Chapter 5

Parallel CFD in PC clusters

5.1 Implementation details of the SIMPLE-like in parallel

From chapter 2 to chapter 4, we have introduced the different ingredients needed for the computation in parallel of CFD problems by means of the domain decomposition technique in MIMD machines.

- Governing equations, discretization, SIMPLE-like and time marching algorithms in chapter 2.

- Computation of the solution of linear systems of algebraic equations in single processor architectures in chapter 3.

- Parallel computation of the solution of these systems of equations in a PC cluster and the Cray T3E by means of the domain decomposition technique, in chapter 4.

In this chapter, all of these ingredients are reused and applied to an incompressible fluid flow problem with a three dimensional and complex flow structure that requires a fine grid discretization. The solution of a large size problem such as the presented later on, cannot be carried out by a single processor architecture due to high demand of computational resources, i.e., it requires both a high speed of processor and large RAM storage devices.

However, if the original problem is split in sub domains small enough to be fitted and distributed among processors of a MIMD machine and a communication system is provided for sharing data, then each processor can solve independently and in parallel their own part being feasible the solution of the overall problem.

To do so, we re-write the SIMPLE-like algorithm embedded in the time marching method to be used in parallel with the communication steps explicitly remarked. The flowchart and its implementation are described in Fig. 5.1 and Alg. 40 respectively

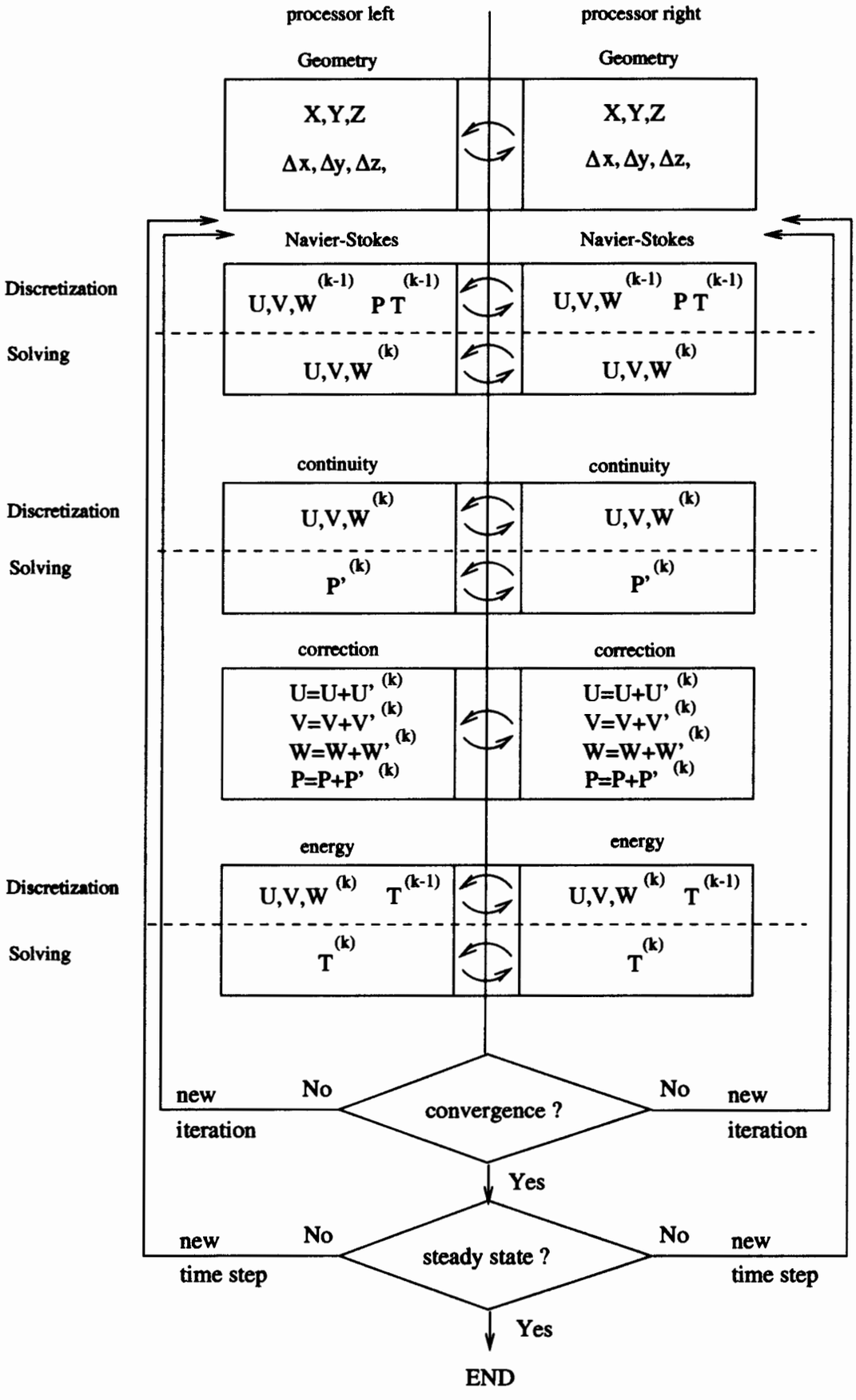

Figure 5.1: parallel SIMPLE plus time marching flowchart.

Algorithm 40 Parallel SIMPLE-like plus time marching

start *with* $\tau = 0$
  $\vec{V}^{\tau},\ P^{\tau},\ T^{\tau}$
do
  new *time step* $\tau = \tau + \Delta\tau$
  start *with* $k = 0$
    $\vec{V}^{(k)} = \vec{V}^{\tau-\Delta\tau},\ P^{(k)} = P^{\tau-\Delta\tau},\ T^{(k)} = T^{\tau-\Delta\tau}$
  do
    new *iteration* $k = k + 1$
    guess *fluid flow variables*
      $\vec{V}^{*} = \vec{V}^{(k-1)},\ P^{*} = P^{(k-1)},\ T^{*} = T^{(k-1)}$
    evaluate *coefficients of Navier-Stokes equations*
    solve $a_{P,u}u_P^{(k)} + \sum_{NGB} a_{NGB,u}u_{NGB}^{(k)} = b_{P,u}$
    solve $a_{P,v}v_P^{(k)} + \sum_{NGB} a_{NGB,v}v_{NGB}^{(k)} = b_{P,v}$
    solve $a_{P,w}w_P^{(k)} + \sum_{NGB} a_{NGB,w}w_{NGB}^{(k)} = b_{P,w}$
    communication *of du, dv, dw*
    evaluate *coefficients of continuity equation*
    solve $a_{P,P'}P_P^{\prime(k)} + \sum_{NGB} a_{NGB,P'}^{(k)}P_{NGB}^{\prime(k)} = b_{P,P'}$
    correct *the pressure*
      $P^{(k)} = P^{(k)} + \alpha_P P^{\prime(k)}$
      communication *of* $P^{(k)}$
    correct *the velocity*
      $\vec{V}^{(k)} = \vec{V}^{(k)} + \alpha_V \vec{V}^{\prime(k)}$
    evaluate *coefficients of energy equation*
    solve $a_{P,T}^{(k)}T_P^{(k)} + \sum_{NGB} a_{NGB,T}^{(k)}T_{NGB}^{(k)} = b_{P,T}^{(k)}$
    communication *of* $\vec{V}^{(k)},\ P^{(k)},\ T^{(k)}$
  until (*mass, momentum and energy conservation*)
  $\vec{V}^{\tau} = \vec{V}^{(k)},\ P^{\tau} = P^{(k)},\ T^{\tau} = T^{(k)}$
until (*steady state of all variables*)

By comparison with the sequential one, it is worth noting that only few new steps have been introduced while the remaining steps, and in general, the algorithm has been kept unchanged. From the implementation point of view of this algorithm, it is better to start with a description of the grid generation and the grid point indexation of centered and staggered grids for each subdomain.

For simplicity, a two dimensional rectangular domain is considered and divided by two in x direction obtaining two parts the left and the right (see Fig. 5.2). The division or partition is done on the centered grid (see grid painted in black for the original and in blue

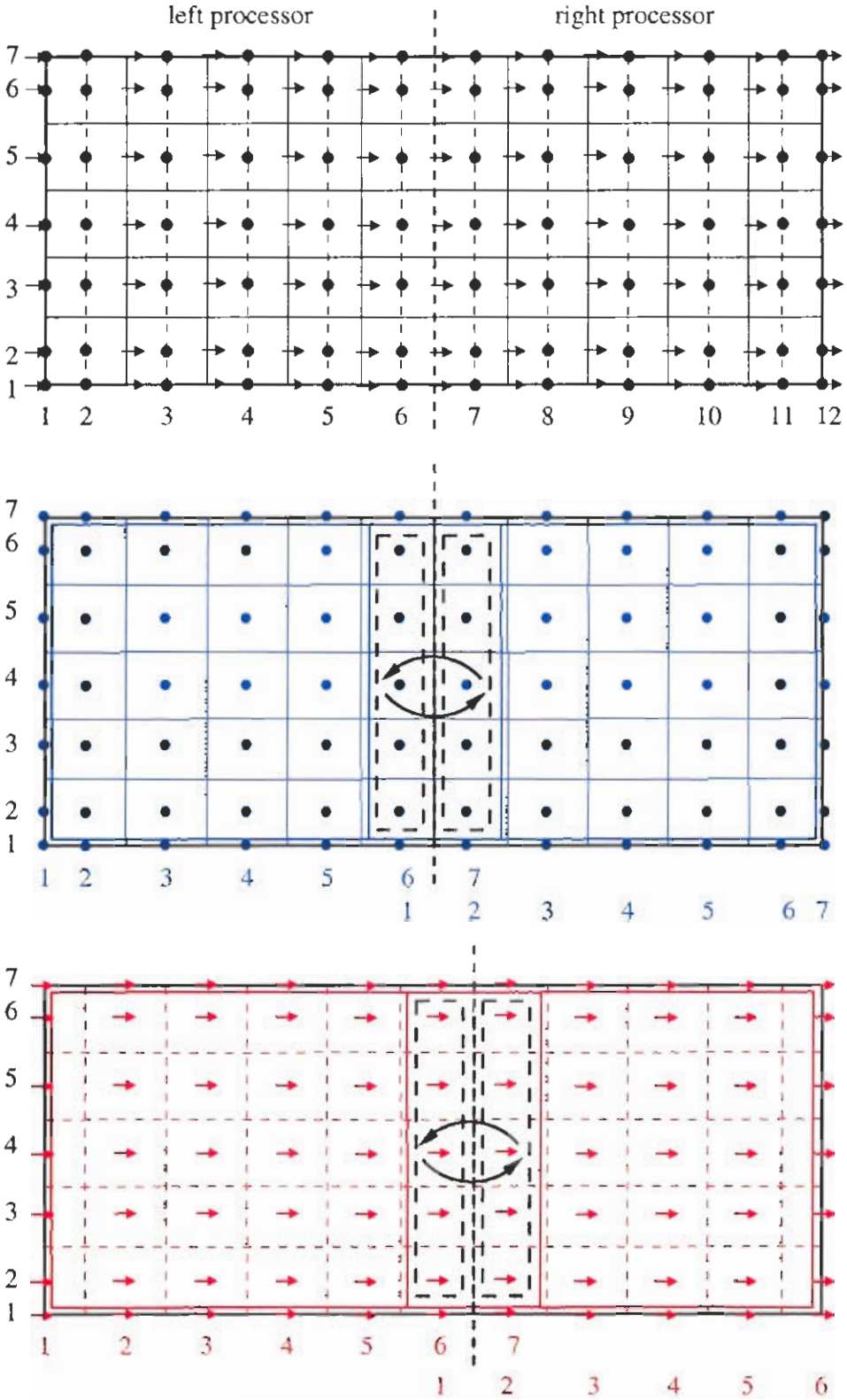

Figure 5.2: Top, centered (solid line) and staggered (dashed line) grids in x direction for the overall domain. Center, centered grid in blue split in x direction. Bottom, staggered grid in red in x direction.

for the partitioned) as balanced as possible. Once we have the centered grids associated to each processor, the staggered grids can be build independently at each processor (shown here in red for the x direction).

It is shown in the figure the index of each grid point for both partitions left and right. The zones enclosed with dashed lines contain the grid points that are involved in the communication processes between both processors. For the centered grids, these points stand for the pressure and pressure correction variables while for the staggered grid ones in x direction the zone enclose the velocity variable in x direction that will be involved in the communication processes.

These zones define the boundaries in contact between the processors for each subdomain. If the boundaries of a subdomain do not keep in contact with other subdomains, then we have to fix them with the boundary conditions explained in chapter 2, i.e., the Dirichlet, Neumann or periodic boundary conditions in order to close the problem for each subdomain.

Following with the explanation and related with the communication issues there are two points inside the algorithm where the communications are required: the communication of the set of variables before the evaluation of the coefficients of each system of the SIMPLE algorithm and the communication of a single variable inside the solver when needed. The first type of communications, say update of variables, are needed before the evaluation of coefficients near the boundaries in contact with neighbour processors. For instance, in order to evaluate the convective term of the Navier-Stokes equations at such boundaries, it may be necessary the velocity placed at the neighbour processor. The amount of velocity points from the neighbour processor depends on the scheme used. For our purpose, i.e., the Upwind and the Central Difference Schemes only one line of points is required. For higher order schemes, two or even more lines of points are needed. If that is extrapolated to the three dimensional case, a thin slice of one point is needed for Upwind and Central Difference Schemes.

The Navier-Stokes equations require also the gradient of the pressure variable so it is also necessary to obtain those points placed at the neighbour processors. It is worth noting that each variable has its own indexation system per processor and the communication of the variables of these systems must be considered carefully in order to place at the right location these variables on the receiver processor. Conversely it is also important to give the right location of the variables to be sent to the neighbour processors.

Fig. 5.3 show the communications needed from the processor right to the processor left before the evaluation of the Navier Stokes coefficients in x direction. The velocities U, V (figured with straight arrows) and the pressure P (figured with filled dots) are sent (figured with curved arrows) from the enclosed zones of the right processor to the left processor.

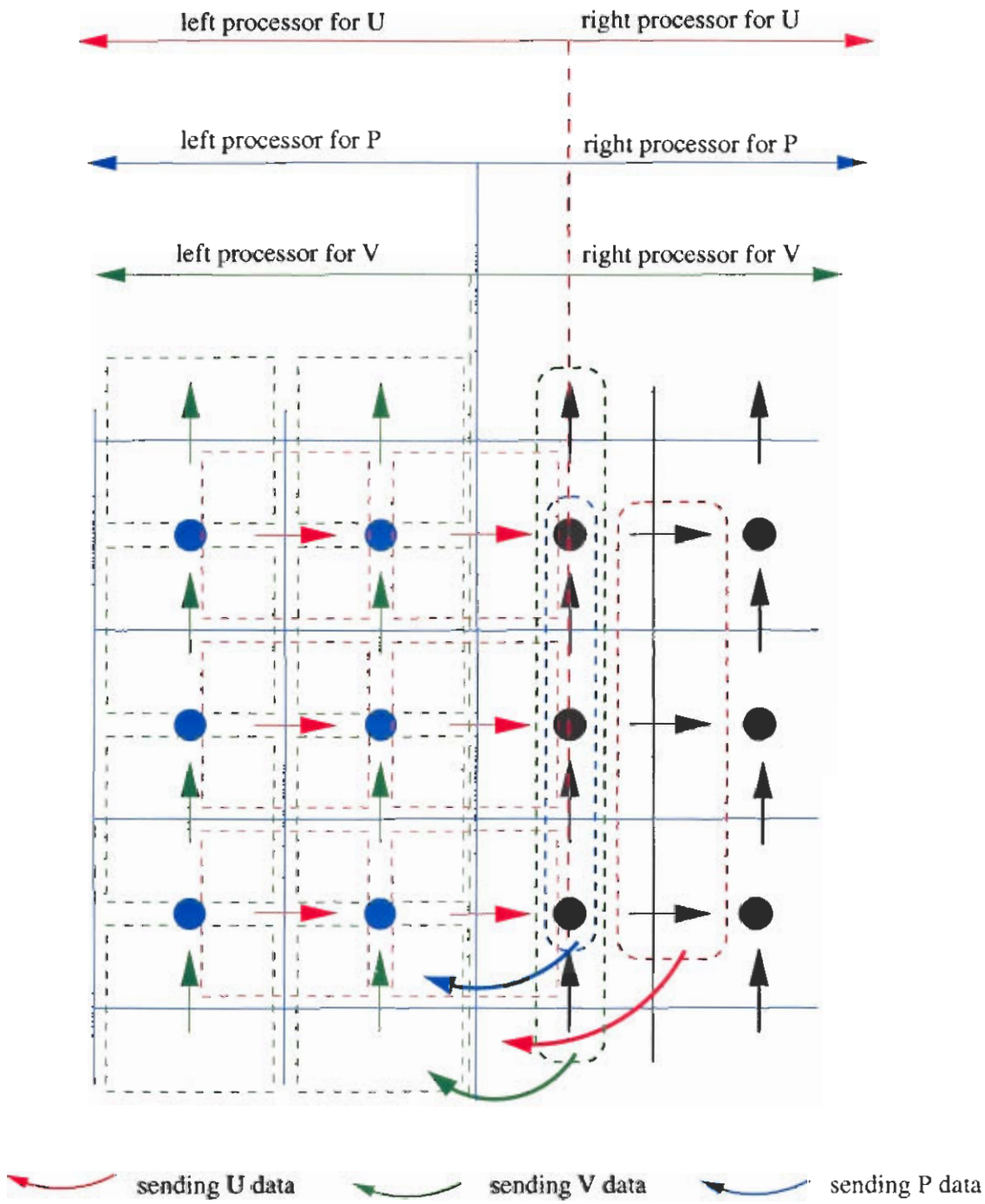

Figure 5.3: Communications from the right processor to the left processor before the evaluation of the coefficients of the Navier Stokes equation in x direction.

Once, all of these communications are done, i.e., the U, V, W and P the algorithm proceeds to the evaluation of the coefficients of Navier-Stokes equations for each direction.

Another point within the algorithm that requires such communications is placed before the evaluation of the coefficients of the continuity equation. In this case, the evaluation of these coefficients require the values of du, dv and dw that have been evaluated in the Navier-Stokes equations. Since these values are placed on staggered grids and the continuity equation is placed on the centered grid, we have take carefully the indexation of each original and destination location for each value.

Finally, the correction step and for instance, the correction of the velocities requires the gradients of the correction pressure so it is necessary to receive the correction pressures that are placed in neighbour processors. Again, the index of the centered grid associated to the pressure correction is different to the index of the staggered grids. Therefore, we have to consider the locations from the centered grid of the neighbour processors to the staggered grids of the receiver processors.

Once the systems of equations are defined the solver can compute the solution for each subdomain in parallel. Since these systems, as explained in chapter 3, are featured by non symmetric coefficient matrices, we have adopted the incomplete factorization solver MSIP for the three systems of Navier-Stokes equations and the Krylov space based method BiCGSTAB preconditioned with MSIP for the computation of the solution of the remaining equations, i.e. the continuity equation and the energy equation.

5.2 Benchmarking the parallel implementation

The lid driven cavity flow is one of the well known benchmark test for a CFD code. It has been chosen among other benchmark tests due to its geometric simplicity but complex flow structure. As was shown by Koseff and Street [60], the flow structure in a 3D cavity becomes more complex than a 2D cavity. Their experiments demonstrated that the presence of side-walls produced a three-dimensional structure that significantly altered the primary flow in the central plane, behaviour that 2D simulations simply could not capture. In Fig. 5.4 the experimental setup and a sketch of the flow structure are shown. The flow structure is composed by a primary eddy, a downstream secondary eddy, an upstream tertiary eddy and a corner eddy.

For laminar flows ($Re < 6000$) they found two sources of 3D structure. Examining a plane parallel to the downstream wall, corner eddies were caused at the juncture of the side-walls and the ground. Downstream secondary vortices caused as well by centrifugal forces along the downstream eddy separation surface were found along the span. These came to be known as Taylor-Gortler-Like vortices [61] in reference to their curvature-induced origins. The number and location of these vortices were function of Reynolds number.

Simulations were performed for cavities of SAR (Span Aspect Ratio) 1:1 and 3:1 for a wide range of Reynolds numbers: $10^3 - 10^4$. Koseff and Street reported on one hand the nondimensionalized height of the downstream secondary eddy with the Reynolds number and on the other hand the central and end-wall plane velocity profiles for a Reynolds number of 3200.

These experimental results are compared with a 3D simulation based on the SIMPLE and time marching algorithms onto a $120 \times 120 \times 120$ grid of points. In order to reduce the computational effort, the domain has been divided among 8 processors with a partitioning

configuration of $2x \times 2y \times 2z$ (see Fig. 5.4), such that each processor contains $60 \times 60 \times 60$ grid points.

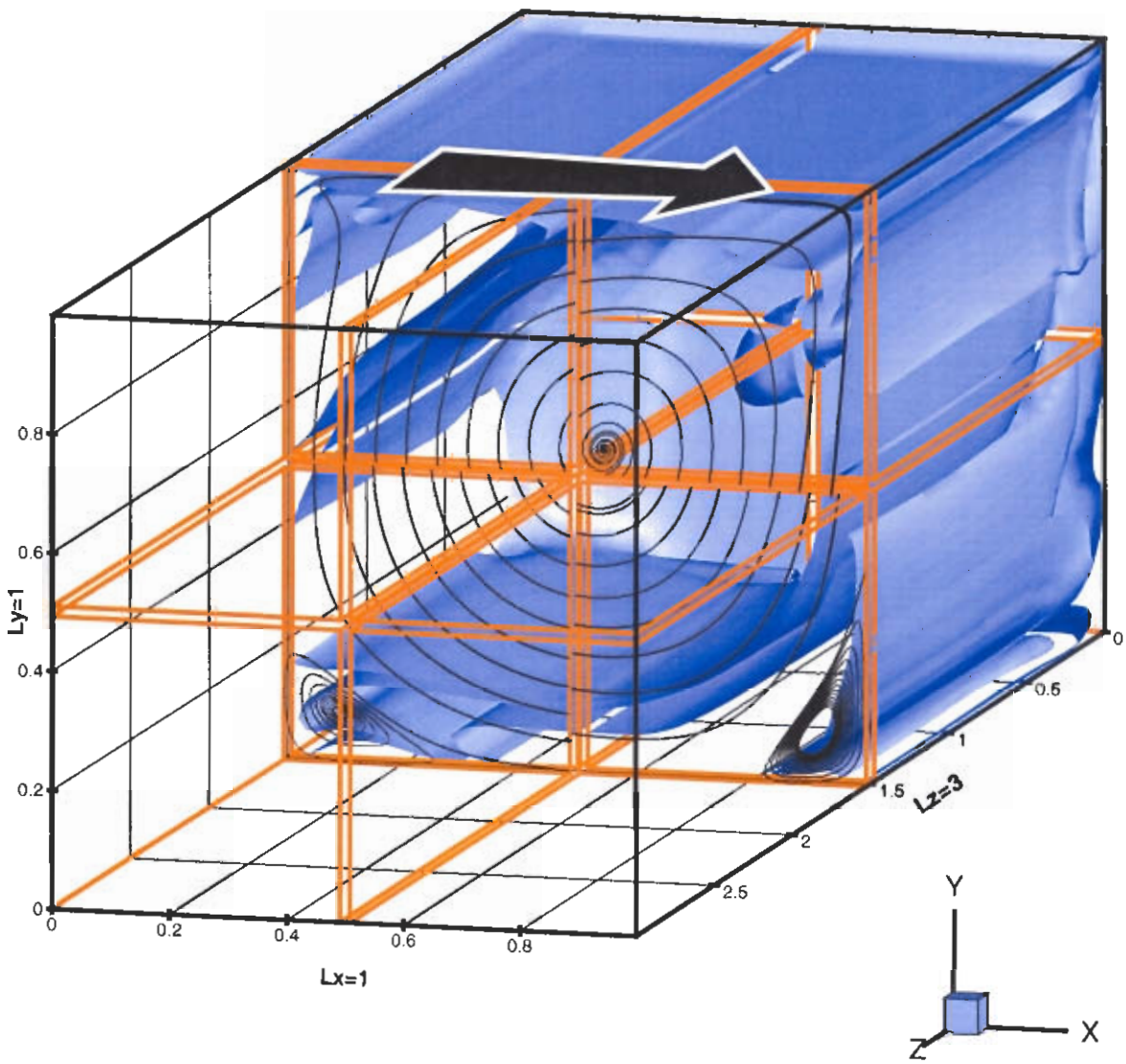

Figure 5.4: Domain partitioned in three directions $2x \times 2y \times 2z$ in order to be simulated with 8 processors

The first results provided by the simulation refer to the Reynolds number 3200. The fluid-flow variables are depicted in figure 5.5 with stream lines at few locations inside the domain and they are conducted to show the 3D structure of the flow. The three orthogonal central planes X, Y and Z are also shown for clarity of the structure of flow. The velocity

vectors have been scaled for a better visualization.

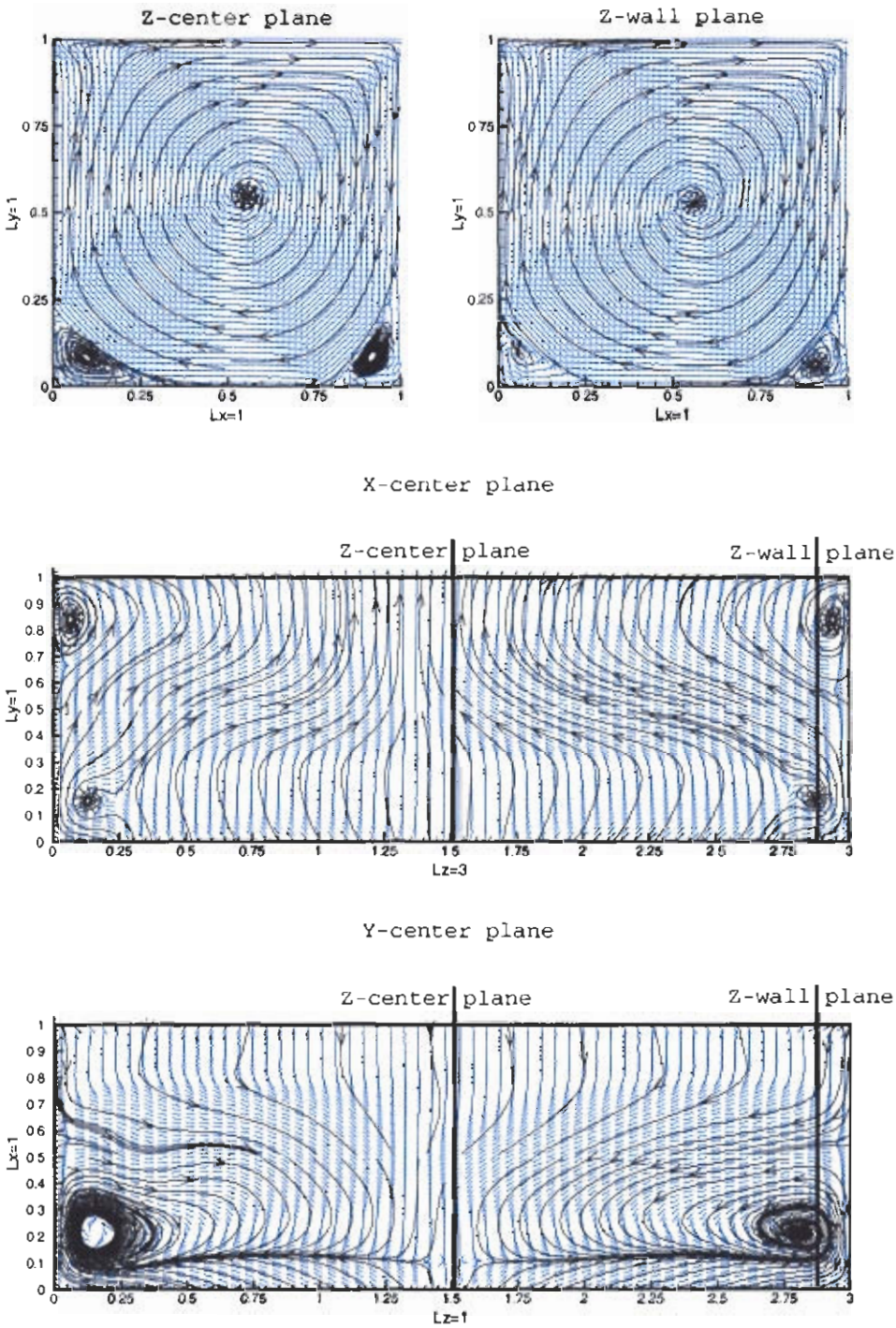

Figure 5.5: Left-top, Z central plane. Right-top Z end-wall plane. Center, Y central plane. Bottom X central plane.

The U and V velocity profiles for the Z central plane have been compared with the experimental data of Koseff and Street (see Fig. 5.6). The figure shows the good agreement for two simulations with grids $60 \times 60 \times 60$ and $120 \times 120 \times 120$ executed in parallel with 8 processors at PC cluster.

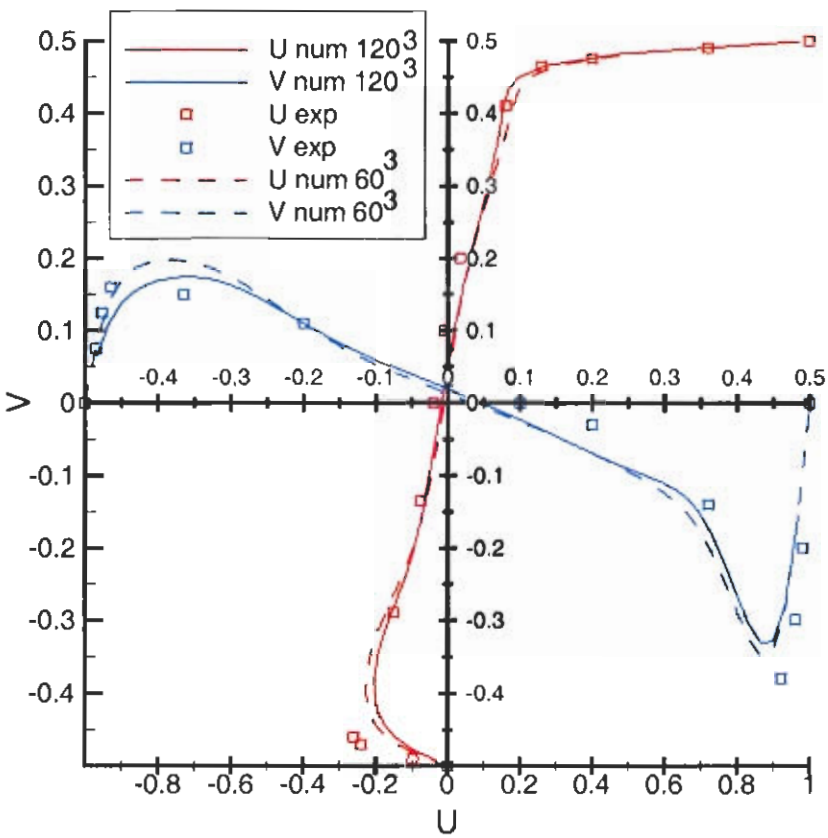

Figure 5.6: Numerical and experimental velocity profiles for Reynolds number of 3200 at Z central plane.

It is worth noting the main differences produced at the vivicinity of walls that could be reduced by refining the grid, concentrating grid points onward walls and using high order schemes.

Moreover, numerical results provided by DPC code developed at CTTC (see DPC and CTTC in acronyms page) have been added to the validation of the current code. The DPC code is also based on the finite volume method with SIMPLE-like coupling algorithm, and hence, the numerical results at same grid should be identical.

For DPC code, a maximum of $60 \times 60 \times 60$ regular grid points with upwind scheme may be fitted in a single processor. Concentration grid factors at the vicinity of the walls as well as higher order schemes such those based on the deferred correction, produce several instabilities for this Reynolds number.

The agreement of the numerical results with the experimental data give small discrepancies at maximun values of center lines (see table 5.1).

data source	experimental	DPC	present code	present code
grid	–	$60 \times 60 \times 60$	$60 \times 60 \times 60$	$120 \times 120 \times 120$
V_{max}	0.19	0.18	0.20	0.25
x location V_{max}	-0.41	-0.31	-0.39	-0.35
V_{min}	-0.38	-0.32	-0.35	-0.45
x location V_{min}	0.46	0.45	0.45	0.45
U_{min}	-0.28	-0.20	-0.23	-0.29
y location U_{min}	-0.40	-0.33	-0.39	-0.40

Table 5.1: Discrepancies of results.

The second results presented refer to the height of the downstream secondary eddy when varying the Reynolds number from 1000 to 8000 (see Fig. 5.7).

The differences respect to the experimental data may be due to different factors. From the numerical point of view, we do not know if the grid mesh is fine enough to simulate accurately the fluid flow phenomena. Furthermore, since the upwind scheme has been used for the treatment of the convective terms, it introduces a numerical diffusion of the velocity values and hence it affect to the global result. A more detailed simulation by means of a finer grid would reveal the tendency of the numerical solution. However, the storage capacity of computers is limited even in distributed machines so this study has been stopped at this stage. From the experimental point of view, Koseff et al. reported a measured error of $1 - 10\%$ at all ranges of velocities. Therefore, the agreement between numerical and experimental results has been considered satisfactory.

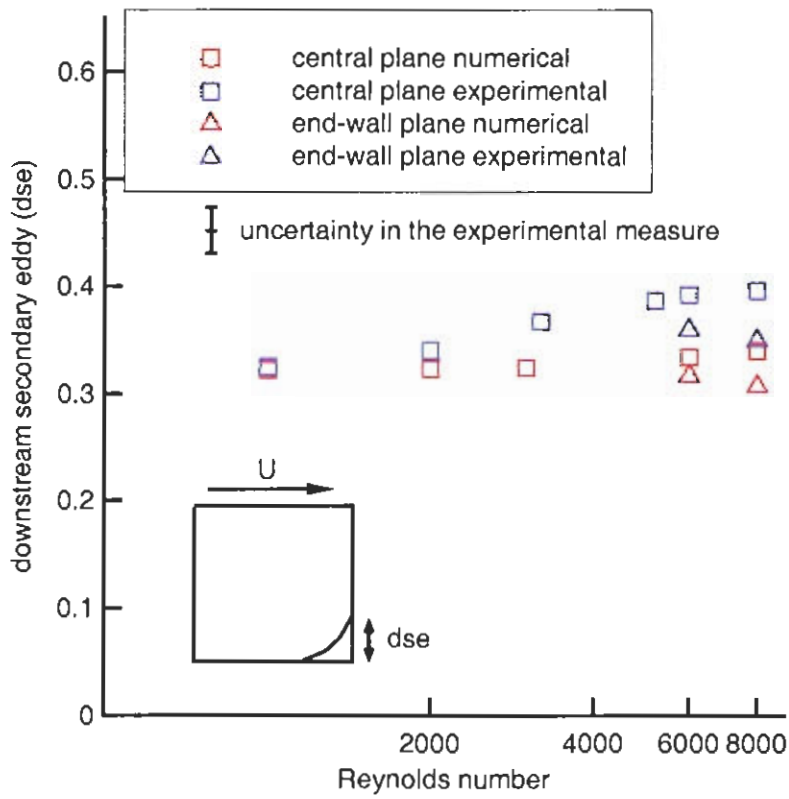

Figure 5.7: Height of the downstream secondary eddy for different Reynolds numbers

Finally, if the simulation is carried out for Reynolds numbers larger than 6000, the structure of the flow becomes unstable [60] being more difficult the convergence. At those Reynolds, the Taylor-Gortler-like vortices and the corner vortices are clearly visible. The longitudinal Taylor-Gortler-like vortices are formed because the surface of separation between the primary eddy and the downstream secondary eddy is effectively a concave wall. The curved separation surface promotes the instability caused by the centrifugal forces on this wall, leading to the formation of Taylor-Gortler-like structures. The corner vortex originates from the adjustment of the shear and pressure forces acting on the recirculating fluid to the non-slip condition imposed by the presence of the end-wall.

These structures are the most evident manifestation of three-dimensionally in this flow and they have been simulated at Reynolds uumber 8000 onto a 120x120x120 grid of points with 8 processors. The streamlines of the flow near the corner composed by the end-wall, ground and downstream-wall are represented in Fig. 5.8.

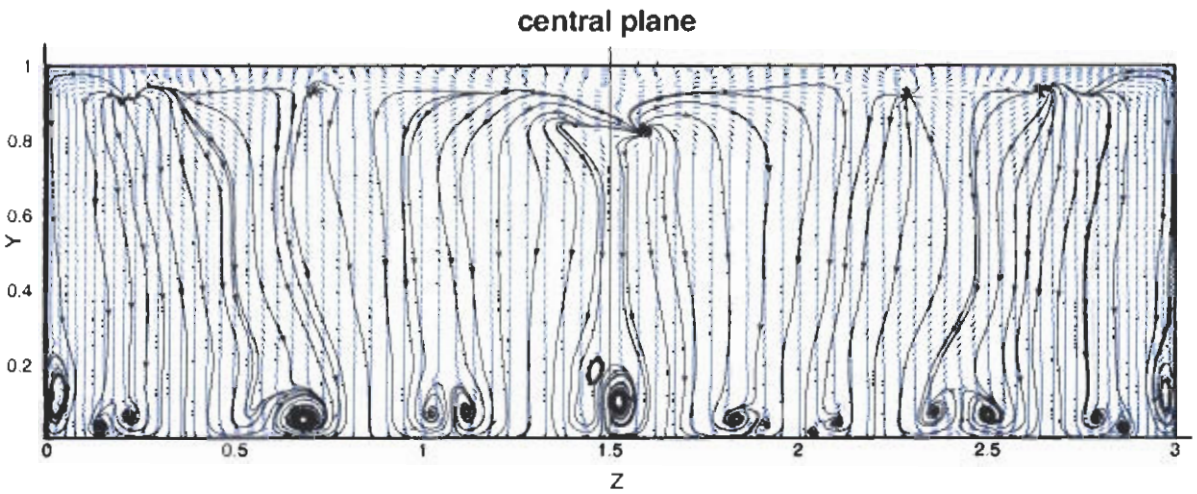

Figure 5.8: Pairs of Taylor-Gortler-like vortex along the span wise direction for a Reynolds number of 8000. There are also shown the corner vortex at both end walls.

5.3 Performance measure of the parallel implementation

In previous chapter, we have done an analysis of the parallel performance of several solvers once given a model problem and submitted to different number of processors, partitioning configurations and varying the size of the problem. Following this scheme, the 3D lid driven cavity for a Reynolds number of 1000 in a $1 \times 1 \times 1$ cubic box has been chosen for the parallel performance of the SIMPLE and time-marching algorithm in parallel. The measured wall clock time catches the time elapsed from the initial state up to the steady state. Several grid sizes $20 \times 20 \times 20, 40 \times 40 \times 40, 60 \times 60 \times 60, 80 \times 80 \times 80i100 \times 100 \times 100$ have been used in order to see the effect of the grid size per processor and to give an idea of the scalability. The measures of the speed-up have been done within a range of 1-12 processors.

Analogously to the results presented in the previous chapter, the speed-ups are given in tables 5.2, 5.3 and represented in Figs. 5.9, 5.10 for both machines, i.e., PC cluster and the Cray T3E. It is worth noting that in the case of the Cray T3E, the maximum grid size that can be hold in RAM by a single processor is limited to a $60 \times 60 \times 60$.

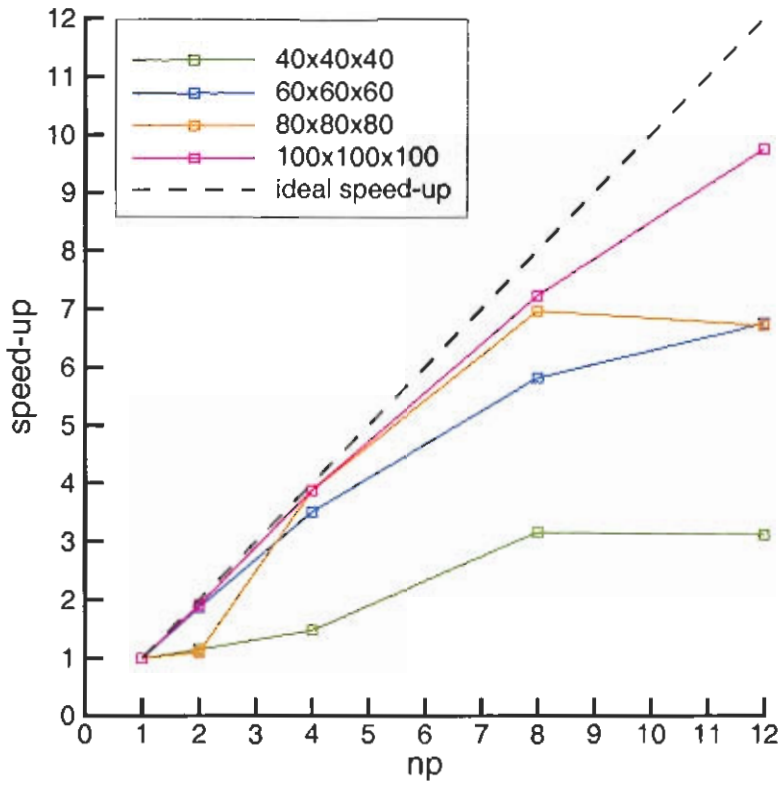

Figure 5.9: Speed-up in PC cluster.

np	partition	40 × 40 × 40	60 × 60 × 60	80 × 80 × 80	100 × 100 × 100
1	1x1y1z	1.00	1.00	1.00	1.00
2	2x1y1z	1.16	1.88	1.10	1.92
4	2x2y1z	1.49	3.51	3.87	3.88
8	2x2y2z	3.16	5.82	6.96	7.22
12	3x3y2z	3.12	6.75	6.71	9.74

Table 5.2: Speed-up in PC cluster.

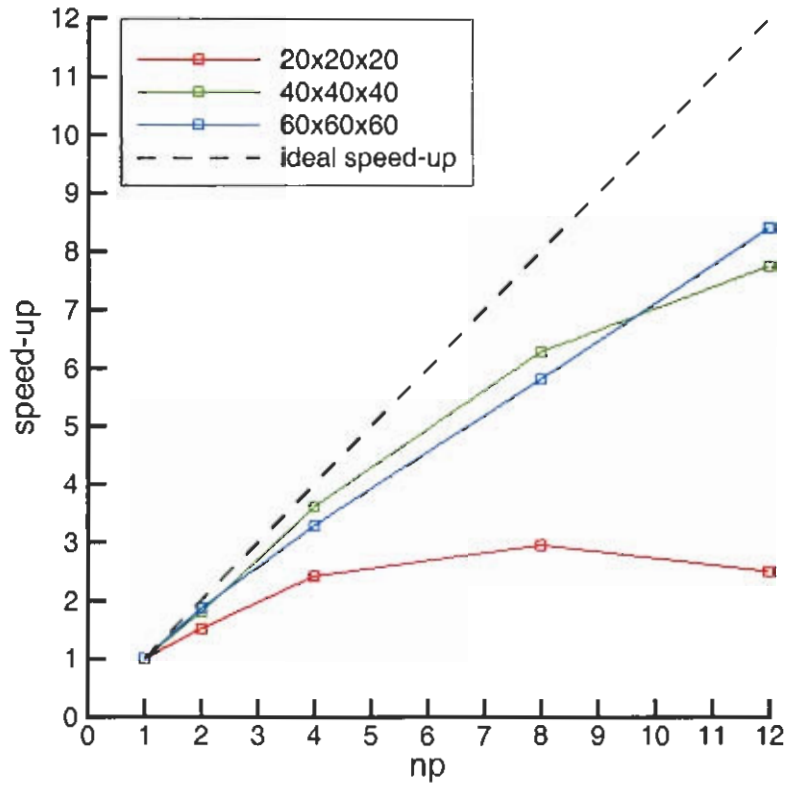

Figure 5.10: Speed-up in the Cray T3E

np	partition	$20 \times 20 \times 20$	$40 \times 40 \times 40$	$60 \times 60 \times 60$
1	1x1y1z	1.00	1.00	1.00
2	2x1y1z	1.51	1.79	1.86
4	2x2y1z	2.41	3.59	3.27
8	2x2y2z	2.95	6.27	5.80
12	3x3y2z	2.50	7.73	8.38

Table 5.3: Speed-up in the Cray T3E.

From the results presented above, we state that the performance at both machines is similar and the configurations of 8 and 12 processors have reported good speed-ups of 7 and 10 respectively when the grid size of the problem increase. This behaviour is due to the two factors mentioned in previous chapter. The minimum grid size per processor constrains the efficiency of the parallel computation to large grid sizes. Conversely, given a problem with a fixed grid size, there is a loose of efficiency when increasing the number

of processors in one direction. This fact is due to the degradation of the solver and more precisely to the degradation of the incomplete factorizations used within the solver or within the preconditioner.

This loose of efficiency is much more significative within the solution of the continuity equation than the solution of the Navier-Stokes equations. In order to show this, an analysis of the timings of each step of the SIMPLE-like algorithm are provided. The computation and the communication times have been expressed in terms of the percentages respect to the overall wall clock time of the simulation.

The two bar-graphs (see Figs. 5.11, 5.12), one for PC cluster and the other for the Cray T3E, show this percentages from one of the examples with a grid size of $80 \times 80 \times 80$ and executed with 8 processors with a partitioning configuration of $2x \times 2y \times 2z$.

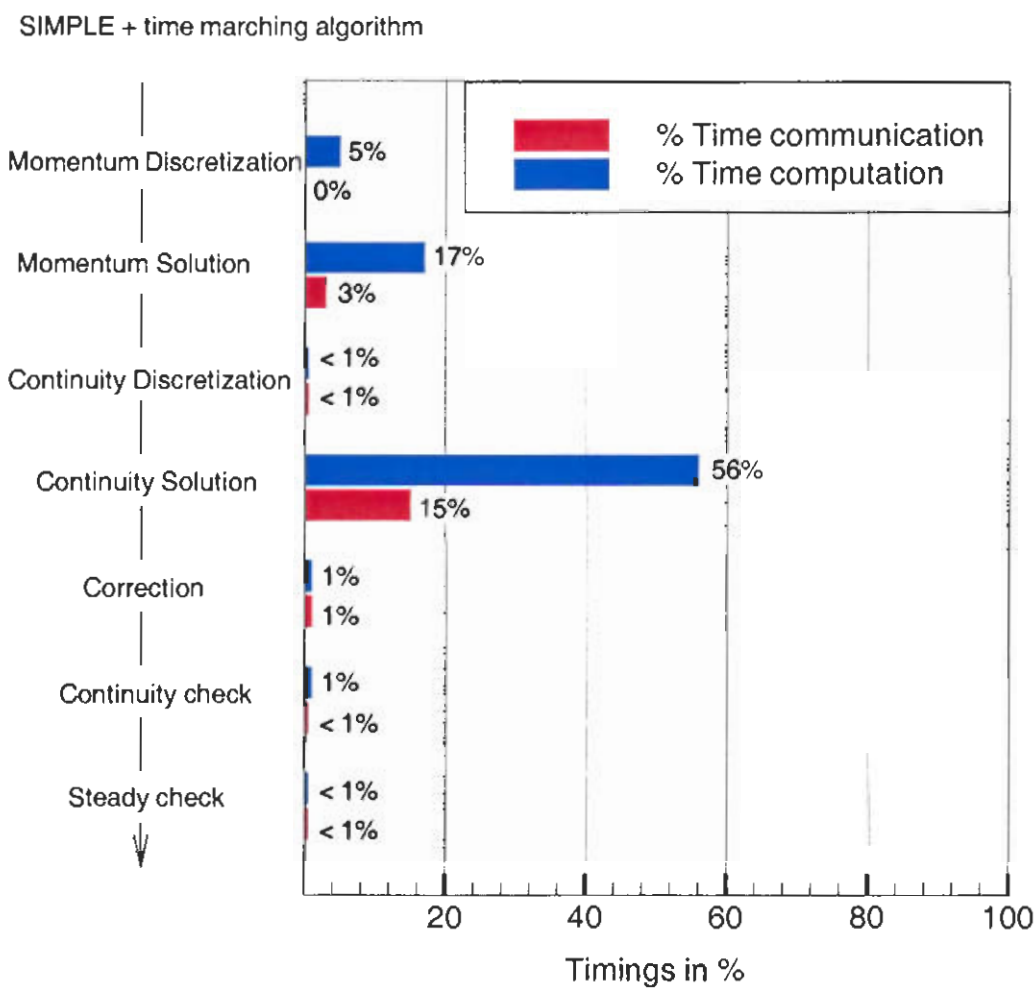

Figure 5.11: Timings in % of SIMPLE at PC cluster

At glance, there are no significative differences between both machines. The results show that roughly the $60-70\%$ from the overall wall clock time of the simulation is carried out at the solution of the continuity equation, meanwhile the remaining time is distributed among the solution of the Navier-Stokes equations, the correction of fluid flow variables, and in less percentage, in the continuity and the steady checkings.

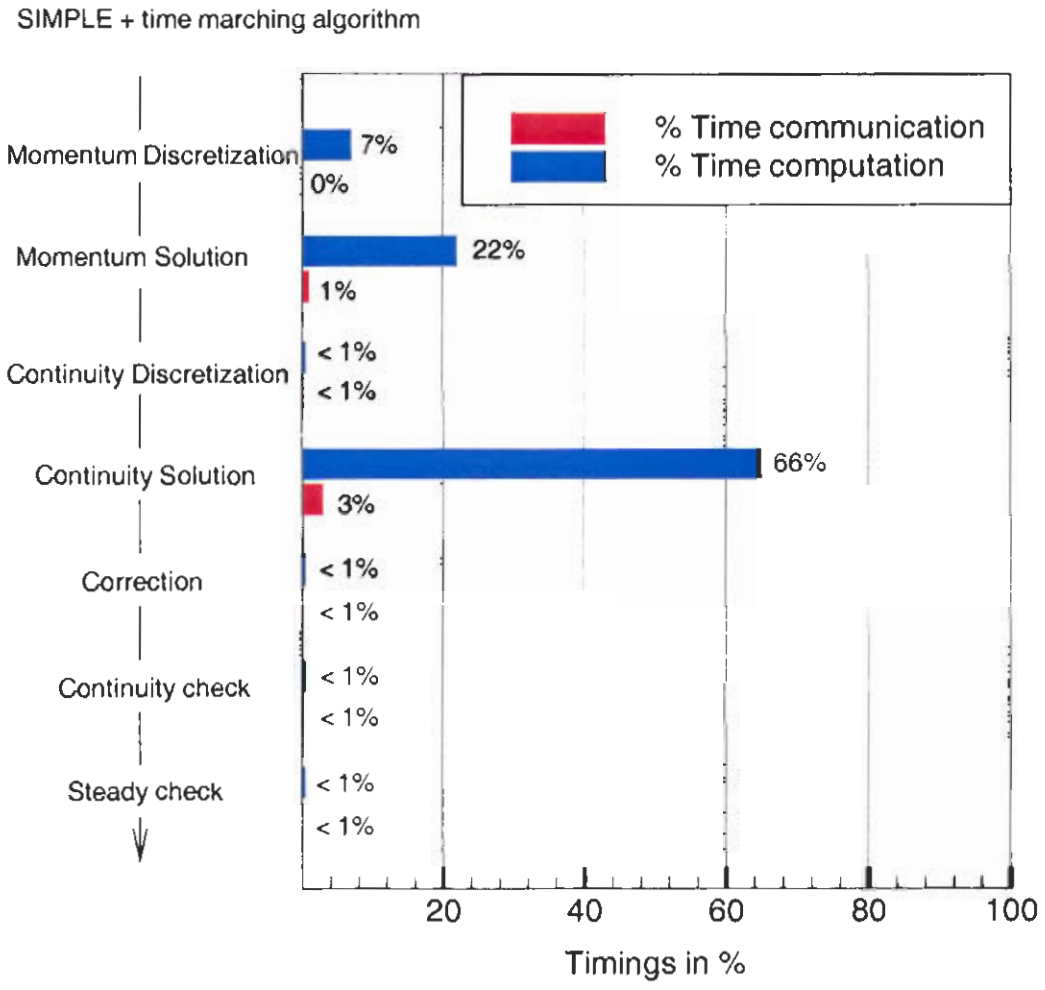

Figure 5.12: Timings in % of SIMPLE in the Cray T3E

This $60-70\%$ of time is composed by a percentage of time of computation and a percentage of time of communication that differ from one to the other machine. On PC cluster, the 56% of time is taken in the computation and the 15% is taken in the communication. On the Cray T3E, the 66% of time is taken in the computation and only the 3% of the time is taken in the communication. Clearly, this is the main difference between both machines, i.e., the ratio of the time of computation with the time of communication.

From the optimization point of view of the algorithm, it is remarkable that a reduction of the times of computation and communication of the solution of the continuity equation would reduce globally the overall wall clock time of the simulation.

Therefore, an analysis more detailed of the operations involved within the BiCGSTAB preconditioned with MSIP enable us to evaluate the bottlenecks that affect to the overall time.

The two bar-graphs 5.13,5.14 show the percentages of time of computation and communication of each algebraic operation involved within the solver respect to the wall clock time of the solver, i.e., the 60 – 70% of the overall time of the simulation.

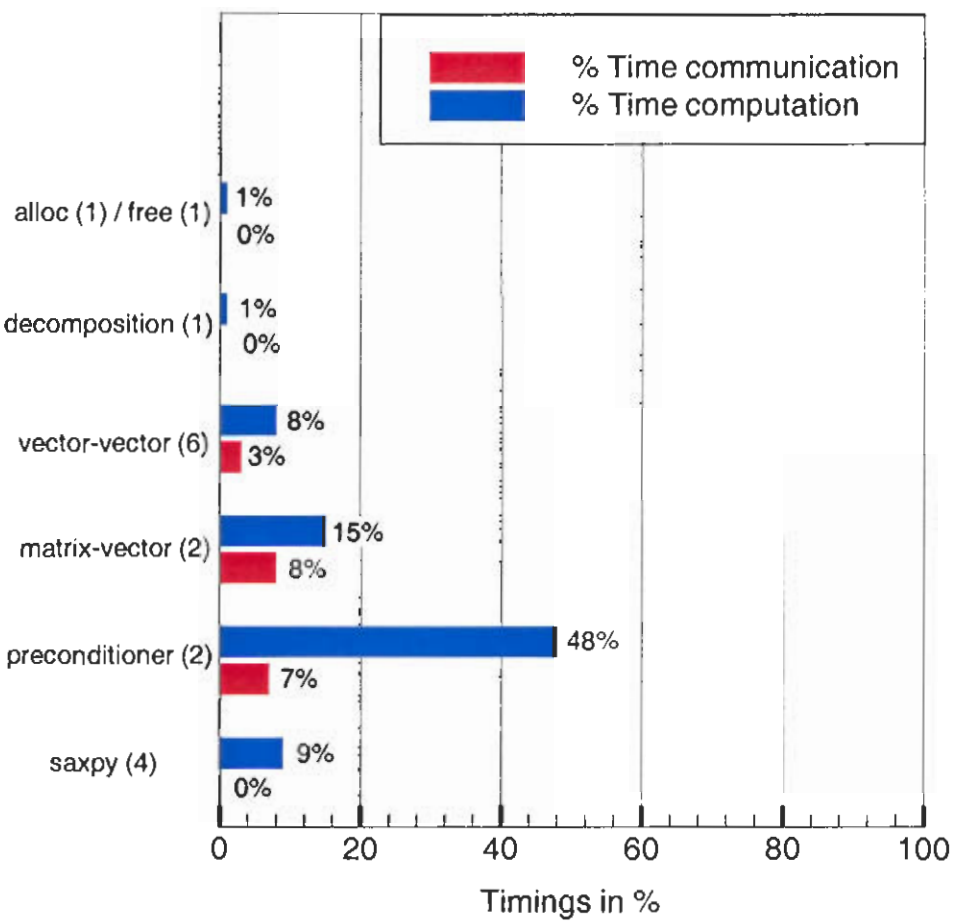

Figure 5.13: Timings in % of BiCGSTAB at PC cluster

Both graphs show analogous percentages for each algebraic operation. Clearly, the largest percentage is related with the preconditioner. The 60% of the time is taken within

the preconditioner, the 20% in matrix-vector product operations and the 10% of time in the vector-vector product. All of them require communications. It is worth noting that the time of computation of the incomplete decomposition is small compared to the above timings.

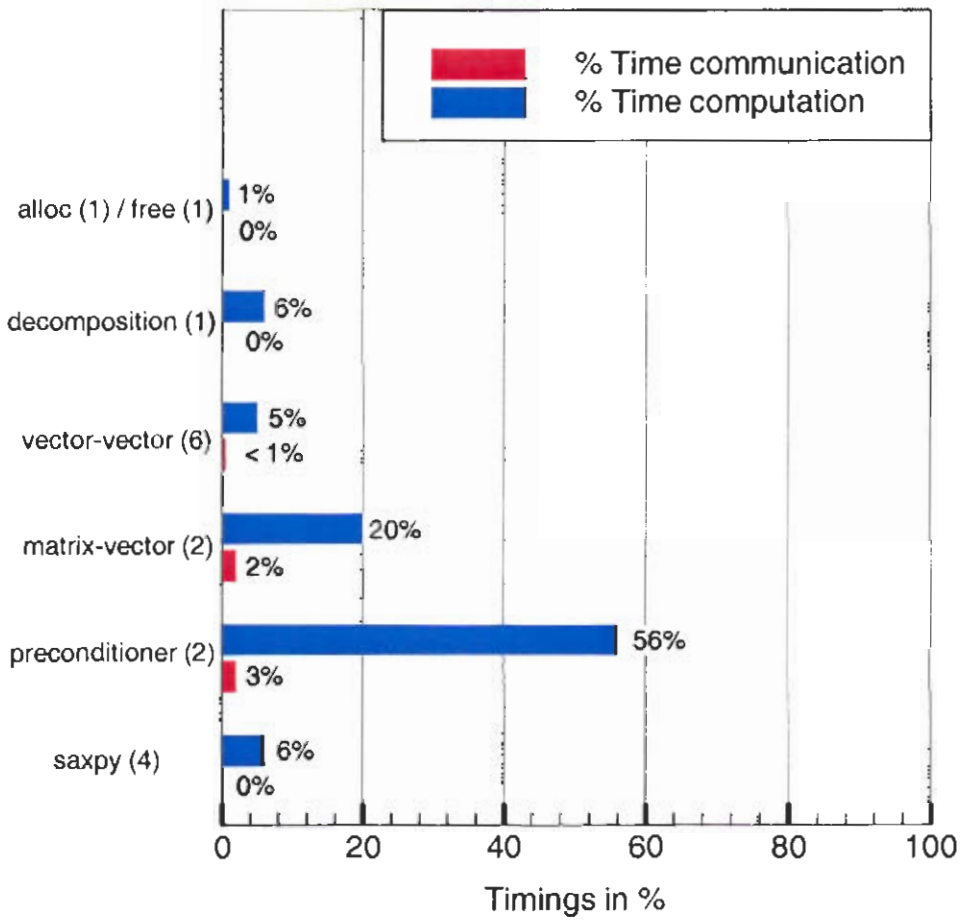

Figure 5.14: Timings in % of BiCGSTAB at Cray T3E

Therefore, the bottleneck arises in the preconditioner that not only it takes the longest time of computation and communication but also, since its algorithm is inherently sequential, there is a degradation of the efficiency when the number of processors is increased at each direction.

5.4 Other alternatives not based on Krylov solvers

As seen in previous section, the time of computation needed for the solution of the continuity equation is critical for the overall time of computation of the simulation. Research efforts in parallel preconditioners such as SPAI and polynomial preconditioners (see chapter 3, preconditioners section) are contributing to the improvement of the performance of Krylov solvers.

Other alternatives not based on Krylov solvers are the Schur Complement (SC) and the algebraic multigrid (ACM). In this work, BiCGSTAB has been compared with DDACM (Domain Decomposed Additive Correction Multigrid) described in [39] and SC described in [24, 62].

- SC is a direct parallel method, based on the use of non-overlapping subdomains with implicit treatment of interface conditions. Its main feature is that each processor has to solve only twice its own subdomain plus an interface problem in order to obtain the exact solution of the subdomain, with just one global communication operation.
- DDACM is a multigrid algorithm based on ACM, with both pre and post smoothing for the finest levels and the Schur Complement solver for the coarsest level. Although ACM is formulated with a recursive formulation, a non-recursive formulation is used for better control of the communications of DDACM.

5.4.1 Results of the alternatives

Since the available implementations of SC and DDACM are two dimensional and the implementation of BiCGSTAB is only three dimensional, only a qualitative comparison is possible. The extension of SC and DDACM to three dimensions is not straightforward.

The execution times compared are:

- BiCGSTAB: 3D lid driven cavity problem (described in this chapter), ϵ=10^{-4}, np=8, solved with a PC cluster described in the Appendix.
- DDACM: 2D model problem (described in [39]), ϵ=10^{-6}, np=9, solved with a 700 MHz PC cluster with a fastethernet (100Mb/sec) network.
- SC: 2D natural convection problem (described in [63]), $\epsilon \approx 10^{-11}$, np=8, solved with a PC cluster described in the Appendix.

The results of the times of computation are shown in Fig. 5.15.

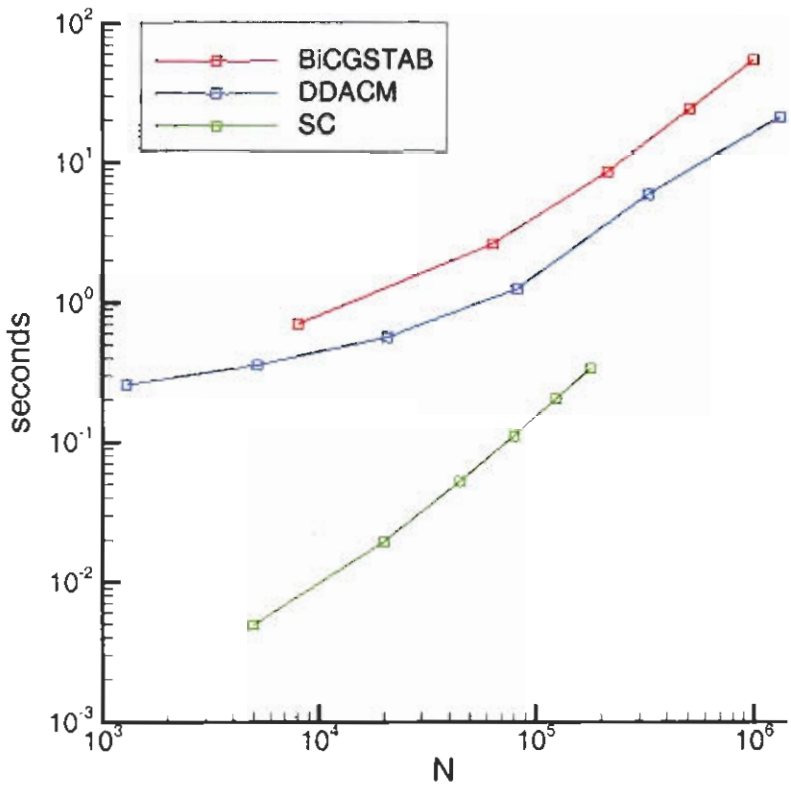

Figure 5.15: Times of computation of the continuity equation solved with the BiCGSTAB, SC and DDACM. N is the problem size.

Computation times of SC and DDACM are notably smaller than the BiCGSTAB ones. A part of this difference might be due to the differences between the problems solved. Additionally, in order to compare the different solvers, other factors have to be considered, such as preprocess time, storage requirements and algorithmic degradation with the number of processors *np*. The differences of these factors for the SC and BiCGSTAB are outlined below:

- With respect to the preprocess time, the SC takes a preprocess much higher than the solution time (on the order of hours), and hence, this solver is only feasible for constant coefficient matrices, i.e. for incompressible fluid flow problems. Therefore, SC is not applicable to compressible fluid flow problems. However, the BiCGSTAB takes a meaningless preprocess so it is applicable to both incompressible and compressible fluid flow problems, where the coefficients of the continuity equation change at each iteration.

- With respect to the storage, the requirements of memory are not a limiting factor for the BiCGSTAB, and hence, it enables the solution of large three dimensional problems. For the SC, the requirements of memory are large in two dimensional problems and even much more in three dimensional problems. See [62] for an estimation of the memory requirements with the problem size N and number of processors np.
- With respect to the algorithmic degradation with np, the preconditioner of the BiCGSTAB based on the incomplete factorization degrades with np. For the SC, the efficiency is kept for any np (but at the expense of an increase of memory).

With respect to DDACM, the preprocess, storage requirements and efficiency of DDACM are strongly dependent on the number of levels. More levels implies a reduction of both preprocess time and storage. Therefore, this reduction enables the application of DDACM to incompressible three dimensional fluid flow problems. However, the solver becomes more iterative and the number of communications per processor increases. Moreover, the high latency in networks of PC clusters reduce the efficiency with the number of levels.

5.5 Nomenclature

A	discretization matrix
a	coeff. in A
b	discretization right hand side
np	number of processors
P	pressure
Re	Reynolds number
r	residual
T	temperature
$\vec{V}$	fluid flow velocity vector
$\{u, v, w\}$	fluid flow velocity components
$\{x, y, z\}$	cartesian coordinates

Greek symbols

ϵ	precission
τ	time

Subscripts

NGB	general neighbour grid point
P	central grid point under consideration

Superscripts

(k)	k-th iteration of
τ	new value at time $\tau + \Delta\tau$ of
$*$	guessed value of
$'$	correction value of

Chapter 6

Conclusions

The detailed simulation of complex heat and mass transfer phenomena (i.e. detailed CFD simulation) requires a fine discretization of the governing equations (conservation of mass, momentum and energy) in both time and space. This leads to a set of strongly coupled non linear systems of equations with a large number of unknowns per system. Although the use of high order schemes is intended for the reduction of the size of these systems without loosing the accuracy of results, the number of unknowns per system is over several millions. E.g., a detailed three dimensional simulation of an incompressible and isothermal laminar flow within a $200 \times 200 \times 200$ grid points, leads to 4 coupled systems of equations (three velocity components $\{u, v, w\}$ and pressure P) with 8 millions of unknowns per system.

Dealing with such problems means computing the solution of these systems by combining the different numerical algorithms with the current hardware and software within a reasonable time. Hence, we have focussed our attention on solvers, parallelization by domain decomposition and computation in distributed memory machines (for instance, in clusters of personal computers).

It has been done, with respect to the solvers, a revision of those iterative methods for non symmetric matrices (hence, the symmetric matrices are considered a particular case) suitable for CFD problems: the incomplete factorizations or ILUs such as MSIP and SIS, Krylov space based solvers such as BiCGSTAB and GMRESR with their preconditioners and accelerating techniques such as the algebraic multigrid (AMG) and multiresolution analysis (MRA) with wavelets. These solvers exploit the matrix sparcity of these systems and reduce the storage in memory. Hence, GMRESR and BiCGSTAB have also been preconditioned with ILUs.

Furthermore, each of them may be used as black box and even combined among them because the operations are purely algebraic and there are no dependencies with the geometry of the problem, boundary conditions (Dirichlet, Neumann and periodic) and phenomena (the discretization of the advection and diffusion terms leads to non symmetric and symmetric coefficients within the matrices respectively).

This revision is completed with a comparative study of both the computing time in sequential and memory storage for a two dimensional problem with few thousands of unknowns. For different boundary conditions and for a wide range of convective with diffusive ratios the results show, with independence of the case, that the acceleration with either AMG or MRA improves the convergence rates but at expense of an slightly increase in

memory storage. However, these techniques are unable to bear problems with several millions of unknowns in single processor machines since their processing speed (i.e. number of floating point operations per second) and storage capacity are limited.

To overcome these difficulties, the path of the parallel computation in distributed memory multiprocessors, and for instance, in PC clusters has been adopted. The data exchange among processors is performed by means of message passing software such as MPI. By comparison with conventional distributed memory supercomputers, PC clusters have powerful processors at lower cost and larger scalabilities that compensate the slower communication throughputs (i.e. high latency and low bandwidth).

A comparative study, from the parallel performance point of view, carried out on two different machine architectures, the Cray T3E and a PC cluster (see appendix), has strengthened this position. The evaluation of the parameters affecting to this performance has shown, for both machines, similar processor performances but different network througputs, being at Cray T3E 10 times faster than PC cluster.

This implies, from the design and implementation points of view of the parallel algorithms that the computational load per processor must be large and balanced, and the communication among processors must be minimized (e.g. nonblocking point to point communications), thus reducing the penalty in the performance. Hence, larger computation with communication ratios will provide better performances, i.e. better speed-ups.

For this reason, given the same computation load (number of flops) and communication load (number of communications and data to be transferred) for both computers, the performance is greater at computers with faster networks, and for instance, at Cray T3E. This fact oblige to work at PC clusters with larger problems per processor in order to keep a similar performance. E.g. in order to have a minimum efficiency of 50% in a parallel matrix-vector product, having the matrix a 7-point stencil, the estimation of the minimum size per processor is $20 \times 20 \times 20$ at Cray T3E and $30 \times 30 \times 30$ at PC cluster.

Indeed, this strategy is concerned with the domain decomposition of the problem and the application of the numerical algorithms at each subdomain. Thus, the discretization of the governing equations over the decomposed domain leads to a set of coupled systems of equations composed by matrices and vectors distributed among processors in block matrix and block vector structures respectively.

Krylov space based solvers are suitable algorithms for the parallel solution of these systems since the involved algebraic operations (matrix-vector and vector-vector products) may be computed efficiently in parallel. However, these solvers need a preconditioner for better convergence behaviour. In this work, it has been used an incomplete factorization wich exploits the sparsity of block matrices. The drawback of the preconditioned solvers is the loss of efficiency or degradation of such preconditioners when the number of processors is increased. This is due to the evaluation whithout communications of local factorizations (at each subdomain), and hence, losing global convergence rate.

In this sense, a numerical experiment has been performed by partitioning a large three dimensional model problem (i.e. a large linear system of equations) into several directions. The results have pointed out, with independence of the computer, the degradation of the parallel algorithms with the number of processors and their limited performance. The best results have been obtained with BiCGSTAB preconditioned with MSIP.

Finally, these techniques have been applied to the solution of large coupled systems of

equations derived from a well known CFD problem: the three dimensional isothermal lid driven cavity flow for laminar Reynolds numbers. The detailed simulation with a $120 \times 120 \times 120$ grid has been carried out at PC cluster. It has been successfully benchmarked with both experimental and numerical simulation (in single processor) data provided by other authors. By using MSIP for the momentum equations and BiCGSTAB preconditioned with MSIP for the continuity equation, the simulation has been computed for several processors (from 1 up to 12 processors) and partitioning configurations.

The most significative results are speed-ups of 7 with 8 processors and 10 with 12 processors. Slightly better speed-ups have been obtained at Cray T3E with differencies of 15% at most of the overall time of simulation.

From the timing analysis of SIMPLE algorithm it is shown, as expected, that the most time consuming part (roughly the 60-70%) is wasted in the solution of the continuity equation. Therefore, any improvement on numerical algorithms aimed to the computing time reduction in the solution of the continuity equation, will represent a notably reduction in the overall time of simulation.

In this sense, alternative methods such as Algebraic Multigrid and Schur Complement have been qualitatively compared with BiCGSTAB. The comparison has shown that these alternatives provide better speed-ups. However, they present several constraints such as the application only to incompressible fluid flow problems, the need of a large preprocess step and the requirement of a large storage capacity.

Hence, further improvements may be stressed in several in combinations of the above mentioned methods. Following this working line, the research should be focused on efficient parallel preconditioners with the number of processors, and following the cited alternatives, on the reduction of the preprocess step and storage requirements.