Performance Analysis of an IBM supercluster

Daniele Tessera
Dipartimento di Informatica e Sistemistica
Università degli Studi di Pavia
Via Ferrata 1, I-27100 Pavia, Italy
tessera@gilda.unipv.it

Abstract

Cluster computing has emerged as an alternative approach to deliver high performance to HPC applications. Cluster machines, built on top of Commodity Off The Shelf components and based on Open Source Software are becoming very popular. Moreover, state of the art cluster machines rank among the top most powerful machines, thanks to specialized hardware and software components. It is then important to characterize the actual performance achieved by these machines.

In this paper, we present a performance characterization of a large Linux cluster, that is, the IBM NetFinity located at the Maui High Performance Computing Center. The novelty of this study is that we have analyzed the times spent by the allocated processors to accomplish the various activities required by the application. Our performance characterization follows a bottom up approach. We initially focus on basic communication performance. Analytical models of the times spent in sending and receiving messages, as a function of their size, will be presented. Then, we will discuss the behavior of a few widely used numerical algorithms when executed over both a standard Ethernet and an high performance Myrinet interconnection networks. A statistical clustering analysis of these behaviors will complete our performance study.

1 Introduction

Nowadays, industrial and scientific applications are increasing their high performance demands. Cluster computing has emerged as a cost effective solution to match with these demands [10]. Indeed, cluster computing allows the applications to exploit the aggregated computing capabilities of personal computers and workstations connected via local area networks [16]. This idea, initially exploited by PVM [21], has been fueled by the increased computation power of personal computers.

Various approaches, such as, distributed concurrent computing [21], network of workstation [2, 7], Beowulf architectures [6, 20], multicomputer operating system [5, 13] have been proposed to design machines with better performance and flexibility.

A very popular approach for building cluster machines is based on using Commodity Off The Shelf (COTS) components, such as Intel Pentium processors and Ethernet networks. This approach is often based on open source software to provide flexibility and application code portability. Hence, parallel applications developed for traditional massively parallel supercomputers, can be easily adapted to run on cluster machines.

Specialized hardware and software components have been developed to improve the performance of cluster machines, as well as their capabilities to meet the requirements of HPC applications. For example, a typical performance issue of these applications is related to the times spent by allocated processors in communication activities. Hence, state of the art cluster machines are equipped with ATM/Giganet/Quadrix/Myrinet interconnection networks to address such an issue. High performance network interface cards (NIC) and novel network user level interfaces, such as, the Virtual Interface Architecture [23]), have been proposed to improve the communication performance.

On the other hand, libraries and tools, as well as optimized middleware, aimed at improving the actual performance achieved by specialized hardware and parallel task schedulers, are examples of software components developed to increase the actual performance of cluster machines.

These machines, initially developed as prototypes by research centers, are becoming very popular. Many studies [3, 4, 8, 9, 11, 12, 14, 15, 17, 18, 22, 24] present the characteristics of various cluster machines. More-

over, commercial clusters are offered by various hardware vendors as viable alternatives to traditional parallel supercomputers. It is then important to investigate the performance that can be achieved by cluster machines. In this paper we present a performance study of a large commercial Linux cluster, that is, an IBM NetFinity cluster. In particular, we have analyzed the performance of the cluster when processing a few numerical algorithms resembling the computational cores of many scientific and industrial applications. The aim of this study is to characterize the performance achieved by the cluster when processing various numerical algorithms.

The paper is organized as follows. Section 2 presents our experimental approach to the characterization of the cluster performance. An overview of the IBM NetFinity and of the testbed kernels used to investigate its performance is presented in Section 3. Section 4 describes the performance results, whereas Section 5 presents our final remarks and outlines future works.

2 Experimental Methodology

The performance actually delivered by cluster machines is the result of complex interactions between software and hardware components of the machine, as well as of the application characteristics. Our approach for benchmarking these machines is based on the analysis of performance measures collected during application run time. The originality of this study is that it goes beyond the mere wall clock times by focusing on the times spent to accomplish basic activities, such as communications and computations. In particular, the performance results presented in the Section 4 are based on the analysis of timing measurements collected during various kernel executions. These measurements consist of the wall clock times spent by the allocated processors in a few selected routines, as well as in their communication activities. In order to characterize the machine performance, all the analyzed measures have been collected by running the kernels on dedicated nodes, that is, no other applications were running concurrently with our test kernels. Moreover, an ad hoc monitoring system has been developed in order to minimize the perturbations on kernel executions, due to monitoring activities. Indeed, the monitoring facilities embedded in communication libraries were not well suited for profiling the kernels since they might issue a very large number of measurement records.

Note that, our study is aimed at evaluating the long run behaviors of the various kernels from measurements collected on testbed runs. Hence, statistical techniques have been applied to measured timings in order to minimize the impact of non deterministic random effects. Measurements have been then repeated several times, depending on individual kernel characteristics. Statistical outliers deletion, based on the 99^{th} percentile, was then applied to these measures. Hence, the longest 1% times are discarded as due to anomalous, sporadic effects.

Our performance characterization follows a bottom up approach. Basic low level communication performance has been initially analyzed. Analytical models of the wall clock times required for sending and receiving messages, as a function of the message size, have been derived. We have then investigated the behavior of various numerical kernels, varying the number of allocated processors. In particular, we have analyzed the kernel scalability when executed over a standard Ethernet interconnection network, as well as over specialized hardware, such as the Myrinet network. Finally, the impact of these networks on the kernels' performance has been studied by means of statistical clustering analysis. The aim of this analysis is to identify classes of algorithms experiencing similar performance.

3 Hardware & Software Environments

This section describes the hardware and software environments used for collecting the performance measurements. As a testbed machine we have used an IBM NetFinity cluster located at the Maui High Performance Computing Center [19]. This machine is a cluster composed of 260 nodes running Linux as operating system. Each node consists of two Intel Pentium III processors, clocked at 933Mhz, and 1Gbyte of memory. Nodes are connected via both high performance Myrinet switches and a Fast 100Mbps Ethernet network. The cluster ranks among the largest and most powerful cluster machines. Note that the architecture of the IBM NetFinity is the classical architecture of Beowulf clusters. Hence, the performance results derived from our measurements can be extended to standard Linux clusters. The software environment consists of both system software, such as commodity tools and libraries, as well as of the testbed kernels used for benchmarking the machine performance. System software includes the communication library, that, in our case, is the mpich from the Argonne National Laboratory, and the parallel batch scheduler. The Maui Scheduler Open Cluster Software has been used to run all the analyzed testbed kernels on dedicated nodes.

The performance of the IBM NetFinity has been analyzed by evaluating the behavior of a few kernels from them well known ParkBench suites. In partic-

ular, low level communication performance has been benchmarked by means of the `comms1` kernel from the ParkBench suite. The behaviors of various test kernels from the NAS Parallel Benchmarks suite, have been analyzed with the aim to characterize the performance of the cluster on HPC scientific computing. In particular, the considered numerical algorithms are:

`BT` multiple, independent, non diagonally dominant, block tridiagonal equations iterative solver;

`CG` conjugate gradient evaluation by computing an approximation of the smallest eigenvalue;

`EP` integral computing by means of pseudo-random trials derived by a Monte Carlo process;

`FT` time integration of a three-dimensional partial differential equations using the FFT;

`IS` integer sorting;

`LU` triangular factorization of a matrix by a SSOR relaxation scheme;

`MG` V-cycle multigrid algorithm applied to a two dimensions discrete Poisson problem.

Table 1 presents an overview of the performance requirements of each considered kernel. In particular, the problem size, the number of iterations to be performed, and the amount of floating point operations to be computed are provided. As can be seen from the table,

Kernel	Problem size	Iterations	MFLOP
`BT`	64×64×64	200	168275.6
`CG`	14000	15	1495.4
`EP`	536870912	9	536.8
`FT`	256×256×128	6	7136.9
`IS`	8388608	10	83.7
`LU`	64×64×64	250	119298.7
`MG`	256×256×256	4	3889.3

Table 1. Static performance characterization of considered kernels.

the computing demands of the various kernels range from 83.7 MFLOP up to 168.3 GFLOP. The lower CPU requirement is for sorting (i.e., `IS`) an 8388608 distributed integer array, whereas the most computing intensive kernel (i.e., `BT`) solves a 64×64×64 block tridiagonal equation system with a 5×5 block size.

4 Performance Characterization

As a preliminary step towards the characterization of the performance of the Linux cluster we have analyzed the behavior of basic communication statements. For such a purpose, the `comms1` kernel from the ParkBench suite has been used as testbed kernel. Indeed, this kernel tests the performance of the blocking point-to-point communications by performing basic ping-pong data exchanges between two processors, namely, master and slave. Within each data exchange, the master sends a message to the slave, which sends it back to the master. Communication time, on a per message basis, is then derived by halving the time elapsed since a message sent from the master is returned by the slave. Timings related to data exchanges been collected by monitoring the kernel runs.

Figure 1 shows the communication time, over both Myrinet and Ethernet networks, as a function of the message size. Note that, the figure uses logarithmic

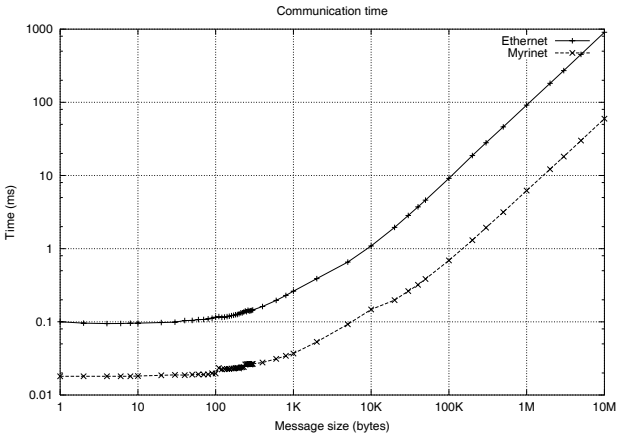

Figure 1. Communication time, as a function of the message size, over Myrinet and Ethernet.

scales on both diagram axes, that is, message size and communication time. Messages sizes up to 10 Mbytes have been considered. Communication times over Ethernet are about 5.5 times longer with respect to their Myrinet counterparts for messages shorter than 512 bytes. Larger messages result in further increases of the ratio between communication times over Ethernet and the corresponding times over Myrinet. For example, sending 1 Mbyte of data over Myrinet is about 15 times faster with respect to Ethernet.

Further insights into the performance of communication activities have been derived by investigating the individual contributions due to `MPI_Send` and

MPI_Recv. Hence, the times accounted, on a per message size basis, by the various communication statements, have been analyzed with the aim to derive analytical models for their behavior. We have observed two distinct behaviors, from short and large messages, respectively. Figure 2 outlines these behaviors by plotting the times spent in MPI_Send and MPI_Recv over Myrinet, as a function of the message size. As can be

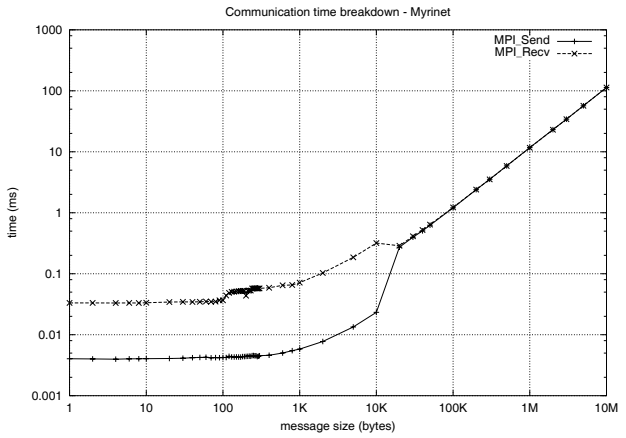

Figure 2. Times spent in MPI_Send and MPI_Recv over the Myrinet, on a per message size basis.

seen from the figure, MPI_Send requires, for sending up to 10 Kbytes of data, about 10 times less of the corresponding time accounted by MPI_Recv. Larger messages result in almost identically times accounted for either sending or receiving the data. Eager protocols, embedded in the communication library, are responsible for such a behavior. Indeed, in order to improve the communication performance, short messages are immediately sent to the destination processors, whereas large messages require a rendez-vous protocol. In this case, the sender processor has to be acknowledged, by the receiver processor, that an appropriate buffering space has been reserved. This two steps protocol results in a synchronization of MPI_Send and MPI_Recv activities. Such a synchronization results in alike times accounted by both sending and receiving activities. A similar behavior is experienced by communications over Ethernet. Our communication time models mimic these behaviors, in that we use two analytical expressions, depending on the size of the message. Numerical fitting techniques have been applied to the measured times. The analytical expression for the time spent in sending messages is:

$$t_{SEND}(n) = \begin{cases} a_0 + a_1 n & : \quad n \leq \bar{n} \\ b_0 + b_1 n & : \quad n > \bar{n} \end{cases}$$

where:

t_{SEND}	is the sending time (in μs);
n	is the message size (in bytes);
$\bar{n}$	is the eager versus rendez-vous threshold;
a_0,a_1,b_0,b_1	are the model parameters.

Table 2 reports the parameters and threshold values derived for sending messages over either Myrinet and Ethernet. Note that, parameter b_0, derived for both Myrinet and Ethernet does not represent any latency time and is used, for large message only, to improve the numerical accuracy of our model.

	Myrinet	Ethernet
a_0	4.02794	16.03866
a_1	0.00188	0.00578
b_0	57.43650	-21212.75604
b_1	0.00610	0.07966
$\bar{n}$	10K	256K

Table 2. Model parameters for the time spent in MPI_Send over both Myrinet and Ethernet networks.

Similar models have been derived for the time spent in receiving messages, that is, in performing MPI_Recvs. Indeed, the identified models have the same analytical expression of $t_{SEND}(n)$. The main difference is that a two thresholds model is most appropriate to actually represent the MPI_Recv behavior over the Myrinet. Hence, the analytical expression of the receive time is:

$$t_{RECV}(n) = \begin{cases} a_0 + a_1 n & : \quad n \leq \tilde{n} \\ c_0 + c_1 n & : \quad \tilde{n} < n \leq \bar{n} \\ b_0 + b_1 n & : \quad n > \bar{n} \end{cases}$$

	Myrinet	Ethernet
a_0	30.89969	192.69104
a_1	0.03339	0.20585
b_0	57.43650	26084.29131
b_1	0.00610	0.09761
c_0	37.23730	–
c_1	0.02935	–
$\bar{n}$	10K	256K
$\tilde{n}$	100	–

Table 3. Model parameters for time spent in MPI_Recv over both Myrinet and Ethernet networks.

Table 3 reports the parameter values. Note that, the receiving time of very short messages (i.e., up to

100 bytes) is mostly influenced by latency issues. Indeed, the amount of received data affects the receive time as much as about 10%, only.

Further insights into the characterization of the performance of the Linux cluster have been derived by analyzing the behaviors of a few kernels from the NAS Parallel Benchmarks suite. Indeed, these kernels resemble the computational cores of popular numerical algorithms widely adopted by many scientific and industrial applications. The kernels are based on data parallel paradigms aimed at distributing the computational load among all the allocated processors. Hence, the kernels are well suited for investigating the scalability of these numerical algorithms when executed on a cluster machine. An overview of the amount of computation performed by each kernel is provided in Figure 3. For each kernel, the time spent, on average, by each allocated processor in computation activities is plotted as a function of the number of allocated processors themselves. As can be seen from the figure,

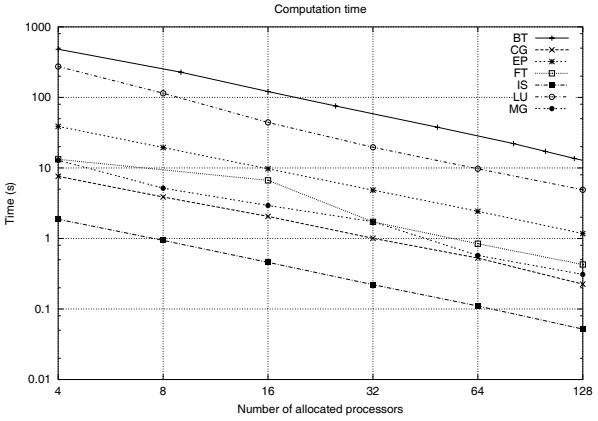

Figure 3. Computation time, on a per kernels basis, as a function of the number of allocated processors.

all kernels are characterized by regular behaviors. Indeed, the computation time of all kernels, but `FT`, can be expressed by:

$$t_{comp}(p) = a_0 + \frac{a_1}{p}$$

where p is the number of allocated processors and a_0, a_1 are the model parameters, derived on a per kernel basis. Computation times of `FT` kernel can be better expressed by:

$$t_{comp}(p) = a_0 + \frac{a_1}{p} - \frac{a_2}{p^2}$$

where a_2 addresses an extra time reduction factor. Such a factor is due to the computation, on each processor, of local two dimensional FFT which do not have been optimized for the Pentium Intel architecture.

On the other hand, the time spent by the allocated processors in communication activities plays a critical role by limiting the overall performance of the various kernels. In what follows, we have analyzed such a time for runs over either the Myrinet and the Ethernet networks. Figure 4 depicts the communication time experienced by each kernel as a function of the number of allocated processors. In particular, Fig. $4(a)$ refers to runs over Myrinet, whereas Fig. $4(b)$ refers to runs over Ethernet. As can be seen from the figure, the behavior of communication times over Myrinet is quite regular, even when large number of processors are allocated. On the other hand, the Ethernet network limits the performance of the kernels, in that communication times might overwhelm their computation time counterparts. For example, allocating 64 processors to the `FT` run over Ethernet results in an execution time of $11.21s$. The `FT` execution time with 32 processors is $8.90s$, only. Hence, although computation time decreases from $1.72s$ (32 processors) to $0.90s$ (64 processors), the communication time increases from $7.19s$ to $10.32s$. Moreover, the times spent in communication activities over Myrinet are shorter than their Ethernet counterparts for all kernels, but `LU`. An in-depth analysis of the `LU` behavior has identified a large amount of short blocking communications that are performed during the kernel execution. In particular, blocking receives (i.e., `MPI_Recv`) account for about 80% of the communication time, although they collect only 30% of received data. On the other hand, Ethernet performs very well short data transfers. Indeed, the 46500 blocking receives issued by the 16 processors allocated to an `LU` execution, account for $14.48s$ for communication over Ethernet. When communication takes place over Myrinet such receives account for $49.99s$.

As a final step towards the performance characterization, we have applied statistical clustering techniques to the performance measurements collected for each kernel run. The aim of statistical cluster analysis is to identify sets of runs which are characterized by similar performance measures. In what follows we define the communication rate as the ratio of the overall volume of exchanged data over the wall clock communication time. Each run is then described by a set of performance parameters, that is, the communication rate, the volume of exchanged data and the communication frequency. Parameters related to 93 runs over either Myrinet and Ethernet networks, varying the number of allocated processors have been considered. The an-

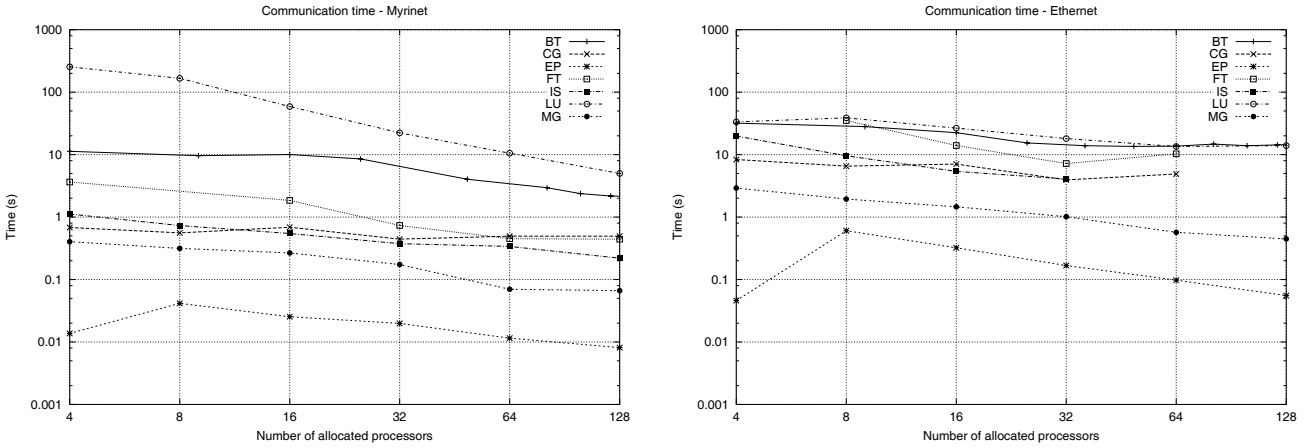

Figure 4. Communication time of each kernel over Myrinet (a) and Ethernet (b) as a function of the number of allocated processors.

alyzed runs have been subdivided into three clusters by means of statistical clustering. Figure 5 depicts the considered runs as points into a 3D-space where the axes are the communication ratio, the overall amount of exchanged data, and the communication frequency. The various point shapes highlight the identified clus-

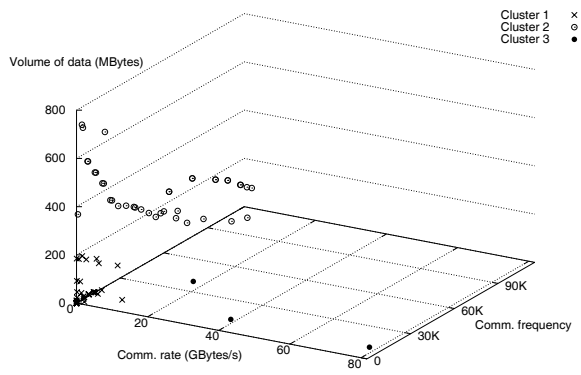

Figure 5. Performance measures of analyzed kernels highlighting identified clusters.

ters. Indeed, crosses, white and solid black rounds refer to run belonging to Cluster 1, Cluster 2, and Cluster 3, respectively. Table 4 reports the statistic centroids, that is the average values of parameters, of each identified cluster, as well as the number of kernel runs belonging to it. Note that, Cluster 3 consists of only three

	Cluster 1 (51 runs)	Cluster 2 (39 runs)	Cluster 3 (3 runs)
Comm. rate (GBbyte/s)	1.00	2.78	52.12
Vol. of Data (MBytes)	35.55	298.32	87.63
Communication Frequency	2310.75	50486.80	1823.00

Table 4. Statistical centroids of the identified clusters.

kernel runs which are characterized by high communication rate and low communication frequency. Namely, they are `MG` runs over Myrinet, with 32, 64, and 128 allocated processors. On the other hand, `BT` and `LU` runs over both interconnection networks belong to Cluster 2. These runs are characterized by very large communication frequencies and volumes of exchanged data. `CG`, `EP`, `FT`, and `IS` runs belong to Cluster 1. The remaining `MG` runs belong to Cluster 1 for all the processor sets but one with 4 allocated processors which belongs to Cluster 1.

5 Conclusions

Cluster computing has emerged as an alternative approach to deliver high performance to scientific and industrial applications. Solutions, ranging from personal computers and workstations connected by commodity

local area networks up to large dedicated clusters with specialized hardware have been proposed. Cost constraints and the large availability of open source software including the development and runtime environments enlarge the framework of cluster machines users. It is then important to characterize the performance of cluster machines and to investigate the scalability of numerical algorithms when large number of processors are allocated.

In this paper we have presented a performance characterization of a large commercial cluster, that is a 520 processors IBM NetFinity. In particular, we have analyzed the performance measures collected while benchmarking the machine with kernels from the NAS Parallel Benchmarks suite. The impact of a dedicated interconnection network, based on Myrinet switches, with respect to the standard Ethernet network, has been addressed. The scalability of all considered kernels benefits from Myrinet network, although the corresponding execution times do not always reflect the the superiority of Myrinet with respect to Ethernet network. Indeed, we have unexpectedly found out that when a large number of blocking communication requests are issued by a small number of processors Ethernet performance is superior.

Future works will be dedicated to the analysis of the performance overhead due to various processors allocation strategies on the overall application performance as well as to investigate the performance of real live applications on cluster machines.

Acknowledgments

This work was partially supported by the Italian Research Council (CNR) under the Project "Agenzia 2000 - Progetto Giovani". This research, in part conducted at the Maui High Performance Computing Center, was also sponsored in part by the Air Force Research Laboratory, Air Force Materiel Command, USAF, under cooperative agreement number UNIVY-0282-U00. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory, the U.S. Government, the University of Hawaii, or the Maui High Performance Computing Center.

References

[1] R. Alfieri et al. Status of APE projects. *Nuclear Physics B*, 94:846–853, 2001.

[2] T. Anderson, D. Culler, D. Patterson, and the NOW Team. A Case for NOW (Networks of Workstations). *IEEE Micro*, 15(1):54–64, 1995.

[3] E. B. Bal et al. The Distributed ASCI supercomputer project. *Operating Systems Review*, 34(4):76–96, 2000.

[4] A. Barak, I. Gilderman, and I. Metrik. Performance of the communication layers of TCP/IP with the Myrinet gigabit LAN. *Computer Communications*, 22:989–997, 1999.

[5] A. Barak and O. La'adan. The MOSIX multicomputer operating system for high performance cluster computing. *Future Generation Computer Systems*, 13:361–372, 1998.

[6] The Beowulf Project. http://www.beowulf.org, 2001.

[7] The Berkeley Network of Workstations. http://now.cs.berkeley.edu, 2002.

[8] R. Brightwell and S. Plimpton. Scalability and Performance of Two Large Linux Clusters. *Journal of Parallel and Distributed Computing*, 61(11):1546–1569, 2001.

[9] F. Capello, O. Richard, and D. Etiemble. Understanding performance of SMP cluster running MPI programs. *Future Generation Computer Systems*, 17:711–720, 2001.

[10] T. Deng and A. Korobka. The performance of a supercomputer built with commodity components. *Parallel Computing*, 27:91–108, 2001.

[11] S. Donaldson, J. Hill, and D. Skillicorn. BSP clusters: High performance, reliable and very low cost. *Parallel Computing*, 26:199–242, 2000.

[12] M. Golbiewski and J. Larsson. MPI-2 One-Sided Communications on a Giganet SMP Cluster. In *Lecture Notes in Computer Science*, volume 2131, pages 16–23, 2001.

[13] A. Grimshaw, A. Ferrari, F. Knabe, and M. Humphrey. Wide area computing: resource sharing on a large scale. *IEEE Computer*, 32(5):29–37, 1999.

[14] D. Houzet and M. Albegne. A shared memory model on a cluster of PCs. *Microprocessors and Microsystems*, 23:125–134, 1999.

[15] J. Hsieh, T. Leng, V. Mashayekhi, and R. Rooholamini. Architectural and Performance Evaluation of GigaNet and Myrinet Interconnects on Clusters of Small-Scale SMP Servers. In *Proceedings of Supercomputing 2000*. IEEE Computer Society Press, 2000.

[16] L. P. Huse and H. Bugge. High-End Computing on SHV Workstations Connected with High Performance Network. *Lecture Notes in Computer Science*, 1947:324–331, 2001.

[17] Cluster Computing White Paper. http://www.dcs.port.ac.uk/~mab/tfcc/WhitePaper/final-paper.pdf, 2000.

[18] M. S. Warren and D. J. Becker and M. P. Goda and J. K. Salmon and T. Sterling. Parallel Supercomputing with Commodity Components. In H. R. Arabnia, editor, *Proc. of the Intnl. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*, pages 1372–1381, 1997.

[19] Maui High Performance Computing Center. http://www.mhpcc.edu, 2002.

[20] T. Ridge, D. Becker, P. Merkey, and T. Sterling. Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs. In *Aerospace Conference*, volume 2, pages 79–91. IEEE Press, 1997.

[21] V. Sunderam, J. Dongarra, A. Geist, and R. Manchek. The PVM Concurrent Computing System: Evolution, Experiences, and Trends. *Parallel Computing*, 20(4):532–547, 1993.

[22] S. Vazhkudai, J. Syed, and T. Maginnis. PODOS - The design and implementation of a performance oriented Linux cluster. *Future Generation Computer Systems*, 18(1):335–352, 2002.

[23] Virtual Interface Architecture. http://www.viarch.org, 1997.

[24] D. Womble, S. Dosanjh, B. Hendrickson, M. Heroux, S. Plimpton, J. Tomkins, and D. Greenberg. Massively parallel computing: A Sandia perspective. *Parallel Computing*, 25:1853–1876, 1999.