Design and Evaluation of an HPVM-based Windows NT Supercomputer

A. Chien, M. Lauria,* R. Pennington, M. Showerman† G. Iannello‡
M. Buchanan, K. Connelly, L. Giannini, G. Koenig, S. Krishnamurthy,
Q. Liu, S. Pakin, G. Sampemane§

Abstract

We describe the design and evaluation of a 192-processor Windows NT cluster for high performance computing based on the High Performance Virtual Machine (HPVM) communication suite. While other clusters have been described in the literature, building a 58 GFlop/s NT cluster to be used as a general-purpose production machine for NCSA required solving new problems. The HPVM software meets the challenges represented by the large number of processors, the peculiarities of the NT operating system, the need for a production-strength job submission facility and the requirement for mainstream programming interfaces.

First, HPVM provides users with a collection of standard APIs like MPI, Shmem, Global Arrays with supercomputer class performance (13 μs minimum latency, 84 MB/s peak bandwidth for MPI), efficiently delivering Myrinet's hardware performance to application programs. Second, HPVM provides cluster management and scheduling (through integration with Platform Computing's LSF). Finally, HPVM addresses Windows NT's remote access problem, providing convenient remote access and job control (through a graphical Java-applet front-end).

Given the production nature of the cluster, the performance characterization is largely based on a sample of the NCSA scientific applications the machine will be running. The side-by-side comparison with other present generation NCSA supercomputers shows the cluster to be within a factor of two to four of the SGI Origin 2000 and Cray T3E performance at a fraction of the cost. The inherent scalability of the cluster design produces a comparable or better speedup than the Origin 2000 despite a limitation in the HPVM flow control mechanism.

1 Introduction

The advent of "killer micros" and "killer networks" is enabling clustered commodity machines to compete with supercomputers in aggregate performance. Many supercomputers today are microprocessor-based [35, 34, 24, 12, 37] because such microprocessors deliver instruction and floating point processing rates in excess of 1 billion operations per second. In addition, emerging high speed networks [17, 3, 6, 19, 25]) provide the aggregate capability for supercomputing performance. While the development of these technologies has long been anticipated, their potential impact has been accentuated by the development of new communication technology described below.

Over the past four years, the research community has produced dramatic progress in delivering hardware communication performance to applications (Fast Messages (FM) [30], Active Messages (AM) [9],

*Dept. of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Dr., Dept 0114, La Jolla, CA 92093-0114, USA

†National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, 605 E. Springfield Ave, Champaign, IL, 61820, USA

‡Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II", via Claudio 21, 80125 Napoli, Italy. Visiting at time of writing.

§Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 W. Springfield Ave., Urbana, IL 61801, USA

U-Net [40], VMMC-2 [14], PM [38], BIP [32], MINI [22], and Osiris [16]). These efforts have forged a consensus on core requirements for network interfaces to deliver high communication performance to application programs. This consensus includes the following key features which have recently been incorporated in the Intel/Compaq/Microsoft standard for cluster interfaces – the Virtual Interface Architecture [1].

- user-level protected network access,
- hardware support for gather-scatter,
- early demultiplexing of incoming traffic, and
- lightweight protocols.

Challenges to building a large cluster, particularly one based on PC's and Windows NT include: system selection, integration, delivering communication performance, cluster scheduling and management, and remote access. Our cluster is based on technology from the High Performance Virtual Machines (HPVM) project[1] which provides:

- **High Performance Communication:** $10\mu s$ latency and 90MB/s bandwdith employing *Fast Messages* to implement MPI, Global Arrays, Shmem Put/get.
- **Management and Scheduling:** Flexible scheduling, system monitoring, and job monitoring using Platform Computing's Load Sharing Facility
- **Remote Access:** Job submission, monitoring, and control of jobs via a Java applet front-end which runs on any Java-enabled system

Using these technologies, we built a 192-processor cluster, employing dual-processor 300 MHz Pentium II PC's and 160MB/s Myrinet. The resulting 58 GFlops system has 45 GB of DRAM, 1.6 GB/s of bisection bandwidth, and 400 GB of disk storage. The cluster is a National Computational Science Alliance project built in collaboration by the Concurrent Systems Architecture Group (CSAG) and the NCSA. It was integrated in six weeks, though it exploited a much longer chain of technology developed in Fast Messages and High Performance Virtual Machines, and was demonstrated at the Alliance '98 meeting on April 27, 1998. It went into production at NCSA in Fall 1998 in time to be demonstrated at Supercomputing 98: at the Alliance booth the NT cluster located in Urbana was used to perform simulations of Einstein's general relativity theory using Cactus, and to perform two-dimensional Navier Stokes calculations using AS-PCG achieving 6.9 Gflops on 128 of the processors. Because of the success of the project, a follow-on cluster of 512 processors is planned for early 1999.

Several aspects set apart this cluster from a number of similar projects. The first is the use of Microsoft Windows NT 4.0[2], an operating system with 30 million installed units but not traditionally associated with high performance computing and clusters. The second is the scalability analysis of a complete stack of communication software layers (from the network interface card firmware all the way up to MPI) over a two orders of magnitude range in the number of processors and machines (1 to 192, 1 to 96 respectively). Finally the cluster was built to serve as a general-purpose production machine in a national supercomputing center rather than a research prototype, targeting high-end applications with requirements in excess of 10 Gflops / 10 GB RAM / 1GB/s bisection bandwidth.

We pioneered the use of Windows NT because we sought to extend the notion of commodity cluster to the software environment seen by the users. None of the other inexpensive commodity operating systems available for the Intel platform can compete with NT's abundance of tools and applications, degree of industrial acceptance and amount of investments for support and development. And most importantly, the pervasiveness of the Windows user interface eases familiarization and acceptance by new users. We believe this to be a relevant concern as clusters of PC are putting high performance computing within reach of new scores of industrial and commercial users.

The contributions of this paper include lessons learned from building and evaluating such a large cluster.

[1] HPVM also provides this for Linux clusters as well.

[2] Windows is a registered trademark of Microsoft in the U.S. and other countries.

First, the use of the cluster as a production machine required that adequate cluster management and scheduling functionalities had to be built on top of NT. This task was complicated by the desktop-centric nature of NT. By integrating a commercial strength product like Platform Computing's LSF into HPVM, we were able to achieve the advanced level of service required with a modest effort. Over the course of a long and productive interaction with the HPVM team, Platform Computing accrued parallel job management services on top of LSF's native distributed computing concept.

Second, building the cluster exposed a wealth of software scaling issues in the HPVM software, Platform Computing's LSF software, and the underlying Windows NT operating system software. Many of these have been patched, some have been permanently fixed, but all of these are documented in this paper.

Third, the performance of the cluster itself is interesting, both microscopically and macroscopically, and we study and document the performance of the overall system both on microbenchmarks and on a sample of the cluster's planned workload of scientific applications. Depending on the application, the cluster is within a factor of two to four from SGI Origin 2000 and Cray T3E performance, with overall comparable or better scalability. While the achievement of comparable absolute performance would require a factor of two improvement in floating point performance in next generation Intel processors, the cluster is already ahead in terms of price/performance.

Finally, the intention of the authors, as well as both the HPVM project and NCSA, is to disseminate knowledge about building similar clusters, and thus we describe our experience in building the cluster – problems, snafus, and gotchas.

The rest of the paper is organized as follows. Section 2 describes the background technologies which are exploited in the HPVM cluster. In Section 3, the hardware and software in the cluster is described in detail. Basic system microbenchmarks and related performance figures are found in Section 4. Cluster performance on application programs is also presented in Section 4. Section 5 analyzes the results and discusses our experience in building the cluster. Finally Section 7 concludes the paper and presents the future developments in progress.

2 Background

Commodity cluster computing has been an active area of research since well before the advent of high performance interconnects. There is an obvious trade-off between application granularity and communication performance on one side, and between communication performance and cost on the other side. Different approaches to cluster computing have explored different segments along the axes of cost and of degree of generality.

The idea of tapping unused CPU cycles on networked machines is at the base of idle cycle scavenging systems like Condor [29] and Legion [28]. These systems transparently schedules applications, or application components, on machines detected to be in a idle state. In these kinds of systems application execution is subjected to available space / available time constraint in exchange for a zero-cost source of computing power.

The simultaneous reduction in PC price and increase in x86 processor performance turned the notion of dedicated clusters into something economically feasible. The reliance of Beowulf clusters [4] on basic, well-established Fast Ethernet technology represent the lowest cost version of the dedicated cluster approach. By embracing the use of legacy protocols with their inherent latency/bandwidth limitations, the Beowulf approach is better suited to large grained applications. The Beowulf concept emphasizes performance/price ratio over generality of use.

The advent of fast interconnect technologies [17, 3, 6, 19, 25]) has changed the role of cluster architectures - from machines for coarse granularity applications to inexpensive general purpose parallel computers. Over the past four years, the research community has produced dramatic progress in delivering hardware communication performance to applications (Fast Messages (FM) [30], Active Messages (AM) [9], U-Net [40], VMMC-2 [14], PM [38], BIP [32], MINI [22], and Osiris [16]). These efforts have forged a consensus on core requirements for network interfaces that has eventually been cristallized in the Intel/Compaq/Microsoft standard for cluster interfaces – the Virtual Interface Architecture [1].

Some of these and other efforts have also produced large scale clusters. In the Berkely NOW cluster a Myrinet interconnect links together 104 UltraSparc workstations running communication software based on AM (`http://now.cs.berkeley.edu`). The Real World Computing Partnership's PM effort has produced the PC Cluster II consisting of 64 (128 since Spring 1998) 200 MHz Pentium Pro interconnected by a Myrinet network and running Linux (`http://www.rwcp.or.jp/home-E.html`). In the Netherlands, the DAS initiative links clusters in four universities with a ATM WAN as a research testbed for parallel programming and wide area distributed computing; the larger cluster at the Vrije Universiteit of Amsterdam is built of 128 200 MHz PentiumPro interconnected by a Myrinet network and running BSD/OS (`http://www.cs.vu.nl/~bal/das.html`).

The HPVM research project has focused on crucial aspects of the design of high performance messaging layers such as the trade-offs between protocol processing overhead and the functionalities exposed in the programming interface. The key design decisions taken for the original Fast Messages (FM) [31] implementation provided key communication services while retaining microsecond-level overhead, and over the course of two more releases led to the present 9 μs min latency and 92 MB/s peak bandwidth of HPVM. Since its first release in Spring 1995, FM has been used in other research projects exploring high performance communication. The implementation of the higher level APIs on top of FM (MPI [26], Global Arrays, Shmem [20]) turned into a study on efficient software layering, which resulted in 90% of the FM performance being delivered to the applications [27]. FM has been used as a testbed for research on novel mechanisms for coscheduling on network of workstations [36], and on the introduction of QoS guarantees in a wormhole routing interconnect [11].

HPVM is available for download on our web site at `http://www-csag.ucsd.edu`.

3 The Machine: HPVM III

Cluster Software The software is based on the Illinois HPVM technology, which includes Illinois Fast Messages[3] [31] to deliver high bandwidth (92MB/s) with low overhead (a few μseconds), and low latency ($< 9\mu$s), achieving a half-power message size of 250 bytes while providing reliable and in-order delivery, and flow control.

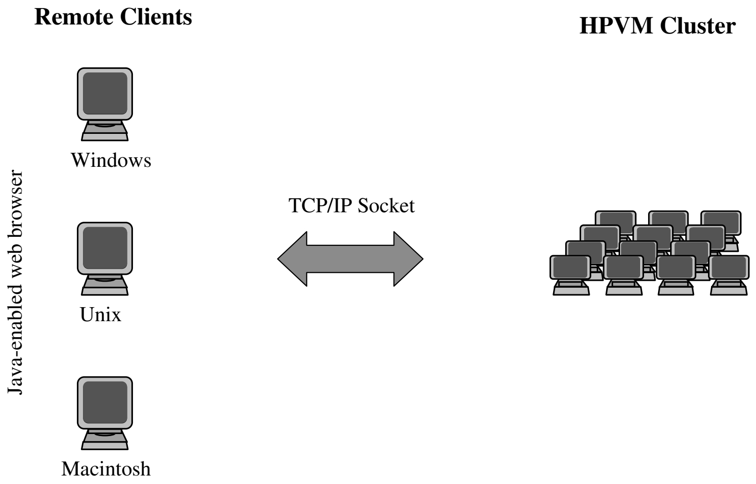
HPVM also includes efficient implementations of standard scientific computing APIs (MPI [26, 27] Global Arrays, Shmem [20]) atop FM. Performance highlights include 13.3 μs min latency and 84.2 MB/s peak bandwidth for MPI-FM, 14.4 μs min latency and 76 MB/s peak bandwidth for a Shmem's shmem_put operation. These APIs enable the easy porting and the high performance required to integrate the existing NCSA scientific applications in the cluster environment.

HPVM provides basic cluster monitoring and scheduling by integrating Platform Computing's Load-Sharing Facility [41], and adding a Java front-end for convenient remote access from any platform supporting a Java-enabled browser (Figure 1(a)).

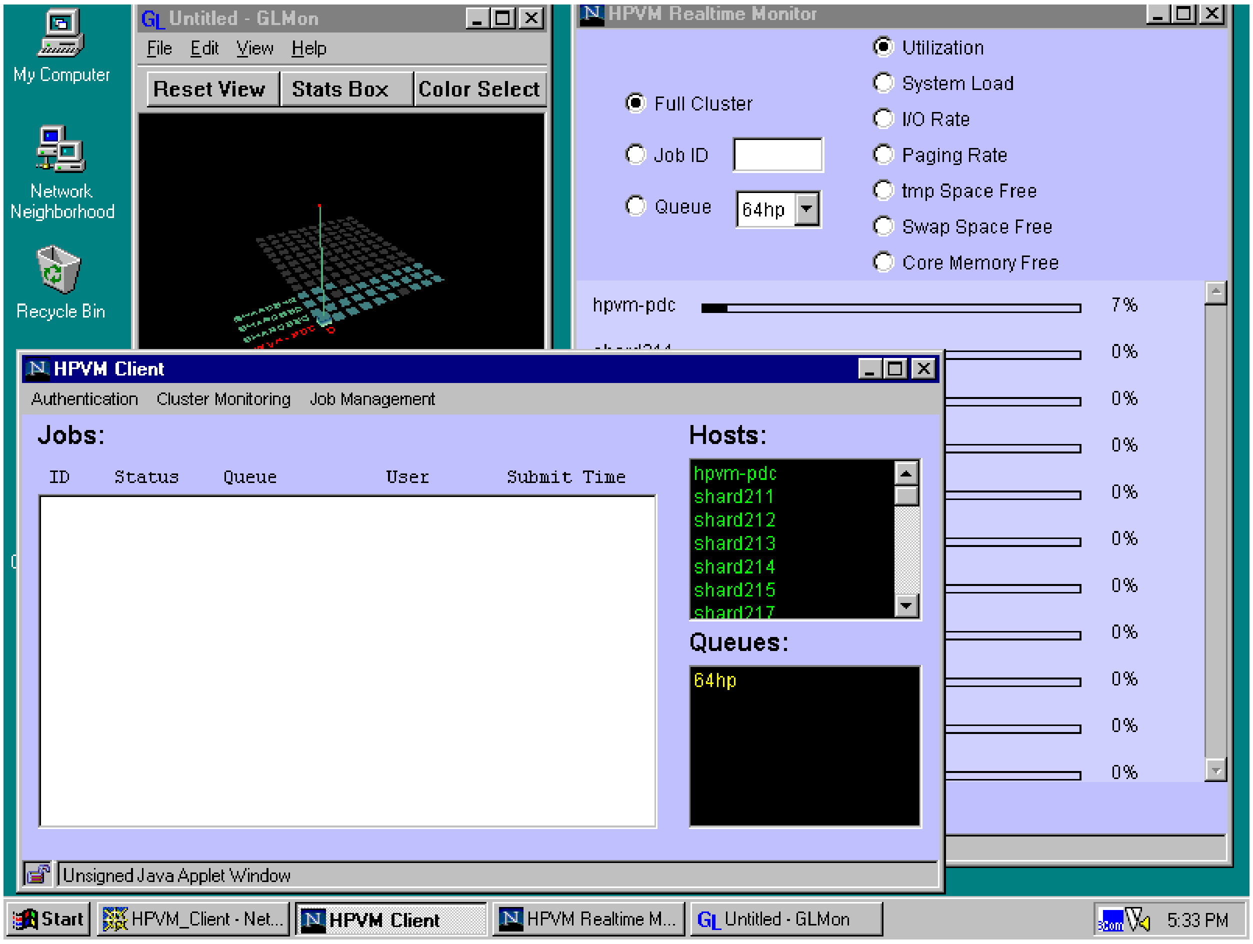
The main component of the front-end is an applet whose graphical interface displays the list of available hosts, queues and currently enqueued jobs (the *"HPVM client"* window in Figure 1(b)). Through the menu bar, other applets can be started, each giving access to the LSF services providing user authentication, cluster monitoring and job management. In Figure 1(b) the realtime monitor applet window is shown (the *"HPVM Realtime Monitor"* window), with the list of available node statistics visible in the upper half.

Cluster Hardware HPVM III comprises 192 300 MHz Pentium II processors in 96 nodes (2-way SMPs), 48 GB of memory (512MB per node), 384 GB of disk (4GB per node), 160MB/s full duplex Myrinet networking (per node), 100 Megabit Ethernet. Each machine is running its copy of Windows NT 4 Server, with the Fast Ethernet being used for file system sharing and machine administration. The machines are also connected as shown in Figure 2 using a collection of Myrinet Octal-8 and Dual-8 switches. A fat-tree topology was chosen for being deadlock-free, readily expandable and reasonably cost-effective, giving 1.62 GB/s of bisection bandwidth. The tree is organized in three levels of switches, the lowest one built of Myrinet octal switches each connected directly to 16 nodes (each line in the figure is a dual-link cable). Given the low routing latency of Myrinet, we found the exact topology to be less critical than the bisection bandwidth or the balanced distribution of routes between switches (HPVM is statically routed).

[3] This technology is one of the contributors to emerging Virtual Interface Architecture standard from Intel/Compaq/Microsoft.



(a) The Java front-end enables access to the cluster from any Java-enabled browser.



(b) The HPVM front-end and real-time monitor applets. The real-time monitor statistics are also displayed in a 2D bargraph format by GLmon, a small graphical utility developed at NCSA (upper left).

Figure 1: The HPVM front-end.

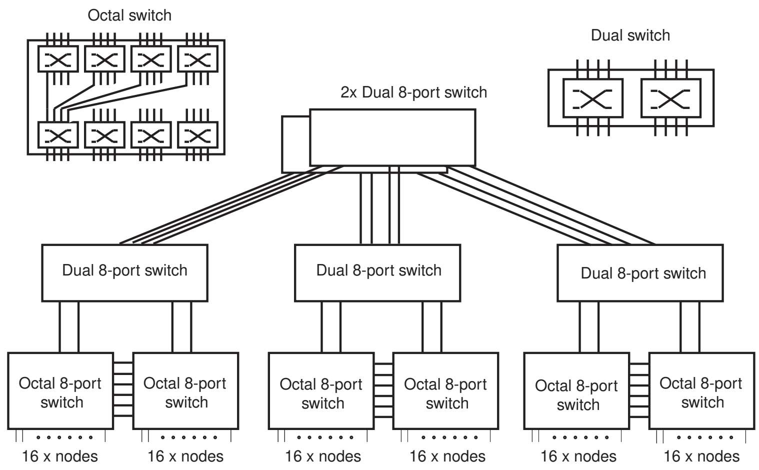


Figure 2: The fat-tree cluster topology

The node type and configuration was determined by extensive testing of a wide range of machines (9 machines from 5 vendors) with a focus on evaluating I/O subsystem performance. Testing targeted the elements critical for communication performance, namely memory bandwidth and I/O performance for both DMA and PIO transfers. The apparent homogeneity of PC hardware is illusory, with performance varying factors of 2 or 3 across different vendors, motherboards, and configurations. For example, we observed variations of 44 to 108 MB/s memory-to-memory copy bandwidth.

We chose the Hewlett-Packard Kayak XU, a dual Pentium II 300 MHz model, as our cluster building block. This model sustained a single-CPU sustainable memory copy bandwidth of 108.4 MB/s, a DMA-In/DMA-Out peak bandwidth of 122.5/120.9 MB/s, and a PIO-Read/PIO-Write peak bandwidth of 10.6/132.2 MB/s. The detailed list of measurements for this and other machines is available on our web site at `http://www-csag.cs.uiuc.edu/projects/comm/hcl.html`.

The PCs occupy five meters of two-meter high racking. The 96 machines machines require approximately 10 kW of power and cooling. Each keyboard/monitor/mouse is shared by sixteen machines through a Raritan console switch, resulting in a large saving on space, energy, cooling, and on the total cost. The console switch deals with Windows boot sequences which require the presence of a keyboard, and allow access to each machine for system administration purposes.

Local disks are used for operating system, local paging, and temporary storage. We are planning a number of additional PCs to act as file server with enhanced storage performance and capability. A disk imaging utility is being employed to clone the configuration of a master and simplify the task of updating communication and system software.

4 Evaluation

4.1 Microbenchmarks

Microbenchmarks were used to determining the peak communication performance and the basic limitations of HPVM. Microbenchmarks involve pairwise and multiparty communication. We used them to test and isolate the contribution to global performance of individual parts of the system like flow control and the route allocation policies.

Since HPVM is a cluster of SMPs, the microbenchmarks also explore the impact of different process placements (one or two per node). Critical issues here include memory, I/O bus, and NIC contention. All the benchmarks are performed through the MPI interface, using MPI_Send, MPI_Recv and MPI_Barrier calls. Table 1 summarizes the results. These and the other results presented in this section have been measured at the end of April 1998.

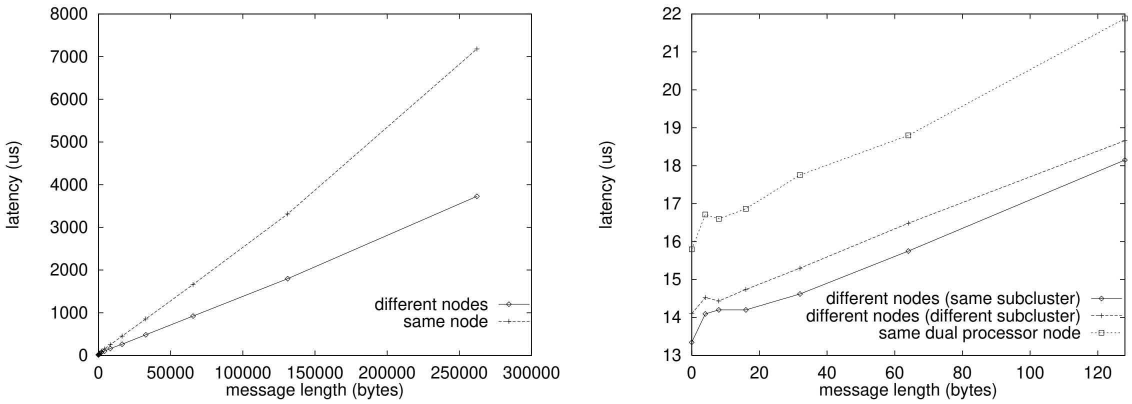
Test	Performance
Point-to-Point Bandwidth	84.2 MB/s
Point-to-Point Latency	13.3μs
Scaling P-to-P Bandwidth	various, see Figure 4
192 party Barrier	290 μs
Bisection Bandwidth	1.62 GB/s

Table 1: Summary of MPI-FM microbenchmark results

Latency Latency was measured using the classical repeated ping-pong scheme; the result is the median value of a series of measurements. Figure 3 shows both the latency for processes allocated on two different nodes and on the same dual processor node. In the first variant, both nodes directly connected through an octal switch and nodes connected by multiple switches (> 4) have been considered.

The minimum latency for a 0-byte message between different nodes is 13.3 μs for nodes connected through one octal switch. An additional latency of less than one microsecond is seen between nodes connected through multiple switches. These numbers are comparable to those found on current generation supercomputers. For example, MPI minimum latency on the the T3E is 14 μs [2].

Process allocation makes a difference in latency. In the dual processor allocation, this is due mainly due to the sharing of the I/O bus and of the LANai for both sending and receiving. This implies a lesser peak bandwidth (see next section) which explains the different slope of the two curves. Note that since bandwidth does not matter for very short messages, the two curves grow at the same rate with a higher fixed cost when the dual processor allocation is used.



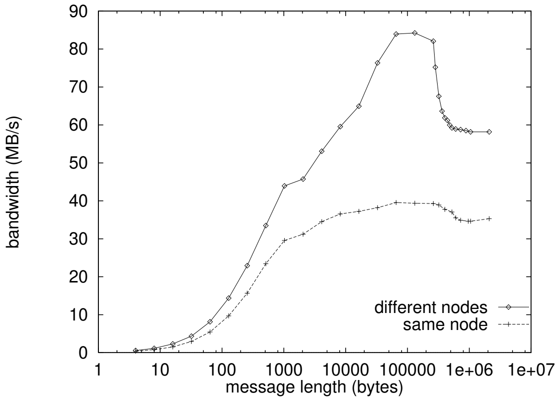
(a) long messages

(b) short messages

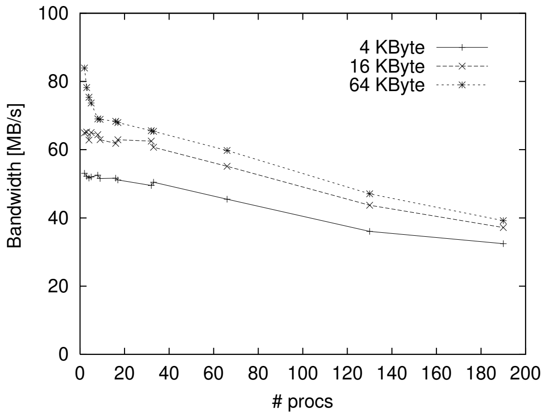
Figure 3: Latency between two MPI processes.

Bandwidth Bandwidth was measured by sending a sequence of messages in a row, and stopping the watch on the acknowledgement for the last one. The reported result is the median of several consecutive tests.

In Figure 4(a) bandwidth is shown both for processes allocated on two different nodes and on the same dual processor node. The peak bandwidth for the multiple node case is 84.2 MB/s for 128 Kbytes messages. The sharp decrease in bandwidth for messages longer than 128 Kbyte is due to cache effects in connection with the second-level cache. The half-performance message length $N_{1/2}$ is 970 bytes, which is less than half the FM packet size. The bandwidth for the dual processor case is lower as expected, since the I/O bus and the network interface are shared between the sender and the receiver. As a reference, the T3E peak bandwidth stands at 260 MB/sec [2].



(a) Bandwidth between two MPI processes.



(b) Dependence of bandwidth on the number of processes.

Figure 4: MPI bandwidth.

Scaling bandwidth The credit-based flow control scheme of FM equally allocates an amount of credits to all processes of a computation. To evaluate the impact of static credit partitioning as the number of processes increases, we used a test similar to the bandwidth test, in which $N \geq 2$ processes are spawned instead of two. Process 0 and process 1 act as the sender and the receiver of a standard bandwidth test, while all the other processes are idle waiting for program termination.

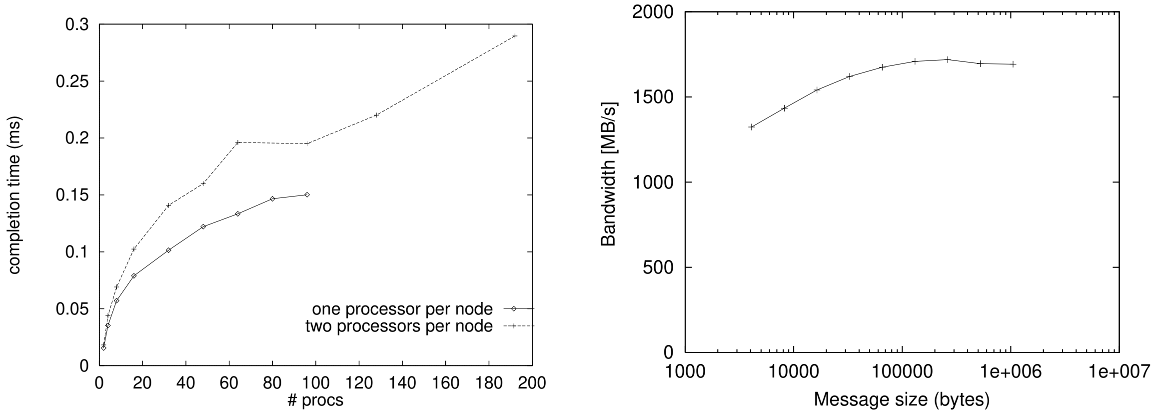
Figure 4(b) shows how the bandwidth varies with the number of processes for a given message length. Peak bandwidth decreases by as much as 50% over the entire range as the number of processes increases. By statically allocating a equal number of credits to all processes, the scheme trades simplicity for efficiency in the buffer management. The impact on application performance will be discussed later in the paper.

Barrier We measured the execution time of the MPI_Barrier operation as a function of the number of processes involved. The test, after a preliminary barrier to synchronize the processes, performs `num_reps` of barrier operations in a row.

In Figure 5(a) the barrier completion time is reported for two allocation schemes, one and two processes per node. In both cases the curve increase logarithmically as expected. We observe a higher completion time when both the processors in each node are involved. This is likely due to contention on the I/O bus which increases the gap between successive messages.

The absolute performance is remarkable, considering that, for example, on the Origin 2000 the barrier takes more than 1 ms to complete on 64 nodes (the T3E hardware supported barrier completes in 7μs on 256 nodes). If N is the number of processes involved, for single processor allocation the barrier completion time grows as $22 \log N$ μs, while for dual processor allocation it grows as $31 \log N$ μs.

Bisection bandwidth In this test, the bandwidth test is performed in parallel between $N/2$ pairs of processes allocated in such a way to saturate the bisection cut. The present peak value of 1.62 GB/s (Figure 5(b)) is the result of an improved route allocation policy and an upgraded interconnect with respect to the initial one used in the April '98 Alliance meeting. We doubled the top half of the tree to achieve a more balanced topology. This required modifying the HPVM static routing algorithm, based on a up*/down* scheme, to handle trees with more than one root. The up*/down* routing scheme [15] works by first computing a breadth-first logical tree starting from an arbitrary node (ideally the physical root in case of tree topolgy), and assigning a "up" and a "down" direction to each link in the network based on the computed tree. The scheme then mandates that all legal routes be composed of zero or more links in the "up" direction, followed by zero or more in the "down" direction. Such restriction prevents deadlock by avoiding the formation of dependency cycles between links. In our modified scheme the "up" and "down" assignement is modified to accomodate multiple roots. The result is a balanced allocation of routes across the multiple trees; however the use of more than one root makes the scheme potentially deadlock-prone in non-tree topologies.



(a) Barrier completion time

(b) Cluster bisection bandwidth

Figure 5: Barrier and bisection bandwidth

4.2 Applications

We present the performance measurements of four applications taken during the Alliance 98 meeting and in the following months. These application are also running on other NCSA supercomputers, allowing a direct comparative assessment of the system:

- ZeusMP [18] is a computational fluid dynamics code in Fortran using MPI developed at the Laboratory for Computational Astrophysics (University of Illinois at Urbana-Champaign) for the simulation of astrophysical phenomena. ZeusMP solves problems in three spatial dimensions with a wide variety of boundary conditions.

- Cactus. The Cactus code is a modular high performance 3D tool for numerical relativity. Cactus is developed jointly by researchers at AEI-Potsdam, NCSA, Washington University, and elsewhere.

- AS-PCG kernel. The AS-PCG is a Preconditionated Conjugate Gradient method with an Additive Schwarz-Richardson preconditioner for solving linear systems. It was developed by Danesh Tafti et al. at NCSA.

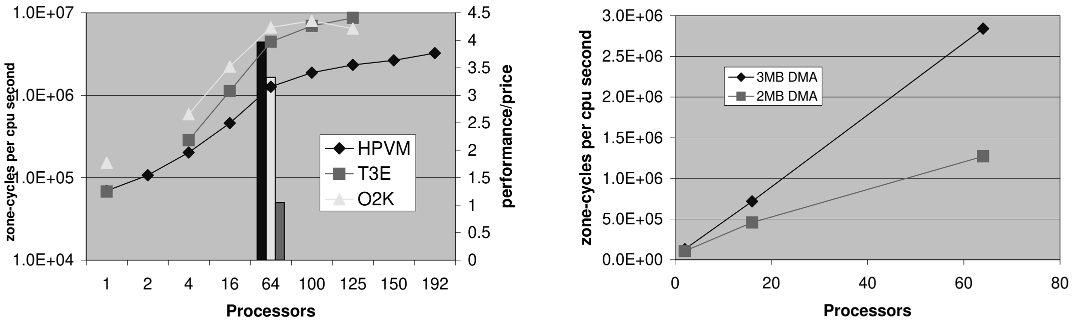
- QMC kernel. This is a Quantum Monte Carlo simulation method developed by the NCSA Condensed Matter Physics group at NCSA and it is being used to perform studies of electronic structure of molecular and condensed matter systems

The applications encompass a wide range of computation granularity, with ZeusMP representing one extreme (low granularity) and Cactus the other. ZeusMP (Figure 6(a)) and AS-PCG (Figure 7) where instrumental in evaluating and contrasting scalability. (We could not use less than 8 processors in the AS-PCG comparison because of memory limitations.) The other applications, Cactus and QMC (Figure 8) provide useful information on the comparative floating performance of the different architectures. All the applications are examples of pre-existing scientific codes that required only a modest recompilation effort to run unmodified on the cluster. No particular tuning was performed on the new platform to get the results shown.

Depending on the application, performance is a between a factor of 2 (AS-PCG) and of 4 (Zeus-MP) lower than the Origin 2000 and the Cray T3E. With respect to the two coarse-grained applications, the factor for Cactus is 2.5, for QMC is 1.8. Contrasting these values against the latency benchmark results, the conclusion is that the floating point performance of the nodes accounts for a large share of the performance gap. The baseline factor of two found in application performance is consistent, for example, with the SPECfp95 ratings of a Intel 300 MHz Pentium II motherboard and of the 195 MHz version of the Origin 2000, respectively 8.82 and 19.2, as reported on the SPEC web site (`http://www.spec.org`).

The additional factor of two seen in some applications is the effect of the reduced bandwidth produced by the flow control limitations on bandwidth intensive applications. We tested this effect using Zeus-MP. The graph in Figure 6(b) shows that a 50% increment of the size of the kernel DMA region from the current value of 2MB to 3MB results in an approximate doubling of performance. The increase to 3 MB adds enough buffering to substantially flatten the bandwidth curve of Figure 4(b) given the current delays in the credit recirculation mechanism. Such value represents only a modest additional cost in absolute terms, and less than 1% of the physical memory of the machines.

While extending the size of the DMA region is a practical solution for the short term, other solutions are under investigation. For example, we are testing some low level optimizations aimed at increasing the efficiency of the credit circulation in the system and that are showing an improved flatness of the bandwidth graph of Figure 4(b) without additional buffering. In another approach, a dynamic credit allocation scheme which adapts to the effective level of communication activity has been proposed [8].



(a) Comparison across machines

(b) The effect of a simple improvement in the flow control mechanism - a larger DMA kernel buffer

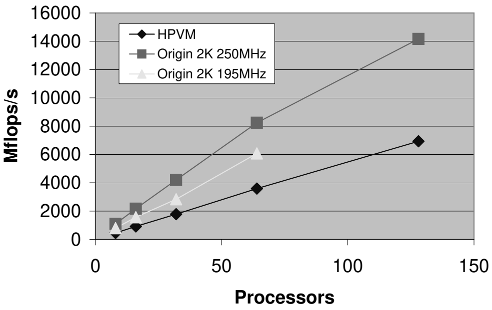
Figure 6: ZeusMP performance.

5 Discussion and Lessons

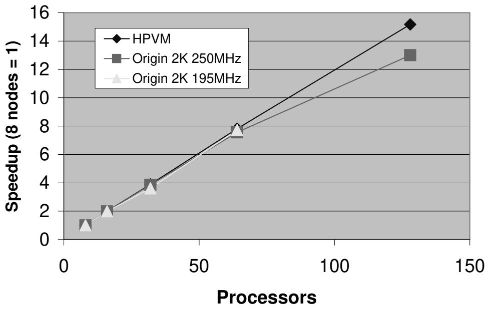
The cluster started working in mid-April, right after the last PC was plugged in. By using fully functional, proven technology building blocks (PCs, Myrinet LAN, HPVM) we found ourselves comparing performances a mere six weeks after opening the boxes of the last shipment of PCs. We learned a number of things in the process.

Given the focus of the project on programmability and standard interfaces, MPI-based applications happened to be among the first codes to run on the cluster. The effort of porting Unix applications to Windows NT was less than anticipated, with a number of annoying minor compiler incompatibilities taking center stage over differences in the runtime environments.

Connected to the porting issue is the relatively scarce availability of optimized standard libraries for scientific computing under Windows NT. In porting some of the applications we had to use our own build of the BLAS and LAPACK libraries, whereas highly optimized versions are currently available for other

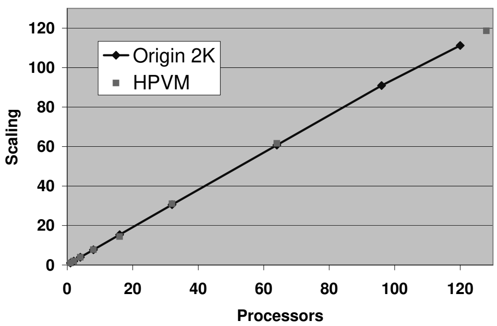


(a) Absolute performance

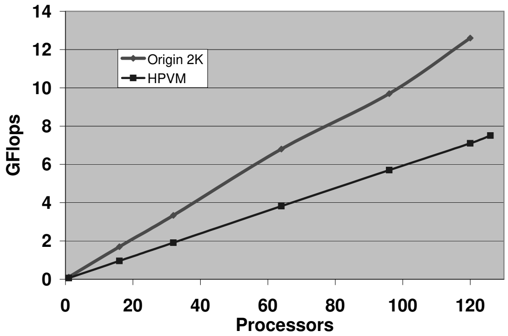


(b) Speedup

Figure 7: AS-PCG performance.



(a) Cactus



(b) QMC

Figure 8: Scaling of Cactus and QMC.

platforms. This problem is mitigated by the availability of commercial versions (existing or announced) by a number of vendors, including Intel.

The size of the cluster has proved a severe test for all the software components running on it. As shown above, our benchmarks revealed a problem within HPVM, the unsatisfactory scaling of the flow control scheme. By improving on this aspect of the design, we expect to see a further improvement in the performance scalability of bandwidth sensitive applications like ZeusMP.

Scaling problems were not limited to HPVM, but concerned also Windows NT and LSF. For example we discovered that by default Windows NT can not handle more that 64 TCP connections at a time on a single socket[4]. In its first release supporting NT clusters, LSF inherited this limitation; LSF basic design, in which one remote process was managed at a time, had been extended for parallel job execution by simply iterating the startup operations over all the job's processes. During the course of two minor releases, LSF managed to fix this and other initial problems, like a long startup time over a large number of processes and an imperfect trapping of the Ctrl-C signal in some circumstances.

From a system administration point of view, we found that Windows NT provides useful tools in some areas, like the remote administration of services (the NT equivalent of Unix demons), while is lacking in others, like remote registry access. For this and other reasons, in managing large clusters we found almost indispensable the use of disk reimaging tools.

6 Related work

A number of projects focusing on high performance communication have produced a real prototype. Most efforts involve either custom network hardware or high-performance low-level messaging layers.

High Performance Communication Layers Active Messages (AM) [39] has been one of the first realizations of high performance messaging layers. The AM project started as a communication library for the CM-5, and has culminated today into the realization of a network of workstations (NOW). The NOW cluster is composed of 105 UltraSparc1/170 connected by a Myricom network [13] and running the Solaris operating system. The network is a variant of the fat-tree topology, like ours. Built on top of AM, the high level APIs available on the cluster are MPI and Fast Sockets [33], a high performance version of the Berkeley Socket; MPI achieves a minimum latency of 36 μs and a peak bandwidth of 24.6 MB/s, Fast Sockets respectively 60 μs and 33 MB/s. A number of high-performance computing benchmark have been demonstrated [13], including the fastest disk-to-disk sorting program, and a distributed web search engine. An implementation of Split-C has been realized that uses AM in its run-time support.

Princeton's SHRIMP project is based on two different platforms. The first uses network interfaces built as part of the project, Pentium PCs and a Intel Paragon backplane as the network switch. In the second platform, Myrinet interfaces and switches are used in place of the custom network fabric. Princeton's Virtual Memory Mapped Communication (VMMC, VMMC-2) communication software has been developed on these two platforms [14]. A number of small scale clusters up to 16 nodes heve been built. The available high level API is an implementation of the Socket library, achieving 20 μs minimum latency and 84 MB/s peak bandwidth.

In some respects similar to FM is the Real World Computing Partnership's PM [38]. Like FM, PM runs on clusters of Myrinet-connected workstations and performs flow control and buffer management. The main difference with FM is in the optimistic flow control mechanism, and variable-sized packets. An implementation of MPI is available achieving 13 μs minimum latency and 98.8 MB/s peak bandwidth. An implementation of MPC++, a C++ with extensions for message passing, has been realized employing PM for its run-time support.

Another high performance messaging layer is U-Net [40]. Developed originally on a ATM network, it provides buffer management, demultiplexing in hardware but no flow control, and thus data can be lost due to overflow. Contrary to FM, U-Net and other messaging layers try to avoid the passage of data through kernel memory by performing a DMA transfer directly into the user buffer. The disadvantage of such feature

[4]This limit can be removed by assigning a different value to a compiler constant in the winsock library

is that the user must declare in advance the regions of memory to be used for communication, so to allow the library to permanently pin them down.

BIP [32] is another messaging layer developed for the Myrinet at the Ecole Normale Superieure de Lyon. It has a higher level, more traditional message passing interface, with both blocking and non blocking send/receive primitives. BPI provides high bandwidth, low latency, unreliable communication, with an adaptive packet format. It has been specifically designed to support standard message passing libraries like MPI and PVM, for which its interface represents a good match.

Hardware approaches An alternative to optimizing protocol performance in software is to develop hardware that delivers performance to software by presenting the system with an interface for which it is easier to optimize the protocol stack. Hamlyn [7], ServerNet [23], and hardware based on the new Virtual Interface Architecture standard [10] migrate protection checks from software to hardware, enabling user-level programs to access the network directly once the operating system has established a connection between endpoints. Similarly, Memory Channel [21] and SHRIMP [5], each of which exports a put interface (i.e. an interface in which messages are sent to a sender-specified address on the receiver), requires the operating system only to establish mappings from between nodes. All data transfers are performed user-level.

The Fast Messages interface is designed to be portable to a variety of network interfaces, including the new, non-traditional interfaces just mentioned. FM was originally implemented on the Cray T3D [12], which uses a put/get interface. It currently runs on Myrinet [6]. And we are in the process of porting FM to ServerNet. In all cases, the value added by FM is in its programmability. By providing flow control and buffer management—two important features almost never implemented in hardware—FM removes the burden of delivering reliable, ordered delivery from higher-level messaging layers an applications.

7 Summary and Future Work

Leveraging on a number of readily available technologies we were able to assemble a 192 node cluster in a matter of weeks. The use of mainstream software interfaces and operating systems enabled us to quickly demonstrate a number of HPC scientific applications and to use them in the testing and evaluation phases. The cluster is currently in use as a production machine at the NCSA in Urbana.

The direct comparison using stock NCSA scientific applications demonstrate that HPVM successfully integrates a large number of PCs into a tightly coupled machine which is within a factor of two to four of SGI Origin 2000 and Cray T3E performance at a fraction of the cost. The inherent scalability of the cluster design produces a comparable, and in some applications better, speedup than the Origin 2000.

To achieve these results, we first built standard APIs (MPI, Global Arrays, Shmem) capable of delivering the performance of a high speed interconnect to the applications (13 μs minimum latency, 84 MB/s peak bandwidth for MPI). We then provided system management services that were lacking in the NT operating system, and then we solved a number of scalability issues at different levels in the software hierarchy.

Since HPVM runs entirely in user space it was relatively straightforward to port to NT. The appeal of NT is the affordable and extensive base of available tools, and the potential for a larger acceptance by a growing number of nontechnical users. Thanks to the integration with Platform Computing's LSF, HPVM provides robust remote job execution and cluster monitoring services.

The major scalability issue that we found came from the static nature flow control mechanism we had built in HPVM. In addition to effective short-term remedies – essentially increase in buffer space and hand tuning of critical parts of the mechanism, we are considering longer term solutions like a more dynamic scheme.

We believe the HPVM design to have a potential for decisive performance improvements in the future. The above changes to the flow control scheme will benefit bandwidth limited applications; we have reported a doubling of performance for the ZeusMP application when increasing the size of a critical buffer. Perhaps more importantly, the next generation of the Intel CPU architecture is going to have a decisive improvement in CPU floating performance, an area in which the cluster is most distant from the Cray T3E and the SGI Origin 2000.

8 Acknowledgements

The research efforts of the UCSD and UIUC members of the CSAG group are supported in part by DARPA orders #E313 and #E524 through the US Air Force Rome Laboratory Contracts F30602-96-1-0286 and F30602-97-2-0121, and NSF Young Investigator award CCR-94-57809. Support from Microsoft, Intel Corporation, Hewlett-Packard, and Tandem Computers is also gratefully acknowledged. The NCSA staff is supported by the National Science Foundation, the state of Illinois, the University of Illinois, industrial partners, and other federal agencies. Scott Pakin is supported by an Intel Foundation Graduate Fellowship. Giulio Iannello has been partially supported by the Italian Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST) in the framework of the MOSAICO (Design Methodologies and Tools of High Performance Systems for Distributed Applications) Project, and by the University of Napoli Federico II in the framework of the Short-Term Mobility Programme. Mario Lauria has been supported in part by a NATO-CNR Advanced Science Fellowship.

The authors wish to thank the developers of the applications used in the performance characterization: the Laboratory for Computational Astrophysics at UIUC for ZeusMP, the Cactus group, Danesh Tafti for AS-PCG, the NCSA Condensed Matter Physics group for QMC.

References

[1] *The Virtual Interface Architecture Specification Version 1.0*, December 1997. Promoted by Intel, Compaq, and Microsoft and available from http://www.viarch.org/.

[2] Ed Anderson, Jeff Brooks, Charles Grassl, and Steve Scott. Performance of the Cray T3E multiprocessor. In *Supercomputing 97*, San Jose, California, November 1997.

[3] T. M. Anderson and R. S. Cornelius. High-performance switching with Fibre Channel. In *Digest of Papers Compcon 1992*, pages 261–268. IEEE Computer Society Press, 1992. Los Alamitos, Calif.

[4] D. Becker, T. Sterling, D. Savarese, J. Dorband, U. Ranawak, and C. Packer. Beowulf: A parallel workstation for scientific computing. In *Proceedings of the International Parallel Processing Symposium*, 1995.

[5] Matthias A. Blumrich, Cezary Dubnicki, Edward W. Felten, Kai Li, and Malena R. Mesarina. Virtual-memory-mapped network interfaces. *IEEE Micro*, pages 21–28, February 1995.

[6] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet—a gigabit-per-second local-area network. *IEEE Micro*, 15(1):29–36, February 1995. Available from `http://www.myri.com/research/publications/Hot.ps`.

[7] Greg Buzzard, David Jacobson, Milon Mackey, Scott Marovich, and John Wilkes. An implementation of the Hamlyn sender-managed interface architecture. In *Proceedings of the 2nd Symposium on Operating Systems Design and Implementation (OSDI '96)*, pages 245–259, October 1996. Available from `http://www.hpl.hp.com/personal/John_Wilkes/papers/HamlynOSDI96.pdf`.

[8] Roberto Canonico, Rosario Cristaldi, and Giulio Iannello. A scalable flow control algorithm for the Fast Messages communication library. In *Workshop on Communication, Architecture and Applications for Network-based Parallel Computing (CANPC '99)*, Orlando, Florida, January 1999.

[9] B. Chun, A. Mainwaring, and D. Culler. Virtual network transport protocols for Myrinet. In *Proceedings of Hot Interconnects V*. IEEE, 1997.

[10] Compaq Computer Corp., Intel Corp., and Microsoft Corp. *Virtual Interface Architecture Specification*, December 1997. Available from `http://www.viarch.org/html/Spec/vi_specification_version_10.htm`.

[11] Kay Connelly and Andrew A. Chien. FM-QoS: Real-time communication using self-synchronizing schedules. In *Proceedings of SC97*, November 1997. Available from `http://www.supercomp.org/sc97/program/TECH/CONNELLY/INDEX.HTM`.

[12] Cray Research, Inc., Eagan, MN. *Cray T3D System Architecture Overview*, March 1993.

[13] David E. Culler, Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau, Brent Chun, Steven Lumetta, Alan Mainwaring, Richard Martin, Chad Yoshikawa, and Frederick Wong. Parallel computing on the berkeley now. In *9th Joint Symposium on Parallel Processing (JSPP'97)*, Kobe, Japan, 1997.

[14] Cezary Dubnicki, Angelos Bilas, Yuqun Chen, Stefanos Damianakis, and Kai Li. VMMC-2: efficient support for reliable, connection-oriented commnication. In *Proceedings of Hot Interconnects V*. IEEE, August 1997. Available from `http://www.cs.princeton.edu/shrimp/Papers/hotIC97VMMC2.ps`.

[15] M. D. Schroeder et al. Autonet: A high speed, self-configuring local area network using point-to-point links. Technical Report SRC Research Report 59, DEC, April 1990.

[16] Peterson et. al. Experiences with a high speed network adaptor: A software perspective. In *SIGCOMM*, 1994.

[17] Fiber-distributed data interface (FDDI)—Token ring media access control (MAC). American National Standard for Information Systems ANSI X3.139-1987, July 1987. American National Standards Institute.

[18] Robert A. Fiedler. Optimization and scaling of shared-memory and message-passing implementations of the ZEUS hydrodynamics algorithm. In *Proceedings of SC97*, 1997. Available from `http://www.supercomp.org/sc97/program/TECH/FIEDLER/INDEX.HTM`.

[19] David Garcia and William Watson. Servernet ii. In *Proceedings of the Parallel Computer Routing and Communications Workshop*. Springer-Verlag LNCS, 1997.

[20] Louis A. Giannini. and A. A. Chien. A software architecture for global address space communication on clusters: Put/Get on Fast Messages. In *Proceedings of High-Performance Distributed Computing Conference*, 1998. Available from `http://www-csag.cs.uiuc.edu/papers/hpdc7-giannini.ps`.

[21] Richard B. Gillett. Memory Channel network for PCI. *IEEE Micro*, 16(1):12–18, February 1996. Available from `http://www.computer.org/pubs/micro/web/m1gil.pdf`.

[22] F. Hady, R. Minnich, and D. Burns. The Memory Integrated Network Interface. In *Proceedings of the IEEE Symposium on Hot Interconnects*, 1994.

[23] Robert W. Horst and David Garcia. ServerNet SAN I/O architecture. In *Hot Interconnects V*, Stanford, California, August 1997. Available from `http://http.cs.berkeley.edu/~culler/hoti97/horst.ps`.

[24] IBM Corporation, White Plains, NY. *Scalable POWERparallel System*, 1995. `http://ibm.tc.cornell.edu/ibm/pps/sp2/sp2.html`.

[25] *IEEE Std. 1596-1992: Standard for Scalable Coherent Interface (SCI) Specification*, August 1993. ISBN 1-55937-222-2.

[26] Mario Lauria and Andrew Chien. MPI-FM: High performance MPI on workstation clusters. *Journal of Parallel and Distributed Computing*, 40(1):4–18, January 1997. Available from `http://www-csag.cs.uiuc.edu/papers/jpdc97-normal.ps`.

[27] Mario Lauria, Scott Pakin, and A. A. Chien. Efficient layering for high speed communication: Fast Messages 2.x. In *Proceedings of High-Performance Distributed Computing Conference*, 1998. Available from `http://www-csag.cs.uiuc.edu/papers/hpdc7-lauria.ps`.

[28] Mike Lewis and Andrew Grimshaw. The core legion object model. Technical Report UVa CS Technical Report CS-95-35, August 1995, University of Virginia, 1995.

[29] Michael J. Litzkow, Miron Livny, and Matt W. Mutka. Condor—a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, June 1988.

[30] Scott Pakin, Vijay Karamcheti, and Andrew A. Chien. Fast Messages: Efficient, portable communication for workstation clusters and MPPs. *IEEE Concurrency*, 5(2):60–73, April-June 1997. Available from `http://www-csag.cs.uiuc.edu/papers/fm-pdt.ps`.

[31] Scott Pakin, Mario Lauria, and Andrew Chien. High performance messaging on workstations: Illinois Fast Messages (FM) for Myrinet. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, volume 2, pages 1528–1557, San Diego, California, December 1995. Available from `http://www-csag.cs.uiuc.edu/papers/myrinet-fm-sc95.ps`.

[32] Loic Prylli and Bernard Tourancheau. Protocol design for high performance networking: a Myrinet experience. Technical Report N. 97-22, LIP, Ecole Normale Superieure de Lyon, July 1997. Available from `http://www-bip.univ-lyon1.fr/`.

[33] Steve Rodrigues, Tom Anderson, and David Culler. High-performance local-area communication using Fast Socket. In *Proceedings of the USENIX 1997 Technical Conference*, San Diego, California, January 1997. USENIX Association. Available from `http://now.cs.berkeley.edu/Papers2/`.

[34] Steven L. Scott. Synchronization and communication in the T3E multiprocessor. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, pages 26–36, Cambridge, Massachusetts, October 1996. Available from `http://reality.sgi.com/sls_craypark/Papers/asplos96.html`.

[35] Silicon Graphics, Inc., Mountain View, CA. *Origin Servers: Technical Overview of the Origin Family*, 1996. `http://www.sgi.com/Products/hardware/servers/technology/overview.html`.

[36] Patrick Sobalvarro, Scott Pakin, Andrew Chien, and William Weihl. Dynamic coscheduling on workstation clusters. In *12th Annual International Parallel Processing Symposium & 9th Symposium on Parallel and Distributed Processing (IPPS/SPDP), 4th Workshop on Job Scheduling Strategies for Parallel Processing*, Orlando, Florida, March 1998. Available from `http://www.research.digital.com/SRC/scheduling/papers/pgs/nfmdcs.ps`.

[37] Sun Microsystems. *Ultra Enterprise 10000 System Overview*, 1997. Available from `http://www.sun.com/servers/enterprise/10000/wp/E10000.ps`.

[38] Hiroshi Tezuka, Atsushi Hori, and Yutaka Ishikawa. PM: A high-performance communication library for multi-user parallel environments. Technical Report TR-96-015, Tsukuba Research Center, Real World Computing Partnership, November 1996. Available from `http://www.rwcp.or.jp/papers/1996/mpsoft/tr96015.ps.gz`.

[39] T. von Eicken, D. Culler, S. Goldstein, and K. Schauser. Active Messages: a mechanism for integrated communication and computation. In *Proceedings of the International Symposium on Computer Architecture*, pages 256–266, 1992.

[40] Thorsten von Eicken, Anindya Basu, Vineet Buch, and Werner Vogels. U-Net: A user-level network interface for parallel and distributed computing. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, pages 40–53, December 1995. Available from `http://www2.cs.cornell.edu/U-Net/papers/sosp.pdf`.

[41] Songnian Zhou, Jingwen Wang, Xiaohu Zheng, and Pierre Delisle. Utopia: A load sharing facility for large, heterogeneous distributed computer systems. *Software—Practice and Experience*, 23(12):1305–1336, December 1993. Als appeared as Technical Report CSRI-257, April, 1992. Available from `http://platform.com/products/lsf.paper.ps.Z`.