Benchmarking of a commercial Linux cluster

Daniele Tessera
Dipartimento di Informatica e Sistemistica,
Università degli Studi di Pavia, Italy.

Abstract

Cluster computing has emerged as a viable alternative to massively parallel supercomputers to cope with the computing demands of scientists and HPC application developers. Cluster machines, built on top of Commodity Off The Shelf components and based on open source software are becoming very popular. It is thus important to characterize the performance of these machines and to investigate the scalability of numerical algorithms when large numbers of processors are allocated.

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 aim of this study is to evaluate the performance benefits to a few numerical applications resulting from specialized high bandwidth communication components, i.e., Myrinet switches, over a typical Beowulf cluster based on Ethernet networking. Our performance characterization study focused on the behavior of a few numerical kernels and of a climate model benchmark whose performance relies on complex iterations among various parallel algorithms. We investigated the scalability of the benchmarks under various hardware and software configurations.

The study of overheads on the overall performance of a few numerical kernels deriving from different processor allocation policies concludes this paper.

1 Introduction

Exploiting the cumulative computing capability of workstations and PCs has been pursued for a long time, as an inexpensive solution for delivering high performance to many scientific and industrial applications. The underlying idea is to allow parallel applications to be executed over a set of independent systems connected via local area networks [20]. Various approaches such as distributed concurrent computing [27], network of workstations [2, 7], Beowulf architectures [6, 26], and the multicomputer operating system [5, 16] have been proposed to design machines with better performance and flexibility.

Cluster machines, built on top of commodity off the shelf components and initially developed by research centers as prototype machines are nowadays becoming very popular. The increasing performance of commodity components and the availability of interconnection networks able to link a large number of such components have fueled the development of cluster machines. The development of specialized hardware and software components tuned for matching the demands of HPC applications have resulted in cluster machines approaching the supercomputers' performance and ranked among the top 500 most powerful supercomputers. Many studies [1, 3, 4, 9, 12, 14, 19, 21, 28, 29] present the characteristics of different cluster machines.

The fact that hardware vendors now offer scalable cluster machines also testifies to the maturity of cluster computing as a cost effective solution for the high performance needs of scientists and numerical engineers.

The success [15] of cluster architectures is also related to the large availability of their building components, as well as their community support. The maturity of open source software which addresses the operating system, commodity libraries, and development tools, has significantly contributed to that popularity. Indeed, open source software eases the portability of application code between different cluster architectures by providing a flexible common environment for application development. Moreover, parallel applications developed for traditional massively parallel supercomputers can be easily adapted to run on cluster machines.

It is thus important to evaluate the performance offered by these machines. This paper presents a detailed study of the performance achieved by the large IBM NetFinity cluster at the Maui High Performance Computing Center [23] on various scientific benchmarks. Although we have analyzed the performance of a specific machine, our results have a broader applicability. Indeed, the IBM NetFinity adopts a typical cluster architecture based on Intel Pentium processors interconnected via both Myrinet and Ethernet networks, and the Linux operating system.

The aim of our study is to characterize cluster performance under various scientific workload and system configurations. For such a purpose, we have analyzed the times spent in communication, computation, and resource contentions activities when a large number of processors has been allocated to different numerical kernels.

The paper is organized as follows. Section 2 provides an overview of the hardware and software testbed environments and describes our methodological approach to the performance characterization. Section 3 describes the performance achieved by the analyzed benchmarks. The impact of different processor allocation policies on the benchmark performance is presented in Section 4. Finally, Section 5 summarizes the performance behavior of the Linux cluster and outline future works.

2 Experimental Environment and Methodology

A large number of cluster machines are nowadays proposed as a viable alternative to traditional parallel supercomputers for scientific and industrial computing. It is thus important to evaluate the performance that cluster machines actually deliver to number intensive applications.

The objective of this study is to benchmark a Linux cluster with large numbers of processors allocated to very popular scientific and numerical testbed kernels. For such a purpose, we have investigated the performance of a state of the art Linux supercluster, that is the IBM NetFinity [25] at the Maui High Performance Computing Center. This machine is a cluster composed of 260 nodes running Linux as operating system. Each node houses 1Gbyte of memory and two Intel Pentium III processors clocked at 933Mhz. Nodes are connected via both a high performance Myrinet switch [24] and a Fast 100Mbps Ethernet network. The cluster ranks among the largest and most powerful Linux clusters. The HPC cluster environment includes the Maui Scheduler Open Cluster Software [22], for executing applications on dedicated nodes, and optimized middleware drivers for the Myrinet interconnection network. Communications over Ethernet are managed by the p4 [10] facility embedded in the MPI communication library.

As testbed kernels we have considered a few kernels from the well known ParkBench [18] and NAS Parallel Benchmarks [30] suites, as well as a more complex climate model benchmark, that is the PSTSWM v6.7.2 [8] from the Oak Ridge National Laboratory. These kernels resemble the performance of very popular numerical algorithms which are widely adopted by many scientific and industrial applications. A brief description of each analyzed kernel is provided in Table 1.

The analyzed kernels range from the simple Embarrassingly Parallel routine (i.e., `EP`) consisting of 306 lines of C language up to a quite complex climate model (i.e., `pstswm`) composed of 204 routines accounting for 30,055 lines of Fortran 77.

Our approach to the benchmarking and performance evaluation of the IBM NetFinity is based on an experimental approach. The actual performance achieved by this machine has been measured by monitoring the executions of the testbed kernels. Such a monitoring consists of collecting the times spent in computation and communication activities by the main routines of each analyzed kernel. An ad-hoc monitoring system has been developed in order to minimize the perturbations, in kernel executions due to monitoring activities. We have then monitored the execution of various kernels varying the number of processors allocated, the problem size, the processor allocation policy, and the interconnection network.

Table 1: Overview of the analyzed kernels

Kernel	Description
comms1	ping-pong communications between a pair of processors using explicit send and receive directives. (ParkBench suite)
comms2	ping-pong communications between a pair of processors using full duplex point-to-point data exchanges. (ParkBench suite)
BT	multiple, independent, non diagonally dominant, block tridiagonal equations iterative solver. (NAS suite)
EP	integral computing by means of pesudorandom trials derived by applying a typical Monte Carlo process. (NAS suite)
LU	triangular factorization of a matrix by a SSOR relaxation schema. (NAS suite)
MG	V-cycle multigrid algorithm applied to a two dimensional discrete Poisson problem. (NAS suite)
pstswm	nonlinear shallow water equations on a rotating sphere. (PSTSWM suite)

The measurements, collected by such a monitoring activity, have been analyzed with the aim to benchmark the machine and to derive workload models [11, 13]. The derived performance figures have been related to machine and benchmark characteristics.

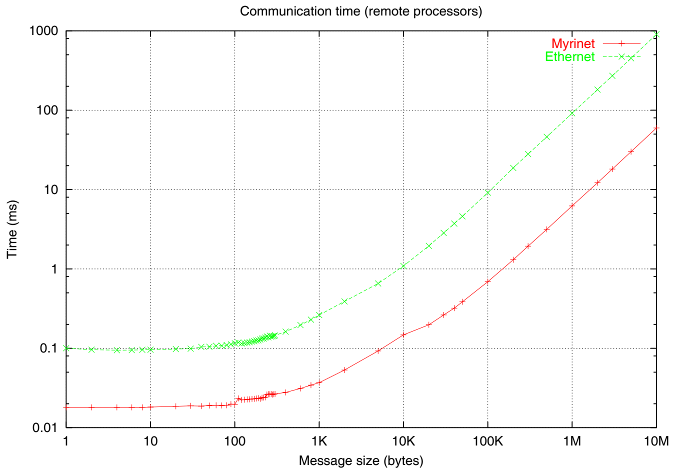
Our methodology is based on a bottom-up characterization of the performance of the Linux cluster. As a starting point, basic communication performance on node-to-node communications will be analyzed. The performance of a few numerical algorithms, such as those implemented by the NAS Parallel Benchmark kernels have been investigated. Further insights into the performance of the machine are then derived by analyzing the behavior of a complex benchmark, that is, the pstswm climate model simulation.

Note that our study is aimed at evaluating the behavior of production runs of various kernels from measurements collected on testbed runs. The main difference between testbed and production runs is that testbed runs compute only a few time steps of the solution. Hence, statistical techniques have been applied to measured timings in order to “sanitize” them, that is, to minimize the impact of non deterministic random effects. Measurements have then repeated several times, depending on individual kernel characteristics. Statistical outlier deletion, based on the 99^{th} percentile, was then applied to these measures with the aim to discard anomalous values. Mean values and coefficients of variation have been used to analyzes the actual behavior of the phenomenon under investigation.

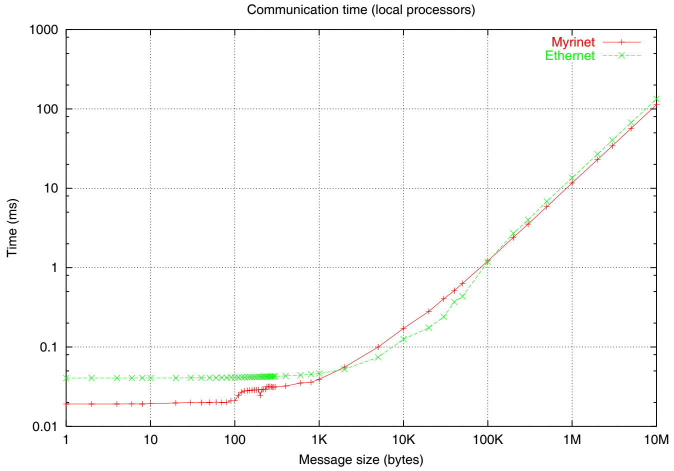
Performance models, summarizing the behavior of measured activities, have been derived by means of statistical clustering techniques [17]. Each kernel execution is then described by a component, that is, a tuple of parameters which represent various timings and performance figures related to that execution. Clustering partitions all components into homogeneous groups, i.e., clusters, which are characterized by similar parameters' values. These cluster are described by both their centroids, which are the averages of the component values and the corresponding indices of variation (e.g., the standard deviations). Each centroid provides a compact description of all the components belonging to its group. We have applied clustering techniques to identify executions with similar behaviors. These behaviors have been related with machine and application parameters.

3 Performance Analysis

Communication cost, that is, the time spent by the processors in exchanging data, is among the most critical factors which affect the overall performance of parallel and distributed applications. Hence, as a preliminary step to benchmark the Linux cluster we have analyzed the performance of basic MPI communication directives. The basic ping-pong communication kernels, from the ParkBench suite have been used for such a purpose. In particular, we have analyzed the performance of `comms1` kernel which is based on blocking communication protocols, that is, MPI_Send and MPI_Recv and the behavior of the `comms2` kernel, based on point-to-point data redistribution, that is, the MPI_Sendrecv directive. In the `comms1` kernel, a processor (i.e., the master), sends a message to its neighbor (i.e., the slave) which, in turn, after receiving the message sends it back to the sender. Communication time, on a per message basis, is then derived by halving the time elapsed on the master from the start of MPI_Send to the end of the matching MPI_Recv. Note that since each IBM NetFinity node houses two processors, the performance of the communication activities also depends on physical location issues. Indeed, communications between processors located on the same node (i.e., local processors) are based on shared memory paradigms. whereas communications among processors belonging to different nodes (i.e., remote processors) are performed over the interconnection network. Communication libraries, such as MPI, automatically uses of the most appropriate communication paradigms. Figure 1 shows the communication times, as a function of the message size, for remote (a) and local (b) processors. Red and green plot refer to communication over Myrinet and Ethernet, respectively. Logarithmic scales have been used on both axes to represent message size ranging from one byte up to 10MB and communication time varying from tens of microseconds up to hundreds of milliseconds. From the figure it can be seen that Myrinet outperforms Ethernet in remote processors communications. Small messages, i.e., less than 100 bytes experience almost only latency times, that is, about $18\mu s$ and $97\mu s$ for communication over Myrinet and Ethernet, respectively. When exchanging large messages, i.e., about 100KB, Myrinet is faster than Ethernet by a factor ranging from 5 to 15, depending on the length of the message. We have noticed an unexpected result in the Myrinet behavior: messages exchanged between local processors account for longer times with respect to messages exchanged between remote processors. For example, commu-



(a)



(a)

Figure 1: Communication times over Myrinet (red crosses) and Ethernet (green stars) networks for both remote (a) and local (b) processors.

nication time for small messages between local processors are 10% longer than for small messages between remote processors. This difference increases as message size increases. Exchanging a message of 1MB of data when communication take place between remote and local processors requires 6.23ms and 11.67ms, respectively.

Further insights into the behavior of the communication performance can be derived by comparing the communication time accounted by explicit MPI_Send and MPI_Recv directives with the corresponding time of the MPI_Sendrecv directive. Figure 2 plots these times, on a per message size, for communication over Myrinet. We noticed that the explicit MPI_Send/MPI_Recv directives achieve better performance than MPI_Sendrecv. Indeed, it experienced an overhead of about 20% with respect to explicit MPI_Send and MPI_Recv directives for message size up to 10KB. As message size increases so does the difference in the performance. For example, exchanging 10MB of data with MPI_Send and MPI_Recv requires 59.67ms, whereas the corresponding MPI_Sendrecv accounts for 136.41ms. Communications over Ethernet with MPI_Sendrecv also result in a performance loss. However such a performance loss is less marked, being in the order of about 8% for message up to 100KB.

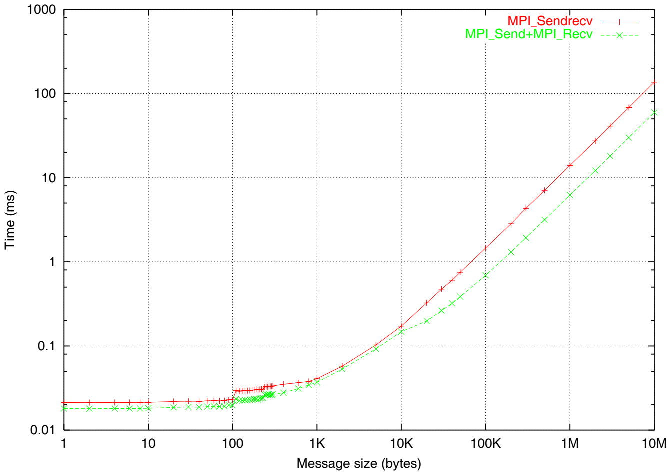


Figure 2: Communication time, on a per message size basis, for MPI_Sendrecv (red crosses) and MPI_Send and MPI_Recv (green stars) over Myrinet

As a further step towards the characterization of the performance of the Linux cluster we have analyzed the behavior of the performance achieved by a few well know numerical benchmarks. In particular, Section 3.1 presents the performance analysis of a few numerical kernels from the NAS Parallel Benchmarks, whereas the performance of the `pstswm` climate model is discussed in Section 3.2.

3.1 Performance analysis of a few numerical algorithms

The NAS Parallel Benchmarks is a well know suite of very popular numerical algorithms which resemble the performance of the computational cores of many industrial and scientific applications. In this section we present a detailed performance study of a few NAS kernels with the aim at investigating the differences in the performance behavior due to the characteristics of the interconnection network. As a preliminary step we have analyzed the performance requirements of these kernels. Table 2 provides a static description of the amount of computation performed by each kernel. As can be seen, the computing needs of the kernels ranges from 536.8MFLOP, for the embarrassingly parallel EP kernel, up to 168.2GFLOP required by the BT equation solver. We have analyzed the execution of each kernel over both Myrinet and Ethernet networks, varying the number of allocated processors from 4 to 128.

Table 2: Static characterization of analyzed NAS kernels.

Kernel	Problem size	Iterations	MFLOP
BT	64×64×64	200	168275.6
EP	536870912	9	536.8
LU	64×64×64	250	119298.7
MG	256×256×256	4	3889.3

Figure 3 plots the time spent in computation activities for each execution. The color identifies the kernel, while the shape of the points refers to execution over Myrinet (square) or over Ethernet (stars). Note that both the number of allocated processors and the computation times are plotted with logarithmic scales. As can be seen from the figure, the computation times accounted by the various kernels scale proportionally with the number of processors allocated to their executions. Indeed, these computation times can be approximated by:

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

where $t_{comp}(p)$ is the computation time, expressed in milliseconds, p is the number of allocated processors ($4 \leq p \leq 128$), and a_0 is the model parameter. The value this parameter is derived for each kernel: $a_0 = 1859.514$ for BT, $a_0 = 154.556$ for EP, $a_0 = 934.971$ for EP, $a_0 = 154.556$ for LU, and $a_0 = 46.212$ for MG. Such a parameter is related to the amount of computation to be distributed among the allocated processors.

In order to get further insights into the behavior of the performance delivered by the IBM NetFinity we have then focused our analysis on the times accounted for communication activities by the four kernels. For such a purpose, Figure 4 shows, for each kernel execution, the time spent in communication as a function of the number of allocated processors. Colors and symbol shapes are used as in Figure 3.

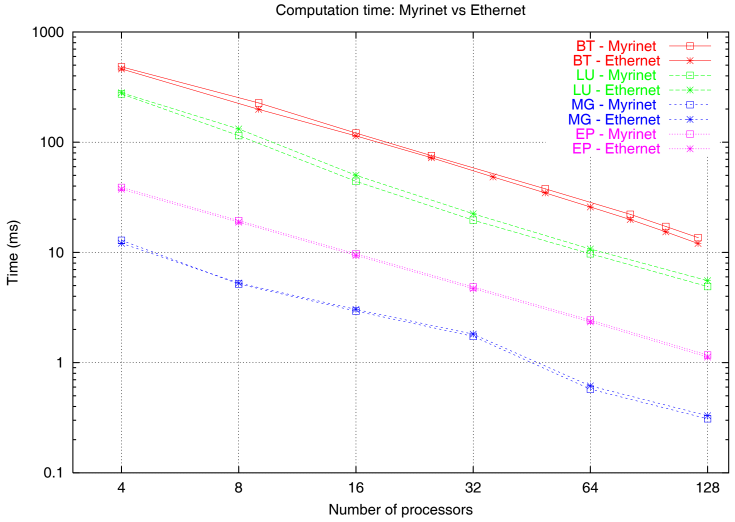


Figure 3: Computation times, as a function of the number of allocated processors, accounted by each kernel executed over either Myrinet and Ethernet.

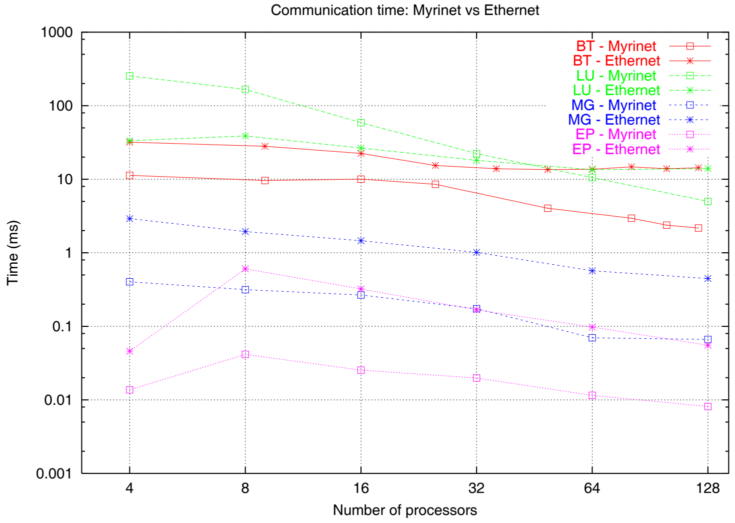


Figure 4: Communication times, accounted by each kernel over either Myrinet or Ethernet, as a function of the number of allocated processors.

Communications over Myrinet are at least 5 times faster that their Ethernet counterparts for almost all kernel executions. A rather surprising result shown in the figure is that, executions over Myrinet of the LU kernels with up to 32 allocated processors experience longer communication times than their Ethernet counterparts. To explain this phenomenon we have analyzed the communication activities performed by the LU executions. Two communication policies are responsible for all data exchanges, namely blocking send/blocking receive and blocking send/non blocking receives. Hence, all data is sent by the MPI_Send directive, whereas on the destination side, MPI_Recv (i.e., blocking receive) and MPI_Irecv together with MPI_Wait (i.e., non blocking receive) are used. In what follows we focus our analysis to the executions with 8 processors. Figure 5 summarizes the volume of data managed by each communication protocol. As can be seen from the figure, most of the data is received with the non blocking communication protocol. The kernel uses these communication protocols with very different message sizes. Indeed, the average message size of MPI_Recv is about 1KB, whereas MPI_Irecv collects messages with an average size of about 130KB.

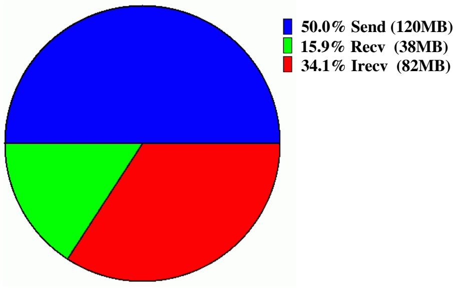


Figure 5: Volume of data transfered by each protocol.

The communication activities, presented in Figure 5 account for 166.20s and 38.73s on execution over Myrinet and Ethernet, respectively.

Figure 6 shows the breakdown of the communication times on a per protocol basis. The percentage reported for each protocol refers to the fraction of the overall communication time accounted by that protocol. The length of each bar is proportional to the time accounted by the given protocol. The main difference between the two executions is in the time spent by the MPI_Recv protocol. Indeed, in the execution over Myrinet such a protocol accounts for 137.1s, whereas in the corresponding execution over Ethernet it accounts for 19.6s only. Note that various factors are involved in these communication times, such as, the peculiarities of the middleware communication software, hardware characteristics, the physical location of the allocated nodes, as well as delays in the behavior of the allocated processors. In order to derive further insights into the performance of the communication protocols we have analyzed the behavior of the LU behavior with larger numbers of processors allocated. For example, on the execution with 128

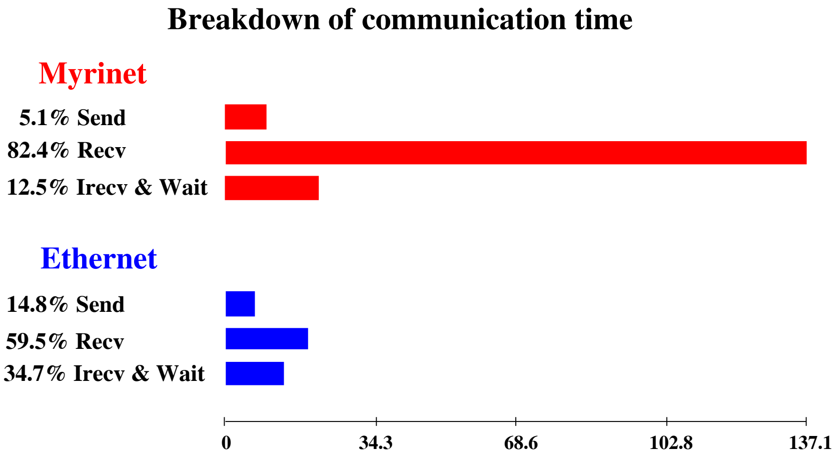


Figure 6: Breakdown of the communication time accounted by executions of the LU kernel with 8 processors over Myrinet and Ethernet.

processors the performance experienced by Myrinet is superior than its Ethernet counterpart. Figure 7 shows the time spent in each communication protocol for such an execution over both the interconnection networks.

The performance of MPI_Recv over Myrinet is greatly increased in the execution with 128 processors with respect to the 8 processors one. In the execution with 128 processors MPI_Recv occurred about 56,000 times with an average message size of 235 bytes. We can identify in the reduction of the message size a possible explanation of this very different performance behavior. This conclusion is also substantiated by the comparison with the performance achieved by the execution of the LU kernel with 4 processors. In this execution MPI_Recv occurred 31,000 times for collecting messages of about 1240 bytes. When this communication protocol is used over Myrinet its performance is about 10 time worse than its performance over Ethernet.

3.2 Performance Analysis of a climate benchmark

The performance achieved by real-live applications relies on the complex iterations of their embedded algorithms. In this section we discuss the behavior of the IBM Linux cluster on a complex climate model benchmark, that is, the pstswm kernel from the Oak Ridge National Laboratory. Many numerical algorithms, including distributed FFT, Gaussian integrations, and spectral transformation methods are used to solve the non linear shallow water equations. Performance figures related to various testbed simulations, varying the size of the physical grid, the number of simulated time steps, as well as the number of allocated processors, have been analyzed. In particular, we have analyzed grid size ranging from $128\times64\times$NVER

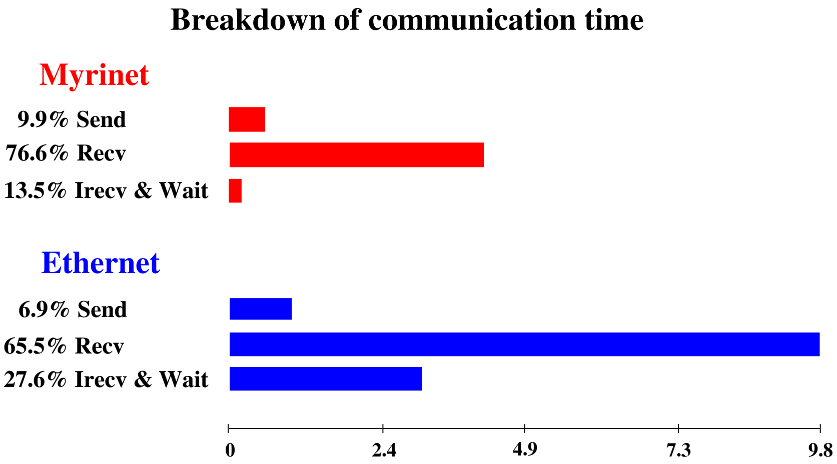


Figure 7: Breakdown of the communication time accounted by executions of the `LU` kernel with 128 processors over Myrinet and Ethernet.

to 512×128×`VER`, with `NVER` being the number of vertical levels and varying from 1 to 16. The performance requirement for simulating one time step of these two grid bounds with `NVER`=1 is 4MFLOP and 153MFLOP, respectively.

As a preliminary overview of the performance achieved by the Linux cluster, we have benchmarked the machine with a set of simulations varying the simulation grid and the number of time steps. Figure 8 shows the computation and communication times accounted by the full set of benchmarks (i.e., 76 simulations), as a function of the number of processors allocated to the benchmark executions over both Myrinet and Ethernet. Color identifies the interconnection network (red for Myrinet and green for Ethernet), whereas crosses and stars refer to computation and communication times, respectively. Computation times accounted over both the two interconnection network are identical. On the other hand, there is a large difference (i.e., a factor of about five) in the times spent in communication activities over the two interconnection networks. The longer communication times for kernel executions over Ethernet limit the number of processors to be allocated. Indeed, allocating more than 64 processors does not lead to any performance improvement.

Figure 9 shows the breakdown of execution times of each analyzed simulation. Timings are related to an execution with 32 processors over Myrinet. Colors highlight the contributions due to computation (blue) and communication activities (red). Problem IDs ranging from 1 to 16 refer to small grid (e.g., 128×64) simulations in which the number of vertical levels is varied from 1 to 16. Problem IDs from 17 up to 32 refer to the longer simulations of these grids. Problems from 33 to 64 are similar to problems 1 to 32 but with intermediate grid (e.g., 256×128), whereas IDs from 65 to 70 refer to large grids (e.g., 512×256). The

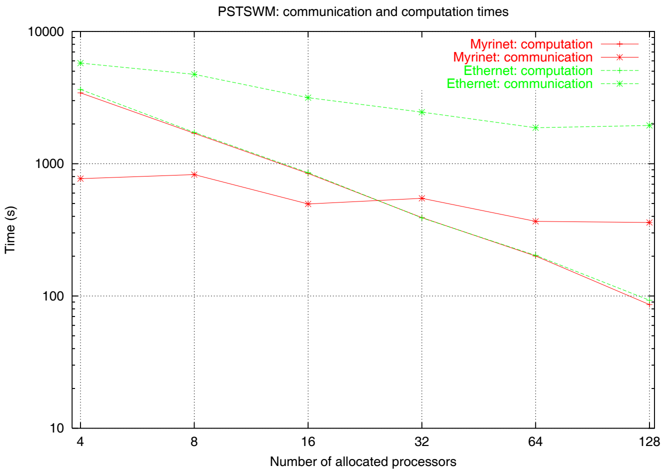


Figure 8: Computation and communication times as a function of the number of allocated processors.

last 6 problems refer to small grid with different physical simulation parameters. Several performance parameters have been measured for each simulation. These measures have then been normalized by considering their averages on a per simulation time step basis. We have applied statistical clustering to the measures collected on all the 456 simulations executed varying the number of allocate processors from 4 up to 128. Two clusters characterize all the simulations executed over either Myrinet or Ethernet. Table 3 presents the centroids of each cluster. The parameters n_sim, n_comms, and ndata refer to the number of simulation within each cluster, and to the average number of communications and to the average volume of exchanged data, in each of these simulations. Time parameters, that is, t_{comp}, t_{send}, and t_{recv} refer to the average time accounted by each simulation

Table 3: Centroids of the identified clusters.

	Myrinet		Ethernet	
	Cluster 1/2	Cluster 2/2	Cluster 1/2	Cluster 2/2
n_sim	384	72	408	48
$t_{\mathtt{comp}}$	52.65	642.13	68.00	862.07
$t_{\mathtt{send}}$	5.43	30.20	14.25	30.57
$t_{\mathtt{recv}}$	3.38	210.58	24.54	953.30
n_comms	90	37	88	29
ndata (MB)	1.59	9.53	1.83	11.44

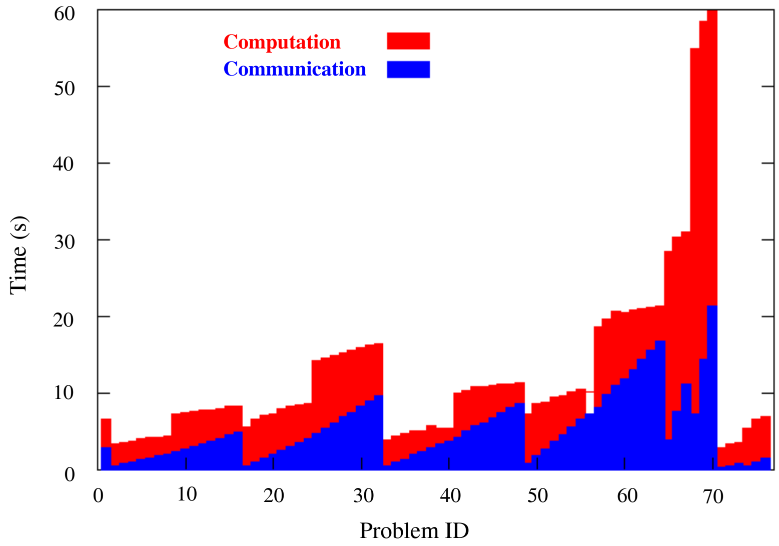


Figure 9: Computation and communication times for all the analyzed problems for an execution of the `pstswm` over Myrinet with 32 processors.

for computing, transmitting, and receiving activities, respectively.

From the table, we can see that most of the simulations (both on Myrinet and Ethernet executions) are grouped in the first cluster. These simulations are characterized by short computation and communication times. Nevertheless, these simulations issue a larger number of communication directives with respect to their Cluster 2 counterparts. In turn, although the simulations within Cluster 2 issue fewer communications they exchange about 6 times the volume of data exchanged by the simulations within Cluster 1.

4 Overheads of Processor Allocation Policies

As a further characterization of the performance achieved by the IBM NetFinity cluster, we have studied the impact of the processor allocation policies on the various kernel execution times. As previously stated, the cluster consists of 260 nodes, each of them housing two processors. MPI applications, in turn, are composed of tasks to be allocated to the various processors. Two allocation policies are thus available: either one or two application tasks can be assigned to each node. When one task only is assigned to each node, both processors might concur to the execution of such a task. Indeed, when a task requires operating system calls, they might activate some concurrent processes/threads. In this case, the operating system will then schedule all these processes/threads on both processors. For example, communication routines might require network daemon services for managing the communication activities, such as, in the `p4` parallel programming library used for communications over Ethernet.

We have seen that the behavior of the embarrassingly parallel `EP` kernel over Myrinet does not experience any significative performance difference between the two allocation policies until a large number of processors is allocated. Indeed, we have noticed that varying the allocation policies results in differences smaller than 5% of the overall execution time. On the other hand, runs with more then 32 processors, experience a dramatic increase of the wall clock times when both the processors available on each node are allocated. For example, a run with 64 nodes (i.e., 128 processors) is characterized by an overall execution time of $2.511s$, whereas the execution times decreases to $1.179s$ in a run with 128 processors over 128 nodes. This difference is not explained in terms of a decrease of the communication time that is equal to $1.208s$ and $8ms$, respectively. Pure computation time, such as, the time spent by `EP` in computing Gaussian deviates, increases of about 12% in runs with 64 nodes. Such an increase can be explained in terms of both the contentions due to accesses to the memory which is shared among the two processors of each node, and the concurrent execution of operating system processes. It is worth noting that when a node has a “spare” processor, such a processor can be used to execute operating system processes. Hence, being fixed the number of allocated nodes, the benefit of sharing the computation among all the available processors might not lead to any performance improvement. For example, allocating 64 processors on 64 nodes results in an wall clock time of $2.442s$, whereas allocating all the 128 processors available on the considered 64 nodes results in an increased wall clock time of $2.511s$.

Runs of `EP` over Ethernet also lead to similar results, although significant performance degradations are experienced with smaller numbers of processors. For example, runs with 32 processors over 16 and 32 nodes results in wall clock times of $3.452s$ and $7.842s$, respectively.

Figure 10 shows the profiling of two runs of the `BT` kernel with 100 processors. For each run, the figure shows the four most time consuming routines, which account for 99% of the overall execution time. The size of the problem solved by each run is the same, that is, a $162 \times 162 \times 162$ matrix. The two runs differ in the number of node the allocated processors belong to, that is, 100 and 50 nodes, respectively.

Note that the run on 50 nodes is 44% longer than the run on 100 nodes. As can be seen from the figure, this difference is mainly due to increases in the times spent in computation activities. Contentions for memory access within each node are responsible for this performance degradation.

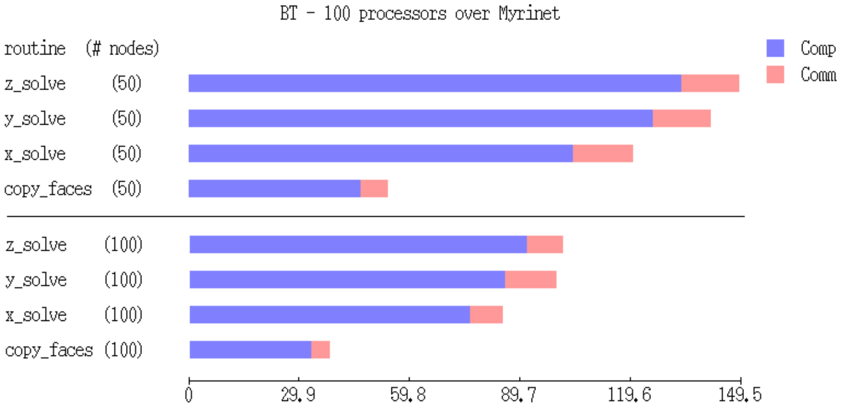


Figure 10: Profiling and breakdown of the execution times in their computation and communication components of two BT runs with 100 processors, allocated on 50 and 100 nodes, respectively.

5 Conclusions

Cluster computing has emerged as a viable alternative to massively parallel supercomputers to cope with the computing demands of scientists and HPC application developers. In this paper we have presented an experimental evaluation study of the performance that a typical cluster machine actually delivers to scientific and industrial applications. Statistical techniques, such as clustering and fitting, have been used to characterize the actual performance of the machine.

In particular, we have investigated the performance of a few numerical algorithms, as well as the impact of different processor allocation policies on their performance. As a rule of thumb, Myrinet outperforms the standard Ethernet interconnection network. However, when a large number of messages is simultaneously exchanged among a few processors Myrinet experiences severe performance degradations. An in depth analysis of this phenomenon has identified that the performance of simultaneously concurrent blocking receives over Myrinet is strongly influenced by the size of exchanged messages. Note that as already stated various factors, such as hardware, software and synchronization issues are responsible for such an unexpected outcome.

Another rather surprising outcome has been derived from our analysis: communications between processors housed in the same node take longer times than communication between processors located in different nodes. Performance issues related to the Intel SMP architecture are responsible for such a phenomenon.

Finally, experimental measures collected during the execution of various benchmarks have substantiated that allocating all the processors available on each node results in performance losses.

Future work will be dedicated to the performance characterization of Linux clusters under more complex real live applications, as well as, to characterize the performance of different processor architecture, such as, AMD Athlon.

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**, pp. 846–853, 2001.

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

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

[4] A. Barak, I. Gilderman, & I. Metrik. Performance of the communication layers of TCP/IP with the Myrinet gigabit LAN. *Computer Communications*, **22(11)**, pp. 989–997, 1999.

[5] A. Barak & O. La'adan. The MOSIX multicomputer operating system for high performance cluster computing. *Future Generation Computer Systems*, **13(4–5)**, pp. 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] J. Brehm, P.H. Worley, & M. Madhukar. Performance modeling for SPMD message-passing programs. *Concurrency: Practice & Experience*, **10(5)**, pp. 333–357, 1998.

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

[10] R. Butler & W. Lusk. Monitors, Messages, and Clusters: The p4 Parallel Programming System. *Parallel Computing*, **20(4)**, pp. 547–564, 1994.

[11] M. Calzarossa, L. Massari, & D. Tessera. Workload Characterization - Issues and Methodologies. In G. Haring, C. Lindemann & M. Reiser, eds., *Performance Evaluation - Origins and Directions*, volume 1769 of *Lecture Notes in Computer Science*, pp. 459–484. Springer-Verlag, 2000.

[12] F. Cappello, O. Richard, & D. Etiemble. Understanding performance of SMP cluster running MPI programs. *Future Geenration Computer Systems*, **17(6)**, pp. 711–720, 2001.

[13] D. Feitelson. Workload Modeling for Performance Evaluation. In M. Calzarossa & S. Tucci, eds., *Performance Evaluation of Complex Systems: Techniques and Tools*, volume 2459 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[14] S. Donaldson, J. Hill, & D. Skillicorn. BSP clusters: High performance, reliable and very low cost. *Parallel Computing*, **26(2–3)**, pp. 199–242, 2000.

[15] S. Gottlieb. Cost-effective clustering. *Computer Physics Communications*, —bf 142, pp. 32–48, 2001.

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

[17] J. Hartigan. *Clustering Algorithms.* Wiley, 1975.

[18] R. Hockney & M. Berry. Public International Benchmarks for Parallel Computers: PARKBENCH Committee Report-1. *Scientific Computing*, **3(2)**, pp. 101–146, 1994.

[19] J. Hsieh, T. Leng, V. Mashayekhi, & 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.

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

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

[22] The Maui Scheduler Open Cluster Software.
http://mauischeduler.sourceforge.net, 2002.

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

[24] Guide to Myrinet-2000 Switches and Switch Networks.
http://www.myri.com, 2001.

[25] IBM Redbooks. Linux HPC Cluster Installation. http://ibm.com/redbooks, 2001.

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

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

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

[29] D. Womble, S. Dosanjh, B. Hendrickson, M. Heroux, S. Plimpton, J. Tomkins, and D. Greenberg. Massively parallel computing: A Sandia perspective. *Parallel Computing*, **25(13–14)**, pp. 1853–1876, 1999.

[30] M. Yarrow, A. Woo, R. Wijngaart, & W. Saphir. New NAS Parallel Benchmarks Resulst. In *Proceedings of Supercomputing'97*. ACM SIGARCH and IEEE, 1997.