PERFORMANCEEVAIUATION OF THE QUADRICS INTERCONNECTION NETWORK

Ankur Kumar Singhal

Department of Computer Science Engineering SGI, Samalkha(India)

ABSTRACT

In this paper we present an in-depth description of the Quadrics interconnection network (QsNET) and an experimental performance evaluation on a 64-node Alpha Server cluster. We explore several performance dimensions and scaling properties of the network by using a collection of benchmarks, based on different traffic patterns. Experiments with eructation patterns and uniform traffic are conducted to illustrate the basic characteristics of the interconnect under conditions commonly created by parallel scientific applications. Moreover, the behavior of the QsNET under I/O traffic, and the influence of the placement of the I/O servers are analyzed. The effects of using dedicated I/O nodes or shared I/O nodes are also exposed. In addition, we evaluate how background I/O traffic interferes with other parallel applications running concurrently. The experimental results indicate that the QsNET provides excellent performance in most cases, with excellent contention resolution mechanisms. Some important guidelines for applications and I/O servers mapping on large scale clusters are also given.

Keyword: *Interconnection Networks, Performance Evaluation, User-level Communication, Operating System Bypass.*

I. INTRODUCTION

Some interconnect technologies used in high-performance computers include Gigabit Ethernet [2], Giganet [5], SCI [8], Myrinet [2] and GSN (HIPPI 6400) [4]. Each one provides a different level of programmability, raw performance and integration with the operating-system.

The Quadrics interconnection network (QsNET) provides a number of innovative design issues, some of which are very similar to those defined by the Infinite Band specification. Some of these salient aspects are the presence of a programmable processor in the network interface that allows the implementation of intelligent communication protocols, fault-tolerance, and remote direct memory access. In addition, the QsNET integrates the local virtual memory into a distributed virtual shared memory. The Quadrics network is currently utilized in some of the largest parallel systems in the world, mainly connecting Compaq Alpha-based servers, but increasingly other compute platforms too. The performance of interconnection networks and, in particular, switch-based wormhole networks has been extensively analyzed by simulation in the literature [10] [3]. Since performance is strongly influenced by the load, it is very important to evaluate the QsNET under more varied traffic patterns to get a complete view of the network behavior. The patterns considered in this work are representative of real scientific applications in use at Los Alamos National Laboratory. One example of workload analysis is presented in [9] for SAGE (SAIC's Adaptive Grid Eulerian hydrocode), an important ASCI application.

1.1 THE QsNET

The QsNET is based on two building blocks, a programmable network interface called Elan [2] and a low-latency high bandwidth communication switch called Elite [1]. Elites can be interconnected in a fat-tree topology [10]. 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 and fault-tolerance.

1.2 ELAN

The Elan2 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 high-level message-passing protocols such as MPI. The internal functional structure of the Elan, shown in Figure 1, centers around two primary processing engines: the microcode processor and the thread processor. The 32-bit microcode processor supports four separate threads of execution, where each thread can independently issue pipelined memory requests to the memory system. Up to eight requests can be outstanding at any given time. The scheduling for the microcode processor is lightweight, enabling a thread to wake up, schedule a new memory access on the result of a previous memory access, and then go back to sleep in as few as two system-clock cycles. The four microcode threads are described below:

(1) inputter thread*:* Handles input transactions from the network.

(2) DMA thread: Generates DMA packets to be written to the network, prioritizes outstanding DMAs, and time-slices large DMAs so that small DMAs are not adversely blocked.

(3) processor-scheduling thread: Prioritizes and controls the scheduling and de scheduling of the thread processor.

(4) command-processor thread: Handles operations requested by the host processor at user level. The thread processor is a 32-bit RISC processor used to aid the implementation of higher-level messaging libraries without explicit intervention from the main CPU. In order to better support such an implementation, the thread processor's instruction set was augmented with extra instructions that construct network packets, manipulate events, efficiently schedule threads, and block-save and restore a thread's state when scheduling.

II. ELITE

The other building block of the QsNET is the Elite switch. The Elite provides the following features:

(1) 8 bidirectional links supporting two virtual channels in each direction,

(2) an internal 16x8 full crossbar switch3,

(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.

The Elite switches are interconnected in a quaternary fat-tree topology, which belongs to the more general class of the k-ary-trees [7] [6]. A quaternary fat-tree of dimension n is composed of 4n processing nodes and Switches n*4n-1 interconnected as a delta network, and can be recursively built by connecting 4 quaternary fat trees of dimension n-1.

2.1 Packet Routing and Flow Control

Each user- and system-level message is chunked in a sequence of packets by the Elan. An Elan packet contains three main components. The packet starts with the (1) routing nformation, that determines how the packet will reach the destination. This information is followed by (2) one or more transactions consisting of some header information, a remote memory address, the context identifier and a chunk of data, which can be up to 64 bytes in the current implementation. The packet is terminated by (3) an end of packet (EOP) token, as shown in Figure 3. Transactions fall into two categories: write block transactions and non-write block transactions. The purpose of a write block transaction is to write a block of data from the source node to the destination node, using the destination address contained in the transaction immediately before the data. A DMA operation is implemented as a sequence of write block transactions, partitioned into one or more packets (a packet normally contains 5 write block transactions of 64 bytes each, for a total of 320 bytes of data payload per packet). The non-write block transactions implement a family of relatively low level communication and synchronization primitives. For example, non-write block transactions can atomically perform remote test-and-write or fetch-and-add and return the result of the remote operation to the source, and can be used as building blocks for more sophisticated distributed algorithms.

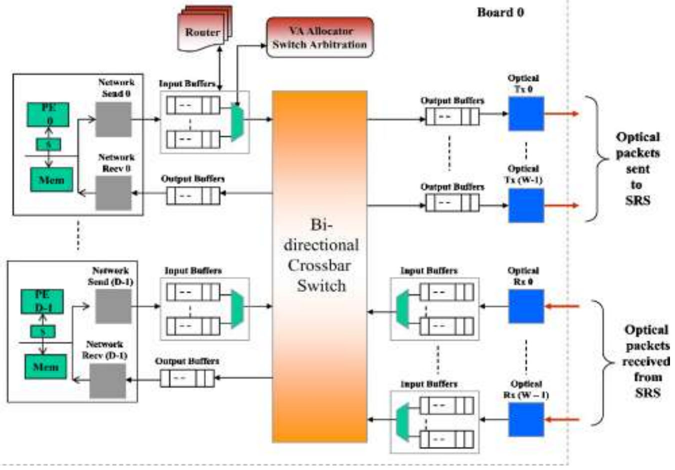

Fig 1: Process Interconnection

III. PROGRAMMING LIBRARIES

The Elan network interface can be programmed using several programming libraries [9], as outlined in Figure 4. These libraries trade speed with machine independence and programmability. Starting from the bottom, Elan3lib is the lowest programming level available in user space which allows the access to the low level features of the Elan3. At this level, processes in a parallel job can communicate with each other through an abstraction of distributed virtual shared memory. Each process in a parallel job is allocated a virtual process id (VPID) and can map a portion of its address space into the Elan. These address spaces, taken in combination, constitute a distributed virtual shared memory. Remote memory (i.e., memory on another processing node) can be addressed by a combination of a VPID and a virtual address. Since the Elan has its own MMU, a process can select which part of its address space should be visible across the network, determine specific access rights (e.g. write- or read-only) and select the set of potential communication partners.

3.1 Elan3lib

The Elan3lib library supports a programming environment where groups of cooperating processes can transfer data directly, while protecting process groups from each other in hardware. The communication takes place at user level, with no data copying, bypassing the operating system. The main features of Elan3lib are: (1) event notification, (2) the memory mapping and allocation scheme and (3) remote DMA transfers.

3.1.1 Event Notification

Events provide a general purpose mechanism for processes to synchronize their actions. The mechanism can be used by threads running on the Elan and processes running on the main processor. Events can be accessed both locally and remotely. Thus, processes can be synchronized across the network, and events can be used to indicate the end of a communication operation, such as a completion of a remote DMA. Events are stored in Elan memory, in order to guarantee the atomic execution of the synchronization primitives.

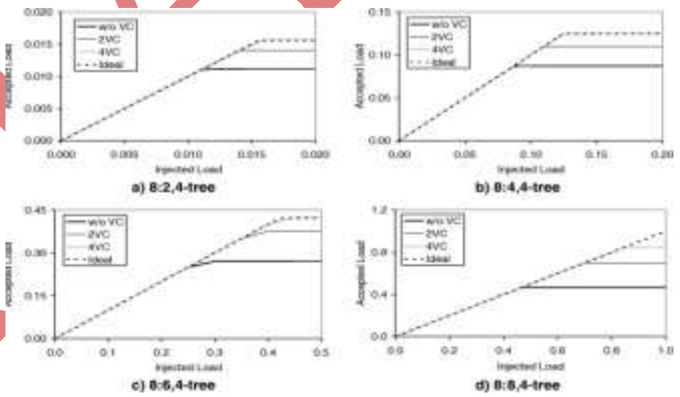

Fig. 2 A sample line graph using colors which contrast well both on screen and on a black-and-white hardcopy

3.1.2 Memory Mapping and Allocation

TheMMU in the Elan can translate between virtual addresses written in the format of the main processor (for example, a 64-bit word, big Endian architecture as the AlphaServer) and virtual addresses written in the Elan format (a 32-bit word, little Endian architecture). For a processor with a 32-bit architecture (for example an Intel

Pentium), a one-to-one mapping is all that is required. In Figure 5 the mapping for a 64-bit processor is shown. The 64-bit addresses starting at 0x1FF0C808000 are mapped to Elan's 32 bit addresses starting at 0xC808000. This means that virtual addresses in the range 0x1FF0C808000 to 0x1FFFFFFFFFF can be accessed directly by the main processor while the Elan can access the same memory by using addresses in the range 0xC808000 to 0xFFFFFFFF.

IV. EXPERIMENTS FRAMEWORK

We tested the main features of the QsNET on a 64-node cluster of Compaq AlphaServer ES40s, running Tru64 Unix. Each AlphaServer node is equipped with 4 Alpha 667MHz 21264 processors, 8 GB of SDRAM and two 64-bit, 33MHz PCI I/O buses. The Elan3 QM-400 card is attached to one of these buses and links the SMP to a quaternary fat tree of dimension three, as the one shown in Figure 2 c). Unless otherwise stated, the communication buffers are allocated in Elan memory in order to isolate I/O bus-related performance

limitations, except for the ping tests, whose goal is to provide basic performance results that are a reference point for the following experiments.

4.1 Unidirectional Ping

At Elan3lib level the latency is measured as the elapsed time between the posting of the remote DMA request and the notification of the successful completion at the destination. The unidirectional ping tests for MPI are implemented using matching pairs of blocking sends and receive.

4.2 Bidirectional Ping

The unidirectional ping experiments can be considered as the "peak performance" of the network. By sending packets in both directions along the same network path we can expose several types of bottlenecks.

4.3 Uniform Traffic

The uniform traffic is one the most frequently used traffic patterns for evaluating the network performance. With this pattern each node randomly selects its destinations. This distribution provides what is likely to be an upper bound on the mean internodes distance because most computations exhibit some degree of communication locality [9][5].

V. CONCUSION

In this paper, we present a comprehensive description and evaluation of the Quadrics interconnection network (QsNET), which can prove particularly useful for network designers, library developers and users of high performance parallel clusters. In the initial part we describe in depth the building blocks that compose the QsNET, the Elan network interface and the Elite routing chip. Relevant details are the internal architecture and the functional units of the Elan, and the network topology, routing algorithms and flow control algorithms that characterize the Elite. We also describe the hardware communication protocol that is at the base of the distributed virtual shared memory implemented by the QsNET. In the performance evaluation section, we analyze the communication performance and the scaling properties of the network using a broad set of communication patterns, on a 64-node AlphaServer cluster. We start with the unidirectional and bidirectional pings to analyze the basic network performance. Results show a remarkable latency of 2.2 and a bandwidth over 300 MB/sec. The results with uniform traffic show that the network doesn't scale well, due to internal congestion. For example, with 64 nodes, the delivered asymptotic bandwidth is only 42% of the optimal bandwidth. On the other hand, the QsNET can perform well on some important collective communication patterns, such as the complement and butterfly traffic. In order to identify the bottlenecks for the patterns that are not handled so efficiently by the network, like shuffle, transpose and bit-reversal, we visualize the network hot-spots using a congestion matrix that displays the aggregate wait times during the execution of the communication pattern for each link and routing switch. We presented an extensive performance evaluation of several types of I/O traffic. Although this analysis applies only to the topology investigated, it can prove particularly useful for system and network designers and users of high-performance parallel clusters.

REFERENCES

[1] InfiniBand Specification 1.0a. InfiniBand Trade Association, June 2001.

[2] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawick, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet: A Gigabitper- Second Local Area Network. IEEE Micro, 15(1):29–36, January 1995.

[3] Helen Chen and Pete Wyckoff. Simulation studies of Gigabit ethernet versus Myrinet using real application cores. In Proceedings of CANPC'00,

Workshop of High-Performance Computer Architecture, Toulouse, France, January 2000.

[4] William J. Dally and Charles L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. IEEE Transactions on Computers,

C-36(5):547–553, May 1987.

[5] José Duato, Sudhakar Yalamanchili, and Lionel Ni. Interconnection Networks: an Engineering Approach. IEEE Computer Society Press, 1997.

[6] Al Geist, William Gropp, Steve Huss-Lederman, Andrew Lumsdaine, Ewing Lusk, William Saphir, Tony Skjellum, and Marc Snir. MPI-2: Extending the Message Passing Interface. In Second International Euro-Par Conference, Volume I, number 1123 in LNCS, pages 128–135, Lyon, France, August 1996.

[7] Steve Heller. Congestion-Free Routing on the CM-5 Data Router. In Kevin Bolding and Lawrence Snyder, editors, First International Workshop,

PCRCW'94, volume 853 of LNCS, pages 176–184, Seattle, Washington, USA, May 1994.

[8] Hermann Hellwagner. The SCI Standard and Applications of SCI. In Hermann Hellwagner and Alexander Reinfeld, editors, SCI: Scalable Coherent Interface, volume 1291 of Lecture Notes in Computer Science, pages 95–116. Springer-Verlag, 1999.

[9] Darren Kerbyson, Hank Alme, Adolfy Hoisie, Fabrizio Petrini, Harvey Wasserman, and Mike Gittings. Predictive Performance and Scalability Modeling of a Large-Scale Application. In Supercomputing 2001, Denver, CO, November 2001.

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