Interconnect for Open High-Performance Computing

Barry Wood, Principal Product Applications Engineer

Interconnect for Open High-Performance Computing

Exascale computing is the pursuit of a machine capable of 1018 floating point operations (FLOPs) in a second. This is approximately the available compute capacity of the human brain. One of the major engineering problems that must be solved to realize exascale computing is power consumption. The average human brain uses 20 watts. The current best in class efficiency achieved in the November 2014 Green 500 list of supercomputers is 5.27 gigaflops per watt. Assuming the power associated with this architecture would scale linearly, an exascale computer with this efficiency would consume almost 190 MW. That's enough power to run 130,000 homes or 9.5 million human brains!

A significant portion of the power consumed by an exascale computer must be spent on inter-chip interconnect. The millions of processors in an exascale computer are just so many "coffee warmers" without the ability to communicate with each other and the outside world. Communication between processors on the same chip takes little power (microwatts) compared to the power required to drive signals externally (hundreds of milliwatts to watts). An ideal interconnect maximizes the FLOPs spent doing useful application work, and minimizes the number spent waiting for data or performing processor communication.

"Interconnect efficiency" has a specific meaning in this whitepaper. Interconnect X is more efficient than Interconnect Y when the power consumed by Interconnect X to transfer the right bits at the right time from one memory system into another is less than the power consumed by Interconnect Y. Interconnect efficiency is therefore not a measure of any single communication operation, but an end-to-end system measurement. Efficiency is most affected by the time taken to perform a transfer ("the right bits at the right time"), since an entire system may be burning power waiting for one transfer to complete.

Interconnect efficiency is important not just for exascale computing, but for every application large enough to require multiple processors. This includes many examples of "Big Data" analytics, as well as the social media and search applications we depend on as a society. The amount of power used to run and cool a server for a year can equal the cost of the server.

This white paper proposes some attributes of an “ideal” efficient interconnect for open high-performance computing, and compares some existing interconnect technologies against those criteria.

Attributes of an Ideal Interconnect

Table 1: Attributes of an Ideal Interconnect

Attribute		Description
Interoperability		Multiple vendors, open standard demonstrated interoperability
Integration		Interface integrated into SoC
Transaction Types:	Cache Coherency	Extension of cache coherent SoC bus
	Reads/Writes	Supports non-cache coherent SoC bus extension
	Messaging	Connection and non-connection oriented messaging
	RDMA	Read/write data for another SoC software entities memory space
Any topology		SoCs can be connected in any manner, multicast/broadcast support
Goodput		Amount of data delivered matches raw bandwidth, many raw bandwidth options, 100 Gbps dense links available
Fault tolerance		Supports systems design with no single point of failure
Software		Enables application to leverage all of the innovations in the ideal interconnect

Interoperability

The first attribute of the ideal interconnect is interoperability. The ideal interconnect enables heterogeneous systems design. It should be possible to use processing and interconnect devices from multiple vendors in a system to optimize the performance of that system. For example, a well-engineered acceleration engine that is purpose built for a function should be able to outperform a software implementation of that function, regardless of the capacity of the processor running that function. Each vendor’s devices are optimized for different applications, with different acceleration engines such as digital signal processor cores, graphics processors, encryption/decryption/authentication engines, pattern matching search engines, programmable logic, and numerous options for file system acceleration. Interconnect vendors provide continual improvements in power, throughput, and latency. As different vendors improve their devices, an interoperable interconnect allows the new devices to be integrated easily and the overall system to evolve. Interoperability drives processing and interconnect innovation.

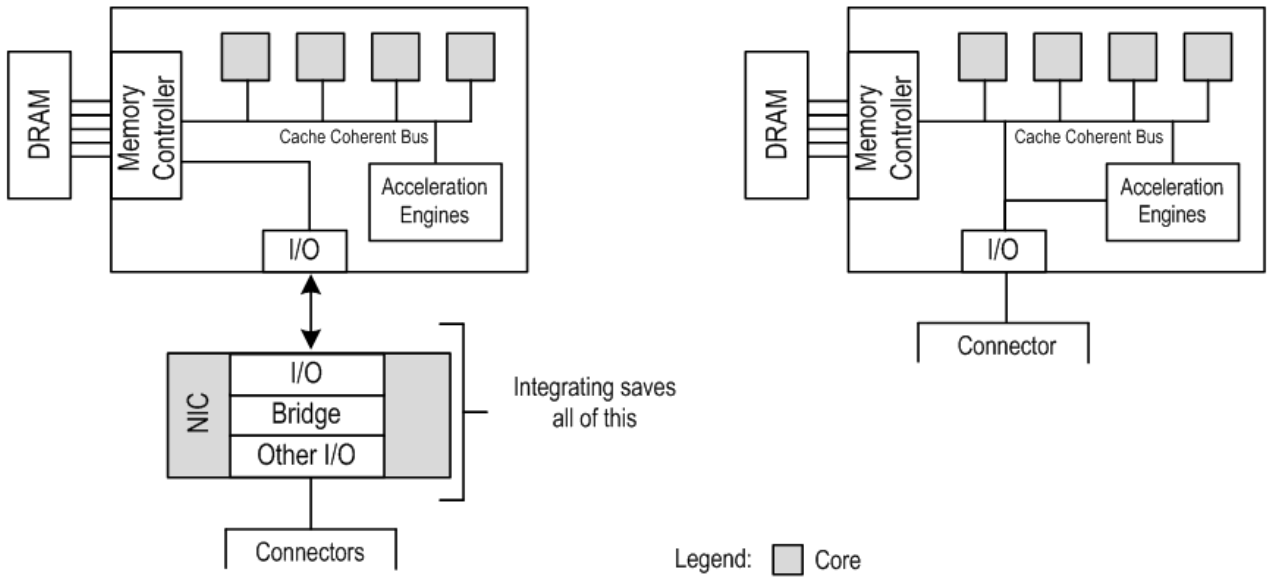
Integration

The second attribute of the ideal interconnect is integration. As shown on the left side of Figure 1, an external bridge that translates from one input/output protocol to an input/output protocol native to the chip requires an additional external component, and power for two more interfaces, as compared to the integrated I/O solution shown on the right side. The ideal interconnect should be integrated into the bus structure and memory hierarchy of every computing device chosen, since integration minimizes the power required to transfer data back and forth between the memory hierarchy and the outside world.

Integration provides more value than just reducing the number of components on a board. As shown on the right side of Figure 1, the ideal integrated interconnect would be integrated into the entire system-on-chip (SoC) design, ensuring efficient, low latency interactions with acceleration engines. The SoC design would optimize memory controller operation to balance CPU performance with interconnect throughput and latency.

The SoC design for the ideal interconnect would use cache coherent buses inside the device to provide two interconnect optimizations. The ideal interconnect would deliver selected data directly into a processor cache to minimize latency and the amount of power required to process that data. Similarly, cache coherency allows data to be transmitted directly from a processor cache. In contrast, SoC design for an external I/O is not optimized for that I/O, and so all communication is through the DRAM. This design increases the amount of memory controller/DRAM power used for communication, and increases the latency of communication operations.

Figure 1: External NICs versus Integrated Interconnect

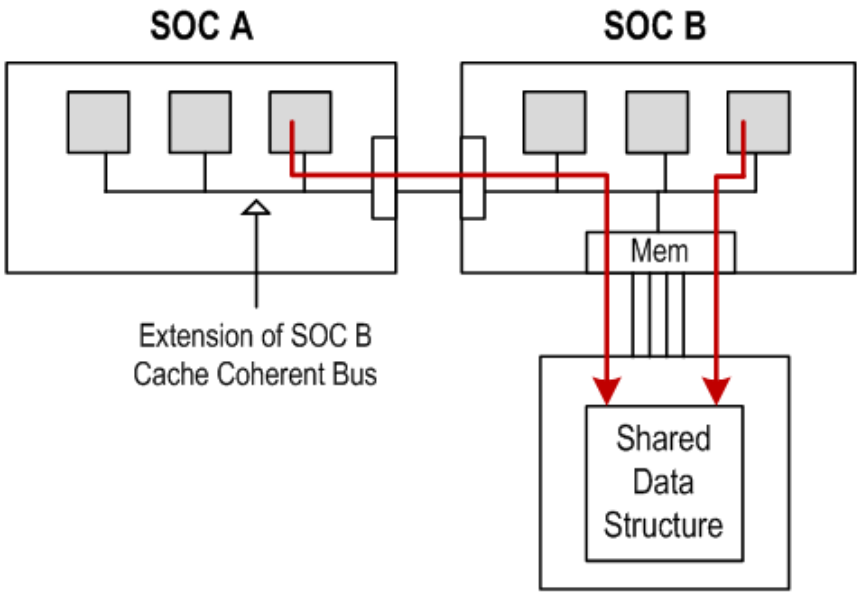


Transaction Types

Cache Coherency

The ideal interconnect would support a cache coherency protocol for memory regions that are shared between devices, simplifying the programming model for access to shared memory data structures (see Figure 2) and enabling efficient inter-process communication across multiple System on Chip (SOC). Cache coherent bus extensions allow the ratio of compute power to interconnect power to change based on the current application. For example, performance of file system applications are almost by definition limited by the interconnect. In contrast, searching for prime numbers is limited by the number of processors searching, not the speed of communicating the relatively small amounts of prime numbers found. Cache coherency thus allows the size and composition of nodes in a system to change to achieve the optimal balance of I/O resources and compute resources for the currently executing application. Balancing compute and I/O resources ensures that all energy used for compute and I/O operations is expended doing useful application work.

Figure 2: Cache Coherent Heterogeneous Multiprocessing

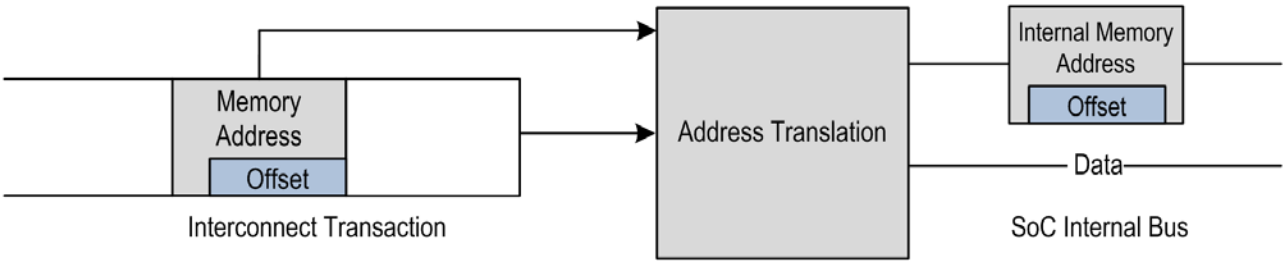


Read/Write Bus Extension

Unfortunately, the amount of bandwidth required for cache coherency expands exponentially as the system size grows, so a cache coherent exascale computer is just not feasible. The ideal interconnect would support non-cache coherent extension of internal SoC buses to efficiently scale to larger systems.

Typically, internal SoC buses support read and write transactions, and may support atomic transactions or other more complex operations for mutual exclusion and communication between software entities. As shown in Figure 3 the ideal interconnect can extend a SoC bus efficiently. The ideal interconnect should guarantee that the read and write transactions always get to the target memories in a predictable order that preserves the programming model of the SoC bus. Additionally, the ideal interconnect should provide a mechanism to verify the control information (that is, memory address, destination, transaction type) in the transaction before all of the data has arrived, to reduce latency. That way, delays in processing are limited to ensuring that the data associated with each received I/O transaction is correct before processing the write, read response, or more complex operation on the internal bus and memory controller.

Figure 3: Read/Write Packets Mapping Directly to Internal Bus

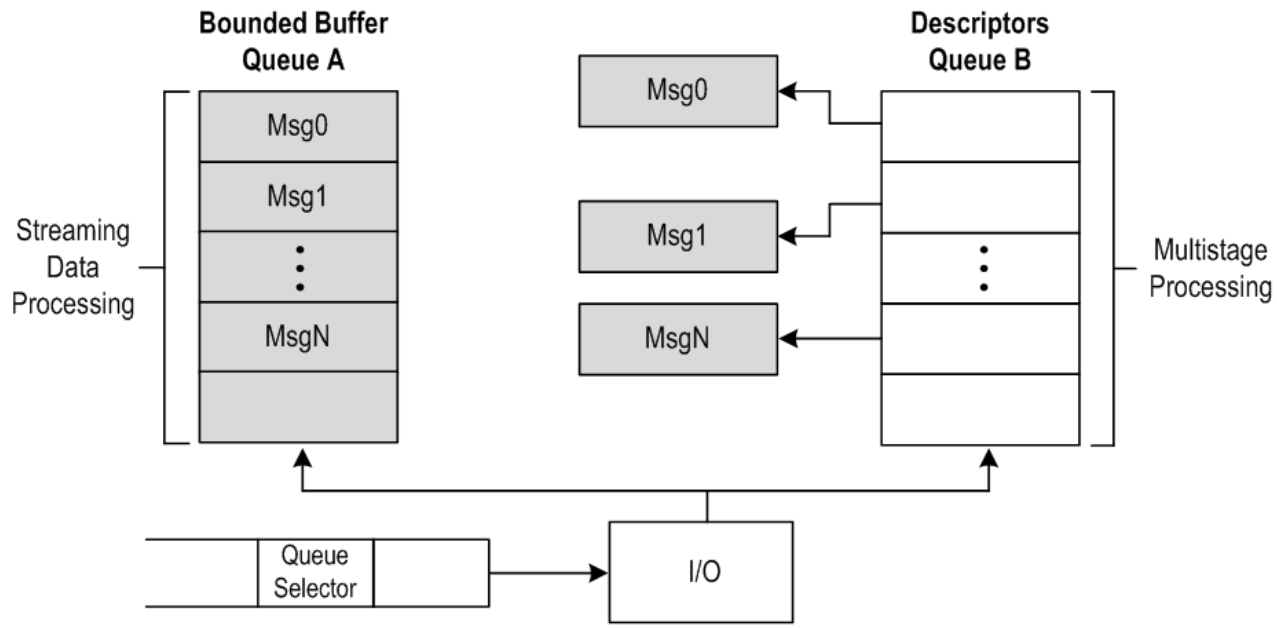


Messaging

Many systems have topologies that are dynamic. As a result, the application must discover where to read and write in order to complete these transactions safely and effectively. However, if the only tool available for communication with a node are reads and writes, learning where to read and write on that node becomes a “chicken-and-egg” problem. Messaging semantics were introduced to avoid this paradox. Messages do not depend on a memory address to be correctly processed by the receiver. Instead, it is up to the receiver to place the message contents and associated metadata into the receiver’s choice of memory location. For larger transfers and/or long lived connection oriented transfers, messages can be used to coordinate the memory addresses used by each end of the transfer. For short lived connections, small transfers, or connectionless transfers, message semantics are optimal.

As with reads and writes, messaging communication is most efficient when the ideal interconnect guarantees in-order delivery of those messages. Additionally, the ideal interconnect supports connection oriented and connectionless messaging. It should be possible, but not mandatory, for a received message to be routed to one of many different process or application specific queues, and for the contents of those queues to be directly accessible by software.

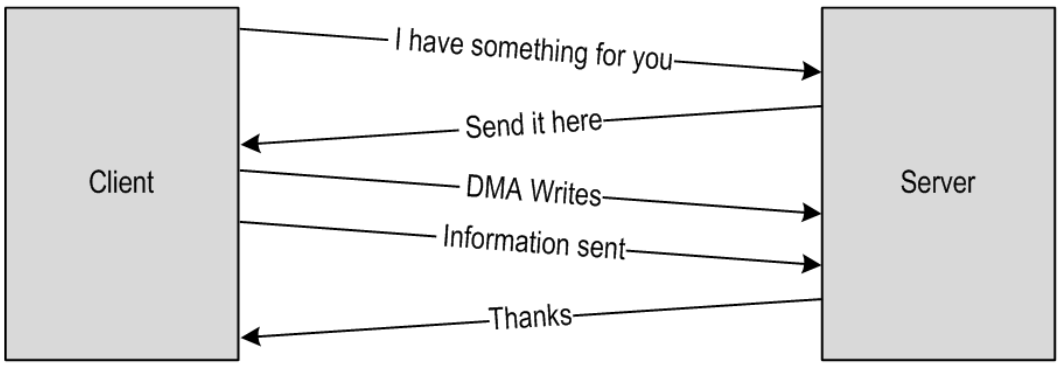
Figure 4: Descriptor-Oriented Messaging Queues Specific to a Process



RDMA

The ideal interconnect combination of read/write and messaging semantics enables a third capability: Remote Direct Memory Access (RDMA). RDMA protocols use messages to coordinate the use of memory regions that can be remotely accessed with reads and writes (see Figure 5). RDMA technology has the potential to deliver the lowest power information transfer between software entities running on different processors, since as shown in Figure 3, the reads and writes map directly to the internal bus semantics of devices. DMA and messaging engines integrated with the internal bus and the ideal interconnect offload the burden of data transfer from the processor, preserving as many CPU cycles as possible for real application work.

Figure 5: Messages Used to Set up RDMA Region, Reads/Writes, and Polling



Any Topology

The optimal configuration for large systems is different for different applications. An ideal interconnect must have a routing scheme that supports any topology, and is consistent for cache coherency, read/write, and messaging. The implication is that transaction routing must be independent of the memory address used for cache coherency, read and write transactions, since messaging does not have a memory address. Separating routing from memory address allows the amount of interconnect memory space supported by different types of nodes to vary widely without impacting the efficiency and extensibility of the routing scheme. The routing scheme should efficiently support systems with tens of nodes up to millions of nodes. In addition, the ideal interconnect routing scheme would support broadcast, multicast, and multiple routes to the same destination, all with minimal latency and low power.

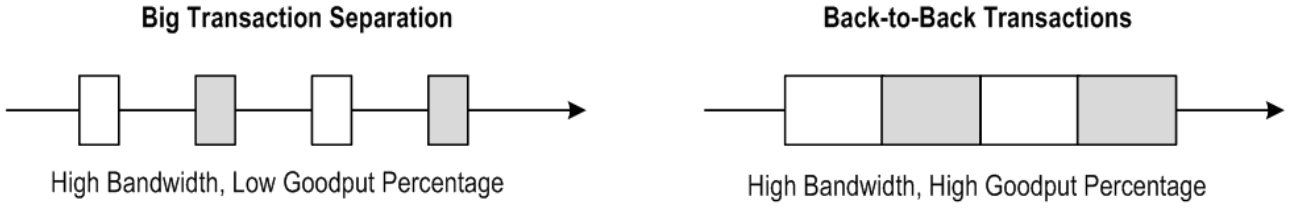
Goodput

As seen from the discussion of cache coherency transaction support, efficiency can decline as the size of a system increases, since communication requirements may expand exponentially. One way to minimize system size, and by implication total power consumed, is to increase the goodput of each element of the system.

On a processor, increasing the internal clock speeds and memory interface speed may result in more operations being performed in a given unit of time. The increase in goodput is proportional to the increase in power. Of course, if the processor is doing is waiting for memory accesses faster than ever before, more power is being spent to produce the same goodput. Processor power management is increasingly sophisticated, automatically changing internal clock speeds to deliver optimal amounts of goodput and power consumption.

As shown in Figure 6, increasing an interconnects' raw bandwidth requires more power but does not guarantee a corresponding increase in goodput. For an ideal interconnect, goodput increases in proportion to the raw bandwidth. An ideal interconnect should allow multiple raw bandwidth options so that interconnect power can be optimized for compute-constrained and I/O-constrained applications.

Figure 6: Contrasting High Raw Bandwidth and High Goodput



In an ideal interconnect, the amount of raw bandwidth can be optimized for each direction of communication. Depending on topology, the amount of goodput required in one direction of a link can be radically different than what is needed for the other direction. For example, depending on the application, a file server can send much more data than it receives. The asymmetric links provided by an ideal interconnect are optimal for these applications.

Fault Tolerance

The risk of a fault occurring is directly proportional to the size of the system. A system is least efficient when it is consuming power but is not capable of doing any useful work due to a fault. An ideal interconnect therefore supports fault tolerance.

Fault tolerance starts with limiting the impact of transient errors through immediate detection and transparent recovery. The ideal interconnect supports rapid detection, isolation, notification, diagnosis, and recovery from link conditions where it is no longer possible to exchange transactions. The ideal interconnect should support various redundancy strategies, including no redundancy, cold/warm/hot sparing, N+M redundancy, and parallel/redundant execution.

Software

The ideal interconnect has a software ecosystem that supports all of the attributes of the ideal interconnect. The software must be interoperable among multiple different device vendors. It must support utilization of integrated on-chip cache coherency and acceleration engines. It should support off-chip cache coherency, read/write, and messaging operations, ideally integrating these into efficient RDMA support. It should support different system topologies, goodput optimization, and fault tolerance.

Applications most concerned about optimal performance could use the ideal interconnect software directly. However, to support migration from other interconnects, software for an ideal interconnect must be integrated into common software platforms. For example, many high-performance computing applications are built on software platforms, such as OpenMPI, that support multiple different interconnects. Similarly, various flavors of Hadoop are used in Big Data applications.

Interconnect Evaluation Summary

Table 2 shows ideal interconnect attributes for RapidIO®, Infiniband™, Ethernet, PCIe®, and Proprietary evaluations.

Table 2: Summary of Ideal System Interconnect Attributes

Attribute		RapidIO	Infiniband	Ethernet	PCIe	Proprietary
Interoperability		✓	–	✓	✓	–
Integration		✓	✓	✓	✓	✓
Transaction Types	Cache/Coherency	✓	–	–	–	?
	Read/Write Bus Extension	✓	–	–	✓	?
	Messaging	✓	✓	✓	–	?
	RDMA	✓	✓	✓	✓	?
Any Topology		✓	✓	✓	–	?
Goodput		–	✓	✓	–	✓
Fault Tolerance		✓	✓	✓	–	?
Software		–	✓	✓	✓	?

PCI Express

For years, one of the mainstays for connecting devices has been PCIe and its ancestors. The freely available PCI Express® specifications, and the PCI Special Interest Group's commitment to interoperability, have historically enabled a large ecosystem of devices and software. By definition, PCIe is integrated into components, which should lead to power efficient interconnect solutions.

PCIe and its ancestors strength is its support for read/write semantics. Recently, PCIe added atomic transactions for mutual exclusion. The flow control mechanisms for PCIe allow high goodput for available bandwidth. PCIe can support connection oriented communication, with or without the use of DMA engines to offload the processor.

The original purpose of PCIe and its antecedents was to connect very smart processors with very dumb peripheral devices, not to connect intelligent processors together. There was no need for messaging support in this usage model. The current lack of messaging semantics makes it difficult to implement connectionless communication between intelligent processors over PCIe.

It is still possible to support RDMA semantics using PCIe, but the scale of RDMA may be limited unless a secondary interconnect that supports messaging is available.

The lack of messaging semantics is indicative of the general inability to scale systems using PCIe. Standard PCIe devices are part of a global memory map that uses a tree routing topology. There are proprietary extensions, such as non-transparent bridging (NTB), that allow PCIe devices to be used in topologies other than a tree. There are still limitations to the types of topologies that can be supported. For example, it is difficult to imagine using PCIe in a hypercube or 3D toroidal topology. Similarly, it is not possible to use PCIe for cache coherency between processors. PCIe does support multicast for writes, which makes distribution of data to multiple processors faster.

Historically, PCIe and its ancestors were not fault tolerant nor were they expected to be. If a key peripheral did not respond to a transaction, a system reset with accompanying “blue screen” notification was a reasonable outcome. System reset is less reasonable as the number of components in the system grows to the thousands or millions, and is especially heinous when those components are not peripherals but other intelligent processors. PCIe has added fault tolerance capabilities to the base specification to the point where boards can be removed and inserted into a PCIe system without prior notice to the operating software, and without crashing the system. However, these mechanisms do not translate automatically to NTB operation, which means they may or may not work on systems at scale. Additional proprietary fault tolerance solutions may be necessary.

PCIe also does not offer the best possible goodput, although it does support excellent goodput for each raw bandwidth option. While PCIe links do support 8 Gbps lanes, and up to 16 lane wide ports, these connections are possible only between devices on a board or using short cables between boards. The PCIe 4.0 roadmap does not offer the dense,100 Gbps links between boards and chassis that are necessary to achieve exascale systems.

The opportunities for PCI Express switch vendors have been falling as the number of PCI Express ports integrated into processor chip sets has increased. Directly connecting peripherals, such as disk drives, to the processor eliminates the cost of an aggregation switch. Of course, there is also no need or requirement to support switching between ports on a processor chip set, eliminating the possibility of peer-to-peer scalability between peripherals. The inability to scale the size of nodes using cache coherency, and the limitations on topology and the number of nodes caused by lack of connectionless messaging and open fault tolerance, make PCIe a risky choice when contemplating high performance compute in large systems. A forklift system upgrade to a different interconnect may be necessary in future.

Ethernet

It is hard to imagine an interconnect with a larger ecosystem than Ethernet. Ethernet is the basis for much of the Internet; the largest interconnect implementation on the planet. The specifications for Ethernet are completely open, and anyone can participate and contribute.

The large number of Ethernet users requires substantial amounts of goodput as groups of users are aggregated onto faster and faster links. Ethernet networks have traditionally increased their top-line bandwidth by a factor of 10 every five years. This has allowed the interconnect to grow from the 10 and 100 megabit-per-second technology of the 1980’s and 1990’s, to the 100 and 400 Gbps technology that is used in Internet networks today.

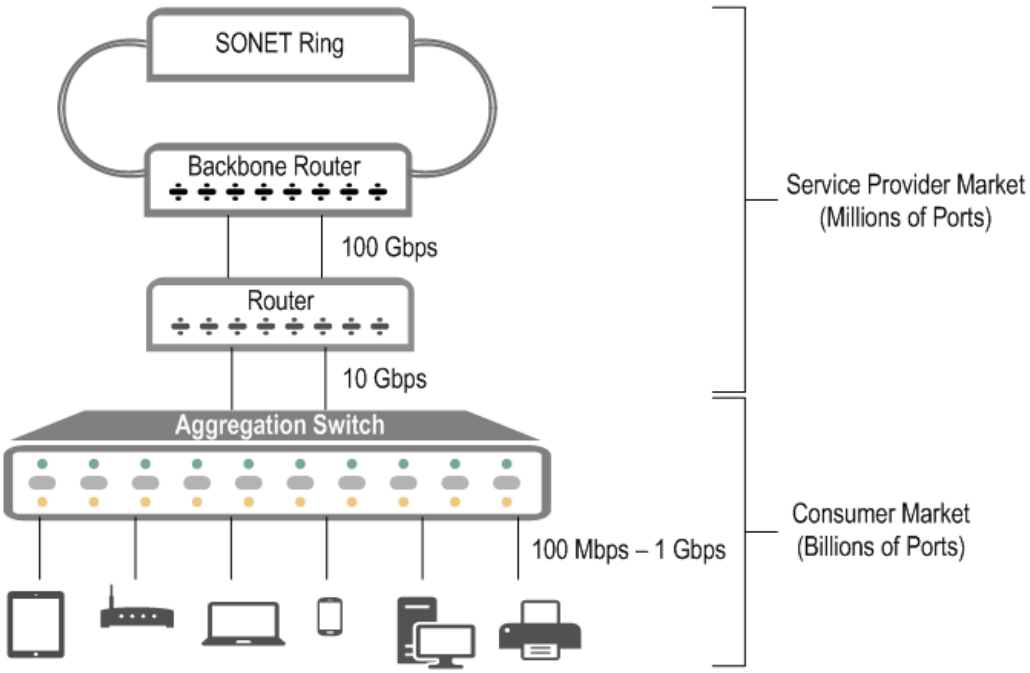
While Ethernet connectivity traditionally has been implemented in a peripheral device (a NIC) over PCIe, it has been integrated on a large number of devices and so can allow a system designer to mix and match. The Ethernet software ecosystem makes Ethernet easy to use. Ethernet connectivity is ubiquitous to the point where it is synonymous with networking. Messaging capabilities are the strength of Ethernet networking. Ethernet software and hardware have evolved to provide highly reliable, fault tolerant messaging capabilities over arbitrary, dynamic interconnect topologies. Ethernet supports a wide variety of connection-oriented and connectionless protocols,

multicast/broadcast, and flow control schemes. As such, it can be used to implement RDMA, for example with iWARP and RoCE.

Mass market Ethernet technology has been optimized for relatively low bandwidth, high latency connections and networks. This basic assumption makes Ethernet unsuitable for high-performance compute applications that require high bandwidth, low latency connections. As various cloud-based applications have become more pervasive, the market for high bandwidth, low latency Ethernet has also increased. However, the centralization and sharing enabled by the cloud computing model ensures that the size of the high-bandwidth Ethernet market is much smaller than that of mass market Ethernet, as shown in Figure 7.

Ethernet networks do not guarantee delivery of transactions, nor are those transactions guaranteed to arrive in order. For example, the flow control scheme of the widely used TCP protocol, on which iWARP is based, depends on packet loss to operate correctly. Packet loss may be accidental or it may be intentionally triggered to manage congestion. The basic assumption that packet loss will occur makes it impossible to efficiently implement read/write semantics over Ethernet networks. It also makes efficient cache coherency over Ethernet impossible.

Figure 7: Ethernet Architecture



Another impact of Ethernet technologies need to discard packets is that goodput is much less than the amount of raw bandwidth available. In some cases, this is a direct implication of the difficulty of terminating an Ethernet link in hardware or software. It makes no sense to deliver 100 Gbps of bandwidth to a processor that cannot handle that many Ethernet frames. The Ethernet flow control mechanisms, which were designed for long distance communication, are slow in the context of today's link speeds. The round trip end-to-end timeouts which detect packet discard, and trigger flow control, must be set high to support the wide variety and uncertainty of switch scheduling

behaviors and traffic conditions. The SDN and OpenFlow initiatives are designed to overcome this limitation by centralizing bandwidth management and packet routing decisions to ensure the required bandwidth is available for each connection, at the cost of additional software/hardware complexity and potentially additional latency.

It is an interesting question if such centralized real-time coordination is feasible for exascale computing. Eliminating the short-term congestion that leads to packet loss in Ethernet networks is a difficult computing problem. Eliminating the transmission errors through such mechanisms as Forward Error Correction increases latency. Both factors place a significant drag on the efficiency of Ethernet networks for high performance computing.

Infiniband

As of mid 2014, Infiniband technology is the interconnect of almost half of the world's Top 500 supercomputers, as ranked by size. Infiniband has many strengths, including freely available software support developed by the Open Fabrics Alliance (OFA).

The Open Fabrics Enterprise Distribution (OFED) software stack supports Infiniband, as well as iWARP and RDMA over Converged Ethernet (RoCE). Infiniband has industry leading latency, as shown in many benchmarks. Many independent comparisons of Infiniband with other technologies have shown that Infiniband has superior latency, and superior performance for a number of different applications.

Infiniband is a switched technology with routing independent of memory address and payload content, which allows it to support any network topology. The Infiniband programming model supports messaging and read/write semantics, which like Ethernet, allows Infiniband to support RDMA capability. Indeed, the Infiniband "verbs" based programming model has become synonymous with RDMA.

Infiniband has link-level flow control that ensures packets are not dropped due to short-term congestion. The Infiniband system model includes centralized control of connection management, including bandwidth allocation. Infiniband also has excellent top-line bandwidth options (14 Gbps). Infiniband systems can efficiently use available raw bandwidth to deliver high goodput.

However, there are some drawbacks to Infiniband technology. Infiniband technology to date has relied on external NIC devices connecting to processors using PCIe. As argued earlier, this makes Infiniband significantly less power efficient than integrated interconnect technologies, and also increases the design challenges for reducing latency. Since PCIe does not support cache coherency semantics, it is not possible to scale the size of nodes using Infiniband.

Infiniband recovers from transmission errors using the same high latency mechanisms as Ethernet. An important implication is that packets may not be delivered in order. The follow-on conclusion is that it is not possible to immediately map received Infiniband transactions to an internal bus, which increases Infiniband latency and the amount of power required to process Infiniband transactions. Infiniband is unsuitable for bus extension support.

Infiniband does have interoperability test facilities at the University of New Hampshire Interoperability Labs (UNH IOL) that support the two current vendors of Infiniband devices. However, future development of Infiniband technology is vigorously supported by only one silicon supplier, Mellanox®. Mellanox is responsible for both switch and NIC silicon, as well as some of the products that use this silicon. There are no public specifications of Infiniband technology after the first release in 2007, on which this article is based. While the private story may differ (see Mellanox's alliance with IBM® for their high-end servers), Infiniband has many attributes of a proprietary interconnect technology.

RapidIO

RapidIO is used widely within wireless base stations. If you use a 4G cell phone, the base station uses RapidIO. RapidIO also has significant design wins in high-performance military compute, industrial control, video, and other high performance compute applications. RapidIO is optimized for systems that have size, weight, and power constraints. For example, NASA selected RapidIO as the Next Generation Space Interconnect Standard (NGSIS) for satellite applications, which of course present extreme size, weight, and power constraints.

RapidIO has many attributes of an open, interoperable technology. All versions of the RapidIO specification are publicly available. The RapidIO specification has evolved in a way that allows devices created based on earlier versions of the specification to be compliant to later versions of the specification. Interoperability is further supported by RapidIO.org's Bus Functional Model (BFM). The BFM is the practical yardstick used by all RapidIO device and IP developers to measure compliance to the RapidIO specification.

RapidIO is integrated into devices from many processor vendors, including Freescale®, Texas Instruments, Broadcom®, LSI/Intel®, and Cavium. It is also supported by major FPGA vendors such as Xilinx®, Altera®, and Lattice®. Other PCIe based processors are supported by an external PCIe-to-RapidIO bridge device. Major processor vendors such as ARM®, AMD™, and Intel (through acquisition), have recently joined RapidIO.org, so the number of processor vendors with integrated RapidIO ports may expand in future.

Since the inception of RapidIO.org, the RapidIO specification has been optimized for low latency, high bandwidth, and power efficient communication. RapidIO packets are guaranteed to be delivered in the order of origination through link-level flow control and error recovery mechanisms. RapidIO technology is thus well suited for cache coherency and read/write bus extensions: the RapidIO specification has supported cache coherency and read/write semantics since 2002. A RapidIO.org task group is now working to standardize mapping the cache coherency protocol to ARM's internal interconnect technologies such as AMBA and ACE. The RapidIO protocol and RapidIO devices also support messaging semantics. RapidIO systems commonly use RDMA-style communication for low latency, energy efficient data transfer.

RapidIO packets must be routed according to a quantity known in RapidIO vernacular as a "device ID." The register programming model for packet routing is standardized in the RapidIO specification. RapidIO supports multicast and broadcast routing in any topology. As noted earlier, this allows

RapidIO devices to support independent memory maps, simplifying addressing of cache coherency and read/write semantics.

Motivated by requirements from the telecom industry, the RapidIO ecosystem supports perpetual back-to-back packet transfers, enabling high goodput for all kinds of data transfers. Numerous top line bandwidth options, and dynamic and permanent asymmetric link support, allow raw bandwidth/power/goodput to closely match application requirements.

In addition, RapidIO supports fault tolerant system design. Transient errors, such as packet CRC errors, are corrected through a standard link error recovery mechanism. Conditions that prevent packet transfer on a link are detected in nanoseconds to microseconds, triggering hardware isolation and notification mechanisms. Fault diagnosis and recovery is supported by a standard register programming interface. The RapidIO 3.1 specification, released in November 2014, meets space industry fault recovery requirements for satellite systems that have a mean time to repair of "never." Note that, depending on the operator, terrestrial containerized high-performance compute implementations can have mean time to repair requirements that are similar to satellite systems.

Like all of the previously analyzed interconnect options, RapidIO does have some weaknesses. Since RapidIO development has followed the cadence of base station design, new devices only appear approximately every 5 years. RapidIO is currently behind other interconnects in terms of top-line bandwidth, and hence the maximum goodput that can be used in systems. The appearance of new RapidIO devices has been foreshadowed by the 2014 announcements by IDT and Mobiveil.of IP compliant to the 3.0 and 3.1 specifications.

Software is also a weakness of the RapidIO ecosystem. Most RapidIO users view their software infrastructure as a competitive advantage, and so have developed their own private system management and high performance RDMA implementations. The Linux kernel has included RapidIO support for a number of years. However, this support is limited to RapidIO system initialization and high latency, low performance messaging support. The RapidIO.org Software Task Group is working to correct this deficiency, with public contributions to the Linux kernel expected to begin in March 2015.

Lastly, like other interconnect solutions, the number of RapidIO switch vendors has dwindled over the years. IDT is now the major switch vendor in the RapidIO ecosystem. Unlike other interconnect options, though, having a single switch vendor does not make RapidIO a proprietary solution, since the RapidIO ecosystem is supported by multiple processor and FPGA vendors and based on an open specification that provides all of the facilities necessary to support interoperable exascale computing.

Most members of the RapidIO ecosystem are dedicated to open source development. Linux support for RapidIO includes complete drivers for IDT switches. Other RapidIO device vendors provide open source drivers for their devices. RapidIO is pursuing for the next step in the "open" movement: open silicon. With the support of the RapidIO.org, IDT is contributing the RapidIO specification to the Open Compute Projects (OCP) High Performance Compute group as the first step in open interconnect for high-performance computing devices.

Proprietary Solutions

There are some aspects of each of the evaluated technologies that are proprietary. The key technology difference between a truly proprietary technology and the others listed in Table 2 is the availability of public specifications for interconnect interoperability. The public specifications allow the creation of devices that can interoperate with others in the ecosystem, which in turn stimulates competition and innovation. Without open specifications there is a high barrier to market entry for competition. There is also a high barrier to market exit for customers, since changing interconnect technologies requires wholesale porting of infrastructure and applications.

Integrated proprietary solutions can provide excellent power efficiency and top line bandwidth. What is unclear is the other capabilities, and the quality of those capabilities. Even if some capabilities are present in one generation of devices they may be absent in others. Evaluation of proprietary technology is problematic since accurate comparisons may involve porting applications to proprietary interfaces. This can be a significant effort. Without such an evaluation, the majority of information available is marketing hype.

The termination of Moore's Law is within sight, implying that the semiconductor industry is now exploring the limits to the size and number of processors that can fit into a single device. As previously noted, there are already many classes of problem that cannot be solved with single processors. The implication is that high performance computing improvements will be driven by the creation of acceleration engines, software, and interconnect technologies. Proprietary solutions cannot maintain the pace of innovation of an open ecosystem.

Conclusions

With the demise of Moore's Law, it is no longer possible to shrink silicon problems to a more manageable size by using a smaller geometry process technology. The increasing rate of data generation, and the huge variety of Big Data analytics applications, can be supported only by specialized innovations by many different companies. Connecting these innovations to realize the dream of exascale computing can only be accomplished by an ideal, open interconnect technology.

However, that interconnect technology does not exist today. Proprietary solutions cannot innovate fast enough or broadly enough to cover all applications. PCIe and Infiniband have significant limitations that affect efficiency and scalability. The basic assumptions of Ethernet technology make it completely unsuitable for high performance, low latency, energy efficient computing. The RapidIO ecosystem is too slow, and software support too limited, in comparison to the competition.

However, of all the competitors, RapidIO has the best chance of becoming that ideal, open interconnect. The RapidIO specification is open, has silicon proven interoperability, and addresses key issues for small, medium, and large scale energy efficient, performance critical computing. RapidIO has a large interoperable ecosystem with broad industry support. Major processor vendors are driving the RapidIO specification forward using a proven model for collaboration and innovation. Lastly, RapidIO has a clear path to overcome the short-term limitations on top line bandwidth and software support.

There's a well publicized quote from Henry Ford that is appropriate to the subject of exascale: "If I had asked people what they wanted, they would have said faster horses." Ford's solution, the automobile, was faster than the horse. What is less commonly recognized is that the automobile solved many other horse-related transportation problems, not the least of which was manure removal from densely populated cities. Moore's Law has enabled high-performance system developers to ask for faster horses for a long time. Just asking for an exascale horse does not solve the real problems of exascale computing. We have to use our brains.

www.IDT.com

Corporate Headquarters
6024 Silver Creek Valley Road
San Jose, CA 95138 USA

Sales
1-800-345-7015 or
408-284-8200
Fax: 408-284-2775
www.idt.com

Tech Support
email: srio@idt.com