, P., Walkup, B., Weems, L.D., Yin, J.: The Design, Deployment, and Evaluation of the CORAL Pre-exascale Systems. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis. pp. 52:1–52:12. SC '18, IEEE Press, Piscataway, NJ, USA (2018), `http://dl.acm.org/citation.cfm?id=3291656.3291726`
28. Wang, H., Potluri, S., Bureddy, D., Rosales, C., Panda, D.K.: GPU-Aware MPI on RDMA-Enabled Clusters: Design, Implementation and Evaluation. IEEE Transactions on Parallel and Distributed Systems **25**(10), 2595–2605 (Oct 2014). https://doi.org/10.1109/TPDS.2013.222