Parallel Matrix Multiplication on Heterogeneous Networks of Workstations

Fernando Tinetti[1], Emilio Luque[2]

[1]Universidad Nacional de La Plata
Facultad de Informática, 50 y 115
1900 La Plata, Argentina
fernando@ada.info.unlp.edu.ar

[2]Universidad Autónoma de Barcelona
Facultad de Ciencias
08193 Barcelona, España
e.luque@cc.uab.es

Abstract

Matrix multiplication is taken as a test bed for parallel processing on heterogeneous networks of workstations (local area networks) used as parallel machines. Two algorithms are proposed taking into account the specific kind of parallel hardware provided by local area networks, and experimentation is used to drive the evaluation and identification of possible performance loss. A specific broadcast communication between processes of a parallel application is also proposed, taking advantage of the Ethernet interconnection network to achieve optimized performance. A special emphasis is place on already installed networks of workstations, which provide a *hardware zero cost* parallel computer; but a homogeneous Beowulf-class system is used to show how the algorithms are also useful on current *classical* high performance parallel computing with clusters.

Keywords: Cluster Computing, Heterogenous Parallel Computers, High Performance Computing, Performance Evaluation, Linear Algebra Applications.

1.- Introduction

Computation intensive applications take advantage of the growing processing power of standard workstations, along with their low cost and the relatively easy way in which they can be available for parallel processing. Usually, computation intensive areas have been referred to as scientific processing, such as linear algebra applications, where a great effort has been made in order to optimize solution methods for serial as well as parallel computing [1] [2]. Along with the definition of the LAPACK (Linear Algebra PACKage) library, linear algebra operations are characterized in terms of memory as well as computing requirements. BLAS (Basic Linear Algebra Subroutines) definition on three different levels (Level 1, 2, and 3) represents the main result of these efforts. It has been shown that routines within Level 3 BLAS are the most appropriate to be optimized in order to have optimized performance in sequential and parallel computers [4]. Also, it has been shown how the complete Level 3 BLAS can be implemented using matrix multiplication [6]. All of this leads to choosing the matrix multiplication as the main operation to be solved in sequential as well as parallel computers.

On the parallel hardware side, installed networks of workstations that can be used for parallel processing provide a “hardware zero cost parallel computer”. Hardware installation as well as maintenance cost is “zero”, because LANs are already installed and each computer has its own application programs, user/s, etc., independently of parallel computing. However, parallel

computing on these platforms is not “zero cost”. Even if the minimum installation of libraries for developing and running parallel programs -such as PVM (Parallel Virtual Machine) [3] or implementations of MPI (Message Passing Interface) [7]- are discarded, there are other costs involved, such as applications parallelization and computers availability.

The parallelization of applications represents one of the main costs involved in parallel computing, since there does not exist a generally defined guideline to be applied. Also, debugging parallel software is still one of the main drawbacks for parallel processing in this area. To obtain an acceptable performance, parallel algorithms are designed having into account the hardware. This leads to having certain algorithms for a task to be solved (efficiently enough) by certain parallel computers. In this context, networks of workstations, clusters, or LAN constitute a *new* kind of parallel computers taking into account that a) the computing hardware (processors) is usually heterogeneous on already installed LAN, and b) the interconnection network is usually Ethernet based. These two factors imply new problems in parallel applications in order to obtain acceptable performance, such as: a) computing workload, which should be assigned according to the computers relative speed, and b) special considerations should be taken for the communication pattern between processes, because Ethernet imposes unavoidable performance penalties (high startup time-low data bandwidth).

Section 2 describes two matrix multiplication algorithms along with general guidelines for the parallelization. The experimental evaluation is presented in Section 3, along with the considerations for an optimized broadcast message. Section 4 presents the immediate conclusions based on the experimental work as well as further work intended to improve/explore algorithmic and performance issues.

2.- Two Parallel Matrix Multiplication Algorithms

Two algorithms for parallel matrix multiplication are proposed taking into account distinguishing characteristics of heterogeneous NOW, one of them already published in [9]. Both are based in the same data distribution, which is made according to the relative processing power of each computer. This relative processing power can be obtained in several ways; two of them would be: 1) using a general benchmark suite, such as that proposed by the SPEC [5], 2) using an “*ad hoc* benchmark” according to the application area, or a short version of the application. The latter approach is chosen, even at the expense of generality, because: it is more specific and it thus renders more accurate values, and b) it is simpler at least in this application, because running a matrix multiplication is easier than installing and running a general benchmark suite on each machine. The relative processing power of each computer ws_0, ..., ws_{P-1}, is normalized obtaining pw_i such that $pw_0 + ... + pw_{P-1} = 1$.

2.1 Common Data Distribution

If square matrices of order n are involved in the matrix multiplication C = A×B, data distribution to computers is defined in terms of row blocks for matrices A and C and column blocks for matrix B:

- ws_i contains $rA_i = \lfloor n \times pw_i \rfloor$ rows of matrix A,
- ws_i contains $rC_i = \lfloor n \times pw_i \rfloor$ rows of matrix C ($rC_i = rA_i$), and
- ws_i contains $cB_i = \lfloor n/P \rfloor$ columns of matrix B.

where $\lfloor x \rfloor$ denotes the greatest integer number such that $\lfloor x \rfloor \leq x$.

Thus, the number of rows of matrix A (and C) assigned to each ws_i, (rA_i) is proportional to the computer relative processing power. This data distribution is not uniform when the machines are heterogeneous. According to the previous definition, it is possible that $dr = rA_0 + ... + rA_{P-1} < n$. The

remaining rows can be uniformly distributed among computers, ws_0, ..., $ws_{(n-dr-1)}$, one row for each computer. Given that the usual case is $P << n$, this reassignment of rows can be considered irrelevant from the point of view of proportional (according to the machines relative processing power) data distribution. The same kind of reassignment can be accomplished with $dc = cB_0 + ... + cB_{P-1}$ columns of matrix B. Fig. 1 shows this data distribution, where ws_i contains: a) $A^{(i)}$, the local block of matrix A, $rA_i \times n$ elements, b) $B^{(i)}$, the local block of matrix B, $n \times cB_i$ elements, and c) $C^{(i)}$, the local block of matrix C, $rA_i \times n$ elements ($rC_i = rA_i$). Also in Fig. 1 is shown that ws_i has local data only for a fraction of the local submatrix $C^{(i)}$, $C^{(ii)}$.

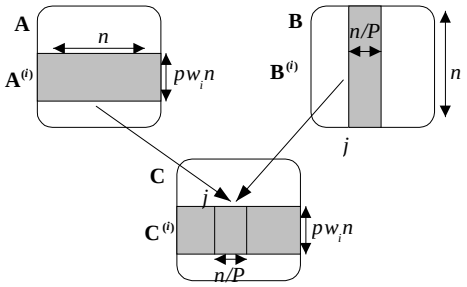
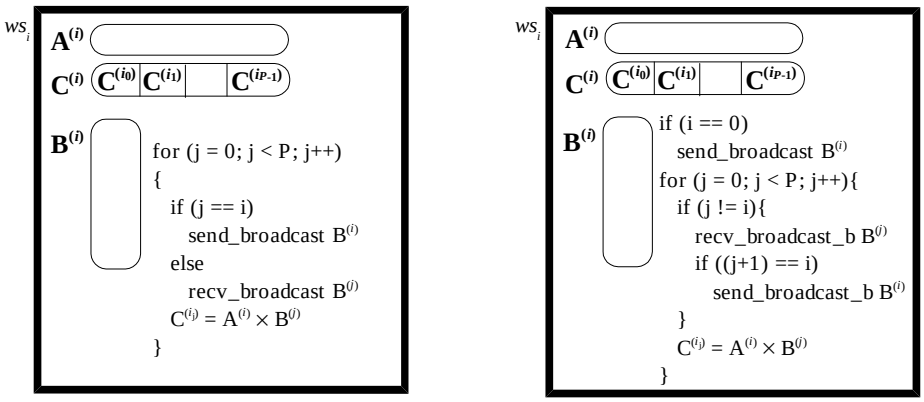


Figure 1. Local Data and Partial Computing in ws_i.

2.2 Algorithms

The complete $C^{(i)}$ will be calculated as a sequence of matrix multiplications $C^{(ij)}$ between $A^{(i)}$ and $B^{(j)}$ blocks with $j = 0, ..., P-1$. There are several ways of arranging computing and communication steps in order to have the complete submatrix $C^{(i)}$ calculated on each ws_i. Fig. 2 shows the pseudocode of the local processing in ws_i for both of the parallel algorithms. In Fig. 2.a, *SeqMsg*, every communication is carried out prior to the local computing step, imposing a sequence of communication-computing steps. The pseudocode in Fig. 2.b, *OvrMsg*, is arranged to make use of overlapped communication in the computers where it is available. The operations send_broadcast_o and recv_broadcast_o are used to send and receive broadcast data in "background", i.e. overlapped with other processing in the computer. Performing overlapped (in *background*) communication while computing depends on many factors such as the NIC (Network Interface Card) hardware. Heterogeneous machines in a LAN do not necessarily have this facility though it will be used where available.



a) Sequential Steps (SeqMsg) b) Overlapped Communication (OvrMsg)

Figure 2. Local Computing on ws_i for Two Parallel algorithms.

Both algorithms have well defined characteristics, which are taken as guidelines for the

parallelization of linear algebra operations: a) SPMD (Single Program-Multiple Data) model, b) broadcast-based, oriented to taking advantage of the most used hardware interconnection in LANs: Ethernet, c) coarse granularity, which is always a good approach and even better on high startup time-low data bandwidth networks.

SeqMsg is very simple and oriented towards synchronized and simultaneous computing on each machine. However, the communication steps imply a huge performance penalty on Ethernet-based LAN. It can be used for the evaluation of local computing as well as communication performance. OvrMsg is oriented to obtain the maximum performance of the complete parallel processing taking into account computing and data communication.

3.- Experimental Evaluation

Two installed local area networks were used to evaluate both algorithms by experimentation. The computers on one of the networks (LQT) are heterogeneous, and their details are shown in Table 1; they are interconnected by 10 Mb/s Ethernet. The second network (LIDI) is homogenous with eight identical computers, whose characteristics are summarized in Table 2; they are interconnected by 100 Mb/s switched Ethernet.

Table 1. Characteristics of the LQT LAN Computers.

Name	CPU	Clock	Mem	Mflop/s
lqt_07	Pentium III	1 GHz	512 Mb	625
lqt_06	Pentium III	1 GHz	512 Mb	625
lqt_02	Celeron	700 MHz	512 Mb	479
lqt_01	Pentium III	550 MHz	512 Mb	466
lqt_03	Pentium II	400 MHz	512 Mb	338
lqt_04	Pentium II	400 MHz	512 Mb	338

Table 2. Characteristics of the LIDI LAN Computers.

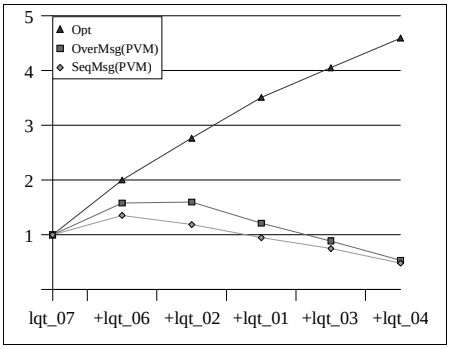
Name	CPU	Clock	Mem	Mflop/s
lidipar{14, 13, 12, 9, 8, 7, 6, 5}	Pentium III	700 MHz	64 Mb	579

On the LQT network, the matrix size used in the experiments is 5000×5000 elements, which is the greatest size which does not imply using swapping space during multiplication. On the LIDI network, the matrix size is 2000×2000 elements. Initially, the implementation was made using PVM. The same results were obtained using the broadcast (in a group) and the multicast message.

The results in terms of the speedup value on the LQT network are shown in Fig. 3.a. The reference value of the sequential algorithm was taken in the fastest computer (lqt_07), as expected in heterogeneous networks [10]. Computers are included (indicated by a “+” prefixed to the name) to the “parallel virtual machine” according to its relative processing power following the usual better-to-worse approach. The speedup values shown as “Opt” are the optimal expected for each set of machines. The obtained performance is clearly unacceptable for both algorithms.

Fig. 3.b shows the local execution times on each computer taken with OvrMsg when the six machines are used, corresponding to the last speedup value shown in Fig. 3.a. Most of the running time each computer is waiting for (*executing*) a message. The average computing time is approximately 110s while the average time used for communications is approximately 629s. The same kind of behavior in performance (local running times) is found for every combination of

computers and algorithms. Evidently, the performance loss is due to communications. Considering that: a) a whole matrix, B, is transmitted using broadcast messages, b) every matrix is square of order 5000, c) Ethernet broadcast facility is used, and d) assuming (rather optimistically on Ethernet 10 Mb/s) 1 MB/s data rate, the expected time for communications is about 100s. Then, the PVM broadcast is the origin of the performance penalty imposed to the application.



Name	Comp. time	Comm. time
lqt_07	89.21	657.71
lqt_06	89.15	657.57
lqt_02	109.13	637.87
lqt_01	174.52	572.56
lqt_03	92.79	654.50
lqt_04	102.52	592.84

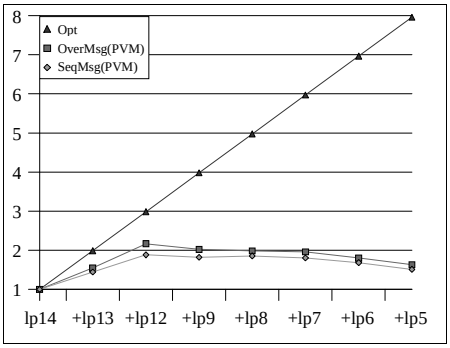
Comp. time: total local running time for $C^{(i)}$ (in seconds).
Comm. time: total local waiting time for communications (in seconds).

a) Speedup values using PVM b) OvrMsg(PVM) local times

Figure 3. Algorithms Performance with PVM on LQT (5000×5000).

The performance of the algorithms is not better on the LIDI network, which has a 10 times faster (Ethernet 100 Mb/s) interconnection network. Experimental results in terms of speedup value on the LIDI network are shown in Fig. 4.a, where the names of the machines are shortened to save space. Fig. 4.b shows the local execution times on each computer taken with OvrMsg when the eight machines are used, corresponding to the last speedup value shown in Fig. 4.a.

A facility is needed for broadcasting messages between application processes which take advantage of the Ethernet broadcast facility. The low likelihood of finding a library with a broadcast message implemented in this way is mainly due to: a) the general purpose nature of the libraries, such as PVM and MPI, and b) the (usually) large number of message operations provided by the libraries, which implies having a little number of optimizations to reduce implementation and maintenance costs. Also, if the previously mentioned broadcast-based guideline is used for parallelization, the design and implementation of an optimized version of the broadcast communication between processes of parallel programs are highly encouraged.



Name	Comp. time	Comm. time
lidipar14	3.78	13.49
lidipar13	3.75	13.52
lidipar12	3.73	13.54
lidipar9	3.74	13.54
lidipar8	3.73	13.55
lidipar7	3.74	13.54
lidipar6	3.72	13.56
lidipar5	3.73	12.08

a) Speedup values using PVM b) OvrMsg(PVM) local times

Figure 4. Algorithms Performance with PVM on LIDI.

A new and broadcast message based on the UDP protocol [8] was implemented to replace the PVM broadcast in the experiments. This new function is independent of PVM. The experimental results in terms of speedup values on the LQT and LIDI networks are shown in Fig. 5.a and Fig. 5.b

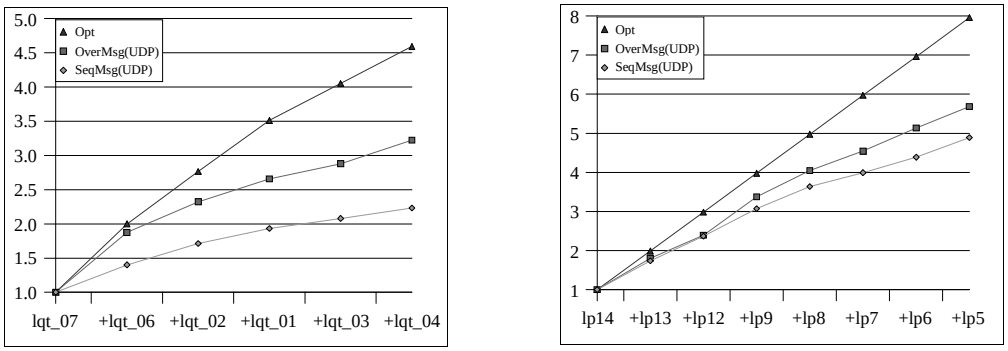
respectively. In every case, the same algorithms with the new UDP-based broadcast have better values than those obtained with PVM (Fig. 4). Further conclusions can be drawn from Fig. 5:

1. SeqMsg(UDP) and OvrMsg(UDP) performance is improved when more computers are included to solve the matrix multiplication, as at least expected for the granularity level given by the matrix size and the interconnection network performance (among other factors).
2. The difference between SeqMsg(UDP) and the optimum (Opt) speedup values is the *real* performance penalty imposed mainly by data communications. The optimum speedup values do not take into account any data communication.
3. Computers are able to overlap communication with computing in at least a fraction of the computing time. This is shown by the speedup values of OvrMsg(UDP), which are higher than those obtained by SeqMsg(UDP).

Differences between OvrMsg(UDP) and SeqMsg(UDP) in the LQT network, Fig. 5.a, are greater than in the LIDI network, Fig. 5.b, and this may be due to:

- Application granularity is greater on the LQT, because matrices of order 5000 elements are used (matrices of order 2000 are used in the LIDI network). This implies more time to communicate, but also more time to compute, and the computing time grows $O(n^3)$ compared to the $O(n^2)$ growth in communication.

  The LIDI network is 10 times faster than the LQT network which, among other facts, implies that communication time is less important in the total execution time, and this implies that every other overhead (operating system, library function, etc.) becomes more relevant and is not avoided by the use of an optimized broadcast function.



a) LQT Network (5000×5000) b) LIDI Network (2000×2000)

Figure 5. Algorithms Performance with UDP on LQT and LIDI.

4.- Conclusions and Further Work

Two algorithms (SeqMsg and OvrMsg), specifically designed for heterogeneous networks of workstations are presented and their performance is verified by experimental work. Both follow the guidelines for the parallelization of linear algebra operations: a) SPMD (Single Program-Multiple Data) model, b) broadcast-based, and c) coarse granularity. It is shown that the algorithms do not have acceptable performance due to the overheads imposed by the PVM library, more specifically, broadcast and multicast messages.

Having in mind the *general* guidelines to parallelize linear algebra operations, (in particular, broadcast-based parallel applications), a UDP-based broadcast message between parallel application processes is designed and implemented. This version of broadcast takes advantage of the Ethernet broadcast and the experimental work shows a substantial improvement in performance. Also, it is shown that most of the currently installed computers (or, at least, the PCs used) are capable of

overlapping communication with computing in at least a fraction of the processing time.

The proposed parallel matrix multiplication algorithms (SeqMsg and OvrMsg) are specially designed to avoid performance penalties in heterogeneous LAN used for parallel processing. While SeqMsg can be used to have an accurate measure of data communication weight in the total task to be solved, OvrMsg tends to take advantage of communication overlapped with local processing facility in the computers where it is available.

Matrix multiplication is representative but it is not the only linear algebra operation that should be parallelized. The initial extension of this wok is addressed to the other operations included in L3 BLAS. The next step is oriented towards the complete LAPACK library, in particular those having the highest requirements in computing power. The next step is even broader, including numerical computing not included in the linear algebra area.

Further work is necessary in the line of using more than one local area network or more than one cluster in order to solve one or more operations in parallel. It is clear that having two or more LAN implies many new and not measured characteristics, such as the different transmission times depending on the location (i.e. LAN) of the machines being communicated. Also, the UDP-based broadcast should be intensively and extensively evaluated in the presence of multiple inter-LAN (IP) routers, each with its own communication (possibly different) performance.

Acknowledgments

Prof. Alicia Jubert and Prof. Reinaldo Pis Diez, from the Laboratorio de Química Teórica, CEQUINOR, Departamento de Química, Facultad de Ciencias Exactas, Universidad Nacional de La Plata, provided the LQT LAN. Prof. Armando De Giusti, from the Laboratorio de Investigación y Desarrollo en Informática, Facultad de Informática, Universidad Nacional de La Plata provided the LIDI LAN as well as numerous suggestions. Prof. Antonio A. Quijano has also offered continuos support in many ways to this work.

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] Bilmes J., K. Asanovic, C. Chin, J. Demmel, Optimizing matrix multiply using phipac: a portable, high-performance, ansi c coding methodology, Proc. Int. Conf. on Supercomputing, Vienna, Austria, July 1997, ACM SIGARC.

[3] 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.

[4] Dongarra J., D. Walker, “Libraries for Linear Algebra”, in Sabot G. W. (Ed.), High Performance Computing: Problem Solving with Parallel and Vector Architectures, Addison-Wesley Publishing Company, Inc., pp. 93-134, 1995.

[5] Henning J. L., SPEC CPU2000: Measuring CPU Performance in the New Millenium, Computer, IEEE Computer Society, July 2000.

[6] Kågström B., P. Ling, C. Van Loan, “Portable High-Performance GEMM-based Level 3 BLAS”, R. F. Sincovec et al., Editor, Parallel Processing for Scientific Computing, Philadelphia, 1993, SIAM, pp. 339-346.

[7] Message Passing Interface Forum, MPI: A Message Passing Interface standard, International Journal of Supercomputer Applications, Volume 8 (3/4), 1994.

[8] Postel J., “User Datagram Protocol”, RFC 768, USC/Information Sciences Institute, Aug. 1980.

[9] Fernando Tinetti, Antonio Quijano, Armando De Giusti, Emilio Luque, “Heterogeneous Networks of Workstations and the Parallel Matrix Multiplication”, Euro PVM/MPI 2001, Santorini (Thera) Island, Greece, Apr 15-18, 2002.

[10] Zhang X., Y. Yan, “Modeling and characterizing parallel computing performance on heterogeneous NOW”, Proceedings of the Seventh IEEE Symposium on Parallel and Distributed Processing, *(SPDP'95)*, IEEE Computer Society Press, San Antonio, Texas, October 1995, pp. 25-34.