Mecánica Computacional Vol. XXI, pp. 2945–2957
S. R. Idelsohn, V. E. Sonzogni and A. Cardona (Eds.)
Santa Fe–Paraná, Argentina, October 2002

A DISTRIBUTED COMPUTATIONAL IMPLEMENTATION OF A FINITE ELEMENT CODE USING FORTRAN IN WINDOWS 9X ENVIRONMENT

João R. Masuero[*], Armando M. Awruch[*]

[*] Centro de Mecânica Aplicada e Computacional, CEMACOM
Programa de Pós-Graduação em Engenharia Civil, PPGEC
Universidade Federal do Rio Grande do Sul, UFRGS
Av. Osvaldo Aranha, 99, 3º andar, Porto Alegre, RS, Brasil
e-mail: masuero@cpgec.ufrgs.br, awruch@adufrgs.ufrgs.br

Key words: Parallelization, Distributed Computing, Finite Element Analysis.

Abstract. *In order to solve large finite element problems, engineers and scientists have used both supercomputers and cluster of workstations. Supercomputers have high acquisition and maintenance costs, and has obsolescence in short period. Clusters of workstations are usually associated with specific network hardware and software, and special implementation techniques are necessary. The aim of this work is to test the viability of obtaining high performance computing using usual low cost PCs connected by low cost network hardware, using neither special operational system nor special programming languages, Only resources usually available in computing laboratories at research centers are used. Specifically, in this work were used PCs with at least 700 MHz processors and 128 Mbytes of RAM, connected by a 100 Mbit/s local network. All machines use windows 9x and the implementation was made using FORTRAN 90 language.*

An initial application implemented in this computational environment was a Finite Element code for 3D Solid Mechanics analysis in linear elasticity, using 8-nodes 1-integration point bricks with hourglass control, and a conjugated gradient solver using preconditioners. Results of scalability of solution time with the number of PCs used and the size of the problem are shown. Initial results show the viability of this approach, both in solution time and in size of problem that can be solved.

1 INTODUCTION

Numerical solution of Solid Mechanics, Fluid Mechanics and other Engineering fields using the Finite Element Method demands frequently high performance computational systems, with both high speed processors and large main and secondary memory systems for large data structures storage.

An usual approach for this needs of computational power is the use of supercomputers with several vector processors and shared memory. This is usually the most powerful solution, but its acquisition and operational costs are extremely high, demanding special facilities, its maintenance is complex and specialized and in few years its computational power is reached and surpassed by much cheaper machines. Its use is only viable in open centers serving several users. Besides, only optimized codes (with respect to vectorization and parallelization) achieve acceptable performances.

Another approach is the use of conventional clusters of machines connected by high performance networks and using specific operational systems, programming languages and code libraries. These clusters seldom allow the use of each machine as an ordinary computer for other common activities like text processing, graphic processing, code development and test. The operational system management and programming techniques are much more complex than those corresponding to single computers.

A new approach to high performance computing is the use of temporary clusters through internet-based personal computers[1]. Following this approach, this work presents the first results of a temporary cluster formed by ordinary computers, operational systems, programming languages and network hardware used in scientific computing laboratories, without specific components of software and hardware. By this way, operational and maintenance costs are very low using standard non-expensive components, avoiding the need of personal training in specific operational systems and programming languages and allowing a full time use of the diferent computers, both as cluster for high performance numerical processing and as isolated workstations for general computer activities.

Even if the performance of the solutions using common tools employed in high performance computing is nor reached, initial results points that the use of temporary clusters is very attractive for mid range computational problems, due to its good flexibility, easy operation and low costs.

2 METHODOLOGY

For the implementation of the temporary cluster, it was used only common hardware and software usually available in many scientific computing laboratories.

The workstations chosen to form the temporary cluster are mid range personal computers with CPUs of Pentium III class (intel Pentium III, AMD Athlon Thunderbird) with frequencies above 700 MHz, 128 Mbytes of main memory, hard discs of 10 Gbytes, 5400 rpm and above. This class of computers has high computing power, standard components and low cost, being usual in scientific laboratories for multiple purposes (text publishing, graphic publishing, CAD, numerical analysis and code development).

Each computer has a 10/100 Mbps standard Ethernet network card, and the cluster is connected by a low cost Hub-Switch Encore ENH908-NWY+ with eight 10/100 Mbps ports.

The operational system used in each computer is the Microsoft Windows 98SE. Although Linux has native support for clusters, the choice for Windows workstations is based in the fact that it was the operational system used in the computers of CEMACOM laboratory were the temporary cluster was implemented, and all personnel works with this operational system. Then, no additional training is necessary for the use of temporary cluster under Windows environment.

The programming language used for the temporary cluster implementation is FORTRAN 90/95, due to its high performance in numerical simulations, large use in scientific computing and many numerical codes already implemented using this language. Besides, almost all personnel in scientific computing laboratories use FORTRAN 90/95, being a natural choice for easy portability of cluster routines to existing codes.

Although the chosen programming language has no native support for network communication, no other language or library besides FORTRAN 90/95 was used to implement the routines of message passing and synchronization necessary for cluster operation. The information sharing among the several computers of the cluster was done using two important features of the operational system: a) local folders and files can be shared through the network for remote access; b) shared folders can be mapped as network units and consequently seen by FORTRAN as normal local disk units. Then, to get information from or to send information to a remote computer, FORTRAN reads or writes information in a file on a mapped network unit, making the routines for data sharing and synchronization extremely easy to be implemented and understood, since it corresponds to normal FORTRAN file access.

To send data or messages to another computer corresponds to write data in a disc file. To receive data or message from another computer corresponds to read data from a disc file. To wait for a message corresponds to a continuous data reading from a given record of a disc file, until other computer writes this record with the data or message waited.

A possible bottleneck in the performance of this approach is that sharing information allies two classes of slow operations: network communication (limited by a bandwidth of 100 Mbps for all simultaneous access to a given computer) and disk file access (which is one of the slower operations in modern computers). These characteristics lead to cluster implementation algorithms that minimize data sharing between computers in the cluster.

The network bandwidth available is shared by all the simultaneous accesses to a given computer. If there is only one computer B accessing data from a computer A, the communication occurs at full network bandwidth, 100 Mpbs. If there are two computers, B and C, accessing data from a computer A simultaneously, the communication between A and B, and A and C occurs at half network bandwidth, 50 Mbps. If there are three, four or five computers accessing data from a same compute, communications occur at 1/3 (33,33 Mbps), ¼ (25 Mbps) or 1/5 (20 Mbps) of network bandwidth, respectively, causing network

saturation and performance degradation. Otherwise, if computer B access data from computer A, and computer D access data from computer C, and computer F access data from computer E simultaneously, due to the presence of a switch connecting all computers, each communication occurs at full network bandwidth (100 Mbps) and no network saturation or performance degradation happens.

Considering these characteristics, the following cluster configuration was implemented:

Each computer has (n-1) mapped network units, where n is the number of computers in the cluster. Though these network units data sharing occurs performed by FORTRAN file access. This configuration is shown in figure 1.

Message passing and small data sharing is made by the computer that is sending the message writing data in remote files mapped by network units for all computers present in the cluster. This method, although implies in (n-1) writing operations for each message, avoid bandwidth saturation, since the receiving message process is performed by a continuous file reading operation by each computer in its own local disc unit. The time cost of multiplying writing operations is acceptable once the amount of data to be written is small.

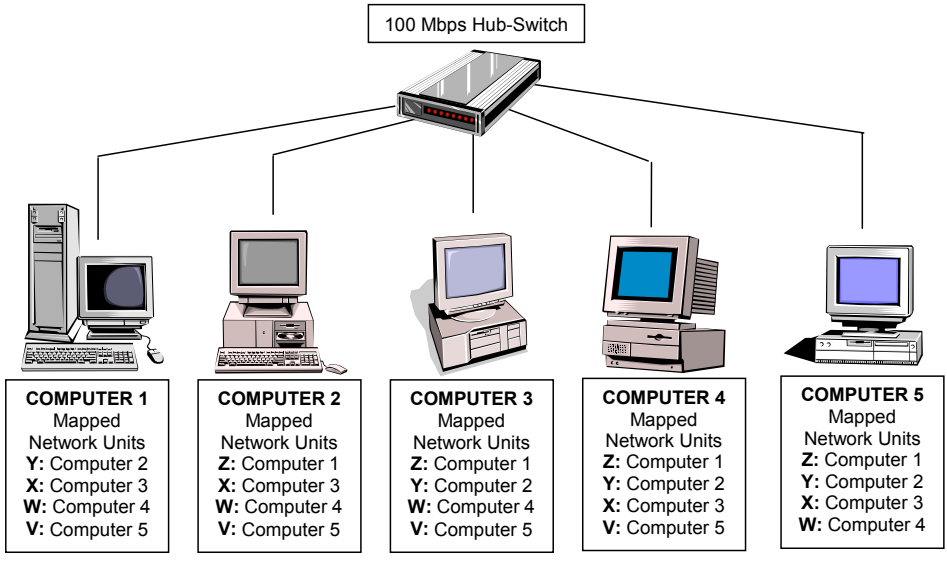

Figure 1: Typical configuration of a temporary cluster

Otherwise, large data sharing is done through an opposite approach: the computer that is sharing data writes files in a local disc unit, and all other computers present at the cluster read these files remotely through the network. Since the amount of data is large, the cost in time of writing (n-1) files on all remote units is too heavy to be supported with efficiency. The

waiting state for this case does not affect network bandwidth because it is done for a message (small data) indicating that the file with large data is available, as described previously.

To maximize de use of network bandwidth, all large data readings are made by each computer *i* in the following order: from computer *i+1* to *n* and from computer *1* to *i-1*. This rule to read a large amount of data is easy to implement and usually avoids the simultaneous access of data from one computer by more than one other computer.

3 IMPLEMENTATION

The methodology described above was first implemented in a code for static analysis of 3D linear elastic structures using hexahedral elements with 8 nodes, 1 point quadrature and hourglass control[2]. The solver used is the conjugated gradients interative process, with Cholesky and diagonal preconditioners[3].

As the solver is responsible for more than 90% of the total solution time, most of the parallelization effort was concentrated in this point. Conjugated-gradients is widely used in parallel computers with sharing memory and can be easily adapted for distributed computing due to low amount of data that must be shared among the different computers in each interaction.

For ill-conditioned problems, the incomplete Cholesky's factorization preconditioner presents better performance when compared to diagonal preconditioner, both in total time of solution and number of interations. As Cholesky's preconditioner is composed by a forward-substitution and a back-substitution over all equations, difficulties arise in the division of the tasks over several computers in a distributed computing cluster and low memory saving in this division, when compared to the memory used by a single computer solving alone the system of equations.

The diagonal preconditioner is less efficient than incomplete Cholesky's factorization with respect to the number of interations for all kind of problems, but in terms of total time of solution, it is slower only for ill-conditioned problems. Besides, the use of this preconditioner allows a good task division among the computers present in the cluster, low memory consumption and high scalability of the solver performance with the number of computers used, both in time of solution and size of problem. A diagonal preconditioner was used in all examples in this paper.

As the solver works directly over each equation of the system, the division of the task over the cluster is based directly in the nodes (or group of equations) of the model. In a n-computers cluster, each machine works over 1/n of the total set of nodes. Regarding elements, each computer works over all elements connected to the set of nodes assigned to it, resulting in more than 1/n of the total set of elements.

An algorithm of the conjugated gradient method[4] with modifications for distributed computing is shown in Figure 2, where, K^l is the stiffness matrix of each element l, *b* is the nodal load vector, *diag* is the vector of global stiffness matrix diagonal terms, *ug* is the nodal displacement vector, *re* is the residue vector, *pr* is the search vector, *z* is the preconditioned residue vector and *ugn, rn, prn* and *zn* has the same meaning, but in inteaction *n*. *Tole* is the tolerance obtained and *Lim* is the tolerance limit to stop the process.

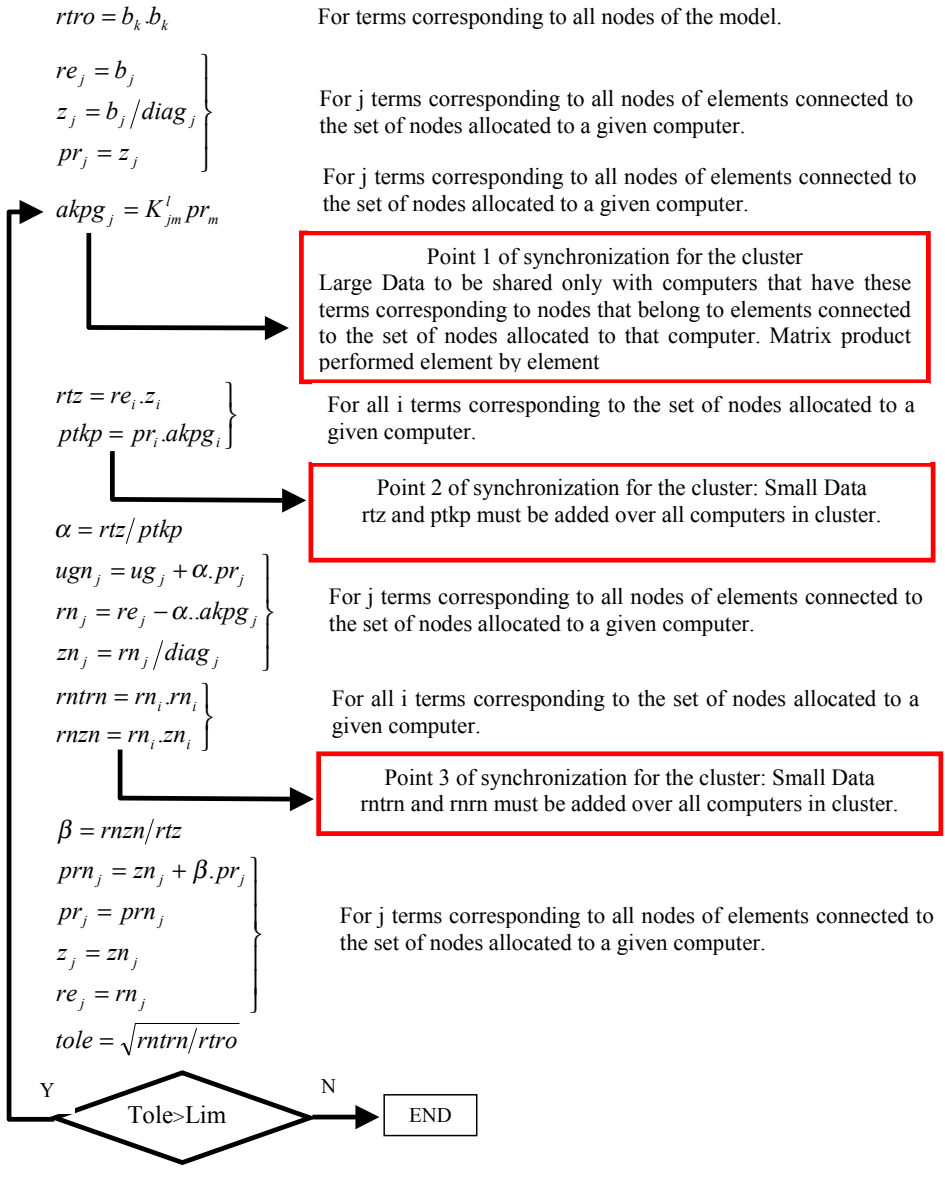

Figure 2 – Algorithm of conjugated gradient with diagonal preconditioner for distributed computing.

Considering that there are superposition of nodes and elements in the solver division of tasks and that the data sharing through network (and disc file operations) are relatively slow, the scalability of the solver with the number of computers will be less than 100% (1/n of the solution time for a n-computers cluster).

4 RESULTS

As a first test it was used a thick square plate with sides of 4 length units and thickness of 0,8 length units, simply supported over all the lower surface perimeter and subjected to a 10 force units concentrated load applied in the center of the upper surface. Young modulus is 10^7 force units by surface unit, and Poisson constant is 0,3. The general shape of the problem is despicted in Figure 3.

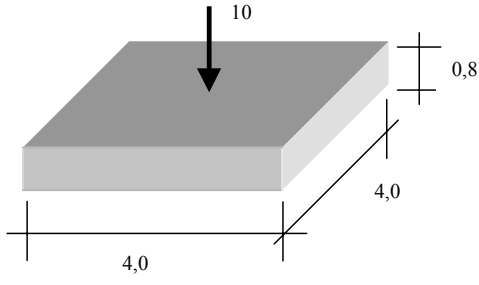

Figure 3 – Structure used in the performance tests.

This structure was modeled by successive meshes of elements with fixed aspect ratio of 1:1:1, given a family of examples with the same characteristics but increasing size. The use of elements with this aspect ratio in a thick plate leads to a very well-conditioned system and allows the maximum performance of the solver in number of interations. As the objective of this work is to compare solutions using only one computer to those obtained using distributed computing in temporary clusters, the choice of this family of examples seems to be adequated. The family of examples and its characteristics are shown in Table 1.

Table 1 – Characteristics of the family of examples used

Example	Mesh	Number of Elements	Number of Nodes	Number of Equations	Number of Interations
PLATEX040	40 x 40 x 8	12800	15129	45387	189
PLATEX050	50 x 50 x 10	25000	28611	85833	236
PLATEX060	60 x 60 x 12	43200	48373	145119	283
PLATEX070	70 x 70 x 14	68600	75615	226845	331
PLATEX080	80 x 80 x 16	102400	111537	334611	377
PLATEX090	90 x 90 x 18	145800	157339	472017	425
PLATEX100	100 x 100 x 20	200000	214221	642663	471
PLATEX110	110 x 110 x 22	266200	283383	850149	518

In order to evaluate the effect of the presence in temporary clusters of computers with different computing power, two clusters with two computers each were configured with the machines described below:

im700S: Intel mobile Pentium 700 MHz, RAM 128MB PC100, Hard disk 10 GB, 5400rpm
T1000S: AMD Athlon 1000 MHz, RAM 512MB PC133, Hard disk 40GB, 7200 rpm.
T1400S: AMD Athlon 1400 MHz, RAM 512MB PC133, Hard disk 40GB, 7200 rpm.

Network with speeds of 10 and 100 Mbps were used. Results are shown in Figure 4.

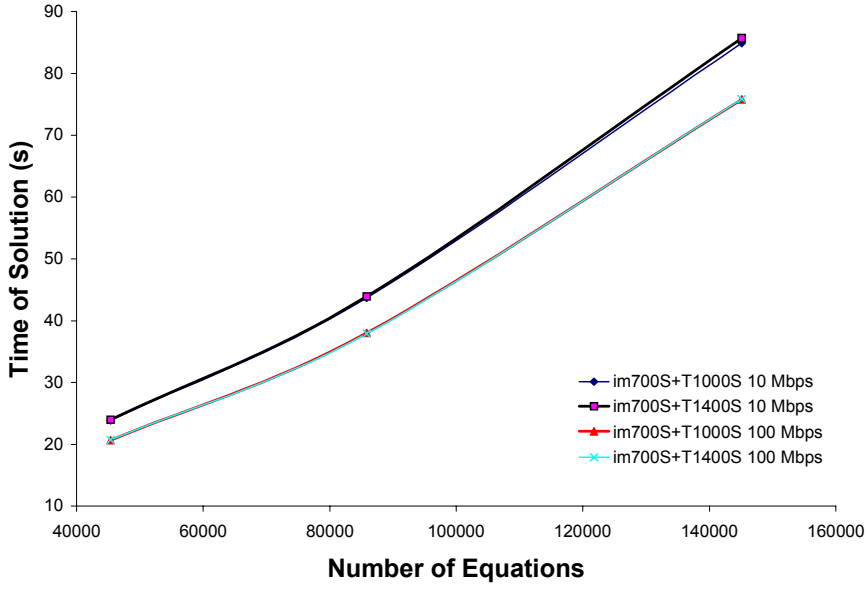

Figure 4 – Solution time of clusters with computers of different power.

Since a uniform task division was used, the global performance is governed by the slowest computer. This conclusion is confirmed by the identical performance reached by each cluster tested, for both network speeds, although one cluster has a computer with 40% more computing power than the other.

A similar conclusion could be stablished regarding the size of problem that can be solved by a cluster, governed by the computer with the smallest RAM.

So, for performance sake, it is better to use only similar computers to form a temporary cluster than try to use every computer available in a given laboratory, despite its characteristics.

The influence of the network speed is better understood observing Figure 5, where the total solution time was converted in processing speed (Mflops) using the solution obtained with a Cray T94 as comparison. T1333D computer has the following description:
T1333D: AMD Athlon 1333 MHz, RAM 256 MB PC266, Hard disk 20 GB, 7200 rpm.

Figure 5 shows that the slowest computer, im700S has an average computing speed of 151 Mflops, and a cluster of two identical computer (since in uniform task division, all computers in a cluster are equivalent to the slowest one, im700S + T1000S = 2 x im700S) have an average computing speed of 246 Mflops for 10 Mbps network and 279 Mflops for 100 Mbps network, corresponding to 163% and 185% of the performance of a single computer, respectively. A cluster of faster computers like T1333S (359 Mflops average speed) presents 496 Mflops for 10 Mbps network, and 646 Mflops for 100 Mbps network, corresponding to 138% and 180% of the performance of a single computer, respectively.

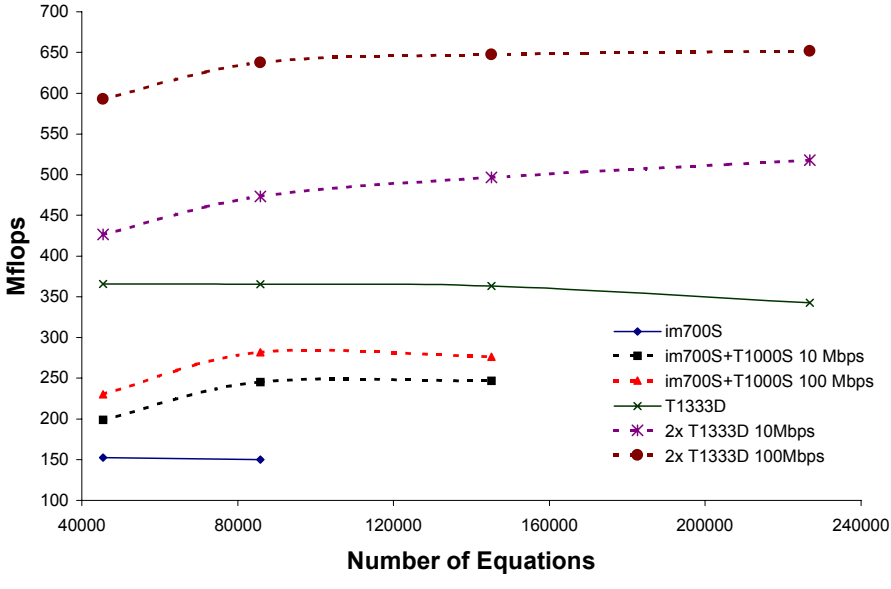

Figure 5 – Computing speed of temporary clusters for different network speeds

These results point out that network speed must be compatible with the speed of each computer in the cluster. An ordinary 10 Mbps network can be efficient for clusters made of low end performance computers, but it is totally inadequate for clusters made of high speed computers. The network bandwidth saturation with the increase of the number of computers in a cluster must be considered in the choice of the network speed; this aspect is shown in Figure 6.

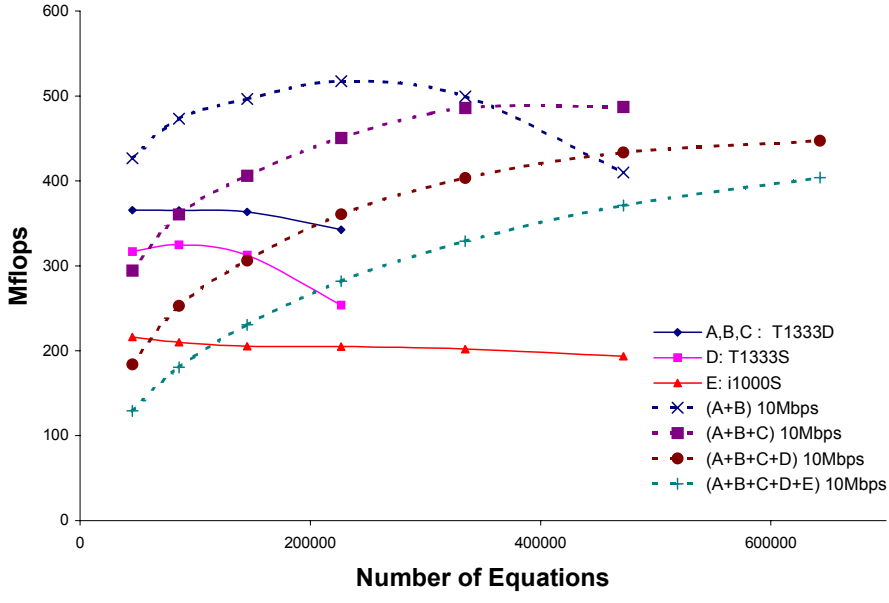

Figure 6 – Network saturation in a cluster of fast computers and slow network

For fast machines like T1333D, the use of a simple 10 Mbps hub for network connection is worth only for 2 computers clusters. Three or more computers in the cluster lead to worse performance in computing speed, being useful only to solve larger problems.

With a fast Ethernet network (100 Mbps) its is possible to form clusters that are efficient both in computing speed and size of problem. Figure 7 shows temporary clusters made of 2 up to 5 computers, being the first 3 of one class (359 Mflops of average speed) and the last two are slower (302 Mflops and 208 Mflops, respectively), with the characteristics shown below.

T1333D: AMD Athlon 1333 MHz, RAM 256 MB PC266, Hard disk 20 GB, 7200 rpm.
T1333S: AMD Athlon 1333 MHz, RAM 256 MB PC133, Hard disk 20 GB, 7200 rpm.
i1000S: Intel Pentium III 1000 MHz, RAM 256 MB PC133, Hard disk 20 GB, 7200 rpm.

T1333S differs from T1333D by the use of SDR memory (PC133) instead of DDR (PC266).

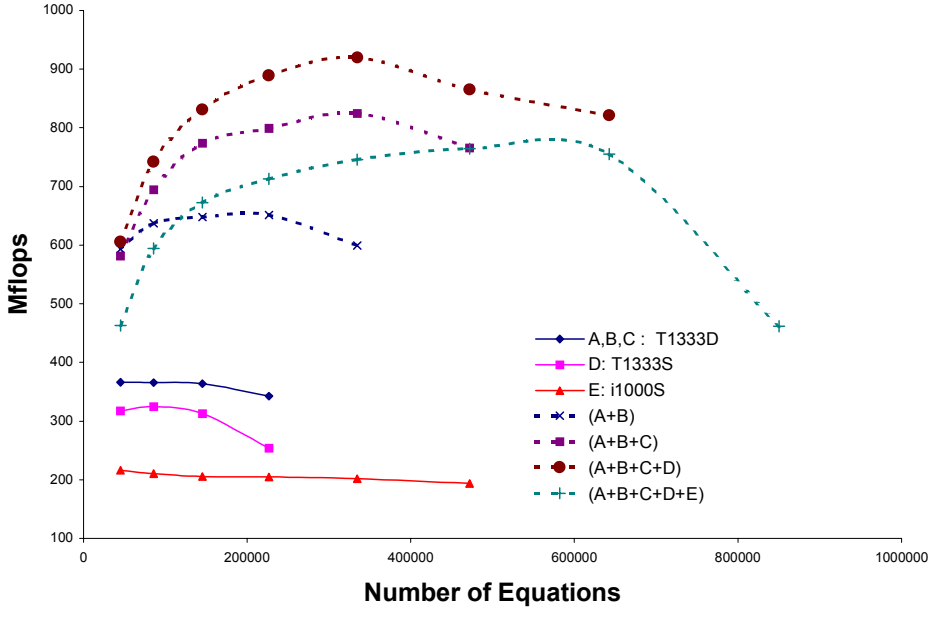

Figure 7 – Temporary clusters of fast computers and fast Ethernet network

The clusters have the following average performance:

2 x T1333D:	646 Mflops or 180% of a single T1333D
3 x T1333D:	791 Mflops or 220% of a single T1333D
3 x T1333D + T1333S:	877 Mflops or 290% of a single T1333S
3 x T1333D + T1333S + i1000S:	744 Mflops or 358% of a single i1000S

The cluster with 4 computers presents the best performance in computing speed. The cluster with 5 computers is worse because the fifth computer is much slower than the others, and the slowest computer rules the cluster. An average computing speed of 877 Mflops was reached, with 920 Mflops of peak. This performance is comparable to the result that can be reached in a single processor of a CRAY T94 supercomputer currently available at the CESUP/UFRGS (from 700 up to 1000 Mflops).

Regarding efficiency of parallelization, clusters have the following values:

2 x T1333D:	646 Mflops / (2 x 359 Mflops) = 90,0%
3 x T1333D:	791 Mflops / (3 x 359 Mflops) = 73,4%
3 x T1333D + T1333S:	877 Mflops / (4 x 302 Mflops) = 72,6%
3 x T1333D + T1333S + i1000S:	744 Mflops / (5 x 208 Mflops) = 71,5%

where the efficiency was calculated by the division of the average computing speed of the cluster by n times the average computing speed of the slowest computer in the cluster. These results show that it is possible that no network bandwidth saturation will occurs in clusters up to 8 computers of that class. With 8 computers similar to T1333D and an efficiency of 60%, an average computing speed of 1720 Mflops could be reached, which is almost twice the average performance of a single processor of Cray T94 supercomputer.

The curves shown in Figure 7 represent only the problems that could be solved without intense use of virtual memory (memory in disk, which is automatically allocated by the operational system). Therefore, a computer with only 256 Mbytes of RAM (main memory) solved problems up to 220000 equations. Larger problems could be solved by these machines, but the performance would be $1/10^{th}$ to $1/20^{th}$ of that verified in smaller problems. The descending final branch of the curves represent an intense use of virtual memory. So, how much the cluster performance is better than a single computer performance depends on how large is the problem to be solved. Huge problems can be 20 times faster in a cluster only by the fact that in a single computer where practically the whole problem must be solved using virtual memory (where disk performance rules) and in a cluster of several computers the problem fits inside the main memory (RAM). All comparisons presented here consider problems that does not use virtual memory intensively.

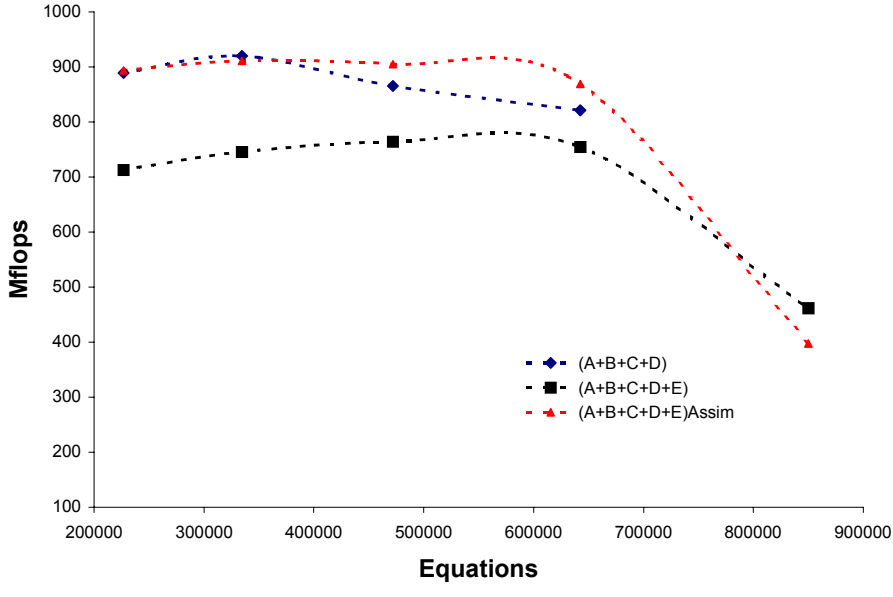

Figure 8 – Performance of non-uniform cluster

Computers with less computing power rule the cluster. It is possible to avoid this by considering a non-uniform task distribution. In Figure 8 the task division for a 5 computers cluster (the same of Figure 7) was pondered by the relative performance of each machine. As the task of each machine is defined by the number of nodes allocated on it, it is extremely easy to implement non-uniform task division. The uniform task division 5 computer cluster has an average performance of 744 Mflops, and the non-uniform cluster has 895 Mflops, around 20% more.

The slowest computer in the cluster has the larger memory (512 MB against 256 MB of the others). So, with 850000 equations, the performance of the non-uniform cluster was worse than the uniform one, due to the fact that computers with less memory carried bigger tasks, using virtual memory more intensively.

5 CONCLUSIONS

The methodology used in the cluster implementation seems to have acceptable efficiency for application to small clusters and problems of reasonable size. The implementation is easy and the use of common programming language is a very desirable characteristic. The possibility of using temporary clusters of low cost hardware make this approach very useful for small research centers

Clusters of similar computers in speed and memory seem to be the best configuration. Slow networks (10 Mbps) are bottleneck for clusters of mid range and more powerfull computers, and fast Ethernet networks (100 Mbps) must be used. Clusters of high-end machines probably will need 1 Gbps networks.

Non-uniform clusters are good alternatives to use all available computers in a given laboratory, but the best configurations still seems to be clusters of equivalent computers.

6 REFERENCES

[1] S. J. Kim and C. S. Lee, “Large-scale structural analysis by parallel multifrontal solver through internet-based personal computers”, *AIAA Journal*, **40,** No.2, 359-367,(2002).

[2] T. J. R. Hughes and R. M. Ferencz, “Large-scale vectorized implicit calculations in solid mechanics on a Cray X-MP/48 utilizing ebe preconditioned conjugate gradients”, *Comp. Meth. Appl. Mech. Engng.,***61**, 215-248, (1987).

[3] L. A. Duarte Filho, *Static and dynamic linear and geometrically non linear analysis with eight node hexahedrical elements with one point quadrature (in portuguese: Análise estática e dinâmica, linear e não-linear geométrica, através de elementos hexaédricos de oito nós com um ponto de integração).* MSc. Thesis, PPGEC, Universidade Federal do Rio Grande do Sul, 112p. (2002).

[4] E.L. G. Alquati, *Preconditioning of the conjugate gradient method with na element-by-element formulation (in portuguese: Precondicionamento do método dos gradientes conjugados numa formulação elemento-por-elemento).* MSc. Thesis, PPGEC, Universidade Federal do Rio Grande do Sul, 117p. (1991).