An Efficient Implementation for Broadcasting Data in Parallel Applications over Ethernet Clusters

Fernando G. Tinetti, Andrés Barbieri
LIDI - CeTAD, Computer Science School - Engineering School, UNLP
fernando@ada.info.unlp.edu.ar, barbieri@lidi.info.unlp.edu.ar

Abstract

This paper introduces a natural implementation for broadcasting data on Ethernet based clusters used for parallel computing. Initially, it will be shown that libraries for Message Passing between processes such as PVM and implementations of MPI do not implement efficiently this operation or there is no reliability in terms of its performance. An implementation for broadcast messages is presented, taking into account the Ethernet based hardware layer found in most of the clusters used for parallel computing. The proposed implementation for broadcasting data is compared with the broadcast message available in PVM and LAM/MPI. Also, some comments are made about the proposed broadcast messages when the hardware layer is not Ethernet based. Finally, it will be shown by experimentation how the proposed broadcast message is used in the context of parallel linear algebra operations, specifically for parallel matrix multiplication.

1. Introduction

Although PVM [4] (Parallel Virtual Machine) and some free implementations of MPI [5] (Message Passing Interface) are the most widely used libraries for parallel processing in networks of workstations [5], they do not seem to be specifically designed and/or implemented to take advantage of Ethernet broadcast to optimize communication performance.

The idea underlying the present work is not to replace PVM or the chosen implementation of MPI, but to add an optimized broadcast message to these libraries for overall performance improvement. In some way, this implies a loss in portability at the source code level. However, for computer clusters used for parallel processing it also implies: a) a better communication performance for broadcasting , and b) parallelization flexibility (given that many times broadcast messages are avoided due to performance penalties).

In the context of cluster computing, where the message passing model is usually adopted as the programming model, the performance of (collective) communications has a direct impact on the performance of the whole parallel application as well as on the parallel algorithm to be selected. Moreover, communication performance implicitly defines the minimum granularity (computing/ communication ratio) without loss of overall parallel computing performance.

Many parallel algorithms are based on broadcasts, (e. g. matrix multiplication) developed to be used on specific interconnection networks of multicomputer architectures [2]. Most of these algorithms have been modified and/or adapted to be used on clusters. On the other hand, it is possible to develop optimized communication routines and parallel algorithms taking into account the hardware available in computers clusters.

The most important factors affecting (communication) performance will be analyzed later in this paper, such as data bandwidth and startup (latency) communication times for parallel processes messages. Startup time in local area networks are relatively much longer (sometimes measured in order/s of magnitude) than in the parallel computers interconnection networks, and thus, special considerations need to be made for parallel computing in clusters.

2. Broadcast Messages and Performance

It is rather usual to identify some classes of communication routines,such as point-to-point operations, and collective operations. Point-to-point operations involve two processes in a data transmission, where the primitives are *send* and *receive*. Collective operations involve communicating more than two processes. Many collective operations (or communications) have been proposed (and sometimes used) -broadcast and multicast messages are among the most accepted ones.

Collective communications can be built on top of the point-to-point routines, which is commonly found in parallel programming environments for clusters such as PVM and implementations of MPI (LAM/MPI [7], MPICH [MPICH]). Even when this approach simplifies libraries development and maintenance, it usually implies poor performance for collective communication routines. Each call to a collective routine implies -at lower level- many point-to-point executed routines.

The *natural* collective communication routines to be dealt with for optimization in the context of computer clusters are *broadcast* and *multicast* (which is very similar to *broadcast*). Ethernet networks are based on a logical bus which can be reached from any attached workstation. The method to access the bus is known as random access. Basically, in every data transference, the sender floods the bus, and only the destination/s will take the information from the channel. The destination/s might be a group of workstations and, in this case, the sender uses a multicast/broadcast address, sending data only once. All the referenced computers will take the data from the channel. at the same time.

The time to transmit *n* data units is usually determined by the equation

$$t(n) = stup + n\ dr \qquad (1)$$

Where *stup* is the communication startup time or latency and *dr* is the data rate directly related to communication channel data bandwidth, which are well known in Ethernet LANs at the hardware level. However, it is very difficult to characterize and quantify the overheads involved in communicating processes of a parallel application, such as: 1) Operating system: data movement through the memory hierarchy, NIC handling (e.g. interrupts), etc., 2) General parallel software library implementation: protocol usage (TCP/IP), protocol packets handling (Maximum Transfer Unit: MTU), etc., and 3) Routines specific implementation, which is directly related to the (Ethernet-logical bus) hardware.

Communication performance at the user's level process is usually measured by experimentation. In addition, more than two processes are involved in every collective communication, and this fact has to be carefully measured in the experimental work.

3. Experimentation

PVM (Parallel Virtual Machine) and LAM/MPI [7] (Local Area Multicomputer/Message Passing Interface) were chosen as the software libraries for communicating processes of a parallel application. The communication hardware was a switched 100 Mb/s Ethernet 100BaseTx The cluster included eight PCs (PIII 700 MHz, Linux 2.2.12-20, 64 MB).

Given the asynchronous definition of the broadcast operation in PVM and MPI and the highly distributed parallel architecture (without any kind of synchronization), time measurements were carried out by the following method:

1. Every process synchronizes with a barrier operation, just before beginning the broadcast.
2. Then, each takes time locally.
3. The broadcast operation is carried out. The sender process (which is unique) is usually called the root for the operation.
4. Once broadcast message is finished (when every process has received its data), each process takes time locally again, calculates the local elapsed time, and sends it back to the root process.
5. The root process receives all the elapsed times and evaluates the maximum value. This is considered as the total time of the collective operation.

Since there are many ways of making broadcasts in PVM and LAM/MPI, all of them are measured. The various configurations depend on the routing method (daemon-to-daemon or task-to-task) and the data encoding (XDR or none). Every experimentation was carried out without any other traffic on the local network in order to avoid any unexpected behavior (performance loss) due to a busy local area network.

Different message lengths were analyzed to identify any performance dependence on the number of transferred bytes. Also, every message length was measured with increasing number of receivers (from two to seven).

4. PVM-LAM/MPI Broadcast Performance

Fig. 1 shows the best measured times for PVM broadcast messages (*pvm_bcast*). Times measured using PVM multicast routine (*pvm_mcast)* are almost the same. The elapsed time is shown in a logarithmic scale, and dependent on message length (8, 100, 1000, 60000, 10^6 bytes respectively) and number of receivers (from 2 to 7 receivers). In general, for every message length, the time needed to complete the broadcast operation is proportional to the number of receivers.

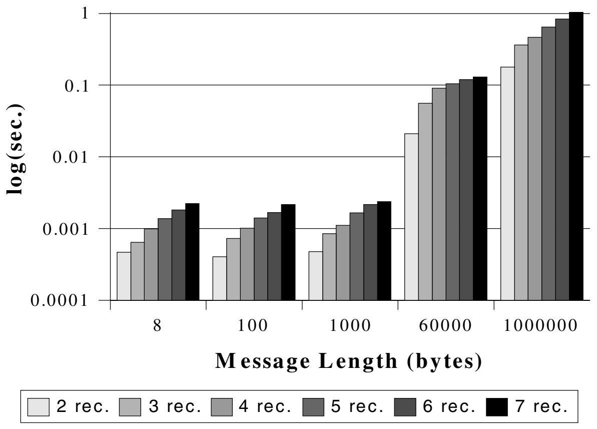

Figure 1: PVM Broadcast Performance (Time).

In addition, from Fig. 1 it is clear that for message lengths ranging from 0 to 1000 bytes, the whole elapsed time is dominated by the communications startup time. Fig. 2 shows the same experimentation but in terms of MB/s, where two characteristics of performance can be pointed out: 1) For message lengths between 8 and 1000 bytes, startup time implies poor performance (below 25% the optimum network data bandwidth - 100 Mb/s) independently of the number of receivers, and 2) Broadcast performance is strongly dependent on the number of receivers. Performance is below 1 MB/s (below 10% of the hardware data rate) for more than 3 receivers independently of the message length.

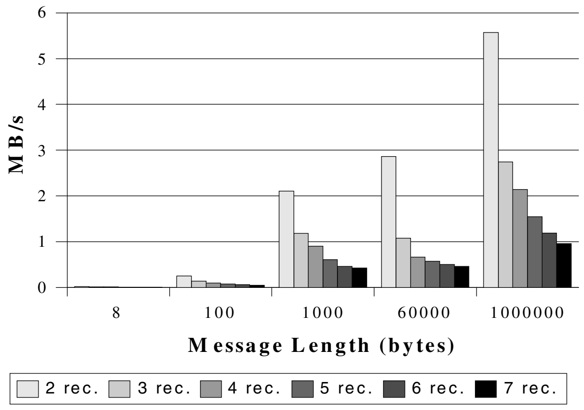

Figure 2: PVM Broadcast Performance (MB/s).

The LAM/MPI library has much better performance values: a) LAM/MPI startup time is about one order of magnitude below the PVM startup time, and b) LAM/MPI data rate is about twice PVM data rate.

Even when the LAM/MPI documentation explicitly states that broadcast implementation is made by constructing binomial spanning trees with nodes involved in the collective data communication, there are several points that should be taken into account: 1) Spanning trees are useful only when only one Ethernet switches is used for interconnecting the whole network, and 2) Spanning trees reduce time complexity at the most to logarithmic scale (which is considered good enough most of the times), but logarithmic time complexity is considerably worse than the constant time complexity which can be obtained, at least theoretically, by taking into account the logical Ethernet bus.

In general, MPI implementations do not necessarily have to optimize broadcast performance and/or optimize the availability of the Ethernet logical bus since the MPI standard itself does not impose any restriction and/or characteristic on any of the routines it defines.

5. UDP Broadcast Performance

A broadcast message was implemented directly on top of the UDP protocol, which makes use of the Ethernet broadcasting facility. Some consequences of this approach are: a) Portability (since the UDP protocol is available on every computer), b) User level communication routine (it is not necessary to modify the operating system kernel or to impose any special or root level priority), and c) If Ethernet is not found at the hardware level for communications, the UDP protocol tends to optimize the hardware capabilities. On ATM networks for example, it is likely that the UDP broadcasting facility will be better than every user-designed broadcast message. One of the main reasons for this assumption is that the UDP protocol is specifically oriented to obtain the best available performance.

Fig 3 shows the measured times for the proposed UDP implementation of broadcast messages, which can be compared with the values in Fig. 1 for the PVM broadcast message. Fig. 4 shows the same values in terms of MB/s, which can be compared with those in Fig. 2.

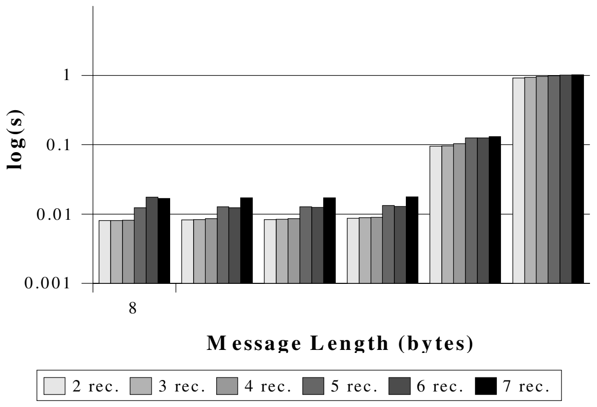

Figure 3: UDP-based Broadcast Performance (t).

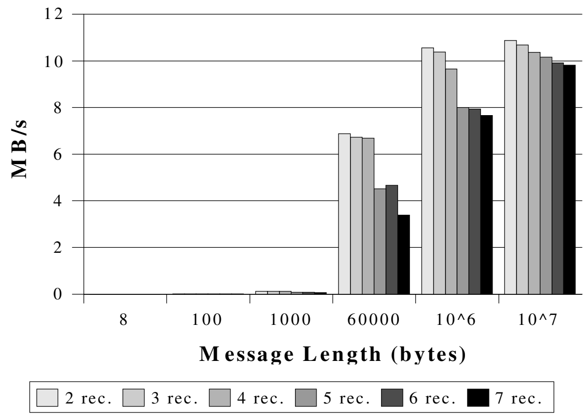

Figure 4: UDP-based Broadcast (MB/s).

The implemented broadcast routine requires significantly less memory space than the PVM broadcast, and some experiments were carried out for message lengths of 10^7 bytes (about 10 MB).

The proposed implementation has startup times significantly worse than those found with PVM. However, for message lengths equal to 10^6 bytes or greater, three very important performance characteristics are found: 1) Broadcast elapsed time is almost independent of the number of receivers, thus a "constant" communication time for a given message length is obtained, 2) Performance is near optimal for 10 MB/s Ethernet used. The theoretical optimum is 1.25 MB/s in 10Mb/s Ethernet networks, and 3) If latency time is reduced, then the two previous characteristics will be found in the messages performance for lengths of less than 10^6 bytes.

From the performance point of view, the UDP-based implementation for broadcasting data is far better than

PVM and LAM/MPI routines for message lengths equal to 60000 bytes or greater.

6. Broadcast-Based Matrix Multiplication

Linear algebra applications are extensively and intensively solved taking advantage of the parallel architecture provided by clusters. The SPMD (Single Program - Multiple Data) processing model is followed by most of the parallel algorithms in this area. Also, broadcast messages are very useful in many of the algorithms in order to reduce their complexity and/or communication requirements. Many parallel matrix multiplication algorithms have been proposed and many of them have been adapted to

Fig. 5 shows experimental performance values of a parallel matrix multiplication algorithm based on the broadcast message. Square matrices of order 3200 elements were multiplied. The speedup obtained with the PVM as well as the UDP-based broadcast are shown for every number of computers.

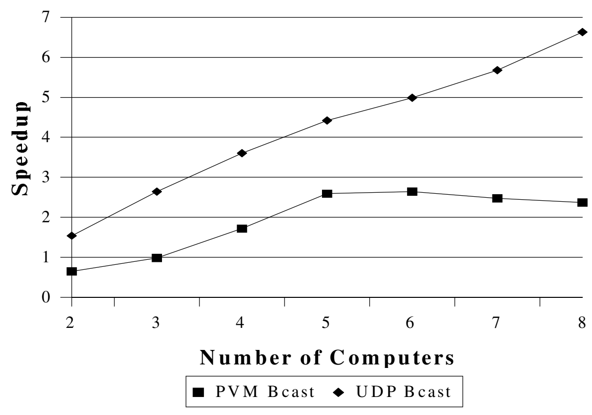

Figure 5: Speedup Values for Matrix Multiplication.

As Fig. 5 shows, even when the optimal values (corresponding to the straight line y = x) are not obtained, the algorithm using UDP-based broadcast renders some satisfactory characteristics for the speedup values: 1) Performance grows as the number of computers grows, 2) The difference between obtained speedup values and the optimal ones is mainly due to the low granularity of the problem. Better values can be obtained increasing the matrices order according to the number of computers

very simple problem (matrix multiplication) and with a low granularity, the obtained speedup values are near 20% of the optimal ones.

7. Conclusions and Further Work

Collective communications can be directly translated to the most commonly found network interconnection hardware on local area networks: Ethernet 10/100 Mb/s. Many algorithms can use the optimized performance of a specifically implemented broadcast over these networks. However, the software libraries for parallel computing on networks of workstations are not designed for optimal usage of any communication hardware.

It was shown by experimental work that PVM has very low broadcast communication performance on a real local area network. LAM/MPI takes advantage of switching networks building a binomial tree for distributing data, but the reached performance is very low compared with the physical network bandwidth. A preliminary

implemented over UDP/IP has proven to be successful in achieving near optimal broadcast communication performance for message lengths of about 10^6 bytes or larger. Broadcast time for message sizes ranging from 8 to 10^6 bytes is mainly dominated by the high latency, and the resulting performance can be considered as poor.

Some preliminary optimizations to the UDP-based broadcast message has shown a dramatic improvement in startup time of about 1 order of magnitude in some cases. This implies to improve overall communication performance as well as decreasing the parallel processing granularity in an order of magnitude too.

The optimized broadcast message should be used for other parallel linear algebra algorithms such as the matrix factorizations included in libraries like LAPACK [1] and ScaLAPACK [3].

8. References

[1] Anderson E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK: A Portable Linear Algebra Library for High-Performance Computers, Proceedings of Supercomputing '90, pages 1-10, IEEE Press, 1990.
[2] Choi J., "A New Parallel Matrix Multiplication Algorithm on Distributed-Memory Concurrent Computers", Proceedings of the High-Performance Computing on the Information Superhighway, IEEE, HPC-Asia '97.
[3] Choi J., J. Dongarra, R. Pozo, D. Walker, "ScaLAPACK: A Scalable Linear Algebra Library for Distributed Memory Concurrent Computers", Proc. 4th Symposium on the Frontiers of Massively Parallel Computation, Ieee Computer Society Press, pp. 120-127, 1992.
[4] Dongarra J., A. Geist, R. Manchek, V. Sunderam, Integrated PVM framework supports heterogeneous network computing, Computers in Physics, (7)2, pp. 166-175, April 1993.
[5] Message Passing Interface Forum, MPI: A Message Passing Interface standard, International Journal of Supercomputer Applications, Volume 8 (3/4), 1994.
[6] Wilkinson B., Allen M., Parallel Programming: Techniques and Applications Using Networking Workstations, Prentice-Hall, Inc., 1999.
[7] G. Burns, R. Daoud, and J. Vaigl, LAM: An Open Cluster Environment for MPI. Ohio Supercomputer Center, May 1994. LAM/MPI is available at University of Notre Dame (http://www.mpi.nd.edu/lam) - 1998-2001
[MPICH] MPICH Home Page http://wwwunix.mcs.anl.gov/ mpi/mpich/