Scalable Collective Communication on the ASCI Q Machine*

Fabrizio Petrini Juan Fernandez Eitan Frachtenberg
Salvador Coll
CCS-3 Modeling, Algorithms, and Informatics Group
Computer and Computational Sciences (CCS) Division
Los Alamos National Laboratory, Los Alamos, NM 87545 USA
{fabrizio,juanf,eitanf,scoll}@lanl.gov

Abstract

Scientific codes spend a considerable part of their run time executing collective communication operations. Such operations can also be critical for efficient resource management in large-scale machines. Therefore, scalable collective communication is a key factor to achieve good performance in large-scale parallel computers.

In this paper we describe the performance and scalability of some common collective communication patterns on the ASCI Q machine. Experimental results conducted on a 1024-node/4096-processor segment show that the network is fast and scalable. The network is able to barrier-synchronize in a few tens of μs, perform a broadcast with an aggregate bandwidth of more than 100 GB/s and sustain heavy hot-spot traffic with a limited performance degradation.

1. Introduction

The efficient implementation of collective communication patterns in a parallel machine is an important factor to achieve good application performance. Recent analytical [4] and experimental [7] studies conducted on ASCI-class machines have shown that scientific codes spend a considerable part of their run time executing collective communication, in some cases up to 70%. Unfortunately, there are very few results in the literature that provide an experimental evaluation of the performance and scalability of the collective communication patterns on large-scale machines. In fact, due to their high costs, these machines enter in production mode very quickly, leaving very little time for performance evaluation and optimization.

Another issue that has often been neglected is the performance and scalability of the system software on large-scale machines. As we demonstrated in [3], some collective communication operations can be used as building blocks for high-performance, scalable resource management. For example, our prototype resource-management system, STORM, relies on two collective communication operations: multicast and global comparison to perform most of its tasks. When using a cluster of similar architecture to that of ASCI Q, STORM can exploit collectives to perform tasks such as job launching and context switching in sub-second times, even on thousands of nodes.

Some typical uses of several collectives operations are shown in Table 1. Due to their prevalence and importance, an efficient and scalable implementation is critical to large-scale machines. In this paper we will describe how collective communication performs and scales on the ASCI Q machine (Figure 1), the second largest supercomputer in the world at the time of this writing.[1] The paper provides a quick overview on the Quadrics network, the backbone of the ASCI Q machine, to bring the reader up to speed with its relevant features. We describe the network architecture and topology of the ASCI Q machine and the main mechanisms that are at the base of several collective communication patterns. In the final part of the paper, we report an extensive performance evaluation of the most common patterns, barrier, broadcast, allreduce and hot spot on one of the two ASCI-Q clusters, containing 1024 nodes with 4 processors each.

2. Quadrics Network

The Quadrics network [6] is based on two building blocks, a programmable network interface called Elan [10] and a communication switch called Elite [11]. Elite switches can be interconnected in a fat-tree topology [5].

*This work is supported by the U.S. Department of Energy through Los Alamos National Laboratory contract W-7405-ENG-36

[1] See http://www.top500.org for more information.

Proceedings of the 11 th Symposium on High Performance Interconnects (HOTI'03)
0-7695-2012-X/03 $17.00 © 2003 IEEE

Table 1. Collective communication operation examples

Operations	Typical usage
Multicast	Applications: Dissemination of data from a single source Resource management: Binary and data dissemination; job coscheduling
Barrier, global comparison	Applications: Process synchronization Resource management: Heartbeats, fault detection
Allreduce, Scatter-Gather	Exchange and processing of multi-source data

Figure 1. The ASCI Q machine at Los Alamos National Laboratory

The network has several layers of communication libraries which provide trade-offs between performance and ease of use. Other important features are hardware support for collective communication patterns and fault-tolerance.

The Elan network interface links the high-performance, multi-stage Quadrics network to a processing node containing one or more CPUs. In addition to generating and accepting packets to and from the network, the Elan also provides substantial local processing power to implement communication protocols.

The other building block of the Quadrics network is the Elite switch. The Elite provides the following features: (1) 8 bidirectional links supporting two virtual channels in each direction, (2) an internal full-crossbar switch, (3) a nominal transmission bandwidth of 400 MB/s on each link direction and a flow-through latency of 35 ns, (4) packet error detection and recovery, with routing and data transactions CRC protected, (5) two priority levels combined with an aging mechanism to ensure a fair delivery of packets in the same priority level, (6) hardware support for broadcasts, (7) and adaptive routing.

3. Network Topology

The Elite switches are typically interconnected in a quaternary fat-tree topology, which belongs to the more general class of the k-ary n-trees [8]. A quaternary fat-tree of dimension n is composed of 4^n processing nodes and $n * 4^{n-1}$ switches interconnected as a delta network, and can be recursively build by connecting 4 quaternary fat trees of dimension $n - 1$.

The Elite switches can be assembled using a limited set of packages. The main building block is a board with 8 Elite switches that implements a two-level fat tree with 16 up and 16 down connections. Four of these boards and a backplane can be configured as a switch with 64 up and 64 down connections.

The network topology of the ASCI Q machine is outlined in Figure 2. At the time of writing, the computer is divided into two clusters, QA and QB, each one with 1024 nodes/4096 processors, for a total of 2048 nodes/8192 processors. Each segment is a full fat tree of dimension five and is organized in two levels of switches. The bottom level has 16 switches (labeled as QA/B-B[0-15] in Figure 2) with 64 down links to nodes and 64 up links to top-level switches. The top level has 64 switches (labeled as QA/B-T[0-63] in Figure 2) with 16 up and 16 down connections. Each of the 16 down connections is connected to a distinct bottom switch. The set of switches at the top, called *supertop*, connects both segments and is expected to have a reduced level of connectivity. Within each segment, the network maintains a bisection bandwidth that is linear with the number of processing nodes. Thanks to the wormhole flow control and to the low latency of the Elite switches, the point-to-point latency between any pair of nodes is almost constant and practically insensitive to the network distance.

4. Hardware and Software Support for Collective Communication

The Quadrics network provides hardware support in both the network interface and the switches to implement

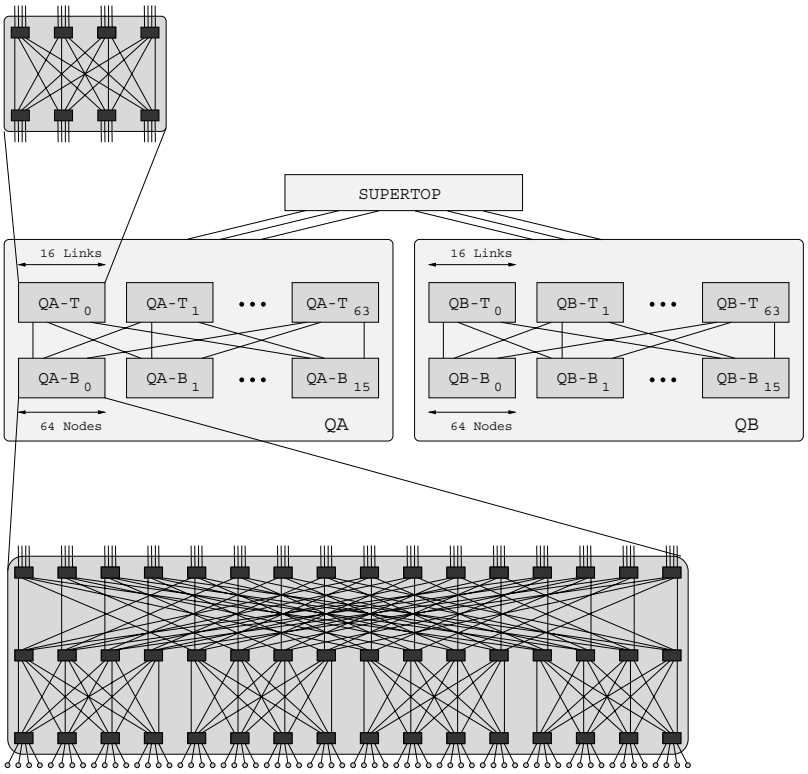


Figure 2. The network topology of the ASCI Q machine and its building blocks

scalable collective communication. Multicast packets can be sent to multiple destinations using either the *hardware* multicast capability of the network or a *software* tree implemented with point-to-point messages exchanged by the Elans without interrupting their processing nodes. These mechanisms constitute the basic blocks to implement collective communication patterns such as barrier synchronization, broadcast, reduce and allreduce.

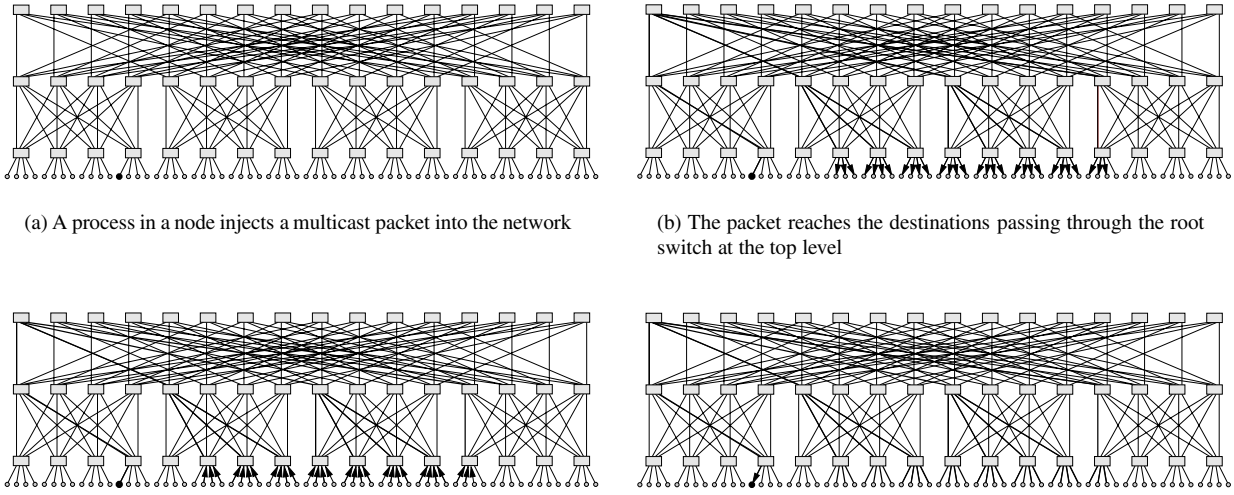
4.1. Hardware Multicast

The implementation of the hardware multicast is outlined in Figure 3. A process in a node injects a multicast packet into the network (see Figure 3(a)). This packet can only take a predetermined ascending path in order to avoid deadlocks. By default, the top leftmost switch is chosen as the logical root for collective communications, and every request, in the ascending phase, must pass through the root switch. In Figure 3(b) the packet reaches the root switch, and then multiple branches are propagated in parallel. If another collective communication is issued while the first one is still in progress, it is serialized at or before the root switch. The second communication will be able to proceed when all the circuits of the first one are cleared. All nodes are capable of receiving a multicast packet, as long as the multicast set is physically contiguous.

For a multicast packet to be successfully delivered, a positive acknowledgment must be received from all the recipients of the multicast group. The Elite switches combine the acknowledgments, as pioneered by the NYU Ultracomputer [9], returning a single one to the source (see Figures 3(c) and 3(d)). Acknowledgments are combined in a way that the 'worst' ack wins (a network error wins over an unsuccessful transaction, which on its turn wins over a successful one), returning a positive ack only when all the partners in the collective communication complete the distributed transaction with success. The network hardware guarantees the atomic execution of the multicast: either all nodes successfully complete the operation or none. It is worth noting that the multicast packet opens a set of circuits from the source to the destination set, and that multiple transactions (up to 16 in the current implementation) can be pipelined within a single packet. For example, it is possible to conditionally issue a transaction based on the result of a previous transaction. This powerful mechanism allows the efficient implementation of sophisticated collective operations and high-level protocols [2, 3].

4.2. Software Tree

In the current implementation of the hardware multicast, the destination set must be physically contiguous. If this is



(a) A process in a node injects a multicast packet into the network

(b) The packet reaches the destinations passing through the root switch at the top level

(c) The acknowledgments are combined

(d) The issuing process receives a single acknowledgment

Figure 3. Hardware multicast

not the case, or there exists the need to execute more complex protocols that require fast packet processing, it is possible to take advantage of the thread processor in the Elan. The Elan thread processor can receive an incoming packet, perform some basic processing (e.g., an atomic increment of a variable) and send one or more replies in a few μs, without any interaction with the main processor.

Software collectives can be implemented using the communication and computation capabilities of the Elan thread processor, e.g., software multicast trees. Software collectives can be based on trees with programmable arity, depth and regularity, and do not suffer from the limitation that the destination set must be composed of adjacent nodes.

4.3. Barrier Synchronization

A barrier synchronization is a logical point in the control flow of a parallel program at which all processes in the group must arrive before any of the them is allowed to proceed. Typically, a barrier synchronization involves a logical reduce operation followed by a broadcast.

If all the nodes in the barrier synchronization set are contiguous, it is possible to use the hardware multicast. When the barrier is performed, all processes in the group write a barrier sequence number in a memory location and wait for a global 'go' signal (e.g., polling on a memory location). The master process within the root node (the one with the lowest id) uses an Elan thread to send a special test-and-set multicast packet. This packet spans all the processes and checks if the barrier sequence value in each process matches with its own sequence number (it does if the corresponding process reached the barrier). All the replies are then combined by the Elite switches on the way back to the root node which receives a single acknowledgment. If all the nodes are ready, an end-of-packet token is sent to the group to set an event or write a word to wake up the processes waiting in the barrier. This mechanism is completely integrated into the network flow control and is expected to give the best performance when processes enter the barrier fairly close together, otherwise it backs off exponentially (to stop flooding the network with test-and-set multicast packets).

The software algorithm based on point-to-point messages uses a balanced tree to send a 'ready' signal to the process with lowest id. Each process in the tree waits for 'ready' signals from its children, and when it receives all of them sends its own signal up to the parent process. This phase of the barrier is illustrated in Figure 4. When the root process receives all its 'ready' signals, it performs a broadcast using point-to-point messages to send the 'go' signal with the same tree structure.

4.4. Broadcast

The broadcast operation consists of a root node sending a chunk of data to multiple destinations. The user-level broadcast can take advantage of the hardware multicast. The current implementation requires the destination buffer to be at the same virtual address in all destinations. In the more general case where there are multiple destination processes on the same node, the original message is initially sent into a piece of shared memory, and the destination processes makes private copies using a double buffering technique.

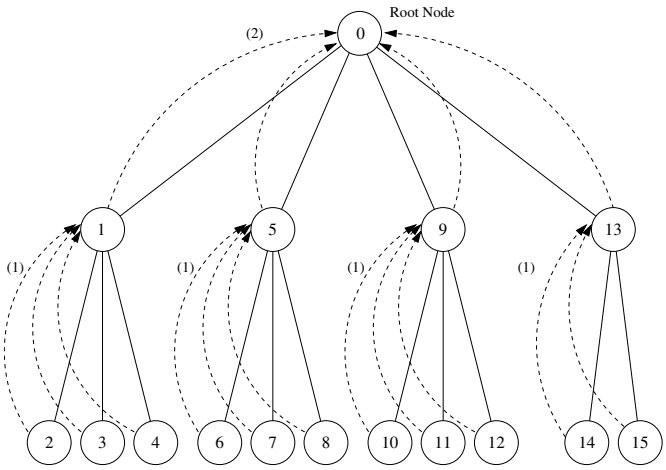


Figure 4. A software tree implemented with point-to-point messages between Elan network interfaces

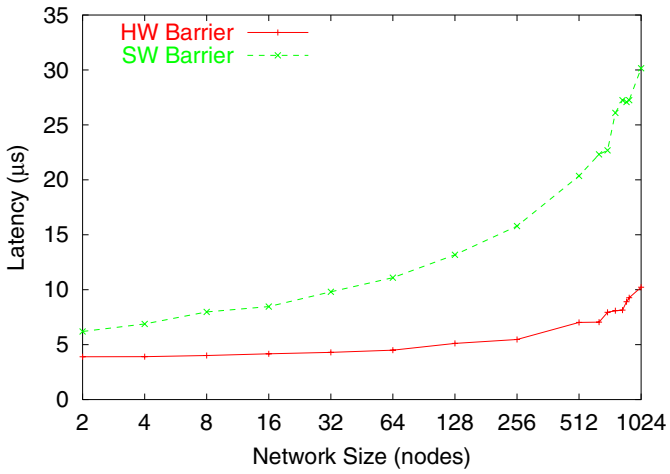


Figure 5. Barrier synchronization

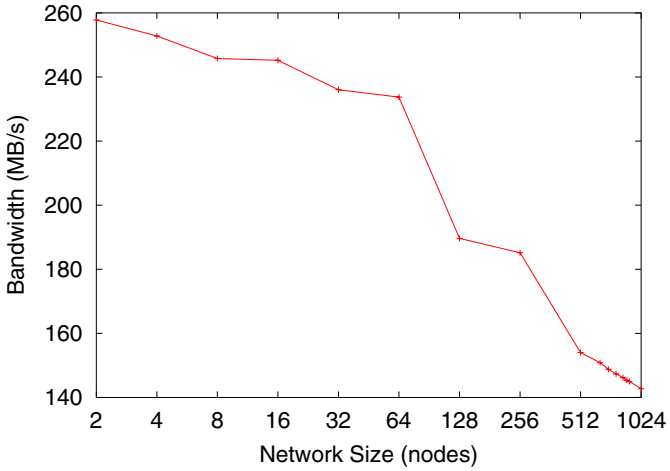


Figure 6. Broadcast

4.5. Allreduce

A common collective communication pattern in many scientific codes is the allreduce. The allreduce computes an arithmetic operation on all the elements of a vector (e.g., a floating-point sum) distributed on a set of processes, collects the result and updates another vector of the same size with the final result on all processes. The allreduce is logically organized into two phases: the collection phase and the distribution phase. The collection phase can be implemented with point-to-point messages. Given that the semantics of the operation require an arithmetic operation, every time a message is received the host processor must intervene. For the distribution phase, we can use the broadcast algorithms.

4.6. Hot Spot

One of the most demanding types of communication is the hot spot. All nodes send messages to a single node, which becomes the hot spot. This pattern is representative of system network traffic (e.g. a metadata server in a parallel file system) rather than user-level communication.

5. Experimental Results

In this section, we present results on a 1024-node/4096-process segment, that is, one cluster of the ASCI Q machine, for the above-mentioned communication patterns.

The results for both the hardware-based and the software-based barrier synchronization are shown in Figure 5. As expected, the one based on the hardware multicast is faster than the software one that uses point-to-point messages. In both cases the latency for the largest configuration is impressive: 10 μs for the hardware barrier and 30 μs for the software one. The network is capable of synchronizing 1024 nodes with a latency comparable to a single point-to-point communication in most commodity networks [1].

The broadcast bandwidth seen at the source node is shown in Figure 6. In order to compute the aggregate bandwidth, we multiply this value by the number of nodes. With the largest configuration, the total bandwidth is 140 MB/sec $\times$ 1024 = 140 GB/sec. The performance degradation that is experienced when we increase the number of nodes is mostly related to the delays of the communication protocol, and reflects the number of levels in the fat tree. In fact, a multicast packet is blocked until all of its children send an acknowledgment, and is therefore very sensitive to even minor delays.

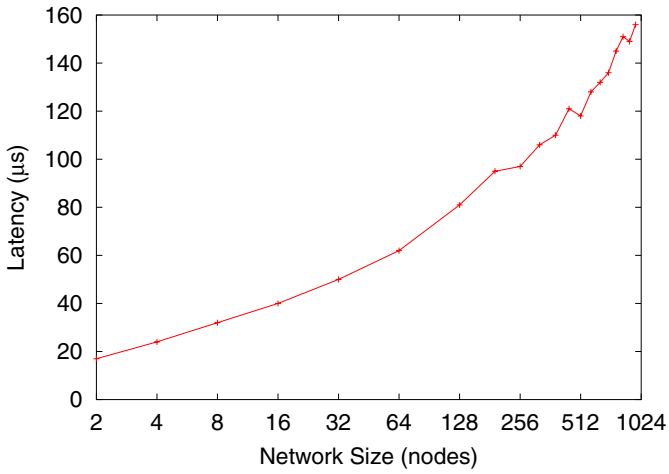


Figure 7. Allreduce

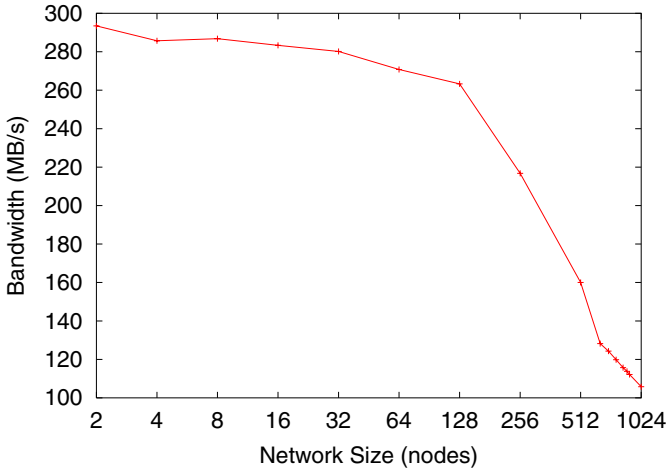


Figure 8. Hot Spot

The performance of the allreduce is again relatively high. In the collection phase the processing nodes must perform the arithmetic operation, slowing down the execution of the communication pattern. In the largest configuration the latency is only 160 μs.

Finally, in Figure 8 we can see the effect of the hot spot. The performance drop for configurations of more than 128 nodes is caused by a glitch in the low-level communication protocol. The largest packet size is only 384 bytes, and the hot node is blocked receiving a packet until the circuit between source and destination is released. This introduces a communication gap when the network has many levels because the packet is not large enough to cover the receipt of the acknowledgment at the source node.

6 Conclusion

We presented some performance results that prove the high scalability of the interconnection network of the ASCI Q machine. These results show that it is possible to execute collective communication patterns on a large configuration in a scalable way. In fact, in this cluster of 4096 processors, barrier synchronization broadcasts, and allreduce operations are achieved with a latency comparable to a single point-to-point communication in most commodity networks.

References

[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawick, C. L. Seitz, J. N. Seizovic, and W.-K. Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–36, Feb. 1995. Available from `http://www.myri.com/research/publications/Hot.ps`.

[2] J. Fernandez, F. Petrini, and E. Frachtenberg. BCS MPI: A New Approach in the System Software Design for Large-Scale Parallel Computers. In *Proceedings of IEEE/ACM Supercomputing 2003 (SC'03)*, Phoenix, AZ, November 2003.

[3] E. Frachtenberg, F. Petrini, J. Fernandez, S. Pakin, and S. Coll. STORM: Lightning-Fast Resource Management. In *Proceedings of IEEE/ACM Supercomputing 2002 (SC'02)*, Baltimore, MD, November 2002.

[4] D. Kerbyson, H. Alme, A. Hoisie, F. Petrini, H. Wasserman, and M. Gittings. Predictive Performance and Scalability Modeling of a Large-Scale Application. In *IEEE/ACM SC2001*, Denver, CO, November 2001. Available from `http://www.c3.lanl.gov/~fabrizio/papers/sc01.pdf`.

[5] C. E. Leiserson. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, Oct. 1985.

[6] F. Petrini, W. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network: High-Performance Clustering Technology. *IEEE Micro*, 22(1):46–57, Jan./Feb. 2002. Available from `http://www.c3.lanl.gov/~fabrizio/papers/ieeemicro.pdf`.

[7] F. Petrini, D. Kerbyson, and A. Hoisie. Further Improvements in ASCI Q Performance and Variability. Technical Report LAUR-03-1031, Los Alamos National Laboratory, March 2003.

[8] F. Petrini and M. Vanneschi. Performance Analysis of Wormhole Routed k-ary n-trees. *International Journal on Foundations of Computer Science*, 9(2):157–177, June 1998.

[9] G. F. Pfister and V. A. Norton. "Hot Spot" Contention and Combining in Multistage Interconnection Networks. *IEEE Transactions on Computers*, C-34(10):943–948, Oct. 1985.

[10] Quadrics Supercomputers World Ltd. *Elan Reference Manual*, Jan. 1999.

[11] Quadrics Supercomputers World Ltd. *Elite Reference Manual*, Nov. 1999.