The Cray XT4 and Seastar 3-D Torus Interconnect

April 6, 2010

BYLINE

Dennis Abts
dabts@google.com
Google Inc.
Madison, WI
USA
dabts@google.com

SYNONYMS

Cray Red Storm, Cray XT3, Cray XT4, Cray XT5, Cray XT, Interconnection Networks

DEFINITION

The Cray XT4 system is a distributed memory multiprocessor combining an aggressive superscalar processor (AMD64) with a bandwidth-rich 3-D torus interconnection network that scales up to 32K processing nodes. This chapter provides an overview of the Cray XT4 system architecture and a detailed discussion of its interconnection network.

DISCUSSION

The physical sciences are increasingly turning toward computational techniques as an alternative to the traditional "wet lab" or destructive testing environments for experimentation. In particular, computational sciences can be used to *scale* far beyond that of traditional experimental methodologies; opening the door to large-scale climatology and molecular dynamics, for example, which encompass enough detail to accurately model the dominant terms that characterize the physical phenomena being studied [2]. These large-scale applications require careful orchestration among cooperating processors to ply these computational techniques effectively.

The genesis of the Cray XT4 system was the collaborative design and deployment of the Sandia "Red Storm" computer which provided the computational power necessary to assure safeguards under the nuclear Stockpile Stewardship Program which seeks to maintain and verify a nuclear weapons arsenal without the use of testing. It was later renamed the Cray XT3 and sold commercially in configurations varying from hundreds of processors, to 10s of thousands of processors. An improved processor, faster processor-network interface, along with further optimizations to the software stack and migrating to a lightweight Linux kernel prompted the introduction of the Cray XT4; however, the underlying system architecture and interconnection network remained unchanged.

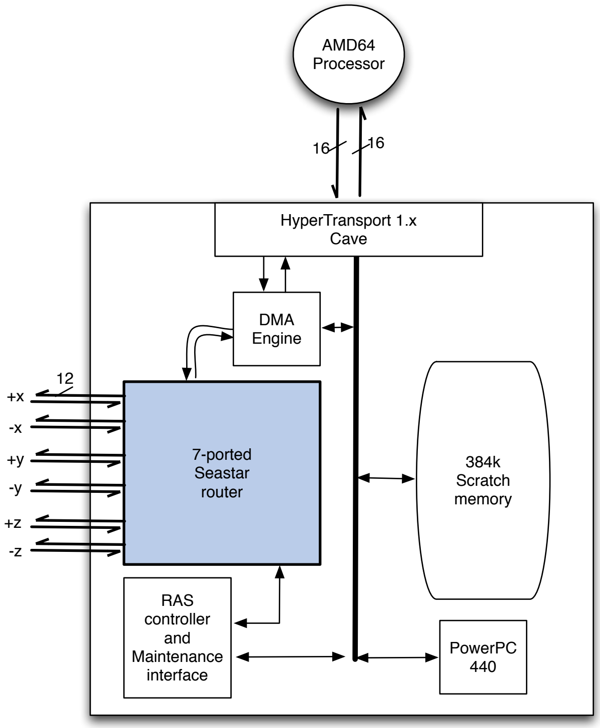


Figure 1: High level block diagram of the Seastar interconnect chip.

System Overview

The Cray XT4 system scales up to 32k nodes using a bidirectional three-dimensional torus interconnection network. Each *node* in the system consists of an AMD64 superscalar processor connected to a Cray Seastar chip [5] (Figure 1) which provides the processor-network interface, and 6-ported router for interconnecting the nodes. The system supports an efficient distributed memory message passing programming model. The underlying message transport is handled by the Portals [3] messaging interface.

This chapter focuses on the Cray XT interconnection network which has several key features that set it apart from other networks:

- scales up to 32K network endpoints,
- high injection bandwidth using HypterTransport (HT) links directly to the network interface,
- reliable link-level packet delivery,
- multiple virtual channels for both deadlock avoidance and performance isolation, and
- age-based arbitration to provide fair access to network resources.

Subsequent sections cover these topics in more detail.

There are two types of nodes in the Cray XT system. Endpoints (nodes) in the system are either *compute* or *system and IO (SIO)* nodes. SIO nodes are where user's login to the system and compile/launch applications.

Topology

The Cray XT interconnect can be configured as either a *k*-ary *n*-mesh or *k*-ary *n*-cube (torus) topology. As a torus, the system is implemented as a *folded* torus to reduce the cable length of the wrap around link. The 7-ported Seastar router provides a processor port, and six network ports corresponding to +x, -x, +y, -y, +z, and -z directions. The port assignment for network links is not fixed, any port can correspond to any of the six directions. The non-coherent HyperTransport (HT) protocol provides a low latency, point-to-point channel used to drive the Seastar network interface.

Four virtual channels are used to provide point-to-point flow control and deadlock avoidance. Using virtual channels avoids unnecessary head-of-line (HoL) blocking for different network traffic flows, however, the extent to which virtual channels improve network utilization depends on the distribution of packets among the virtual channels.

Routing

The routing rules for the Cray XT are subject to several constraints. Foremost, the network must provide error-free transmission of each packet from the *source* node identifier (NID) to the *destination*. To accomplish this, the distributed *table-driven* routing algorithm is implemented with a dedicated *routing table* at each input port that is used to lookup the destination port and virtual channel of the incoming packet. The lookup table at each input port is not sized to cover the maximum 32K node network since most systems will be much smaller, only a few thousand nodes. Instead, a hierarchical routing scheme divides the node name space into *global* and *local* regions. The upper three bits of the destination field (given by the destination[14:12] in the packet header) of the incoming packet are compared to the global partition of the current SeaStar router. If the global partition does not match, then the packet is routed to the output port specified in the global lookup table (GLUT). The GLUT is indexed by destination[14:12] to choose one of eight global partitions. Once the packet arrives at the correct global region, it will precisely route within a local partition of 4096 nodes given by the destination[11:0] field in the packet header.

The tables must be constructed to avoid deadlocks. Glass and Ni [9] describe *turn* cycles that can occur in *k*-ary *n*-cube networks. However, torus networks are also susceptible to deadlock that results from overlapping virtual channel dependencies (this only applies to *k*-ary *n*-cubes, where $k > 4$) as described by Dally and Seitz [7]. Additionally, the SeaStar router does not allow 180 degree turns within the network. The routing algorithm must both provide deadlock-freedom and achieve good performance on benign traffic. In a fault-free network, a straightforward dimension-ordered routing (DOR) algorithm will provide balanced traffic across the network links. Although, in practice, faulty links will occur and the routing algorithm must route around the bad link in a way that preserves deadlock freedom and attempts to balance the load across the physical links. Furthermore, it is important to optimize the buffer space within the SeaStar router by balancing the number of packets within each virtual channel.

Avoiding deadlock in the presence of faults and turn constraints

The routing algorithm rests upon a set of rules to prevent deadlock. In the turn model, a positive first (x+, y+, z+ then x-, y-, z-) rule prevents deadlock and allows some routing options to avoid faulty links or nodes. The global/local routing table adds an additional constraint for valid turns. Packets must be able to travel to their local area of the destination without the deadlock rule preventing free movement within the local area. In the Cray XT network the localities are split with yz planes. To allow both x+ and x- movement without restricting later directions, the deadlock avoidance rule is modified to (x+, x-, y+, z+ then y+, y-, z+ then z+, z-). Thus, free movement is preserved. Note that missing or broken X links may induce a non-minimal route when a packet is routed via the global table (since only y+ and z+ are “safe”). With this rule, packets using the global table will prefer to move in the X direction, to get to their correct global region as quickly as possible. In the absence of any broken links, routes between compute nodes can be generated by moving in x dimension, then y, then z. Also, when y=Y_{max}, it is permissible to dodge y- then go x+/x-. If the dimension is configured as a mesh — there are no y+ links, for example, anywhere at y=Y_{max} then a deadlock cycle is not possible.

In the presence of a faulty link, the deadlock avoidance strategy depends on the direction prescribed by dimension order routing for a given destination. In addition, toroidal networks add *dateline* restrictions. Once a dateline is crossed in a given dimension, routing in a higher dimension (e.g. X is "higher" than Y) is not permitted.

Routing rules for X links

When x+ or x- is desired, but that link is broken, y+ is taken if available. This handles crossing from compute nodes to service nodes, where some X links are not present. If y+ is not available, z+ is taken. This z+ link must not cross a dateline. To avoid this, the dateline in Z is chosen so that there are no nodes with a broken X link and a broken y+ link. Although the desired X link is available, the routing algorithm may choose to take an alternate path when the node at the other side of the X link has a broken y+ and z+ link (note the y+ might not be present if configured as a mesh), then an early detour toward z+ is considered. If the X link crosses a partition boundary into the destination partition or the current partition matches the destination partition and the current Y matches the destination Y coordinate, route in z+ instead. Otherwise, the packet might be boxed in at the next node, with no safe way out.

Routing rules for Y links

When the desired route follows a Y link that is broken, the preference is to travel in z+ to find a good Y link. If z+ is also broken, it is feasible to travel in the opposite direction in the Y dimension. However, the routing in the node in that direction must now look ahead to avoid a 180 degree turn if it were to direct a packet to the node with the faulty links. When the desired Y link is available, it is necessary to check that the node at that next hop does not have a z+ link that the packet might prefer (based on XYZ routing) to follow next. That is, if the default direction for this destination in the next node is z+ and the z+ link is broken there, the routing choice at this node would be changed from the default Y link to z+.

Routing rules for Z links

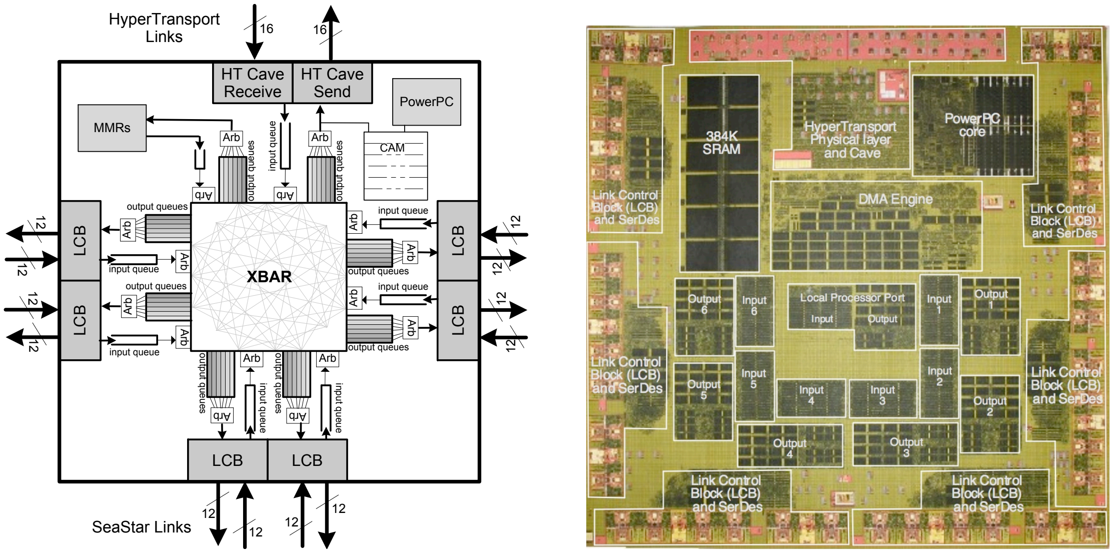
When the desired route follows a z+ link that is broken, the preference is to travel in y+ to find a good z+ link. In this scenario, the Y link look ahead is relied up to avoid the node at y+ from sending the packet right back along y-. When the y+ link is not present (at the edge of the mesh), the second choice is y-. When the desired route is to travel in the z- direction, the logic must follow the z- path to ensure there are no broken links at all on the path to the final destination. If one is found, the route is forced to z+, effectively forcing the packet to go the long way around the Z torus.

Flow Control

Buffer resources are managed using credit-based flow control at the data-link level. The link control block (LCB) is shown at the periphery of the Seastar router chip in Figure 2. Packets flow across the network links using virtual cut-through flow control — that is, a packet does not start to flow until there is sufficient space in the receiving input buffer. Each virtual channel (VC) has dedicated buffer space. A 3-bit field (Figure 3) in each flit is used to designate the virtual channel, with a value of all 1's representing an *idle* flit. Idle flits are used to maintain byte and lane alignment across the plesiochronous channel. They can also carry VC credit information back to the sender.

SeaStar Router Microarchitecture

Network packets are comprised of one or more 68-bit *flits* (flow control units). The first flit of the packet (Figure 3) is the *header* flit and contains all the necessary routing fields (destination[14:0], age[10:0],



(a) Seastar block diagram.

(b) Seastar die photo.

Figure 2: Block diagram of the Seastar system chip.

vc[2:0]) as well as a tail (t) bit to mark the end of a packet. Since most XT networks are on the order of several thousand nodes, the lookup table at each input port is not sized to cover the maximum 32k node network. To make the routing mechanism more space-efficient, the 15-bit node identifier is partitioned to allow a two-level hierarchical lookup: a small 8-entry table identifies a *region*, the second table precisely identifies the node within the region. The region table is indexed by the upper 3-bits of the *destination* field of the packet, and the low-order 12-bits identifies the node within 4k-entry table. Each network port has a dedicated routing table and is capable of routing a packet each cycle. This provides the necessary lookup bandwidth to route a new packet every cycle. However, if each input port used a 32k-entry lookup table, it would be sparsely populated for modest-sized systems, and use an extravegent amount of silicon area.

67	66 65 64	63 62 61 60 59 58 57 56 55 54 53 52 51 50 49	48 47	46	45	44 43 42 41	40	39 38 37 36 35 34 33 32 31 30 29 28	27 26 25 24 23 22 21 20	19	18 17 16 15 14 13 12	11	10 9 8 7 6 5 4 3 2 1 0
t	vc	destination[14:0]	dt	k	V	Length	S	TransactionID[11:0]	source[14:7]	R	source[6:0]	u	Age[10:0]
t	vc	Data[63:0]											
… up to 8 data flits (64 bytes) of payload …													
t	vc	Data[63:0]											

Figure 3: Seastar packet format.

A two-level hierarchical routing scheme is used to efficiently lookup the egress port at each router. Each router is assigned a unique node identifier, corresponding to its destination address. Upon arrival at the input port, the packet destination field is compared to the node identifier. If the upper three bits of the destination address match the upper three bits of the node identifier, then the packet is in the correct *global partition*. Otherwise, the upper three bits are used to index into the 8-entry *global lookup table* (GLUT) to determine the egress port. Conceptually, the 32k possible destinations are split into eight, 4k partitions denoted by bits `destination[11:0]` of the destination field.

The SeaStar router has six full-duplex network ports and one processor port that interfaces with the Tx/Rx DMA engine (Figure 2). The network channels operate at 3.2 Gb/s ×12 lanes over electrical wires, providing a peak of 4.8 GB/s per direction of network bandwidth. The link control block (LCB) imple-

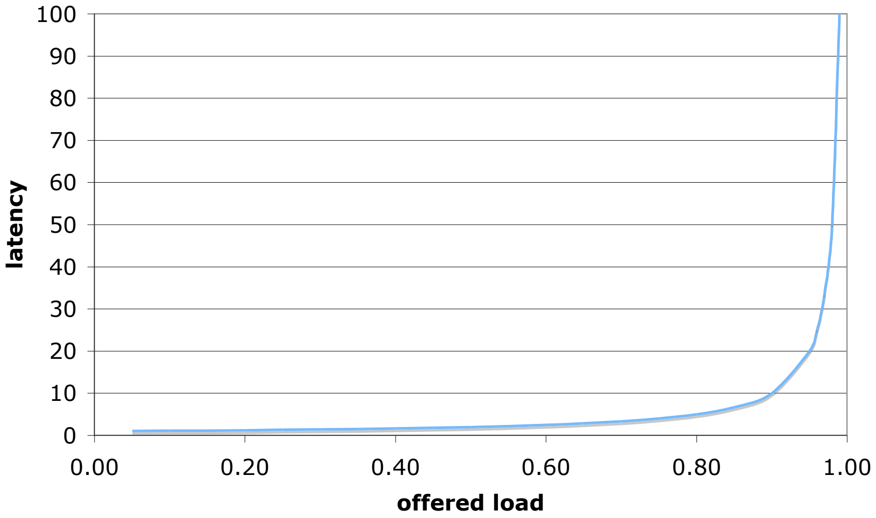


Figure 4: Offered load versus latency for an ideal M/D/1 queue model.

ments a sliding window go-back-N link-layer protocol that provides reliable chip-to-chip communication over the network links. The router switch is both input-queued and output-queued. Each input port has four (one for each virtual channel) 96-entry buffers, with each entry storing one flit. The input buffer is sized to cover the round-trip latency across the network link at 3.2 Gb/s signal rates. There are 24 staging buffers in front of each output port, one for each input source (five network ports, and one processor port), each with four VCs. The staging buffers are only 16 entries deep and are sized to cover the crossbar arbitration round-trip latency. Virtual cut-through [11] flow control into the output staging buffers requires them to be at least 9 entries deep to cover the maximum packet size.

Age-based output arbitration

Packet latency is divided into two components: *queueing* and *router* latency. The total delay (T) of a packet through the network with H hops is the sum of the queueing and router delay.

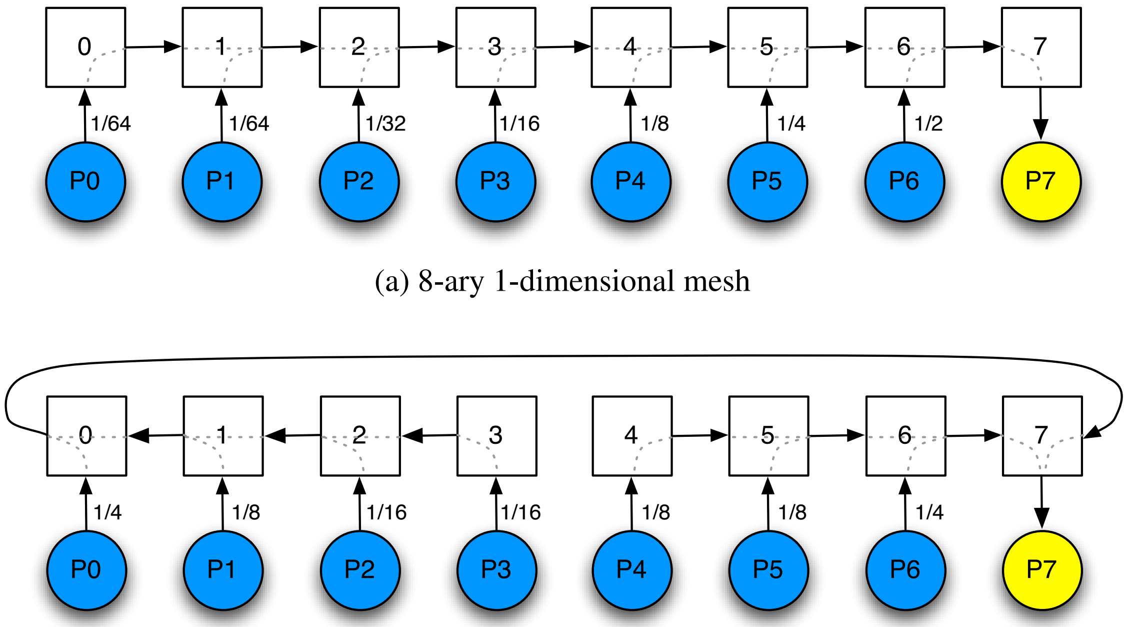
$$T = HQ(\lambda) + Ht_r \tag{1}$$

where t_r is the per-hop router delay (which is $\approx$ 50 ns for the Seastar router). The queueing delay, $Q(\lambda)$, is a function of the offered load (λ) and described by the latency-bandwidth characteristics of the network. An approximation of $Q(\lambda)$ is given by an M/D/1 queue model (Figure 4).

$$Q(\lambda) = \frac{1}{1-\lambda} \tag{2}$$

When there is very low offered load on the network, the $Q(\lambda)$ delay is negligible. However, as traffic intensity increases, and the network approaches saturation, the queueing delay will dominate the total packet latency.

As traffic flows through the network it merges with newly injected packets and traffic from other directions in the network (Figure 5). This merging of traffic from different sources causes packets that have further to travel (more hops) to receive geometrically less bandwidth. For example, consider the 8-ary 1-mesh in Figure 5(a) where processors P0 thru P6 are sending to P7. The switch allocates the output port by granting packets fairly among the input ports. With a round-robin packet arbitration policy, the processor closest to the destination (P6 is only one hop away) will get the most bandwidth — 1/2 of the



(a) 8-ary 1-dimensional mesh

(b) 8-ary 1-dimensional torus

Figure 5: All nodes are sending to P7 and merging traffic at each hop.

available bandwidth. The processor two hops away, P5, will get half of the bandwidth into router node 6, for a total of 1/2×1/2 = 1/4 of the available bandwidth. That is, every two arbitration cycles node 7 will deliver a packet from source P6, and every four arbitration cycles it will deliver a packet from source P5. A packet will merge with traffic from at most $2n$ other ports since each router has $2n$ network ports with $2n-1$ from *other* directions and one from the processor port. In the worst case, a packet traveling H hops and merging with traffic from $2n$ other input ports, will have a latency of:

$$T_{worst} = \frac{L}{(2n)^H} \tag{3}$$

where L is the length of the message (number of packets), and n is the number of dimensions. In this example, P0 and P1 each receive 1/64 of the available bandwidth into node 7, a factor of 32 times less than that of P6. Reducing the variation in bandwidth is critical for application performance, particularly as applications are scaled to increasingly higher processor counts. Topologies with a lower diameter will reduce the impact of merging traffic. A torus is less affected than a mesh of the same radix (Figure 5a and 5b), for example, since it has a lower diameter. With dimension-order routing (DOR), once a packet starts flowing on a given dimension it stays on that dimension until it reaches the ordinate of its destination.

Key parameters associated with age-based arbitration

The Cray XT network provides *age-based arbitration* to mitigate the affects of this traffic merging as shown in Figure 5, thus reducing the variation in packet delivery time. However, age-based arbitration can introduce a starvation scenario whereby *younger* packets are starved at the output port and cannot make forward progress toward the destination. The details of the algorithm along with performance results are given by Abts and Weisser [1]. There are three key parameters for controlling the aging algorithm.

- `AGE_CLOCK_PERIOD` – a chip-wide 32-bit countdown timer that controls the *rate* at which packets age. If the age rate is too slow, it will appear as though packets are not accruing any queueing delay, their ages will not change, and all packets will appear to have the same age. On the other hand, if the age rate is too fast, packets ages will saturate very quickly — perhaps after only a few hops — at the maximum age of 255, and packets will not generally be distinguishable by age. The resolution of `AGE_CLOCK_PERIOD` allows anywhere from 2 nanoseconds to more than 8 seconds of queueing delay to be accrued before the age value is incremented.

- `REQ_AGE_BIAS` and `RSP_AGE_BIAS` – each hop that a packet takes increments the packet age by the `REQ_AGE_BIAS` if the packet arrived on VC0/VC1 or by `RSP_AGE_BIAS` if the packet arrived on VC2/VC3. The age *bias* fields are configurable on a per-port basis, with the default bias of 1.
- `AGE_RR_SELECT` – a 64-bit array specifying the output arbitration policy. A value of all 0s will select *round-robin* arbitration, and a value of all 1s will select *age-based* arbitration. A combination of 0s and 1s will control the ratio of round-robin to age-based. For example, a value of 0101···0101 will use half round-robin and half age-based.

When a packet arrives at the head of the input queue, it undergoes routing by indexing into the LUT with destination[11:0] to choose the target port and virtual channel. Since each input port and VC has a dedicated buffer at the output staging buffer, there is no arbitration necessary to allocate the staging buffer — only flow control. At the output port, arbitration is performed on a per-packet basis (not per flit, as wormhole flow control would). Each output port is allocated by performing a 4-to-1 VC arbitration along with a 7-to-1 arbitration to select among the input ports. Each output port maintains two *independent* arbitration pointers — one for round-robin and one for age-based. A 6-bit counter is incremented on each *grant* cycle and indexes into the `AGE_RR_SELECT` bit array to choose the per-packet arbitration policy.

RELATED ENTRIES

Clusters

DE Shaw's Anton

Distributed-Memory Multiprocessors (Survey)

Infiniband

Massively Parallel Distributed-Memory Machines (Survey)

Petascale Computing

System on a Chip

Top500 LIst

BIBLIOGRAPHIC NOTES AND FURTHER READING

The genesis of the Cray XT4 system was the collaborative design and deployment of the Sandia "Red Storm" computer which provided the computational power necessary to assure safeguards under the nuclear Stockpile Stewardship Program which seeks to maintain and verify a nuclear weapons arsenal without the use of testing. Brightwell, Pedretti, and Underwood [4] provide an early look at the Seastar interconnection network used by the Sandia Red Storm supercomputer.

Hoisie, Johnson, Kerbyson, Lang, and Pakin [10] use common high-performance computing (HPC) benchmarks and modeling to compare performance of three leading supercomputers: the Cray XT (Red Storm), IBM BlueGene/L, and ASCI Purple supercomputers.

Dally presents a performance analysis of *k*-ary *n*-cube networks [6]. While a more comprehensive analysis, with several examples from industry, is found in Dally and Towles [8].

BIBLIOGRAPHY

References

[1] D. Abts and D. Weisser. Age-based packet arbitration in large-radix k-ary n-cubes. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–11, New York, NY, USA, 2007. ACM.

[2] S. R. Alam, J. A. Kuehn, R. F. Barrett, J. M. Larkin, M. R. Fahey, R. Sankaran, and P. H. Worley. Cray xt4: an early evaluation for petascale scientific simulation. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.

[3] R. Brightwell, B. Lawry, A. B. MacCabe, and R. Riesen. Portals 3.0: Protocol building blocks for low overhead communication. In *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 268, Washington, DC, USA, 2002. IEEE Computer Society.

[4] R. Brightwell, K. Pedretti, and K. D. Underwood. Initial performance evaluation of the cray seastar interconnect. In *HOTI '05: Proceedings of the 13th Symposium on High Performance Interconnects*, pages 51–57, Washington, DC, USA, 2005. IEEE Computer Society.

[5] R. Brightwell, K. T. Pedretti, K. D. Underwood, and T. Hudson. Seastar interconnect: Balanced bandwidth for scalable performance. *IEEE Micro*, 26(3):41–57, 2006.

[6] W. J. Dally. Performance Analysis of k-ary n-cube Interconnection Networks. *IEEE Transactions on Computers*, 39(6):775–785, 1990.

[7] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput.*, 36(5):547–553, 1987.

[8] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, San Francisco, CA, 2004.

[9] C. J. Glass and L. M. Ni. The turn model for adaptive routing. In *ISCA '92: Proceedings of the 19th annual international symposium on Computer architecture*, pages 278–287, 1992.

[10] A. Hoisie, G. Johnson, D. J. Kerbyson, M. Lang, and S. Pakin. A performance comparison through benchmarking and modeling of three leading supercomputers: blue gene/l, red storm, and purple. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 74, New York, NY, USA, 2006. ACM.

[11] P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.