Chapter 4: Experimentation

This Chapter will present the entire experimentation context with parallel computing algorithms for matrix multiplication computation. It presents a detailed explanation in terms of hardware (three local networks) and matrix sizes with which the experiments were carried out. Initially, computer relative speed values are obtained and, with these values plus an (optimistic) a priori estimation of the interconnection networks performance, the maximum possible speedup value is estimated in each of the local networks with the available machines.

The first approximation to the algorithm implementation is based on the PVM library, and all the results of the experimentation are shown. The obtained performance is far from the optimal and it is not accepted. Thereby, some details of the parallel execution are shown and reveal that the problem is not the algorithms but the broadcast message implementation of the PVM library.

Finally, a broadcast message directly based on UDP communications protocol is proposed and implemented; the experiments are carried out once again, and the results obtained with this implementation of broadcast messages are shown. Within this context, we conclude that: either algorithms obtain an acceptable performance or computers generating performance problems can be detected automatically and, thus, they can be isolated in order to achieve a optimal performance.

4.1 Characteristics of the Local Area Networks

Each of the local area networks used in the experimentation are previous to the present paper and were not changed nor adapted so better results in terms of performance could be obtained. The following subsections describe briefly each of the networks, identifying the most important characteristics of the computers making up such network and the topology of the interconnection network. Basically, the local networks are:

- CeTAD: belongs to the Center of Digital-Analog Techniques, Electrotechnics Department, Faculty of Engineering, National University of La Plata. It was installed long time ago and computers making it up are used with several purposes.
- LQT: belongs to the Laboratory of Theoretical Chemistry, CEQUINOR, Chemistry Department, Faculty of Exact Sciences, National University of La Plata. It is a network that aims at the resolution of numerical problems. It was installed several years ago; it runs sequential and parallel works developed with PVM and Linda.
- LIDI: belongs to the Laboratory of Research and Development in Computer Science, Faculty of Computer Science, National University of La Plata. It is dedicated to the teaching of parallel programming and research. It can be considered as a Beowulf-type installation, though it does not belong to the most costly ones as regards the quantity of machines and interconnection network.

In more general terms, the parallel software was developed using PVM (Parallel Virtual Machine). In the particular case of the PCs used, Linux operating system was chosen for the execution [44] [PVM].

The reasons for having chosen PVM (apart from its free availability) are basically the following two:

- There exists a single software development and source group. This might be a disadvantage, but it simplifies the analysis of the results obtained in terms of performance since there is no chance of different implementations. This cannot be asserted in the case of MPI (Message Passing Interface) [88] [92] [107], which has several implementations and potentially different characteristics as regards performance.
- It is widely used, it has several years of evolution, and its characteristics are very well known, all of which simplifies the interpretation of the results obtained in terms of the parallel applications' performance that make use of it.

In most of the local networks, the PCs already had Linux installed, mainly with RedHat [LinuxRH] distribution. Whenever possible, Linux installation was attempted in a separated disk partition, and the distribution used in such case is that of RedHat.

4.1.1 CeTAD Local Area Network

As previously mentioned, the CeTAD local network has two general characteristics implying a great heterogeneity of machines:

1. It was installed more than ten years ago, and has undergone changes, additions and updates.
2. Machines have several purposes, which encompass administrative issues, signal

processing algorithm prototypization, and integrate circuit design of specific purpose.

Table 4.1 shows the most important characteristics of the computers that compose the local network and that are used in the experimentation.

	Name	Type	CPU	Clock	Mem.
1)	purmamarca	PC	Pentium II	400 MHz	64 MB
2)	cetadfomec1	IBM PC	Celeron	300 MHz	32 MB
3)	cetadfomec2	IBM PC	Celeron	300 MHz	32 MB
4)	sofia	IBM RS6000	IBM PPC604e	200 MHz	64 MB
5)	fourier	PC	Pentium MMX	200 MHz	32 MB
6)	Josrap	PC	AMD K6-2	450 MHz	62 MB
7)	tilcara	PC	Pentium	133 MHz	32 MB
8)	paris	SPARCstation 4	MicroSPARC-II	110 MHz	96 MB
9)	cetad	SPARCstation 5	MicroSPARC-II	85 MHz	96 MB
10)	prited	SPARCstation 2	CY7C601	40 MHz	32 MB

Table 4.1: CeTAD Computers.

Appendix A includes more details of each machine. Except for **cetadfomec1** and **cetadfomec2** identified with the "type" IBM PC, all PCs are built by parts - which is often the case for PCs.

Once more, it must be said that nothing already installed was changed, though software tools necessary for the development, implementation, and execution of parallel programs were added in those computers that did not have them before the experimentation. The highest installation cost with this respect appeared with PCs that do not have Linux nor the software tools necessary for parallel computation (libraries such as PVM [44] [PVM]):

- **Purmamarca**, hard disk partitioned, and Linux (RedHat) and PVM installed.
- **Fourier** and **Josrap**, where Winlinux [WinLinux] distribution was installed since it was really difficult (or risky) to partition the hard disk, and Winlinux distribution was considered the simplest to install in such conditions.

The machine interconnection network is Ethernet of 10 Mb/s, and the wiring is shown in Figure 4.1, where

1. Cascade hubs are used, and the computer logical interconnection is still that of a bus.
2. "Trans." implies Transceiver, which is necessary because the computer netboard -**prited** - has only BNC connection output (for coaxial cable).
3. "**cf1**" and "**cf2**" are used as abbreviations of **cetadfomec1** and **cetadfomec2**, respectively (and such abbreviations will be used for space's sake).

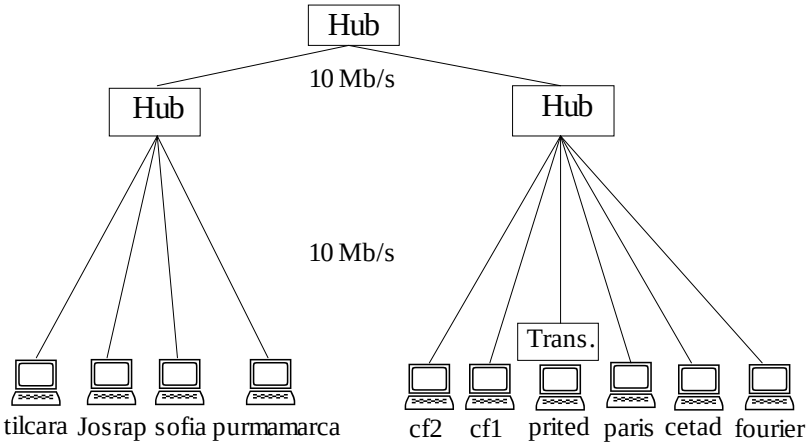


Figure 4.1: CeTAD Local Area Network Wiring.

4.1.2 LQT Local Area Network

Unlike the CeTAD local area network, that of LQT is meant to solve numerical problems. In fact, they only have just the necessary installed for such end, without the *classical* office tools, such as text or spreadsheets editors/formatters. However, like the CeTAD local network, it was installed, and in use from, many years ago and it has been consequently updated (and *enhanced*) several times.

Table 4.2 shows the most important characteristics of computers composing the local network and that are used in the experimentation. Appendix A includes more details of each machine.

	Name	**Type**	**CPU**	**Clock**	**Mem.**
1)	lqt_07	PC	Pentium III	1 GHz	512 MB
2)	lqt_06	PC	Pentium III	1 GHz	512 MB
3)	lqt_02	PC	Celeron	700 MHz	512 MB
4)	lqt_01	PC	Pentium III	550 MHz	512 MB
5)	lqt_03	PC	Pentium II	400 MHz	512 MB
6)	lqt_04	PC	Pentium II	400 MHz	512 MB

Table 4.2: LQT Computers.

Unlike the CeTAD local area network, all the computers available in this network are PCs built in parts with a rather higher capacity in terms of computation (processors and operation clocks frequency) and storage (installed main memory).

The machines interconnection network is Ethernet of 10 Mb/s, and the wiring is shown in Figure 4.2, where it can be noticed that the interconnection is one of the simplest possible ones.

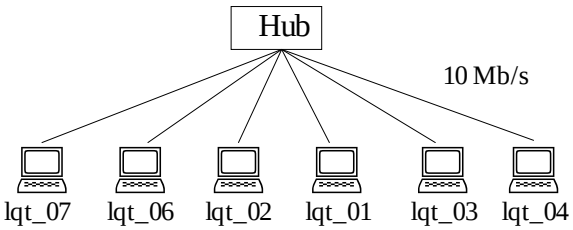


Figure 4.2: LQT local Area Network Wiring.

4.1.3 LIDI Local Area Network

Unlike the two previous local networks:
- LIDI local network was planned and built up exclusively for parallel computation and, thereby, it also coincides with the Beowulf installation.
- It is no more than one year old and, thus, it has suffered no changes since its installation, being still homogeneous.

Table 4.3 shows the most important characteristics of the computers that make up the local network and that are used in the experimentation. Appendix A provides more details of each machine. Computers appear in a table just to use the same format as for the previous networks; however, since the computers are equal, it is enough to describe just one of them.

	Name	Type	CPU	Clock	Mem.
1)	lidipar14	PC	Pentium III	700 MHz	64 MB
2)	lidipar13	PC	Pentium III	700 MHz	64 MB
3)	lidipar12	PC	Pentium III	700 MHz	64 MB
4)	lidipar9	PC	Pentium III	700 MHz	64 MB
5)	lidipar8	PC	Pentium III	700 MHz	64 MB
6)	lidipar7	PC	Pentium III	700 MHz	64 MB
7)	lidipar6	PC	Pentium III	700 MHz	64 MB
8)	lidipar5	PC	Pentium III	700 MHz	64 MB

Table 4.3: LIDI Computers.

As Figure 4.3 shows, LIDI network wiring coincides in terms of simplicity with that of the LQT network, but not in terms of cost since:
- Installed net boards are Ethernet of 10/100 Mb/s, which implies that the communication speed depends on the hub or switch interconnecting them.
- Instead of a hub, a switch is used, which, like computer netboards, is also of 10/100 Mb/s.
- All the net has thus the capacity of carrying out several (up to four) simultaneous point-to-point communications of 100 Mb/s.

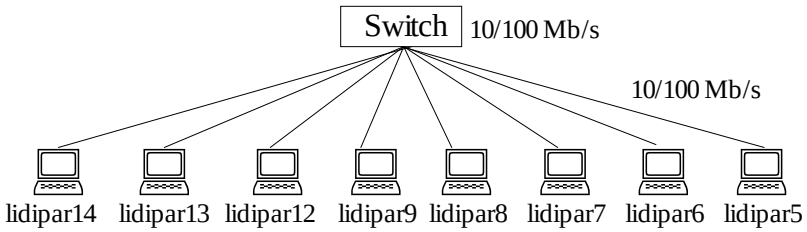


Figure 4.3: LIDI Local Network Wiring.

4.2 Computers' Sequential Performance

The computation of computers' sequential performance is based on two main reasons:
1. Speedup computation obtained when using parallel processing. In order to acknowledge the benefits of using parallel processing it is necessary to know at least the performance of the fastest available computer in the local network.
2. As advanced in the previous chapter, computers' relative speed is computed according to the sequential computing capability of each of them in order to solve a matrix multiplication.

For these two reasons, performance values obtained for each of the machines in each of the local networks are included in this section. Additionally, Appendix B explains in detail how these values were obtained according to the different experimentations carried out.

Given the almost numerical nature of data processing, performance values are expressed in Mflop/s (millions of floating point operations per second). The representation of the numerical data is that of simple precision floating point (IEEE 754 norm [72]) in all the computers. The option of double precision floating point –though generally it is advised [11] – is discarded mainly because:
- Computers used have the same or similar capacity of processing simple precision floating point numbers as with double precision, due to the characteristics of their floating point units.
- When using numbers represented in simple precision, larger sizes of matrices in memory can be obtained and, thus, greater requirements can also be obtained in terms of number of floating point operations necessary for solving a matrix multiplication.

4.2.1 Matrix Sizes

Without doubts, computers' performance in terms of data processing, in general, and floating point number processing, in particular, depends on the relationships established between:
- Amount of Data.
- Memory hierarchy (levels and sizes of cache memory).
- Data access pattern.

Thereby, performance values are shown in function of the significant sizes of the matrices.

These relations and the most significant details of the chosen sizes in particular are explained in more detail in Appendix B.

On the one hand, it is important to have a reference of the computer processing capability when all or most of the data to be processed can be included in the cache memory levels closer to the processor (levels 1 and 2, for instance). In this context, some relatively small matrix sizes –in comparison to the main memory- were taken as reference: matrices of order n = 100, 200, 400. Considering that numerical data are represented with simple precision floating point numbers, for n = 100, the quantity of data necessary to store a matrix will be of $100^2 \times 4$ bytes, less than 40 KB of data.

The use of the computers to the limit of their capacity (at least in terms of main memory) has been a constant and, in some way, reflects the "review" of Amdahl's Law [6] [60]. Two values were taken as representatives of matrix sizes handled in a main memory of 32 MB: n = 800 and n = 1600. With 800×800 data matrices, the quantity of necessary memory to contain the three matrices participating in a multiplication (C = A×B) is of approximately 7.3 MB of data (approximately, 22.8% of the 32 MB of main memory). In the case of 1600×1600 data matrices, the quantity of required memory is of 29.3 MB approximately, which represents the 91.6% of the 32 MB of main memory.

Thus, all computers underwent experiments with square matrices of order n = 100, 200, 400, 800 and 1600. In the case of computers with a main memory of 64 MB or 512 MB, and in order to have reference values to be used in speedup calculations, experiments with greater matrices were also carried out. In addition, since there is always the tendency to use computers at their capacity limit, experiments were also carried out with the possible maximum matrix sizes. As expected, this depends not only on the size of the main memory installed but also on the swap space setup in the system.

For computers with 64 MB of main memory, the sizes considered as representative of the problems requiring most or all the maim memory correspond to the values of n = 1900, 2000, 2200, and 2400. These matrix sizes imply the following approximate percentages of memory requirements (assuming a total of 64 MB): 65%, 72%, 87%, and 103%, respectively. It should be born in mind that it is possible to test values near or above 100% of the main memory requirements depending on the size of the setup swap memory.

Even though when data take up all or most of the main memory it can be said that the computer is being used to its maximum (at least in terms of data in main memory), the extreme case is evident when the sizes exceeding the main memory are taken into account and when there is a need for the swap memory space. In 64 MB main memory computers, the maximum size with which the matrix multiplication could be carried out was for n = 3200, and, as reference, tests were also performed with n = 3000. These matrix sizes imply the following approximate percentages of memory requirements (assuming a total of 64 MB): 183% and 161%, respectively. As stated before, the maximum sizes of the problem depend on three aspects: a) installed main memory, b) setup swap space, and c) operating system; the latter being that which decides when a process is to be cancelled for lack of memory. And these three aspects match the speediest machines of the CeTAD network and the LIDI network.

For 512 MB main memory computers, the sizes considered as representative of the problems requiring most or all the main memory correspond to the values of n = 4000, 5000, 6000, and 7000. These matrix sizes imply the following approximate memory requirements (taking a total of 512 MB): 36%, 56%, 80%, and 110% respectively. Notice that it is possible to test with values near or above the 100% of the maim memory requirements depending on the size of the swap memory setup.

In 512 MB main memory computers, the maximum size with which the matrix multiplication could be carried out was for n = 9000, and, for reference, test with n = 8000 were also carried out. These matrix sizes imply the following approximate percentages of memory requirements (assuming a total of 512 MB): 181% and 143%, respectively.

4.2.2 CeTAD Local Area Network

The performance values obtained for each of the CeTAD local network computers are shown in Figure 4.4.

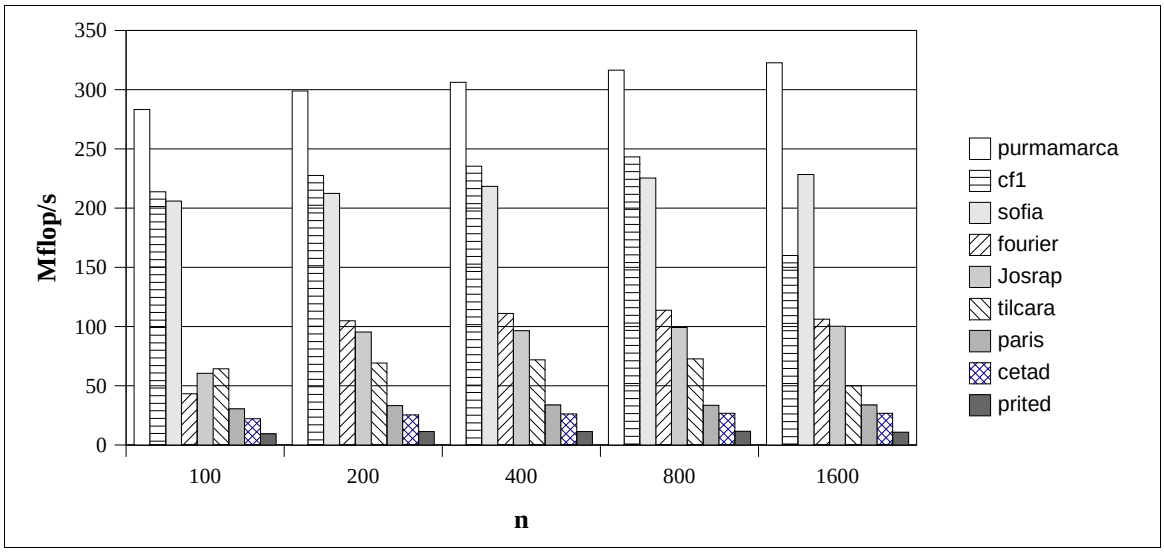


Figure 4.4: CeTAD Computers Performance.

Figure 4.4 shows that:

- **purmamarca** is the computer with the highest relative speed and is approximately 30 times faster in terms of processing than prited, with the lowest capacity.
- In reference to **cetadfomec1**, only **cf1** appears, since the performance of **cetadfomec2** is the same.
- It can be proved that, with the values of Table 4.1, the clock frequency at which the computers operate does not necessarily determine the processing capability (at least in floating point operations).

The performance characteristics and/or the values obtained are explained in detail in Appendix B.

The computer with highest processing power - **purmamarca** – underwent tests with

greater matrices and the results obtained appear in Figure 4.5.

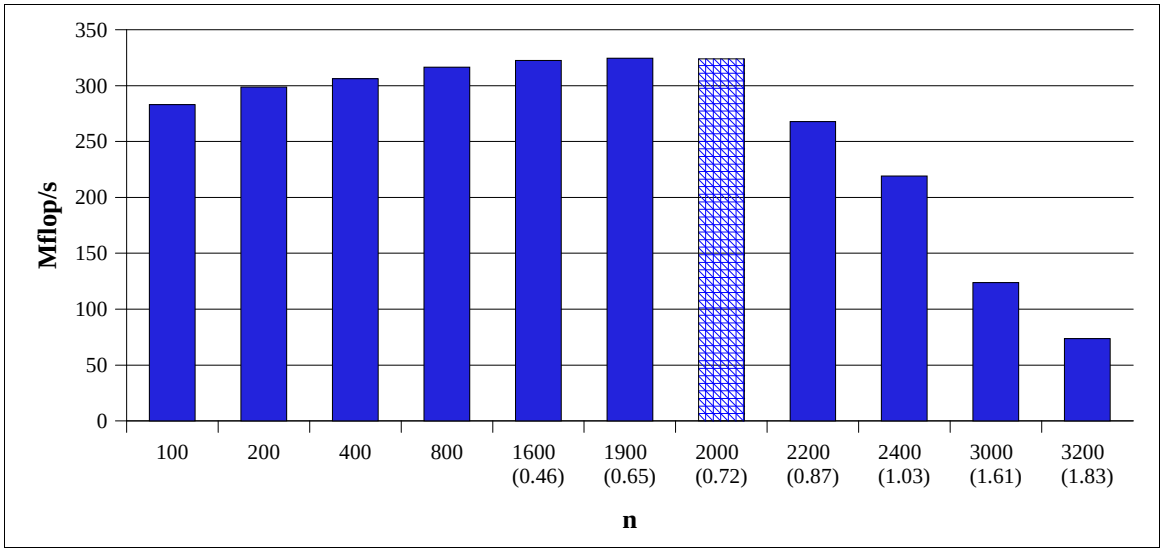


Figure 4.5: **Purmamarca**'s Performance for a Matrix Multiplication.

where:

- The memory proportion necessary for containing the three matrices is indicated between brackets. For n = 1900, for instance, the 65% of the memory is necessary for data storage.
- The largest matrix size for which swap space is not used appears highlighted (filled differently from the other bars). It must be noticed that, when the space to contain the data of the matrices exceeds the 72% of the memory, the swap space is used and, thus, the performance decreases.

4.2.3 LQT Local Area Network

Performance values for LQT local network computers are shown in Figure 4.6, where

- lqt_07 and lqt_06 computers are the speediest and the relative differences are not that great as in CeTAD computers.
- Since there are no space problems for bar representations, all machines are included, even though lqt_06 is equal to lqt_07, and lqt_03 is equal to lqt_04. It must be noticed that, in the context of PCs built in parts (even when computers are "equal" in terms of processor, system clock frequency and quantity of installed memory), it is still possible that they might have different performance because, for instance, they have different system bus speed.
- Once more, the operation clock frequency does not necessarily determine the relative speed of machines (see Table 4.2).

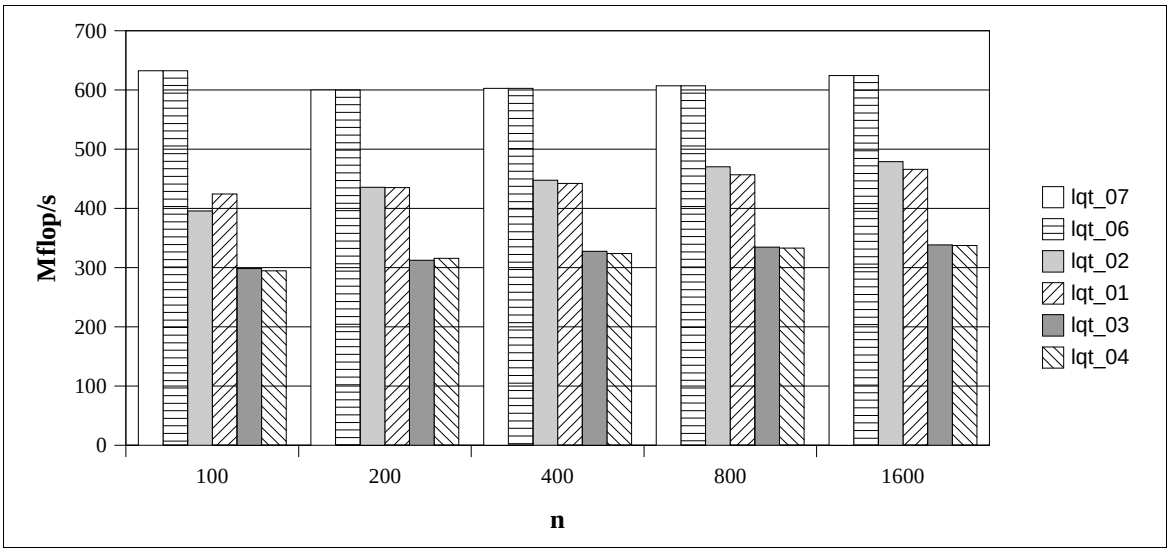


Figure 4.6: LQT Computers Performance.

The computer with greater processing power, **lqt_07**, underwent tests with bigger matrices, and the results obtained appear in Figure 4.7.

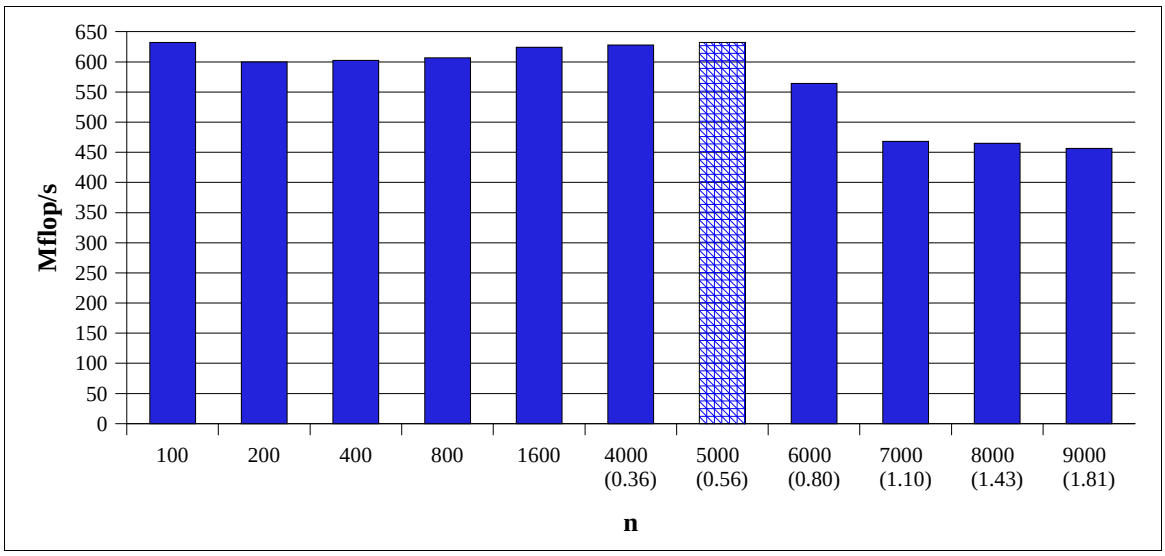


Figure 4.7: **lqt_07** Performance for a Matrix Multiplication.

Figure 4.7 shows that:

- The largest matrix size for which swap space is not used during the processing is *n*=5000. The memory destined to the storage of 5000x5000-element matrix data represents approximately the 56% of the whole memory.
- Again, the performance is reduced as more swap memory space is needed, even though this fall is not so abrupt with respect to that produced in **purmamarca** (Figure 4.5).

4.2.4 LIDI Local Area Network

Given the homogeneity of LIDI's machine, Figure 4.8 shows the results obtained in one of the machines **-lidipar14 -** for all matrix sizes.

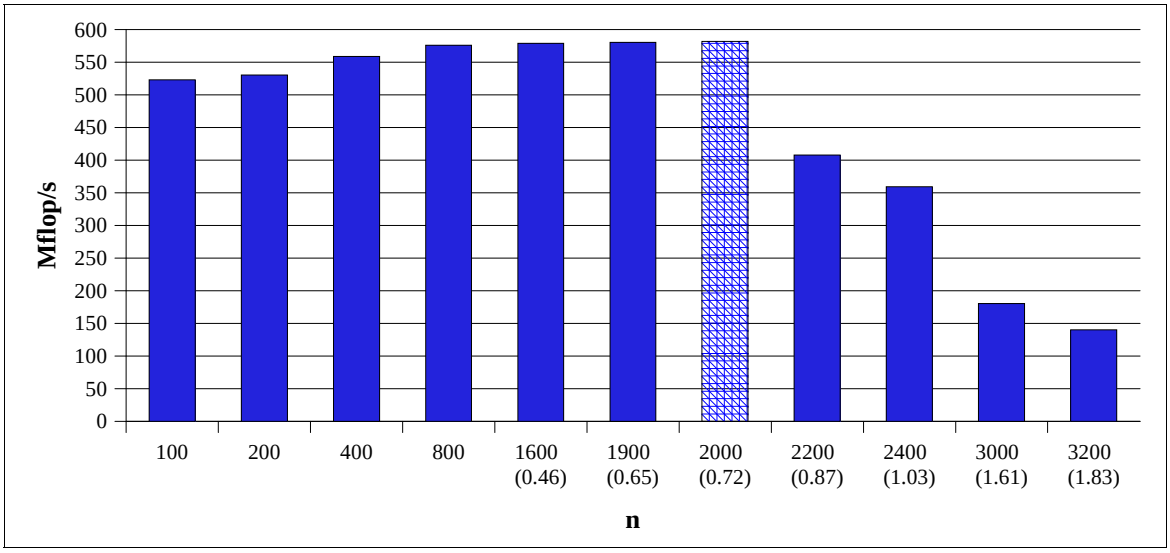


Figure 4.8: Performance of **lidipar14** for a Matrix Multiplication.

The performance of LIDI network computers is similar to that of **purmamarca** of the CeTAD local network since:

- The largest matrix size for which swap memory space is not used during the processing of matrix multiplication is n = 2000, all of which is coherent since they have the same main memory installed: 64 MB.
- From the very moment in which the swap memory space is used during the processing of a matrix multiplication, the performance falls abruptly.

However, LIDI local network computers are relatively faster, since they are capable of processing in the ratio of 580 Mflop/s, whereas **purmamarca** does not reach 350 Mflop/s.

Since computers of the LIDI local network are all equal, they can be used, and are used indeed, as a reference of the matrix multiplication algorithm in homogeneous networks. In addition, this network is the most suitable for parallel computation since the interconnection network is of 100 Mb/s (the others are of 10 Mb/s); also, in the wiring, a switch is used instead of one or more hubs.

4.3 Parallel Performance Analysis of Local Area Networks

There has been always an attempt to define and compute analytically the optimum or potential performance of parallel computers and also the performance attainable with a

parallel algorithm over a parallel computer in particular. The most important reasons for computing analytically the best possible performance of a parallel computer are:

- Estimating in advance whether the parallel computer is capable of providing a result in a given time. It would not be useful, for instance, to know in two weeks time the weather prediction of a given day of the next week.
- Assessing and comparing the parallel machines in order to, for instance, compute the cost/benefit relation between of each of them.

On the other hand, the analytical estimation of a parallel algorithm performance is useful in order to determine whether the designed algorithm is capable of obtaining the maximum performance of the computer for which it is implemented, and where it is executed to solve the posed problem.

Since

- the performance characteristics of each machine of all local networks,
- the performance characteristics of the interconnection network of each of the local networks, and
- the main characteristics of the parallel computing algorithms for the calculation of the matrix multiplication (with or without overlapped messages, for instance),

are already known, the best possible value for the speedup ratio can be calculated and used as reference in the subsequent assessment of the experimentation results. In a way, the analytical calculation of the best speedup value (or optimal speedup) tries to predict the performance of the workstation networks used as parallel machines.

4.3.1 *Actual* Speedup Calculation

The basic idea of a parallel computer speedup ratio is to determine how better a parallel computer is in terms of capacity in relation to a processor or a sequential computer. The classical definition of speedup is:

Execution time of the best sequential algorithm/ Parallel execution time

Thus, the *Execution time of the best sequential algorithm*, together with the *Parallel execution time*, should be determined. In the heterogeneous context of the installed computer networks, the

Execution time of the best sequential algorithm

"becomes" [125] the

Execution time of the best sequential algorithm in the fastest computer

or, briefly,

Execution time in the fastest computer

directly assuming that the best algorithm is going to be used. Basically, it is the best

possible sequential execution time in the computer-station network, i.e. using
- the computer with greatest computing capability, and
- the best sequential algorithm.

In the case of the three local networks already presented, this task is solved, since:
- In CeTAD local network, **purmamarca** is that of highest relative speed, and the experimentation has already determine its capacity in Mflop/s which, in turn, determines unambiguously the computing time.
- In LQT local network, **lqt_07** is that of highest relative speed, and the experimentation has already determined its capacity in Mflop/s which, in turn, determines unambiguously the computing time.
- In LIDI local network, all the computers are equal, and the experimentation has already determined the capacity in Mflop/s of **lidipar14** that, in turn, determines unambiguously the computing time.

Similarly, the *parallel execution time* is determined by experimentation, using the available machines of each of the local area networks.

4.3.2 *Optimal* Speedup Calculation

From the theoretical point of view, the best that can happen in a parallel machine is that all the processors are used all the time or that all the processors are used to their maximum computing capacity. This leads to the assumption that the computing capability of the parallel computer is equal to the sum of the computing capabilities of each processor being part of it. In the context of parallel machines with homogeneous processors, this means that using one more processor implies a proportional reduction of the parallel execution time. That is, if P processors are used, the best parallel execution time is given by

Parallel execution time = Execution time of the best sequential algorithm / P

In fact, this implies no more than the assumption that the computing power of the parallel machine with P processors is P times higher than the computing power of the sequential machines (a processor). In other words, speedup consists in identifying the relation between the power of a machine with a processor and a parallel machine with P processors. In this way, the optimal speedup in the *classical* parallel homogeneous computers is equal to the quantity of processors that are used. This is the same as defining the "the relative computing power of the parallel machine with respect to a processor" or, directly, the optimal speedup value as

$$OptimalSpeedup = \sum_{i=0}^{P-1} rpw(proc_i) \qquad (4.1)$$

where $proc_0$, $proc_1$, ..., $proc_{P-1}$, are the P processors of the parallel machine and $rpw(proc_i)$ is the relative computing power of $proc_i$ in relation to the rest or any of the other processors. In the context of numerical applications, it can be computed using the

computing power in Mflop/s as in the previous Chapter, but when assuming that the processors are equal,

$$rpw(proc_i) = 1; \quad \forall\, i = 0, ..., P\text{-}1 \tag{4.2}$$

and, thus

$$OptimalSpeedup = P \tag{4.3}$$

In the context of numerical applications, this is equal to the optimal speedup calculation using directly the computer powers of the sequential and parallel machine in Mflop/s, i.e. as

$$OptimalSpeedup = \frac{\sum_{i=0}^{P-1} Mflop/s(proc_i)}{Mflop/s(proc_0)} \tag{4.4}$$

In the context of parallel machines with heterogeneous processors, it is not possible to assert what Equation (4.2) expresses, since processors have or may have a different relative speed. Thus, the *Optimal Speedup* must be calculated according to Equation (4.1), i.e. using the calculation of each $rpw(proc_i)$, or as stated in the previous Chapter, $pw(ws_i)$, given by

$$pw(ws_i) = \frac{Mflop/s(ws_i)}{\underset{j=0..P-1}{máx}\ (Mflop/s(ws_j))} \tag{4.5}$$

And, similarly, Equation (4.4) is to be adapted to the heterogeneous environment as shown by Equation (4.6), which determines as reference the processor with highest computing power of those used.

$$OptimalSpeedup = \frac{\sum_{i=0}^{P-1} Mflop/s(ws_i)}{\underset{j=0..P-1}{máx}\ (Mflop/s(ws_j))} \tag{4.6}$$

In this way, two underlying ideas in the interpretation of speedup graphics arising from homogeneous parallel machines are also lost:

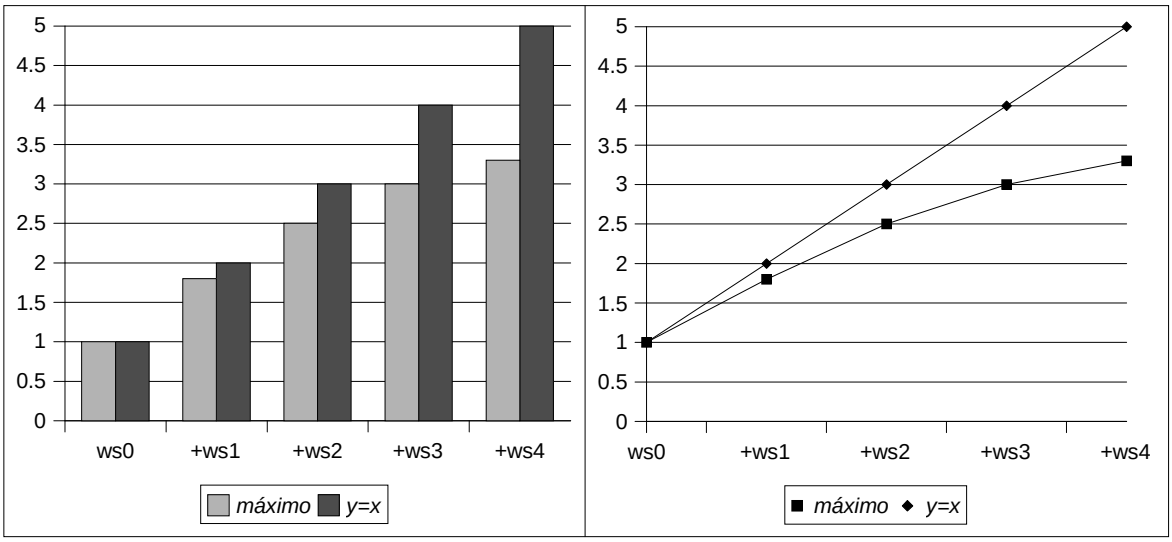
- It can no longer be asserted that the theoretical maximum is given by line $y = x$, or that for x number of processors the theoretical speedup maximum is given by x. In the heterogeneous environment it is no longer possible to relate the processors number with the complete parallel machine computing power. More specifically, adding a processor means adding computing power not necessarily related to the total number of processors but rather with the sum of the processors computing powers.

- It can no longer be asserted that a linear speedup could be reached as a minimum, or that, even when the speedup is not precisely equal to the number of processors, it should be directly proportional to the number of processors. It is not longer possible to keep this idea for the same reason stated above: it is not possible to relate or quantify the relation between the number of processors (or machines) and the parallel machine computing power.

Figure 4.9 shows the maximal speedup values in a network of five computers ws0, ..., ws4, each with their relative computer power given by

$$pw(\text{ws0}) = 1.0 \qquad pw(\text{ws1}) = 0.8 \qquad pw(\text{ws2}) = 0.7$$
$$pw(\text{ws3}) = 0.5 \qquad pw(\text{ws4}) = 0.3$$

Figure 4.9-a) shows with bars the speedup values, and in Figure 4.9-b) the values are connected with lines, showing quite more clearly how the maximum possible speedup for these five computers "moves away" from the line $y = x$ as computers are added. In addition, Figure 4.9 shows the tendency to incorporate the fastest computers of the available. In the case of using computers ws0 and ws1, ws2 is more likely to be incorporated since it has the highest computing power of the three available: ws2, ws3, and ws4; hence, in the speedup graphics, computers are added from "greater to lesser" according their computing capacity – unless some other criterion is established.



a) b)

Figure 4.9: Speedup of Five Heterogeneous Computers.

It is important to notice that this way of computing the maximum speedup value assumes that *all* computers, and in particular the speediest, *always* have the same computing capacity. When the computers are used to their maximum capacity set up in terms of swap space, it is possible and very likely that problems greater than the allowed by the main available memory will be solved and the swap space used. This, in turn, generates two well-known drawbacks from the point of view of performance:

- While the computation is carried out, there is more activity in the operating system due

to the data handling that should be transferred from and towards the swap space (generally, in the disk).

- It is possible and, in fact, very likely that the processor will have to wait for the data that are in swap space (disk) until they are transferred to the main memory from where the processor can make use of them in order to operate.

And, in fact, the performance declines markedly, though the quantification of this decrease depends on the computer (disk speed, input/output subsystem, etc.), and also on the problem (data access pattern, amount of data in swap space, etc.).

Thus, if the problem solved in the sequential computer or in the computer with highest computer capacity - in the case of the local networks - implies the use of swap space, a running time affected by the use of swap space will be taken as reference. When the possibility of executing in parallel in local network computers is taken into account, the distribution of the problem data is implicitly assumed and, thus, it is likely that the same size of problem will be solved so that each machine uses only the main memory and not the swap memory (with the implicit decrease of the performance). Therefore, depending on the size of the problem and the used machines, the obtainable speedup value can be greater than the optimum calculated according to Eq. (4.1). In other words, the optimal speedup value computed according to Eq. (4.1) assumes that the speediest computer *always* has the computing capacity established by the execution of part of the problem in swap space or, similarly, is slower than its real speed (when the whole problem can be handled in the main memory without recurring to the swap space).

Since sequential experiments implying the use of swap space are carried out, a non-"deviated" optimal value for the speedup should be provided. In the context of numerical problems, it is convenient to recur once more to the idea of computing capacity given in quantity of floating points operations per second, or Mflop/s. Since Mflop/s are used directly for this "new" way of optimal speedup computing, the quantity of operations to be carried out should also be taken into account. Recalling the example of the five computers ws0, ..., ws4, the capacity of each of them should be now used in terms of Mflop/s that could be for instance,

Mflop/s(ws0) = 1000
Mflop/s(ws1) = 800
Mflop/s(ws2) = 700
Mflop/s(ws3) = 500
Mflop/s(ws4) = 300

In addition, it is also necessary to know the number of floating point operations required to solve the problem, which could be, for instance, 10^9. Hence, if the computer with highest capacity, ws0, is able to solve the problem without recurring to the swap space, it then solves the computations to its maximum capacity, i.e. in the ratio of 1000×10^6 operations per second. In this case, the maximum speedup "coincides" with the computed using Eq.(4.1), i.e.:

$$OptimalSpeedup = \sum_{i=0}^{4} pw(proc_i) = \sum_{i=0}^{4} \frac{Mflop/s(ws_i)}{1000} = 3.3$$

If, on the contrary, the computer with highest capacity, ws0, uses the swap space during the resolution of the problem, it no longer carries out the computations to its maximum speed. Assuming that the degradation due to the use of the swap space during the computations is of 30%, this implies that computations are carried out in the ratio of 700×10^6 operations per second, thus assuming that during the parallel execution all computers operate to their maximum capacity; using Eq. (4.4), the maximum speedup would be:

$$OptimalSpeedup = \frac{\sum_{i=0}^{4} Mflop/s(ws_i)}{700} = 4.71$$

which is apparently higher to that computed according to the values *pw*(ws0), ..., *pw*(ws4). In any case, there would be two points of view for the calculation of the optimal speedup: that computed according to Eq. (4.1), and that computed by Eq. (4.3). The former depends, in turn, on the relative computer powers, *pw*(*wsi*), computed according to Eq. (4.5), which assumes that computers have always the same computing capacity, and which could be called "computing optimal speedup according to relative speeds", or Comp(rsf).

$$\mathrm{Comp}(\mathrm{rsf}) = \sum_{i=0}^{P-1} pw(proc_i) \tag{4.7}$$

The underlying idea supporting the computation of Comp(rsf) is basically the following: if a machine is added, its relative computing power is added to that of highest capacity. Following the example given by ws0, ..., ws4, this means that, if instead of using only ws0, ws0 and ws1 are used, there should be a (parallel) computer with 1.8 times the capacity of ws0.

The second point of view for the calculation of the optimal speedup is that computed by Eq. (4.4), which assumes that all computers always run at their maximum capacity, independently of whether it is necessary to use the swap space during the processing. Thus, it could be called "computing optimal speedup according to each computer's computing capacities given in Mflop/s", or Comp(Mf).

$$\mathrm{Comp}(\mathrm{Mf}) = \frac{\sum_{i=0}^{P-1} Mflop/s(ws_i)}{\max_{j=0..P-1}(Mflop/s(ws_j))} \tag{4.8}$$

The underlying idea supporting the computation of Comp(Mf) is basically the following: if a machine is added, its computing power is directly added in Mflop/s. Following the example given with ws0, ..., ws4, this means that, instead of using only ws0, ws0 and ws1 are used; then, there should be a (parallel) computer with a computing capacity of 1000+800 Mflop/s = 1800 Mflop/s, which implies that the parallel computer has 1800/7002.57 times the capacity of ws0, since the reference value in terms of Mflop/s of

ws0 is 700 Mflop/s, obtained by using the swap space.

It is evident that the optimal speedup computation will be the same, Comp(rsf)=Comp(Mf) if the maximum Mflop/s values are taken as reference, i.e. without using the swap space of each machine. Comp(rsf) is derived directly from the classical way of computing speedup, and Comp(Mf) should be viewed rather more carefully when the sequential running time is affected by the use of the swap space. In a way, when the swap memory space is used during the sequential running, Comp(Mf) could be understood as what has been occasionally called "superlinear speedup". Following the example from this point of view:

- Solving a problem in ws0, the running time derived from ws0's capacity is obtained when using the swap space, i.e. directly proportional to 700 Mflop/s.
- Solving the same problem with parallel computation and using ws1, the problem is "expected" to be solved considering a computer with a capacity of (700 + 0.8x700) Mflop/s = 1260 Mflop/s, since ws1 has 0.8 times the computing capacity of ws0.
- If the distribution of the whole problem between ws0 and ws1 makes the use of the swap space unnecessary, both computers solve computations to their maximum capacity and, thus, the parallel running time would be proportional to (1000+800) Mflop/s = 1800 Mflop/s and, thus, the running time will be less than the "expected" considering only the relative speeds.

Figure 4.10 shows the different values of optimal speedups, Comp(Mf) and Comp(rsf), taking into account the five computers of the example, whose performance characteristics are summarized in the following table,

Computer	***Mflop/s (Maximum)***	***pw***
ws0	1000	1
ws1	800	0,8
ws2	700	0,7
ws3	500	0,5
ws4	300	0,3

and also considering the fact that the computer with highest computing capacity uses the swap space to solve the given problem with its subsequent performance penalization of 30%. In Figure 4.10-a) the bars represent the values, and in figure 4.10-b), the lines join the values showing clearly that:

- The different ways of computing the optimal speedup provide various values if the sequential solution implies the use of the swap space.
- Line $y = x$ does not provide any meaningful information in the context of heterogeneous processors.

Lastly, it should be noticed that, in the optimal speedup computation there is no consideration in terms of communications; *only* the computing capacity of all the processors (computers) used is taken into account.

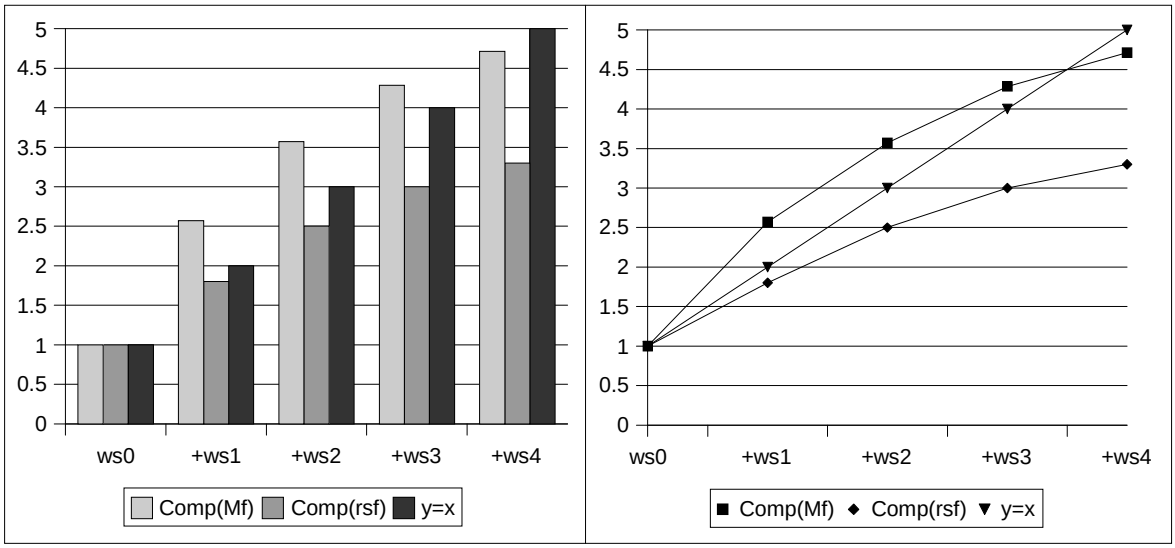


Figure 4.10: Computations of Optimal Speedups for Five Computers.

4.4 Analysis of Algorithms Performance

Normally, the analytical computation of parallel algorithms performance has two general purposes:

- To determine whether the algorithm is capable of taking advantage of the parallel computer's performance on which it can be implemented.
- To compare and assess different designed algorithms for the same task.

The analytical computation of the algorithms performance not only takes (or should take) into account the computers computing characteristics but also incorporates at least another factor affecting the performance: communications. Even though the parallel computer architecture is already known in detail, it is really difficult to quantify the effect communications have on the performances of the applications to be solved. However, the situation changes when a specific parallel algorithm is designed, since this last clearly determines what is needed in relation to communications and the synchronization among processes. In general, synchronization points among processes are considered as a type of communication in particular.

Although they are very similar, the algorithms to be considered for the analysis are two, and have been already presented in the previous chapter with:

- the explicitly sequential messages with respect to computation. That is, at any moment, a computer can be performing one of the following two tasks:
    - Local computation, i.e. solving a partial computation of the result matrix portion that should be computed.
    - Solving data communication, more specifically sending and receiving a

broadcast message.

- overlapped messages, so most of messages can be sent while the local computation is being carried out (simultaneously). In fact, in order to make this true, each computer should be capable of computing and delivering data at the same time.

Both algorithms were already presented in the previous Chapter, together with the analytical form for each performance computation; they will be called SeqMsg and OverMsg, respectively. In all of the cases, it is assumed that computing periods are carried out to the maximum computing capacity of the involved computers.

4.4.1 SeqMsg: Sequential Computing and Communication

According to what has been explained in the previous Chapter, the parallel time of the algorithm during which the computing and communication periods are sequentially carried out is given by

$$t_{par\_seqmsg} = P\alpha + \beta n^2 + \frac{2n^3 - n^2}{pw}$$

where

- P is the number of computers.
- n is the order of the square matrices that are multiplied.
- α is the latency time of the communication network.
- $1/\beta$ is the asymptotic bandwidth (transfer rate) of the communications network, expressed in terms of the type of the multiplied matrices elements.
- pw is the sum of all of the computers' computing capacities used, expressed in terms of Mflop/s, i.e.

$$pw = \sum_{i=0}^{P-1} Mflop/s(ws_i) \tag{4.9}$$

Even though it is not explicitly defined, there is a tendency to assume that

- The latency time is not so important provided that messages are big enough, or, what would be the same, the problem is big enough, since the messages size is directly related to the size of the problem [71] [52] [124].
- The asymptotic bandwidth of the communications network is independent of the number of processes that are communicated with a broadcast or, more specifically, the number of receiving processes of each broadcast message. This is possible in Ethernet networks provided that communication routines take advantage of the communication hardware characteristics.

Thus, the way of computing the parallel running time can be simplified eliminating the latency time of messages completely (or, what would be the same, considering equal to zero), thus obtaining

$$t_{par\_seqmsg} = \beta n^2 + \frac{2n^3 - n^2}{pw}$$

And this parallel time is used for the computation of the optimal speedup obtainable by this algorithm in a network of workstations, giving place to what will be called **SeqMsg(Mf).**

4.4.2 OverMsg: Overlapped Computing and Communications

According to what has been explained in the previous Chapter, the parallel time of the algorithm during which most of the computing and communication periods are carried out in an overlapped (simultaneous) manner is given by

$$t_{par\_overmsg} = t_{bcast} + (P\text{-}1)\ max(t_{bcast}, t_{comp}) + t_{comp}$$

where

$$t_{bcast} = \alpha + \beta n^2/P \qquad \text{and} \qquad t_{comp} = \frac{2n^3 - n^2}{P\ pw}$$

And this parallel time is used for the computation of the optimal speedup obtainable by this algorithm in a network of workstations, given place to what will be called **OverMsg(Mf)**.

By this way of computing analytically the parallel running time, it is assumed that:
- All computers are capable of overlapping the computation with communications.
- Communications overlapping does not affect the local computing time nor the communication time of the broadcast messages.

It should be noticed that both assumptions are really difficult to prove, at least in standard computers of the installed local networks.

4.5 Local Area Networks and Algorithms

Since we already count with:
1. the sequential performance of all computers of every local network: Mflop/s(*wsi*);
2. the analytical way to compute the performance of each of the local networks Comp(Mf) and Comp(rsf);
3. the analytical way of the two proposed algorithms: t*par_seqmsg* and t*par_overmsg*;
4. the asymptotic bandwidth estimation -at least at hardware level- of all the local networks: Mb/s of the Ethernet networks;

it is now possible to estimate the performance of both each network and these networks' algorithms, at least with respect to the *theoretical* maximum.

4.5.1 CeTAD Local Area Network

Figure 4.11 shows the maximum speedup values to be considered in the CeTAD local network when the main memory is capable of containing all the data of the problem, so that the utilization of the swap memory becomes unnecessary during computations. In this case, the reference computer (that of highest computing capacity) is **purmamarca,** and the matrices size is n = 2000. In addition, since an Ethernet network of 10 Mb/s is used, it is assumed that, due to the degradation caused by all the layers of the operating system, data can be transferred among user processes in the ratio of 220 bytes (1 MB) per second. This could be considered as an optimistic, though acceptable, assumption since the maximum values are estimated.

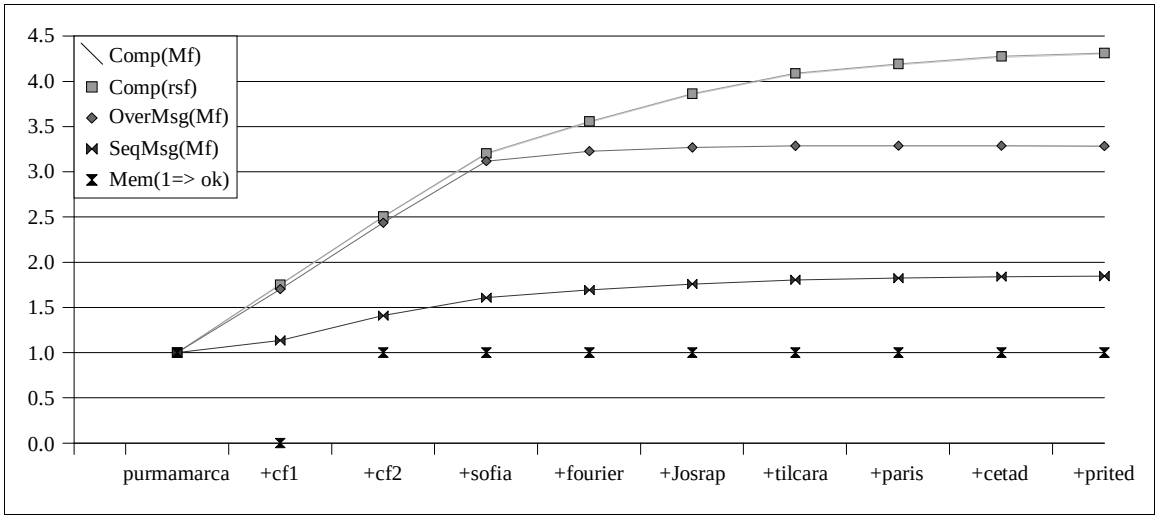


Figure 4.11: Speedup Analysis of the CeTAD network for n = 2000.

Figure 4.11 shows that:

- Computers' relative computing capacity provides a reasonable order to use them in parallel. When a given quantity of computers is used for parallel computation, there is a tendency to include those of greater computing capacity among the available.
- Computers **cetadfomec1** and **cetadfomec2** are shown with the names **cf1** and **cf2**, respectively.
- As expected, Comp(Mf) and Comp(rsf) do coincide, since the reference computing capacity of **purmamarca** is the maximum (approximately 324 Mflop/s, Figure 4.33) because it does not use the swap space during the sequential running.
- The speedup computed for the algorithm with the overlapped messages OverMsg(Mf) is similar to that of the computation, until the computer **sofia** was used; however, when adding more computers there is almost no improvement in terms of performance. In other words, as from the addition of **fourier**, communication times are longer than those of the computation and, thus, there is almost no improvement in terms of parallel running time by the incorporation of more computers.
- The relative weight of the communication time in relation to the computing time is evident for the algorithms with sequential messages and computation. The computed speedup for this algorithm, SeqMsg(Mf), thus shows it through the difference in the

values in relation to the other speedup computations, including that of the algorithm with overlapped messages that takes into account at least part of the communications total time.

- The whole computing power of the ten computers is slightly less than 4.5 times the computing power of **purmamarca.**

Also, Figure 4.11 shows an "empirical" estimation of memory requirements indicated in the graphic as Mem (1 => ok). When it has a value equal to 0, it is likely that, in one or more computers, it will be necessary to recur to the swap space during the execution. When it has a value equal to 1, it is likely that the swap space will not be necessary in *any* of the computers during the algorithm execution. Notice that in the graphic it appears equal to 0 only when two computers are used in parallel: **purmamarca** and **cf1.** This is due to the following facts:

- When **purmamarca** is only used –which has a main memory of 64 MB (Table 4.1)–, experimentations show that it is not necessary to use the swap space.
- When **cf1 –**which has a memory of 32 MB - is added, it is likely that **purmamarca** will not need to recur to the swap space, though **cf1** will.
- When **cf2** is added, there already exist three machines among which data are distributed and, from this moment, it is actually possible that there will not be any problems for the memory.

Even though this memory estimation is not so precise (and, in fact, it is really difficult to make one which really is), it is always useful to have at least one reference idea since, as previously explained, the computing capacity can be really affected.

Figure 4.12 shows the maximal speedup values to be considered in the CeTAD local network for the maximum problem size that can be solved by the computer with greatest computing capacity (recurring to the swap memory). The reference computer is still **purmamarca** and the size of matrices is n = 3200. As for the previous estimation, it is assumed that the data can be transferred among user processes in the ratio of 220 bytes (1 MB) per second.

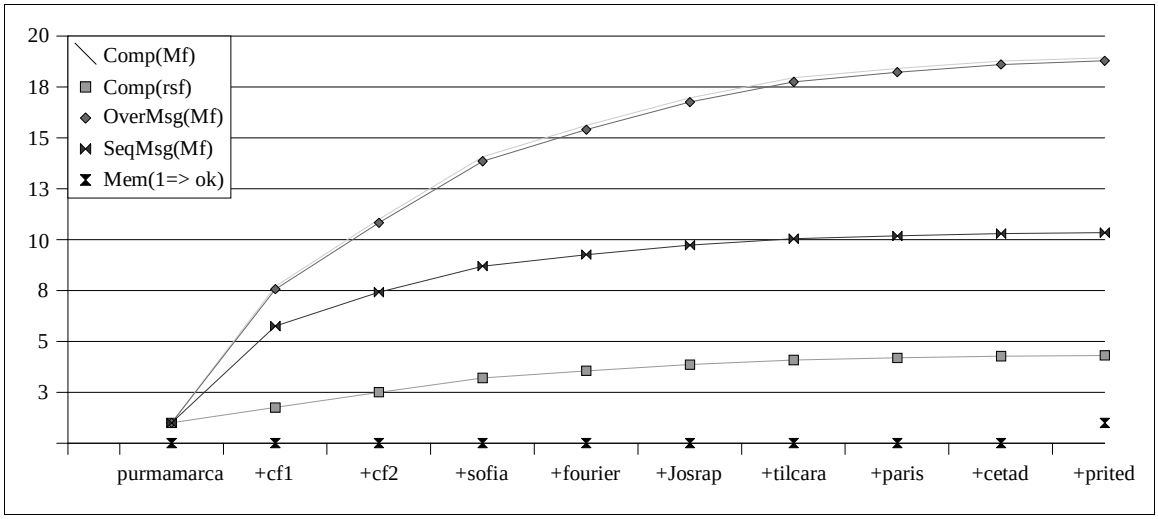


Figure 4.12: Speedup Analysis of the CeTAD Local Network for n = 3200.

Since the greatest problem (with 3200x3200-element matrices) that can be solved by **purmamarca** implies the use of the swap space, the reference computing capacity is of approximately 74 Mflop/s (Figure 4.33). Therefore, it is no longer possible for Comp(Mf) to coincide with Comp(rsf); more specifically, the optimal speedup value computed with the maximal computing capacity (in Mflop/s) will necessarily be greater than the optimal speedup value taking into account the relative speeds among computers. This implies that the maximal speedup value using all CeTAD machines (10 computers) results in a parallel computer that has almost 19 times **purmamarca**'**s** computing capacity for matrices of 3200x3200 elements, i.e. when **purmamarca** has to recur to the swap space during the execution.

In addition, from Figure 4.12 it can be said that:

- The values of Comp(rsf) do not change in relation to those of Figure 4.11, since relative speeds are taken as equal.
- Assuming that each computer can efficiently solve its computations and communications simultaneously, the values of OverMsg(Mf) are almost equal to those of Comp(Mf).
- The weight of the communication time is still relatively high in relation to the computing time, and this is evidenced by the differences between the values of Comp(Mf) and SeqMsg(Mf). More specifically, when the communication time is taken into account plus the computing time (as it should be done for the algorithm with message transmission and computing periods solved sequentially), the optimal speedup values are markedly reduced with respect to those obtained with the sum of the computing capacities.
- Since for n = 3200, **purmamarca "**becomes" a computer with much lesser computing capacity than for n = 2000, even with sequential computations and communications the profit is expected to be significant, and rather higher than expected according to relative speeds. The rather lower values of Comp(rsf) thus show it.
- The estimation of memory requirements in each computer - Mem in the graphic- shows that until only all machines are used, the requirement for swap memory becomes less necessary in all computers. This is mainly due to the fact that two of the computers with highest relative computing capacity, **cf1 and cf2**, have low memory capacity in relation to that with the highest computing capacity (**purmamarca**).

4.5.2 LQT Local Area Network

Figure 4.13 shows the maximum speedup values to be considered in the LQT local network when the main memory is capable of containing all the data of the problem, so that the use of the swap memory becomes unnecessary during computations. In this case, the reference computer (that of highest computing capacity) is **lqt_07** and matrix sizes are of n = 5000. Also, in this case, since an 10 Mb/s Ethernet network is used, it is assumed that due to the degradation produced by all the operating system layers, data can be transferred among user processes in the ratio of 220 bytes (1 MB) per second.

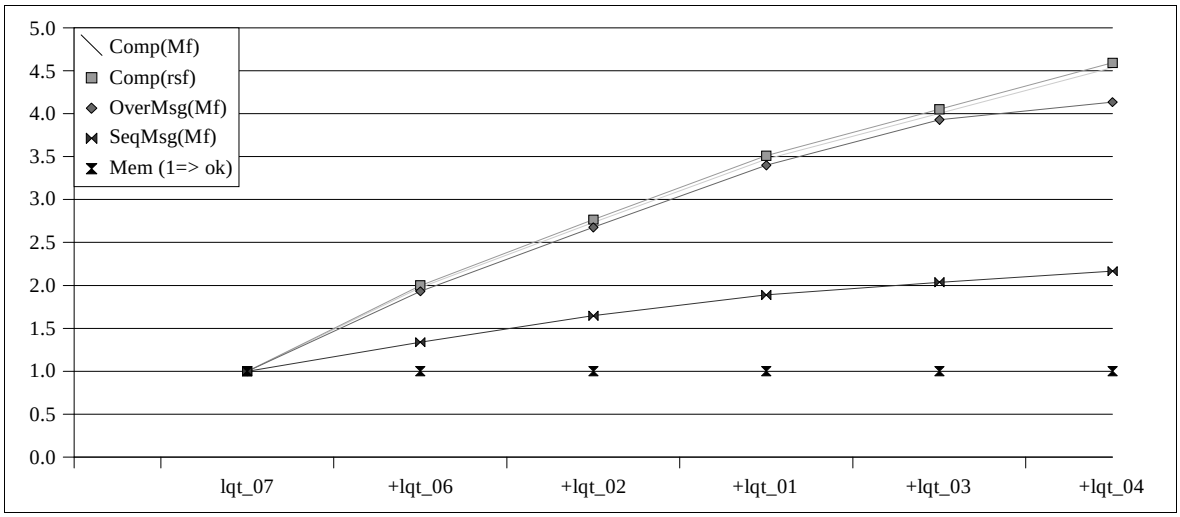


Figure 4.13: Speedup Analysis of the LQT Network for n = 5000.

The parallel computer *obtained* using all the machines provides, in the best of the cases, slightly more than 4.5 times the capacity of **lqt_07**, approximately 2.9 Gflop/s. Once more, the analytical computation of the maximum, possible speedup obtainable with the sequential computation and communication algorithm shows the relative weight of the communication time in relation to the computing time, and it is kept in values approximately equal to the half of the other estimations values.

Figure 4.14 shows the maximum speedup values to be considered in the LQT local network for the maximum size of problem that can be solved by the computer with highest computing capacity (recurring to the swap memory). The reference computer is still **lqt_07** and the size of matrices is n = 9000. As for the previous estimation, it is assumed that data can be transferred among user processes in the ratio of 220 bytes (1 MB) per second.

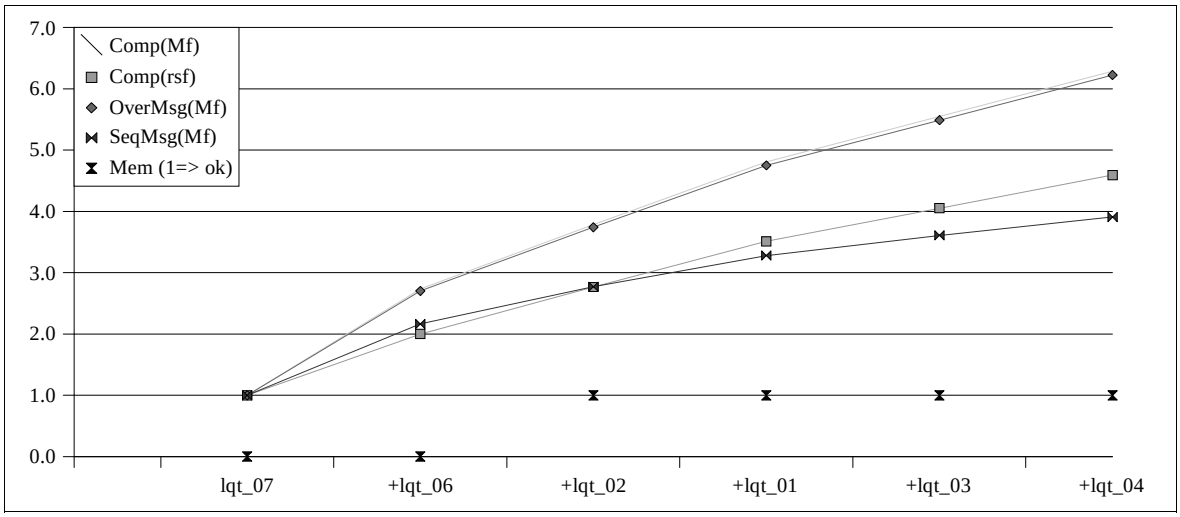


Figure 4.14: Speedup analysis of the LQT Network for n = 9000.

From Figure 4.14, it can also be deduced that:

- When all computers are used to solve the multiplication problem of 9000×9000 element matrices, the running time *could* be reduced more than six times the running time of **lqt_07**, even when, according to the relative speeds, this reduction might not reach five times.
- If it is possible to overlap completely local computations with communications, then the possible and attainable maximum speedup is similar to the absolute maximum. In other words, the computing time is equal or higher than that of communications.
- When messages and local computations are carried out sequentially, the running time will be longer than expected, taking into account computers' relative speeds as from the addition of **lqt_01**, i.e. Comp(rsf) > SeqMsg(Mf) as from the addition of **lqt_01**.
- According to memory estimations, it would not be necessary to recur to the use of swap memory as from the addition of **lqt_02**.

4.5.3 LIDI Local Area Network

Figure 4.15 shows the maximum speedup values to be considered in the LQT local network when all the data of the problem can be contained in the main memory, so that the use of the swap memory becomes unnecessary during computations. In this case, the reference computer (that of highest computing capacity) is **lidipar14**, but it should be remembered that all computers are equal, and the size of matrices is n = 2000. Unlike the previous local networks, a 100 Mb/s Ethernet network is used assuming that, due to the degradation caused by all the operating system layers, data can be transferred among user processes in the ratio of 10×220 bytes (10 MB) per second. As in the previous cases, an optimistic, though acceptable, assumption could be considered because maximum values are being estimated.

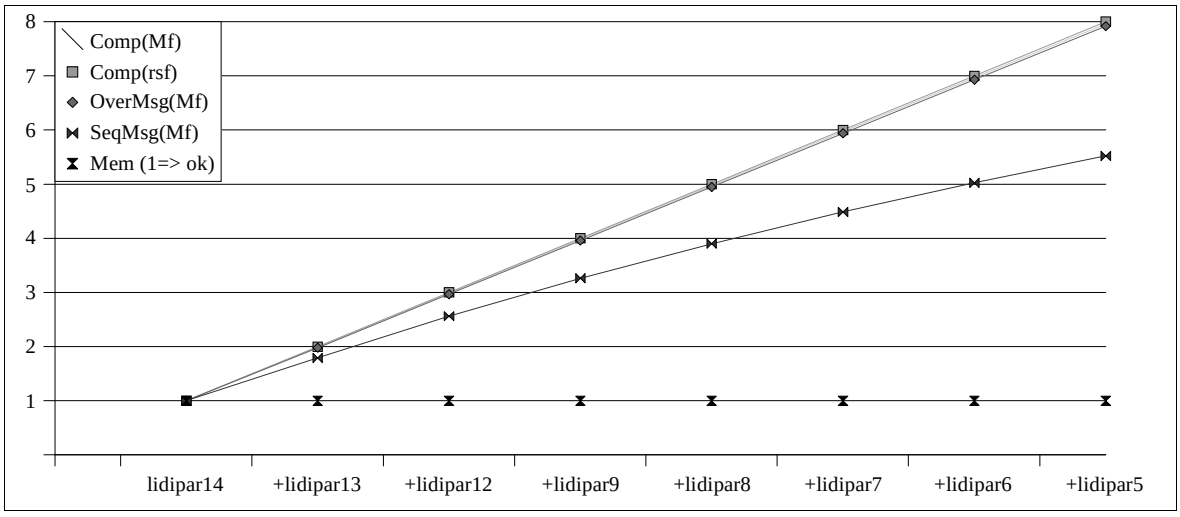


Figure 4.15: Speedup Analysis of LIDI Network for n = 2000.

Speedup values appearing in Figure 4.15 more or less coincide with those of *classical* (homogeneous) parallel computers, since:

- Comp(rsf) values *correspond* to line $y = x$.

- At least until the addition of **lidipar5,** all maximum estimated values for the speedup follow a lineal growing pattern with respect to the quantity of processors (computers) used, even considering the sequential computation and communications algorithm.

Also, from Figure 4.15, it can be said that:

- Comparing this network to the previous ones, and having an interconnection network ten times better in terms of bandwidth, communications do not have such a big relative weight. In fact, the algorithm carrying out computation with communications overlappedly has an optimum speedup equal to the computed without taking into account communications, OverMsg(Mf) ≅ Comp(Mf).
- With the algorithm carrying out sequential computation and communications, a (parallel) computer with slightly more than 5.5 times the computing capacity of **lidipar14 (**or any of the others, since they are all equal**)** can be obtained by using the eight available computers - in the best of the cases**.**
- As computers are all equal, the relation between communications and computing time and the maximum speedup values of the sequential computation and communications algorithm - SeqMsg(Mf) - can be more clearly identifiable than in the previous cases (CeTAD and LQT). As the number of machines increases, the same work is distributed among all of them. Consequently, the total computing time decreases (there is more quantity of computers processing simultaneously), though the total communication time is still the same. Thus, as more machines are added, the communication time affects more significantly the total running time (computation plus communications).

In order to exemplify this last point, the specific values computed for four and eight machines can be taken into account. Independently of the quantity of computers that are used, when the size of the matrices is the same (in this case n = 2000), the quantity of data that are communicated is the same since the data of matrix B should always be transmitted among computers. The estimated transmission time of matrix B (sum of the algorithm broadcast messages times), in a 100 Mb/s Ethernet network, is of approximately 1.5 seconds. When **lidipar14**, **lidipar13**, **lidipar12**, and **lidipar9** are used, the estimated computing time is approximately 13.8 seconds. When the computers **lidipar8**, **lidipar7**, **lidipar6**, and **lidipar5** are added to the previous, the estimated computing time is of 3.4 approximately. Thus when messages and communications are run sequentially:

- The total running time, when four computers are used, is (summing computing time and communication time) 1.5 + 13.8 = 15.3 seconds. This implies that around the 10% of the total running time is used for communications.
- The total running time, when eight computers are used, is (summing computing time and communication time) 1.5 + 3.4 = 4.9 seconds. This implies that around the 30% of the total running time is used for communications.

Figure 4.16 shows the maximum speedup values to be considered in the LIDI local network for the maximum size of problem that can be solved by the computer with highest computing capacity (recurring to the swap memory). The reference computer is still **lidipar14** and the matrices are of 3200×3200 elements. As for the previous estimation, it is assumed that data can be transferred among user processes in the ratio of 10×2^{20} bytes (10MB) per second.

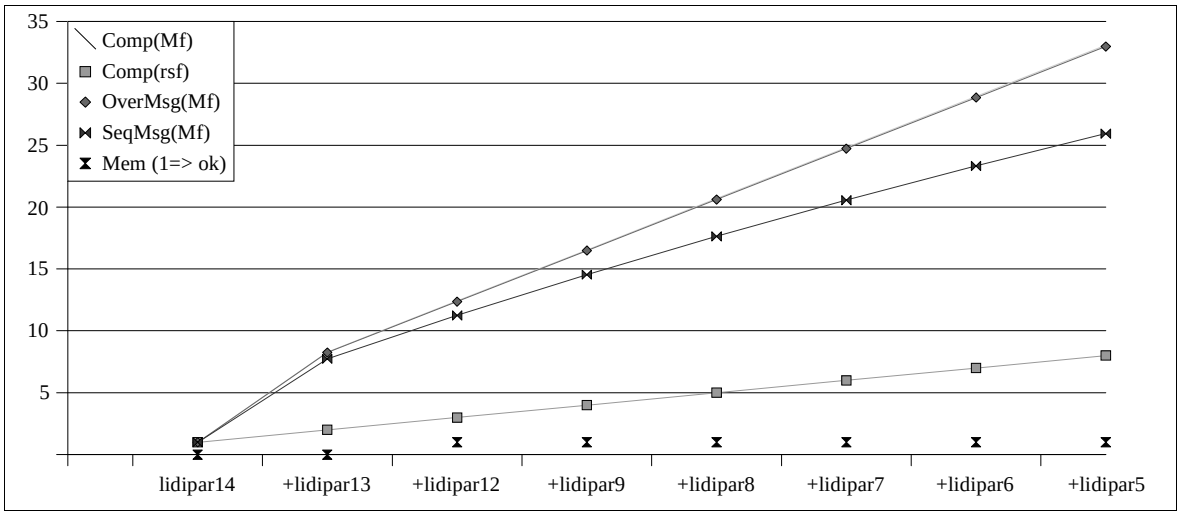


Figure 4.16: Speedup Analysis of the LIDI Network for n = 3200.

Apart from showing that the maximum speedup values computed by using relative speeds - Comp(rsf) - do not change in relation to those shown in Figure 4.15, in Figure 4.16 it can also be seen that:

- The problem of multiplying matrices of 3200×3200 elements *could* be solved almost 35 times faster in the eight computers than in one of them. This is asserted not only by the computing power of the eight machines processing to their maximum capacity (without considering communications), Comp(Mf), but also by the algorithm that runs computation overlapped with communications, OverMsg(Mf).
- With the algorithm that sequentially solves communications with computations, the problem of multiplying 3200×3200 element matrices *could* be solved almost 25 times faster in the eight computers than in one of them.
- Both the performance penalization due to the use of swap space in **lidipar14** for 3200×3200 element matrices and the interconnection network data transference rate are combined to have these “superlinear” speedup values.
- Except for one or two machines, memory requirements estimations do not identify potential problems in terms of the need to use the swap space.

4.6 Actual Performance of Local Networks Using PVM

The algorithms proposed in the previous Chapter were directly implemented by using the PVM library (Parallel Virtual Machine) for communication routines among processes. In each computer, the best sequential code is used for local computing periods. In each local network (CeTAD, LQT, and LIDI) the same experiments were carried out, i.e.:

- Matrix multiplication with the sequential computation and messages algorithm (SeqMsg).
- Matrix multiplication with the overlapped computing and messages algorithm (OverMsg).
- Size of matrices
    - so that, in the machine with greatest computing capacity, the swap space will not

be used;
- the biggest possible – without the use of swap space.
- In the local network, this corresponds to matrices of order
- n = 2000 and n = 3200, respectively in the local networks of CeTAD and LIDI.
- n = 5000 and n = 9000 respectively in the LQT local network.

4.6.1 CeTAD Local Area Network

Figure 4.17 shows the speedup values obtained in the CeTAD local area network by the sequential computation and communication algorithm, and by the overlapped computation with communication algorithm, implemented by means of the PVM message-passing library, SeqMsg(PVM) and OverMsg(PVM) respectively, for n = 2000, together with those shown previously in Figure 4.11.

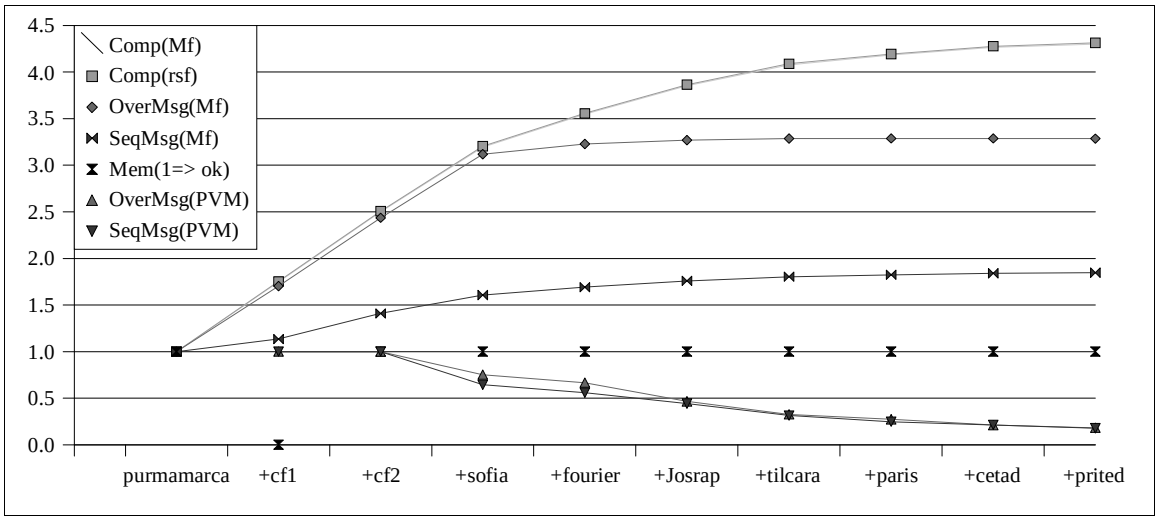


Figure 4.17: Algorithms Speedup with PVM in the CeTAD Network for n = 2000.

It is clear that the results are far from being satisfactory. In fact, the two most disappointing conclusions are
- None of the running times, i.e. independently of the quantity of computer used, was better than the sequential execution, with the problem solved in **purmamarca.**
- As more computers are used, the running time increases instead of decreasing.

Moreover, algorithms' speedup values that are equal to one - i.e. for the cases in which **purmamarca** and **cf1,** and **purmamarca**, **cf1,** and **cf2,** are used respectively- are not even *real*. In both cases, the execution of parallel program processes in **cf1** and / or **cf2** is cancelled due to the lack of available memory. In consequence, they appear to be equal to one because, in fact, the only feasible chance of solution for the matrix multiplication in this context is the sequential running (using only **purmamarca**). Even though there exists an approximation to memory requirements (Mem, in the graphics), it is evident that with PVM that approximation is not correct.

Even leaving aside the memory problem, the problem of performance is evident. In

principle, the alternatives of what caused the low performance obtained could be:

- Low computing performance, when solving each of the intermediate computations. This problem is basically related to the performance in terms of computation of each computer.
- Low communications performance, when sending and receiving broadcast messages. This problem is basically related to PVM and the interconnection network. In this sense, there are two possibilities:
    - PVM does not properly implement broadcast messages, or
    - communications among user processes from different computers are highly penalized in terms of performance with respect to the network interconnection capacity.

Figure 4.18 shows the speedup values obtained in the CeTAD local network by the sequential computation and communication algorithm, and computation overlapped with communication algorithm, implemented by means of the PVM message-passing library, SeqMsg(PVM) and OverMsg(PVM) respectively, for n = 3200, together with those shown in Figure 4.12.

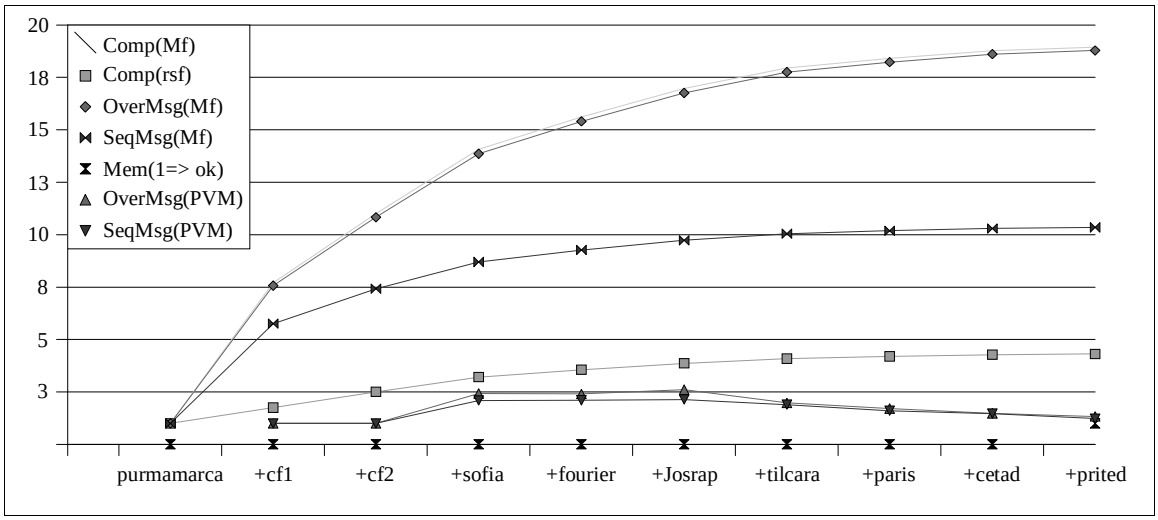


Figure 4.18: Algorithms Speedup with PVM in the CeTAD Network for n = 3200.

In this case, i.e. taking as reference the computing power of **purmamarca** to multiply the square matrices of the order n = 3200:

- Once more, memory requirements in **cf1** and **cf2** make the parallel program running possible and processes are cancelled by the operating system.
- In the best of the cases, speedup values close to three are obtained when all the estimations are higher for the same quantity of machines, including that which takes into account only the relative speeds: Comp(rsf).
- From the addition of **tilcara** in the parallel machine, the parallel running time gets worse, reaching –with the ten machines- speedup values of approximately 1.33 with the algorithm that carries out overlapped computation and communications, and 1.23 with the algorithm that carries out sequential computation and communications, OverMsg(PVM) and SeqMsg(PVM) respectively.
- Algorithms speedup estimations are very far from the values obtained.

4.6.2 LQT Local Area Network

Figure 4.19 shows the speedup values obtained in the LQT local network by the sequential computation and communication algorithm, and by the overlapped computation with communication algorithm, implemented by means of the PVM message-passing library, SeqMsg(PVM) and OverMsg(PVM) respectively, for n = 5000, together with those shown before in Figure 4.13.

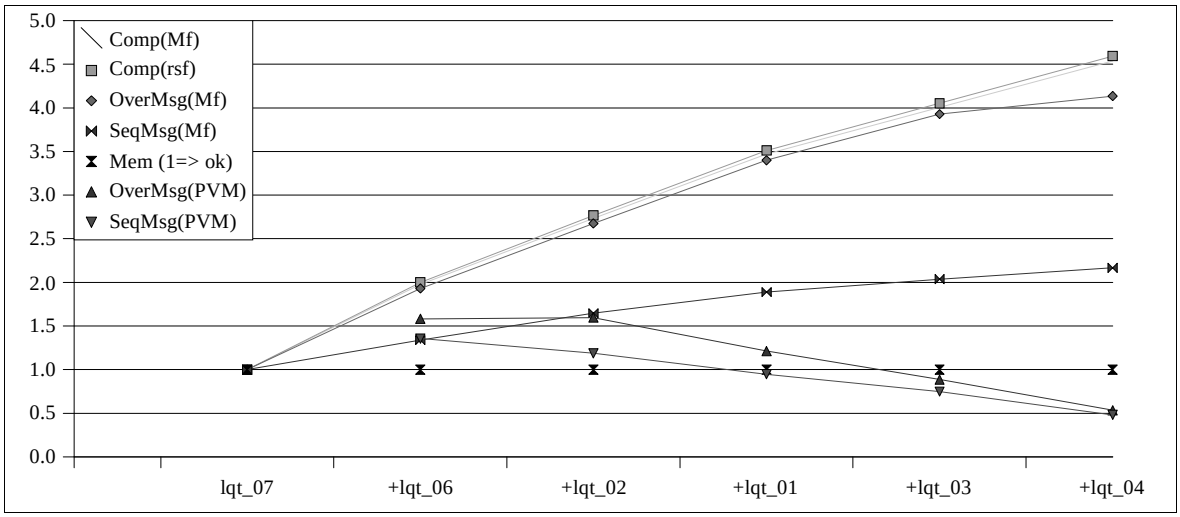


Figure 4.19: Algorithms Speedup with PVM in the LQT Network for n = 5000.

Comparing these results with those corresponding to CeTAD (Figure 4.17), the likelihood is remarkable. Basically, the characteristics of the speedup values obtained are the same:

- Almost no performance is gained by using machines processing in parallel,
- In most of the cases, adding machines to process in parallel implies loss of performance, even though the computers and the size of the problem are very different form each other. In consequence, these results confirm that there exists one or more problems and that the problem/s are not typical of the CeTAD local network nor of the LQT local network.

Figure 4.20 shows the speedup values obtained in the LQT local network by the sequential computation and communication algorithm, and the overlapped computing with communication algorithm, implemented by means of the PVM message-passing library, SeqMsg(PVM) and OverMsg(PVM) respectively, for n = 9000, together with those shown in Figure 4.14. Similarities in terms of speedup values in relation to the CeTAD in a similar context (Figure 4.18) are, once more, rather evident, despite the differences between the machines and the size of the problem:

- Memory requirements imposed by the parallel computation with PVM communications routines make the operating system (when two computers are used, **lqt_07** and **lqt_06**) cancel one or several processes involved due to the lack of memory. For this reason, the graphic shows the speedup equal to one for two computers for both algorithms.
- Up to a given quantity of computers, speedup increases. In this case, up to the addition

of **lqt_02** for the sequential computation and communication algorithm, SeqMsg(PVM) in the graphic, and up to the addition of **lqt_01** for the overlapped communications algorithm with computation OverMsg(PVM) of the graphic.

- The use of all the machines does not improve the performance in relation to the sequential solution alternative. The real speedup values when the six computers are used are 1.3 for OverMsg(PVM) and 1.08 for SeqMsg(PVM).

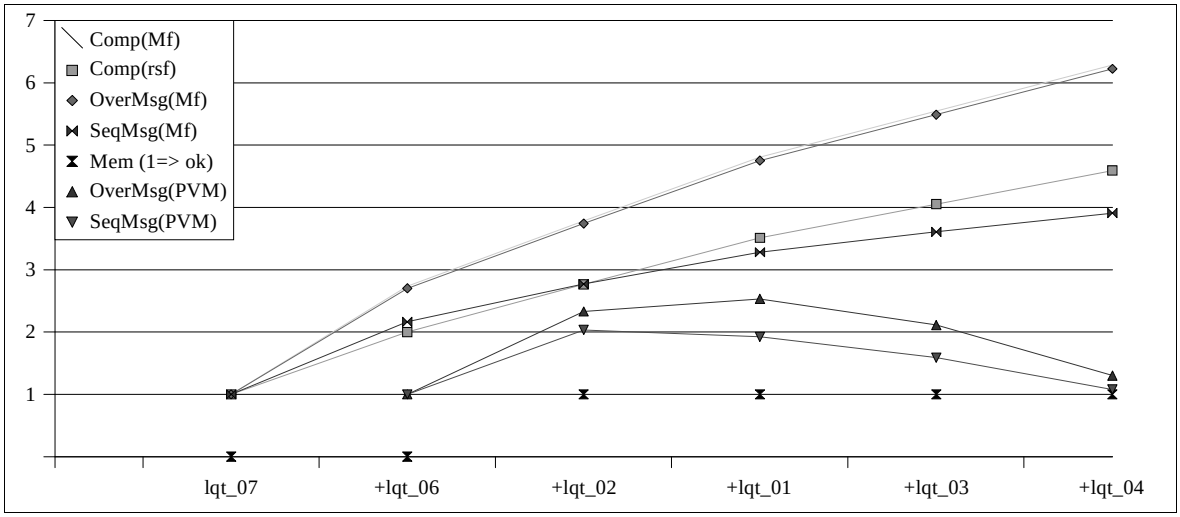


Figure 4.20: Algorithms Speedup with PVM in the LQT Network for n = 9000.

Comparing the results shown in Figure 4.20 with those of Figure 4.19, some differences can also be found:

- Some of the obtained speedup values are rather closer to the estimated, at least for three or four machines, i.e. when **lqt_07**, **lqt_06**, **lqt_02**, and **lqt_07**, **lqt_06**, **lqt_02** and **lqt_01** are used respectively.
- The algorithm that carries out computation overlapped with communications has better performance than that which does not try to make use of any overlay. The difference is slightly more than 30% when **lqt_07**, **lqt_06**, **lqt_02**, **lqt_01** and **lqt_03** are used.

4.6.3 LIDI Local Area Network

Figure 4.21 shows the speedup values obtained in the LIDI local network by the sequential computation and communication algorithm, and those by the overlapped computation with communication algorithm, implemented by using the PVM message-passing library, SeqMsg(PVM) and OverMsg(PVM) respectively, for n = 2000, together with those previously shown in Figure 4.15. Like in the CeTAD and LQT networks in the *corresponding* context (Figure 4.17 and figure 4.19):

- Speedup estimations are rather far-off from the obtained values. From the utilization of four computers the difference is even greater.
- In general terms, the addition of computers implies performance loss; the exceptions are given for two or three computers since performance increases in those cases when more machines are used.

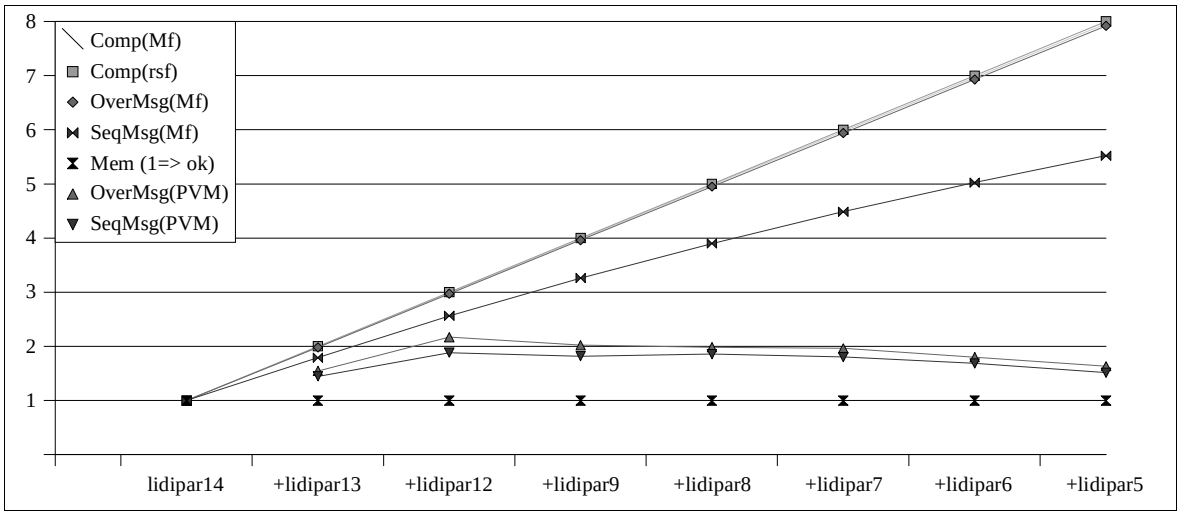


Figure 4.21: PVM Algorithms Speedup in the LIDI Network for n = 2000.

Unlike CeTAD and LQT networks:

- The computing time is not worse than that of the sequential solution, despite the tendency indicates that, as the number of computers used increases, this situation can be reached (with speedup values less than one).
- The performance loss is, as the number of computers used increases, rather more gradual.

Figure 4.22 shows the speedup values obtained in the LIDI local network by the sequential computation and communication algorithm, and those by the computation overlapped with communication algorithm, implemented by means of the PVM message-passing library, SeqMsg(PVM) and OverMsg(PVM) respectively, for n = 3200, together with those previously shown in figure 4.16.

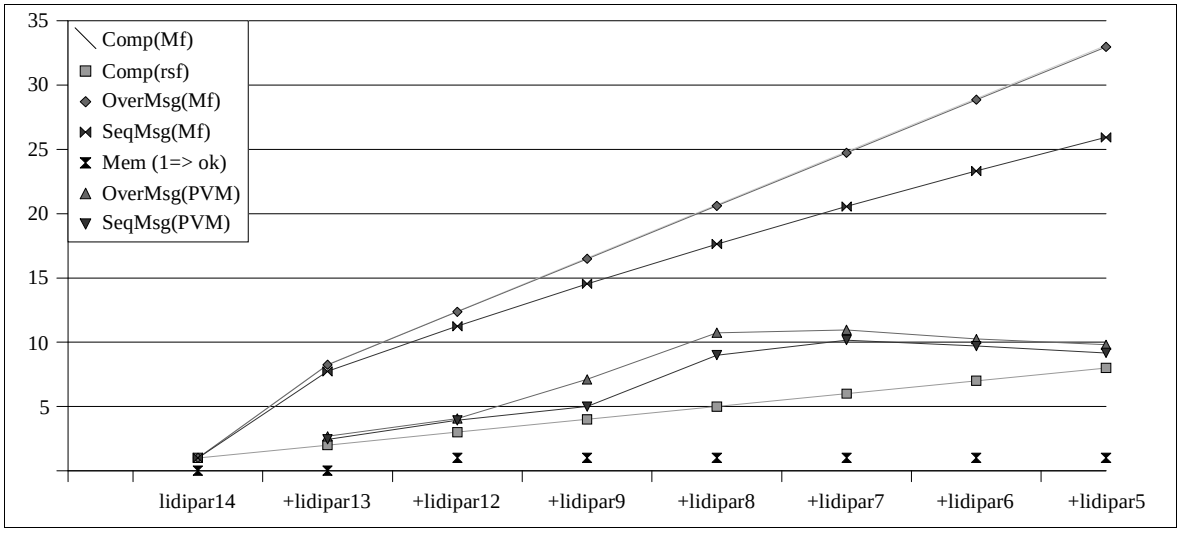


Figure 4.22: PVM Algorithms Speedup in the LIDI Network for n = 3200.

Even though the values obtained from the experimentation are not close to those estimated

for the algorithms, it is the first time that higher speedup values are obtained in comparison with at least the speedup values computed according to the relative speeds. According to Figure 4.22, it can also be said that:

- In the best of the cases, given when **lidipar14**, **lidipar13**, **lidipar12**, **lidipar9** and **lidipar8** are used, the matrix multiplication is solved ten times faster than in **lidipar14** (it must be bear in mind that **lidipar14** uses swap memory in order to solve this problem).
- From the inclusion of **lidipar6,** performance decreases with both algorithms, all of which implies that both OverMsg(PVM) and SeqMsg(PVM) are reduced.
- "Superlinear" speedup values are obtained (greater than the quantity of homogeneous processors), since the reference time of the problem solved sequentially in **lidipar14** is penalized in terms of performance by the use of the swap space during computations, a fact that can be clearly seen in Figure 4.8

The values obtained from the experimentation within the LIDI local network seem to be quite better than those obtained from the experimentation in the CeTAD and LQT local networks. Beyond the particular differences between the machines – from the point of view of the parallel computers that are built up with each local network-, the main differences are:

- The LIDI local network is ten times faster than that of CeTAD and LQT.
- The parallel computer built with the LIDI local network is homogenous, while both CeTAD and LQT parallel computers are (*very*) homogenous.

Intuitively, the most important reason for which better results are obtained in the LIDI local network is the really superior capacity of the interconnection network, even though more data is apparently needed to render a more justified explanation.

4.7 Execution Profiles in Local Aera Networks with PVM

In order to be more precise about parallel running times and the reasons for which speedup estimations are so far off from those obtained, the same experimentations were carried out but with a minimum of instrumentation so that:

- the part of the total running time used for parallel computation and the part used for communications is clearly identified. This information is very useful for identifying whether the problem is communications or not;
- the running state of each process (local network computer) is graphically identified during each running time *instant*. This type of information is more detailed than the previous one, and is useful for identifying whether there exists some particular computer producing a general delay. For instance, if for any local reason a computer does not send a broadcast message in the expected time, the rest of the computers will be affected since they will not receive it.

Since experimentations are so numerous to show each running time profiles, and what's more, they are mostly similar, the present work presents and explains the most significant in each of the local networks.

4.7.1 CeTAD Local Area Network

Figure 4.23 shows the execution profile when using the five best machines of the CeTAD local network for a parallel matrix multiplication of 2000×2000 elements with the sequential computation and communications algorithm, where:

- The time appears in seconds.
- At every time instant, each computer can be:
  - running a partial computation, shown as "Computation" in the graphic;
  - sending or receiving a broadcast message, shown as "Bcasts" in the graphic, and during which computations cannot be carried out and thus, in order to go on with the computation, the finalization of the broadcast must be "waited".
- Bcasts identifies each broadcast message, in which a process run in a computer sends data to the rest (in this case, to another four processes) that are run in the rest of the computers (four computers).
- Since in PVM the delivery of *all* the messages is overlapped with computations, when a broadcast is sent simultaneously, a partial computation period of the result matrix is carried out.

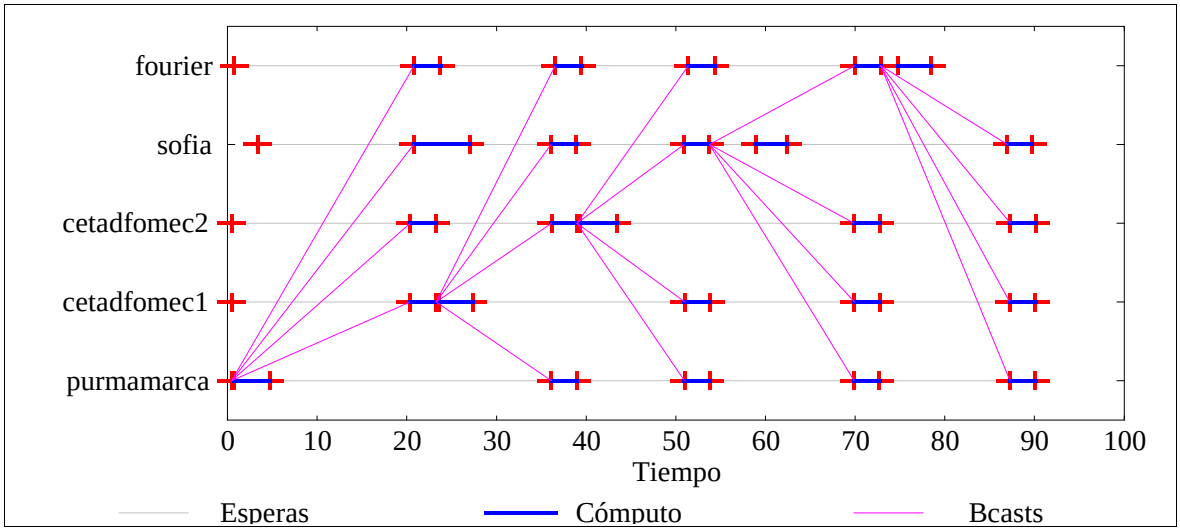


Figure 4.23: SeqMsg(PVM) Profile with Five Machines and *n* = 2000 in CeTAD.

Beyond some particular details of the running time, it can be clearly noticed that during most of each computer's running time, the finalization of a broadcast message is awaited (more precisely, the reception of a broadcast message from another computer).

Table 4.4 shows the summary information of the parallel program running that corresponds to the running profile of Figure 4.23, where:

- ***Name*** identifies the name of the computer used.
- ***Rows*** identifies the quantity of rows of the result matrix assigned to each computer, which is proportional to the relative speed of each computer in relation to the parallel computer.
- ***Tot. Comp.*** identifies the quantity of local running time during which operations with

floating point numbers have been executed.

- ***Per It.*** identifies the quantity of local running time of a computation step (operations with floating point numbers).
- ***Tot. Msg.*** identifies the quantity of local running time using the wait of the broadcast message finalization (one delivery and four receptions, since there is a total of five computers).

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
purmamarca	555	15,17	3,03	68,29
cf1	426	15,05	3,01	68,39
cf2	426	15,59	3,12	67,80
sofia	394	17,00	3,40	65,22
fourier	199	15,25	3,05	55,91

Table 4.4 : Summary of SeqMsg(PVM) with Five Machines and n = 2000 in CeTAD.

Figure 4.24 shows the running profile when all the machines (ten) available in CeTAD are used. In addition, and from what the graphic shows, it is clear that most of the running time is spent in messages.

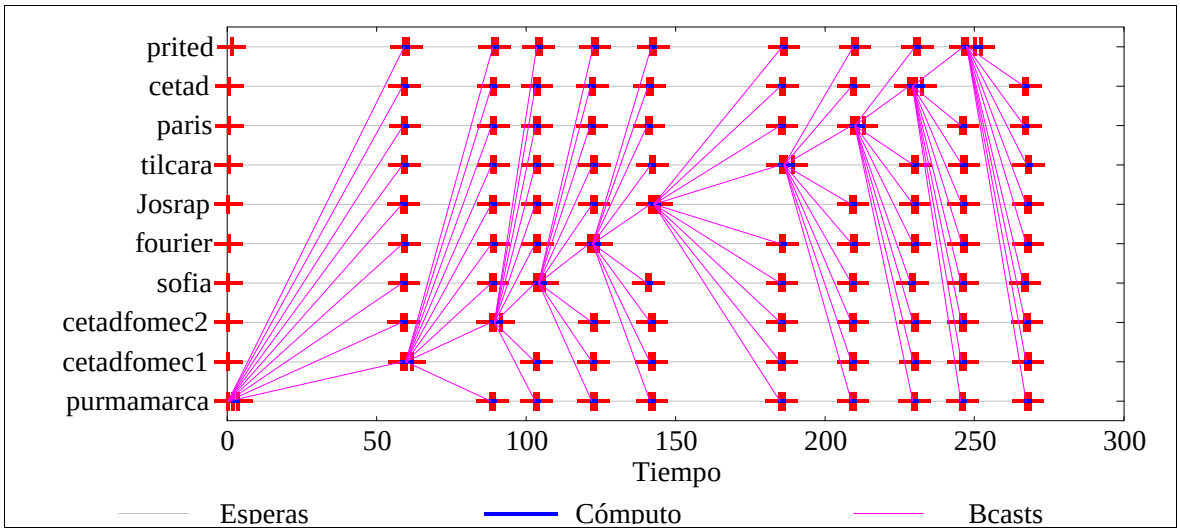


Figure 4.24: SeqMsg(PVM) Profile with Ten Machines and n = 2000 in CeTAD.

Figure 4.24 also shows quite clearly that both the first broadcast - sent from **purmamarca** - and the sixth - sent from **Josrap** - use more transmission time than the rest. But even if these two messages use the average communication time used by the rest, the total communication time is still dominated by communications. In consequence, the first problem to be solved according to these two running profiles just described (Figure 4.23 and Figure 4.24) is the excessive communication time.

Table 4.5 shows the summary information of the parallel program running that corresponds

to the running profile of Figure 4.24, where the relative weight of communications can be quantified more clearly in relation to the computation (***Tot. Comp.***)

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
purmamarca	454	12,35	1,24	255,78
cf1	349	12,44	1,24	255,75
cf2	349	12,48	1,25	255,31
sofia	324	12,21	1,22	255,05
fourier	164	12,85	1,28	255,22
Josrap	142	12,24	1,22	256,03
tilcara	104	13,29	1,33	254,94
paris	48	12,08	1,21	255,01
cetad	38	12,84	1,28	254,16
prited	28	13,68	1,37	236,92

Table 4.5: Summary of SeqMsg(PVM) with Ten Machines and n = 2000 in CeTAD.

The situation does not vary much when the communications algorithm overlapped with computation is used, as Figure 4.25 shows.

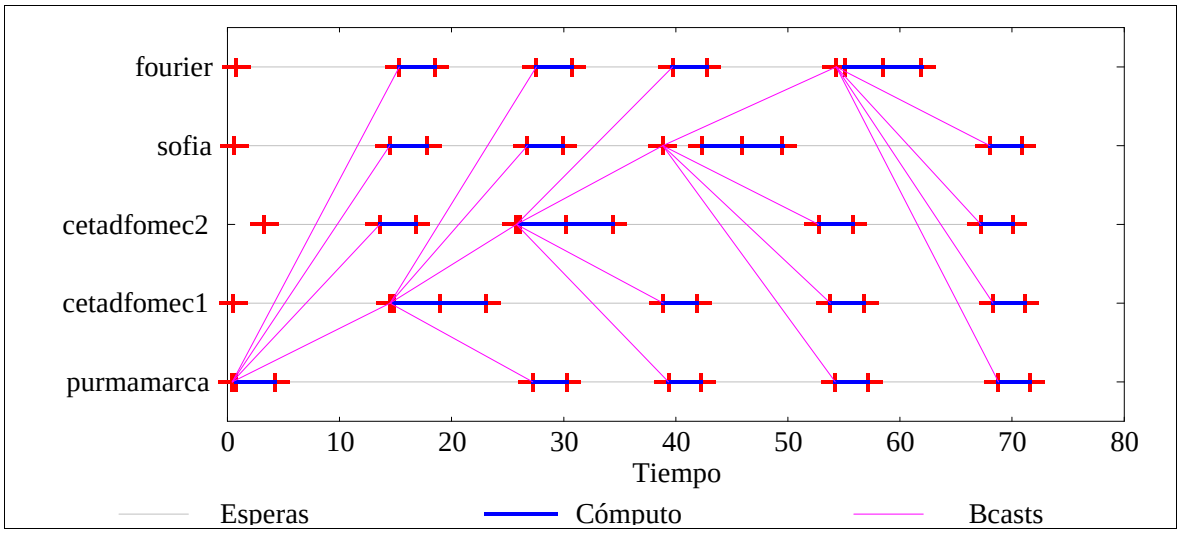


Figure 4.25: OverMsg(PVM) Profile with Five Machines and n = 2000 in CeTAD.

In brief, Figure 4.25 shows the running profile when using the CeTAD local network's five best machines for a parallel matrix multiplication of 2000×2000 elements with the overlapping computation and communications algorithm. Also in this case, during most of the running time of each computer, the finalization of a broadcast message (more precisely, the reception of a broadcast message from another computer) is expected. The total running time of OverMsg(PVM) is less than that of SeqMsg(PVM), due to the overlapping that partially masks the weight of the communication time. The running summary shown in

Table 4.6 is similar to that shown in Table 4.4, even though the communication times are inferior, since part of the time of each broadcast message is *overlapped* with local computation.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
purmamarca	555	15,20	3,04	55,97
cf1	426	17,07	3,41	53,54
cf2	426	17,31	3,46	49,44
sofia	394	16,50	3,30	53,78
fourier	199	16,18	3,24	44,97

Table 4.6: Summary of OverMsg(PVM) with Five Machines and n = 2000 in CeTAD.

When the maximum size that can be solved in the computer with highest computing capacity within CeTAD –i.e. n = 3200 - is taken as reference, the characteristics in terms of running and performance profiles are still the same. Figure 4.26 shows the execution profile when using the seven best machines of the CeTAD local network for a parallel matrix multiplication of 3200× 3200, with the overlapping computation and communications algorithm.

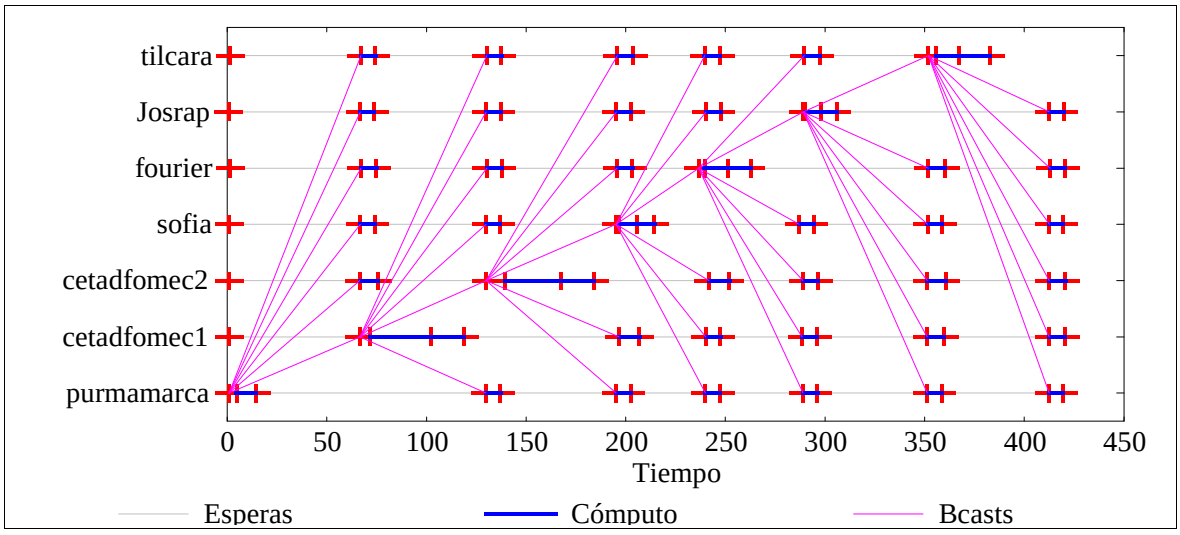


Figure 4.26: OverMsg(PVM) Profile with Seven Machines and n = 3200 in CeTAD.

Table 4.7 completes the information of Figure 4.26 with the summary of the execution, showing, in each computer:

- the quantities of assigned rows (***Rows***).
- the total computing and communication times (***Tot. Comp***. and ***Msg. Tot***.).
- the time dedicated to local computation in each iteration (***Per. It.***).

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
purmamarca	771	53,36	7,62	365,40
cf1	593	88,55	12,65	330,97
cf2	593	89,01	12,72	330,66
sofia	549	53,26	7,61	365,44
fourier	276	62,05	8,86	356,99
Josrap	242	51,96	7,42	367,13
tilcara	176	65,33	9,33	316,02

Table 4.7: Summary of OverMsg(PVM) with Seven Machines and *n* = 3200 in CeTAD.

Figure 4.27 and Table 4.8 show all that is related to a 3200×3200-element matrix multiplication in parallel, with the overlapped computation and communications using the ten machines available in CeTAD.

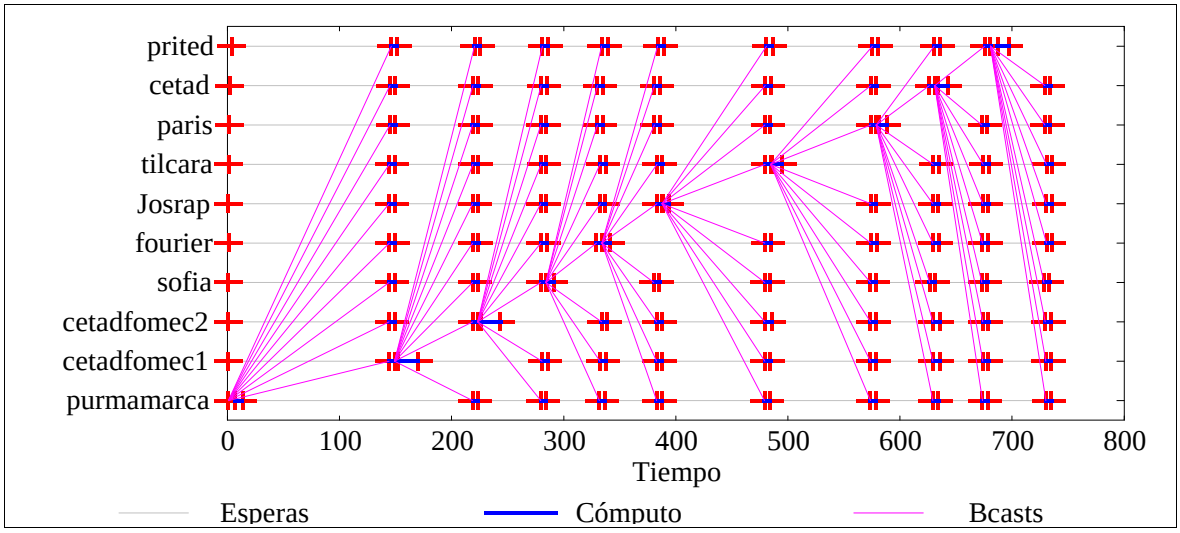


Figure 4.27: OverMsg(PVM) Profile with Ten Machines and *n* = 3200 in CeTAD.

It is really interesting to compare the details in terms of computing time (***Tot.Comp.***) and communication time (***Msg. Tot***) shown in Table 4.7 and Table 4.8. The total time computing average in each computer is of approximately 66.22 seconds when the seven computers with highest computing capacity of the CeTAD are used. When all the computers are used, this average is of approximately 53.5 seconds. In principle, it should not happen the same with communications because, in all the cases, all the data of matrix B must be transferred among the computers (via broadcast messages) and, thus, the communication time should be kept more or less invariant. This is based on the assumption that each broadcast message implementation among processes takes the utmost advantage of the Ethernet network broadcast capacity, in which there should be at least a minimum increase of time due to the highest quantity of receiving processes of each of the messages

to be carried out. However, the total communication time average when the seven best computers of CeTAD are used is of 347.52 seconds, and when all the computers are used is of 676.14 seconds, i.e. almost twice the time necessary to transfer the same data.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
purmamarca	726	49,16	4,92	685,10
cf1	559	63,12	6,31	671,01
cf2	559	63,72	6,37	670,37
sofia	518	48,72	4,87	683,82
fourier	261	50,20	5,02	684,24
Josrap	228	48,08	4,81	686,53
tilcara	166	52,02	5,20	682,36
paris	77	50,20	5,02	682,36
cetad	61	52,03	5,20	680,20
prited	45	57,78	5,78	635,42

Table 4.8: Summary of OverMsg(PVM) with Ten Machines and n = 3200 in CeTAD.

Since the performance problem is given by communications, it is convenient to keep on analyzing in more detail the communication times, and in this sense, the summaries of the parallel runnings can offer more information. As previously mentioned, for a given size of matrices, the quantity of data to be communicated between computers is the same and independent of the quantity of computers used in parallel. From the point of view of the messages, the data of matrix B (C=A×B) should always be transferred among all computers.

According to Table 4.4, each computer spends an average of 65.12 seconds to communicate data of matrix B with the non-overlapped computation and communication algorithm in five computers. According Table 4.5, when the ten computers are used (the same algorithm and the same matrix sizes), the average time of is 253.42 seconds. Even though the absolute values are completely different, the general situation does not seem to change much with the algorithm designed to overlap communications with local computation and for matrices of order n = 3200. According to Table 4.7, the communication average time is of 347.52 seconds when seven computers are used, and according to Table 4.8, the total communications time is of 676.14 seconds when ten computers are used.

Table 4.9 shows a summary of what happens with communications in terms of performance given in MB/s (2^{20} bytes per second) with the data of the previously mentioned tables, where

- ***n*** is the order of the matrices to be multiplied (an the order of matrix B transferred among machines)
- ***Comp. Number*** is the number of computers used in parallel and in which the code to obtain communication times, among many, was instrumented.

- ***Algorithm*** indicates the parallel algorithm used.
- ***Time*** is the average local time used for communications.
- ***MB/s*** indicates the performance of the interconnection network in Megabytes (220 bytes) per second, computed according to matrix B's transference time (the only to be transferred among computers).

n	*Comp. Number*	*Algorithm*	*Time*	*MB/s*
2000	5	SeqMsg	65.12	0.23
2000	10	SeqMsg	253.42	0.06
3200	7	OverMsg	347.52	0.04
3200	10	OverMsg	676.14	0.02

Table 4.9: Communications Performance in CeTAD.

The two most important conclusions as regards communications' performance using the data of Table 4.9 are that

1. In general, it is really low, since, in the best of the cases, less than the 25% of the maximum theoretical capacity of the interconnection network is used.
2. For a given data quantity to be transferred, the higher the number of computers used the lower the performance.

4.7.2 LQT Local Area Network

Both each algorithm running profiles and the summaries of the computing and messages running times are really similar to those shown in the CeTAD. It is clear that they do not coincide in absolute terms due to the differences as regards computers and matrix sizes used to carry out the processing. What is significantly similar is the "behavior" in terms of performance of the various parallel machines, and their consequent conclusion in terms of the problem to be solved: that of communications.

Figure 4.28 shows the running profiles that correspond to the best parallel time used to solve a 5000x5000 element matrix multiplication. This is obtained with the overlapped computation and communication algorithm using three computers. It is interesting to notice two relevant aspects from the running profile of Figure 4.28:

- Communications overlay with local computation is used quite effectively. More specifically, computers that receive broadcast messages sent from **lqt_06** and **lqt_02** wait for these data a little more than what is expected.
- Almost all the communication time affecting the total running time is that of the first broadcast, for which machines have to wait in the first place, i.e. that sent from **lqt_07** and received in **lqt_06** and **lqt_02**.

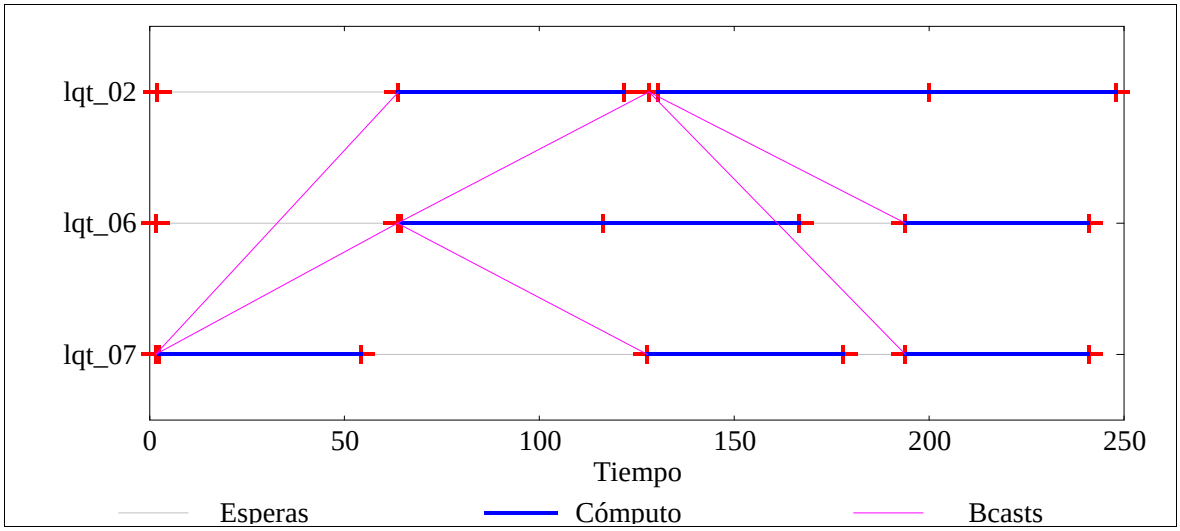


Figure 4.28: OverMsg(PVM) Profile with Three Machines and *n* = 5000 in LQT.

Table 4.10 shows the summary of the parallel running corresponding to the profile of Figure 4.28.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
lqt_07	1808	148,87	49,62	90,60
lqt_06	1807	149,06	49,69	90,34
lqt_02	1385	175,20	58,40	70,66

Table 4.10: OverMsg(PVM) Summary with Three Machines and *n* = 5000 in LQT.

As already explained with the speedup values obtained by the algorithms, performance gets worse when more machines of the LQT local network are used. This can be clearly seen in Figure 4.29, which shows the running profile of a 5000x5000 element matrix multiplication using the overlapped communications and computing algorithm and all the available computers of LQT.

Figure 4.29 evidently shows that there is a computer in particular that behaves differently from the rest, or at least that the broadcast message sent from **lqt_01** uses a transmission time significantly higher than the rest.

Once more, as when the same situation was identified in the CeTAD local network (Figure 4.21), it must be said that, even if this broadcast used the average communication time used by the rest, the total communication time would still be dominated by the sum of the times used for communications.

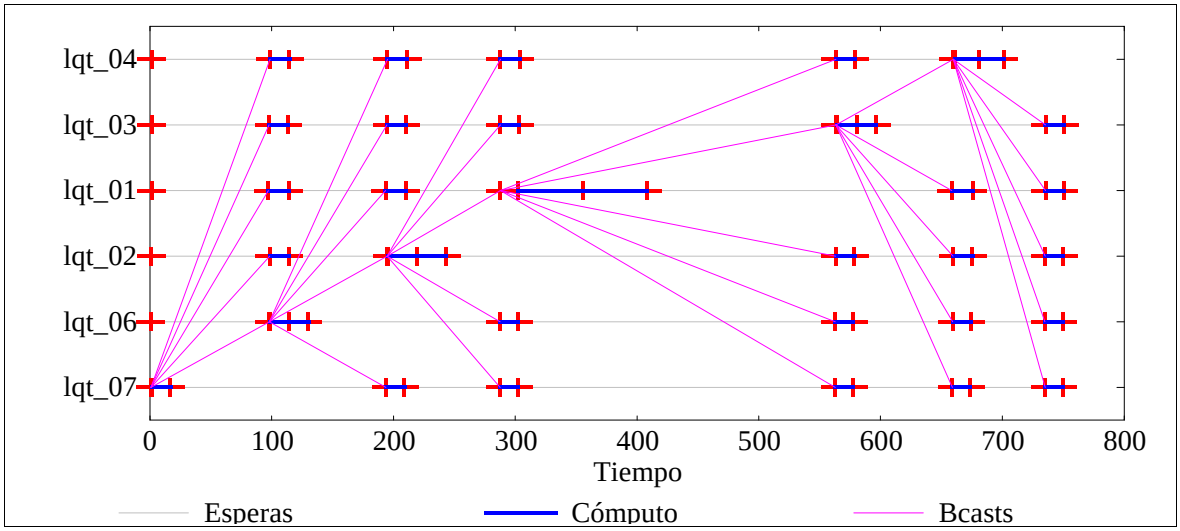


Figure 4.29: OverMsg(PVM) Profile with Six Computers and n = 5000 en el LQT.

Table 4.11 shows the summary of the 5000x5000 elements matrix multiplication running using the overlapped communications and computing algorithm and all the computers available in LQT in *relation* to Figure 4.29. As previously identified with the computers of CeTAD, the communication time is what changes significantly in terms of performance. This behavior is verified in LQT, comparing the column that shows the times used for communications in each machine (***Tot. Msg.***) of Table 4.10 and that of Table 4.11. When more machines are used, the transmission of the same quantity of data (B matrix's elements) among computers takes much more time.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
lqt_07	1089	89,21	14,87	657,71
lqt_06	1089	89,15	14,86	657,57
lqt_02	835	109,13	18,19	637,87
lqt_01	811	174,52	29,09	572,56
lqt_03	589	92,79	15,47	654,50
lqt_04	587	102,52	17,09	592,84

Table 4.11: Summary of OverMsg(PVM) with Six Machines and n = 5000 in LQT.

In the case of the matrices of order n = 9000, the situation in terms of performance is similar. Figure 4.30hows the running profile corresponding to the best parallel time used to solve a 9000x9000 element matrix multiplication. This time is obtained with the overlapped computing and communication algorithm using the four computers with highest computing capacity.

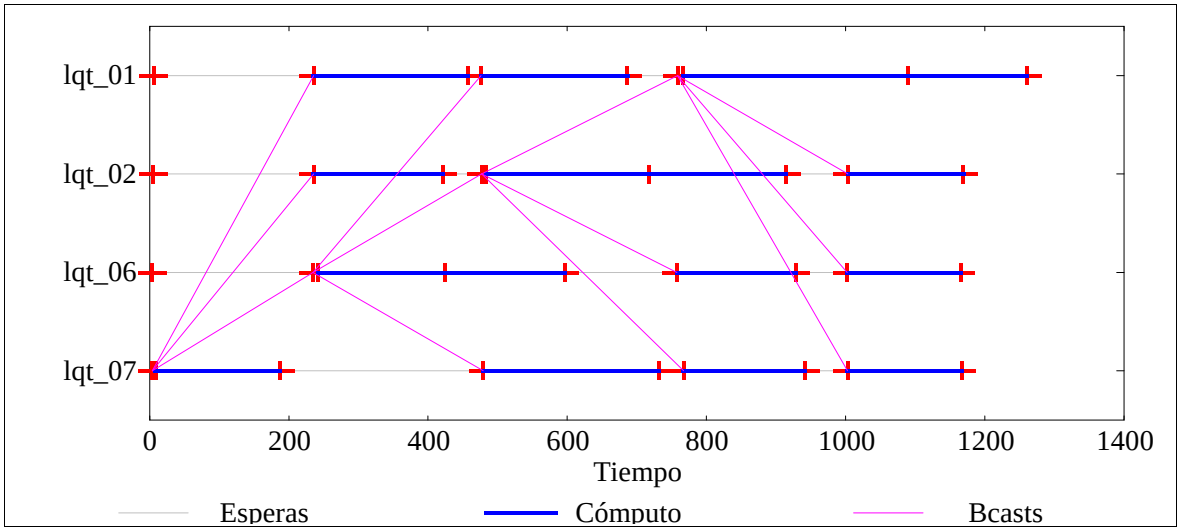


Figure 4.30: Profile of OverMsg(PVM) with Four Computers and *n* = 9000 in LQT.

Once more, as more machines are used to solve the same problem (a matrix multiplication of 9000x9000 elements), with the same algorithm (that solves overlapped communications with local computation), the performance worsens and the total running time is greater. Figure 4.31 shows the running profile to solve a 9000x9000 element matrix multiplication in parallel using all the available computers of LQT and the overlapped computing and communication algorithm. In this case, the summaries of the execution are not shown, even though they are nothing but the confirmation of the apparent conclusions made from Figure 4.29 and Figure 4.30: the loss of time is due to the excessive communication time when more computers are used.

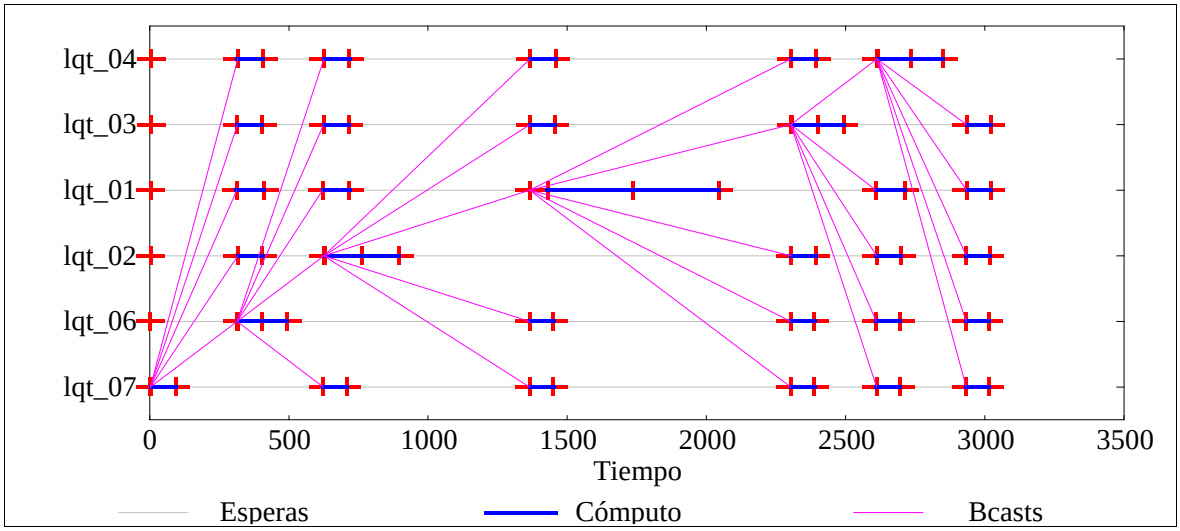


Figure 4.31: Profile of OverMsg(PVM) with Six Computers and *n* = 9000 in LQT.

Specifically as regards communications performance, when analyzing the running time

values for matrices of order n = 5000 (shown in Table 4.10 and Table 4.11), and of order n=9000 (not specifically shown), it occurs the same as in the CeTAD local network:

1. It is generally very low.
2. For a data quantity to be transferred, the higher the number of computers, the lower the performance. The time necessary to carry out a broadcast increases linearly with the number of computers involved.

4.7.3 LIDI Local Area Network

In the LIDI Local Network the previous results are confirmed, though with a difference given by the best performance of the interconnection network – ten times better than that of the CeTAD and LQT local networks. In this sense, the running profiles (and the summaries of computing and communication times) show that:

- Unlike the CeTAD and LQT local networks, the communication time has not such an important weight in the total time of the parallel programs.
- Like in the CeTAD and LQT local networks, the communication time increases notably as more computers are used to solve a same problem.

As an example, Figure 4.32 shows the execution profile of the parallel program with sequential computation and communications using four computers to solve a multiplication of matrices of order n = 2000.

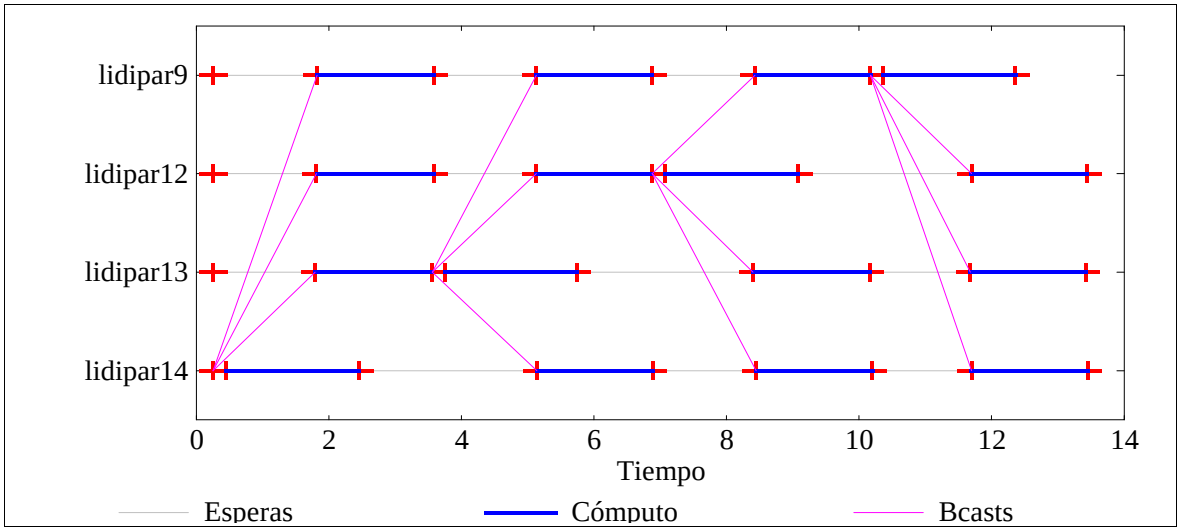


Figure 4.32: Profile of SeqMsg(PVM) with Four Computers and n = 2000 in LIDI.

In addition, Table 4.12 shows the summary of the parallel program running with sequential computation and communications using four computers to solve a multiplication of matrices of order n = 2000.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
lidipar14	500	7.25	1.81	6.30
lidipar13	500	7.29	1.82	6.16
lidipar12	500	7.26	1.81	6.31
lidipar9	500	7.24	1.81	5.24

Table 4.12: Summary of SeqMsg(PVM) with Four Machines and n = 2000 in LIDI.

Figure 4.33 and Table 4.13 show the profile and the summary of the parallel program running with sequential computation and communications using all the computers to solve a multiplication of matrices of order n = 2000.

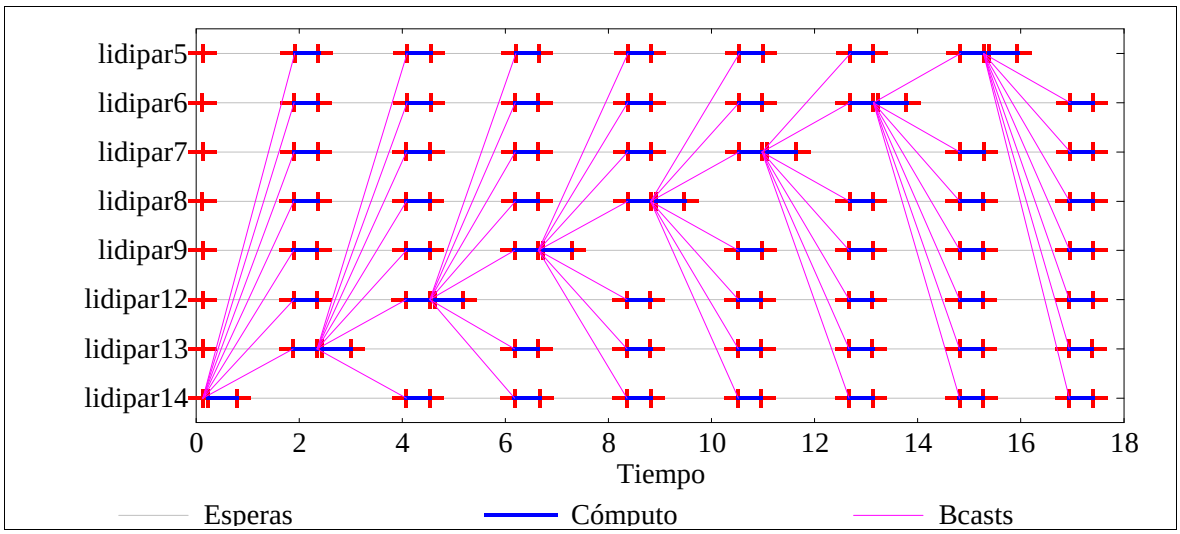


Figure 4.33: Profile of SeqMsg(PVM) with Eight Computers and n = 2000 in LIDI.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
lidipar14	250	3.78	0.47	13.49
lidipar13	250	3.75	0.47	13.52
lidipar12	250	3.73	0.47	13.54
lidipar9	250	3.74	0.47	13.54
lidipar8	250	3.73	0.47	13.55
lidipar7	250	3.74	0.47	13.54
lidipar6	250	3.72	0.46	13.56
lidipar5	250	3.73	0.47	12.08

Table 4.13: Summary of SeqMsg(PVM) with Eight Machines and n = 2000 in LIDI.

From the values shown in Table 4.12, it is clear that each computer uses slightly more time in local computation than in communications. The situation changes when eight computers are used, such as Table 4.13 shows, since each computer uses for communications an average more than three times the time used for local computation.

It is interesting to notice what happens when a multiplication of matrices of order n= 3200 is solved. Figure 4.34 and Figure 4.35 show the execution profiles of the parallel program with sequential computation and communications that solve a matrix multiplication of order n= 3200 using four and eight computers, respectively. It is evident that the running time with four computers is rather greater than the running time with eight computers, though this does not assure on its own the performance acceptability.

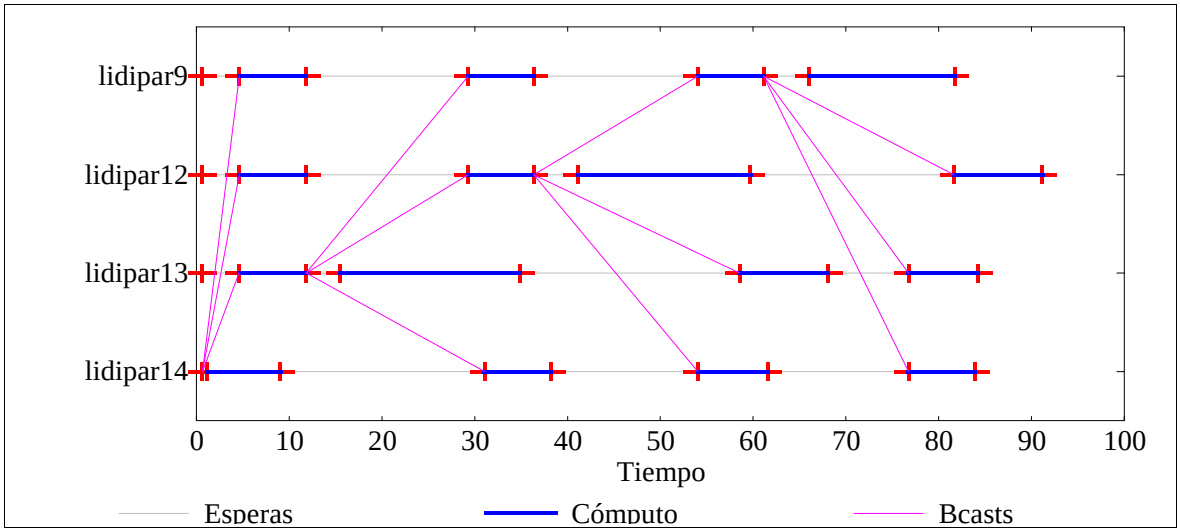


Figure 4.34: Profile of SeqMsg(PVM) with Four Computers and n = 3200 in LIDI.

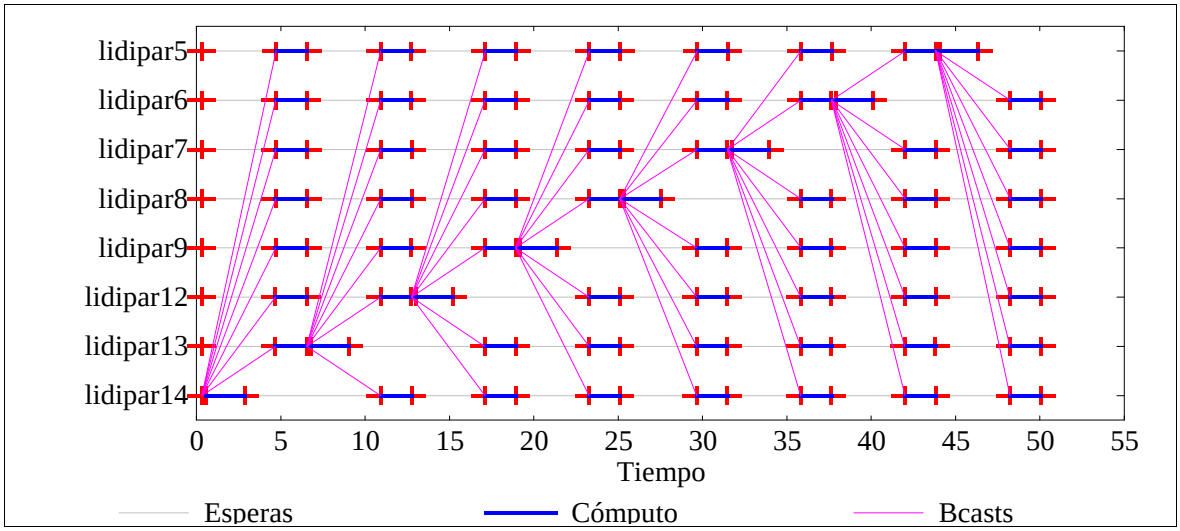


Figure 4.35: Profile of SeqMsg(PVM) with Eight Computers and n = 3200 in LIDI.

Table 4.14 shows the summary of the running times corresponding with the profile of Figure 4.34, and Table 4.15 shows the summary of the running times corresponding to the profile of Figure 4.35.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
lidipar14	800	30,20	7,55	55,40
lidipar13	800	43,00	10,75	42,79
lidipar12	800	43,24	10,81	49,37
lidipar9	800	38,20	9,55	44,69

Table 4.14: Summary of SeqMsg(PVM) with Four Machines and n = 3200 in LIDI.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
lidipar14	400	14,97	1,87	35,13
lidipar13	400	14,98	1,87	35,12
lidipar12	400	14,97	1,87	35,13
lidipar9	400	14,94	1,87	35,17
lidipar8	400	14,95	1,87	35,16
lidipar7	400	14,98	1,87	35,14
lidipar6	400	14,89	1,86	35,22
lidipar5	400	15,01	1,88	31,47

Table 4.15: Summary of SeqMsg(PVM) with Eight Machines and n = 3200 in LIDI.

In addition, as shown in Figure 4.34, there are computing periods that are substantially greater than others. More specifically, in computers **lidipar13**, **lidipar12**, and **lidipar9**, the computing period after the broadcast message's delivery is greater than all the remaining local computing periods. Due to the very definition of the algorithm, in all the computing periods the same task is carried out, and thus the computing time should be the same.

One of the main reasons for a computer to have a lesser performance (assuming that the same precise task is carried out, as in this case) is the use of the previously mentioned swap memory. Consequently, the most *acceptable* explanation is entirely related to the memory. Since PVM is used in order to carry out a broadcast message, each communication routine has its own overhead (aggregate memory requirement), basically for the storage of messages in *buffers* (intermediate memory), which PVM, in turn, transfers among machines. In general, it can be accepted that the quantity of extra memory in this sense is, at the very least, equal to the data quantity transferred. This memory overhead reduces the available main memory and, thus, the swap memory space is used. In fact, this creates a reduction of the performance of

- Computation, since part of the data to be processed might be assigned in swap memory.

- Communications, since part of the data to be transferred might be assigned in swap memory.

And, for this reason, the computing periods posterior to a broadcast are “slower” than the rest.

Unlike what has happened in CeTAD and LQT Local Networks, the total average communication time is quite higher for four machines than for eight, all of which can be proved with the data of Table 4.14 and Table 4.15, respectively.

The execution profile and summary for the same problem but with six machines (Figure 4.36 and Table 4.16) demonstrate that the performance problem is related to that of memory because

- All the computing periods in all the computers use approximately the same running time.
- The total communication time is lesser than for four and eight computers.

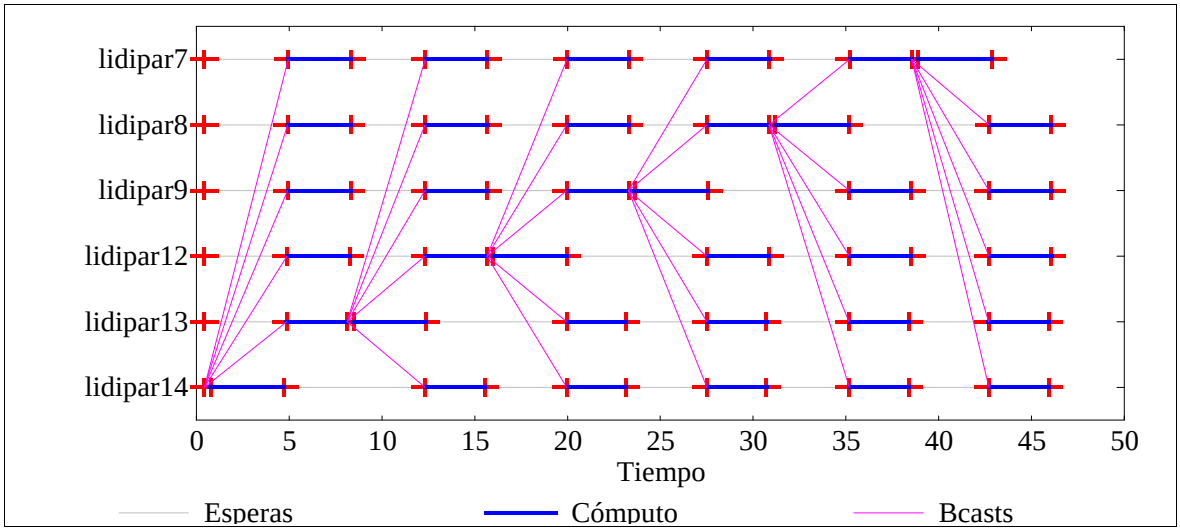


Figure 4.36: Profile of SeqMsg(PVM) with Six Computers and n = 3200 in LIDI.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
lidipar14	534	19,99	3,33	25,53
lidipar13	534	19,98	3,33	25,53
lidipar12	533	20,75	3,46	24,89
lidipar9	533	20,73	3,46	24,91
lidipar8	533	20,75	3,46	24,90
lidipar7	533	20,77	3,46	21,68

Table 4.16: Summary of SeqMsg(PVM) with Six Machines and n = 3200 in LIDI.

4.8 Real Performance of Local Networks using "UDP"

Since the performance problem is created almost exclusively by communications, specific tests were carried out in order to evaluate the performance of broadcast and point-to-point messages using the PVM library. Though the results appear in detail in Appendix C, the main conclusions are:

- The ways of sending a same message to more than one target process when each process is assigned to a different machine are implemented with multiple point-to-point messages. Both
  - the multicast operation , **pvm_mcast()**, and
  - the broadcast operation in a group, **pvm_bcast()**,

  imply that, at least, the same message is sent *m* times from the origin computer (where the process sending the message is running) towards the *m* machines where there is at least one target process of the message. If, for instance, a broadcast or multicast message has five receptors and each of these receptor processes is being run in a different machine (and different from the machine in which the process that sends the message is being run), the total time of the message will be approximately five times the time of the same message if it is sent to another process run in another machine.
- Messages latency time depends on their origin and target computers. However, for larger messages, the latency time is not relevant in relation to the data transference time and, thus, the total message time is independent of the machines that communicate among themselves.

The proposed broadcast communication routine is oriented to the utilization of the physical characteristics (specifically broadcast) of the Ethernet networks which the communication library (PVM) does not use. The direct consequence is that the broadcast communication time with PVM will be much higher than the expected and, thus, the results are not so good in terms of total performance. In this sense, there exist several alternatives, being the following the two most important:

1. To use another message passing library, such as some MPI implementation (usually suggested for this type of parallel architectures).
2. To implement a broadcast message routine (and, eventually, a whole collective communication library) to explicitly use the Ethernet networks' broadcast capacity.

A priori, the use of another message passing library has a fundamental drawback from the point of view of the performance, or from that of "prediction" of broadcast messages' good performance. In the specific case of MPI, it is clear that the performance is independent of the implementation. More specifically, the implementation would be that which determines the degree of utilization of the Ethernet networks' characteristics for broadcast messages. In this sense, MPI and particularly all its implementations share *a certain degree of uncertainty* in relation to the broadcast message performance with the rest of the message passing libraries, including PVM. In this case, the difference are the specific experimentations carried out, which have determined the broadcast (and multicast) message performance characteristics for PVM, and not for the remaining libraries. In fact, it is quite difficult for message passing libraries to be to optimized with the characteristics of the Ethernet networks since:

- In general, libraries are proposed, one way or the other, as standards for message passing parallel machines and, thus, there is no sense in orienting them towards a specific type of interconnection networks. In fact, both PVM and MPI have been implemented for different types of parallel machines and, thus, there is no sense in orienting them *a priori* to Ethernet interconnection networks.
- In general, libraries provide a large quantity of communication routines. Even though we can assert that for point-to-point process communications the primitive **send – receive** are theoretically enough, it is also true that there exists a great variety of communication routines that are considered useful and even necessary in some cases. Perhaps, the most clear example in this respect is the very definition of the MPI standard. In this context, it is very difficult to orient or optimize one or one type of communication routine for one or one type of interconnection network without producing a library excessively costly (in terms of development, maintenance, etc.) and/or *too* specific.

For these reasons, a broadcast message routine among user processes was chosen to be implemented with a set of design and implementation premises, so that:

- It makes use of the *very* broadcast of Ethernet networks, and in this way can be optimized in terms of performance. Since the algorithm depends exclusively on broadcast messages, making use of the Ethernet networks' broadcast automatically creates a good expectation in terms of scalability because the communication time is expected to be kept and not increased proportionally to the number of computers used.
- It is simple enough so as to not impose a too heavy load in terms of implementation and maintenance. In addition, it is clear that the simplicity *per se* largely contributes to the optimal performance. On the other hand, the proposal is specific enough to make the implementation simple.
- With the maximum possible portability, it could be used –if possible- even in the context of interconnection networks that are not Ethernet.
- It can be implemented and installed from the user mode, without changing the operating system (*kernel*) and without the necessity to obtain special licenses (*superuser*). It is common to obtain the best results in terms of performance adapting the kernel and/or with the possibility of handling the processes priorities, such as in [31] [25] [GAMMA]. These possibilities are discarded since:
    - In general, free-use libraries do not employ these characteristics and, thus, it would be like changing the parallel software development context. Basically, a user that has always used PVM has never had, nor has, any reason for obtaining special priorities nor even changing the very operating system.
    - The original proposal is directed to installed computers networks and, thus, each computer does not necessarily consider parallel computing as single and/or main objective. In fact, different administrators can be used in parallel for each of the computers and this produces, at least, a multiple administration work that in general is not easy to solve.
- It could eventually extended to a whole collective communications library, such as those proposed in [15] [14] [16], though oriented specifically to Ethernet interconnected networks.

Most (if not *all*) the previous premises are fulfilled when the complete broadcast routine design and implementation is based on the UDP standard protocol (User Datagram

Protocol) [95] over IP (Internet Protocol) [96] since:

- UDP allows to send a same datum or set (package) of data to multiple targets at a user's applications level.
- Such as proved in all the machines used, the implementation of the UDP protocol takes direct advantage of the Ethernet networks broadcast capacity.
- In principle, it seems reasonable that the broadcast directly implemented as a part of the UDP protocol has a better performance than that implemented by a user. If, for instance, there exists the possibility of using UDP in an ATM network, it is very likely that UDP broadcast will be *better* (in terms of performance) than the potentially implemented by user processes. Even though the performance is not taken into account, whenever there exists a UDP protocol implementation, the proposed broadcast will be capable of being employed, independently of whether the interconnection network is Ethernet or not [93].
- The user's interface provided by the sockets is simple enough and highly extended to all the versions of UNIX, so as to simplify the broadcast routine implementation, even when problems related to process *synchronization* (in a same or different computers) and communications *reliability* are to be solved.
- UDP, TCP and IP protocols are easily usable from user processes, at least in the standard computers of the installed local networks.

To summarize, there exists a *new* broadcast message routine based on UDP and portable to at least all the UNIX versions used in all the local networks over which the experimentation is carried out. With this new broadcast message routine the same experiments are carried out again and the results appear in the next subsections.

The results of the experimentation show several characteristics not previously found. The appearance of most of these characteristics is given by:

- The markedly superior performance of the broadcast based on UDP in relation to that of the PVM library. This makes the communication time comparable (at least in the same order of magnitude) to that of computation, and thus each computer's computing performance is important given its weight in the total running time. Up to the present moment, the communication time has been so high that all or most of the parallel running time is basically given by communications.
- Computers' heterogeneity, which in itself contributes with an important innovation degree to what is generally shown with respect to parallel machines performance.

Since now heterogeneity -and, more specifically, the differences in computers' computing performance - becomes relevant, most of the new characteristics appear in the CeTAD local network because it is the most heterogeneous of the three over which the experimentation has been out.

In general, some of the characteristics that appear in the CeTAD local network can also be found in the LQT and LIDI local networks or they just do not appear. At the most, the influence of the computing requirements is enhanced in the case of the LQT local network, when the problems to be solved are (or could be) bigger due to the higher quantity of available memory. For this reason, the experimentation results in the CeTAD local network will be explained with the highest, possible degree of detail; and, in the case of the other two local networks, the differences in terms of "behavior" will only be dealt with always from the performance point of view.

4.8.1 CeTAD Local Area Network

With the aim of presenting and discriminating even better the results of the experimentation in the CeTAD local network, the whole analysis of the data obtained is divided in two subsections that follow the size of the problem. The first section is dedicated to the speedup results taking as reference the running time of the matrix multiplication of order n = 2000. Then, the speedup results taking as reference the running time of the matrix multiplication of order n = 3200 are subsequently analyzed.

4.8.1.1 Matrices of 2000×2000 Elements

Figure 4.37 shows the speedup values obtained in the CeTAD local network by the algorithms implemented using UDP, SeqMsg(UDP) and OverMsg(UDP) for matrices of order n = 2000 together with those previously shown in Figure 4.17.

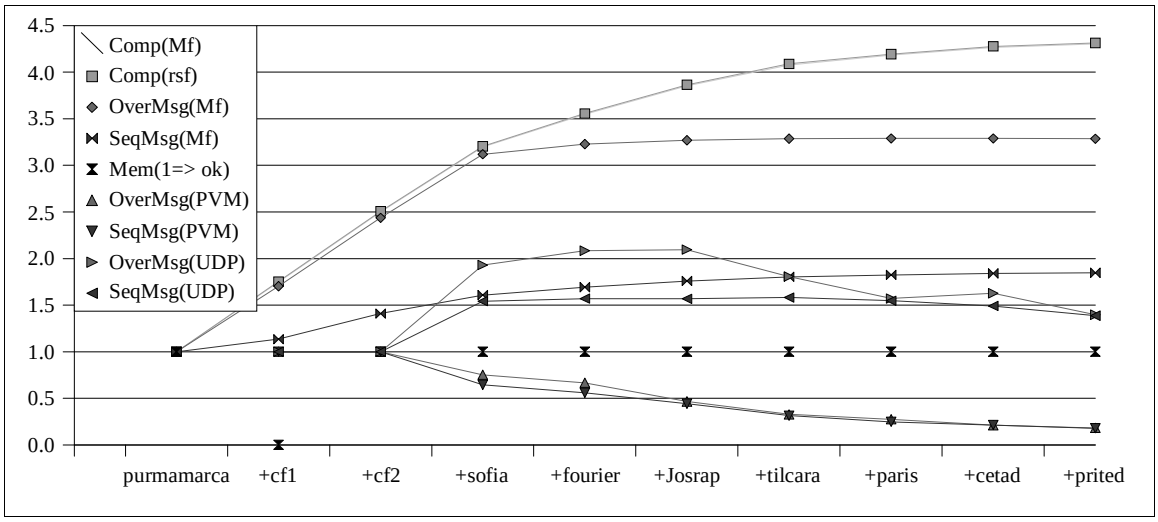


Figure 4.37: Algorithms Speedup with UDP in the CeTAD Local Network for n = 2000.

As it can be seen in Figure 4.37, the performance of the algorithms SeqMsg(UDP) and OverMsg(UDP) –the only added in relation to those shown in Figure 4.17– varies depending on the number of computers used. This variation of the algorithms does not seem, *a priori*, to be related. Initially, the algorithm's results with the sequential computation and communication periods, SeqMsg(UDP), will be analyzed in detail from three points of view:

- The relation of the obtained performance results with those obtained using the PVM communication library.
- The algorithm's capability of using to the utmost the machines computing capacity, in the computing periods.
- The algorithm's capability of using to the utmost the communication network capacity, in the communication period.

Then, the same kind of analysis of the result (in general, always from the performance point of view) will be carried out with the algorithm oriented to the use of the capacity of overlapping computation with communications, OverMsg(UDP).

SeqMsg(UDP). The algorithm that carries out the computing and communication periods sequentially, SeqMsg(UDP), is not far off the estimation of the corresponding maximum speedup, SeqMsg(Mf), at least until the ten available machines are used.

Table 4.17 shows the running summary when six machines are used to multiply 2000x2000 element matrices with this algorithm. Comparing the values appearing in Table 4.17 with those shown in Table 4.4 – in which the only difference is that the broadcast routine provided by PVM is used -, it can be noticed that:

- In terms of local computing time, they are very similar; when PVM is used, the total average is of 15.61 seconds, and with the routine directly based on UDP, the average is of 14.31 seconds.
- Communication times are completely different; when PVM is used, the total average is of 65.12 seconds, and with the routine directly based on UDP, the average is of 15.69 seconds, slightly more than four times lesser.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
purmamarca	555	14,15	2,83	15,89
cf1	426	14,14	2,83	15,86
cf2	426	14,09	2,82	15,91
sofia	394	14,40	2,88	15,71
fourier	199	14,78	2,96	15,08

Table 4.17: Summary of SeqMsg(UDP) with Five Machines and n = 2000 in CeTAD.

The difference in communication times is evidently given by the way PVM solves the broadcast messages: multiple point-to-point messages among the involved computers. But beyond the comparison with the implementation based on PVM, it is necessary to prove if each computer is used to the maximum of its capacity, and if the interconnection network is used with optimum performance. This analysis will be carried out for the case in which ten computers are used, since the performance decreases and the reasons for this performance degradation can be more clearly seen.

Table 4.18 shows the running summary when all the machines (ten) are used to multiply matrices of 2000x2000 elements with the algorithm that carries out sequential computation and communication, SeqMsg(UDP). Comparing the values of Table 4.18 with those of the previous table, it can be noticed that:

- In terms of computing, there are not so many changes since the five added computers have, in fact, really low computing capacity in relation to the total of the remaining. The total processing quantity of the computers is directly proportional to the quantity of assigned rows, which in turn is given by each computer's relative speed, and the first five machines are in charge of the 82% of the total.
- The communication time has increased from 15.69 seconds with five machines to 21.2

seconds with ten machines, representing a penalization in the communications performance of more than 35%. Evidently, a 35% of degradation in the communications performance is much better than the 500% implied when using PVM, though, anyway, it seems rather elevated. Without being too specific with respect to what happens with broadcast messages in performance terms, it can be said that:

- As more processes are involved in collective communications and, in particular, in broadcast messages, a higher penalization in performance terms is expected. When all the processes are run in different computers, and the computers are interconnected with an Ethernet network of 10 Mb/s, the degradation will still be even greater.
- The computers that are added are relatively much slower than the remaining. Even when the transference capacity of the network interface cards is of 10 Mb/s independently of the computers in which they are installed, it has been proved that, at least experimentally, messages latency time depends on the machines computing capacity. As the number of computers increases, the size of messages decreases and, thus, the relative importance of the latency time in each message's total time increases as well.

- And these factors (more machine and higher communication latency times) are combined so as to reach slightly more than 35% of the communication performance penalization.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
purmamarca	454	11,81	1,18	21,63
cf1	349	11,84	1,18	21,58
cf2	349	11,82	1,18	21,60
sofia	324	11,97	1,20	21,48
fourier	164	12,67	1,27	20,72
Josrap	142	12,10	1,21	21,43
tilcara	104	12,08	1,21	21,27
paris	48	11,59	1,16	21,53
cetad	38	11,98	1,20	21,03
prited	28	12,65	1,27	19,75

Table 4.18: Summary of SeqMsg(UDP) with Ten Machines and n = 2000 in CeTAD.

At this point, the performance of *this* interconnection network can be analyzed for broadcast messages when five and ten machines are used. It should be reminded that, in all the cases, the total quantity of data to be delivered is given by the quantity of matrix B's elements (C = A×B). In the case of matrices of order n = 2000 with simple precision floating point numbers, the quantity given in bytes is exactly $4x2000^2$. From the performance point view, then:

- When the five computers with highest computing capacity of the CeTAD are used, the total communication time in each machine is, in average, 15.69 seconds. This represents a real bandwidth among processes of slightly more than 8Mb/s. Consequently, there

exists less than 20% of penalization in relation to the absolute hardware's maximum in the communication of broadcast messages among processes.

- When all the computers of CeTAD are used, the total communication time in each machine is 21.2 seconds in average. This represents a real bandwidth between processes of slightly more than 6 Mb/s. Thus, there exits a penalization of 40 % less than the maximum hardware absolute in broadcast messages' communication among processes.

Messages time metrics has hitherto been simplified by assuming that the time dedicated to data transference among computers is equal to the waiting time in communicating data in each computer. However, it is necessary to remember that the each computer's algorithm communication time is in fact the time to be waited until a broadcast message is completed. This means that all processing load unbalance makes some computers wait more or less time for the data. For instance, Table 4.18 shows that computers with more time dedicated to computation - **fourier** and **prited** - have less communication times because, in fact, part of the communication time counted on the rest of the computers is this "extra" computing time used by computers to process matrix data. Since load unbalance in terms of computing time does not reach the 10% of this time, it is not discriminated (at least, for the moment) as different from the communication time.

Apart from considering the communication performance, it is evident that the assessment of two aspects influencing parallel performance can also be considered: computing performance of each machine in particular, and the *real* load balance of the parallel machine.

Local-sequential computing performance of SeqMsg(UDP). From the point of view of each machine's computing performance, we should analyze how much penalization is imposed in relation to the maximum processing capacity due to the use of computers in parallel. In this sense, the most important factor to be identified or quantized might be the communication "interference" over each computer sequential computing performance. In other words, from the performance point of view, it is not the same to *only* compute-process data as to carry out computing periods and communication periods. It is clear that, for instance, the cache memory must be shared or used during all the periods and, thus, it is not the same to make a whole matrix multiplication as to process a third, communicate data, process the second third, communicate data, and process the last third.

When there are heterogeneous machines, the impact of communications in the computing performance will not be necessarily the same. Using the example of the use of cache memory, the impact on the computing performance can be different in principle, depending on the size of the cache memory. Appendix A shows the great variety of sizes and types of cache memory of CeTAD's machines.

Each computer local performance can be assessed by using the values that sum up the parallel running for five and ten data machines in Table 4.17, and in Table 4.18, respectively. Table 4.19 shows the Mflop/s of each machine when the parallelization of a 2000x2000 element matrix multiplication in five computers with the algorithm SeqMsg(UDP) is considered. Table 4.20 shows the Mflop/s when all computers are used. It is important to notice that:

- The performance in each computer remains invariant as more machines are added and,

thus, as granularity decreases. It must be recalled that increasing granularity implies increasing the quantity of messages for this algorithm and, thus, there exists more quantity of computing periods during which less quantity of operations are carried out.

- Each computer performance is quite near to the optimum, taking as optimum that obtained when *only* sequential computing operations are run. In this sense, the tests carried out can be taken as reference to assess the relative computing capacity, which for the case of CeTAD machines appears in Figure 4.4, and for **purmamarca i**n particular in Figure 4.5.

Name	***Rows***	***Tot. Comp.***	***Mflop/s***
purmamarca	555	14,15	314
cf1	426	14,14	241
cf2	426	14,09	242
sofia	394	14,40	219
fourier	199	14,78	108

Table 4.19: Mflop/s of SeqMsg(UDP) with Five Machines and n = 2000 in CeTAD.

Name	***Rows***	***Tot. Comp.***	***Mflop/s***
purmamarca	454	11,81	307
cf1	349	11,84	236
cf2	349	11,82	236
sofia	324	11,97	216
fourier	164	12,67	104
Josrap	142	12,10	94
tilcara	104	12,08	69
paris	48	11,59	33
cetad	38	11,98	25
prited	28	12,65	18

Table 4.20: Mflop/s of SeqMsg(UDP) with Ten Machines and n = 2000 in CeTAD.

Load Balance of SeqMsg(UDP). As it can be calculated from the values summed up by the parallel running times of the different tables, there appear some differences in terms of the time dedicated to local computation that should be explained in accordance with the load balance of that defined for the algorithm. The differences in local computing time that can be identified are:

- In Table 4.19, the shortest computing time is that of **cf1**, 14.09 seconds, and the longest computing time is that of **fourier**, 14.78 seconds. The time percentage implied by this unbalance is inferior to the 5% of the shortest computing time.
- In Table 4.20, the shortest computing time is that of **paris**, 11.59 seconds, and the

longest computing time is that of **fourier**, 12.67 seconds. The time percentage that this unbalance implies is inferior to the 10% of the shortest computing time.

It is clear that, even though from this theoretical point of view we can reach an equation or expression that allows an exact load balance, there exist differences in running times. Only as example, computers **cf1** and **cf2** are *equal* (Table 4.1 and Appendix A), and it can be proved both in Table 4.19 and Table 4.20 that computers have been assigned with the same processing load, though with different local computing times. In general, and independently of the types of computers, algorithms, and the ways of balancing the load, there will always be a minimum of differences in terms of running times.

Another source of a minimum load unbalance is given by the very definition of the algorithm. As explained in the previous Chapter, the load balance is implemented by assigning the result matrix row quantity corresponding to each computer's relative computing capacity. Consequently, the amount of data to be calculated in each computer is given in *number of rows* of the result matrix. However, the relative computing capacity of computers cannot always be expressed as multiples of each other (or multiples of the reference one) and, besides, the total quantity of rows will not always be distributed completely following this strategy. Thus, the correction factor "of a row" in the distribution of data explained in the previous Chapter can be considered as the cause of a possible workload unbalance.

Last, but not least, there is another important thing: the very heterogeneity of computers. Even though each computer holds a computing capacity near to the optimal, this "closeness" will not necessarily be equal in all of the cases. In other words, all the computers count with a minimum percentage of performance penalization as regards sequential computation when the parallel computation is being carried out, and this percentage can vary according to the physical characteristics of computers. The variation in this penalization percentage also generates a certain unbalance.

Even though all the load unbalance sources are accumulative, and despite the high heterogeneity in terms of computers' computing capacity, the worst that can be obtained from experimentation in terms of load unbalance is less than 10% of the best local computation.

OverMsg(UDP). All the details provided in terms of the analysis of result correspond to the algorithm with sequential computation and communications in each machine. What happens with the algorithm oriented to communications overlapped with local computation is different, and Figure 4.37 clearly shows that some values are quite different from the expected. In order to make a more complete comparison of the results obtained by both algorithms, and in order to identify more clearly the differences with the PVM-based implementation, the summary of parallel running with five machines is presented in the first place in Table 4.21.

Comparing these values with those of Table 4.6 corresponding to the implementation using PVM broadcast, the conclusions are similar to those resulting from the comparison of the algorithm SeqMsg, i.e., SeqMsg (PVM) with SeqMsg(UDP):

- Computing times are almost the same, i.e., in terms of local computing performance;

each computer has the same computing capacity (in Mflop/s, for instance).
- Communication times are very different; in this case, the broadcast performance using UDP is approximately ten times better than that of PVM.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
purmamarca	555	16,63	3,33	5,96
cf1	426	17,30	3,46	5,45
cf2	426	17,31	3,46	5,42
sofia	394	18,29	3,66	4,33
fourier	199	16,53	3,31	5,98

Table 4.21: Summary of OverMsg(UDP) with Seven Machines and n = 2000 in CeTAD.

When the values of Table 4.21 are compared to those of Table 4.17, it can be noticed that:
- Local computing times of OverMsg(UDP) are higher than those of SeqMsg(UDP), which is to be expected since not only data are transferred among computers, but also, in a same time interval, computing processes are run in each computer with communication processes, and thus the competition of the CPU and the cache memory, for instance, is greater and this is translated into a longer computing time.
- OverMsg(UDP) communication times are approximately a third of those obtained with SeqMsg(UDP). Since in fact the same data quantity is transferred among computers, a large part of each data transference occurs *while* a partial matrix of the result matrix is being solved in each computer. In this sense, at least for five computers, the overlapping of communications can be considered satisfactory.

However, as clearly seen in Figure 4.37, from the inclusion of **tilcara**, the performance not only is less than the expected, but also decreases significantly. Even though it can be proved by the inclusion of each of the machines from **tilcara** (**paris**, **cetad** and **prited**), what happens can be identified quite clearly with all machines' running summary shown in Table 4.22.

Such as Table 4.22 shows, the local computing time of the six computers with highest computing capacity of CeTAD (**purmamarca**, **cetadfomec1**, **cetadfomec2**, **sofia**, **fourier**, and **Josrap**) is quite similar among each other - of approximately 13±0.45 seconds. The computing time of the four remaining computers is quite longer and with greater disparity among each other - of approximately 20±5 seconds. The explanation of this is quite simple, and is related to the competition for the resources previously mentioned. In this algorithm, which carries out communications overlapped with local computation, it is clear that there should exist one or more processes in each computer being run (transferring data) while a partial computing period is being solved. This means that during the execution of a computing period, the most important resources (CPU and cache memory) will be shared with the communication process/es. This generates a degradation of the computation performance, and, in addition and according to Table 4.22, some computers can overlap computation and communications much better than others.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
purmamarca	454	12,39	1,24	21,08
cf1	349	12,99	1,30	20,48
cf2	349	13,01	1,30	20,46
sofia	324	13,45	1,34	20,07
fourier	164	13,28	1,33	20,18
Josrap	142	12,82	1,28	20,80
tilcara	104	18,34	1,83	15,08
paris	48	16,91	1,69	16,39
cetad	38	23,26	2,33	9,90
prited	28	24,81	2,48	7,91

Table 4.22: Summary of OverMsg(UDP) with Ten Machines and n = 2000 in CeTAD.

Thus, the differences in the performance of computations overlapped with communications depend on several factors, such as the system bus architecture or the I/O method by means of which data are communicated through the installed interface netboards. Even though it is quite difficult to explain in more detail what happens in each computer, this means that some computers are not really capable of *actually* overlapping local computation with communications but they sequentialize computation with communications and, thus, their performance is the same to that obtained with the sequential computation and communication algorithm. It could even be worse because, with OverMsg, there is much overhead at the level of processes being run and being handled by the operating system (generating a higher quantity of context changes, for instance).

Even though from the point of view of the total performance this explanation is not satisfactory (in fact, imposes a physical limit since it depends on the hardware of the same computer), the very algorithm with computation overlapped with communications *becomes* a *benchmark*. In other words, OverMsg(UDP) allows, automatically and independently of computers, to know with enough precision whether they are capable of overlapping computation with communications. In fact, this case may render an approximate quantification that could be based on the differences in time applied for computation in each computer. This benchmark is more significant when the fact that standard sequential computing computers are being used is acknowledged, when the possibility of overlapping computation with communications via one or more network interfaces does not seem to be among the primary objectives of the design.

Even though there exists an approximate explanation for the loss of the parallel computing performance as from the inclusion of **tilcara**, it seems reasonable to analyze in more detail what happens with each machine computing performance in particular and the *actual* load balance of the parallel machine.

OverMsg(UDP) local-sequential computing performance. As expected, and due to the

reasons already mentioned, computers' computing performance decreases in relation to the optimal situation, i.e., when only local computation is carried out. Since machines are heterogeneous, the decrease is also *heterogeneous*, depending on each computer's architecture.

Leaving aside the machines sequentializing computation and communications (from **tilcara**), it could be said that the cost of overlapping communications with local computation is less than the benefit since, in fact, the total performance increases.

OverMsg(UDP) Load Balance. Since the load balance is made by using the sequential computing performance and due to the *heterogeneous* interference of data transference processes on those of computation, OverMsg(UDP) load balance can be even worse than that of SeqMsg(UDP). Leaving aside the analysis of the load balance of those computers incapable of "effectively" overlapping communications with local computation from the performance point of view, the unbalance is just below the 10% in the case of the CeTAD computers. However, when this unbalance becomes significant, the running summary data can be used to redefine the distribution of data and distribute data in function of the computing performance assuming communications overlapping.

4.8.1.2 Matrices of 3200×3200 Elements

The speedup values obtained when we take as reference the greatest problem's running time that can be solved in the computer with highest computing capacity - i.e. matrices of order n=3200 in **purmamarca -** are shown in Figure 4.38, together with those shown in Figure 4.18.

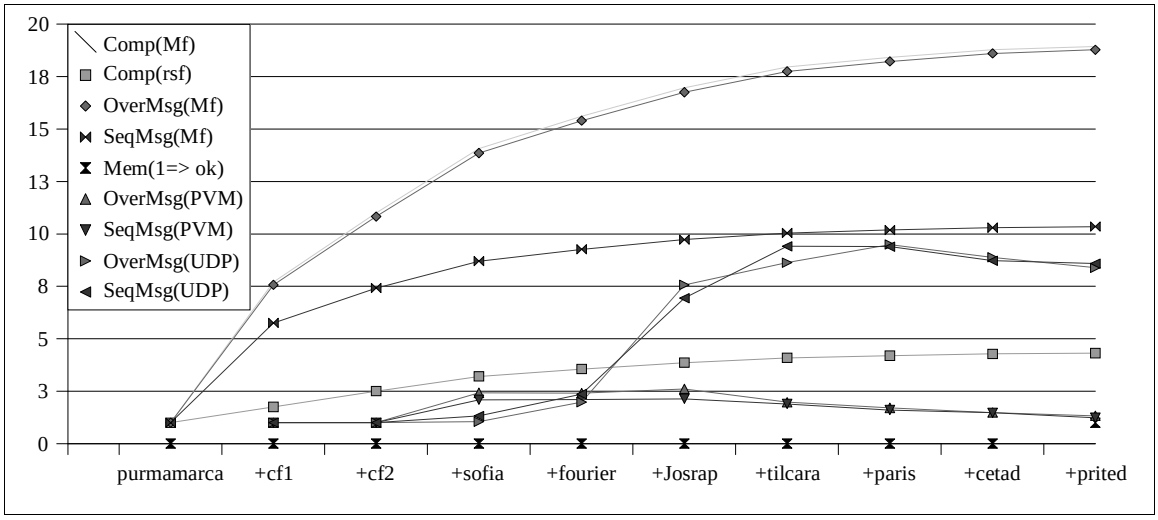


Figure 4.38: Speedup of Algorithms with UDP in the CeTAD Network for n = 3200.

As it can be noticed in Figure 4.38, none of the cases exceeds the estimation of SeqMsg(Mf), which is the optimal performance taking into account only the computing contribution of each of the computers without taking into account any time related to communications.

The really low performance obtained up to the inclusion of **fourier** is directly related to the algorithm memory requirements in each computer. More specifically, **cetadfomec1**, **cetadfomec2** and **fourier** have only 32 MB of installed memory and, thus, they are more likely to use swap memory space during the partial computations of the result matrix.

The values that sum up the parallel running appearing in Table 4.23 make clear that **fourier** is the computer that requires much more time than the rest in order to solve local computations. However, the values of Table 4.23 also show that a large part of the running time in all the machines is dedicated to the wait of broadcast messages transference and, in this case, this time can be affected by the use of the swap memory during the execution.

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
purmamarca	887	56,94	11,39	334,81
cf1	682	78,11	15,62	316,04
cf2	682	79,75	15,95	314,29
sofia	631	57,31	11,46	334,65
fourier	318	193,88	38,78	224,60

Table 4.23: Summary of OverMsg(UDP) with Five Machines and n = 3200 in CeTAD.

In some way, the memory estimation, Mem, in Figure 4.38 is showing that just with all the computers it is likely that there will not be any memory problems. The need of recurring to the swap memory makes:

- Both computing and communications processes have part of their executable code in secondary memory and, thus, they can be discontinued at some moment of the execution in order to recover the code.
- Both computing and communications processes have part of their data in secondary memory and, thus, they can be discontinued at some moment of the execution in order to recover those data.
- For all the processes, the swap time generates a delay in terms of running times, but in the case of communication processes the penalization can be even greater since they can lose data that are being sent during the swap time.

According to Figure 4.38, from the inclusion of **Josrap**, the performance is much higher, even though there are not so many variations with the inclusion of the rest of the computers. From the inclusion of **Josrap,** the swap memory is not used, or the use does not significantly penalize the execution of computing and communication processes. Table 4.24 shows the parallel execution summary with seven computers. Notice the important advance in computing performance as well as in communications performance.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
purmamarca	771	56,16	8,02	45,35
cf1	593	77,28	11,04	25,67
cf2	593	78,11	11,16	25,37
sofia	549	59,70	8,53	42,75
fourier	276	58,26	8,32	44,41
Josrap	242	52,93	7,56	47,07
tilcara	176	73,54	10,51	29,25

Table 4.24: Summary of OverMsg(UDP) with Seven Machines and n = 3200 in CeTAD.

In this particular case, the total computing time is "dominated" by the local computing time of computer **cetadfomec2**, which is still using the swap memory space. Since memory requirements on **fourier** have decreased with respect to the use of five computers, its running time dedicated to computation has significantly decreased in relation to that of Table 4.23.

Thus, when the swap memory use no longer influences the total parallel computation of one or more machines, the expected performance of the algorithm is obtained, interspersing computing periods with communications. In the specific case of SeqMsg(UDP), this goes on until communication times become very relevant or dominate the total time, which is giving an idea that the granularity of the problem is not suitable for more than eight machines. Still, there is no much loss of performance. In the specific case of OverMsg(UDP), once machines are used up to their maximum capacity (or at a relatively high percentage of their maximum capacity), the computers not capable of overlapping computation with communications are starting to be used, and, thus, the same performance as with SeqMsg(UDP) is obtained.

If the memory estimation is taken as reference to establish the quantity of computers (in an *a priori* selection) - Mem in the speedup graphics-, all the machines should be used. The result obtained for the matrix multiplication of 3200x3200 elements using the ten machines of CeTAD is approximately nine times the performance of **purmamarca** for the same memory size. This performance could be considered as "superlinear", since the sum of the relative computing powers of all CeTAD's machines is less than five times the computing **purmamarca**. Once more, it should be born in mind that the reference time for this speedup computation is the "actual" multiplication running time for matrices of order n = 3200, that in **purmamarca** implies the use of swap memory and a really high penalization with respect to the maximum processing capacity of this computer (without the use of swap memory space).

4.8.1.3 General Conclusions of the Experimentation in CeTAD

Even though the results obtained in CeTAD do not appear to be encouraging – since both SeqMsg(UDP) and OverMsg(UDP) decrease instead of increasing as from a given number of computers (Figure 4.37 and Figure 4.38) -, some quite significant information can be obtained from the experimentation carried out:

- The algorithm OverMsg(UDP) can be used as *benchmark* in order to identify which computers are capable of overlapping successfully computation with data transferences.
- In general, the optimal speedup estimation is not that which is eventually obtained by the overhead imposed by netboards, the operating system, etc.; however, by knowing the computers capable of overlapping computation with communications, it will be possible to define the quantity of computers to be used for a certain problem. More specifically: *which* ones.
- Memory requirement estimation, though not exact, can be useful when it is possible that some or all the computers may have problems with respect to the use of the swap memory space. When this information is obtained, it will be possible to choose the algorithm to be used (SeqMsg or OverMsg), depending on whether the computers to be included in the parallel computation are capable of overlapping computation with communications.

4.8.2 LQT Local Area Network

Most of the explanations made within the context of the experimentation in the CeTAD local network can be applied (may be on another scale) to the experimentations made in the LQT local network.

Figure 4.39 shows the speedup values obtained in the LQT local network by the algorithms implemented using UDP, SeqMsg(UDP) and OverMsg(UDP), for matrices of order n=5000 together with those of Figure 4.19.

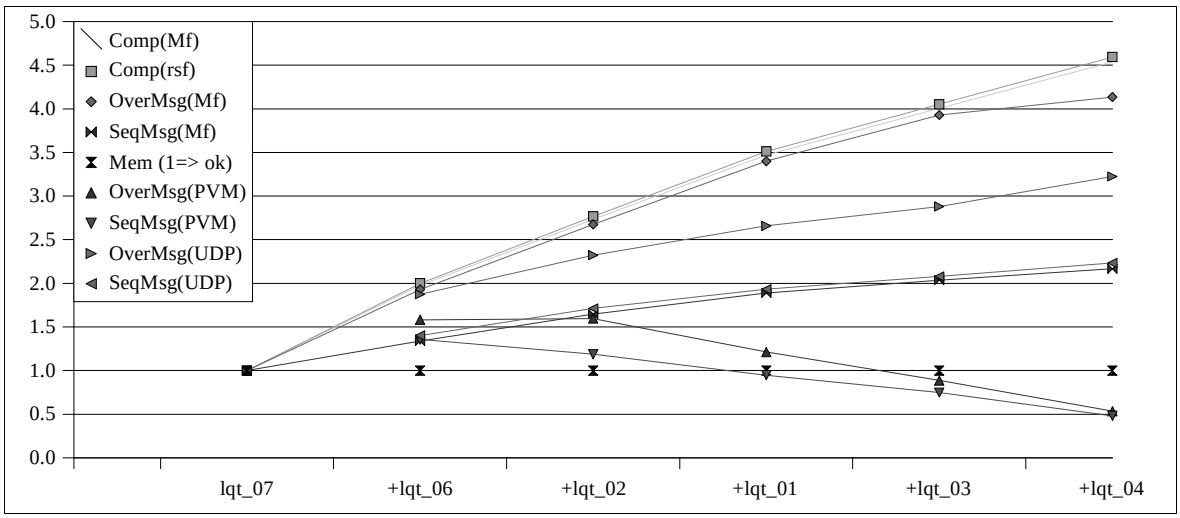


Figure 4.39: Algorithm Speedup with UDP in the LQT Network for n = 5000.

The speedup estimation for the sequential computation and communication algorithm is almost exact in relation to what is obtained from the experimentation, which is really satisfactory. In addition, it proves that the problem of SeqMsg(PVM)'s low performance, i.e. of the algorithm implemented using the PVM library, is precisely PVM's broadcast, which does not make use of the characteristics of the Ethernet network.

It is also evident that $O(n^3)$ –as regards this problem's requirements for floating-point operations - is as influencing in the total parallel running time as the $O(n^2)$ of the data quantity to be transferred through broadcast messages. For this reason, both the estimation of SeqMsg(Mf) optimal speedup and the speedup obtained in the SeqMsg(UDP) experimentation represent approximately the 50% of the optimal speedup independently of the algorithm used and of the communication network Comp(Mf).

Even though the algorithm with overlapped computation and communications does not reach the optimal speedup calculated for this algorithm, OverMsg(Mf), in the experimentation, it does not only have a better performance than the sequential computation and communication algorithm, but it also shows at least three aspects:

- Each computer's capacity to overlap, at least in part, the local computation with communications via the interconnection network. In fact, as it can be noticed in Figure 4.39, the final result in terms of performance is capable of overlapping approximately the 50% of communications.
- The overhead influence imposed by the operating system on other software layers is not worthless, and in fact is that which "consumes" the difference between OverMsg(Mf) and OverMsg(UDP).
- The application's granularity makes the obtained speedup, OverMsg(UDP), increase as more computers are added. In other words, the local computing time is comparable to the communications time and a good percentage of communications can be overlapped with the processing.

As regards computation overlapping with communications, what happens is in fact the same as (or similar to) what has been explained for the CeTAD network. Table 4.25 shows the running summary of SeqMsg(UDP) with all the machines of LQT in order to solve a matrix multiplication of order n = 5000 in parallel, and Table 4.26 show the running summary of OverMsg(UDP).

Name	***Rows***	***Tot. Comp.***	***Per It.***	***Tot. Msg.***
lqt_07	1089	85,58	14,26	89,66
lqt_06	1089	86,36	14,39	88,94
lqt_02	835	87,94	14,66	87,41
lqt_01	811	90,54	15,09	85,17
lqt_03	589	90,15	15,02	85,20
lqt_04	587	89,94	14,99	85,36

Table 4.25: Summary of SeqMsg(UDP) with Six Machines and n = 5000 in LQT.

Name	*Rows*	*Tot. Comp.*	*Per It.*	*Tot. Msg.*
lqt_07	1089	91,40	15,23	28,69
lqt_06	1089	91,11	15,19	28,95
lqt_02	835	99,53	16,59	20,59
lqt_01	811	95,30	15,88	24,38
lqt_03	589	96,68	16,11	22,80
lqt_04	587	96,43	16,07	22,89

Table 4.26: Summary of OverMsg(UDP) with Six Machines and n = 5000 in LQT.

From the comparison of the running times shown in each table, it can be said that:

- The computing periods in the execution of OverMsg(UDP) are quite higher than those of SeqMsg(UDP) due to the competition for the resources with the processes in charge of communications "in background".
- In the execution of OverMsg(UDP), a large percentage of communications is carried out "during" the local computing time. Whereas each computer must wait in average approximately 87±3 seconds for the data transference during the execution of SeqMsg(UDP), the wait is, in average, of approximately 25±4 seconds during the execution of OverMsg(UDP).

Figure 4.40 shows the speedup values obtained in the LQT local network by the algorithms implemented using UDP, SeqMsg(UDP) and OverMsg(UDP), for matrices of order n = 9000 together with those shown previously in Figure 4.20.

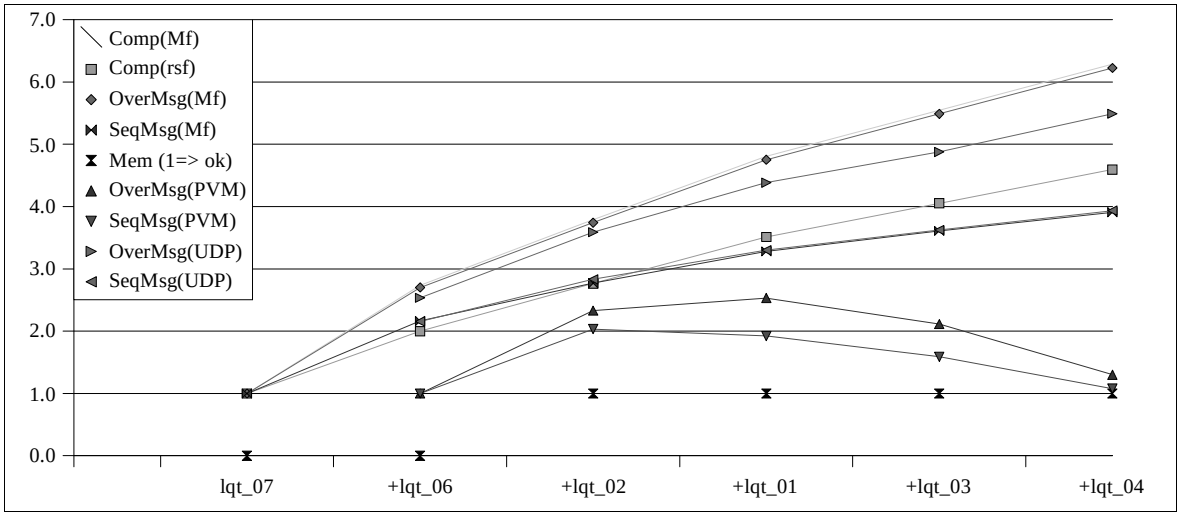


Figure 4.40: Algorithms' Speedup with UDP in the LQT Network for n = 9000.

Once more, the optimal performance estimation of the sequential computation and communication algorithm, SeqMsg(Mf), is almost the same as the obtained in the experimentation, which is SeqMsg(UDP). In the case of the optimal speedup estimation for

the overlapped computing and communication algorithm, OverMsg(Mf) is quite near to the obtained, OverMsg(UDP). The relative difference among values is quite inferior to that shown in Figure 4.339, and this is given due to the influence of the $O(n^3)$ of computing requirements on the $O(n^2)$ of communications requirement. And, it is evident that this difference makes the computing time percentage quite superior to that of communications for 9000×9000 element matrices than for 5000×5000 element matrices.

The performance obtained for the multiplication of matrices of 9000×9000 elements using the six machines of LQT is of approximately 5.5 times the performance of **lqt_07** for the same memory size. This performance could be considered as "superlinear", since the sum of the relative computing powers of all LQT's machines is of approximately 4.5 times the computing capacity of **lqt_07**. Once more, it should be born in mind that the reference time for this speedup calculation is the "actual" running time for matrices of order n = 9000, which, in **lqt_07,** implies the use of swap memory.

4.8.3 LIDI Local Area Network

Figure 4.41 shows the speedup values obtained in the LIDI local area network for the algorithms implemented using UDP, SeqMsg(UDP) and OverMsg(UDP), for the multiplication of matrices of order n = 2000 together with those previously shown in Figure 4.21.

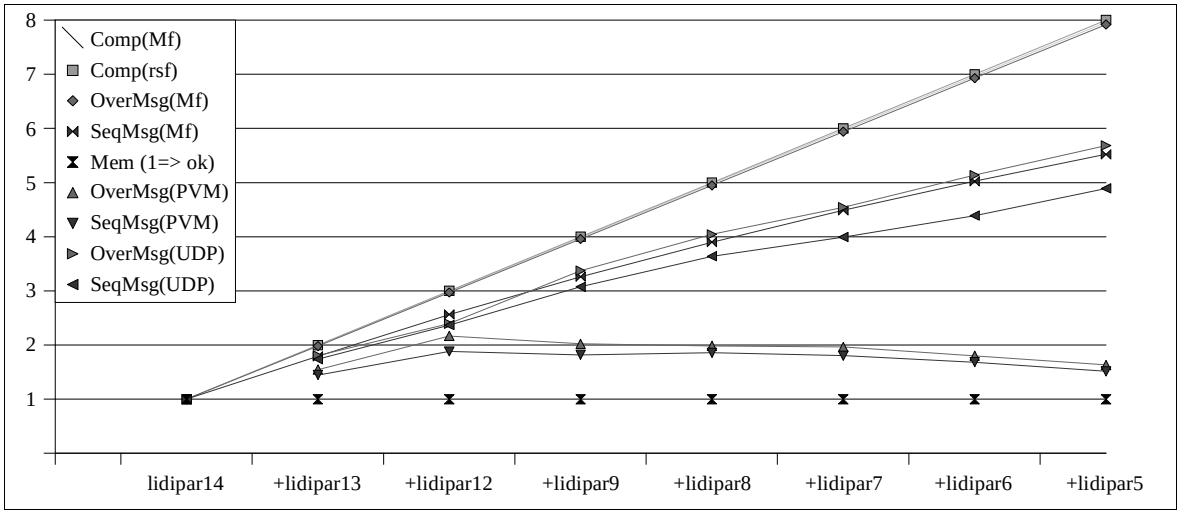


Figure 4.41: Algorithms' Speedup with UDP in LIDI Network for n = 2000.

In order to compare the estimation of the algorithms' speedup optimal values, SeqMsg(Mf) and OverMsg(Mf) with those obtained, SeqMsg(UDP) and OverMsg(UDP), it should be remembered that communication times are estimated basing on a network ten times faster than those of CeTAD and LQT. This, in turn, makes any overhead over the optimal computing time have a greater impact on the LIDI network than the two remaining. As general conclusions from Figure 4.41, it can be said that:

- Both SeqMsg(UDP) and OverMsg(UDP) are inferior to the corresponding estimations.

OverMsg(UDP) is the farthest from the estimations.

- Both algorithms have better performance as the quantity of computers increases. It is evident that the highest capacity of the communication network as well as the acceptable performance obtained with broadcast messages are important factors for this to happen.
- The algorithm designed to overlap communications with local computation in each machine, OverMsg, is better than that of sequential computation and communications, SeqMsg. This in turn indicates that computers are capable of overlapping effectively, at least in part, local computation with communications through the interconnection network.

Figure 4.42 shows the speedup values obtained in LIDI local network by the algorithms implemented using UDP, SeqMsg(UDP) and OverMsg(UDP), for the multiplication of matrices of order n = 3200 together with those shown previously in Figure 4.22.

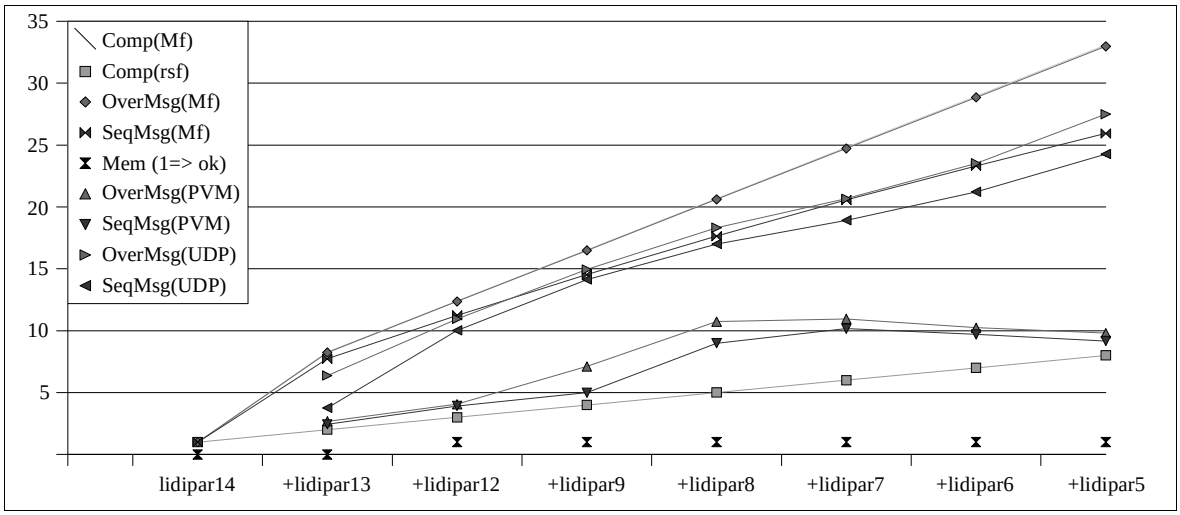


Figure 4.42: Algorithms' Speedup with UDP in LIDI Network for n = 3200.

These might be the best results of all the obtained in terms of performance. Such as Figure 4.42 shows, the eight computers of LIDI can solve a matrix multiplication of 3200x3200 elements more than 25 times faster than the very multiplication in one of them (they are all equal). Anyway, it should be reminded that the sequential reference time for the multiplication of matrices of order n = 3200 is penalized -with respect to the optimal- by the use of swap memory during the computations.

Once more, OverMsg(UDP) is better than SeqMsg(UDP), since computers can effectively overlap communications and local computation, and the algorithm also makes effectively use of this capacity. Figure 4.42 also shows that the algorithms' performance always improves when the number of computers is increased. This indicates that, as expected, the algorithm is scalable (at least up to eight computers) and, in particular, the implementation of broadcast messages among user processes using UDP is also scalable. Unlike with matrices of order n = 2000, with matrices of order n = 3200, the speedup values obtained are near to those estimated for the algorithm, all of which does nothing but prove the importance of the $O(n^3)$ of computation over the $O(n^2)$ of communications, which over the LIDI local network is heightened in comparison to those of the CeTAD and LQT.

4.9 Conclusions - Experimentation Summary

Once the characteristics of local networks in terms of computers and interconnection networks presented at the beginning of the chapter are identified, the results of the experimentation carrying out PVM as message-passing library, in general, and broadcast messages, in particular, are presented. From this experimentation:

1. The performance with PVM is unacceptable. In all local networks, i.e. independently of the number of machines, their heterogeneity or homogeneity, sizes of matrices used, and physical interconnection network performance, the results were the same: the performance worsens as more computers are used.
2. When making the execution profile with the minimum instrumentation, it is clear that the performance problem is always triggered by the PVM library broadcast routine, which is implemented by multiple point-to-point messages.

A broadcast message directly based on UDP protocol is proposed and implemented since, in principle, none of the general purpose message-passing libraries can assure a priori the optimized performance of broadcast messages. The same experiments previously carried out with PVM are repeated, concluding that:

3. The algorithm with computing and communication periods sequentially run (SeqMsg) provides, in most of the cases, the expected the performance. Exceptions can arise in the case of using swap memory in some computers.
4. The algorithm SeqMsg provides performance that improves when the size of matrices is not the highest (or scalates together with the quantity of computers).
5. The algorithm organized to overlap the computing and communication periods (OverMsg) has a better performance than the SeqMsg in all the cases, reason why the performance obtained by this algorithm can be considered as acceptable in all the local networks used.
6. The algorithm OverMsg can be used quite simply as benchmark with two purposes:
    1. Identification, in a simple manner, of the communications that can be carried out overlappedly (in background) while local computing is carried out in each computer.
    2. Identification of computers which are not capable of running computing and communications overlappedly and which thus penalize the parallel performance of the **whole** network used. In this sense, OverMsg can be used to discard such computers, or to yield a maximum of usable computers for a given application.
7. Both SeqMsg and OverMsg provide a very satisfactory speedup if the application or, more precisely, the size of the application affects the sequential performance due to he swap memory of the computer in which it is solved.
8. Both SeqMsg and OverMsg can be satisfactorily used both in homogeneous and heterogeneous clusters since in all the cases we can obtain optimized performance of
    1. Local computing of each computer and specifically balanced.
    2. Communications over the Ethernet interconnection network with broadcast messages.