Intel® Omni-Path Architecture

Enabling Scalable, High Performance Fabrics

Mark S. Birrittella, Mark Debbage, Ram Huggahalli, James Kunz, Tom Lovett, Todd Rimmer, Keith D. Underwood, Robert C. Zak

Data Center Group
Intel Corporation
USA

{Mark.S.Birrittella, Mark.Debbage, Ram.Huggahalli, James.Kunz, Thomas.D.Lovett, Todd.Rimmer, Keith.D.Underwood, Robert.C.Zak} @intel.com

***Abstract*—The Intel® Omni-Path Architecture (Intel® OPA) is designed to enable a broad class of computations requiring scalable, tightly coupled CPU, memory, and storage resources. Integration between devices in the Intel OPA family and Intel CPUs enable improvements in system level packaging and network efficiency. When coupled with the new user-focused open standard APIs developed by the OpenFabrics Alliance (OFA) Open Fabrics Initiative (OFI), host fabric interfaces (HFIs) and switches in the Intel OPA family are optimized to provide low latency, high bandwidth, and high message rate. Intel OPA provides important innovations to enable a multi-generation, scalable fabric, including: link layer reliability, extended fabric addressing, and optimizations for high core count CPUs. Datacenter needs are also a core focus for Intel OPA, which includes: link level traffic flow optimization to minimize datacenter jitter for high priority packets, robust partitioning support, quality of service support, and a centralized fabric management system. Basic performance metrics from first generation HFI and switch implementations demonstrate the potential of the new fabric architecture.**

Keywords—fabric; high performance computing; datacenter; scalability; reliability

I. Introduction

The Intel® Omni-Path Architecture (Intel® OPA) defines a next generation fabric with its heritage in the Intel® True Scale [1] product line and the Cray Aries interconnect [2]. The Intel OPA is designed for the integration of fabric components with CPU and memory components to enable the low latency, high bandwidth, dense systems required for the next generation of datacenter. Fabric integration takes advantage of locality between processing, cache and memory subsystems, and communication infrastructure to enable more rapid innovation. Near term enhancements include higher overall bandwidth from the processor, lower latency, and denser form factor systems.

The first implementation of Intel OPA-based products focuses on supercomputing as well as the broader high performance computing environments; however, Intel OPA is generally applicable to a class of data center level computation [3] requiring scalable, tightly coupled, CPU, memory, and storage resources. Intel OPA defines a single interoperable OSI Layers 1 and 2 architecture that can be used to provide connectivity between elements in energy efficient supercomputer systems (e.g. based on Intel® Xeon Phi™ product family), mission critical enterprise computer systems (e.g. based on Intel® Xeon® processors), and inexpensive data center servers (e.g. Intel® Atom™).

To enable the largest scale systems in both HPC and the datacenter, fabric reliability has been substantially enhanced by combining the link level retry typically found in HPC fabrics with the conventional end-to-end retry used in traditional networks. Layer 2 network addressing has also been extended to account for systems with over 10 million endpoints – enabling the largest scale datacenters for years to come. To enable support for a breadth of topologies, Intel OPA provides mechanisms for packets to change virtual lanes (VLs) as they progress through the fabric. In addition, higher priority packets are able to preempt lower priority packets to enable both efficient multi-application environments as well as reduced latency jitter for latency sensitive traffic. Finally, fabric partitioning is provided to isolate traffic between jobs or between users.

A full treatment of the software ecosystem is beyond the scope of this paper, but several elements are key to enabling that ecosystem. The Open Fabrics Alliance (OFA) work on the Open Fabric Interface (OFI) [4] is the long term direction for high performance user level and kernel level network APIs. The Intel OPA Fabric Management Software Suite extends Open Fabrics Enterprise Distribution (OFED) Verbs for management interfaces, and OFED Verbs data movement APIs are supported for legacy applications. In addition, Intel OPA extends the Performance Scaled Messaging (PSM) API to provide HPC focused transports and an evolutionary software path from Intel True Scale. Higher level communication libraries, such as the Message Passing Interface (MPI), and Partitioned Global Address Space (PGAS) libraries are layered on top of these low level APIs.

The rest of this paper is structured as follows: Section 2 provides and overview of the Intel OPA and provides context for subsequent sections; Section 3 covers the Link Transfer Protocol – describing a number of innovative mechanisms allowing reliable data transfer, channel management, flow control management, and physical layer support; Section 4 covers the Link layer – including the mechanisms and abstractions that enable forwarding of packets between fabric

endpoints (HFIs) via a topology of switching elements. Section 5 covers the Application layer and key software components; Section 6 provides preliminary descriptions and performance data for early Intel OPA implementations; and Section 7 covers related work.

II. Intel Omni-Path Architecture (OPA) Overview

Fig. 1 shows a simplified block diagram of an Intel OPA fabric and attached devices. Connectivity to the fabric is via Host Fabric Interfaces (HFIs). Switches may be used in various topologies to connect a scalable number of endpoints. A redundant Fabric Manager provides centralized provisioning and monitoring of fabric resources. Partitions provide a way to isolate groups of endpoints while providing access to shared services.

A. Intel OPA Network Layers

By convention, the Intel OPA architecture is described in a series of Layers, corresponding, notionally, to a subset of the OSI network stack [5]:

- Layer 1.5: Link Transfer Protocol. Responsible for reliable delivery of Layer 2 "packets" as well as flow control and link control information across a link.
- Layer 2: Data Link Layer. Including fabric addressing, switching and resource arbitration mechanisms, and partitioning support.
- Layer 4-7: Application layer. Including the interface between software libraries and the Intel OPA HFI.

Note that the Intel OPA Layer1 (PHY) leverages PHY standards used in both InfiniBand [6] and Ethernet [7], and will not be discussed further in this paper.

B. Host Fabric Interface

Each host is connected to the fabric via a Host Fabric Interface (HFI). HFIs bridge between the semantics of the host processor and the semantics of the fabric. This minimally consists of the logic necessary to implement the physical and link layers of the fabric architecture, such that a node can attach to a fabric and send and receive packets to other servers or devices. An HFI may include specialized logic for executing or accelerating upper layer protocols. An HFI must also support whatever logic is necessary to respond to messages from network management components.

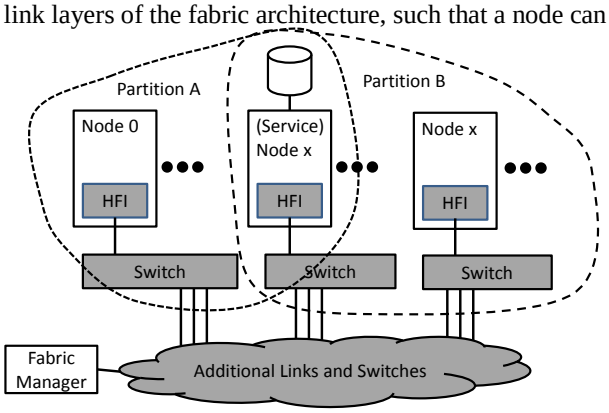

Figure 1: Intel Omni-Path Architecture Elements

C. Intel OPA Switches

Intel OPA switches are Layer 2 devices, and act as packet forwarding mechanisms within a single OPA fabric. Intel OPA switches are responsible for implementing Quality of Service (QoS) features, such as adaptive routing and load balancing. Switches also implement the Intel OPA congestion management functions. Switches are centrally provisioned and managed by the fabric management software, and each switch includes a management agent to respond to management transactions. Central provisioning means that switch configurations are programmed by the fabric management software, including managing the forwarding tables to implement specific fabric topologies, configuring the QoS parameters, and providing alternate routes for adaptive routing. As such, all OPA switches must include Management Agents which communicate with the OPA Fabric Manager.

D. Intel OPA Management

The Intel OPA includes a centrally managed fabric supporting redundant Fabric Managers that provision and manage every device (HFI, switch, gateway) in the fabric through management agents associated with those devices. The Primary Fabric Manager is a software component of a Intel OPA fabric that is selected during the fabric initialization process. The Primary Fabric Manager is responsible for: discovering the fabric's topology; provisioning the fabric components with Fabric Identifiers and other necessary values for fabric operations; formulating and provisioning the Switch forwarding tables; maintaining the Fabric Management Database; and monitoring fabric utilization, performance and error rates

Fabric management natively occurs in-band using dedicated buffers in switches on a specific VL (VL15) for management packets. The management VL may be configured to operate with or without flow control. Without flow control, management packets will be dropped if queue resources are not available at a port. End to end reliability protocols are used to detect dropped packets.

III. Layer 1.5: Link Transfer Layer

The Link Transfer (LT) Layer serves as the interface between the Physical Layer and the Link Layer. The LT layer segments end to end Fabric Packets (FPs) into 64 bit Flow Control Digits (FLITs), and groups 16 FLITs into a Link Transfer Packet (LTP) to reliably transport FP FLITs and control information on the link.. Each LTP is protected using a link LCRC, and is retransmitted when an error occurs. The temporary, local to the link, LTPs run asynchronous to the end to end FPs and can contain FLITs from multiple FPs. Each LTP is striped across up to four lanes provided by the Physical Layer.

A. FLITs and FLIT Types

Each FLIT contains 64 bits of payload and has an associated type bit making it 65 total bits. The *type* bit distinguishes body FLITs from other FLIT types (e.g. head flits) using a Huffman-like encoding scheme. Body FLITs are encoded with the FLIT *type* bit set to 1. All other FLITs are marked with the *type* bit set to 0, and the remainder of their type encoding encroaches onto the space typically reserved for Layer 2 fields. Head FLITs are encoded with FLIT[63] set to 1 (i.e. 01). All other (non-body/non-head) FLITs are encoded with FLIT[63] set to 0. Tail

FLITs are encoded with FLIT[62] set to 1 (i.e. 001). All other (non-body/non-head/non-tail) FLITs are encoded with FLIT[62] set to 0 (i.e. 000). Control FLITs are used to orchestrate the retransmission protocol. Command FLITs are used to return VL flow control credits [8] back to the transmit side of the link. FP FLITs and Command FLITs can be mixed within reliable LTPs for transmission over the link. Control FLITs are sent only in special null LTPs and are not part of any fabric packet. Idle FLITs are inserted into the continuously transmitted LTPs when no FP FLITs are available at the egress port feeding the link.

B. Link Transfer Packets (LTPs)

A single size Link Transfer packet holds sixteen FLITs and the associated FLIT *type* bits for transmission over the link. In addition to the sixteen FLITs each LTP has a two bit VL credit sideband channel and a 14 bit LCRC which covers the contents of each individual LTP. There are a total of 128 Bytes (16 FLITS) of data payload and 4 Bytes of overhead (16 FLIT type bits, 14 CRC, and 2 VL credit bits) resulting in a transfer efficiency of 64/66. There are two types of LTPs. Reliable LTPs contain FP FLITs and VL credit return FLITs and are held in a replay buffer for period of time that is long enough to guarantee that a lack of a retransmission request indicates it has been received successfully by the peer. Null LTPs do not consume space in the replay buffer and are never retransmitted. They are distinguished using a Control FLIT which specifies a specific operation in the retransmission protocol..

C. Interface to the Physical Layer

Each LTP is segmented into link transfer quantities without regard to FLIT and type bit boundaries for transmission over a 1 to 4 lane wide link. Asymmetric link widths of 1x, 2x, 3x, and 4x are supported. Bit alignment within each lane, lane polarity adjustment, peer physical lane identification detection for lane swizzle adjustment, packet framing, and lane de-skew are all accomplished by the Link Transfer (LT) Layer completely external to the physical layer. The LT Layer utilizes an additive/synchronous LFSR-based bit scrambler on the transmit side and de-scrambler on the receive side. The scrambler is turned on and synchronized once using a turn-on frame at the end of the link initialization process. The scrambler provides a high enough probability of edges such that 64/66b type encoding is not required. This enables encoding LTPs at the same link transfer efficiency as basic 64/66b encoding.

D. Intel® Omni-Path Packet Integrity Protection (PIP)

Intel Omni-Path PIP is used to enhance the reliability of transmission across the link. When a failed CRC check occurs, a retry request containing the failed LTP sequence number is sent to the peer informing it to retransmit the failed LTP and all subsequent LTPs. The receive side drops all LTPs until the retry begins, as indicated a special retry marker null LTP. A replay buffer is used to provide enough temporary storage to cover the round trip time of a maximum length cable. The retransmission protocol uses implied acknowledgments to reduce overhead. After link initialization, the transmitter sends a special retry marker null LTP indicating that reliable LTPs will commence immediately. In response, the peer sends a special one-time round trip marker null LTP back to inform the transmitter that the first round trip is complete. As long as a retry request is not received, LTPs are implicitly acknowledged as being successfully received by the peer. The protocol supports link round trip times that exceed the replay buffer depth by sending empty null LTPs when the replay buffer is full. Round-trip time is established by counting the LTPs required while waiting for the initial round trip marker.

E. Intel® Omni-Path Traffic Flow Optimization (TFO) and Interleave

The Link Transport Layer permits FLITs from different packets on different VLs to be interleaved within and across LTPs when they are sent across the link. This allows both higher link utilization, and lower latency for high priority packets. A FP using a high priority VL arriving at the link egress point can preempt an in-progress FP in order to minimize the latency of the high priority FP. Transmission of the in-progress FP is suspended to allow transmission of the high priority packet. Once the high priority FP is transmitted the suspended packet resumes. A low priority FP can be preempted multiple times at an individual link egress point. The link egress point monitors the total amount of time that a low priority FP is delayed by preemption by multiple high priority FPs, and allows the low priority FP to complete if a limit, configured by the fabric manager, is exceeded. Bubbles in a FP are defined as the lack of available FLITS for the in-progress FP, or underrun, at the link egress point. If a sending FP for whatever reason is interrupted by bubbles, FLITS from a second packet can utilize the channel instead of propagating Idles on the channel. In this case the substituted packet can be of equal or lower priority. When FLITS are again available for first packet the link egress point can return to transmitting the first packet, or wait until the completion of the second packet.

F. Virtual Lane Credit Management

The transmit side of the link is informed of the total physical buffer space (in FLITs) on the partner as part of the link initialization process. The receive side treats the space as a single pool for all VLs. The transmit side manages this space on a per VL basis. Up to 31 FP VLs are supported in addition to a single management VL. A dedicated space is maintained for each VL as well as a shared space for all VLs. The ratio of fixed to shared space and the dedicated per VL space size can be adjusted dynamically by the fabric manager. Per VL incremental acknowledgements are returned in units of 8 FLITs as FLITs are removed from the buffer pool for a particular VL. These acknowledgements travel over the link in the 2 bit VL *credit* field in each LTP. 2 bits from 4 sequential LTPs are aggregated into an 8 bit field which is used to indicate the VL and number of 8 FLIT units being returned. There are also optional Command FLITs that can be used to return VL credits across the link. The 2 bit VL LTP *credit* field and the Command FLITs are retransmitted during retry sequences such that no incremental VC credits are lost when a link error occurs.

IV. Layer 2: Link Layer

The Intel OPA Link Layer (Layer 2) is designed for large scale systems. Layer 2 fabric packets enable 24-bit fabric addresses, as well as optimized formats for smaller systems. Up to 10KB of payload can be carried in a fabric packet after accounting for the largest Transport Layer header. Service Channels (SCs) and Virtual Lanes (VLs) provide the building

blocks to enable support for a broad class of topologies as well as the implementation of Quality of Service (QoS) features.

A. Quality of Service Support

Within OPA, Quality of Service (QoS) features provide a number of capabilities, among them: job separation/resource allocation; service separation/resource allocation; application traffic separation within a given job; protocol (eg. request/response) deadlock avoidance; fabric deadlock avoidance; traffic prioritization and bandwidth allocation; and latency jitter optimization by allowing traffic preemption

The Intel OPA provides a very flexible capability for QoS via Virtual Fabrics (vFabrics), Traffic Classes (TCs), Service Levels, (SLs), Service Channels (SCs) and Virtual Lanes (VLs). At the heart of QoS is the SC mechanism which is used to differentiate fabric packets within the fabric. In order to support a wide variety of fabric topologies and configurations, SC assignments are managed by the fabric manager and the SC of a given fabric packet may change as it traverses the fabric to routing dependent deadlock.

The application and sysadmin operations are centered around vFabrics. A vFabric is the intersection of a set of fabric ports and one or more application protocols. For each vFabric a set of QoS and security policies are established by the sysadmin. A given vFabric is associated with a Traffic Class for QoS and a Partition for security.

A Traffic Class represents a group of SLs that a given Transport Layer or application will use. Some Transport Layers may use multiple QoS levels to avoid deadlock (such as separating request and response in certain protocols), while others may take advantage of multiple QoS levels to separate high priority control traffic from lower priority data traffic. Transport layers may simply associate a traffic class with a single SL. The Intel OPA allows up to 32 Traffic Classes, but 4 to 8 is expected to be a more typical configuration. Traffic Classes are realized through the end to end concept of a Service Level. Traffic Classes may span multiple Service Levels, but a Service Level may be assigned to only one Traffic Class. Intel OPA will support up to 32 Service Levels, but 4 to 8 is expected to be a more typical configuration. Service Levels are the lowest layer QoS concept visible to OPA Layer 4 protocols and applications.

Underlying Service Levels are Service Channels (SCs), which differentiate packets of different Service Levels as they pass through the fabric. The SC is the only QoS identifier contained in the fabric packets, which reduces packet overhead. In some fabric topologies, Service Levels may span multiple Service Channels, but a Service Channel may be assigned to only one Service Level. The Intel OPA supports 32 SCs; however, SC15 is dedicated to in-band fabric management.

In each endpoint Service levels are mapped to SCs via SL2SC tables at transmission (mapping each SL to an SC), and SC2SL tables (mapping inbound SCs to a given SL with perhaps multiple SCs mapped to the same SL).

Fig. 2 shows an example usage of TC, SL and SC. In this example we see two HFI endpoints connected via an 8 switch hop route through the fabric. Two TCs are used, one for a request/response protocol (e.g. PGAS) assigned to TC0, and one for storage assigned to TC1. The request/response protocol on TC0 requires two SLs (SL0 and SL1), while the storage protocol on TC1 only requires a single SL (SL2). Each SL is assigned a pair of SCs (SC0/SC1, SC2/SC3 or SC4/SC5) for topology deadlock avoidance in the fabric, such as what is typically used in a torus topology. As the packets traverse the fabric, the SC may change link by link, however, the SL and TC seen by Layer 4 and the application is consistent end to end.

Within a given Link, Service Channels are assigned to Virtual Lanes (VLs). VLs provide dedicated receive buffers for incoming Fabric Packets. VLs are also used for resolving routing deadlocks. The Intel OPA supports up to 32 VLs, though the actual number supported will be implementation-dependent. The Intel OPA supports mapping SCs onto VLs, such that heterogeneous fabric configurations can be supported. SC15 is always mapped to VL15 for in-band fabric management. Individual implementations may choose to support fewer than 32 VLs or may be configured to optimize the amount of buffering per VL by reducing the overall number of VLs available.

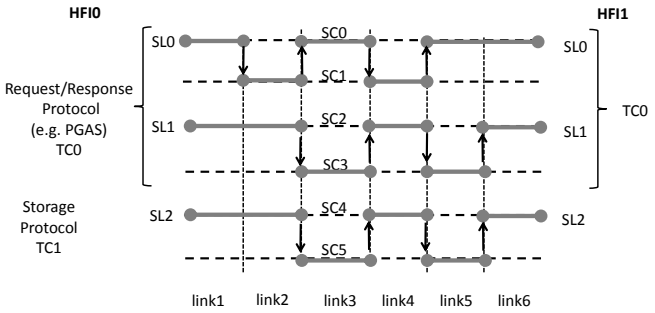

Figure 2: Example of Traffic Class (TC), Service Level (SL), and Service Class (SC) usage in a topology with credit loop avoidance

Each SC carries traffic of a single service level in a single traffic class, and the fabric manager configures how SCs map onto the VL resources at each port. The large number of SCs helps maintain performance at high levels of utilization, and mapping a Service Channel to an independent VL can provide an independent channel through the fabric. Thus, Intel OPA makes extensive use of Service Channels (SC) to avoid routing and protocol deadlocks and to avoid head of line blocking between traffic classes. However, practical constraints may cause any given implementation to support a smaller number of VLs on one oe more links. In these cases, multiple SCs may share a VL, and the fabric manager is responsible for configuring which SCs can correctly share a VL and how SCs share VLs so that QoS characteristics can be honored.

The fabric manager also configures how the transmission of packets across a Virtual Lane is scheduled via a configurable VLArbitration algorithm. In addition, packet preemption can be configured to permit a higher priority traffic to preempt a lower priority packet. This provides further reductions in head of line blocking for high priority traffic.

B. Congestion Management

Intel OPA takes a multi-faceted approach to managing congestion. Both medium grained adaptive routing and

dispersive routing provide load balancing to avoid hot spots in the fabric. Medium Grained Adaptive routing identifies congested inter switch links and dynamically adjusts the routing of traffic to better utilize other links in the fabric. To minimize the impact on transport layers which are not prepared for highly out of order packet delivery, this mechanism limits its frequency of adjustment.

In turn, dispersive routing probabilistically distributes traffic across multiple routes through the fabric or even across multiple virtual lanes within a single route. Dispersive routing makes use of multipathing, which allows multiple paths to be defined between a pair of endpoints so that a source can spread its traffic across the multiple paths. This reduces the formation of hot spots in the fabric which can result from unbalanced traffic patterns. There is no ordering defined across packets that use different paths. Where ordering is required between a pair of packets the source must send the packets on the same path, or use other mechanisms to guarantee ordering or recover from out of order delivery. The Intel OPA does not specify how endpoints choose which path to use for a given packet. Endpoints may use any of the available paths as long as they maintain their own ordering requirements.

When these techniques work well, they avoid congestion at intermediate points in the fabric by balancing utilization across all links for traffic patterns which do not oversubscribe any endpoints. However, some traffic patterns do oversubscribe some endpoints. In these cases, many sources are sending to a single destination endpoint at a rate that exceeds the bandwidth of the link to the destination endpoint. This traffic pattern can cause congestion trees to form as queues fill throughout the fabric due to the destination asserting link level flow control. Congestion trees are not an inherent problem; however, they often block the progress of packets flowing to an uncongested destination. The Intel OPA defines an explicit congestion notification protocol that marks packets passing through congested switch ports. When the packet reaches its destination a backward notification is returned to the source to cause the source to reduce the bandwidth of packets to that destination. This can permit better fairness for all senders as well as reduce the impact on other flows that may be sharing the same VLs on the same links as the congested flows. Over time the source will gradually begin to increase bandwidth, and either explicit congestion notification will cause it to continue running at a reduced bandwidth or it will return to full bandwidth.

C. Partitions

Partitions are an isolation mechanism that operate at the Link Layer (Layer 2), with every Fabric packet being associated with a single partition.. A partition provides isolation to the group of endpoints that are members of the partition for all types of traffic (i.e. all Transpor Layers); however, this does not prevent a Transport Layer from providing finer grained security mechanisms. A partition in an Intel OPA fabric contains a group of Intel OPA endpoints. Communication is allowed within the partition, and is prevented with endpoints that are not in the partition. This allows partitions to be used to provide isolation between applications running on a fabric or between users on a fabric.

Individual endpoints may be identified as a full or limited member of a given partition. Full members are permitted to communicate with any member of the partition, but limited members are only permitted to communicate with full members. This mechanism permits the fabric to have shared services, such as management or a common global file system, while preserving isolation between non-service partitions. Such services often require all end points to be members of the partition for that shared service. By making the providers of the shared service full members and the clients limited members, clients may access the service while still preventing clients from communicating directly to each other.

Each endpoint is a member of at least the management partition, and may be a member of multiple partitions. A typical endpoint will be a member of at least one application partition. When a Host node has multiple endpoints, each endpoint may have membership in different partitions. There are no architectural restrictions regarding which endpoints may be members of which partitions. For example, partitions may overlap and partitions need not be related to the physical topology of the fabric.

Partition security is enforced in the edge port of the switch connected to the HFI. The security mechanisms ensure that a source injects packets only into partitions of which it is a member, and the packets are delivered only to endpoints in the same partition.

Partitions are created and managed by the Fabric Manager. It initializes the registers and tables used to enforce partition security and modifies them as partitions are redefined. Software running on the Host Nodes does not control the partition security registers and tables.

V. Layer 4 and Key Software Components

Multiple Layer 4 (Transport Layer) definitions can be encapsulated within the Layer 2 packets. These Transport Layers are optimized around common HPC and datacenter usage models, and are responsible for carrying the user facing semantics and providing the end-to-end portion of the reliability definition. All current Transport Layer definitions provide an invariant CRC (ICRC) on each packet that covers Layer 2 and Layer 4 header fields. On packet loss or corruption, an end-to-end protocol will retransmit the packets. Software layers are then provided over the Transport Layer to provide the network API to the user. Three key software layers include PSM, OFED Verbs, and the OpenFabrics Alliance (OFA) Open Fabrics Interface (OFI).

A. Performance Scaled Messaging

Performance Scaled Messaging (PSM) is a user-level library that provides a reliable fabric transport Application Programming Interface (API) for the Intel® Omni-Path HFI. The PSM API provides a matched queue (MQ) semantic that is a building block for MPI tag-matching send and receive calls. The PSM API has been extended for the Omni-Path architecture to provide 96 bits of tag matching to support up to 32 bits of user tag, up to 32 bits of source rank information and up to 32 bits of communicator context. This allows for significant scaling beyond the 64 bits of tag provided by the previous PSM implementation. The message-passing primitives provided by

PSM are point-to-point. The PSM implementation is designed to scale to the order of millions of MPI ranks.

Collective MPI operations can be synthesized from the point-to-point send and receive primitives using optimized algorithms that can be selected by parameters such as message size, collective communicator size and topology. Additionally, PSM provides an active messages (AM) API that can be used to implement arbitrary communication protocols using the active messages paradigm. This is used to implement a PGAS programming model using the OpenSHMEM API running over a GasNET conduit.

B. OFED Verbs and Compatibility

The Intel® Omni-Path HFI supports a fully compliant Verbs implementation. This includes support for UD, UC and RC queue pairs. Shared receive queues (SRQ) are also supported. The standard user-level and kernel-level Verbs library interfaces are provided. All standard Verbs management and performance protocols are supported. In addition to the 2KB and 4KB packet Maximum Transfer Unit (MTU) sizes supported in InfiniBand, the Intel OPA introduces an 8KB MTU which can be used by a Verbs implementation to reduce the required packet rate for large messages.

C. OFA Open Fabric Interface

The Open Fabrics Interface (OFI) [4] provides a general purpose software framework that is capable of handling a variety of fabric hardware and provides a standardized set of communication operations to higher code layers. The framework provides a library called *libfabric* that user-level applications can link to.

D. Leveraging Existing Ecosystems and Software

The host software stack integrates into the existing Open Fabrics Alliance software stack and into existing software distributions such as RHEL, SLES and OFED. The standard suite of fabric protocols and APIs are fully supported. This ensures that the investment in host software middleware and applications is preserved.

VI. First Generation Implementation

The first generation implementation of Intel OPA has been announced for introduction later this year. The first generation of products uses a physical layer that leverages industry standard link speeds for 100 Gigabit Ethernet and EDR InfiniBand (4 lanes of 25.78125) to yield 100 Gb/s (12.5 GB/s) of bandwidth available for Link Layer packets after the overhead required by the Link Transfer Layer for reliability.

A. Intel® Omni-Path HFI Product Architecture

The Intel® Omni-Path HFI is connected to the host system by a PCI Express (PCIe) 3.0 compliant port that supports a maximum link speed of 8GT/s and a link width of up to 16 lanes. This gives a maximum host bandwidth of 15.75GB/s in each direction after factoring out the 128b130b encoding. The HFI is connected to the fabric by one Intel® Omni-Path Architecture compliant fabric. The application-specific integrated circuit (ASIC) implementation provides one or two such HFIs in a single package. The ASIC with two HFIs has two logically independent HFIs with separate PCI-e ports, fabric ports, data paths, memory and control logic, and with shared ASIC level infrastructure for power, clock, ASIC reset, low-level control/management and some shared pins. The maximum available fabric bandwidth for two HFIs using both directions of the fabric link is 50GB/s. The ASIC can be used in a variety of product configurations including standard form factor PCIe cards, custom mezzanine card form factors, LAN on Motherboard (LOM) solutions, and integrated into the processor package.

The HFI partitions the layer 4 transport function between the HFI hardware and the host software stack. In addition to typical Link Layer offloads (e.g. packet formation and CRC generation), the hardware provides targeted offloads to accelerate typical per-packet transport layer capabilities and reduce software overhead. In turn, the software on-load component is responsible for higher level transport protocol functionality including maintaining connection state, mapping message-level transactions to packet sequences, and the end-to-end reliability protocol.

Host software can select between two send mechanisms provided by the hardware. Short messages use the Programmed Input/Output (PIO) Send mechanism to minimize latency. A large PIO send buffer is dynamically partitioned into contexts that are mapped for direct user mode application access. Sufficient contexts are provided to support the large core counts on the next generation of Intel® Xeon Phi™ processors. Each context provides a memory mapped FIFO that is used to generate a flow-controlled stream of packets to the link. Host software writes data directly into the send context to generate the packet on the fabric.

For longer messages, typically consisting of multiple packets, the Send Direct memory Access (SDMA) mechanism is used. Each SDMA engine processes an independent descriptor queue held in host memory, and host software appends descriptors to these queues. Taking the CPU core out of the data movement path allows the SDMA mechanism to achieve higher bandwidth and reduced CPU overhead. Additionally, there is an Automatic Header Generation (AHG) feature that allows hardware to generate headers for a stream of packets based on update information held in the descriptors to reduce the overhead of descriptor creation. Hardware arbitrates across all the packet sources, arbitrates for VL resources and link-level credits, performs packet header integrity checking, and finally egresses to the link.

The receive hardware provides a matching number of receive contexts to enable packets to be directly delivered to a user process. A variety of mapping algorithms are provided including the receive context number being specified directly in a packet header field, a lookup table based on the Queue Pair Number (QPN) packet field, and a programmable mapping based on hashes of arbitrary header fields called Receive Side Mapping (RSM). The receive side of the HFI provides eager receive and rendezvous receive capabilities for short and long messages, respectively. Host software chooses between the protocols for a given message. Eager provides lowest latency and highest message rate by delivering packets to FIFO ring buffers in host memory. Payloads are subsequently consumed by host software, typically involving a memory copy to the final

packet destination. Rendezvous is used for long messages, and achieves direct data placement to give the best bandwidth and reduced CPU utilization. The rendezvous protocol uses Token Identifier (TID) values so that destination memory can be pre-registered on the receive side, and then the send side generates packets with the allocated TID values to select the appropriate mapping for direct data placement. An additional mechanism allows for header sequence numbers to be checked in hardware to significantly reduce the number of Transport Layer packet headers that have to be delivered to software for processing. Packet arrival can be determined by polling on header status in host memory, or using a coalesced interrupt scheme.

B. Intel® Omni-Path HFI Software Implementation

1) Performance Scaled Messaging

In the PSM library, short message send and receive is implemented using direct access to the send context and receive context giving low latency and high message rate. PSM actively polls on the receive side for arriving packets to eliminate host interrupt overheads. PSM accesses the SDMA and expected receive protocol mechanisms using system calls into the host driver. This allows the driver to pin host memory pages for I/O DMA, and to register the physical pages into the HFI. A registration caching scheme is used to reduce the overheads on the receive side in the typical case where there is significant reuse of destination memory buffers at the MPI application level.

For each connection between a PSM sender and a PSM receiver, state is required to hold identification, status, sequencing and control information. This is termed connection state or flow state, and the size of this is denoted B (in bytes). The Intel® Omni-Path HFI architecture holds no connection state in the HFI. This means that there are no hardware capacity, caching or scaling limits as the size of the cluster increases. Instead all such connection state is held in the host and benefits from the host memory capacity, cache capacity and bandwidth available in the host memory system.

2) OFA Verbs

The implementation is partitioned between hardware capability in the HFI and host software. The hardware capabilities include the PIO send contexts, SDMA engines and AHG feature on the send side. Under driver control some portion of the PIO send contexts are set aside for driver use for Verbs protocols and these are typically used for small messages and for Verbs ACKs. Some or all of the SDMA engines are used for large message transfers for Verbs transactions. On the receive side the incoming Verbs protocol packets are spread across receive contexts and CPU cores using lower order bits from the QPN value and a mapping table. Interrupt coalescing is used to moderate the host interrupt rate without overly delaying incoming packets that are latency sensitive. These features allow the Verbs performance to scale as more CPU core resources are assigned to running the Verbs protocol code.

3) OFA OFI

The communication operations of *libfabric* are layered on top of the APIs mentioned in the previous sections. The message queue and tagged message queue operations are layered on PSM MQ, and collectives can be synthesized in terms of PSM MQ operations. Put, get, atomics and other operations designed to support PGAS or MPI3 RMA operations are layered on top of PSM AM implementation. Verbs is available through the standard Verbs access libraries.

C. Switch Product Architecture

The first implementation of an Intel OPA switch is a monolithic ASIC with 48 ports, each with 100 Gbps (nominal) links, including integrated long-reach serializer-deserializers (SERDES). Each ports supports 8 user VLs (plus one VL for fabric management). As shown in Fig. 3, the internal architecture is based on a bandwidth over-provisioned hierarchical crossbar, in which 4-ports comprise an MPort, with a local crossbar. The 12 Mports are connected via a central crossbar. Switching decisions are based on 48K (logical) entry unicast route table (URT), as well as a 16K multicast route table (MRT). URTs are replicated per Mport to provide adequate address lookup bandwidth. Arbitration logic supports packet pre-emption as well as per VL bandwidth metering. An on-chip microcontroller (MCU) translates high level management commands, transmitted via the Cport (port 0 of the switch) into low level hardware accesses required to manage on-chip resources. In addition, an external PCIe port is provided to allow an external CPU to directly manage the switch. An I2C interface provides access to off-chip electrically erasable programmable read only memory (EEPROM) and connectivity to baseboard management devices.

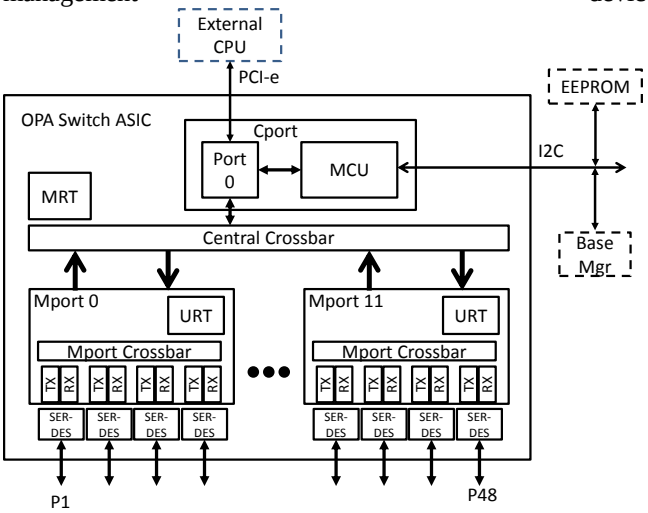

Figure 3:Intel OPA 48-Port Switch Block Diagram

D. Reliability Implementation

Data integrity and reliability is achieved at multiple levels. The fabric links are protected by link-level CRC with a link-level retry mechanism. The HFI and switch hardware paths are protected by error correcting code (ECC) and parity. End-to-end packet integrity is provided by a 32-bit invariant ICRC that is generated and validated in hardware. Packet sequence numbers (PSNs) are used to detect missing, duplicated and reordered packets. The standard 24-bit PSNs are expanded to 31 bits to give improved robustness for excessive delays in the fabric that could otherwise potentially lead to sequence number roll-over in extreme scenarios. Recovery mechanisms are implemented in host software using time-out and retry techniques.

E. Preliminary Performance Results

The HFI and switch implementations maximize the utilization of the high efficiency links through architecture improvements. First generation Intel OPA ASIC-level performance metrics relative to Intel True Scale are shown in Table I. The HFI takes full advantage of the host IO interface, achieving a factor of 4.6 improvement over Intel True Scale in small message rate performance. Switch message rates are also improved by a similar factor through optimizing internal pipelines to deliver maximum line rate for 64B packets. Switch latency under idle conditions is reduced by a significant 65ns (switch latency range indicates a second-order dependency on the choice of ports).

TABLE I. FIRST GENERATION INTEL OPA COMPARED TO INTEL TRUE SCALE

	Intel True Scale	**Intel OPA**
SERDES Rate (Gbps)	10	25.78
Peak Port Bandwidth (Gbps)	32	100
HFI Message Rate (Million Messages per second)	35	160
Switch Ports	36	48
Switch Packet Rate (Million Packets per Second)	42	195
Switch Latency (ns)	165-175	100-110

VII. RELATED WORK

The Intel OPA is directly related to the Cray Aries [2] and Intel True Scale [1] and other InfiniBand [6] architectures. In many ways, the Intel OPA is an evolution of architectural features leveraging the best practices. In other ways, Intel OPA introduces significant new functionality, including :

An extensible, open, high performance API framework in PSM, and more recently, OFA OFI that is amenable to creating a low-impedance interface between evolving HPC middleware (e.g. MPI), and the HFI hardware/software interface [9] (for example, in the first generation HFI, this has resulted in low latency, high bandwidth implementation which *on-loads* portions of the fabric protocol on processor CPU, memory, and cache resources. Intel® Xeon® processor and Intel® Xeon Phi™ processor products based on high numbers of cores, and high performance memory systems); Service Level, and Service Channel extensions to Virtual Lane abstraction which enable efficient allocation of HFI an switch resources; and a Link Transfer Protocol layer in the network stack which enables pre-empting of low priority packets by higher priority packets, enabling efficient bulk transfer using extended (8K) MTU sizes, while simultaneously reducing latency jitter seen in tightly coupled (e.g. MPI) communication.

VIII. CONCLUSIONS

The Intel Omni-Path Architecture (OPA) introduces a multi-generation fabric architecture designed to meet the scalability needs of datacenters ranging from the high-end of HPC to the breadth of commercial datacenters. Link level reliability and pervasive ECC enable the reliability needed for large scale systems. The QoS architecture is coupled with new packet preemption capabilities to enable both bandwidth fairness and low latency jitter for high priority packets. The first implementation silicon is available, and initial performance measurements are promising. Each link provides 100 Gb/s of bandwidth, and each HFI can achieve 160 million messages per second. Switch latency has been reduced to under 110 ns. These are substantial improvements relative to prior generation products while preserving the existing software ecosystem. In addition, a new community standard network API – OFA OFI - is in development that is designed to match user level semantic requirements to enable hardware innovation beneath the API.

REFERENCES

[1] Intel, "Intel True Scale Fabric Architecture: Enhanced HPC Architecture and Performance," Intel, Santa Clara, CA, USA, 2012.

[2] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins and J. Reinhard, "Cray cascade: a scalable HPC system based on a Dragonfly network," in *SC12: International Conference on High Performance Computing, Networking, Storage and Analysis*, Los Alimitos, CA, USA, 2012.

[3] L. A. Barroso, J. Clidara and U. Hozle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition, California, USA: Morgan and Claypool Publishers, 2013.

[4] P. Grun, S. Hefty, S. Sur, D. Goodell, R. D. Russell, H. Pritchard and J. M. Squyres, "A Brief Introduction to the OpenFabrics Interfaces: A New Network API for Maximizing High Performance Application Efficiency," in *IEEE Hot Interconnect 23*, Santa Clara, CA, USA, 2015.

[5] ISO/IEC, 7498-1: Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model, Geneva, Switzerland: ISO, 1996.

[6] InfiniBand Trade Association and others, Infiniband Architecture Specification: Release 1.0, InfiniBand Trade Association, 2000.

[7] IEEE, IEEE Std 802.3bj(TM)-2012; Amendment 2: Physical Layer Specifications and Management Parameters for 100 Gb/s Operation Over Backplanes and Copper Cables, New York, NY, USA: IEEE Computer Society, 2014.

[8] W. J. Dally, "Virtual-channel Flow Control," *Parallel and Distributes Systems, IEEE Transactions on*, vol. 3, no. 2, pp. 194-205, 1992.

[9] M. Luo, K. Seager, K. Murthy, C. Archer, S. Sur and S. Hefty, "Early Evaluation of Scalable Fabric Interface for PGAS Programming Models," in *PGAS 2014, 8th International Conference on Partitioned Global Address Space Programming Models*, Eugene, OR, USA, 2014.