Designing the HPE Cray Message Passing Toolkit Software Stack for HPE Cray EX Supercomputers

K. Kandalla, K. McMahon, N. Ravi, T. White, L. Kaplan, and M. Pagel
Hewlett Packard Enterprise Inc

{kkandalla, kim.mcmahon, nravi, trey.white, laurence.kaplan, mark.pagel}@hpe.com

***Abstract*—The Frontier supercomputer at ORNL, based on the HPE Cray EX architecture, is the world's first supercomputer to break the exascale barrier. The HPE Cray EX architecture is designed to be highly flexible and relies on the HPE Slingshot technology for its high speed interconnect. HPE Cray Programming Environment is a key software component that is tightly integrated with the broader HPE Cray EX hardware and software ecosystem to offer high performance and improved programmer productivity on HPE Cray EX systems. HPE Cray Messaging Passing Toolkit is one of the building blocks of the HPE Cray Programming Environment. It is comprised of HPE Cray MPI and HPE Cray OpenSHMEMX software stacks. HPE Cray MPI is a proprietary implementation of the MPI specification (based on ANL MPICH 3.4a2) and it is the primary MPI stack on HPE Cray EX supercomputers. It was instrumental in surpassing the exascale performance barrier on the Frontier supercomputer. HPE Cray OpenSHMEMX is a proprietary implementation of the OpenSHMEM specification and is the premier SHMEM implementation on HPE EX systems. Both libraries leverage years of innovation to offer high performance and scalable communication capabilities. This paper offers an overview of HPE Cray MPI on HPE Cray EX supercomputers.**

I. Introduction

High density compute nodes that offer high performance CPUs and tightly coupled GPUs have become the "building blocks" of modern supercomputing systems. Modern high performance interconnects offer low latency, high bandwidth communication capabilities at extreme scale. In addition, some modern interconnects also offer hardware capabilities such as congestion management, quality-of-service, adaptive routing, etc. Combined together, modern compute and networking architectures are driving the evolution of high end supercomputing systems and these architectural trends have ushered in an era of Exascale computing capabilities. Such capabilities are critical to satisfy the increasing needs of current and emerging scientific applications that address challenging real world problems.

The HPE Cray EX supercomputer architecture offers a rich set of compute, accelerated compute, software, storage, and networking capabilities to achieve high performance at extreme scale. The HPE Cray EX architecture is designed to be highly flexible and offer a broad range of choices from a compute and system architecture perspectives. The HPE Cray EX architecture relies on the HPE Slingshot technology to offer high performance networking capabilities. The Frontier supercomputer at ORNL [1] is the world's first supercomputer to break the exascale barrier and is based on the HPE Cray EX architecture. The Frontier supercomputer was also the most energy efficient supercomputer on the June 2022 version of the Green500 list. In addition to Frontier, several systems in the Top500 list also rely on the HPE Cray EX supercomputer architecture. Some of the notable systems in this list are LUMI [2], Adastra [3], Perlmutter [4], Crossroads [5], Crusher [6], and Setonix [7]. Several upcoming systems are also being designed to leverage the HPE Cray EX supercomputer architecture. This list includes: El Capitan [8], Aurora [9], and the Alps [10] systems.

Emerging trends in hardware technologies introduce a level of complexity in the system architecture and this motivates the need for careful redesign of various system software stacks and applications. Some of the critical challenges that need to be addressed are portability, programming difficulty, communication/synchronization efficiency, and scaling. To be an effective HPC platform, modern high performance systems need a high-level software development environment with tightly coupled compilers, tools, and libraries that can interoperate and hide the complexity of the system and run efficiently on large scale supercomputers.

HPE Cray Programming Environment [11] is a key software component of the HPE Cray EX supercomputing system. The HPE Cray Programming Environment suite has a strong track record of offering a robust parallel programming environment on high end supercomputing systems. It offers a collection of compilers, tools, and libraries to facilitate the development of current and next generation scientific applications. It is tightly integrated with the broader HPE Cray EX hardware and software ecosystem to offer high performance and improved programmer productivity on modern HPE Cray EX systems.

A majority of pre-exascale and exascale codes running on HPE Cray EX supercomputers rely on the Message Passing Interface (MPI) model. In addition, the SHMEM programming model is also used in many scalable applications. It is absolutely necessary for HPC software stacks to offer high performant and scalable implementations of MPI and SHMEM models to allow scientific applications to achieve their desired scaling and performance goals. The HPE Cray Messaging Passing Toolkit (MPT) is one of the building blocks of the HPE Cray Programming Environment. The HPE Cray MPT stack is the primary software stack that implements the Message Passing Interface and SHMEM programming models on HPE Cray EX supercomputers. The MPT software stack is used heavily by domain scientists and application programmers on various HPE Cray EX supercomputers. This software stack was instrumental in surpassing the exascale performance barrier on the Frontier supercomputer. HPE Cray MPT has

also enabled various DOE, ECP, and CAAR applications to exceed their performance and scaling goals on Frontier and other exascale class HPE Cray EX supercomputers.

HPE Cray MPT is comprised of HPE Cray MPI and HPE Cray OpenSHMEMX software stacks. HPE Cray MPI is a proprietary implementation of the MPI specification. It is based on the open-source MPICH implementation [12] (3.4a2 release) from the Argonne National Laboratory and it implements the MPI-3.1 specification. HPE Cray OpenSHMEMX is a proprietary implementation of the OpenSHMEM specification and is the premier SHMEM implementation on HPE Cray EX systems. Both libraries leverage several years of innovation to offer high performance and scalable communication capabilities. These software stacks are designed to offer sustainable supercomputing capabilities by offering high performance capabilities across a wide array of system configurations and hardware architectures.

This paper offers an overview of HPE Cray MPI on HPE Cray EX supercomputers by delving into various design and implementation methodologies. This paper also includes a preview of some of the important HPE Cray MPI runtime parameters to debug, tune, and optimize real-world scientific applications.

II. Background

This section includes a high level description of HPE Cray EX network architectures and the HPE Cray Message Passing Toolkit software stacks.

A. HPE Cray EX Network Architecture

HPE Cray EX systems are designed to be highly flexible in terms of compute, networking, and storage architectures. HPE Cray EX system networks are typically available in two architecture flavors: HPE Slingshot-10 and HPE Slingshot-11. This section describes both network architectures.

1) HPE Slingshot Rosetta: The HPE Slingshot Rosetta [13], [14] switching infrastructure offers industry leading performance and scalability. Rosetta supports Ethernet standards and protocols, in addition to optimized HPC functionality. Rosetta is a high radix, 64 port switch that offers 12.8 Tb/s bandwidth while supporting both 100 Gbps and 200 Gbps network adapters. Rosetta switches can be used to design a system fabric that connects more than 250,000 compute nodes with a maximum of three hops. In addition, the Rosetta switch also offers key hardware support for congestion management, adaptive routing, and quality of service.

2) HPE Slingshot-10 network: HPE Slingshot-10 network architecture combines ConnectX-5 RoCE network adapters from NVIDIA [15] and HPE Slingshot Rosetta switches [14] from HPE. The ConnectX-5 RoCE network adapters from NVIDIA offer data rates of up to 100 Gbps along with a broad range of high performance data movement capabilities such as RDMA and OS bypass. HPE Slingshot-10 systems can be configured as heterogeneous systems that are comprised of CPUs and GPUs. In addition, these systems also offer the capability to use multiple NICs per node for higher aggregate network bandwidth.

3) HPE Slingshot-11 network: The HPE Slingshot-11 network architecture continues to use the HPE Slingshot Rosetta switching infrastructure. In addition, the Slingshot-11 architecture also leverages 200 Gbps HPE Slingshot Cassini network adapters. The HPE Slingshot-11 network adapters offer reliable, connectionless communication protocols. Communication software stacks can leverage this feature to support very large jobs that span thousands of compute nodes with very low memory footprint. The HPE Slingshot Cassini NIC offers key hardware offload capabilities for MPI message matching operations and MPI's Rendezvous protocol progression. This is an important capability that allows for strong message progression without requiring additional CPU cycles. The Cassini NIC also offers a suite of protocols for data movement operations that can be leveraged by software stacks to optimize communication performance for different payload sizes (Section III-E). HPE Slingshot Cassini NICs also support multiple HPC traffic classes to further isolate different traffic patterns. These important features allows applications that are typically vulnerable to irregular tail latency to experience repeatable performance on large systems. For one-sided data movement operations, Cassini also offers hardware support for atomics operations. In addition, Cassini offers hardware support for triggered operations and counting events. These features enable the development of new lightweight MPI communication capabilities, especially in the context of CPU/GPU heterogeneous systems (Section V-A). Finally, HPE Slingshot-11 also offers support for On-Demand Paging (ODP).

The Rosetta switch also offers hardware features to accelerate global reduction operations for small payload sizes. This capability can be leveraged to optimize the performance of various MPI collectives (MPI_Allreduce, MPI_Bcast, and MPI_ Barrier). This feature is expected to be available during the second half of 2023.

Certain processor and GPU architectures are only available in the HPE Slingshot-11 offering. For example, only HPE Slingshot-11 systems are available with the AMD MI250 GPU architecture [6], [1]. This paper emphasizes the behavior of the HPE Cray MPI stack on systems that rely on the HPE Slingshot-11 network architecture. However, specific salient points that involve the interaction between the HPE Cray MPI implementation and the HPE Slingshot-10 network are called out at appropriate places.

B. HPE Cray Message Passing Toolkit

This section provides an overview of the HPE Cray MPI software stack for HPE Cray EX systems based on the HPE Slingshot network architectures and lists some of the key dependencies.

HPE Cray MPI is designed to be highly flexible to address the broad range of HPE Cray EX system configurations. Figure 1 describes the breadth of coverage of the HPE Cray MPI stack across different hardware architectures and software environments. Supported hardware architectures include different CPU, GPU, and network architectures. Software environments span different compilation environments, operating systems, and tools. It is necessary to highlight that HPE Cray MPI supports both HPE Slingshot-10 and HPE Slingshot-11 network architectures.

FEATURE	COVERAGE
CPU architectures	Intel CPUs, AMD CPUs NVIDIA Grace CPUs (Under Development)
Network architectures	SS-10: OFI (verbs; rxm) provider, UCX driver support SS-11: OFI (CXI) provider InfiniBand clusters (HPE Apollo systems)
GPU architectures	AMD GPUs NVIDIA GPUs Intel GPUs (Under Development)
Operating Systems	COS, RHEL/CENTOS, SLES
PrgEnv and compilers	PrgEnv-cray, PrgEnv-gnu, PrgEnv-nvidia, PrgEnv-amd, and PrgEnv-intel
Launcher support	Slurm, PALS, and Flux

Fig. 1. HPE Cray MPI Coverage

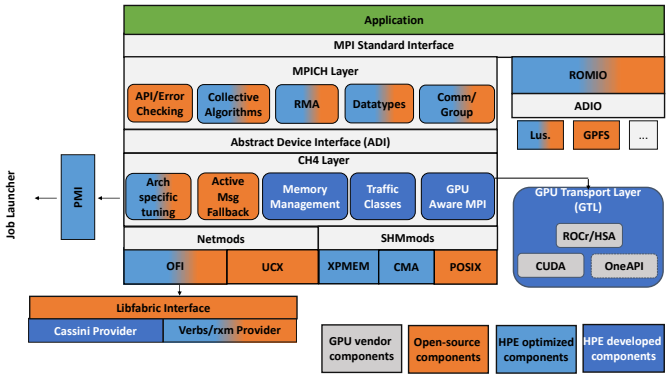

Fig. 2. HPE Cray MPI Software Architecture

Figure 2 offers a high level view of the HPE Cray MPI software stack. The current HPE Cray MPI implementation (version 8.1.26) for HPE Cray EX systems is based on the open-source ANL MPICH implementation (version 3.4a2) and uses the CH4 implementation. In addition to the set of software features ANL MPICH already offers in the CH4 implementation, HPE Cray MPI relies on the OFI Libfabric interface and the underlying network providers to leverage the rich set of capabilities offered by modern interconnects. On HPE Slingshot-10 systems, HPE Cray MPI relies on the ”verbs;rxm” network provider to interact with the Mellanox networking stack. Optionally, the UCX netmod is also available for users on HPE Slingshot-10 systems and HPE Cray MPI man pages document the necessary set of steps to toggle between the Libfabric and UCX netmods. On HPE Slingshot-11 systems, HPE Cray MPI relies on the ”Cassini (CXI)” network provider to leverage the broad set of hardware capabilities offered by the HPE Slingshot-11 network. The tight integration between the MPI library and the HPE Slingshot provider allows the HPE Cray Messaging Passing Toolkit suite to offer high performance communication at extreme scale.

On HPE Slingshot-10 systems, HPE Cray MPI offers low-latency, high bandwidth communication capabilities for both inter-node and intra-node MPI data movement operations. For inter-node MPI operations, HPE Cray MPI and the verbs;rxm Libfabric provider layers leverage RDMA capabilities offered by the Mellanox RoCE NICs. In addition, HPE Cray MPI supports systems that offer multiple Mellanox RoCE NICs per node. For intra-node MPI operations, HPE Cray MPI leverages a combination of POSIX shared memory, XPMEM, and Cross Memory Attach (CMA) implementations. XPMEM support is only available on systems that rely on the Cray Operating systems (COS).

On HPE Slingshot-11 systems, HPE Cray MPI is designed to take advantage of key hardware features offered by the Cassini NICs and the Rosetta switches. HPE Cray MPI has historically offered a broad set of highly optimized and tuned MPI collective operations. On the HPE Cray EX supercomputer architecture, HPE Cray MPI offers a set of collective operations that are specifically tuned for the HPE Slingshot-11 network. Similarly, HPE Cray MPI also leverages the HPE Slingshot-11 network to offer high performance MPI RMA communication operations. In addition to hardware offload capabilities, HPE Cray MPI is tightly integrated with the Cray Operating System (COS) and exposes ability to allocate memory regions that are backed by non-standard huge pages. HPE Cray MPI also relies on new optimizations in PMI and I/O layers and is being enhanced to support the MPI 4.0 specification. This paper includes additional areas of hardware/software co-design that allow HPE Cray MPI to expose extreme scale communication and synchronization capabilities to scientific applications.

On HPE Cray EX systems that are based on heterogeneous compute and accelerated compute architectures (such as GPUs), HPE Cray MPI offers a rich set of capabilities to offer “GPU aware” communication operations via the new GPU Transport Layer (GTL). GPU aware MPI capabilities are available on both HPE Slingshot-10 and HPE Slingshot-11 systems. GTL is an HPE proprietary layer and offers a broad range of GPU aware capabilities and optimizations in Cray MPI for HPE Cray EX systems. The GTL layer is designed in a modular manner and is easily extendable to support new GPU vendors and hardware architectures.

III. Design and Implementation Details

This section describes some of the key design and implementation aspects of the HPE Cray MPT software stack on HPE Cray EX systems based on HPE Slingshot-11 networks. The runtime variables listed in this section are a small subset of the entire set of variables supported by HPE Cray MPI.

This list can be accessed via the "man mpi" command on a HPE Cray EX system.

A. Multi-NIC Support

HPE Cray MPI supports multiple network adapters per compute node. The current implementation allows one MPI process to use a single NIC. Implementation issues and potential benefits related to the use of multiple NICs per process to implement striping techniques are under evaluation.

By default, HPE Cray MPI detects all available NICs per node and attempts to use them for the job. The specific process-to-NIC mapping is decided during MPI Init to make sure that each NIC is utilized in an efficient manner. HPE Cray MPI also offers a range of environment variables that allow users to customize process-to-NIC mapping to best fit the communication patterns and process-to-CPU placement techniques that are being used. Specifically, the MPICH_OFI_NIC_POLICY variable can be used to specify precise process-to-NIC mapping patterns for a given job. Accepted values for this variable are: "BLOCK", "NUMA", "ROUND-ROBIN", "GPU", and "USER". These convenient options allow the user to select the NIC that is closest in proximity to the CPU or GPU being used for each process. This is extremely important to obtain the best performance. Additionally, HPE Cray MPI also displays detailed information of process-to-NIC mapping if the MPICH_OFI_NIC_VERBOSE environment variable is set. The HPE Cray MPI man page documents this variable and its options in more detail.

B. Traffic Classification

HPE has implemented a set of "Best Practice" traffic classes to leverage hardware support for traffic classification for HPC applications on HPE Slingshot-11 systems. This list currently includes the following traffic classes:

1) "Low Latency": The Low Latency traffic class is best suited for applications that are vulnerable to the performance of small message collective operations. Such latency sensitive operations are given a higher priority in the network and this allows applications to benefit from lower latency and potentially lower jitter due to variability in network round trip times. However, this traffic class is also associated with a specific bandwidth cap.
2) "Dedicated Access": The Dedicated Access traffic class allows network packets issued by the communication library to benefit from a guaranteed bandwidth allocation. This traffic class is ideally used for highly specialized users and very high priority jobs that run on production systems.
3) "Bulk Data": The Bulk Data class allows the system fabric to isolate I/O traffic from every other type of traffic in the fabric.
4) "Best Effort": The Best Effort traffic class is the default shared traffic class and provides each application a "fair-share" of networking resources within the same class.
5) "Scavenger": Finally, the "Scavenger" class offers unreliable semantics and is intended only for low priority workflows. This traffic class will not be used for typical HPC workloads and HPE is exploring the possibility of using this class solely for background monitoring tools.

Users can request a specific traffic class via the MPICH_OFI_DEFAULT_TCLASS environment variable. This variable defaults to the "Best Effort" class. It is necessary to highlight that not every traffic class will be available on every HPE Slingshot-11 system and specific traffic classes also require prior Work Load Manager (WLM) authorization.

C. GPU Aware MPI Communication

On heterogeneous systems that offer both CPUs and GPUs, HPE Cray MPI offers "GPU Aware" MPI support for applications that perform MPI operations with communication buffers on GPU-attached memory regions. GPU aware MPI communication allows applications to pass CPU-attached or GPU-attached communication buffers to MPI data movement operations. A GPU aware MPI implementation handles both types of memory regions appropriately and also leverages hardware offload capabilities to implement data movement operations in a high performance manner. HPE Cray MPI is tightly integrated with the rest of the Cray PE stack to offer GPU aware capabilities for NVIDIA and AMD GPU devices. Support for Intel GPUs is under development.

HPE Cray MPI supports the following technologies for MPI operations involving GPU-attached memory regions: GPU-NIC RDMA (for inter-node MPI transfers), GPU Peer2Peer IPC (for intra-node MPI transfers), and preliminary support for GPU NIC Async (Section V-A). GPU-NIC RDMA relies on RDMA capabilities offered by modern high performance NICs to efficiently move data between GPU-attached memory regions across different nodes. GPU IPC leverages DMA engines on GPU devices and high bandwidth communication paths that exist between GPU devices within the same compute node.

D. Slingshot-11 Hardware Congestion Management

HPE Slingshot-11 offers hardware support for congestion management. Studies [13], [16] have shown that applications running on HPE Slingshot-11 systems are much less affected by network congestion when compared to other network technologies. These studies were performed in a systematic manner across different microbenchmarks and HPC applications with a broad range of communication patterns. Hardware congestion control is enabled by default on HPE Slingshot-11 systems.

E. Point-to-Point Communication Protocols

This subsection describes some of the key hardware features used to implement inter-node point-to-point operations.

1) Hardware Tag Matching: Cassini NIC offers hardware support for message matching. This feature is enabled by default in the CXI provider within the OFI framework. HPE notes that the hardware support for tag matching is the preferred execution mode and this capability will be disabled only under severe resource exhaustion scenarios. In such rare use cases, the CXI provider will fall back to using a software implementation of message matching. Since HPE Cray MPI relies on the OFI layer and the underlying CXI provider, HPE Cray MPI leverages this capability by default on Slingshot-11 systems. Users may toggle this feature by using the following environment variable: "FI_CXI_RX_MATCH_MODE". This variable accepts the following values: "hardware", "software", and "hybrid" and "hardware" is the default match mode on HPE Slingshot-11 systems. In the "hybrid" mode, for a given job, the provider first offers the "hardware" match mode but later transitions seamlessly within the same job to the "software" match mode for specific MPI ranks that determine resource exhaustion is imminent.

2) Small message communication protocols: HPE Slingshot Cassini NIC offers two mechanisms to implement inter-node small message transfers. The first mechanism is referred to as the "Immediate Data Command (IDC)". In this mechanism, the CXI provider implementation directly transfers the user payload into a NIC command descriptor and directly enqueues the descriptor to the NIC Command Queues. The current HPE Cray MPI stack and the Cassini provider implementation rely on the IDC mechanism by default for payloads of size less than 192 Bytes. In the second mechanism, the CXI provider enqueues a DMA command descriptor in the NIC Command Queue. The DMA descriptor includes the source address, length, and the match bits that correspond to the MPI_Send operation. When this descriptor is executed, the Cassini NIC performs a DMA operation from user's buffer to perform the operation. HPE Cray MPI can utilize either protocol on Slingshot-11 systems.

3) Hardware Offload for MPI's Rendezvous protocol for Large Payloads: HPE's Cassini architecture offers the ability to offload the entire rendezvous handshake protocol and the data transfer operations to the NIC. When FI_CXI_RX_MATCH_MODE is set to "hardware", this capability is enabled in the CXI provider. In this implementation, the hardware offloaded version of the rendezvous protocol relies on the target NIC to perform the match operations and subsequently perform an RDMA Get operation to fetch the payload from the source node. The hardware implementation also handles "Ready To Send" (RTS) packets arriving unexpectedly at the target process. Unexpected RTS packets are processed in a deferred manner by the NIC. For each new MPI_Recv operation performed by the application, the hardware attempts to match against the deferred RTS packets until a match occurs. Once a match has been detected, the hardware Cassini NIC performs an "RDMA Get" operation and the corresponding "Rendezvous Done" packet to inform the source NIC about the completion of the entire data transfer operation. Since the HPE Slingshot-11 networking hardware handles both expected and unexpected messages, CPU cycles are required only to prepare and initiate the transactions. From a CPU utilization perspective, the source process consumes CPU cycles to prepare and issue the first RTS packet that contains the match bits for the message. The target process consumes CPU cycles to prepare MPI-level and OFI-level objects that correspond to the MPI_Recv operation. The rest of the rendezvous protocol is entirely offloaded to hardware and does not require CPU cycles to progress the tag-matching or the data movement operations. Thus, HPE Slingshot-11 systems can offer improved communication/computation overlap for applications that rely on asynchronous point-to-point operations (Section IV-A4).

F. One-sided Communication Protocols

HPE Slingshot-11 offers highly efficient data movement and synchronization capabilities for implementing one-sided communication operations. MPI_Put and MPI_Get operations can be directly mapped RDMA Put and RDMA Get operations. These operations are fully offloaded to the Cassini NIC and do not require additional CPU cycles. HPE Slingshot-11 also offers hardware support for one-sided atomics. Intra one-sided operations leverage Posix shared memory and XPMEM for CPU-attached communication buffers. For GPU-attached buffers, HPE Cray MPI leverages GPU IPC to optimize intra-node phases of one-sided operations.

G. Collective Communication and HPE Optimized algorithms

HPE Cray MPI has historically implemented a broad range of optimizations for collective operations. These optimizations are geared towards the underlying compute and network architectures. For HPE EX systems, HPE Cray MPI implements a suite of highly tuned collective algorithms that are customized for the HPE Slingshot-10 and HPE Slingshot-11 network architectures. Software optimizations in Cray MPI consider the CPU processor architecture, GPU architecture, and node-level topology. Intra-node phases of collectives operations leverage XPMEM, CMA, or POSIX shared memory for CPU-attached communication buffers. For GPU-attached buffers, HPE Cray MPI leverages GPU IPC to optimize intra-node phases of collective operations. Section IV-B includes a brief discussion of some of the optimized collectives in Cray MPI on HPE Cray EX systems.

IV. Performance Evaluation

This section includes a select set of performance studies with HPE Cray MPI on HPE Cray EX systems. For brevity, this section includes an analysis of performance trends observed on the Bard Peak node architecture [17]. Figure 3 shows a very high level view of the architecture of a compute node in the ORNL Frontier system. Each node offers a single AMD Trento CPU socket, four MI250 GPU modules, and four

HPE SlingShot-11 Cassini NICs devices. Each GPU module consists of two GPU devices (GCDs). Each Cassini NIC is attached to a single MI200 module via PCIe Gen4 with the Extended Speed Mode (ESM) capability. The MI200 modules are interconnected with 1x or 2x high speed xGMI3 links. The Trento CPU can access each MI200 module via high speed 2x xGMI2 links. The PCIe Gen4 ESM links offer a peak throughput of 25 GB/s/dir. Each xGMI3 link offers a peak throughput of 50 GB/s/dir and each xGMI2 link offers a peak throughput of 36 GB/s/dir. A single Trento CPU socket offers 64 cores, with two hardware threads per core. Most applications will typically map one or two application processes per GCD. Each GCD device may be used by up to 8 application processes. Thus, it is important to understand the communication costs experienced by a given process for a particular peer process – whether that peer process is collocated within the same node, or if it is affined to a different node in the system. Recipes needed to ensure the right affinity settings are highly specific to the system configuration. [18] includes some of these recipes needed for the Frontier and Crusher systems at ORNL.

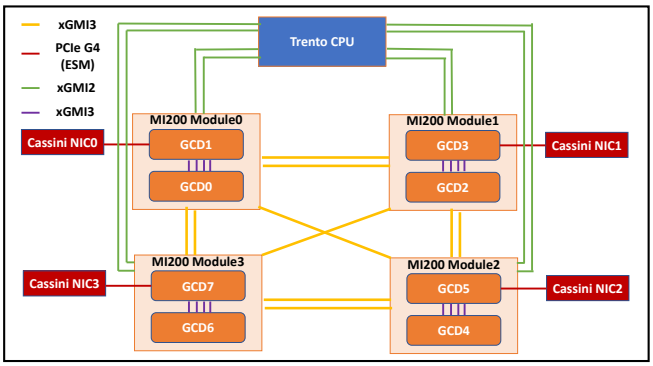

Fig. 3. ORNL Frontier Bard Peak Node Architecture

A. MPI Point-to-point operations

This section includes a summary of MPI point-to-point performance studies with HPE Cray MPI on HPE Cray EX systems based on the Bard Peak node architecture.

1) Inter-node MPI latency with GPU-attached buffers: Figure 4 compares the inter-node latency of small message MPI point-to-point operations when two processes use Cassini NIC0 on both nodes with different process-to-GCD mappings. This experiment involves running the "osu_latency" ping-pong latency benchmark with two Bard Peak nodes attached to the same switch. The benchmark is configured to use GPU-attached communication buffers on both nodes. For the "GCD0" case, both processes use GCD0 and Cassini NIC0 on their respective nodes. For the "GCD1" case, both processes use GCD1 and Cassini NIC0 on their respective nodes. This study demonstrates that the impact of Cassini NIC0 being physically closer to GCD1 has a very small impact (less than 2%) on inter-node small message latency.

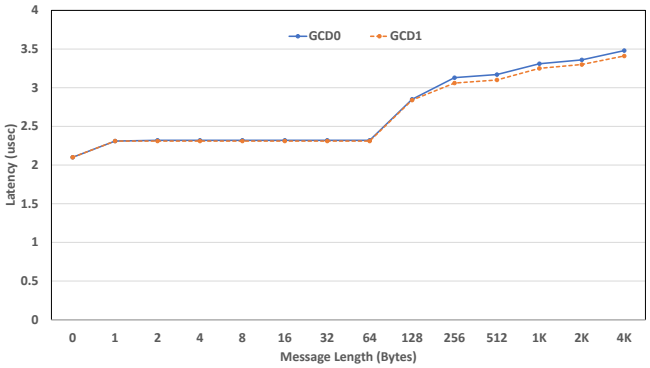

Fig. 4. MPI Point-to-Point Inter-node Latency

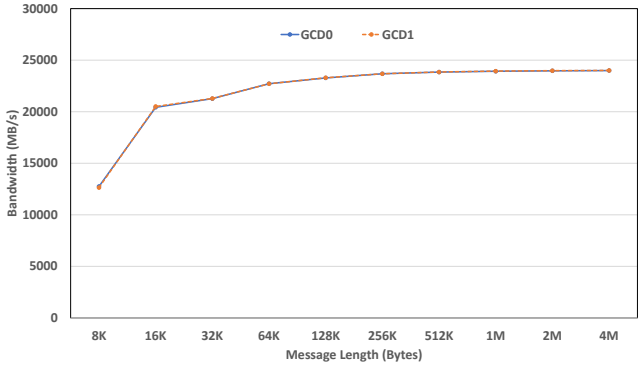

Fig. 5. MPI Point-to-Point Inter-node Bandwidth

2) Inter-node MPI bandwidth with GPU-attached buffers: Figure 5 compares the inter-node uni-directional bandwidth for large message MPI point-to-point operations when two processes use Cassini NIC0 on both nodes with different process-to-GCD mappings. This experiment involves running the "osu_bw" uni-directional benchmark with two Bard Peak nodes attached to the same switch. The benchmark is configured to use GPU-attached communication buffers on both nodes. For the "GCD0" case, both processes use GCD0 and Cassini NIC0 on their respective nodes. For the "GCD1" case, both processes use GCD1 and Cassini NIC0 on their respective nodes. This study demonstrates that the fact that Cassini NIC0 is physically closer to GCD1 has no impact on inter-node large message bandwidth.

3) Intra-node MPI bandwidth with GPU-attached buffers: Figure 6 demonstrates the impact of process-to-GPU mapping on intra-node MPI uni-directional bandwidth for large payloads. This experiment also involves the use of two processes using the same compute node, both processes use GPU-attached communication buffers, and each process uses a different CPU hardware thread. The "GCD0-GCD0" case demonstrates a significantly higher communication throughput

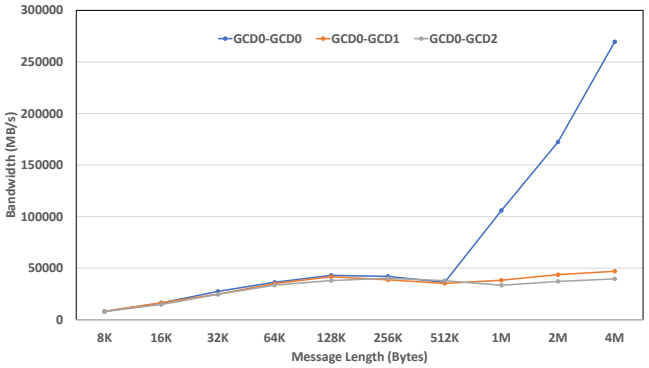

Fig. 6. MPI Point-to-Point Intra-node Bandwidth

for large payloads when compared to "GCD0-GCD1" and "GCD0-GCD2". This is because the MPI operations involve transferring data between processes that are using the same HBM device on GCD0. The AMD ROCm layer optimizes this case by leveraging kernels on the GPU to perform data movement operations instead of offloading the transfers to the DMA "Copy" engines. For "GCD0-GCD1" and "GCD0-GCD2", the AMD ROCm layer relies on the DMA "Copy" engines to transfer data between GCDs. In both cases, the data transfer operations involve a set of xGMI links that exist between GCD devices and the same ROCr IPC software mechanisms implemented within HPE Cray MPI. For the "GCD0-GCD1" case, the hardware offers four xGMI links between GCDs that are on the same MI200 module. However, the AMD ROCm implementation uses a single DMA engine to move data between these GCDs. On the MI200 hardware, HPE and AMD have observed that a single DMA engine can utilize multiple xGMI links between GCDs on the same MI200 module. However, the copy engine cannot effectively utilize all of the throughput offered by the set of four xGMI links. In the case of "GCD0-GCD2", only one xGMI link can be used to implement the data transfer operations. Thus, "GCD0-GCD1" can achieve slightly higher peak communication throughput when compared to "GCD0-GCD2". But "GCD0-GCD1" cannot achieve the peak combined throughput of four xGMI links on MI200. HPE and AMD are continuing to discuss possible software strategies to improve the performance of data transfer operations between GCDs that are collocated on the same MI200 module.

4) Computation/Communication Overlap for Inter-node MPI point-to-point operations: Figure 7 demonstrates CPU % as measured with the Sandia Overlap Benchmark [19]. This benchmark involves two MPI processes on two different compute nodes. Both processes participate in asynchronous MPI point-to-point operations with varying payload sizes and the benchmark reports the percentage of CPU cycles available to perform application-level compute tasks. For a given payload size, higher percentage values indicate that a significant fraction of the communication protocol has been offloaded to the network hardware and CPU plays a very small role in progressing the MPI Rendezvous protocol. On HPE Slingshot-11 systems, the Sandia Overlap Benchmark reports very high CPU availability for inter-node MPI point-to-point operations with large payloads since the network hardware offers strong message progression semantics.

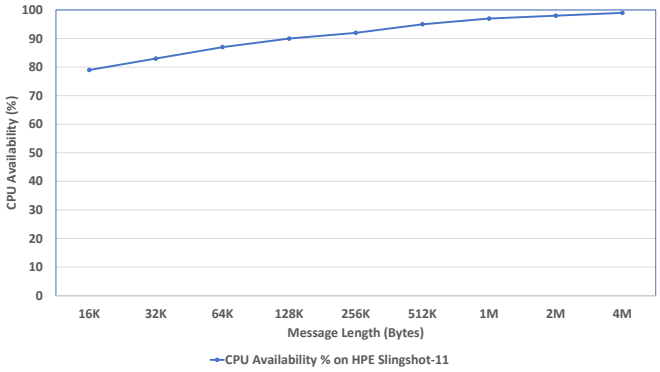

Fig. 7. Computation/Communication Overlap for large message MPI point-to-point operations

B. Collective Operations

For the sake of brevity, this section demonstrates a small subset of MPI collective optimizations implemented in HPE Cray MPI.

Figure 8 describes optimizations in the latest HPE Cray MPI for MPI_Igatherv. This implementation relies on key optimizations in HPE Cray MPI software for systems based on the HPE Slingshot-11 and HPE Slingshot-10 networks. These optimizations improve MPI_Igatherv communication latency for a broad range of payload sizes and communicator sizes. Figure 8 compares the performance of the Baseline and Optimized implementations of MPI_Igatherv in HPE Cray MPI with a communicator size of 65,536, with 512 compute nodes and 128 MPI processes per node. At this scale, the Optimized implementation outperforms the Baseline implementation by about 130X!

Figure 9 demonstrates optimizations for MPI_Reduce_scatter_block on compute nodes based on the Bard Peak node architecture with GPU-attached communication buffers. This experiment was performed on 8 Bard Peak compute nodes with 8 MPI processes per node. Each process uses a single GCD device on a node. The Baseline implementation of MPI_Reduce_scatter_block involves performing the compute phases of the collective on the CPU. The Optimized version of the collective leverages the ability to offload the compute intensive phases of large payload MPI_Reduce_scatter_block operations to the GPU. This optimization improves the performance of large payload MPI_Reduce_scatter_block by about 7X.

A similar optimization is also available on systems with NVIDIA A100 GPUs and this leads to an improvement of about 15X for large payload Allreduce operations on a single node with MPI processes. This optimization is

enabled by default if MPICH_GPU_SUPPORT_ENABLED is set to 1 and can be disabled by setting "MPICH_GPU_ALLREDUCE_US_KERNEL=0"

Both optimizations discussed in this section are enabled by default in the latest HPE Cray MPI implementation.

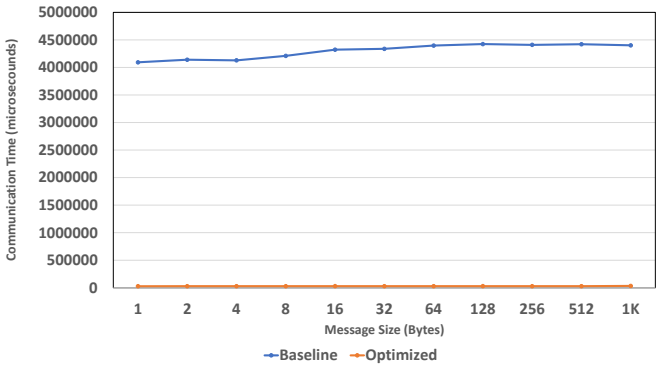

Fig. 8. MPI_Igatherv Optimizations in HPE Cray MPI

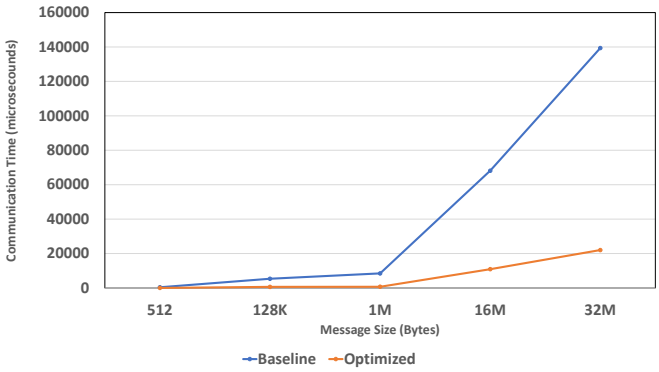

Fig. 9. MPI_Reduce_scatter_block Optimizations in HPE Cray MPI

V. New and Upcoming Features in Cray MPT

This section profiles two new features that are available in Cray MPT.

A. GPU-NIC Async

On modern heterogeneous supercomputing systems that are comprised of compute blades that offer CPUs and GPUs, current generation scientific applications and systems software stacks are "GPU-aware" and data movement operations are offloaded to the NIC (GPU-NIC RDMA), or DMA engines on GPU devices (GPU Peer2Peer IPC). However, CPU threads are still required to orchestrate data moving communication operations and inter-process synchronization operations. Naturally, this requirement results in all communication and synchronization operations occurring at GPU kernel boundaries. An application process running on the CPU first synchronizes with the local GPU device to ensure that the compute kernel has completed. Next, an application process initiates, progresses, and completes inter-process communication/synchronization operations. Typically, subsequent compute kernels can be launched only after the inter-process communication operations have completed. Owing to this behavior, current generation GPU-aware parallel applications are affected by potentially expensive synchronization points that require the CPU to synchronize with the GPU and NIC devices. In addition, the overhead of launching compute kernels is in the critical path and an iterative-parallel application experiences this overhead each time a new kernel is offloaded to the GPU. HPE has been working closely with GPU vendors to explore, prototype, and implement "GPU-NIC Async" strategies to address these problems on emerging supercomputing systems. These strategies also leverage some of the key technologies offered by the HPE Slingshot-11 network [20].

B. MPIXlate

Several shared library implementations of the Message Passing Interface (MPI) specification are available for building MPI applications. Even when they implement the same MPI specification, some MPI shared libraries are not binary compatible with others due to differences in implementation choices. As a consequence, applications built with a specific MPI shared library can be run only on systems which have an MPI shared library that is binary compatible with the one used to build the application. On systems where such a binary compatible MPI library is not available, the application must be recompiled using the native MPI shared library. MPIxlate, a new feature now available in HPE Cray MPI, enables applications compiled using an MPI library that is not binary compatible with HPE Cray MPI, to be run without recompilation on supported HPE Cray systems. MPIxlate does transparent runtime translation of the Application Binary Interface (ABI) between supported combinations of source and target MPI shared library implementations. The MPI shared library used to compile the application determines the source ABI and the HPE Cray MPI shared library using which it will be run determines the target ABI. As HPE builds out the HPC Cloud portfolio, MPIxlate provides a means by which applications built with non-MPICH compatible MPI libraries can take advantage of the optimized Shasta networking infrastructure.

VI. Important Runtime Variables to Debug and Diagnose Issues

This section includes some of the useful HPE Cray MPT environment variables that users can use to debug performance and functionality issues.

1) *Network Timeouts:* On a Slingshot system, certain applications may experience instances of network timeouts. These timeouts are often attributed to flapping links. The HPE Slingshot-11 network handles such events by automatically re-issuing network packets that were affected by timeout events. Depending on the

communication patterns used by specific applications, these events may result in lower than expected MPI communication performance. To help debug such performance issues, HPE Cray MPI tracks network timeout events and summarizes various Cassini counters for each job. If an application experienced network timeouts, HPE Cray MPI displays the following line during MPI_Finalize: "[MPICH Slingshot Network Summary: *N* network timeouts]". Support for collecting a set of user-specified Cassini hardware counters is also available. The "MPICH_OFI_CXI_COUNTER_REPORT" variable is documented in the the HPE Cray MPI man pages.

2) *Running out of Cassini Hardware Resources:* Certain communication patterns can cause applications to experience the following error messages: "PtlTE NN LE resources not recovered during flow control. FI_CXI_RX_MATCH_MODE=[hybrid OR software] is required.". These are fatal errors and are due to select MPI processes running out of hardware resources in the Cassini NIC. Such applications may benefit from using the "hybrid" match mode feature implemented in the Libfabric Cassini provider. Users can enable this feature by setting the "FI_CXI_RX_MATCH_MODE" runtime variable to "hybrid" or "software". (Section III-E1 described different message matching modes on HPE Slingshot-11 systems.)
3) *Issues with "fork()":* On HPE Slingshot-11 systems, applications that rely on "fork()" can experience issues if the child process tries to access memory regions owned by the parent process after a "fork()" is performed. In such cases, users are advised to set the following runtime variables: "export CXI_FORK_SAFE=1", "export CXI_FORK_SAFE_HP=0", and "export FI_CXI_DISABLE_CQ_HUGETLB=1".
   With SLES15 SP4 and newer Linux kernels, some of the "fork()" issues have been addressed in the Linux kernel via changes to the Copy-on-write semantics. Some applications may benefit from these changes directly on newer Linux kernels and this removes the need for users to set the CXI_FORK_SAFE related runtime variables for applications that rely on "fork()".
4) *Corner cases with GPU-NIC RDMA:* On HPE Slingshot-11 systems, GPU enabled applications may experience hangs and the following error messages are seen in the "dmesg" logs: *cxi_core:cass_vma_write_flag:22 VMA does not have write permissions*. This error message is often a result of the following user errors:

   a) HPE Cray MPI's GPU aware logic is not enabled because the followinng runtime variable was not specified in the job submission script: "MPICH_GPU_SUPPORT_ENABLED=1"
   b) Application uses GPU Managed memory regions and HPE Cray MPI's Managed memory support is disabled. HPE Cray MPI currently supports Managed Memory regions by default.
   c) The application executable was not linked correctly against GPU runtime library. On systems with NVIDIA GPUs, this scenario is likely to occur if the "module load cudatoolkit" command was excluded in the environment or in the job submission script.

   Users are encouraged to review HPE Cray MPI's man pages for runtime variables related to GPU support. These man pages also include suggested recipes for building and linking user applications.

VII. Summary and Conclusion

HPE Cray EX systems are powering some of the fastest supercomputing systems in the world. In addition to designing systems with key novel technologies, HPC system software stacks need to be designed carefully with specific performance and scaling goals. HPE Cray Programming Environment is a critical piece of the software stack on HPE Cray EX system and it enables the development and optimization of current and next generation scientific applications. This paper provided a detailed description of the HPE Cray MPT software stack and its importance on HPE Cray EX systems. This paper described some of the key hardware/software interactions and key co-design areas to optimize the performance of scientific applications. Finally, this paper also documented some of the important runtime variables that can be used to debug, diagnose, and tune the performance of scientific applications on HPE Cray EX systems.

References

[1] Oak Ridge National Lab, "Frontier," https://www.olcf.ornl.gov/frontier/.
[2] Lumi Consortium, "LUMI," https://lumi-supercomputer.eu/.
[3] Genci, "ADASTRA," https://www.genci.fr/en/node/1149.
[4] National Energy Research Scientific Computing (NERSC), "Perlmutter," https://www.nersc.gov/news-publications/nersc-news/science-news/2021/berkeley-lab-targets-exascale-with-perlmutter-and-nesap/.
[5] Los Alamos National Lab, "Crossroads," https://www.lanl.gov/projects/crossroads/.
[6] Oak Ridge National Lab (ORNL), "Crusher," https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html.
[7] Pawsey Supercomputing Centre, "Senotix," https://pawsey.org.au/systems/setonix/.
[8] Lawrence Livermore National Lab (LLNL), "El Capitan," https://www.llnl.gov/news/llnl-hpe-partner-amd-el-capitan-projected-worlds-fastest-supercomputer.
[9] Argonne Leadership Computing Facility, "Aurora," https://www.alcf.anl.gov/aurora.
[10] Swiss National Supercomputing Center (CSCS), "ALPS," https://www.cscs.ch/computers/alps/.
[11] HPE, "HPE Cray Programming Environment," https://www.hpe.com/psnow/doc/a50002303enw.
[12] Argonne National Laboratory, "MPICH: A High-Performance, Portable Implementation of MPI," https://www.anl.gov/mcs/mpich-a-highperformance-portable-implementation-of-mpi.
[13] D. De Sensi, S. Di Girolamo, K. H. McMahon, D. Roweth and T. Hoefler, "An In-Depth Analysis of the Slingshot Interconnect," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '20, New York, NY, USA, 2020.

[14] HPE, "HPE Slingshot Rosetta Interconnect," https://www.hpe.com/us/en/compute/hpc/slingshot-interconnect.htmlresources.
[15] NVIDIA, "NVIDIA ConnectX-5 Ethernet adapters," https://www.nvidia.com/en-us/networking/ethernet/connectx-5/.
[16] S. Chunduri, T. Groves, P. Mendygral, B. Austin, J. Balma, K. Kandalla, K. Kumaran, G. Lockwood, S. Parker, S. Warren, N. Wichmann, and N. Wright, "GPCNeT: Designing a Benchmark Suite for Inducing and Measuring Contention in HPC Networks," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356215
[17] Scott Atchley, "Preparing For Frontier Training Series," https://olcf.ornl.gov/wp-content/uploads/Frontiers-Architecture-Frontier-Training-Series-final.pdf.
[18] Oak Ridge National Lab (ORNL), "Frontier User Guide," https://docs.olcf.ornl.gov/systems/frontier_user_guide.html.
[19] Sandia National Laboratory, "Sandia micro-benchmarks (SMB)," https://github.com/openucx/openhpca.
[20] N. Namashivayam, K. Kandalla, T. White, N. Radcliffe, L. Kaplan, M. Pagel, "Exploring GPU Stream-Aware Message Passing using Triggered Operations," 2022.