Abstract:

The purpose of this work is to study the solution of a 2-D compressible viscous fluid flow in a nozzle on a Hypercube.

The analysis of the physical problem on a serial computer and an outline of its parallelization on the hypercube were detailed in [10].

We propose now in this paper to detail all the steps of the parallelization and our results: speedups, efficiencies and communication time versus arithmetic time.

Implicit Finite-Difference Simulation of an Internal Flow on a Hypercube

Pierre Porta

Research Report YALEU/DCS/RR-594
January 1988

This work was supported by ONR grant N00014-86-K-0310

1. Introduction

The purpose of this work is to study the solution of a 2-D compressible fluid flow on a Hypercube, and to see the improvement in the computational time got with such a distributed memory message-passing machine.

In the realization of this project, the chosen methodology was first to solve the physical problem by the implementation of a serial code, then to study the parallelization of this code on a Hypercube [10].

The solution of the fluid flow in a convergent-divergent nozzle is a problem wich has been studied by several authors on sequential [3, 4, 10, 12, 20] and vector [19] computers and we propose now to study the parallelization of this problem on a machine such as the iPSC Intel Hypercube.

A general technique of solution of the Navier-Stokes equations [17, 18] mainly based on the combination of a general curvilinear coordinate transformation and an implicit method has given some good results and is used here.

More precisely, the Navier-Stokes equations are solved with the Beam-Warming implicit me - thod [1], which leads to the computation of two sets of linear systems alternatively solved (ADI method).

The implementation of this numerical simulation on the Hypercube is based on a block method that is to say that the computational domain is decomposed in strips for both the x and y-directions. Two differents schemes of communications among the processors are implemented and gave some good results concerning the efficiency of the parallel code.

In the first chapters, we introduce the physical problem which lead to the computation of a serial code, then we analyze and discuss the results of the code implemented on the Hypercube.

2. The model equations

2.1. Transformed Navier-Stokes equations

The motion inside the nozzle (see figure 1) is governed by the unsteady averaged Navier-Stokes equations in the inertial cartesian coordinates (x,y,t). If we take for the dependent variables, the cartesian velocity components, these basic set of equations can be transformed to a new body-fitted curvilinear coordinate system (ξ, η, τ), while retaining the strong conservation law form [9, 17, 18, 20, 22]. The space (ξ, η, τ) called computational space corresponds to a rectangular domain [21]. Due to the geometrical properties of the nozzle, we can restrain the transformation (ξ, η, τ) to [19]: $(\xi = \xi(x), \eta = \eta(x, y), \tau = t)$.

The resulting equations, written in the non-dimensional form and in the strong conservation law form, are given by [9, 17, 18]:

$$\partial_\tau \hat{q} + \partial_\xi \hat{e} + \partial_\eta \hat{f} = Re^{-1}[\partial_\xi(J^{-1}(\xi_x g_1)) + \partial_\eta(J^{-1}(\eta_x g_1 + \eta_y g_2))] \tag{1}$$

where:

$$\hat{q} = J^{-1}\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ en \end{pmatrix}, \hat{e} = J^{-1}\begin{pmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U \\ (en+p)U \end{pmatrix}, \hat{f} = J^{-1}\begin{pmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ (en+p)V \end{pmatrix}$$

$$g_1 = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ g_{14} \end{pmatrix}, g_2 = \begin{pmatrix} 0 \\ \tau_{yx} \\ \tau_{yy} \\ g_{24} \end{pmatrix}.$$

with:

$$\begin{cases} \tau_{xx} = (\lambda + 2\mu)u_x + \lambda v_y \\ \tau_{xy} = \tau_{yx} = \mu(u_y + v_x) \\ \tau_{yy} = (\lambda + 2\mu)v_y + \lambda u_x \\ g_{14} = u\tau_{xx} + v\tau_{xy} + kPr^{-1}(\gamma-1)^{-1}\partial_x a^2 \\ g_{24} = u\tau_{xy} + v\tau_{yy} + kPr^{-1}(\gamma-1)^{-1}\partial_y a^2 \end{cases}$$

and:

$$U = \xi_x u \ , V = \eta_x u + \eta_y v \tag{2}$$

The velocities U and V are called the contravariant velocity components, and correspond to the decomposition of the vector velocity along the ξ and η curvilinear coordinates.
J is the transformation Jacobian:

$$J = \xi_x \eta_y = 1/(x_\xi y_\eta) \tag{3}$$

The cartesian derivatives such as u_x are expanded in the (ξ, η) space via chain-rule relation:

$$u_x = \xi_x u_\xi + \eta_x u_\eta \qquad (4)$$

And, the metrics ξ_x, η_x, η_y formed from chain-rule extension of x_ξ, y_ξ, y_η are given by the following relations:

$$\xi_x = J y_\eta \ , \ \eta_x = -J y_\xi \ , \ \eta_y = J x_\xi \qquad (5)$$

and:

ρ: density.
u, v: cartesian velocity components.
en: total energy.
p: pressure.
a: sound speed ($a = \sqrt{\gamma R T}$).
T: temperature.
γ: ratio of specific heats (dry air: $\gamma = 1.4$).
μ: dynamic viscosity.
λ: taken with the Stokes hypothesis $\lambda = -(2/3)\mu$. (6)
k: coefficient of thermal conductivity.
R: gas constant.
U: modulus of the velocity ($U = \sqrt{u^2 + v^2}$).
D: specific length.
c_p: specific heat.
Re: Reynolds number ($Re = (UD\rho)/\mu$).
Pr: Prandtl number ($Pr = (\mu c_p / k)$.
M: Mach number ($M = U/a$).

The equations (1) are completed by the equation defining the pressure:

$$p = (\gamma - 1)[en - 0.5\rho(u^2 + v^2)] \qquad (7)$$

As our nozzle is symmetric about the central axis y=0, we can restrict our study to the upper side of the nozzle.

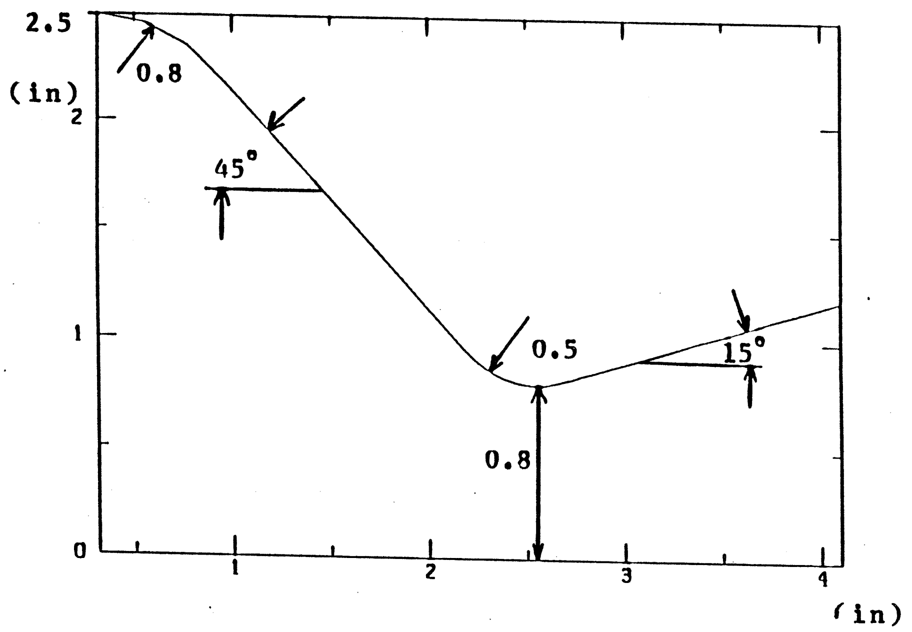

Figure 1: $45^0 - 15^0$ Convergent-Divergent Nozzle

2.2. Simplified form: the parabolic equations

Classically for high Reynolds number flows, we solve the viscous terms only near the rigid boundaries. We also make the hypothesis of the parabolic approximation that the viscous terms in ξ (along the body) are neglected and only the viscous terms in η are retained.
The equations (1) are then equal to [9, 17, 20]:

$$\partial_\tau \hat{q} + \partial_\xi \hat{e} + \partial_\eta \hat{f} = Re^{-1} \partial_\eta \hat{g} \tag{8}$$

where:

$$\hat{g} = J^{-1} \begin{pmatrix} 0 \\ \mu(\eta_x^2 + \eta_y^2) u_\eta + (\mu/3)\eta_x(\eta_x u_\eta + \eta_y v_\eta) \\ \mu(\eta_x^2 + \eta_y^2) v_\eta + (\mu/3)\eta_y(\eta_x u_\eta + \eta_y v_\eta) \\ [kPr^{-1}(\gamma-1)^{-1}(\eta_x^2+\eta_y^2)\partial_\eta a^2 + \mu(\eta_x^2+\eta_y^2)(u^2+v^2)_\eta/2 \\ +\mu/6(\eta_x^2 u_\eta^2 + \eta_y^2 v_\eta^2 + 2\eta_x\eta_y(uv)_\eta)] \end{pmatrix} \tag{9}$$

Note that unlike boundary-layer theory, the pressure p can vary through the viscous layer, and all the inertial terms of the normal momentum equation are retained.

3. Boundary and Initial conditions

3.1. Boundary conditions

The equations (8) have to be solved in the computational domain subject to different conditions on the boundaries.

The curvilinear coordinate system (ξ,η) is defined such that the inflow and outflow boundaries are two ξ-coordinate lines, and the wall and the symmetric axis are two η-coordinate lines.

At the nozzle wall, we have for the velocity the no-slip condition: $u = v = 0$ and, for the temperature we can take the adiabatic condition: $\vec{n} \cdot \nabla T = 0$, with $\vec{n}$ the vector normal to the surface.

Because of the symmetry of the nozzle, we consider the axis y=0 as one of the boundaries of this problem (see figure 1). In order to limit the overloading of the code, we just limit ourselves to a condition on the vertical component of the velocity v : $v = 0$ as it is an odd function of y.

At the upsteam inflow boundary, the pressure p and the temperature T are constant during all the computation, and are equal to their stagnation correspondants that is : $p = p_o$ and, $T = T_o$.

The components u and v of the velocity are given by the relation: $u = v \tan\theta(y)$ where $\theta(y)$ is a function given by Holder et al [5].

And finally, as the downstream outflow boundary corresponds to the case of a supersonic flow $(M > 1)$, no boundary conditions are required. The variables are determined by extrapolation from the interior points.

3.2. Initial condition

All the variables at the beginning of the computation were computed under the 1-D approximation method (isentropic flow, perfect fluid).
In such an approach, the values of the variables at a specified section of the nozzle are given by the geometric ratio of the aera of the section over the throat section, and by the chosen stagnation conditions [6, 16]. In particular, we have taken here: $p_o = 6.2\ 10^5 N/m^2$ and $T_o = 300K$.

The Reynolds number for this initial condition was then equal to: $Re = 1.5 \times 10^6$.

4. Numerical Method

We solve the equations (8) in the computational domain (ξ, η) subject to the boundary conditions described in section 3.1, with the implicit delta-form, approximate-factorization, Beam-Warming algorithm.

An implicit numerical method is employed here in order to avoid the severely restrictive stability conditions of an explicit method, when small grid spacings are used. Such a situation is needed near the wall for an accurate computation of the viscous effects.

In the basic Beam-Warming algorithm, the spatial derivative terms in the Navier-Stokes equations are approximated by standard second-order accurate central-difference operators, and the implicit θ-method of Richtmyer and Morton is taken for the time differencing [9].

The computation of the boundary points can be computed directly by the numerical resolution of the equations (see [20]) or by extrapolation from the interior points at the end of each time step (see [17]). The second case degrades the time accuracy on the boundary to a zero-order but gives a more simple scheme. We have chosen here the second case.

4.1. Time Differencing

The equations of Navier-Stokes in their final form (8) are rewritten here as:

$$\partial_\tau \hat{q} = -[\partial_\xi \hat{e} + \partial_\eta \hat{f} - Re^{-1}\partial_\eta \hat{g}] = \hat{r} \tag{10}$$

Using n for the time index and h for the time step, we apply the Richtmyer and Morton method, and we obtain [9]:

$$\hat{q}^{n+1} = \hat{q}^n + h[(1-\theta)\hat{r}^{n+1} + \theta \hat{r}_n] \tag{11}$$

This method is first-order accurate in time for $\theta = 0$ (Implicit Euler method), and is second-order accurate in time for $\theta = 1/2$.
Since we seek only the asymptotic steady state solution, we can employ the first-order accurate in time method. The accuracy of the solution is given by the spatial difference operators [1].
So we have:

$$\hat{q}^{n+1} = \hat{q}^n + h\hat{r}^{n+1} \tag{12}$$

In order to define the non-linear term $\hat{r}_{n+1}$, we must locally linearized the terms $\hat{e}$, $\hat{f}$ and $\hat{g}$ in terms of $\hat{q}$. This is done by using the Taylor series expansions:

$$\hat{e}^{n+1} = \hat{e}^n + E^n(\hat{q}^{n+1} - \hat{q}^n) + O(h^2). \tag{13.a}$$

$$\hat{f}^{n+1} = \hat{f}^n + F^n(\hat{q}^{n+1} - \hat{q}^n) + O(h^2). \tag{13.b}$$

$$\hat{g}^{n+1} = \hat{g}^n + G^n(\hat{q}^{n+1} - \hat{q}^n) + O(h^2). \tag{13.c}$$

where E, F and G are the flux Jacobian matrices:

$$E = \partial\hat{e}/\partial\hat{q},\ F = \partial\hat{f}/\partial\hat{q}\ ,\ G = \partial\hat{g}/\partial\hat{q} \tag{14}$$

The matrices E or F are given by [9, 17]:

$$E, F = \begin{pmatrix} 0 & K_1 & K_2 & 0 \\ K_1\phi^2 - u\theta & \theta - K_1(\gamma-2)u & K_2u - (\gamma-1)K_1v & K_1(\gamma-1) \\ K_2\phi^2 - v\theta & K_1v - K_2(\gamma-1)u & \theta - K_2(\gamma-2)v & K_2(\gamma-1) \\ \theta(2\phi^2 - \gamma(en/\rho)) & [K_1\beta - (\gamma-1)u\theta] & [K_2\beta - (\gamma-1)v\theta] & \gamma\theta \end{pmatrix} \tag{15}$$

where:

$$\phi^2 = 0.5(\gamma-1)(u^2+v^2)\ ,\ \beta = \gamma(en/\rho) - \phi^2 \text{ and, } \theta = K_1u + K_2v \tag{16}$$

with the following definition of the coefficients K_1 and K_2:

$$\text{for } E\text{: } K_1 = \xi_x,\ K_2 = 0 \text{ and, for } F\text{: } K_1 = \eta_x,\ K_2 = \eta_y \tag{17}$$

And for the viscous flux Jacobian term, we have [9, 17]:

$$G^n = J^{-1}\begin{pmatrix} 0 & 0 & 0 & 0 \\ g_{21} & \alpha_1\partial_\eta(1/\rho) & \alpha_2\partial_\eta(1/\rho) & 0 \\ g_{31} & \alpha_2\partial_\eta(1/\rho) & \alpha_3\partial_\eta(1/\rho) & 0 \\ g_{41} & g_{42} & g_{43} & \alpha_4\partial_\eta(1/\rho) \end{pmatrix} J \tag{18}$$

where:

$$\begin{pmatrix} g_{21} \\ g_{31} \end{pmatrix} = -Q \cdot \begin{pmatrix} \partial_\eta(u/\rho) \\ \partial_\eta(v/\rho) \end{pmatrix}, \text{ with } Q = \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_3 \end{pmatrix} \tag{19}$$

and:

$\alpha_1 = \mu[(4/3)\eta_x^2 + \eta_y^2]$

$\alpha_2 = (\mu/3)\eta_x\eta_y$

$\alpha_3 = \mu[\eta_x^2 + (4/3)\eta_y^2]$

$\alpha_4 = \gamma k Pr^{-1}(\eta_x^2 + \eta_y^2)$

$g_{41} = \alpha_1\delta_\eta(-u^2/\rho) + \alpha_2\delta_\eta(-2uv/\rho) + \alpha_3\delta_\eta(-v^2/\rho) + \alpha_4\delta_\eta(-en/\rho^2) + \alpha_4\delta_\eta[(u^2+v^2)/\rho]$

$g_{42} = -g_{21} - \alpha_4\delta_\eta(u/\rho)$

$g_{43} = -g_{31} - \alpha_4\delta_\eta(v/\rho)$

By means of the local linearizations the equations (12) are then rewritten:

$$[I + h(\partial_\xi E^n + \partial_\eta F^n - Re^{-1}\partial_\eta G^n)]\Delta\hat{q}^n = h\hat{r}^n \tag{20}$$

with:

$$\Delta\hat{q}^n = \hat{q}^{n+1} - \hat{q}^n \tag{21}$$

where the linear operator notation is to be interpreted as following:

$$(\partial_\xi E^n)\Delta\hat{q}^n = \partial_\xi(E^n\Delta\hat{q}^n) \tag{22}$$

The left hand side of equation (20) can be factorized into a product of two one-dimensional operators, with the same order of accuracy [1].
This results in the factorized form of equation (20):

$$([I + h\partial_\xi E^n][I + h(\partial_\eta F^n - Re^{-1}\partial_\eta G^n)])\Delta\hat{q}^n = h\hat{r}^n. \tag{23}$$

the right hand side of equation (20) is defined as:

$$\hat{r}^n = -[\partial_\xi \hat{e}^n + \partial_\eta \hat{f}^n - Re^{-1}\partial_\eta \hat{g}^n]. \tag{24}$$

Finally, the vector solution $\hat{q}^{n+1}$ is given by the following ADI sequence:

$$[I + h\partial_\xi E^n]\Delta\hat{q}^{*n} = h\hat{r}^n \tag{25.a}$$

$$[I + h(\partial_\eta F^n - Re^{-1}\partial_\eta G^n)]\Delta\hat{q}^n = \Delta\hat{q}^{*n} \tag{25.b}$$

$$\hat{q}^{n+1} = \hat{q}^n + \Delta\hat{q}^n \tag{25.c}$$

For convenience, we have omitted the spatial subscript notations, all the terms were taken here at the point (i, j) of the computational domain.
But from now on, we will use the spatial subscripts again.

4.2. Space Differencing

The resolution of the ADI sequence $(25.a, 25.b, 25.c)$ is completed by the choice of the finite difference operators δ for the spatial derivatives ∂_ξ and ∂_η.

We use here to approximate the convective derivatives, the standard central-difference second order accurate operator. For example, we have for the first derivative of equation $(25.a)$:

$$\delta_i E_{i,j} = (\partial E_{i,j}/\partial\xi) = (E_{i+1,j} - E_{i-1,j})/2\Delta\xi \tag{26}$$

From the general form of the viscous term written as follows:

$$\partial[(\alpha_{i,j}\partial\beta_{i,j}/\partial\eta)]/\partial\eta \tag{27}$$

we see that we have to approximate the viscous derivative in a more complicated way.

If, we define the following central-difference operator for a general viscous term h:

$$(\partial h_{i,j}/\partial\eta) \approx (h_{i,j+1/2} - h_{i,j-1/2})/\Delta\eta \tag{28}$$

its application on the viscous term defined in equation (27) gives the generalized three-points central-difference second-order accurate operator:

$$\begin{aligned}&\partial[(\alpha_{i,j}\partial\beta_{i,j}/\partial\eta)]/\partial\eta = \\ &1/(2\times\Delta\eta^2)\alpha_{i,j+1} + \alpha_{i,j})\beta_{i,j+1} - (\alpha_{i,j+1} + 2\alpha_{i,j} + \alpha_{i,j-1})\beta_{i,j} + (\alpha_{i,j} + \alpha_{i,j-1})\beta_{i,j-1}]\end{aligned} \tag{29}$$

corresponding to the following notation : $\delta_j'\hat{g}_{i,j}$ and $\delta_j'G_{i,j}$
From these definitions, the ADI sequence is rewritten as follows:

$$[I + h\delta_i E_{i,j}^n]\Delta\hat{q}_{i,j}^{*n} = h\hat{r}_{i,j}^n \tag{30.a}$$

$$[I + h(\delta_j F_{i,j}^n - Re^{-1}\delta_j' G_{i,j}^n)]\Delta\hat{q}_{i,j}^n = \Delta\hat{q}_{i,j}^{*n} \tag{30.b}$$

$$\hat{q}_{i,j}^{n+1} = \hat{q}_{i,j}^n + \Delta\hat{q}_{i,j}^n \tag{30.c}$$

with:

$$\hat{r}_{i,j}^n = -[\delta_i\hat{e}_{i,j}^n + \delta_j\hat{f}_{i,j}^n - Re^{-1}\delta_j'\hat{g}_{i,j}^n] \tag{31}$$

Note that by the choice of a central-difference scheme for the space derivatives, each step of the ADI sequence (30) involves the solution of a linear system of equations having a block-tridiagonal coefficient matrix.

5. Numerical results for the serial code

The results corresponding to the numerical simulation of the 2-D fluid flow in the nozzle described in figure 1 got on the VAX 8600 can be found in [10].

6. Parallelization of the serial code on the iPSC Intel

6.1. Introduction

We are dealing now with the parallelization of the serial code on the iPSC Intel. More precisely, we want to determine if such a parallel machine can handle a CFD problem and with what level of performance.

We want to point out that this 2-D numerical simulation of a fluid flow for a compressible viscous fluid made on the iPSC Intel can be viewed as a first step for the study of a more general flow of dimension 3 on a distributed memory parallel machine.

We have seen that the numerical algorithm based on the Beam - Warming Implicit method for the solution of the 2-D Navier-Stokes equations leads to an ADI sequence, which has already been studied both theorically and pratically on the hypercube (see [7, 15]).

First, we recall the ADI sequence (see equations (30)), that we can express here as follows:

$$A^n_{i,j}\Delta\hat{q}^{*n}_{i,j} = b^n_{i,j} \quad \text{for } j = 2, \dots, 31 \tag{32.a}$$

$$C^n_{i,j}\Delta\hat{q}^n_{i,j} = \Delta\hat{q}^{*n}_{i,j} \quad \text{for } i = 2, \dots, 63 \tag{32.b}$$

with the updated vector solution $\hat{q}^{n+1}_{i,j}$ given by:

$$\hat{q}^{n+1}_{i,j} = \hat{q}^n_{i,j} + \Delta\hat{q}^n_{i,j} \tag{32.c}$$

for the definition of $A^n_{i,j}$, $C^n_{i,j}$ of size 4×4 , and $b^n_{i,j}$ see chapter 4.

The purpose of this work is to study the parallelization of the specific ADI sequence given above (equations (32)) for our computational domain. The mesh is of size 64×32, where 64 points are used in the horizontal direction and 32 points in the vertical direction, which amounts to 2048 grids points, which in turn correspond to 8192 unknowns.

More precisely, we solve the 30 equations (32.a) (respectively 62 equations (32.b)) with i varying from 2 to 63 (respectively with j varying from 2 to 31) which lead to the computation of 30 linear systems of size 248 (62×4) (respectively 62 linear systems of size 120 (30×4)).

Pratically, we use a one dimensional domain decomposition embedded on the hypercube composed of l=2^d processors where d is the dimension of the hypercube.

6.2. Method

The work on the hypercube is essentially based on a block method approach that is to say that the computational domain is alternatively decomposed in horizontal and vertical strips.

The idea is to assigned to each processor a certain subset of contiguous rows and then columns, the size of the subset depending on the dimension of the cube.

The solution of the resulting linear systems corresponding to each processor can then be solve by using a sequential solver.

With this approach some data have to be shared between the processors, and for the hypercube, this is done by message passing. More precisely, for the solution of this problem we use two types of communications, a nearest neighbor communication scheme between two processors in order to form the linear systems (equations 32 a) and a global communication scheme which makes the transition from the horizontal decomposition to the vertical one and vice versa.

6.2.1. Horizontal decomposition

First, the computational domain is decomposed in l horizontal strips, where each strip is assigned to one processor, see figure below (l=4 i.e. d=2).

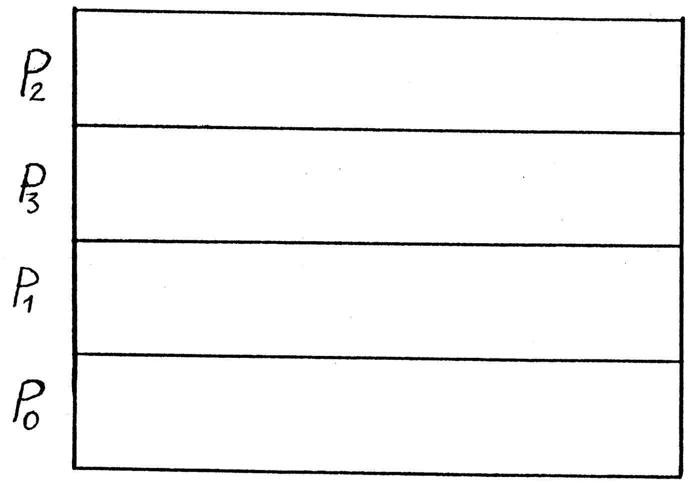

Figure 2: Horizontal decomposition for the 2-d cube.

By a step of nearest neighbor communication for $\hat{q}$ we can form the matrices A, and the r.h.s b for all the rows. This step is done by using a Gray code characterizing the different processors

and the different blocks of the cube (see [13, 14]).
The solution of the $m = 32/l$ linear systems of order 248 are then solved for each processor using locally a Linpack solver.

This step corresponds to the first half step of the ADI sequence (equations 32.a) i.e. for the x-direction of the ADI sequence.

6.2.2. Global communication of $\Delta\hat{q}^*$ and $\hat{q}$

After the solution of the first step of the ADI sequence, chunks of data have to be exchanged among the processors in order to assign a contiguous set of columns to each processor.

This step makes the transition between the horizontal and the vertical decompositions and can be done by different ways (see [8, 14]). I have experimented here two of them, the first called "linear communication" consits of $l - 1$ exchanges of blocks of data from each processor to all the others, and in collaboration with Faisal Saied a second one called "log communication", based on the representation of the different blocks by binary numbers following a Gray code scheme.

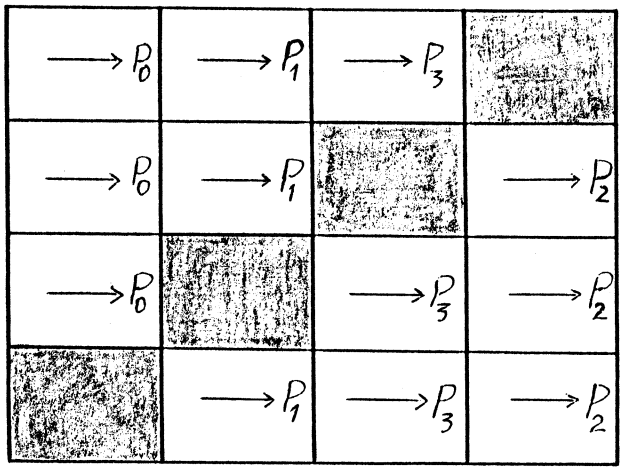

Figure 3: Blocks of data to be exchanged for the 2-d cube.

The arrows inside the blocks show the destination of the chunks.

For the second case, the global communication takes $\log_2(l)$ time steps, if l is the total number of processors of the hypercube.

This scheme is explained in [11] and is presented here in a synthetic way. At each time step of this global communication scheme each processor sends half of its data to one of its neighbors in the cube. The choice of the data sent and how to reorder the received data in each processor is function of the binary number of the corresponding block. This second method is clearly better than the first one in term of efficiency, but also more difficult to implement.

6.2.3. Vertical decomposition

The step of global communication leads to the decomposition of the computational domain in l vertical strips.

Without a new step of nearest neighbor communications, we can form the matrices $C_{i,j}$ for all the columns, and then we solve locally the $n = 64/l$ linear systems of order 120, corresponding to the second half step of the ADI sequence (equations (32.b)).

6.2.4. Update the vector solution

This step consists of updating the vector solution $\hat{q}$ for each section of each vertical strip (equations (32.c)) in each processor.

6.2.5. Global communications of $\hat{q}$

This step makes the transition between the vertical and the horizontal decomposition of the computational domain and is done with the two communications schemes seen before.

More exactly, the vector solution $\hat{q}$ is transposed back among the processors in order to begin a new time step of the ADI sequence.

6.3. Results

We have experimented our parallel code for cubes of dimensions from 1 up to 5, that is to say for 1 up to 32 processors.

Two of the main characteristics of a parallelized algorithm are its speedup and its efficiency (see figures 4 et 5). If T_1 is the execution time of the serial algorithm, and T_p, the execution time for the same type of algorithm with p processors, then the speedup is defined by $S_p = \frac{T_1}{T_p} \leq p$ and the efficiency by $E_p = \frac{T_1}{pT_P} \leq 1$. The algorithm time versus the communication time gives an idea of the proportion of the communication time in the whole computation time. In particular, we can see in the figures (6) and (7) that the linear scheme is better in terms of communication time when the number of processors is small, and the log scheme becomes better when the number of processors increases.

Otherwise, it appears that the code running with the linear communication scheme for the

global communication gives some good results. In particular, we get speedups of 1.96 for 2 processors and 22.12 for 32 processors, which correspond to the following efficiencies of 98 % for 2 processors and 69 % for 32 processors.
In the figure (6), we can see that for this first scheme, the communication time is an increasing function of l and is at most equal to 20 % of the arithmetic time for 32 processors.

As expected, the code running with the "log communication" scheme is better. The resulting speedups range from 1.96 for 2 processors to 25.22 for 32 processors which correspond for this last case to an efficiency of 79 %. In figure (7) the communication time increases with the number of processors up to $l = 8$ and then decreases, in particular we found that it decreases to 14.5 % of the arithmetic time for 32 processors.

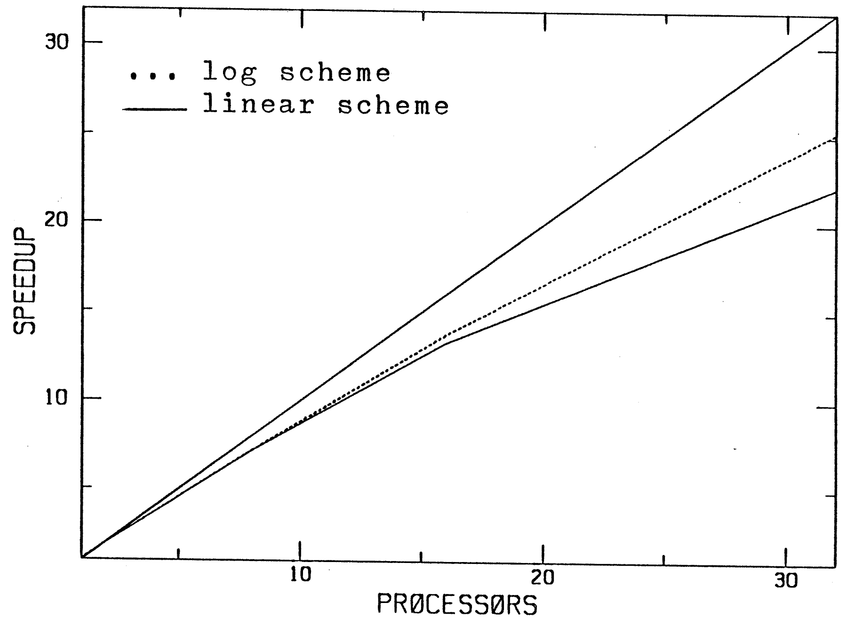

Figure 4: Speedup on the iPSC Intel

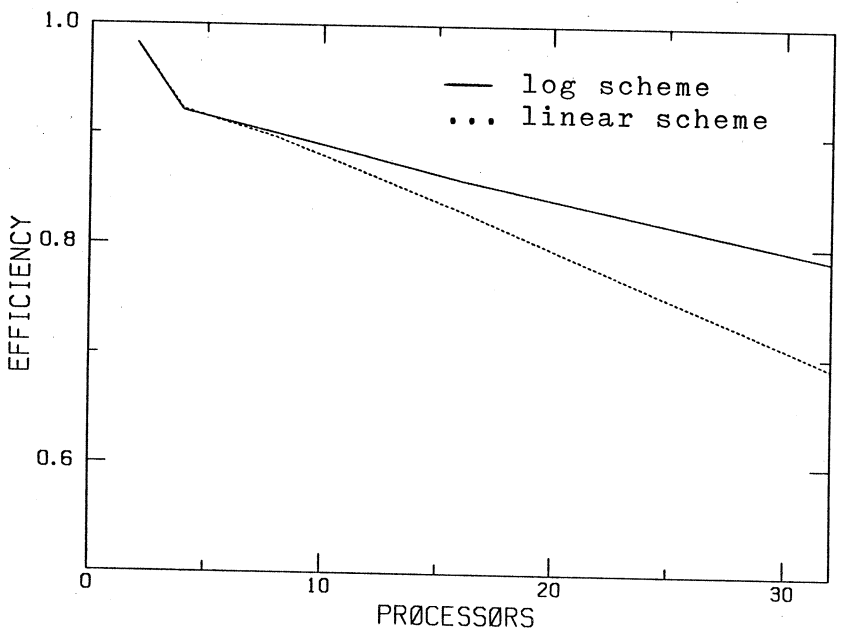

Figure 5: Efficiency on the iPSC Intel

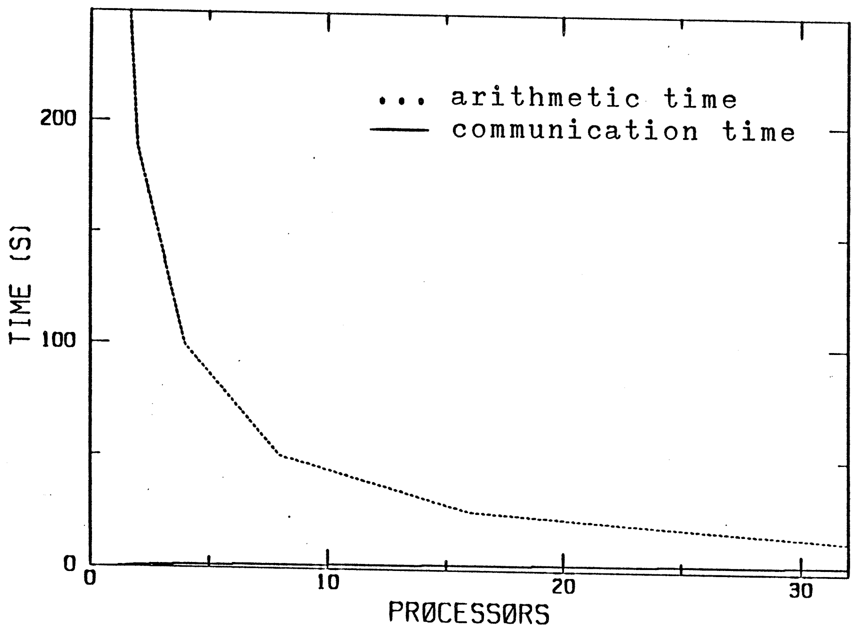

Figure 6: Arithmetic and communication times for the linear scheme

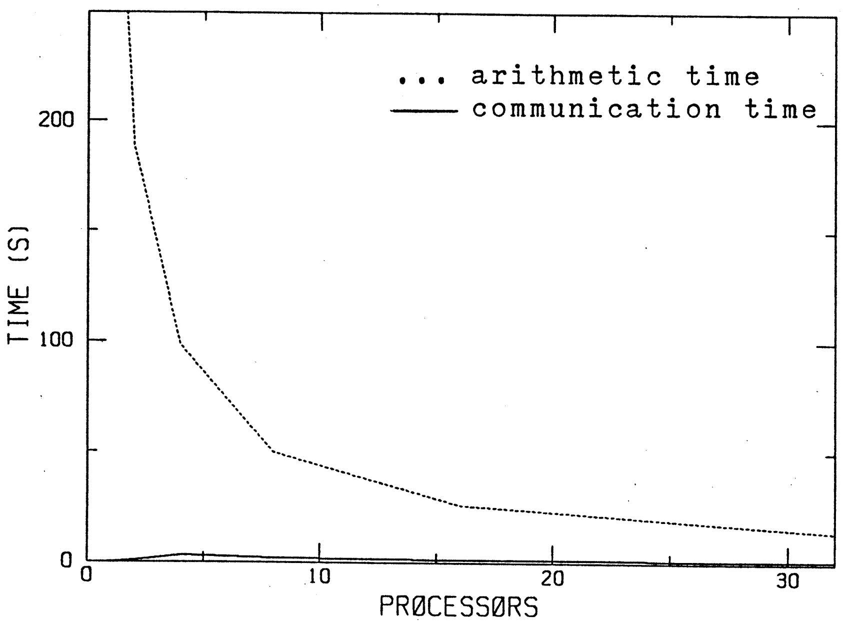

Figure 7: Arithmetic and communication times for the log scheme

7. Conclusions:

The good results got for the simulation of a 2-D fluid flow in a nozzle on the iPSC Intel Hypercube show that a CFD (Computational Fluid Dynamics) problem based on the solution of an ADI sequence can be easily adapted on a distributed memory parallel machine.

The main reason is that this problem can be divided in as many independant parts as needed, depending of the number of processors and this, without adding any computations. This point explains why we got such good speedups.

About the communications among the processors, we have seen that the "linear" algorithm gave some good results, and is easy to implement. The "log" scheme is far more difficult to implement but brings some improvements, especially when the number of processors increases.

The theorical analysis of the different running times and of the speedups got for this computation is actually done and will be detailed in a next paper.

Ackowledgement

I would like to thank Tiba for her help and Faisal for our useful discussions.

References

[1] R.M.Beam; R.F.Warming *An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation-Law Form.* Journal of Computational Physics, vol.22, 1976, pp 87-110.

[2] T.Cebeci; A.M.O.Smith; G.Mosinskis *Calculation of Compressible Adiabatic Turbulent Boundary Layers.* AIAA J, vol.8, no.11, 1970, pp 1974-1982.

[3] M.C.Cline *Computation of Steady Nozzle Flow by a Time-Dependent Method.* AIAA J, vol.12, no.4, 1974, pp 419-420.

[4] M.C.Cline *Computation of Two-Dimensional, Viscous Nozzle Flow.* AIAA J, vol.14 no.3, 1976, pp 295-296.

[5] D.W.Holder; D.C.Macphail; J.S.Thompson *Modern Developpement in Fluid Dynamics High Speed Flow.* vol II, Ed L.Howarth, Oxford University Press, 1953.

[6] B.W.Imrie *Compressible Fluid Flow.* A Halsted Press Book, ed. John Wiler & Sons, 1973.

[7] S.L.Johnsson; Y.Saad; M.H.Schultz *Alternating Direction Methods on Multiprocessors.* Technical Report YALEU/DCS/RR-382, Yale University, Dept. of Computer Science, 1985.

[8] S.L.Johnsson; Ching-Tien Ho *Spanning Graphs for Optimum Broadcasting and Personalized Communication in Hypercubes*, Technical Report YALEU/DCS/TR-500, Dept. of Computer Science, November 1986.

[9] H.Lomax; T.H.Pulliam *A Fully Implicit Factored Code for Computing Three-Dimensional Flows on the ILLIAC IV* in G.Rodrigue *Parallel Computations.* Academic Press, 1982 pp 217-250.

[10] P.Porta *Implicit Finite-Difference Simulation of an Internal Flow in a Nozzle; an Example of a Physical Application on a Hypercube*, Technical Report YALEU/DCS/RR-553, Yale University, Dept. of Computer Science, August 1987.

[11] P.Porta; F.Saied *Numerical Simulation of a 2-D Compressible Fluid Flow on a Hypercube*, Technical Report Yale University, Dept. of Computer Science, to appear.

[12] W.J.Rae *Some Numerical Results on Viscous Low-Density Nozzle Flows in the Slender-Channel Approximation.* AIAA J, 1971, vol.9, no.5, pp 811-820.

[13] Y.Saad; M.H.Schultz *Topological Properties of Hypercubes*, Technical Report YALEU/DCS/RR-389, Yale University, Dept. of Computer Science, May 1985.

[14] Y.Saad; M.H.Schultz *Data Communications in Parallel Architectures*, Technical Report YALEU/DCS/RR-461, Yale University, Dept. of Computer Science, March 1986.

[15] F.Saied; C.T.Ho; S.L.Johnsson; M.H.Schultz *Solving Schrödinger Equation on the Intel iPSC by the Alternating Direction Method*, in *Hypercube Multiprocessors 1987*, ed. Michael Heath, SIAM, 1987.

[16] A.H.Shapiro *The Dynamics and Thermodynamics of Compressible Fluid Flow.*
vol I, The Ronald Press Company, New York, 1953.

[17] J.L.Steger *Implicit Finite-Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries.* AIAA J, vol.16, no.7, 1978, pp 679-686.

[18] T.H.Pulliam;J.L.Steger *Implicit Finite-Difference Simulations of Three-Dimensional Compressible Flow.* AIAA J, vol.18, no.2, 1980, pp 159-167.

[19] J.C.Strikwerda *A Time-Split Difference Scheme for the Compressible Navier-Stokes Equations with Applications to Flows in Slotted Nozzles.* in G.Rodrigue *Parallel Computations.* Academic Press, 1982, pp 251-267.

[20] P.D.Thomas *Numerical Method for Predicting Flow Characteristics and Performance of Nonaxisymmetric Nozzles-Theory.* NASA CR-3147, 1979.

[21] J.F.Thompson; F.C.Thames; C.W.Mastin *Automatic Numerical Generation of Body-Fitted Curvilinear Coordinate System for Fields Containing any Number of Arbitrary Two Dimensional Bodies.* Journal of Computational Physics, vol.15, 1974, pp 299-319.

[22] M.Vinokur *Conservation Equations of Gasdynamics in Curvilinear Coordinate Systems.* Journal of Computational Physics, vol.14, 1974, pp 105-125.