Cray XC® Series Network

Bob Alverson, Edwin Froese, Larry Kaplan and Duncan Roweth
Cray Inc.

Table of Contents

Introduction 3
Cost-Effective, High-Bandwidth Networks 4
Aries 7
Aries NIC 8
- Address Translation 9
- Remote Memory Access 10
- Ordering and Completion 11
- Reductions and Barriers 12
- System Functionality 13

Aries Router 13
Dragonfly Implementation for the Cray XC Series 14
Cray XC Routing 16
Network Reliability, Availability and Serviceability 17
Cray XC Network Software Stack 19
Aries Performance 21
Cray XC Series Configuration 25
Conclusion 27
References 28
Acknowledgements 28

Introduction

The Cray XC series is a distributed memory system developed as part of Cray's participation in the Defense Advanced Research Projects Agency's (DARPA) High Productivity Computing System (HPCS) program. Previously codenamed "Cascade," the Cray XC system is capable of sustained multi-petaflops performance and features a hybrid architecture combining multiple processor technologies, a high performance network and a high performance operating system and programming environment.

Specifically, the Cray XC network provides programmers with global access to all of the memory of their parallel application. The Cray-developed Aries™[1] interconnect provides substantial improvements on standard network performance metrics for high performance computing (HPC): bandwidth, latency, message rate and scalability. In addition, the Aries design addresses a key challenge in high performance networking — how to provide cost-effective, scalable global bandwidth. The network's unique ability to efficiently execute complex global communication patterns broadens the range of applications that perform well on the Cray XC system.

The Cray XC network design provides a wide range of HPC-optimized system attributes:

- Systems can be configured to match bandwidth requirements and cost constraints by varying the number of optical connections
- Address space of general purpose processors is extended to efficiently access all the physical memory in the system
- Suite of communication and synchronization mechanisms: block transfer engines for asynchronous pipelined memory copies, atomic memory operations, barriers, source and destination-side synchronization, message delivery doorbells and collective combining trees
- Network protocol designed for efficient transfer of 8- to 64-byte messages at low overhead; combination of small packet sizes and a rich set of packet-by-packet adaptive routing algorithms ensure high utilization under heavy load

Aries uses a system-on-a-chip design that combines four high performance network interface controllers (NIC) and a high radix router. A single device provides the network connectivity for the four nodes on a blade. The Aries NIC is an enhanced version of the Gemini device used in the Cray XE6™ system, with the addition of a PCI-Express Gen3 host interface enabling connectivity to a wide range of CPU types.

[1] The Aries trademark is owned by Intel.

The Cray XC series uses a novel high-bandwidth, low-diameter network topology called "Dragonfly" [1]. Developed as part of the DARPA program, it provides scalable global bandwidth while minimizing the number of expensive optical links. High-speed, low-cost electrical links connect groups of up to 384 nodes within a pair of Cray XC cabinets and optical cables are used for the long links between groups. The Cray XC system has no external switches and half the number of optical links as an equivalent fat tree. Overall, the Cray XC network is highly modular, supporting flexible configuration of systems with up to 92,544 nodes.

The Cray software stack uses the advanced features of the Cray XC network to provide highly optimized implementations of standard programming environments such as MPI and Cray SHMEM™ together with novel partitioned global address space (PGAS) programming models such as Fortran with Coarrays[2] [12], Unified Parallel C (UPC) [13] and Chapel [14]. A single Cray Linux Environment™ (CLE) release supports both the Cray XE6 and Cray XC systems. System-specific features are hidden behind standard application interfaces ensuring a high degree of compatibility and simplifying application porting between systems.

This whitepaper will discuss why a supercomputer needs a network with high global bandwidth; introduce the Dragonfly design and compare it with a fat tree; describe implementation of the Aries device and its use in Cray XC series; and present performance data.

Cost-Effective, High-Bandwidth Networks

Many HPC applications have nearest neighbor, or local, communication patterns. This feature reflects locality in the underlying physical process being simulated. Applications with regular data decomposition (e.g., a 2-D mesh) can be executed efficiently on a parallel system with network connections between neighboring nodes, such as the 3-D torus used in Cray XT™ and Cray XE6 systems [2, 3]. Mapping an application mesh of one size to a system mesh of another introduces a degree of non-locality, but excess router bandwidth compensates for this.

Applications using high-dimensional or unstructured meshes or adaptive mesh refinement techniques require longer-range communication once distributed over a parallel machine. Some important classes of algorithms require non-local communication from the outset. For example, where Fourier transforms are used, two non-local exchange operations per time step typically occur. Other classes of applications have unstructured traffic with no underlying data locality. When such applications are executed on a machine with only local connectivity, each network packet must traverse multiple hops between source and destination nodes. Each hop increases the network load. Additional network load is generated when such systems execute multiple concurrent jobs. For example, contention is generated when traffic from two or more jobs shares the same network resources. This additional load can arise as a result of the way the scheduler allocates jobs to nodes, or through their access to the file system.

The ability of an HPC network to execute such workloads can be characterized by the ratio of its global bandwidth[3] to its injection bandwidth. The torus routers used in the Cray XE6 system have five times as much routing bandwidth as injection bandwidth. This ratio is many times that of competing systems, but performance will degrade when packets take an average of more than five hops. The bandwidth that each node can sustain on complex traffic patterns (e.g., all-to-all) falls as the system size increases.

[2] Coarray Fortran (CAF) is part of the Fortran 2008 standard and is called Fortran with Coarrays.
[3] The bisection bandwidth of a network is often used in this context. It measures the worst-case bandwidth achieved by dividing a system in two, with all nodes in each half communicating with a peer in the other half. This measure is useful, although for global exchange or pseudo-random communication patterns only half of the traffic crosses the bisection. Global bandwidth measures take this into account, reporting the bandwidth on traffic patterns for which each node (or set of nodes) communicates equal amounts of data with every other node (or set of nodes) in the system.

The objective of the Cascade program was to make Cray systems more productive by increasing their programmability, performance and efficiency. By increasing the global bandwidth, the system can efficiently execute a wider range of applications.

A class of networks known as "fat trees" [4] is widely used in constructing clusters and mainstream HPC systems. These networks can have global bandwidth equal to their injection bandwidth although they are frequently tapered, reducing bandwidth and cost. A fat tree network is constructed using routers of a given radix *k*. In a full bandwidth network, half of the ports connect down to the nodes and half connect up to additional stages of switch. At the top stage all links connect down. The number of nodes connected by a network with *s* stages is $2 \times (k/2)^s$. In a high density design, the lower stages of the tree connect nodes on the same board or within the same chassis. Short electrical links can be used to connect these components to switches within a cabinet, but the upper stages of the network connect widely separated groups of nodes. Larger fat tree networks are built using a three-stage network. Optical cables connect a single stage of top-of-rack switches to high port count external switches providing two additional network stages.

However, the need to add network stages as the system size grows lies behind a number of the problems that make fat trees unsuitable for constructing large, high-bandwidth networks. Specifically:

- High router radix (e.g., 64) or a four-stage network. While low port count top-of-rack switches are relatively inexpensive, the high port count switches necessary for building large systems are significantly more expensive. Furthermore, building top switches that provide more than two network stages is impractical.
- One long link for every node in a large full bandwidth network. With high signal speeds this link generally needs to be optical. The number of these links can be reduced in a tapered fat tree, but it reduces global bandwidth proportionately.
- Many unused switch ports unless the system size and the router radix match well. This feature leads to big discontinuities in the cost per port as the system size crosses boundaries set by the router radix and the switch implementation.

Consider a fat tree network constructed from the 36-way routers typically used in InfiniBand networks. With 18 blades and a router per chassis there are 18 links up from each chassis. The chassis can be connected with 18 × 108-port switches to construct systems with up to 1,944 nodes. However, if the system has 2,000 nodes then it requires 144 or 216 port switches to connect the chassis together. Many of the network ports will go unused. Upgrading a system from 1,500 to 2,000 nodes would require the radix of the top switch to be increased, adding substantially to the cost. Furthermore, the maximum size of a system that can be constructed with three network stages is $2 \times 18^3 = 11664$ ports — less than half of the size of a large Cray system today. Building tomorrow's even larger systems would require a fourth network stage, increasing the cost per node as the relatively inexpensive 36-way switches are replaced by a second set of high port count switches. The Cray "Black Widow" design [5] used in the Cray X2™ and Cray XT5h™ supercomputers demonstrated that a fat tree network can provide the required global bandwidth but is not cost effective at the scale required for the Cray XC system.

The Cray XC network combines the high global bandwidth of Black Widow and the cost effectiveness of the Cray XE6's torus. Dragonfly is a "direct" network, therefore avoiding the need for external top switches and reducing the number of optical links required for a given global bandwidth. Additionally, its design builds on earlier work showing the value of low-diameter networks constructed from high-radix routers. Ideally, a router would have a sufficient number of ports to connect to all of the other routers in the system — the network would have diameter one. However, this topology is not practical for large systems. But if a group of routers — acting in concert as a single, very high radix router —

pooled their links, then the group as a whole could have enough links to directly connect to all of the other groups in the system. This idea is the key one behind the Dragonfly network.

Low-cost electrical links are used to connect the NICs in each node to their local router[4] and the routers in a group (see Figure 1). The maximum group size is governed by the maximum length of the electrical links which will, in turn, be limited by the signaling rate on the links.

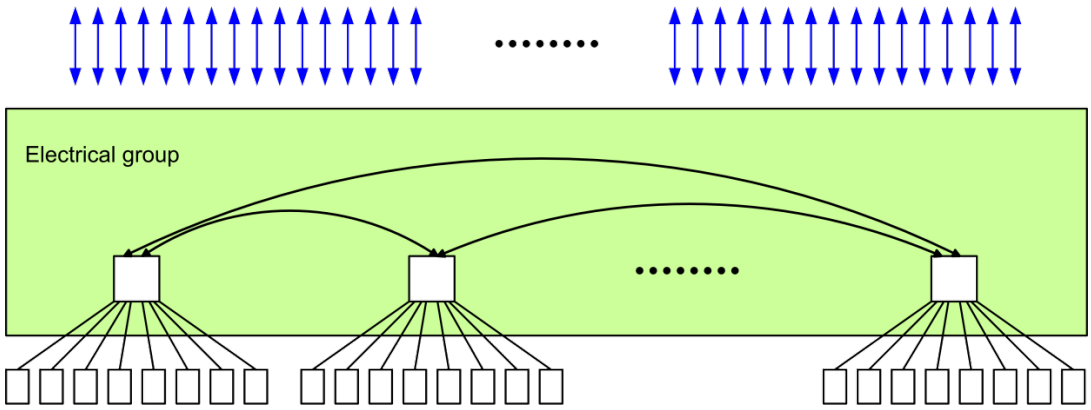

Figure 1: Short electrical links connect both the NICs to the routers and the routers in each group of a Dragonfly network. The routers in a group pool their global links (shown in blue).

Each router provides both "intra-group" links that connect it to other routers in its group and "inter-group" links (also known as global links) that connect it to other groups. The routers in a group pool their inter-group links, enabling each group to be directly connected to all of the other groups (see Figure 2). In large systems optical links are used for the long links between groups. In smaller systems the links between groups can be electrical although this limits system size.

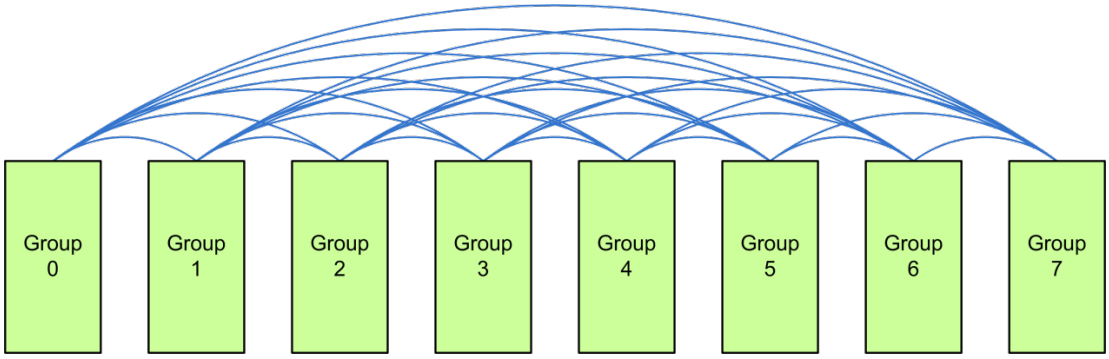

Figure 2: Global links connect Dragonfly groups together — these links are optical in a large system.

Each Dragonfly group has a fixed number of routers and provides a fixed maximum number of global ports, ensuring that network cost scales linearly with the system size. The global bandwidth of a Dragonfly network is given by:

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

[4] NICs and routers are combined in the Cray XC series. While this technique can be useful in reducing cost, the Dragonfly design does not require it.

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 global network can be depopulated to reduce its cost. Notably absent here are the discontinuities in cost that characterize fat tree networks.

The network is described by the group size, the number of groups and the number of links connecting each group to all of the other groups.

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 requires electrical hops in the source and destination groups plus an optical hop between groups. Electrical bandwidth is over provisioned to allow for the higher loading on the intra-group links[5].

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. This situation is known as worst-case traffic. In order to achieve high bandwidth on this type of traffic it is necessary to select "non-minimal" paths in which packets are routed through randomly selected intermediary groups. Non-minimal routing increases the aggregate network load but balances it over all of the available links.

The extent of the advantage a Dragonfly network has over a fat tree can be seen in the extent to which packets are minimally routed. With 100 percent minimal routing, Dragonfly has a 2:1 advantage — each packet takes one optical hop at most as opposed to two in a fat tree. If all traffic is non-minimally routed then load on the optical links is equal to that in a full bandwidth fat tree[6]. Traffic patterns generating high volumes of global traffic are well suited to minimal routing given good diversity in the range of destinations each node is communicating with at any given point in time. 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 will be contained within the electrical connected groups; the load on the global links will be low. The large electrical group size and high-bandwidth optical network between groups ensures that Cray XC 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.

Aries

Aries is a system-on-a-chip device comprising four NICs, a 48-port tiled router and a multiplexer known as Netlink. A single Aries device provides the network connectivity for all four nodes on a Cray XC blade (see Figure 3, next page). The only additional network components required are the cables and backplanes connecting Aries devices together.

Each of the four Aries NICs provides an independent 16X PCI-Express Gen3 host interface. In the current Cray XC blade design these interfaces connect to four independent dual socket Intel® Xeon® nodes. Each node has a pair of Intel Xeon E5 processors (providing 16 or more cores) and 8 DDR-3 memory channels each with a single DIMM. Memory capacity is 64 or 128 GB per node. Future Cray XC

[5] The extent to which intra-group bandwidth needs to be over provisioned depends on the traffic pattern. A factor of two is desirable.
[6] The advantage of reduced numbers of optical links is lost if all traffic is non-minimal, but the reduced costs arising from use of a direct network remain.

series blade designs could include other x86 processors, hybrid CPU/accelerator nodes or different ratios of NICs to CPUs.

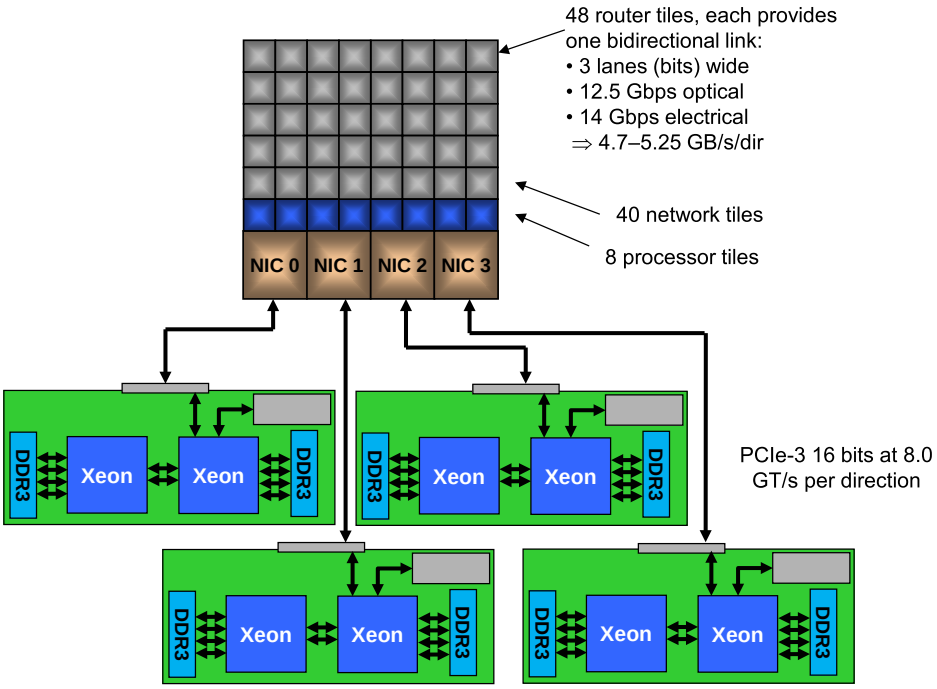

Figure 3: *A single Aries system-on-a-chip device provides network connectivity for the four nodes on a Cray XC blade.*

The Netlink multiplexor provides dynamic load balancing, distributing packets over the injection ports according to load. The Aries router connects 8 NIC ports to 40 network ports, operating at rates of 4.7 to 5.25 GB/s per direction per port. Each router provides both intra-group and inter-group links.

Aries is manufactured using the TSMC 40nm process. Aries has 120 network SerDes operating at rates of 12.5 to 14 Gbps and 64 PCI-Express SerDes operating at 8 Gbps. The die size is 16.6 mm × 18.9 mm. The gate count is approximately 217 million. Aries uses a HiTCE ceramic package developed by Kyocera. The package is 52mm square with a 51 × 51 ball grid array.

Aries NIC

The Aries NIC is based on the hardware pipeline developed for the Gemini design used in the Cray XE6 system. The node issues commands by writing them across the host interface to the NIC. The NIC then packetizes these requests and issues the packets to the network. Packets are routed across the network to a destination NIC. The destination executes the operation specified in the request packet and returns a response to the source. Packets contain up to 64 bytes of data.

The major functional units of the Aries NIC are shown in Figure 4 and described as follows (next page):

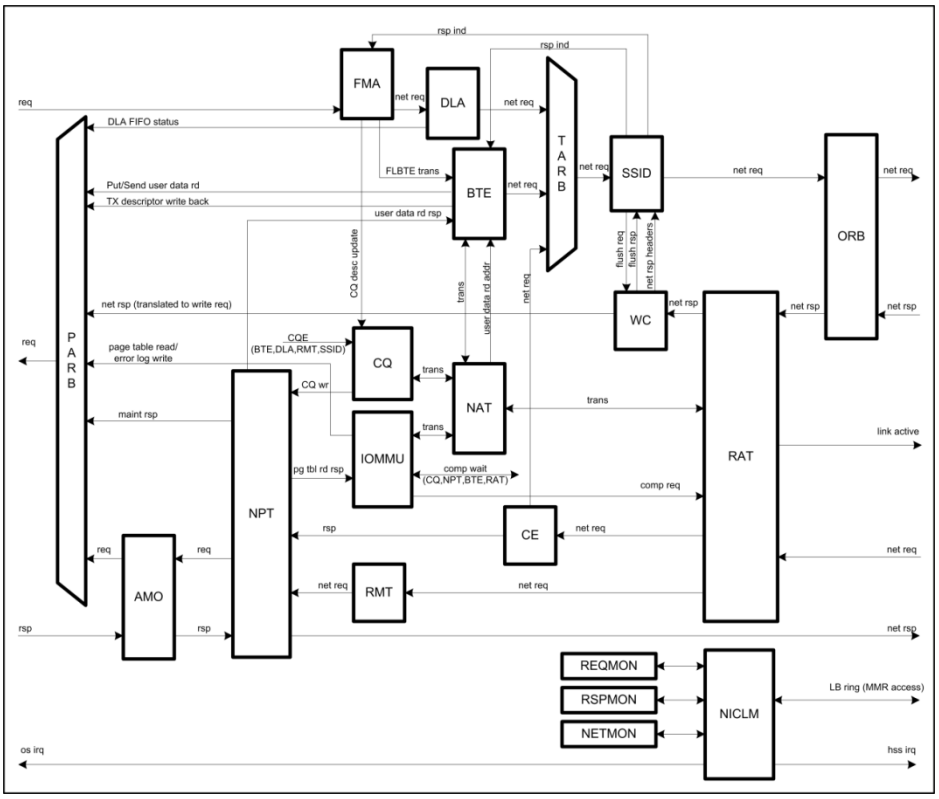

Figure 4: Aries NIC block diagram. Output flows from the host interface on the left towards the network on the right, input flows from right to left.

The Cray XC system runs distributed memory applications that use MPI, Cray SHMEM and PGAS programming models. The system is cache coherent but allows only memory from the local node to be cached. Remote references are performed as gets/puts and atomic memory operations (AMO) and are never cached by the initiator. This approach reduces overhead for the expected case of explicit communication between nodes. For example, a put causes data to flow directly across the network to the target node. If the target line is in the cache, the system is capable of updating the cache directly.

Address Translation

The Aries network employs a 3-tuple — the network address — to specify a logical address in a user process on a remote node. The address consists of an 18-bit node identifier, a 12-bit memory domain handle (MDH) associated with a memory segment registered at the remote node, and a 40-bit offset into this segment. The 70-bit network address extends the physical address space of the x86 node, enabling global access to all of the memory of the system. The MDH is combined with the offset to generate a user virtual address in the target process. The input/output memory management unit (IOMMU) translates this virtual address to a physical address in the target node.

The Aries IOMMU provides virtual-to-physical address translation for incoming network requests, fetching network responses, local reads generated by the NIC and completion event writes. It supports Linux 4 KB pages using two-level translation and huge page sizes of up to 64 GB using a single level of translation. Additionally, the IOMMU includes a four-way set associative page table entry (PTE) cache with 128 entries each of eight PTEs and a deep pipeline to hide the latency of fetching PTEs on a cache miss. The system does not require the use of large pages on Aries; their use is a performance

optimization as it is on the main CPU. Fetching PTEs can consume significant host bandwidth on random access patterns. This load can be reduced by using large pages and, thus, improve performance. Large pages are used by default for the Cray SHMEM symmetric heap. The Cray programming environment makes it straightforward to use large pages for the data segment and heap.

Aries supports virtualization of the MDH pool via a mechanism known as virtual memory domain handles (vMDH). The handle provided by the application is translated using a content-addressable memory (CAM) in each NIC. If a match is found, the corresponding MDH base value is added to the value in the packet to generate the MDH. A parallel job running across multiple nodes can refer to a particular memory segment (e.g., symmetric heap) using the same value on all nodes. The vMDH mechanism avoids the need for non-scaling data structures and reduces the cost of issue.

Aries supports virtual node addressing as well as virtual memory addressing. The NIC translates virtual node identifiers used by the communications libraries to physical network locations. The node identifier translation mechanism supports mapping around failed nodes and those allocated to other communication domains. Each job has the impression of running on a contiguous range of nodes, again avoiding the need for non-scaling data structures and reducing the issue overheads.

Remote Memory Access

Aries provides two remote memory access methods: fast memory access (FMA) and block transfer.

FMA is a mechanism whereby user processes generate network transactions — puts, gets and AMOs — by writing directly to an FMA window within the NIC. The FMA unit translates programmed I/O (PIO) writes by the processor into fully qualified network requests. On initialization, the user process is allocated one or more FMA descriptors and associated FMA windows. Writes to the FMA descriptor determine the remote processing element and the remote address associated with the base of the window. Writes to the put window generate a stream of remote put requests, each transferring up to 64 bytes of data. It is necessary to insert x86 sfence instructions between descriptor and data writes and on completion of each data write. Non-fetching AMOs such as atomic add are generated in a similar fashion. Storing an 8-byte control word to the get window generates a read request (get) of up to 64 bytes or a fetching AMO. FMA provides both low latency and high issue rate on small transfers.

FMA supports scattered accesses by allowing the user library to select which bits in an FMA window address determine the remote address and which the remote PE. Having set the FMA descriptor appropriately, one can, for example, store a cache line of data (64 bytes) to each process in a parallel job by simply storing a contiguous block of data to the FMA window. This approach can be used for addressing a series of remote nodes without incurring the cost of additional descriptor updates or sfence with each additional remote node accessed.

FMA provides source-side synchronization methods for tracking when put requests have reached a "globally ordered" point at the destination node and when responses to get requests have completed at the local node. Additionally, FMA makes it possible to issue puts that generate destination-side synchronization events enabling a process on the remote node to be notified of new data or poll a completion queue for its arrival.

The Aries NIC includes "FMA launch" — a novel fast path which optimizes the issue rate for single word remote memory operations. Once an FMA window has been initialized and a transaction has been opened, remote memory operations can be issued with just a single 128-bit write from host to NIC and an x86 sfence instruction. Each of these remote memory operations can target an arbitrary remote PE and offset. A completion event is generated once the transaction has been closed and all pending remote memory operations have been completed. The DMAPP library provides a lightweight wrapper

around this functionality for use by the CAF, UPC and Chapel compilers together with the MPI and Cray SHMEM libraries.

Aries augments FMA get functionality with an optional "flagged response" feature. When this feature is used, the get response data is accompanied by a flag word written to local memory. The application may poll on the flag word to determine the arrival of responses to individual get requests.

Aries supports a wide range of atomic operations, e.g., those with put semantics such as atomic add and those with get semantics, such as conditional swap. Aries maintains a 64-entry AMO cache, reducing the need for reads of host memory when multiple processes access the same atomic variable. Host memory is updated each time the variable is updated. Lazy update mechanisms are also provided to reduce load on the host interface. Network atomics are not coherent with respect to local memory operations; all processes must use the Aries application interface to update a global atomic variable.

The Aries block transfer engine (BTE) supports asynchronous transfer between local and remote memory. Library software writes block transfer descriptors to a queue and the Aries hardware performs the transfers asynchronously. The BTE supports memory operations (put/get) where the user specifies a local address, a network address and a transfer size. In addition, the BTE supports channel operations (send) where the user specifies a local address and a remote node but no destination address. Channel semantics require the user to have pre-posted a receive buffer with the target BTE.

The BTE unit supports up to four concurrent block transfers. If the transfer is large, then a single active channel is sufficient to sustain full bandwidth. The ability to pipeline transfers on multiple BTE channels ensures that full bandwidth can be maintained at smaller transfer sizes. By default, there is no guarantee of completion ordering amongst block transfers issued by a given NIC. Fence operations are used where necessary to ensure that one transfer is completed before another starts. Aries provides support for user-space launch of block transfer requests, reducing the issue latency and hence the minimum efficient block transfer size.

In general, FMA is used for small transfers and BTE for large. FMA transfers are lower latency but require the involvement of the CPU. BTE transfers take longer to start and impose a higher load on the host interface but once initiated can transfer large amounts of data (up to 4 GB) without CPU involvement.

Ordering and Completion

Cray networks support ordering between transactions which provide an envelope for multiple network operations.

When using FMA, software explicitly defines the bounds of each transaction. With BTE, a transaction consists of one or more block transfers. Hardware counters track the constituent requests of each transaction and their corresponding responses. In this way Aries determines when each transaction has completed at both the local and the remote nodes. The operations within a transaction may complete in arbitrary order. For example, there is no guarantee that bulk data will be delivered in byte order. Completion events are generated when the transaction has completed. The completion event reports the successful execution of the transaction or provides details of a failure. Aries completion queues (CQs) provide a lightweight event notification mechanism. The completion of a BTE or FMA transaction generates an event in a user (or kernel thread) specific queue. Completion events can be generated on either the source or the target node. They include both user data and transaction status information. Errors are reported via the status word in a completion event.

Aries uses a mechanism known as "sequence identification" to track the set of packets that make up a transaction. Every packet in the sequence contains the same synchronization sequence identification (SSID). Packets can be delivered in arbitrary order; each contains a network address and can be committed to memory as soon as it arrives, eliminating the need for re-order buffering. The sequence as

a whole completes and CQ events are generated when all packets have been delivered. This mechanism is implemented using the SSID and output request buffer (ORB) blocks on the output side (see Figure 4) and the receive message table (RMT) block on the input side. The RMT caches active SSID state, avoiding a network round trip for performance critical operations. It also matches BTE send requests to queued receive descriptors.

Reductions and Barriers

The Aries collective engine (CE) provides hardware support for reduction and barrier operations. The CE is optimized for latency sensitive, single-word operations that limit scalability in many large HPC applications. The CE block of each Aries NIC provides four virtual CEs (also simply referred to as CEs), each with its own reduction tree. At any point in time one collective operation may be pending on each of the CEs. The software stack may assign them to different jobs, to different communicators within a job or pipeline their use by a single communicator. The CE supports reduction trees of branching ratio up to 32. The high branching ratio ensures scalability with a tree of depth three covering most system sizes. Reduction trees are created during job initialization and potentially later on demand.

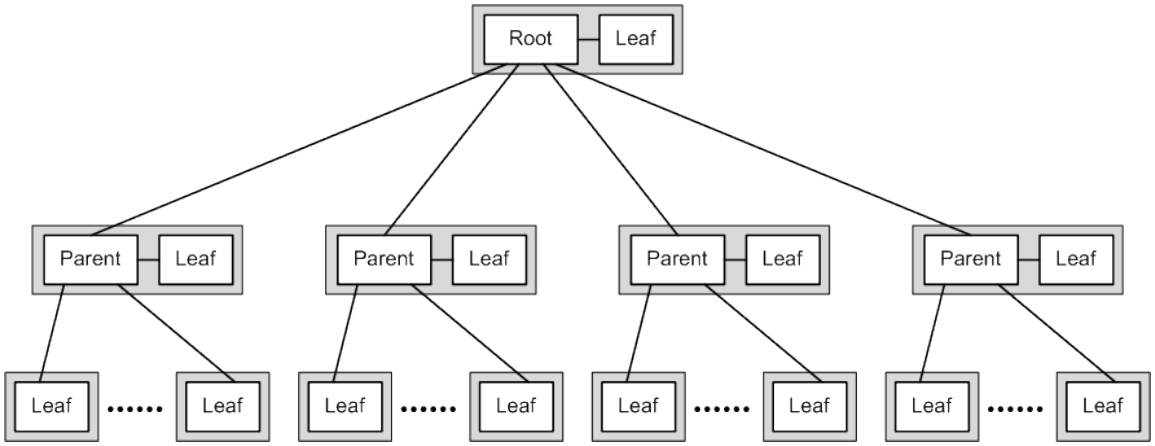

Figure 5: Reductions and barriers are offloaded to the Aries collective engine. Reduction trees with branching ratios of up to 32 are constructed amongst the NICs allocated to each job.

Each node joins a reduction operation supplying its contribution, generally the result of a local shared memory reduction operation. Partial results are passed up the tree towards the root with the NICs performing the reduction operations. The result is scattered back down the tree and written to memory in each of the participating endpoints. CQ events are generated on each node as the reduction completes. This approach reduces latency by avoiding unnecessary host interface crossings and memory updates. The Aries design does not require user processes to be scheduled in order to progress a reduction; the entire network reduction is offloaded to the NICs, reducing sensitivity to operating system jitter.

The Aries CE supports common logical, integer and floating point reduction operations on 32-bit and 64-bit operands[7]. The Min and Max reductions return a value and an identifier provided by each endpoint, supporting the MPI_MINLOC and MPI_MAXLOC operators.

[7] Aries does not support MPI_PROD. This rarely used reduction together with those for user-supplied operators and those requiring bit reproducibility are executed in the main CPU.

System Functionality

The Aries NIC provides a High Precision Event Timer (HPET) and an associated clock frequency generator. The HPET is accessible via memory-mapped I/O (MMIO). It can also provide regular interrupts. Global synchronization pulses are distributed over the network. This mechanism allows for the implementation of a synchronized global clock.

Aries Router

The Aries router is an evolution of the tiled design, used in the Cray Black Widow system [6] and the Cray XE6 system [3]. Using a tile-based microarchitecture facilitates implementation as each tile is identical and produces a very regular structure for replication and physical implementation in silicon.

A number of enhancements to the tiled-router design have been made in Aries to increase performance and support the Dragonfly topology:

- Wider data paths to handle higher bandwidths
- Number of virtual channels increased to eight
- Input buffering increased to tolerate longer cable lengths, up to 35m in the case of optical links
- Dynamic per-virtual channel input buffering allows a single active virtual channel to achieve full bandwidth over a long cable

The router tiles are arranged in a 6 × 8 matrix. Forty of the tiles, referred to as "network tiles," connect to the SerDes linking devices together. Eight of the tiles, called "processor tiles" connect to the Netlink block and hence the Aries NICs. Four of the processor tiles are shared between NIC0 and NIC1, the other four are shared between NIC2 and NIC3. The Netlink block multiplexes traffic over the processor tiles on a packet-by-packet basis. This mechanism distributes traffic according to load, allowing NICs to exceed their share of bandwidth on non-uniform communications patterns such as gather and scatter.

The tiles are made up of an input queue, a sub-switch and a column buffer. The input queue for each network tile receives packets and determines how to route them. It then sends the packet across a row bus to the sub-switch in the appropriate column. The sub-switch receives packets from each of the eight input queues in the row on each of the eight virtual channels (VC) — hence the 64 × 6 cross-point switch (see Figure 6, next page). It switches the packet to the appropriate output and sends it out on one of six column busses to the column buffer in the appropriate output row. The column buffers collect packet data from the six tiles within each column and multiplex them onto the output link.

A SerDes logic block (SLB) is external to each network tile (located around the perimeter of the Aries chip). It contains the SerDes itself, a link control block (LCB) and the SerDes management logic. The SLB provides the serial interface for router-to-router connections over electrical or optical links. The LCB provides reliable data transfer between devices. The LCB transfers data in sets of ten 48-bit flits, adding a 20-bit cyclic redundancy check (CRC) to the last flit in each set — the tenth flit is also used to carry global clock and network load information. The link input logic checks the CRC and acknowledges success or failure. The sending link retransmits on receipt of an error. The LCB includes a send buffer of sufficient size to cover the roundtrip time on a long optical cable.

Each tile has four sets of route tables, local minimal and non-minimal tables for routing within each group, global minimal and non-minimal tables for routing between groups. Minimal routes specify direct paths to nodes within each group and to other groups. Non-minimal routes distribute traffic uniformly over intermediate groups. The routing pipeline generates two minimal and two non-minimal outputs for every packet. Adaptive routing decisions are made by selecting from this choice of outputs

using routing control bits in the packet header, load information for each of the output ports, and a programmable bias mechanism. Aries provides three different load metrics — one local and two remote. Each Aries device computes its load and distributes the information to its neighbors every 10 cycles, ensuring that adaptive routing decisions are made on the basis of up-to-date information. The Aries router implements four VCs for requests and four for responses. Mechanisms are also provided for switching VCs. Each tile is able to determine the output VC based on its location and type, the input VC and a number of Dragonfly-specific routing parameters.

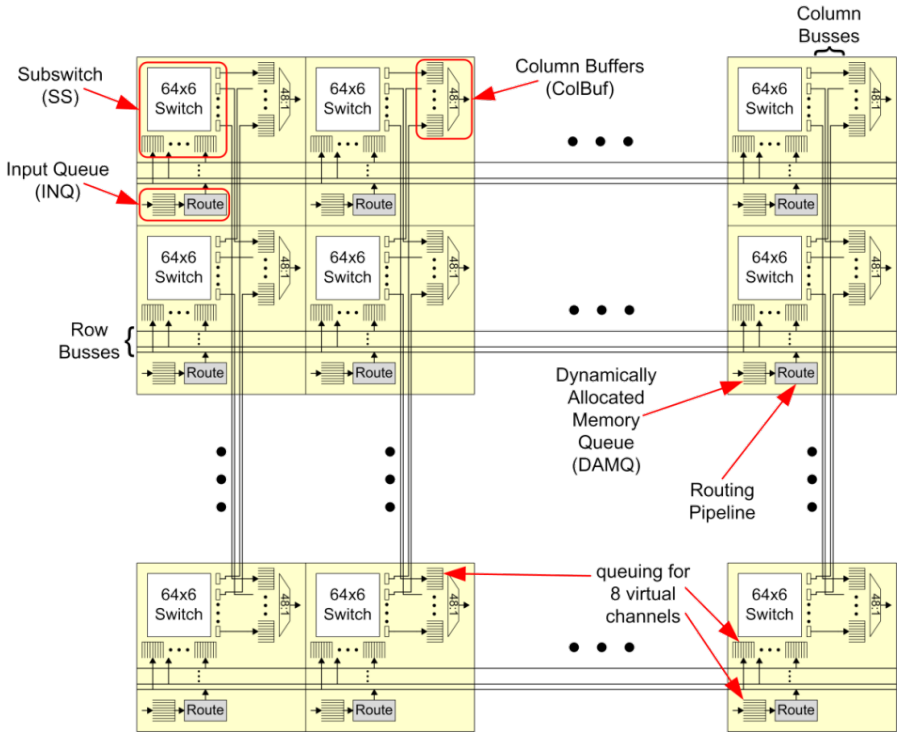

Figure 6: Aries tiled router. Each tile supports a single link, eight of which connect to the NICs and 40 to other Aries devices.

Dragonfly Implementation for the Cray XC Series

The Dragonfly topology was chosen for the Cray XC series to provide good global bandwidth and low latency at lower cost compared to a fat tree. Cray XC systems are constructed from four node blades with the processors and memory on daughter cards. Each blade has a single Aries network ASIC. Each of its NICs is connected to one node over a 16X PCI-Express Gen3 host interface. Each chassis has 16 such 64-node blades (see Figure 7, next page) and each cabinet has three chassis (192 nodes). The Dragonfly network is constructed from two-cabinet electrical groups with 384 nodes per group. A large group size is beneficial to both performance and scalability but is limited by mechanical and electrical constraints. The number of ports in the Aries router and the balance between group and global bandwidth also influences the group size choice. All connections within a group are electrical and operate at 14 Gbps per lane.

A 2-D all-to-all structure is used within each group (see Figure 8). The chassis backplane provides connectivity in the rank-1 (green) dimension; each Aries is connected to each of the other 15 Aries in the chassis. The rank-2 (black) dimension is formed by connecting three links from each Aries to its peer

in each of the other chassis making up the group. Aries link speeds limit the maximum length of the copper cables used to connect chassis to approximately 3m. This length is sufficient to connect the six chassis housed in two adjacent cabinets. Cables are attached directly to the blades in order to minimize the number of connectors per link and reduce the associated signal degradation.

Each Aries provides 10 global links — a total of 960 per group and a sufficient number to connect to 960 other groups. This is more global links than required and so they are combined in sets of four, restricting the maximum system size to 241 groups. The chassis backplane connects 10 global links from each pair of blades to five CXP connectors (see Figure 7). The Cray XC system uses 12X active optical cables (AOCs) operating at 12.5 Gbps to connect the groups together. Electrical cables can be used for the global links, although this restricts the maximum system size.

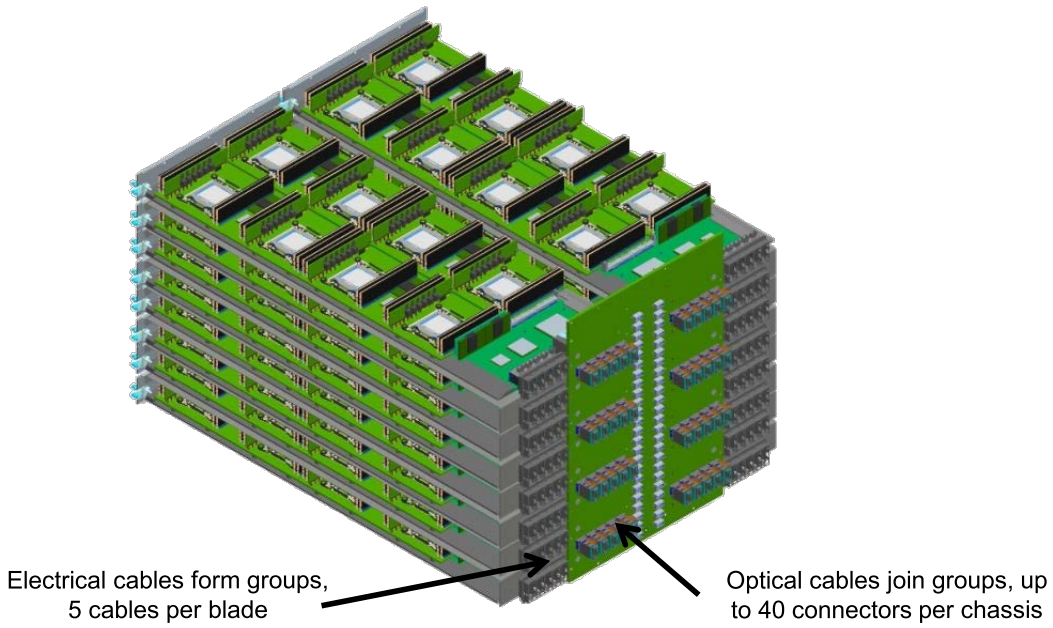

Figure 7: A Cray XC chassis consists of 16 four-node blades with one Aries per blade. The chassis backplane provides all-to-all connectivity between the blades. Each blade provides five electrical connectors used to connect to other chassis within the group. Each Aries also provides 10 global links. These links are taken to connectors on the chassis backplane. Active optical cables are used to connect the groups together.

The four NICs on each Aries connect to eight router ports for packet injection/ejection. When intra-group traffic is uniformly distributed over one dimension, a portion will stay local to each router: one part in 16 for the green dimension and one part in six for black. The design of the Cray XC group (see Figure 8, next page) ensures that intra-group bandwidths meet or exceed the factor of two desirable for Dragonfly networks. With 10 optical ports per Aries, the global bandwidth of a full network exceeds the injection bandwidth — all of the traffic injected by a node can be routed to other groups.

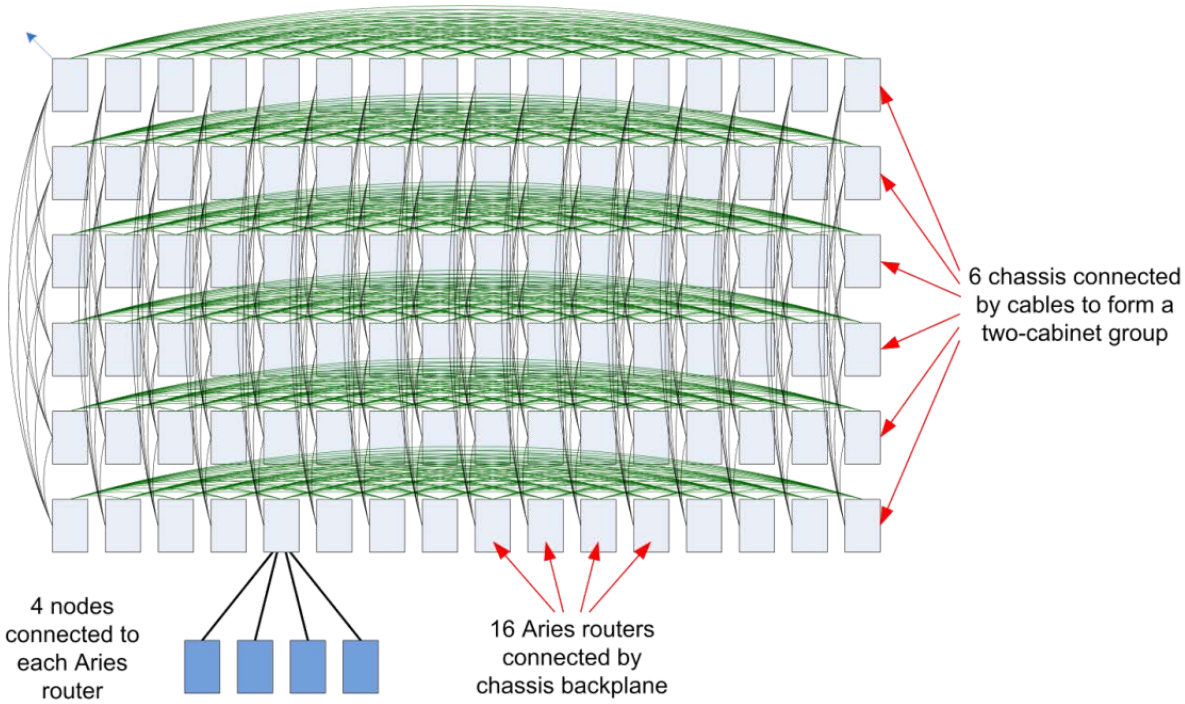

Figure 8: Structure of a Cray XC system's electrical group. Each row represents the 16 Aries in a chassis with four nodes attached to each and connected by the chassis backplane. Each column represents a blade in one of the six chassis per two-cabinet group, connected by copper cables.

A system with full global bandwidth requires the following components:

	Per Group	Per Node
Aries devices	96	0.25
9X copper cable	(5×16×6)/2 = 240	0.625
12X active optical cable	(40×6)/2 = 120	0.3125

Table 1: Cray XC component counts

A comparable fat tree such as FDR InfiniBand requires two 4X active optical links per node running at higher clock speed to provide the same global bandwidth.

Cray XC Routing

In a Cray XC network request packets are routed from source to destination node. Response packets are independently routed back to the source node; request and response traffic are not required to follow the same path. Packets are routed deterministically or adaptively along either a minimal or non-minimal path. Minimal routing within a group will always take at most one green and one black hop. Minimal routing between groups will route minimally in both the source and target groups and will take exactly one global optical hop. Note that minimal routing implies a direct route between a source and a target, not the minimal number of hops required. Minimal paths may differ in hop count if, for instance, one

path does not require a green and/or black hop in the source and/or destination groups due to placement of the global link used between groups. For example, in a full bandwidth system with six groups and 48 cables from each group to each of the other groups, each Aries will be directly connected to all of the other groups, reducing the minimal path length to three.

Non-minimal routing in the Cray XC system is an implementation of Valiant's routing algorithm [6]. This algorithm is used to avoid congestion and spread non-uniform traffic evenly over the set of available links in the system. Non-minimal routes within a group can be thought of as routing minimally from the source Aries to a randomly selected intermediate router known as the "root," and then minimally to the target Aries. Non-minimal routes within a group can take up to two green hops and two black hops. A global non-minimal path routes "up" to an intermediate Aries anywhere in the system and then "down" to the destination node using a minimal path. The route tables are populated so as to distribute non-minimal traffic uniformly over all routers in the system. The maximum path length of a global non-minimal path is 10 hops. Most non-minimal paths are shorter than this (six or seven hops) in all but the largest systems.

Aries supports a sophisticated packet-by-packet adaptive routing mechanism. The routing pipeline selects up to four possible routes at random, two minimal and two non-minimal routes. The load on each of the selected paths is compared and the path with the lightest load is selected. Current link loads are computed using a combination of downstream link load, estimated far-end link load and near-end link load using the methods introduced in reference [1]. Downstream load information is propagated from router-to-router at high frequency so as to ensure that routing decisions are made on the basis of up-to-date information. Load metrics can be biased towards minimal or non-minimal routing using information supplied by the application. With the default routing algorithms, packets will take a minimal path until the load on these paths increases and a non-minimal path is preferred.

Each network packet specifies a routing control mode. Where the runtime requires ordered packet delivery, the route is selected using a deterministic hash computed on fields from the packet header. In general, the Cray XC system uses adaptive routing.

The adaptive routing algorithms can be tuned at both the system and application levels. At the system level, we tune the routing policy based on the packet's path. The network API allows the programmer to select between adaptive routing policies. For example, a library developer might determine that a particular policy is appropriate to a communication intensive function (e.g., a fast Fourier transform) and switch to this policy for the duration of the call.

Network Reliability, Availability and Serviceability

Aries offers sophisticated reliability, availability and serviceability (RAS) capabilities with comprehensive hardware error correction coverage, resulting in fewer retransmissions, higher availability and fewer maintenance events. The network uses a combination of both hardware and software techniques to avoid a broad class of well-understood faults such as single-bit memory errors and transmission errors on network links. However, when a fault does manifest as an error, the hardware and software layers identify and contain the fault, recover from the error if possible and prevent data corruption. Aries provides support for identification of the first error within a sequence of errors occurring in a short interval of time. This mechanism simplifies root cause analysis of problems.

Although modern semiconductor fabrication processes have improved yield and reduced the extent of manufacturing defects or "hard" errors in the silicon, high-density, low-voltage devices such as Aries are susceptible to influences of process variation, electrical noise and natural radiation interference. The resultant errors from these effects are referred to as "soft" errors since they can corrupt state, but generally do not result in any permanent damage to the underlying circuitry. Soft errors are detected

and possibly corrected by mechanisms such as parity, error correcting code (ECC) and CRC. Within Aries, major memories and much of the data path are protected using ECC. Data in transit between devices is protected using CRCs.

Packet level CRC checks in the network transport layer provide immediate indication of corrupt data from end to end. Errors are reported as precisely as possible: Data payload errors are reported directly to the user, but control errors cannot always be associated with a particular transaction. In all cases, either the operating system or the Hardware Supervisory System (HSS) are notified of the error. Router errors are reported at the point of the error with the packet possibly being discarded. The initiating NIC times out the transaction, reporting a timeout error to the user process.

The Aries network link-layer protocol provides reliable packet delivery across each link. A 20-bit CRC protects 60 bytes of data distributed across three SerDes lanes. The LCB provides automatic retransmission in the event of error. For a 25,000 node system with a bit error rate (BER) of 10^{-7} on all SerDes lanes, the expected rate of undetected errors is less than one every 7,250 years. Measured bit error rates on correctly functioning links are better than 10^{-12}.

The SSID mechanism provides functionality to assist in the identification of software operations and processes impacted by errors. This aids error recovery and improves system resiliency by enabling the overall system to survive error events even when remedial action must be applied to those impacted processes. The SSID provides this functionality by monitoring errors detected at the source NIC as well as errors returned from the destination NIC in response packets. Information on all such errors is included in the status field of the completion event. By incorporating error detection and handling mechanisms into the communication protocol, the status of every transaction can be passed back to the originating process. User level libraries, MPI in particular, use this information to ensure that applications are resilient to transient network failures.

Adaptive routing helps mitigate the effects of link failure. The adaptive routing hardware spreads packets over the correctly functioning links according to load. If a link fails, the router automatically stops sending packets in that direction. In the event of lost connectivity, which may occur on blade failures or hardware removal, it is necessary to route around the problem. The HSS performs this task.

PCI-Express traffic between the CPU and Aries is protected using a 32-bit link layer CRC per packet with automatic retry in case of error. PCI-Express includes support for advanced error reporting (AER) so that link level errors can be reported to the host processor (and/or HSS) to assist in fault isolation and recovery. Correctable AER errors are recovered by the PCI-Express protocol without the need for software intervention, and without any risk of data loss. Uncorrectable errors are passed to the operating system.

Cray XC systems minimize the number of passive components in each link path. This approach increases reliability and facilitates the diagnosis of failed links. Each link comprises three separate SerDes lanes. Links can degrade to two or one lane in the event of failure. Additionally, Cray XC systems provide water cooling of the AOCs — regulating their temperature improves both error rates and component lifetimes.

The HSS monitors the Cray XC system (see Figure 9). Daemons running on the blade level controllers are responsible for monitoring the memory mapped registers (MMR) which indicate the health of the network. Fatal errors can be configured to generate interrupts to HSS. Non-fatal errors are generally polled in order to track error rates. These monitors generate events that feed into a link failover manager. It reacts to failed links or routers by computing updates to the routing tables that avoid the problem. Where necessary, HSS can quiesce network traffic so that route table changes can be deployed safely. Applications pause briefly while this operation is in progress; traffic is released once the re-route process has completed.

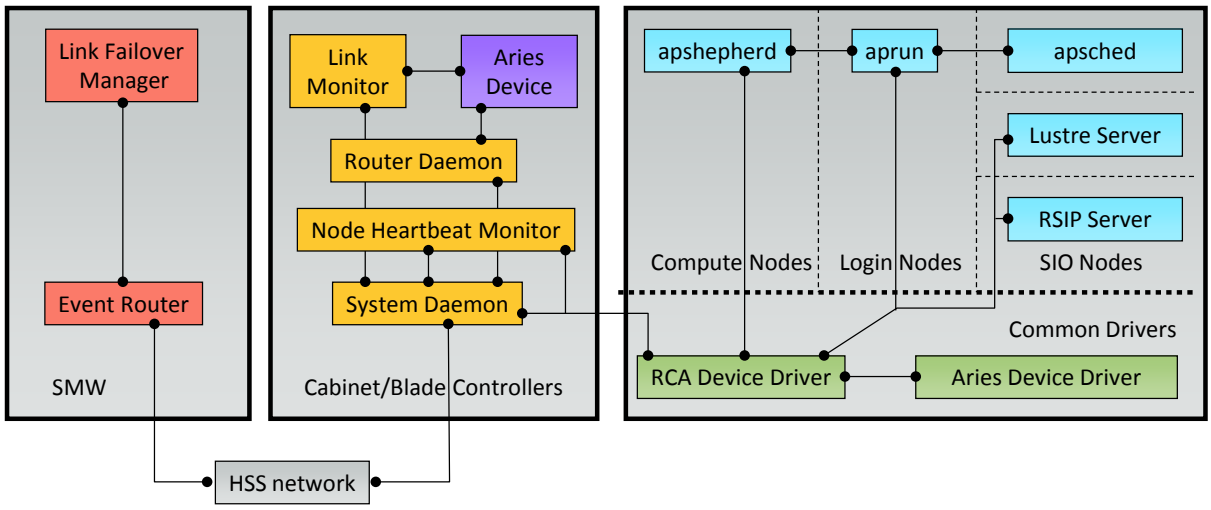

Figure 9: Cray XC Hardware Supervisory System (HSS)

Cray XC Network Software Stack

The Cray XC system supports both kernel communication, through a Linux device driver, and direct user space communication, where the driver is used to establish communication domains and handle errors but is bypassed for data transfer. Parallel applications typically use a library such as Cray MPI or Cray SHMEM in which the programmer makes explicit communication calls. Alternatively, they may use a programming model such as CAF, UPC or Chapel in which the compiler automatically generates the inter-process communication calls used to implement remote memory accesses.

Standard communications libraries are layered over the user level generic network interface (uGNI) and/or the distributed memory applications (DMAPP) library which perform the Cray network-specific operations (see Figure 10). Kernel modules such as the Lustre network driver (LND) communicate via the kernel generic network interface (kGNI).

Each process is allocated its own FMA descriptor allowing independent issue of remote memory access requests. Multithreaded processes can allocate an FMA descriptor per thread provided the number of threads per node is limited; each Aries NIC supports a maximum of 127.

Security is maintained by the use of communication domains. Each such communication domain defines a set of processes or kernel threads that are allowed to communicate. Creating a communications domain is a privileged operation. Any attempt to communicate with a process outside of the communication domain is blocked by hardware and generates an error.

Program initialization follows a sequence of steps in which the placement scheduler (ALPS) first creates a communication domain and then starts the user processes. The processes sign on to the communication domain, create their completion queues and register memory with the communication domain. Having completed this sequence of steps, the processes in a parallel job can initiate either put/get- or send/receive-style communication without operating system intervention. These operations are asynchronous with completion events being generated when an operation or sequence of operations has been completed. This approach promotes programming models in which communications is initiated early so as to hide latency. Aries optimizes for this, supporting both large numbers of pending operations and high issue rates.

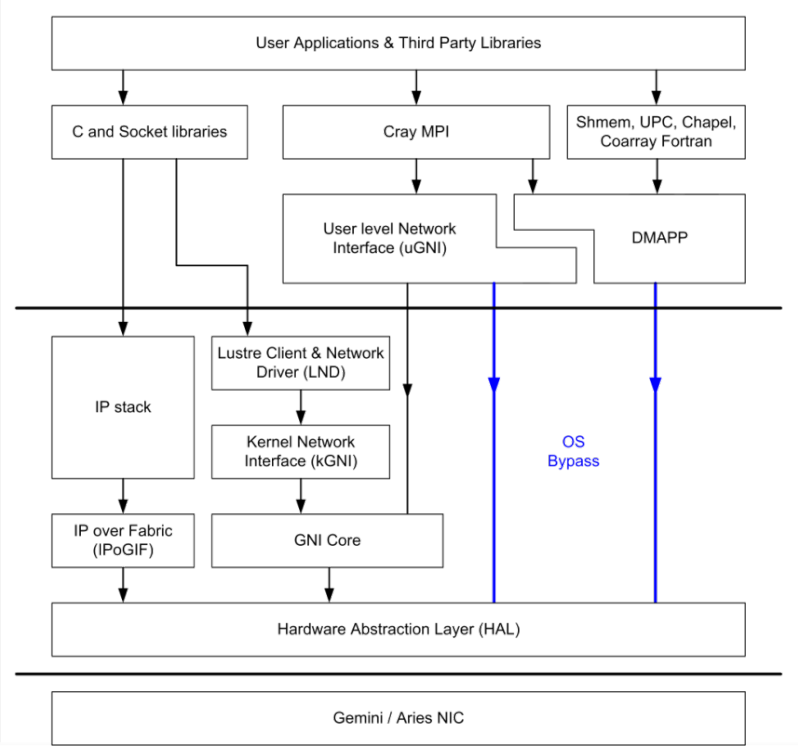

Figure 10: Cray XC network software stack. A high degree of compatibility is maintained with the Cray XE6 system. The Aries implementations of GNI and DMAPP build on those for the Gemini NIC, adding support for Aries features. Compatibility is maintained for software that uses these interfaces.

Cray MPI uses the MPICH2 distribution from Argonne. The MPI implementation uses a Nemesis driver for Aries layered over uGNI [7]. Use of FMA gives MPI applications the ability to pipeline large numbers of small, low latency transfers — an increasingly important requirement for strong scaling on multicore nodes. Where space is available, intermediate size messages are sent eagerly to pre-allocated system buffers. Large messages are transferred using a rendezvous protocol in which bulk data transfer occurs after matching of an MPI message header and a receive request. Message matching is progressed by each call, or in the case of large messages, using an optional progress thread. The Aries block transfer engine is used to provide high bandwidth, good overlap of computation and communication, and efficient use of main memory bandwidth. Implementation of latency sensitive collectives, including MPI_Allreduce and MPI_Barrier, is optimized using the Aries collective engine.

Cray SHMEM [9] provides an explicit one-sided communication model. Each process executes in its own address space but can access segments of the memory of other processes, typically the static data segment and the symmetric heap through a variety of put and get calls, AMO calls and collectives. Since the Cray T3D system, Cray supercomputers have supported Cray SHMEM. Its implementation for Aries provides the application programmer with fine-grain control of communication with minimum overhead.

Aries is designed to provide efficient support for emerging partitioned global address space (PGAS) programming models. Compiler-generated communication typically results in large numbers of small irregular transfers. The Aries FMA launch mechanism minimizes the issue overhead and its high packet rate maximizes performance. Cray's Fortran, UPC and Chapel compilers use a common runtime implemented with DMAPP. The DMAPP library includes blocking, non-blocking and indexed variants of put and get along with scatter/gather operations, AMOs and optimized collectives. The GNI and DMAPP interfaces are provided for third parties developing their own compilers, libraries and programming tools.

The Aries versions of GNI and DMAPP preserve the application interfaces developed for Cray XE6 systems, ensuring compatibility of higher levels of software. Support for Aries hardware features such as FMA launch, hardware collectives and user space BTE transfers is encapsulated within the Aries versions of GNI and DMAPP.

Aries Performance

The following measurements of Aries performance were made on a 750 node pre-production Cray XC system using Intel Xeon E5 (codenamed Sandy Bridge) CPUs operating at 2600MHz and CLE release 5.0[8].

Peak bandwidth of the 16X PCI-Express Gen3 interface connecting each Cray XC series node to its Aries is 16 GB/s, 8 giga-transfers per second, each of 16 bits. However, PCI-Express overheads are 24 bytes on every transfer. In addition, overheads for alignment and flow control reduce the bandwidth available to application data. The Aries NIC can perform a 64-byte read or write every five cycles (10.2 GB/s at 800MHz). This number represents the peak injection rate achievable by user processes. With symmetric traffic both the host interface and the links have additional overheads as requests and responses share data paths. Their effect is to reduce bandwidth on symmetric bulk data transfer to approximately 8 GB/s in each direction.

Each processor tile can inject a 48-bit flit per clock cycle — at 875 MHz this equates to 5.25 GB/s. Links are three lanes wide in each direction and operate at speeds of up to 14 GHz. Their raw bandwidth is 5.25 GB/s. Optical links are over provisioned by 10/8 in order to maintain bandwidth with AOCs operating at 12.5 GHz. Each 64-byte write (put) requires 14 request flits and 1 response flit. A 64-byte read (get) requires three request flits and 12 response flits. LCB overheads are 1 flit in 10. Taking packet and LCB overheads into account, overall payload efficiency is (64 bytes/15 flits) × (9 flits/60 bytes) = 64 percent. With balanced symmetric traffic from 4 NICs running at 8 GB/s, packets must be distributed over at 10 or more of the 40 network ports to match the injection load.

In each electrical group 384 links cross the bisection in the green dimension and 432 cross in the black dimension. The green dimension sets the lower limit of 384 × 5.25 × 2 = 4032 GB/s. Bisection bandwidth of the system is controlled by the number of optical cables. With 12.5 GHz optical links, each AOC provides a peak bandwidth of 18.75 GB/s in each direction between a pair of groups. The peak bisection bandwidth of a system (in GB/s) is given by

$$18.75 \times \frac{G}{2(G-1)} \times optical\ cables\ per\ group\ \times\ G$$

where G is the number of full groups. The ratio G/2(G-1) reflects the proportion of links crossing the bisection, falling from a value of one for two groups to ½ as the system size increases. The maximum number of optical cables per group is 240. For reduced bandwidth systems or systems in which the number of optical ports per group does not divide evenly into the number of other groups, the number of optical cables per group, and hence the global bandwidth, is reduced.

For applications in which traffic is uniformly distributed from each node to each of the other nodes (e.g., all-to-all), global bandwidth controls performance rather than the bisection — and all the optical links contribute. Peak global bandwidth is 11.7 GB/s per node for a full network. With the payload efficiency of 64 percent this equates to 7.5 GB/s per direction. A high proportion of all traffic can be

[8] Performance results may vary with the network configuration, choice of processor or software release. Please contact your local Cray representative for figures relating to a specific configuration.

routed to remote groups. In practice, the large electrical group size ensures that significant volumes of traffic remain local to each group.

Measured end-to-end latencies for user-space communication[9] are 0.8 µs for an 8-byte put, 1.6 µs for an 8-byte get and approximately 1.3 µs for an 8-byte MPI message (see Figure 11). With more software overheads, the MPI latencies depend on the type and speed of CPU. Get latencies are higher than put as a PCI-Express read is required on the remote node. In each case there is little additional latency when transferring 64 or 128 bytes of user data rather than just 8 bytes. Latency is controlled by the number of host interface crossings. With multiple processes per node issuing requests, the host interface crossings are overlapped with data transfer and there is a high aggregate issue rate.

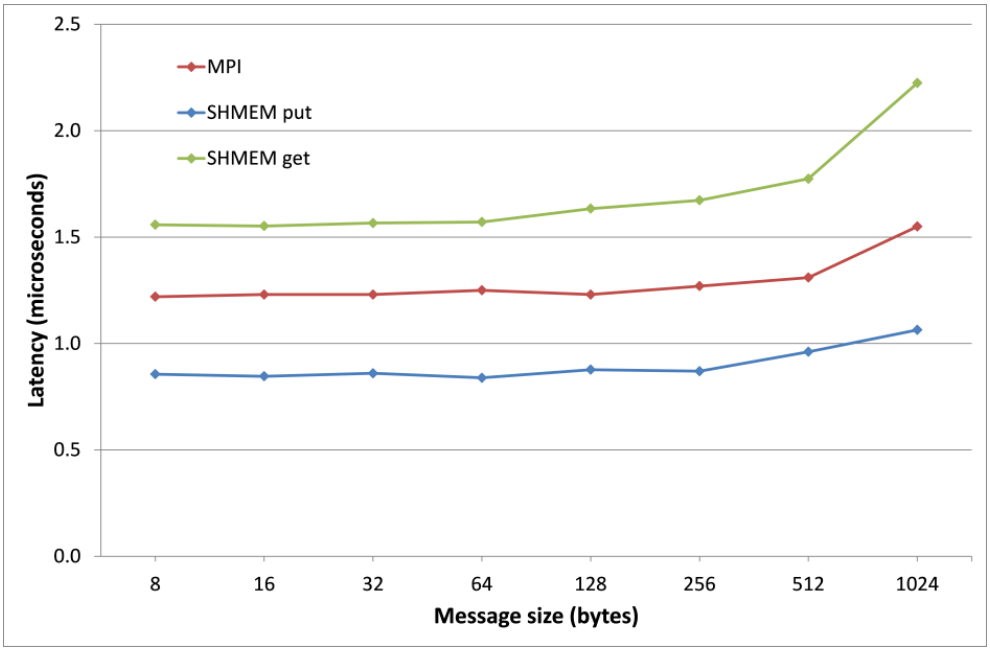

Figure 11: Aries latencies as a function of transfer size. Results are for MPI send/recv, Cray SHMEM puts and gets. Results were measured using Intel Xeon E5 CPUs operating at 2600MHz.

For a quiet network, each router-to-router hop adds approximately 100ns of latency, a maximum of 500ns for minimally routed packets. The adaptive routing hardware will select a minimal path when the network is quiet. Quiet network latency between any pair of nodes is less than 2 µs on the largest systems.

Bandwidths measured between user processes on different nodes are up to 10 GB/s (depending on the test used) for unidirectional traffic. Bidirectional bandwidths exceed 15 GB/s, 7.5 GB/s in each direction (see Figure 12, next page). With multiple issuing processes per node, half-peak bandwidth is obtained for puts of size 128 bytes.

[9] Latency measurements are for processes using CPU 0 — the CPU directly connected to the Aries NIC. Latencies are higher for CPU 1 as traffic must cross the QPI interfaces within each node.

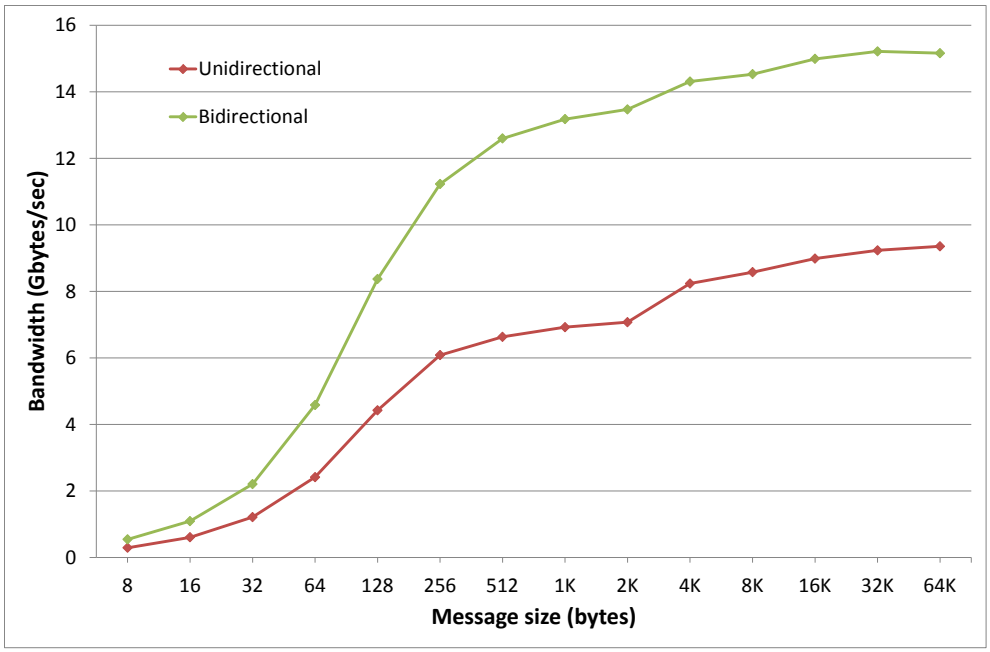

Figure 12: Aries put bandwidth as a function of message size. Results were measured using Intel Xeon E5 CPUs operating at 2600MHz.

A single node can issue put requests at rates of up to 120 million per second using the FMA launch mechanism (see Figure 13). If the MDH must be modified for each request, then the put rate drops to 80 million per second, as an additional write to the FMA descriptor and an x86 additional sfence instruction are required for each request. The issue rate for a single thread/process depends upon the CPU core — rates of up to 7.5 million requests per second per core have been measured to date.

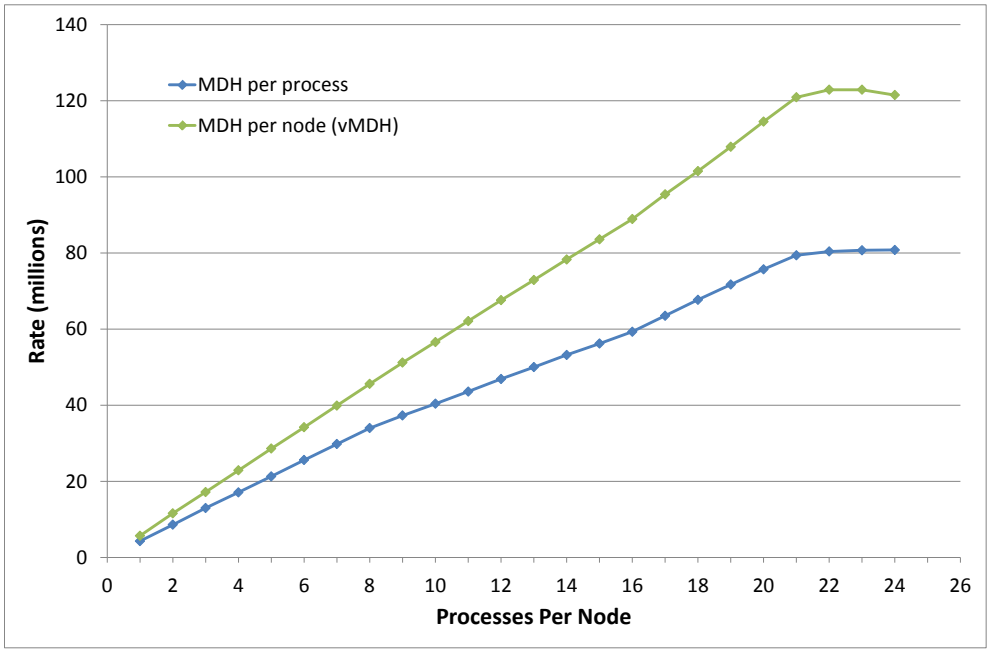

Figure 13: Aries put rates as a function of the number of processes per node. Hyperthreading was enabled for this test. Results were measured using Intel Xeon E5 CPUs operating at 2600MHz.

The Intel IMB benchmark [10] returns a bandwidth of 8.5 GB/s for the "pingpong" test in which one MPI transfer is active at any point in time (see Figure 14). Streaming tests (such as the OSU MPI bandwidth benchmark [11]) and those with multiple processes per node report bandwidths of up to 9.7 GB/s. Each NIC supports up to 1,024 outstanding packets, ensuring that bandwidths can be maintained between widely separated nodes. Cray MPI is optimized to minimize memory requirements on large jobs. Performance can be tuned by adjusting the thresholds at which the MPI implementation switches protocol. See dashed line in Figure 14 where thresholds are adjusted to increase bidirectional bandwidth at intermediate sizes.

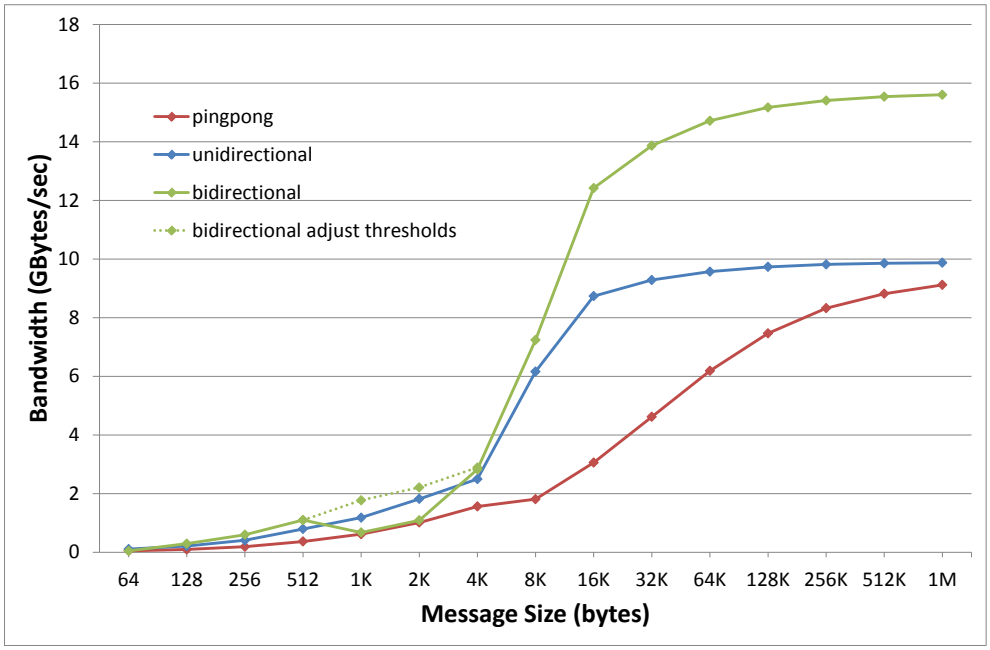

Figure 14: MPI bandwidth as a function of message size for ping pong (single message at a time), unidirectional and bidirectional tests. Results were measured using Intel Xeon E5 CPUs operating at 2600MHz.

Figure 15 (next page) shows measured MPI_Allreduce latencies for 64-bit floating-point sum. We compare the time taken using the optimised software algorithm with that obtained using offload to the Aries collective engine. Data is for 64 to 750 Intel Xeon E5 nodes with 16 processes per node. Latency for MPI_Allreduce over 12,000 processes is less than 10 microseconds. We see significant improvements in performance of the software algorithm from use of core specialization – a CLE feature in which specific cores (or Intel Xeon Hyper-threads) are dedicated to the operating system. The gains from core specialization are smaller at these scales with offload to Aries; the main CPU is required for the local shared memory reduction, but not to progress the network phase.

The Aries AMO unit is capable of one operation every two cycles. Rates of up to 400 million AMOs per second can be obtained when many processes update a single cached variable. AMO rates to random variables that miss in the AMO cache depend on the performance of host memory system and the load generated by user processes. Rates in the range of 50 to 120 million per second are possible depending on size of array being accessed. Large page sizes are required in order to maintain random put or AMO rates over extended target arrays.

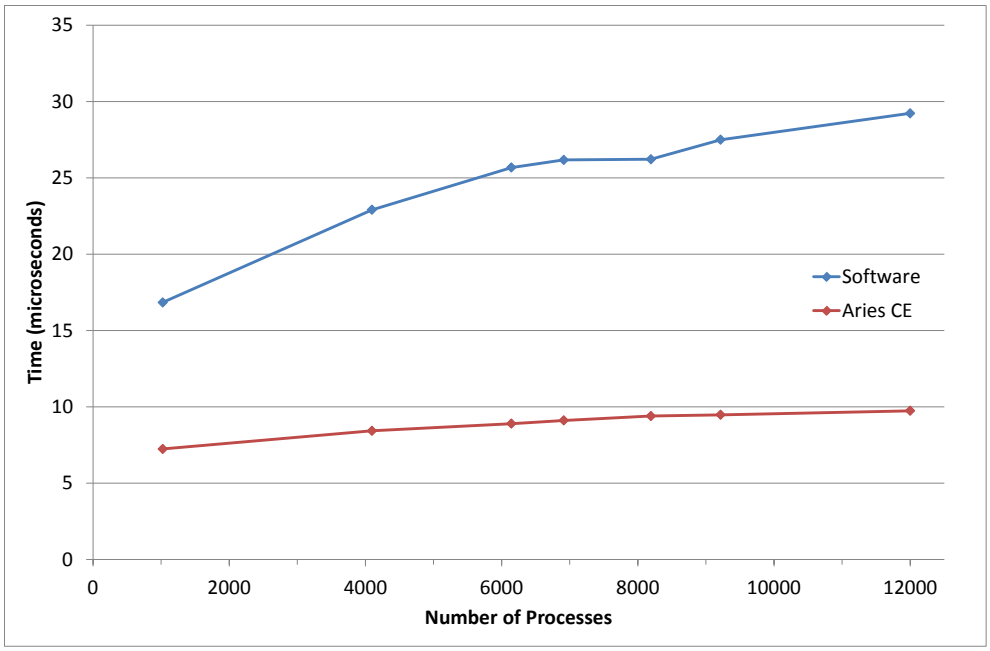

Figure 15: Latency of MPI_Allreduce as a function of number of processes for 64-bit floating point sum. Measurements are for up to 750 Intel Xeon E5 CPUs operating at 2600MHz with 16 processes per node and core specialization enabled.

Cray XC Series Configuration

Cray XC systems are constructed from a number of two-cabinet groups, each containing 384 nodes. The global links from each group are divided into bundles of equal size, with one bundle from each group connecting to each of the other groups. Configurations are characterized by the number of groups and the number of cables between each pair of groups

$$groups \times inter\text{-}group\ cables$$

The number of inter-group cables per bundle is given by

$$inter\text{-}group\ cables \leq FLOOR\left(\frac{global\ cables\ per\ group}{groups - 1}\right)$$

The maximum number of global cables per group is 240; this number may be reduced substantially in systems that do not require full global bandwidth. In general, at least 25 percent of the optical ports should be used. For systems of three to 26 cabinets, each bundle should contain a multiple of six cables. This configuration rule allows for single chassis granularity of upgrade. For systems of up to 27 to 64 cabinets, each bundle should contain a multiple of two cables.

For example, a 12-cabinet (six-group) system may be initially configured with bundles of 12 optical cables, using 60 ports from the maximum of 240 available. If the system is upgraded to 16 cabinets (eight groups) then new cables bundles are added, connecting the new groups together and connecting each of the existing groups to the two new ones. These cables can be added to those in place, eliminating the need to rewire the system. Support for long optical cables allows for positioning the additional cabinets according to facility requirements.

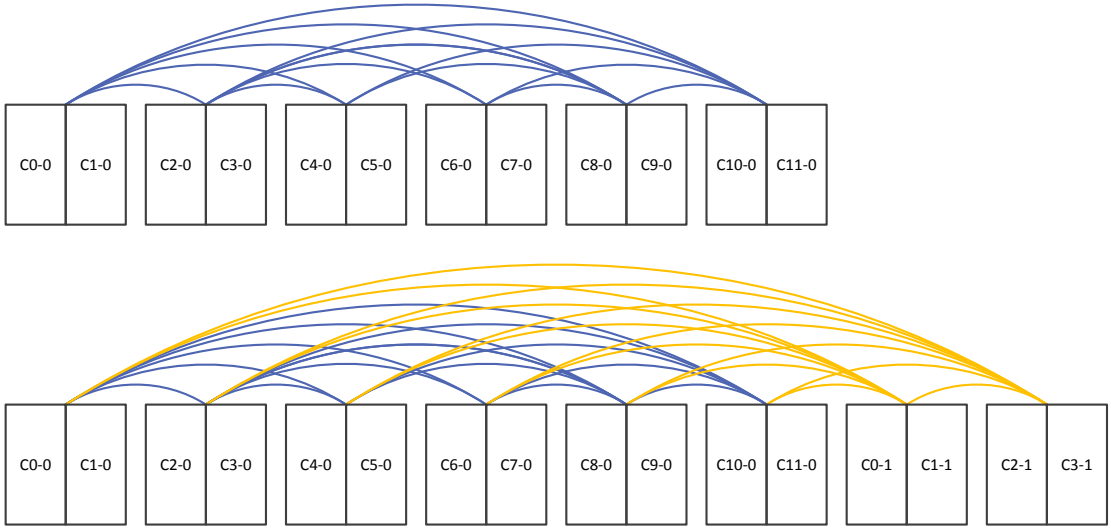

Figure 16: Upgrade from 12 to 16 cabinets. The system is configured with bundles of 12 optical cables connecting each two-cabinet group to each of the other groups. The initial cable mat (shown in blue) remains in place as the system is upgraded. New cable bundles (shown in orange) are used to connect the new groups together and to each of the existing groups.

The number of optical cables required is given by

$$optical\ cables = \frac{1}{2}(inter\text{-}group\ cables\ \times (groups - 1)\ \times groups)$$

Total numbers of optical cables for these configurations are shown in Table 2:

Cabinets	Groups	Inter-group cables	Total cables
12	6	12	180
16	8	12	336

Table 2: Numbers of optical cables required to construct the global network

Bisection bandwidths are determined by the number of cables crossing the worst case bisection, 108 for the 12 cabinet example and 192 for 16 cabinets. Bandwidths of this partial network are shown in Table 3 together with the maximum values for a full optical network.

Cabinets	Groups	Total cables		Cables crossing bisection		Bisection bandwidth (GB/s)	
		Partial	Full	Partial	Full	Partial	Full
12	6	180	720	108	432	4050	16200
16	8	336	952	192	544	7200	20400

Table 3: Bisection bandwidth in GB/s of partial and full 12 and 16 cabinet systems

Small- and medium-size systems (those with 26 cabinets or less) can be upgraded with single chassis granularity. Larger systems are constructed from a whole number of cabinets (to a maximum of 64) or a whole number of two-cabinet groups (above 64 cabinets). All groups except the last must be full. The last group may contain between one and six chassis. These systems are water cooled.

This description focuses on large, water-cooled Cray XC systems. Air-cooled systems can be constructed with a single chassis per cabinet. The backplane provides connectivity between nodes within each chassis. The global network is constructed using electrical cables. Supported configurations range from one to eight cabinets (64 to 512 nodes) with half or fully populated global network. The upgrade granularity is one blade (four nodes).

Conclusion

Research undertaken as part of the Cascade program, together with product development at Cray, has resulted in a single state-of-the-art component – the Aries network ASIC – that powers the communication network of the Cray XC series supercomputer. The Dragonfly network used by the Cray XC series provides cost-effective, scalable global bandwidth. This design together with the Cray Linux Environment and advanced RAS features allows Cray XC systems to scale to millions of cores. The Cray XC programming environment enables efficient use of these resources on the most demanding HPC applications.

References

[1] J.Kim, W.J. Dally, S.Scott and D.Abts. "Technology-Driven, Highly-Scalable Dragonfly Topology" In Proc. of the International Symposium on Computer Architecture (ISCA), pages 77-88, 2008.

[2] R. Brightwell, K. Predretti, K. Underwood, and T. Hudson.Seastar Interconnect: Balanced Bandwidth for Scalable Performance. IEEE Micro, 26(3) pages 41–57, 2006.

[3] The Gemini System Interconnect. Bob Alverson, Duncan Roweth and Larry Kaplan, Cray Inc. High-Performance Interconnects Symposium, pages 83–87, 2010.

[5] S. Scott, D. Abts, J. Kim, and W. J. Dally. "The BlackWidow High-radix Clos Network" In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 16–28, 2006.

[6] A scheme for fast parallel communication. L. G. Valiant. SIAM Journal on Computing, 11(2) pages 350–361, 1982.

[7] H. Pritchard, I. Gorodetsky, and D. Buntinas. A uGNI based MPICH2 Nemesis Network Module for the Cray XE. In Proceedings of the 18th European MPI Users' Group Conference on Recent Advances in the Message Passing Interface, EuroMPI'11, pages 110–119, Springer-Verlag, 2011.

[8] C. Leisserson, "Fat-trees: Universal networks for hardware efficient supercomputing", IEEE Transactions on computer, C-34(10), pages 892-901, October 1985.

[9] Cray Research, Inc. SHMEM Technical Note for C, SG-2516 2.3, 1994.

[10] Intel IMB Benchmarks www.intel.com/software/imb

[11] Ohio State University MPI Micro-Benchmarks. http://mvapich.cse.ohio-state.edu/benchmarks

[12] Co-arrays in the next Fortran Standard, Numrich and Reid, ACM Fortran Forum, 2005, volume 24, no 2, pp 4-17.

[13] UPC: Distributed Shared Memory Programming"; Tarek El-Ghazawi, William Carlson, Thomas Sterling, and Katherine Yelick; ISBN: 0-471-22048-5; Published by John Wiley and Sons - May, 2005.

[14] Parallel Programmability and the Chapel Language Bradford L. Chamberlain, David Callahan, Hans P. Zima. International Journal of High Performance Computing Applications, August 2007, 21(3): 291-312.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0001. Only a small number of the many Cray employees that developed the Cray XC system were involved in the preparation of this paper. The authors would like to acknowledge the enormous effort made by the entire team.