HPE Slingshot Launched into Network Space

Duncan Roweth
CTO Office, HPC/AI BU
HPE, Inc
Bristol, UK
duncan.roweth@hpe.com

Greg Faanes
Advanced Technology, HPC/AI BU
HPE, Inc
Chippewa Falls, , WI, USA
gregory.faanes@hpe.com

Marten Terpstra
Product Management, HPC/AI BU
HPE, Inc
Nashua, NH, USA
marten.terpstra@hpe.com

Jesse Treger
Product Management, HPC/AI BU
HPE, Inc
Portland, OR, USA
jesse.treger@hpe.com

***Abstract*— HPE Slingshot networks are constructed from two components, a PCIe Gen4 NIC "Cassini" and a 64-port switch "Rosetta". Their links use standard Ethernet physical interfaces operating at 200 Gbps designed to construct either dragonfly or fat-tree networks. Rosetta switches operate HPE Slingshot specific adaptive routing and congestion management protocols on the fabric links that connect them together. Their edge ports, including those that connect to Cassini, support both optimized HPC and standard Ethernet protocols. The HPE Cray EX supercomputer system uses them in a dragonfly network as this provides cost effective global bandwidth at scale. Clusters of HPE Apollo servers (HPE Inc, 2021) can use either dragonfly or fat-tree1. HPE Slingshot networks are designed to support 64 to 250,000 or more endpoints. The largest system under construction has approximately 85,000 endpoints. Support for systems of this scale has a significant bearing on the design of the Rosetta and Cassini devices. This paper presents the key features and some early results of performance of Rosetta and Cassini devices. This paper includes information from early papers on the Rosetta ASIC, as well as section of not-yet published papers on the Cassini NIC.**

Keywords—Cassini, Rosetta, Dragonfly, Fabric, HPC, MPI

I. Introduction

HPE Slingshot is a modern, highly scalable, connectionless high-performance RDMA interconnect for high-performance computing (HPC) and artificial intelligence (AI) clusters that delivers industry leading performance, bandwidth, and low latency for HPC, AI/ML, and data analytics applications. HPE Slingshot brings together the best of HPC optimized fabrics - as pioneered by Cray over eight generations of supercomputing interconnect silicon – with the ubiquity of standard Ethernet. HPE Slingshot features fine-grain adaptive routing, advanced congestion control, and sophisticated quality-of-service capabilities (QoS). By leveraging and building on standard Ethernet technology and software, HPE Slingshot delivers cost-effective connectivity and interoperability with third-party NICs and switches, and a broad eco-system of software. Support for both high-performance IP traffic and remote memory operations broadens the range of applications that perform with high performance beyond traditional MPI-based modeling and simulation codes. HPE Slingshot delivers a converged network infrastructure with great performance on both traditional modeling and simulation codes alongside native sockets-based applications and direct Ethernet connectivity to storage without requiring gateway nodes.

HPE Slingshot delivers high bandwidth and low latency – both average and tail latency - consistently and reliably in real-world operating conditions, especially under load with diverse and contending traffic patterns. HPE Slingshot moves beyond just raw performance capability with a system-level approach to ensure actual performance is realized and delivered consistently and reliably. This is because it couples leading edge enhanced Ethernet technology and bandwidth (200 Gbps links using PAM-4 signalling) coupled with advanced features including breakthrough fine grain adaptive routing, congestion management, and QoS features to deliver consistent higher effective application performance at scale.

High raw performance interconnect technology does not guarantee high performance in practice. This is because modern HPC systems are complex and simultaneously host a range of workloads that present an inconsistent set of communications patterns that challenge traditional brute bandwidth HPC interconnects because they cannot deal well with the unavoidable congestion conflicts. Congestion in turn causes the effective latency of operations to spike, causing application performance to suffer and vary from run-to-run.

HPE Slingshot consists of switches built with the HPE-developed Rosetta ASIC, and NICs built with the HPE-developed Cassini ASIC. Rosetta is a 64-port, 200 Gbps/direction/port HPC Ethernet switch chip that provides state-of-the-art small packet routing, an advanced congestion management system, and QoS capabilities. The Cassini NIC chip provides hardware offload of MPI matching and progression, together with one-sided operations and collectives. The host interface for Cassini is an enhanced version of PCIe Gen4.

The HPE Slingshot 64 port high switch radix enables the deployment using a dragonfly topology of low-diameter networks even at very large scale. With HPE Slingshot, a system can scale up to 250,000 physical endpoints with only a three-hop diameter. Reducing the network diameter reduces latency, cost and the power consumed. Low diameter also enhances adaptive routing algorithms that improve application performance and reliability. HPE Slingshot's high peak performance and low network diameter delivers high-performance, cost-effective networks with very low latency.

II. Rosetta Switch Architecture

Rosetta is a 64-port Ethernet switch, implemented as a large monolithic (685 mm^2) ASIC fabricated in the TSMC 16 nm FinFet process and housed in a 62.5 mm package. All main switching logic utilizes an 850 MHz clock, which results in a

typical power dissipation of 160 watts and a maximum of around 300 watts.

A. Ethernet Functionality

All Rosetta ports support IEEE 802.3 Ethernet standard signaling. Each port has 4 lanes operating at 50 Gbps using pulse amplitude modulated (PAM-4) signaling or 25 Gbps non-return to zero (NRZ) signaling to provide 200 Gbps Ethernet (IEEE 802.3bs), 100 Gbps or 50 Gbps (IEEE 802.3cd), or 100 Gbps (IEEE 802.3-2015). While providing an optimized, low latency, high throughput HPC infrastructure, Slingshot is fully Ethernet standards compliant and interoperable, operating as a converged HPC / Ethernet network.

B. Enhanced Link Functionality

The Slingshot architecture provides optimizations within the network stack designed to improve network efficiency. These enhancements, driven by HPC and data analytics workloads, bring performance of the data link layer protocol on par with, or above, the capabilities of proprietary high-speed networks. These improvements benefit both HPC and Ethernet applications since they enable greater throughput, permit a higher transaction rate, and improve reliability. Enhancements include reduced inter-packet gap, optimized packet headers, credit-based flow control, link-level retry, and degraded link operation (i.e. continued operation with two or three of the four lanes disabled). Enhanced functionality is negotiated between pairs of devices using Link Layer Discovery Protocol (LLDP).

C. Network Protocols

HPE Slingshot's data link layer (layer 2) and network layer (layer 3), compliant with the Open System Interconnection (OSI) model, can switch packets based on an L2 endpoint address within a subnet, and can route packets based on an L3 destination between subnets (L3 routing is currently not yet enabled in software). Like many standards-based networks, IPv4 and IPv6 routing is implemented as part of the control plane. This network management overlay loads the appropriate forwarding and routing tables, affecting the way packets traverse the high-speed network data plane. Both in-band and out-of-band network management paths allow standardized routing mechanisms and protocols to communicate with the Rosetta devices. Additionally, network function virtualization (NFV) enables offload of many network services and their supported protocols, providing support for resilient, scalable services, sized to match network requirements and load.

D. Physical Structure

Rosetta uses a tiled physical structure, using two primary logic blocks: *tile functions* (TF) and *peripheral functions* (PF). There are 32 TF blocks, each performing the routing, packet forwarding, and congestion management functions for two physical ports. The 32 TF blocks are arranged in a 4-row by 8-column matrix. Interconnect in the row and column dimensions creates the all-to-all connections within the matrix. The tiled design allows for the crossbars to be distributed across tiles, avoiding global arbitration. Distributed buffering provides a 4X internal speedup of data movement.

There are 32 PF blocks, each also performing its functions for two physical ports. PF functions include Ethernet protocol functionality along with the physical coding sublayer (PCS) and media access control (MAC) layer for each channel. These 32 blocks are arranged on the periphery of the die along with the eight SerDes required to support the two physical ports of each PF block. Each PF is connected to its corresponding TF through global channel routing, which completes the top-level interconnect scheme. In addition to the PF and TF blocks, there is a maintenance block (MB) that provides the configuration and control interface into Rosetta for system management.

E. Congestion Management

Rosetta's adaptive routing carefully changes a packet's path to avoid mid-fabric congestion, which increases network utilization and reduces latency, but adaptive routing cannot address endpoint congestion. In this section, we describe a novel congestion control mechanism implemented in Slingshot that reduces the impact of endpoint congestion.

Under normal circumstances, the flow mechanism in Rosetta rarely needs to stop packets, because the buffering requirements for each flow are carefully controlled. The total mid-fabric buffer requirements are typically less than the capacity of the mid-fabric input buffers. This ensures that flows that are not experiencing any congestion have enough buffer space to operate at full bandwidth.

Congestion often forms around an egress edge port. This can occur for several reasons: an application or service has directed many streams of packets (flows) to a single egress port, a destination node is unable to process new data quickly enough and starts to assert pause, or a destination edge port is operating at a lower bandwidth than other edge ports.

Congestion can cause network buffers to fill, requiring traffic trying to pass through the buffers to be stopped to prevent overflow and packet loss. Such back-pressure can cascade and spread out upstream, a phenomenon known as tree saturation. Traffic not even passing through the congested bottleneck can be blocked if it shares virtual channels with congested traffic on any of the upstream links. To avoid this coupling, effective congestion management must quickly detect and reduce congestion, preventing buffers from filling and blocking unrelated traffic.

The Datacenter Quantized Congestion Notification (DCQCN) protocol [1] is an example of a mechanism to detect and control congestion within a network by observing buffer occupancy and the rate at which a buffer is filling and then sending acknowledgements to reaction points in the network to throttle the traffic that is contributing to the congestion. DCQCN suffers from several problems, which include unfairness in which flows are throttled, and a large number of control parameters that make tuning difficult. It is possible to optimize the configuration for one pattern of traffic and to severely impact other patterns of traffic. For these reasons, DCQCN for a high-speed network struggles to balance high network utilization and stability while responding to congestion. On the other hand, congestion schemes like Explicit Congestion Notification (ECN) [2], have slow convergence in large networks due to the long feedback of ECN markers in the presence of congestion. Moreover, these schemes continue to permit inter-job

interference while they attempt to adjust the transmission rates of congesting flows.

Rosetta congestion management is designed to prevent congestion from growing within the network fabric, and to prevent congestion from interfering with unrelated traffic. Egress switches measure the degree of endpoint congestion and return this indication in *congestion acks* for flows directed to congested egress ports. In upstream switches, the bandwidth of each flow is reduced to match the available egress bandwidth. Critically, only the flows targeting the congested egress port are back-pressured, and this flow control is pushed back all the way to the flows' ingress ports. Other traffic can progress unimpeded, even if sharing virtual channels with congesting flows on upstream links.

The mechanism strongly controls admission for congesting flows. Packets entering the network are forced to buffer in a flow queue at the ingress port and are only allowed to progress through the fabric as packets from the same flow depart downstream switches. This process limits the total buffering requirements of a flow within the fabric to an amount just large enough to sustain the flow's fair share of bandwidth at the congestion point, leaving the remainder of the buffers available to run uncongested flows at full speed. The endpoint output queues compute several metrics for each traffic class related to degree and rate of congestion and number of contributing flows. These are used to feed congestion information back via congestion acks. Different degrees of endpoint congestion can be reported via different types of congestion acks. These congestion acks are used to manage the bandwidth of flows and control the number of bytes allowed to enter the network.

To achieve effective flow rate control, Rosetta limits the number of bytes a flow of packets can have in the network at any point of time. The amount of unacknowledged downstream data belonging to a flow is tracked at every switch along the flow's path. As congestion acks are received from downstream switches, flows contributing to congestion will react by lowering their rates at each hop along the path, guided by the congestion ack values. Flows that are not contributing to the endpoint congestion are unaffected. This action keeps packets away from congestion hot spots, and minimizes buffer utilization by stalled packets, ensuring free buffer space for unrelated traffic and improving overall network throughput.

F. Performance

Performance data was measured on two prototype systems: Malbec and Shandy. Malbec comprises 485 Intel Xeon-based nodes, each with one 100 Gbps Ethernet NIC, connected by a 512-port dragonfly network comprising four groups of eight switches. Each switch has 16 host links operating at 100 Gbps, 28 local links operating at 200 Gbps, and six global links operating at 200 Gbps. Peak bandwidth of the global links is 4.8 TB/s, 79% of injection bandwidth. Shandy comprises 1024 AMD Rome-based nodes, each with a pair of 100 Gbps Ethernet NICs, connected by a 2048-port dragonfly network comprising eight groups of 16 switches. Each switch has 16 host links operating at 100 Gbps, 30 local links operating at 200 Gbps, and seven global links operating at 200 Gbps. Peak bandwidth of the global links is 11.2 TB/s, 44% of the injection bandwidth.

Tests were run using Cray MPI – derived from Argonne National Laboratory MPICH [3] – implementing the MPI-3.1 standard. Measurements were made with the OSU benchmark suite **Error! Reference source not found.**. Programming model libraries use the TCP/IP stack or the libfabric interface [4]. Libfabric is a core component of the OpenFabric Interface (OFI) framework. It is an abstraction layer that provides user applications with access to the network. The libfabric API was designed for high-bandwidth, low-latency NICs, with a goal to scale to tens of thousands of endpoints. The libfabric provider for Slingshot supports the reliable datagram (RDM) endpoint type. An RDM endpoint provides reliable, unconnected data-transfer semantics, supporting remote memory access, atomic memory operations (AMOs) and tagged-messaging operations. RMA operations are used to transfer data directly between local and remote data buffers. The Ethernet NICs generate RoCEv2 traffic. The network is presented with UDP packets containing up to 4K of payload data. All traffic is ordered.

Measurements show quiet network MPI latencies in the 1.5-1.8 µs range depending on the CPU type. Each switch hop adds approximately 300ns to the latency (including FEC on the cable between switches). Full bandwidth is achieved on a wide range of packet sizes.

Fig. 3 reports all-to-all bandwidth measured on Shandy using the MPI_Alltoall collective. Performance scales well with the number of nodes. Note the change in bandwidth at 256-512 bytes. MPICH uses Bruck's index algorithm [5] at small sizes, switching to a permutation as the message size increases. On this system 1/8th of the all-to-all traffic remains local to each group and 7/8 uses the global links. Peak all-to-all bandwidth of the system is 12.8 TB/s. The highest bandwidth measured was 11.4 TB/s, 89% of peak. Tests show that all-to-all bandwidth is maintained in the presence of congestion. Performance counters collected during the run show high numbers of reroutes as the network load-balances ordered flows. No packets were dropped during this test. It is worth underlining that these results were for Ethernet traffic generated by standard RoCE NICs.

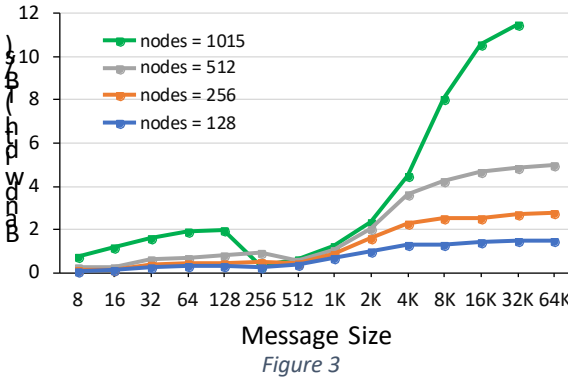

Figure 3

Measurements of Ethernet throughput and latency were performed using an IXIA tester operating at 200 Gbps in accordance with IEEE RFC 2544 [6]. The measured rate for 64B frames was 297,619,047 frames per second (FPS). No packets were dropped. Latency for Ethernet packets was measured at 308-327 ns depending on the combination of ports used. Measurements of the frame rate for smaller (40

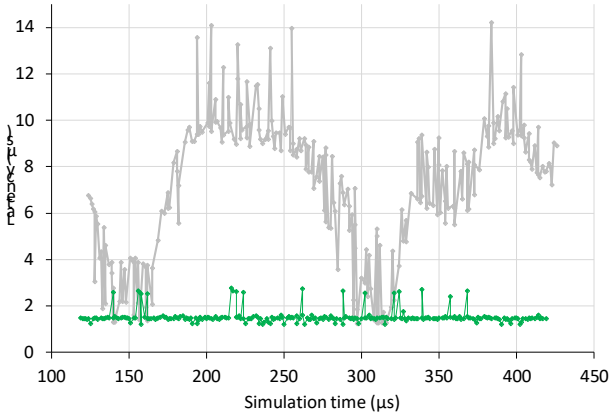

Figure 2. Trace packet latency with congesting background traffic

byte) frames were performed using Rosetta's built-in frame generator. The frame rate measured was 521 M FPS.

III. Dragonfly topology

Dragonfly topology-based networks offer rich bandwidth for nearest neighbor / local communications intensive applications and scalable cost-effective global bandwidth for very large applications with long reach communications patterns. HPE Slingshot's advanced congestion management, adaptive routing and QoS features maximize the effectiveness of the optimized dragonfly topology and minimizes job placement issues.

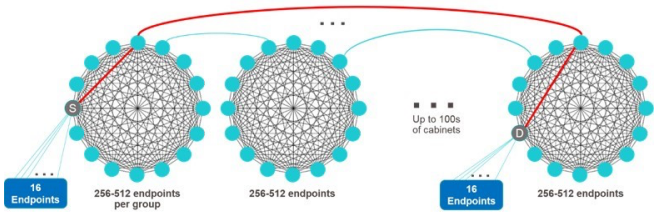

Figure 4. HPE Slingshot Extreme Scale and Performance with Dragonfly Topology

The dragonfly network topology is a low diameter "direct" network, avoiding the need for any external fat tree type "top" switches and reducing the number of costly optical links required for a given global bandwidth for large systems. The dragonfly topology design builds on earlier work showing the value of low-diameter networks constructed from high radix routers. Ideally, a router would have enough ports to connect to all the other routers in the system — the network would have diameter one. However, this topology is not practical for large systems. If a group of routers — acting in concert as a single, very high radix virtual router pooled their links, then the group could have enough links to directly connect to all the other groups in the system. This is the key idea behind the dragonfly network. Low-cost electrical links are typically used to connect the NICs in each node to their local router and the routers in a group – (L0 links).

Each router provides both "intra-group" links (L1 links) that connect the router to other routers in its group and "inter-group" links (also known as global or L2 links) that connect to other groups. The routers in a group pool their inter-group links, enabling each group to be directly connected to all the other groups. Optical links are used for the long links between groups.

There are several possible group sizes with the Slingshot dragonfly topology depending on several factors. These include the desired maximum scale of the system, the desired scalable unit of compute capacity to add at a time, and the optimal cost point driven by the trade-offs of cable cost and switch count depending on the rack density and physical layout.

Each dragonfly group has a fixed number of routers and provides a fixed maximum number of global ports. Sufficient global ports are provided to allow connections to a large number of other groups while network cost scales linearly with the system size. The number of inter-group links used can be adjusted to tailor the global bandwidth of the system.

The global bandwidth of a dragonfly network is given by:

$$Global\ bandwidth = link\ bandwidth \times global\ links\ per\ group \times groups$$

For a large system, the global bandwidth is twice the bisection bandwidth as each group has an equal number of connections to each of the other groups. Global bandwidth grows linearly with system size. In all but the largest systems, each group has many more global links than there are groups, thus multiple connections can be made between every pair of groups. Where full global bandwidth is not required, the number of global links can be reduced to minimize cost. Notably absent in dragonfly topologies are the discontinuities in cost that characterize fat tree networks as they are scaled up.

In a dragonfly network, the shortest path between any two nodes is known as the "minimal" path. It may be within one group or between groups. A minimal path between groups may require hops in the source and destination groups plus an optical hop between groups. Intra-group bandwidth is over provisioned to allow for the higher loading on the intra-group links.

Where traffic from each group to each of the other groups is balanced — as it is for all-to-all or uniform random communication — the links are equally loaded on minimally routed traffic. However, if all nodes in one group were to communicate with nodes in one other group, there are insufficient links to achieve full bandwidth with minimal routing. To achieve high bandwidth on this type of traffic, it is necessary to select "non-minimal" paths in which packets are adaptively routed intermediary groups. (HPE Slingshot supports adaptive routing of both ordered and un-ordered traffic effectively.) Non-minimal routing increases the aggregate network load but balances the traffic over all the available links and delivers scalable effective performance.

The scope of the advantage a dragonfly network has over a fat tree can be understood by the extent to which packets are minimally routed. With 100 percent minimal routing, dragonfly has a 2:1 network hop advantage — each packet takes one global hop at most, as opposed to two in a three-layer fat tree. If all traffic is non-minimally routed, then load on the global links is equal to that in a full bandwidth fat

tree. Traffic patterns generating high volumes of global traffic are well suited to minimal routing given good diversity in the range of destinations to which each node is communicating at any given point in time. Application traffic patterns with low path diversity require a higher proportion of non-minimal traffic. However, if the traffic is local, a high proportion of it is contained within a dragonfly group, then the load on the global links will be low. The large dragonfly group size with high-bandwidth optical network connections between groups ensures that systems are well suited to a wide range of HPC applications. Where high global bandwidth is not required, the optical network can be scaled back to reduce cost.

IV. CASSINI NIC

HPE Slingshot networks are designed to support 250,000 or more endpoints[1]. The largest system under construction has approximately 85,000. Support for systems of this scale has a significant bearing on the design of the NIC.

Cassini is a 200 Gbps HPC NIC ASIC chip developed by HPE. The host interface is PCIe Gen4 with support for extended speed mode (where supported by the CPU or GPU). The network link port conforms to the 200 Gbps (4×50 Gbps PAM 4) and 100 Gbps (4×50 Gbps NRZ) Ethernet standards.

Each node in a system is connected to the network by one or more Cassini NICs. In EX systems with multiple NICs per node all the NICs are attached to a single network slice. This network design, referred to as multi-port, allows each NIC to communicate with any other NIC in the system. It is used in preference to a multi-slice network (where device d on node n connects to port n of slice d). This implementation choice allows processes bound to a socket that is local to their NIC to send and receive using that NIC, improving performance.

A. Supporting HPC on Ethernet

As link speeds increase and the cost of developing new physical interfaces increases, high-speed networks are converging on the same set of physical standards. However, Ethernet was designed for local and wide area networks rather than the system but work within a supercomputer. Packet overheads are large and packet switching rates have been relatively low by the standards of HPC systems. Ethernet relies on being able to drop packets when there is network congestion. Transport protocols such as TCP/IP have developed in this environment. Remote direct memory access (RDMA) protocols used in HPC and storage applications perform badly under these conditions.

The ubiquity of Ethernet physical standards makes it attractive for an HPC vendor to develop networks that combine Ethernet interoperability with HPC functionality and performance. Cray took this direction with Slingshot and it is being continued at HPE.

HPE Slingshot devices negotiate a set of enhancements to Ethernet protocols using link layer discover protocol (LLDP). The minimum packet size is are reduced to 32 bytes and inter-packet gaps are removed. Header sizes are reduced. Packet rates are increased. Link reliability is greatly increased through use of link level retry (LLR) and reduced lane operation. Packet drops are avoided in almost all error and congestion cases. An efficient transport layer protocol has been developed for message passing and remote memory access (RMA). While some of these enhancements are specific to the fabric links connecting Rosetta switches, others can be widely adopted: The low latency forward error correction FEC[2], RS(272,257+1,7,10) is being used by the 25-Gigabit Consortium and flow based congestion management is an active area of research.

This new functionality is fully interoperable with standard Ethernet protocols. The link between a Cassini NIC and a Rosetta switch can carry Ethernet frames, v4 and v6 IP traffic, MPI messages, one-sided operations, low latency barrier operations, and RoCE storage traffic simultaneously. Traffic is assigned to quality of service (QoS) classes using the DiffServ model (IETF Network Working Group, 1998).

B. Reliability

Cassini provides upper layers of software with a reliable RMA and messaging transport. Where a packet is lost, or resources are exhausted the transport will take the actions required to complete the operation. A fatal error is only returned when endpoints remain disconnected for some period.

Support for LLR on links between HPE Slingshot devices and use of ECC within those devices ensures that packets are not dropped as a result of soft errors, but they can be dropped when a device or cable fails. End-to-end retry is required for these cases.

Some network operations are idempotent, they can be replayed. Examples of this include one-sided Put and Get operations. Other operations must be executed once and once only. AMOs are one well-known example, but there are other widely used operations that cannot be replayed. MPI message matching and setting a target side completion event are important cases.

Cassini provides two reliability models: a simplified or “restricted” model for idempotent operations and an “unrestricted” model for operations that manipulate target state. The restricted model is connectionless. The unrestricted model uses dynamically allocated connections that persist while in use. It stores the result of an operation, checking its store for the result before executing a retry, and releasing the result as the operation completes.

Consider execution of a fetching AMO. In normal operation the AMO request is executed and the result is stored. Storage space is released when the operation completes. If the original request is dropped a retried request will find the result store empty. The AMO can be executed safely. If it was the response that was dropped, a retried request will find and return the original result rather than execute the operation.

Maintaining the state necessary to perform a retry is expensive – it can be more expensive than performing the operation itself. Cassini takes away this burden, maintaining the state required for retry in hardware. Use of LLR reduces the rate of packet loss and hence the rate of end-to-end retry.

[1] The largest dragonfly network that can be constructed has 511 groups of 512 nodes.

[2] Ethernet standards mandate use of FEC at speeds of 100 Gbps or higher so as to retain a bit error rate (BER) of 10^{-12} or better.

This allows policy on when to retry to be implemented in software.

C. Ordering and Completion

MPI requires point-to-point ordering of message matching. If a given sender sends two messages to the same destination and both can be matched by a single receive, the receive must match the one that was sent first. Similarly, if a receiver posts two receives and both match the same message, the message must be matched to the first receive. These semantics have a significant bearing on the design of offload NICs.

PGAS programming models require address ordering. Where a write to an address is followed by either a read (raw) or a second write (waw) the second operation cannot start until the result of the first is globally visible. Implementing address ordering in software is expensive. A process or thread must block until it has received confirmation that the first operation has completed. Offloading this step to the NIC is desirable. Cassini does this through use of a fence command. It ensures that operations ahead of the fence have completed before those behind the fence are issued. The progression is in hardware.

Where there is one path between initiator and target, a response can be returned, releasing a fence, when the target write has reached a point of global ordering in the target NIC. For Cassini, this point is on the NIC side of the host interface. Where there are multiple paths between nodes, a response must be delayed until the first operation has reached a point of ordering within the target node. This point is within the target node's PCIe root complex. Cassini provides to the option to specify which behaviour is required on an operation-by-operation basis (in libfabric terms, selecting between the FI_TRANSMIT_COMPLETE and FI_DELIVERY_COMPLETE options).

Completion of an operation is notified by generating events at the initiator and (optionally) at the target. Completion events (and other events indicating errors) are written to an event queue in user memory. MPI point-to-point messaging and most RDMA protocols require individual completion events in the sending and receiving processes, with details of the transfer that has completed. Other APIs use bulk completion, with notification being provided when a set of operations have completed successfully or at least one has failed. This lighter-weight model is known as counting events. It was first provided by Quadrics (Beecroft, et al., 2005) and has been adopted by Portals and libfabric. A counter is incremented as each operation completes. A completion action is triggered when the counter reaches a threshold. This action may be a full completion event or a new triggered operation.

D. Software APIs

Libfabric (OpenFabric Working Group, 2018) provides a set of APIs that match the needs of HPC programming models. They can be implemented in software over hardware interfaces such as verbs or the hardware can implement up to the software interface. Cassini does the latter, providing direct hardware support for many of the libfabric interfaces. The Cray programming environment layers over libfabric providing MPI, OpenSHMEM, UPC, Chapel and a wide range of tools. This environment is now portable across Cray and HPE systems implementing libfabric. Other MPI implementations including MPICH (Argonne National Laboratory, 2021) from which Cray and Intel MPI are derived and OpenMPI (Open MPI Project, 2020) also support libfabric.

The Portals API (Barrett, et al., 2014) provides a set of interfaces designed for scalability and performance. Portals was traditionally focussed on message passing, but version four adds support for one-sided remote memory access. Cassini adopts Portals ideas relating to message matching, completion and triggered operations, but it does not attempt to offload the whole API.

Cassini provides direct support for Ethernet APIs used in a modern Linux stack, the generic send offload (GSO) and generic receive offload (GRO) in particular. Hardware support includes calculation of checksums (including RoCE CRCs), controlling how incoming Ethernet packets are distributed over endpoints (referred to as receive side scaling or RSS), and placement of data into target memory. Ethernet packets can be created by user processes (where permitted) or driver threads.

V. Early Performance Results

This section provides preliminary data on Cassini performance measured on the Shadow system. Tests use up to 64 MPI ranks per NIC, 128 ranks per dual socket node. Figure 5 plots MPI point-to-point bandwidth using one NIC or two. The OSU benchmark test (Ohio State University, 2021) used 16 processes per node. Where the system has multiple NICs, the MPI implementation divides the ranks over the NICs. The receive queue for that rank is offloaded it its NIC. All other ranks send messages for that process to its NIC.

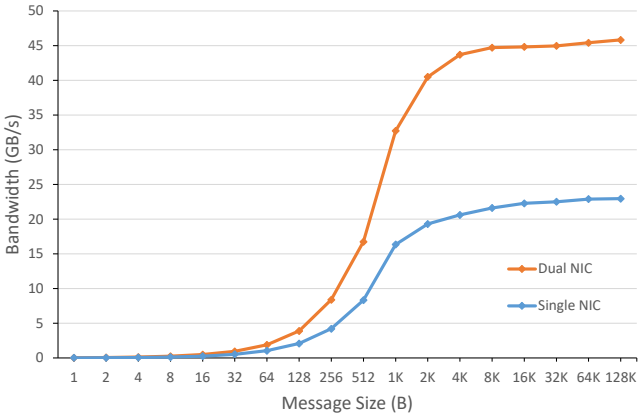

Figure 5: MPI point-to-point bandwidth

Figure 6 illustrates RMA bandwidth as a function of message size for increasing numbers of processes per NIC. Measurements were made on a single NIC using a SHMEM performance test. Half peak bandwidth of 12.5 GB/s is achieved for transfer sizes of approximately 128 bytes using 16 or more PEs per NIC. Asymptotic bandwidth is achieved on transfer sizes of between 1K and 2K depending on the number of active PEs.

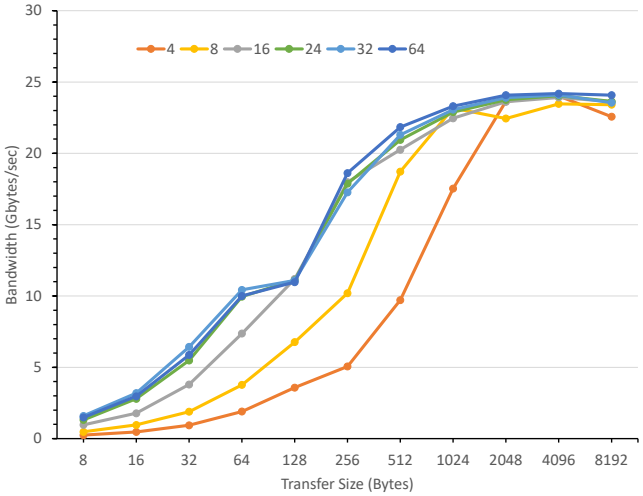

Figure 6: RMA bandwidth as a function of Message Size

Figure 7 illustrates MPI latency for Intel Xeon and AMD Rome CPUs. Latency is measured with the OSU benchmark test using nodes connected by a NIC-switch-NIC path. This measurement includes the full prototype software stack, two NIC crossings, the FEC latency on both links, and one switch crossing. Latency is the same for messages of between 0 and 96 bytes, all of which are issued using a write to the NIC. For payload sizes above 96 bytes the command and any inline data is read by the NIC. Latency increases by the DMA read latency of the CPU.

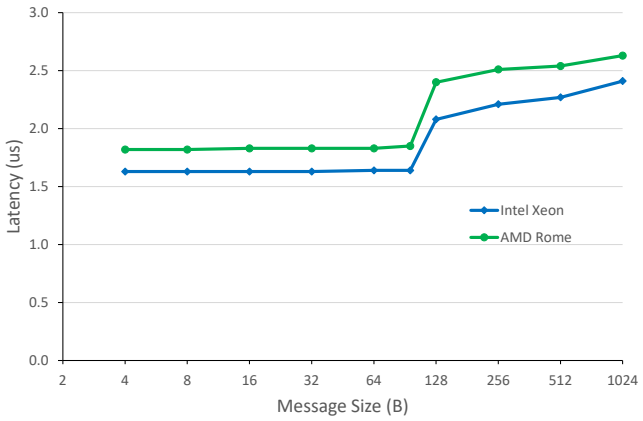

Figure 7: MPI Latency

Figure 8 plots MPI message matching rates for increasing numbers of processes per node. The OSU message rate test was modified to include a number of failed match attempts for each message. This allows the dependency of matching rates on list length to be determined. The results demonstrate that Cassini meets its target of an average of 64 match attempts per message without degrading the message rate.

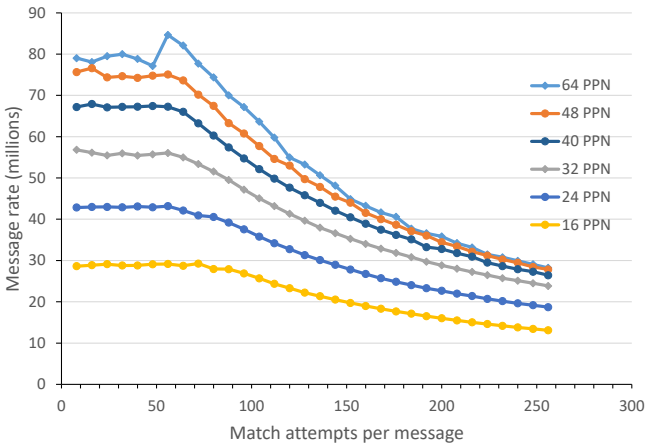

Figure 8: MPI matching rates as a function of match attempts per message

The Sandia message rate test (Sandia National Laboratory, 2021) measures performance under conditions that are more representative of an application: the cache is invalidated before sending messages, messages are sent and received at the same time, and each process communicates with a number of peers. There are four tests:

- The single direction test is similar to a traditional message rate test except that the cache is invalidated mimicking the application using the cache between communication phases,
- In the pair-based test, each process exchanges messages with a number of peers
- In the pre-posted test, each process pre-posts receives before cache invalidation.
- The all-start test mimics an application which finished a computation phase, issues all of its communication calls and then wait for them all to complete.

Aggregate rates for processes using one NIC are shown in Figure 9 to compare with those from the simple micro-benchmark. The pair-based test reaches the message rate seen in the best-case test but requires more processes to do so. The CPU is limiting performance on the other three tests.

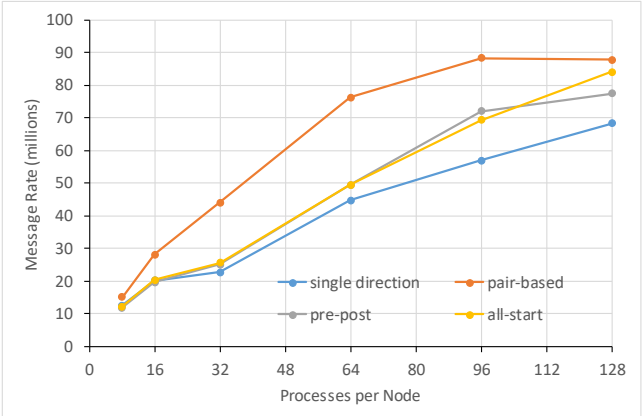

Figure 9: Aggregate message rates measured with the Sandia test.

VI. HPE Slingshot Deployment Status

HPE Slingshot fabrics have been deployed since 2020. As of the publishing date of this paper, all production deployments are based on the Rosetta switch, along with the NVIDIA CX5 NIC. Deployments with Cassini started in fall of 2021, all of which are currently in a pre-production state.

HPE Slingshot is represented by 8 top 500 [7] entries on the November 2021 list, with the NERSC Perlmutter system topping the HPE Slingshot entries at number 5. It is expected that additional HPE Slingshot based entries will be submitted for the summer 2022 top 500 list.

A. Early Cassini Deployment Observations

As mentioned above, several Cassini based deployments are under construction. Early performance data indicates that the in-house results presented in previous sections hold true in larger (pre) production environments. HPE is limited in what can be shared in terms of deployment types, size and performance results as all of such data is owned by the end customer. We fully expect some of these customers to publish some of their results in due time.

VII. Conclusions

HPE Slingshot brings the performance and functionality of supercomputer networks to Ethernet. Significant new functionality is added in the areas of congestion management and adaptive routing. The design overcomes many of the obstacles to adoption of standards-based networks in HPC: latencies and packet overheads are low, performance variability is minimized, links are hardware-protected, and scalability is sustained. The results demonstrate that an Ethernet network can deliver excellent performance on HPC workloads.

Rosetta implements a novel, flow-based congestion control mechanism that minimizes switch buffer utilization and provides freedom from head-of-line blocking across the entire fabric, greatly reducing tail latency and performance variability. GPCNeT results demonstrate that the mechanism almost completely eliminates interference from congesting applications, and significantly reduces interference from bandwidth-intensive applications sharing a network with tapered global bandwidth.

Measured results from early Rosetta-based systems validate the system simulation results. They demonstrate that systems with up to 2048 ports achieve high bandwidth, low latency, high packet rates, and excellent tail latencies, even in the presence of congesting traffic. Systems with tens of thousands of network of endpoints are being deployed in 2022.

The Cassini NIC provides high performance for MPI messaging, RMA access in PGAS programming models and Ethernet. Matching is offloaded to the NIC, minimizing use of CPU resources and freeing up memory bandwidth for the application.

The HPE Slingshot network, comprising Cassini NICs and Rosetta switches, is being used in HPE Cray EX supercomputer systems including all three of the US Exascale systems. It is also available in HPE Apollo clusters.

References

[1] Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., Liron, Y., Padhye, J., Raindel, S., Yahia, M.H. and Zhang, M., 2015. Congestion control for large-scale RDMA deployments. ACM SIGCOMM Computer Communication Review, 45(4), pp.523-536.

[2] Ramakrishnan, K., Floyd, S. and Black, D., 2001. RFC3168: The addition of explicit congestion notification (ECN) to IP.

[3] MPICH - High-Performance Portable MPI. https://www.mpich.org/.

[4] Grun, P., Hefty, S., Sur, S., Goodell, D., Russell, R.D., Pritchard, H. and Squyres, J.M., 2015, August. A brief introduction to the openfabrics interfaces-a new network api for maximizing high performance application efficiency. In 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects (pp. 34-39). IEEE.

[5] Bruck, J., Ho, C.T., Kipnis, S., Upfal, E. and Weathersby, D., 1997. Efficient algorithms for all-to-all communications in multiport message-passing systems. IEEE Transactions on parallel and distributed systems, 8(11), pp.1143-1156.

[6] Ginsberg, E., Alston, V., Wild III, A.A., Sheth, A., Liu, W. and Periakaruppan, R., Ixia, 2009. Generating traffic for testing a system under test. U.S. Patent 7,516,216.

[7] https://www.top500.org/