Performance of Scientific Processing in Networks of Workstations: Matrix Multiplication Example

Fernando G. Tinetti

Centro de Técnicas Analógico-Digitales (CeTAD)[1]
Laboratorio de Investigación y Desarrollo en Informática (LIDI)[2]
fernando@ada.info.unlp.edu.ar

Abstract

Parallel computing on networks of workstations are intensively used in some application areas such as linear algebra operations. Topics such as processing as well as communication hardware heterogeneity are considered solved by the use of parallel processing libraries, but experimentation about performance under these circumstances seems to be necessary. Also, installed networks of workstations are specially attractive due to its extremely low cost for parallel processing as well as its great availability given the number of installed local area networks. The performance of such networks of workstations is fully analyzed by means of a simple application: matrix multiplication. A parallel algorithm is proposed for matrix multiplication derived from two main sources: a) previous proposed algorithms for this task in traditional parallel computers, and b) the bus based interconnection network of workstations. This parallel algorithm is analyzed experimentally in terms of workstations workload and data communication, two main factors in overall parallel computing performance.

Keywords: performance, cluster parallel computing, scientific processing.

1. Introduction

The growing processing power of standard workstations, along with the relatively easy way in which they can be available for parallel processing, have both contributed to their increasing use in computation intensive application areas. Usually, computation intensive areas have been referred to as scientific processing; one of them being linear algebra, where a great effort has been made to optimize solution methods for serial as well as for parallel computing [1] [3].

Since the appearance of software libraries for parallel environments such as PVM (Parallel Virtual Machine) [7] and implementations of MPI (Message Passing Interface) [10], the distributed processing power of networks of workstations has been available for parallel processing as well. Also, a strong emphasis has been made on the heterogeneous computing facility provided by these libraries over networks of workstations. However, there is a lack of

1 Facultad de Ingeniería, Universidad Nacional de La Plata
2 Facultad de Informática, Universidad Nacional de La Plata

published results on the performance obtained on this kind of parallel (more specifically distributed) processing architectures.

From the whole area of linear algebra applications, the most challenging (in terms of performance) operations to be solved are the so called Level 3 BLAS (Basic Linear Algebra Subprograms). In Level 3 BLAS, all of the processing can be expressed (and solved) in terms of matrix-matrix operations. Even more specifically, the most studied operation has been matrix multiplication, which is in fact a benchmark in this application area.

2. Characterization of Heterogeneous Computing

There are a number of distinguishing factors that characterize the heterogeneous computing hardware of a network of workstations such as processor, clock cycle, memory hierarchy, main memory size, etc. All of these factors affect the relative processing power of each workstation.

It is expected that intercommunication times between workstations are almost the same, given that the usual interconnection topology in a network of workstations is a 10 Mb or 100 Mb Ethernet bus. The network dynamically varying workload along with the different kinds of communication subsystems of workstations hardware make communication times not as similar as expected. The communication pattern of a parallel (distributed) computing may be affected by this kind of communication heterogeneity. Communication times are harder to characterize when more than a local area network of workstations is used. In this case, the communication times between two workstations are dependent on the physical location (i. e., LAN) where each of them resides.

Workstations heterogeneity usually implies software heterogeneity, basically at the operating system and development tools levels. Software heterogeneity also produces different ways in which the overheads (e.g. system calls) affect processing performance. Thus, the relative processing power is affected not only by the underlying hardware, but also by the software running along with the computing processes of the application. Development tools heterogeneity usually introduces some problems in the software development phase, but the influence on computing performance is not significant for scientific applications.

3. Parallel Scientific Applications on Heterogeneous Hardware

Parallel applications in general, and parallel scientific applications in particular, face some specific problems when the underlying computing hardware is heterogeneous in order to obtain a near optimal performance. Processing workload and communication workload are two of the most important factors affecting performance.

Traditionally, parallel applications have had homogeneous hardware target machines (i.e., the same processors and a similar message communication time). Thus, algorithms have been designed assuming homogeneous hardware, and when used on heterogeneous hardware their performance is far from optimal. However, it should be pointed out that these traditional algorithms on heterogeneous hardware solve the same applications from the numerical point of view. This is one of the reasons for claiming the goodness of making heterogeneous hardware work as a parallel machine: it can solve the same problems as the traditional (and more expensive) parallel computers with minor adaptations of the algorithms.

Processing workload seems to be easily solved in the field of parallel scientific computing given that most of the programs fit the SPMD (Single Program Multiple Data) model. The key idea is based on sequential relative processing power amongst workstations. If a workstation ws_i is twice as fast as ws_j, then it should receive twice the workload of ws_j, which most of the times implies twice the data to process. Even if this seems to be reasonable, it has to be experimentally justified for the optimized numerical algorithms where, for example, memory hierarchy is strongly used to achieve near peak processor performance. When computing processes have to share the workstation with communication processes, the sequential relative processing power could change, and it could be necessary to define the parallel relative processing power.

Communication workload is particularly changed in networks of workstations. First, because Ethernet bus is the most common LAN architecture to which workstations are connected. Traditional scientific parallel algorithms are based mostly on static meshes or dynamic networks where physical communication is solved point to point without any interference from other communicating processors. Second, the heterogeneity given by more than one interconnected LAN (e.g. by means of routers for Internet traffic) has not been studied from the scientific processing performance point of view.

4. Areas for Experimentation

In order to establish an incremental research project, it is necessary to define a number of experimentation areas to make the effect of heterogeneous networks of workstations on the performance scientific applications clear:

- Selection of a specific application. Matrix multiply has been selected for its many representative characteristics of the scientific processing area [14].
- Analysis of the performance of traditional parallel algorithms without taking heterogeneity into account, or only taking into account balanced processing workload based on sequential relative processing power [12].
- Optimizations of scientific code for maximum sequential performance [2] [14].
- Relationship between sequential and parallel relative processing power.
- Impact of bus interconnection network topology on the parallelization of scientific algorithms.
- Impact of the interconnection network topology (including more than one LAN) on scientific processing performance.

5. Parallel Matrix Multiplication Algorithms

Many parallel algorithms have been designed, implemented, and tested on different parallel computers for matrix multiplication [15]. For simplicity, the algorithms are usually described in terms of $C = A \times B$, where the three matrices A, B, and C are dense and square of order *n*.

The so called "direct implementation", in which every processor computes a portion of the resulting matrix C having (or receiving) the necessary portions of matrices A and B is used mainly for introductory and teaching purposes, and it is hardly ever used in practice.

Divide-and-conquer and recursive algorithms are considered specially suited for multiprocessor parallel computers. In fact, matrix multiplication is inherently good for shared memory multiprocessors because there is not data dependence. Matrices A and B are accessed only for reading to calculate every element of matrix C, and no element of matrix C has any relation (from the processing point of view) with any other element of the same matrix C. Unfortunately, networks of workstations used for parallel computing are not shared memory architectures, and implementations of divide-and-conquer and recursive algorithms are far from optimal in networks of workstations. The main reason for the loss of performance is found in the need of a shared (uniform) memory view of a distributed and loosely coupled memory architecture.

One of the most innovative algorithms for sequential matrix multiplication is due to Strassen [11] and its parallelization is straightforward on shared memory parallel computers. Again, the architecture of a network of workstations is not well suited (form the performance point of view) for this algorithm and its “immediate” ways of parallelization. Also, the Strassen method is defined in terms of different (for matrix multiplication) arithmetic operations such as subtraction, and then special care has to be taken for computer numeric rounding errors.

It is possible that most of the reported parallel algorithms used in practice are based on parallel multicomputers where the processors are arranged (and interconnected) in a two dimensional mesh o torus [9]. They may be roughly classified as “broadcast and shift algorithms” initially presented in [8] and “align and shift algorithms”, initially presented in [4]. Both kinds of algorithms are described in terms of a PxP square processor grid where each processor holds a large consecutive block of data.

Many algorithms have been proposed as rearrangements and/or modifications from those two initial ones ([4] [8]). The underlying concept of blocking factor became intensively and successfully used from two different but related standpoints: load balance and processor local performance. It can be proved that distributing relatively small blocks of matrices the job to be done on each processor is almost the same. Furthermore, the blocking factor is essential in achieving the best performance on each processor given the current memory hierarchies, where cache memories have to be taken into account with special care.

It is very interesting how the broadcast based algorithms have been successively and successfully adapted to the point-to-point interconnection of two dimensional torus. A very long list of publications is available about this subject (e.g. [5] [13] [6]), and all of them aim to obtain the best performance for the whole algorithm by implementing broadcast over the point-to-point communication links of distributed memory parallel computers.

Given the increasing utilization of networks of workstations for parallel computing it seems to be necessary to analyze if the already defined parallel algorithms fit (from the performance point of view) these “new” parallel architectures. If not, some rearrangement should be proposed just as it has been made for the two dimensional mesh based parallel computers.

The initial step should be to analyze the network of workstations (NOW) architecture and identify if the parallel algorithms are able to obtain the best performance. Interconnection (LAN) architecture and processing power heterogeneity are tightly related to the performance to be obtained in a NOW used for parallel processing. In fact, a relatively deeper analysis shows that LAN architectures as well as processing heterogeneity strongly contribute to performance degradation.

Shared memory based parallel algorithms should be discarded because they are not suitable for implementation on a loosely coupled parallel architecture such as a NOW. Mesh based class of parallel algorithms are the following to be analyzed. Initially, the installed networks of workstations do not seem to be related with meshes, because of their interconnection network usually based on a single bus (10/100 Mb Ethernet). This leads to eliminate at least shift based parallel algorithms. The basic idea of broadcast message should be used because it has a direct relationship with the bus interconnection. Unfortunately, as it has been explained, the initial "broadcast and shift algorithms" have evolved to "shift and shift algorithms". The point-to-point communications should be minimal in parallel programs executed on networks of workstations because they imply a bottleneck on a single and shared communication medium as the LAN bus.

The parallel algorithm proposed for installed networks of workstations has two main characteristics:
1. Based only on broadcasting data.
2. Easy workload distribution for heterogeneous processing power.

6. Parallel Matrix Multiplication on NOW

The description of the proposed parallel algorithm to compute C = A×B will be made taking into account:
- A, B, and C are n×n matrices,
- P workstations, ws_1, ..., ws_P,
- pw_i is the normalized relative processing power of workstation ws_i, $\forall\ i = 1...P$, where normalized implies $pw_1 + ... + pw_P = 1$,
- ws_i contains $rA_i = n \times pw_i$ rows of matrix A, and
- ws_i contains $cB_i = n/P$ columns of matrix B.

It is relatively straightforward to compute the normalized relative processing power having the performance in Mflop/s of each workstation ws_i, mfs_i:

$$pw_i = \frac{mfs_i}{\sum_{i=1}^{P} mfs_i} \tag{1}$$

The number of rows of matrix A assigned to each workstation (rA_i) is proportional to the workstation relative processing power, e.g.: if $n = 3000$, $P = 2$, and $pw_1 = 2/3$, and $pw_2 = 1/3$, then 2000 rows of matrix A will be assigned to ws_1 and 1000 rows of matrix A will be assigned to ws_2, reflecting that ws_1 is faster than ws_2. This data distribution is not uniform when the workstations have different processing power. Due to rounding errors for the operation $rA_i = n \times pw_i$ $(0 < pw_i < 1\ \forall\ i = 1, ..., P)$ it is possible that $dr = rA_1 + ... + rA_P < n$.

The remaining rows can be uniformly distributed among workstations $ws_1, ..., ws_{(n-dr)}$ one row for each workstation. Given that the usual case is $P << n$, this reassignment of rows can be considered non relevant from the point of view of proportional (according to workstations relative processing power) data distribution.

Matrix B is equally distributed among workstations, as the usual case in (homogeneous parallel computers) bibliography. The matrix B data distribution is made by columns and uniformly (each workstation has the same amount of data) because of the proposed algorithm and its way of computing matrix C.

Workstation ws_i computes a portion of matrix C, $C^{(i)}$, proportional to its relative processing power, where

$$C^{(i)}_{rA_i \times n} = A_{rA_i \times n} \times B \tag{2}$$

where only a submatrix o B is held locally (cB_i columns). Let $A^{(i)}$ ($A_{rA_i \times n}$) and $B^{(i)}$ ($B_{n \times cB_i}$) the local portions of matrices A and B assigned to ws_i respectively, the algorithm in pseudocode for this workstation (ws_i) is shown in Fig. 1. As it could be expected, it has two main characteristics:

1. Follows the SPMD (Single Program - Multiple Data) parallel computing model. It is the most common model of parallel programs for numerical - linear algebra computing.
2. Follows the message passing parallel programming model. Networks of workstations are between the most loosely coupled parallel machines, and message passing programming is the best suited for this kind of distributed memory parallel computers.

```
$C^{(i)} = 0$;                      /* Matrix initialization */
$B'^{(i)} = B^{(i)}$;                     /* To save local $B^{(i)}$ */
For j = 1 to $P$
{
   $C^{(i)} = C^{(i)} + A^{(i)} \times B'^{(i)}$;     /* Compute partial matrix */
   If (j == i)
      Broadcast $B^{(i)}$;           /* Broadcast local data */
   Else
      Receive $B'^{(i)}$;            /* Receive data broadcast from other ws */
}
```

Figure 1: Parallel Matrix Multiplication Algorithm.

Some of the possible optimizations to be made could be:

- If there is a facility to make communications overlapped with processing, then the broadcast could be called just before $C^{(i)}$ partial computing.
- If there is not enough room to have two sections of matrix $B^{(i)}$, which is a matrix of $n \times cB_i$ elements, computing as well as communication could be made in sections of $B^{(i)}$. This implies another iteration inside the one shown in Fig. 1.

Resuming the characteristics of the proposed algorithm:

- Every data transference is a message broadcast from a given workstation to every other in the parallel computer.

- Local computing is independent on each workstation. Local optimization, as the selection of a blocking factor for matrix multiplication, can be made independently on each workstation.
- The balanced workload is carried out by means of data distribution for matrices A and C according to the workstations relative processing power. Also, the uniform distribution of matrix B tends to equalize the memory requirements on each workstation.

7. Experimentation

The workstations used for experimentation are described in Table 1.

Name	**CPU / Mem**	**Mflop/s**
purmamarca	Pentium II 400 MHz / 64 MB	316
cetadfomec1	Celeron 300 MHz / 32 MB	243
cetadfomec2	Celeron 300 MHz / 32 MB	243
sofia	PPC604e 200 MHz / 64 MB	225
Josrap	AMD K6-2 450 MHz / 62 MB	99

Table 1: Characteristics of the Workstations used in the Experiments.

Also, workstations are interconnected by an Ethernet 10 Mb/s LAN and the PVM library [7] was used as the message passing as well as parallel computing software tool.

The (balanced) workload was verified by means of a synthetic "embarrassingly parallel" version of the matrix multiplication. In this version, the broadcast is eliminated and local assignment of $B^{(i)}$ is made instead of receiving a message. The library PVM was still used to "spawn" the processes remotely as well as to synchronize at the end to record time completion of the whole parallel application. The results obtained with this synthetic version where discriminated by iteration (as if communication were made), and an example for matrix size of n = 2000 is shown in Table 2, (all times are in seconds).

Name	***Assigned Rows***	***Total Time***	***Per It.***
purmamarca	562	14.36	2.87
cetadfomec1	431	14.31	2.86
cetadfomec2	431	14.35	2.87
sofia	400	14.48	2.9
Josrap	176	14.39	2.88

Table 2: Workload Verification for the Synthetic Parallel Algorithm.

The difference between the maximum and minimum local computing times (14.48 and 14.31 respectively) is about 1% which could be highly acceptable given the relative processing power differences: from 316 Mflop/s to 99 Mflop/s according to Table 1. The proposed algorithm (Figure 1) was later used, and the obtained results are shown in Table 3.

Name	***Assigned Rows***	***Total Time***	***Per It.***
purmamarca	562	16.22	3.24
cetadfomec1	431	16.37	3.27
cetadfomec2	431	16.18	3.24
sofia	400	16.42	3.28
Josrap	176	15.6	3.12

Table 3: Algorithm Performance.

The main conclusions taken from comparison of Table 2 and Table 3, are

- The overall computing performance is reduced by approximately 13% when there is data communication. Communication processes have a minimum memory (cache memory, in particular) requirement by which there is contention with computing processes, and it is reflected on the computing performance reduction.
- The processing workload is also affected by communication processes. The difference between the maximum and minimum local computing times (16.42 and 15.6 respectively) is about 5%.

The analysis on workload becomes non relevant when the communication times are taken into consideration (Table 2 and Table 3 just show computing times). The amount of time taken by the communication routines is between 76 and 108 seconds for this problem size (n = 2000). Then, about 85% of the total time is spent on communication, which degrades the overall performance at the point of making parallelism useless. The first question arising under these circumstances is about communication performance: are these communication times (between 76 and 108 seconds) the expected ones? Taking into account that the interconnection hardware is able of 10 Mb/s, a whole matrix of 2000x2000 single precision floating point numbers should take about 13 seconds which is near an order of magnitude of the current times taken for the application.

8. Conclusions and Further Work

It seems to be clear that some overhead is incurred in using a library for parallel computing on a network of workstations, but an overhead of an order of magnitude is clearly non acceptable. It becomes necessary, then, to analyze the (excessively) low communication performance achieved by the PVM library for this application. One of the first alternatives to be investigated is the way in which the broadcast message routine is implemented.

The proposed algorithm seems to be appropriate for parallel computing on network of workstations, but it has to be verified by experimentation to avoid the kind of biases found and explained in the previous section about communication performance. Also, it seems to be reasonable to make exhaustive experimentation to analyze the effectiveness of heterogeneous parallel computing on network of workstations with commonplace communication hardware.

Other parallel computing libraries such as MPI implementations should be examined by experimentation. Some libraries could implement more successfully the collective (e.g. broadcast) communications routines.

9. 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. Asanoviƒ, C. Chin, J. Demmel, Optimizing matrix multiply using phipac: a portable, high-performance, ansi c coding methodology, Proceedings of the International Conference on Supercomputing, Vienna, Austria, July 1997, ACM SIGARC.

[3] Blackford L., J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. Whaley, ScaLAPACK Users' Guide, SIAM, Philadelphia, 1997.

[4] Cannon L. E., A Cellular Computer to Implement the Kalman Filter Algorithm, Ph.D. Thesis, Montana State University, Bozman, Montana, 1969.

[5] Choi J., J. J. Dongarra, D. W. Walker, PUMMA: Parallel Universal Matrix Multiplication Algorithms on Distributed Memory Concurrent Computers, in Concurrency: Practice and Experience, 6:543-570, 1994.

[6] 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.

[7] 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.

[8] Fox G., M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Walker, Solving Problems on Concurrent Processors, Vol. I, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[9] Golub G. H., C. F. Van Loan, Matrix Computation, Second Edition, The John Hopkins University Press, Baltimore, Maryland, 1989.

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

[11] Strassen V., Gaussian Elimination Is Not Optimal, Numerische Mathematik, Vol. 13, 1969.

[12] Tinetti F., A. Quijano, A. De Giusti, Heterogeneous Networks of Workstations and SPMD Scientific Computing, 1999 International Conference on Parallel Processing, The University of Aizu, Aizu-Wakamatsu, Fukushima, Japan, September 21 - 24, 1999.

[13] van de Geijn R., J. Watts, SUMMA Scalable Universal Matrix Multiplication Algorithm, LAPACK Working Note 99, Technical Report CS-95-286, University of Tenesse, 1995.

[14] Whaley R., J. Dongarra, Automatically Tuned Linear Algebra Software, Proceedings of the SC98 Conference, Orlando, FL, IEEE Publications, November, 1998.

[15] Wilkinson B., Allen M., Parallel Programming: Techniques and Applications Using Networking Workstations, Prentice-Hall, Inc., 1999.