Time Parallel Solution of Linear Partial Differential Equations on the Intel Touchstone Delta Supercomputer

Nikzad Toomarian
Amir Fijany
Jacob Barhen

Center for Space Microelctronics Technology
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr., MS 303-310
Pasadena, CA 91109

Contact Author:
Dr. Jacob Barhen
Tel: 818-354-9218
Fax: 818-393-5013
E-mail: Barhen@nips.jpl.nasa.gov

Submitted to:
Concurrency
July 1993

Time Parallel Solution of Linear Partial Differential Equations on the Intel Touchstone Delta Supercomputer

Nikzad Toomarian, Amir Fijany and Jacob Barhen

Center for Space Microelectronics Technology
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

Evolutionary partial differential equations are usually solved by discretization in time and space, and by applying a marching in time procedure to data and algorithms potentially parallelized in the spatial domain. In a departure from such a strictly sequential temporal paradigm, we have developed the concept of time parallel algorithms, which allow the marching in time to be fully parallelized. This is achieved by using a transformation based on the eigenvalue-eigenvector decomposition of the matrices resulting from the discretization process. Since these matrices are involved in the time stepping iterations, the resulting diagonalization yields "time parallel" algorithms, i.e., algorithms that possess a highly decoupled temporal structure, and hence can be efficiently implemented on emerging massively parallel MIMD architectures with a minimum of communication and synchronization overhead. Specifically, our algorithms have been implemented on the Intel Touchstone Delta supercomputer. We have illustrated our approach on a two-dimensional heat equation, and have demonstrated that a linear speedup can be achieved and maintained, even for a very large number of processor nodes.

1. Introduction

A large variety of physical phenomena can be described by means of Partial Differential Equations (PDEs)[1]. Generally, in most practical applications, an analytical solution can not readily be formulated. Hence, numerical solutions of such equations are usually considered. From such a perspective, the development of fast and accurate algorithms has been extensively studied in the literature. In particular, recent advances in massively parallel hardware architectures have provided a strong incentive to develop methodologies that, from the onset, take full advantage of the available large scale parallelism.

The Intel Delta, Intel Paragon, and CRAY T3D are representatives of an emerging class of massively parallel MIMD architectures. The main feature of this class of parallel architectures is that while they provide a large number of very powerful nodes with vector processing capability, they also possess a rather simple and limited communication structure (e.g., a mesh structure for the Delta). Therefore, such supercomputers are most suitable for applications where coarse grain parallel algorithms with limited communication requirements can be developed.

It is widely believed that, in numerical solutions of time dependent PDEs, marching in time is strictly sequential. Recently, however, we have developed[2-5] a concept of time parallel algorithms, which allows the marching in time procedure to be fully parallelized. This is achieved by using a transformation based upon the eigenvalue-eigenvector decomposition of the matrices resulting from the discretization process. Since these matrices are involved in the time stepping iterations, the resulting diagonalization yields "time parallel" algorithms, i.e., algorithms that possess a highly decoupled structure and can, hence, be efficiently implemented on emerging massively parallel MIMD architectures with a minimum of communication and synchronization overhead.

In this paper, we consider the solution of a linear parabolic equation in a bounded domain. Without loss of generality, we limit ourselves to a homogenous, two-dimensional case with Dirichlet boundary conditions. Extension to higher dimensions, nonhomogeneous, space dependent coefficients, and different boundary conditions is being discussed elsewhere[2-5].

For the two-dimensional case under consideration, we take the domain to be a square of length L, i.e., $0 \leq x \leq L$ and $0 \leq y \leq L$. Hence, the parabolic PDE of interest is given as

$$\frac{\partial v}{\partial t} = \alpha(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) \tag{1}$$

The pertaining boundary and initial conditions are specified as follows:

$$v(t,0,y) = V_0^y \qquad 0 \leq y \leq L, \;\; 0 < t \leq t_f \tag{2a}$$

$$v(t,L,y) = V_L^y \qquad 0 \leq y \leq L, \;\; 0 < t \leq t_f \tag{2b}$$

$$v(t,x,0) = V_0^x \qquad 0 \leq x \leq L, \;\; 0 < t \leq t_f \tag{2c}$$

$$v(t,x,L) = V_L^x \qquad 0 \leq x \leq L, \;\; 0 < t \leq t_f \tag{2d}$$

$$v(0,x,y) = f(x,y) \qquad 0 \leq x \leq L, \;\; 0 \leq y \leq L, \tag{2e}$$

where α is constant and t_f denotes the final time. Superimposing a uniform grid on the domain, i.e., $x = j \times \Delta_x$, $1 \leq j \leq N$, $y = i \times \Delta_y$, $1 \leq i \leq N$, and assuming $\Delta_x = \Delta_y = h = L/(N+1)$, will result in discrete values, v_{ij}^k, which approximate the continuous values $v(k\Delta_t, jh, ih)$. In the sequel, the grid points values of v will be referred to either in terms of the $N \times N$ matrix v_{ij}, or in terms of the N^2 vector v_ℓ, where $\ell = (i-1) \times N + j$, and $1 \leq i,j \leq N$.

Using a three point central differencing approximation scheme for the spatial domain yields a family of iterative methods, formalized as:

$$[I + 2\beta\delta M]v^{k+1} = [I - 2(1-\beta)\delta M]v^k - 2\delta[\beta V^{k+1} + (1-\beta)V^k] \qquad 0 \leq k \leq K \tag{3}$$

In the above expression, I denotes the $N^2 \times N^2$ identity matrix, $\delta = \Delta_t/2h^2$, where Δ_t is the magnitude of the time step, and $K = t_f/\Delta_t$. The $N^2 \times N^2$ matrix M arises from the discretization of the second order spatial derivatives, and has a special structure, i.e., "tridiagonal with fingers", as shown in Fig. 1. The N^2 vector V incorporates the time dependent boundary conditions (see Fig. 2)., and has the explicit form:

$$\begin{aligned} V = [&v_{1,0} + v_{0,1}, v_{0,2}, \cdots, v_{0,N-1}, v_{0,N} + v_{1,N+1}, \\ &v_{i,0}, 0, \cdots, 0, v_{i,N+1}, \qquad 2 \leq i \leq N-1 \\ &v_{N,0} + v_{N+1,1}, v_{N+1,2}, \cdots, v_{N+1,N-1}, v_{N+1,N} + v_{N,N+1}]^T \end{aligned} \tag{4}$$

Finally, the constant β determines the implicit degree of the method. Three distinct regimes can be considered in terms of β:

1- Explicit method, $\beta = 0$; then Eq. (3) becomes:

$$v^{k+1} = (I - 2\delta M)v^k - 2\delta V^k \qquad 0 \leq k \leq K \tag{5}$$

2- Implicit method, $\beta = 1$; now Eq. (3) becomes:

$$(I + 2\delta M)v^{k+1} = v^k - 2\delta V^{k+1} \qquad 0 \leq k \leq K \tag{6}$$

3- Crank-Nicholson (C-N) method, $\beta = 1/2$; in this case Eq. (3) can be written as:

$$(I + \delta M)v^{k+1} = (I - \delta M)v^k - \delta(V^{k+1} + V^k) \qquad 0 \leq k \leq K \tag{7}$$

The analysis of these methods in terms of stability, accuracy and domain of applicability is outside the scope of this paper. In the sequel, we will focus our discussion on the C-N method, with the understanding that the parallel algorithms presented in this paper are, in principle, applicable to all three methods.

The marching in time procedure for solving the PDE, Eq. (1), is implied by the time stepping k-iterations in Eqs. (5-7). Throughout this paper, the term *space parallel* is used for algorithms that exploit parallelism in the spatial domain only, while the term *time parallel* refers to algorithms that exploit parallelism in the computation of all vectors v^k, over the discrete time k-domain.

The time stepping formalism implied by Eqs. (5-7) appears to require strictly sequential computation in time. A number of paradigms have been proposed to enable some level of time parallelism. To date, however, only limited success has been reported[6- 11].

In this paper, we present a new class of time parallel algorithms that, for the type of problems defined by Eq. (1), allows the time iterations in Eq. (3) to be completely decoupled and performed fully in parallel. Our decoupling is achieved by casting Eq. (3) into a diagonal form. Specifically, a transformation based on the eigenvalue-eigenvector decomposition of the matrix M, reduces Eq. (3) to a set of First Order Linear Recurrences (FOLR), which allows the solution for all time steps to be computed concurrently.

This paper is organized as follows. The novel time parallel algorithms are described in Section 2. The best serial algorithm known to the authors is given in Section 3. The heat equation, which is used as a specific illustrative framework for benchmarking the proposed formalism, is presented in Section 4. The results of our numerical simulations on the Intel Touchstone Delta supercomputer are given in Section 5. The conclusions summarize our essential findings.

2. Time Parallel Algorithm Description

The time parallel algorithm we propose is based on the derivation of the eigenvalue-eigenvector decomposition of the matrix M. For completeness, we first recall (for proof, see e.g., 13, p.349) the following theorems:

Theorem 1. The eigenvalue-eigenvector decomposition of an $N \times N$ symmetric, tridiagonal Toeplitz matrix $\psi = Tridiag[b, a, b]$ can be written as

$$\psi = \theta \lambda \theta \tag{8}$$

The rows of the matrix θ correspond to the normalized eigenvectors of the matrix ψ, with elements given by:

$$\theta_{ij} = \frac{2}{\sqrt{N+1}} \sin(\frac{\pi i j}{N+1}) \tag{9}$$

The $N \times N$ diagonal matrix λ involves the set of eigenvalues of the matrix ψ, with the values of the i^{th} diagonal element given by:

$$\lambda_i = a + 2b\cos(\frac{i\pi}{N+1}) \tag{10}$$

It should be noted that the matrix θ is the one-dimensional Discrete Sine Transform (DST) operator. Hence, it is a symmetric, orthonormal matrix, i.e., $\theta = \theta^T = \theta^{-1}$.

Now, let us consider a $N^2 \times N^2$ block diagonal matrix $\Theta = Diag[\theta, \theta, \cdots, \theta]$, in which each $N \times N$ block is given by Eq. (9). Furthermore, we define the $N^2 \times N^2$ permutation matrix P, as depicted in Fig. 3. The effect of applying P to the N^2 vector with elements v_ℓ, is equivalent to transposing the $N \times N$ matrix with elements v_{ij}. Since both matrices Θ and P are symmetric, we have $\Theta = \Theta^T = \Theta^{-1}$ and $P = P^T = P^{-1}$.

Theorem 2. The matrix M has an eigenvalue-eigenvector decomposition of the form:

$$M = \Theta P \Psi P \Theta \tag{11}$$

in which Ψ is a $N^2 \times N^2$ block diagonal matrix. Each block, Ψ_i, has a symmetric tridiagonal Toeplitz structure given by $\Psi_i = Tridiag[-1, \lambda_i, -1]$ where λ_i is defined by Eq. (10).

From Theorem 1 and the definition of θ and ψ, we can see that the eigenvalue-eigenvector decomposition of Ψ is given by

$$\Psi = \Theta \Lambda \Theta \tag{12}$$

Here, the value of each element of the $N^2 \times N^2$ diagonal matrix Λ is computed according to:

$$\Lambda_\ell = 4 - 2\cos(\frac{i\pi}{N+1}) - 2\cos(\frac{j\pi}{N+1}) \tag{13}$$

Where $\ell = (i-1) \times N + j, \;\; 1 \leq i, j \leq N$.

By substituting Eq. (12) into Eq. (11), the eigenvector-eigenvalue decomposition of the matrix M can be written as:

$$M = \Theta P \Theta \Lambda \Theta P \Theta \tag{14}$$

Note that the definitions of Θ and P imply that the matrix

$$\Phi = \Theta P \Theta \tag{15}$$

is also symmetric and orthonormal, i.e., $\Phi = \Phi^T = \Phi^{-1}$.

Our time parallel algorithm is derived by substituting Eqs. (14) and (15) into Eq. (7). After some re-arrangement, one obtains:

$$\Phi(I + \delta\Lambda)\Phi v^{k+1} = \Phi(I - \delta\Lambda)\Phi v^k - \delta(V^{k+1} + V^k) \quad 0 \leq k \leq K \tag{16}$$

We now define

$$\hat{v} = \Phi v \tag{17a}$$

$$\hat{V} = \Phi V \tag{17b}$$

and

$$\hat{d}_\ell^{k+1} = \frac{\delta(\hat{V}_\ell^{k+1} + \hat{V}_\ell^k)}{(1 + \delta\Lambda_\ell)} \quad 1 \leq \ell \leq N^2 \tag{17c}$$

Furthermore, we introduce the $N^2 \times N^2$ diagonal matrix D, with elements given by

$$D_\ell = \frac{(1 - \delta\Lambda_\ell)}{(1 + \delta\Lambda_\ell)} \quad 1 \leq \ell \leq N^2 \tag{18}$$

Recalling the orthonormality of Φ, i.e., $\Phi = \Phi^{-1}$, and substituting Eqs. (17) and (18) into Eq. (16), we arrive the recurrence:

$$\hat{v}^{k+1} = D\hat{v}^k - \hat{d}^{k+1} \qquad 0 \leq k \leq K \tag{19}$$

which represents a first order inhomogeneous linear system. The above expression is equivalent to

$$\hat{v}^k = D^k\hat{v}^0 - [\hat{d}^{k+1} + D\hat{d}^k + \cdots + D^k\hat{d}^1] \qquad 1 \leq k \leq K \tag{20}$$

At this stage, it is important to keep in mind that the superscript k generally stands for the value at the iteration number k, except in the case of D^k where it stands for power (i.e., D^k means D to the power of k). If constant boundary conditions over time are assumed (i.e., if $\hat{d}^k = \hat{d},\ 1 \leq k \leq K$), one can simplify Eq. (20) to:

$$\hat{v}^k = D^k\hat{v}^0 - \hat{d}\frac{1 - D^k}{1 - D} \tag{21}$$

Furthermore, if one assumes homogenous boundary conditions, (i.e., $\hat{d} = 0, \quad 0 \leq k \leq K$), this equation will further reduce to:

$$\hat{v}^k = D^k\hat{v}^0 \tag{22}$$

When the boundary conditions are time dependent, the linear recurrence in Eq. (19) can be solved in $O(logK)$ by using a conventional parallel algorithm[16]. However, for time independent boundary conditions, Eq. (21) can be computed in parallel without communication overheads, which preserves the fully decoupled structure of the algorithm.

From a computational complexity perspective, the time parallel algorithm for constant boundary conditions can be summarized in the following three steps:

- Transform the initial and boundary conditions vector, i.e., compute

$$\hat{v}^0 = \Phi v^0$$

$$\hat{b} = \Phi V$$

  This step can be accomplished in $O(N^2 logN)$ multiply-accumulates, using fast transforms on a single processor.

- Calculate each vector $\hat{v}^k$ using either Eq. (21) or Eq.(22). This step consist of N^2 multiply-accumulates for each time step, k.

- Apply an inverse transform to the vector $\hat{v}^k$, to obtain v^k

$$v^k = \Phi \hat{v}^k \qquad 1 \leq k \leq K$$

  This step can be accomplished in $O(N^2 logN)$ multiply-accumulates for each time step k, by using fast transforms, and assuming a single processor.

The overall computational complexity of the time-parallel algorithm on a single processor machine is therefore $O(KN^2 logN)$. Because of the inherently decoupled structure of our algorithm, this complexity scales as $O(\frac{KN^2 logN}{N_p})$ on a system involving N_p processor nodes.

3. Best Serial Algorithm

In order to determine the "best" serial algorithm for the problem under consideration, we make the following observations. The coefficient matrices in Eqs. (5- 7) have a symmetric, positive-definite, and sparse structure. This allows the use of rather generic iterative methods such as SOR, conjugate gradient, etc. [14] for solving the corresponding linear systems. More importantly, however, we note that these matrices also have structures similar to those arising in the solution of the Poisson equation. In that sense, Eqs. (5-7) represent a sequence of Poisson equations. Therefore, the so called Fast Poisson Solvers can be used for the direct solution of the linear systems, Eqs. (5-7), with a greater computational efficiency than the conventional iterative methods [15]. In the sequel, we consider an improved version of the Matrix Decomposition algorithm of [23].

The computational complexity of such a "best serial algorithm" must be evaluated in a framework consistent with the time parallel formalism. Hence, we note that this algorithm is also based on the decomposition of matrix M. However, this decomposition is now limited to that specified in Theorem 2. Substituting Eq. (11) into the C-N scheme, Eq.(7), and rearranging the terms we arrive at:

$$\Theta P(I + \delta\Psi)P\Theta v^{k+1} = \Theta P(I - \delta\Psi)P\Theta v^{k} \qquad 0 \leq k \leq K \tag{23}$$

Let us define

$$\bar{v} = P\Theta v \tag{24}$$

Using the orthonormality of Θ and P, and substituting Eq. (24) into Eq. (23), one obtains

$$(I + \delta\Psi)\bar{v}^{k+1} = (I - \delta\Psi)\bar{v}^{k} \qquad 0 \leq k \leq K \tag{25}$$

One can then rewrite Eq. (25) as follows:

$$(I + \delta\Psi)(\bar{v}^{k+1} + \bar{v}^{k}) = 2\bar{v}^{k} \qquad 0 \leq k \leq K \tag{26}$$

Now, if one defines $\bar{w}^{k+1} = \bar{v}^{k+1} + \bar{v}^{k}$, the C-N method can be recast as

$$(\frac{I}{2} + \frac{\delta\Psi}{2})\bar{w}^{k+1} = \bar{v}^{k} \qquad 0 \leq k \leq K \tag{27}$$

$$\bar{v}^{k+1} = \bar{w}^{k+1} - \bar{v}^k \qquad 0 \leq k \leq K \tag{28}$$

Thus, the best serial algorithm can be summarized in the following three steps:

- o Transform the vector of initial conditions, i.e.,

  $$\bar{v}^0 = P\Theta v^0$$

  This step can be accomplished in $O(N^2 logN)$ multiply-accumulates, using fast transforms on a single processor.

- o Calculate the vector $\bar{w}^k$ by solving the linear system Eq. (27), and $\bar{v}^k$ using Eq. (28); repeat this for all time steps, $k = 1, 2, \cdots, K$; the system of $O(N^2)$ linear equations (27) has a symmetric tridiagonal Toeplitz structure: hence, it can be solved in $O(N^2)$ steps.

- o At each time step one needs to output the vector v^k, which is obtained by applying the inverse transformation to $\bar{v}^k$:

  $$v^k = \Theta P \bar{v}^k \qquad 1 \leq k \leq K$$

  This step can be accomplished in $O(N^2 logN)$ multiply-accumulates at each time step k, again by using fast transforms.

Thus, the overall computational complexity of the best serial algorithm on a single processor is $O(KN^2 logN)$.

4. The Heat Equation

In order to provide a concrete framework for assessing the potential of our proposed approach to time parallelism, we focus our attention on a two-dimensional heat conduction problem modeled by a linear parabolic PDE[12]. This problem has the advantage of exhibiting both sufficient computational complexity, and possessing analytical solutions. This enables a rigorous benchmark of the algorithms under consideration.

To fix the ideas, consider the problem of transient conduction in a long bar having a square cross section, of thickness L. The bar is assumed to be infinite in the z direction, so that the heat profile will vary only in the x and y directions. For simplicity, we furthermore assume that the cross sectional temperature, $u(t,x,y)$ is given at time $t = 0$ by

$$u(t,x,y) = sin(x/L) \cdot sin(y/L) \tag{29}$$

where $0 \leq x \leq L, \quad 0 \leq y \leq L$. The temperature of the bar at the boundaries is kept constant, i.e.,

$$u(t,x,0) = u(t,x,L) = u(t,0,y) = u(t,L,y) = 0 \tag{30}$$

Thus, the differential equation to be solved is:

$$\frac{\partial u}{\partial t} = \alpha(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{31}$$

where the constant α is the thermal diffusivity. Since the initial temperature distribution is a product of two functions, each of which involves only one of the independent space variables, the solution to the Eq. (31) may be expressed as the product of two one-dimensional, transient solutions, namely;

$$u(t,x,y) = X(t,x) \cdot Y(t,y) \tag{32}$$

Substituting Eq.(32) into Eq.(31) and rearranging the terms will results in two one-dimensional problems:

$$\begin{gathered} \frac{\partial X}{\partial t} = \alpha \frac{\partial^2 X}{\partial x^2} \\ X(t,x) = \sin(x/L) \quad 0 \leq x \leq L, \quad t = 0 \\ X(t,0) = X(t,L) = 0 \end{gathered} \tag{33}$$

$$\begin{gathered} \frac{\partial Y}{\partial t} = \alpha \frac{\partial^2 Y}{\partial y^2} \\ Y(t,y) = sin(y/L) \quad 0 \leq y \leq L, \quad t = 0 \\ Y(t,0) = Y(t,L) = 0 \end{gathered} \tag{34}$$

Solution of Eq.(33) by separation of variables is obtained by again assuming that each temperature distribution is the product of two functions, each of which involves only one of the independent variables. That is, if $f(x)$ is a function of x only and if $g(t)$ is a function of t only, then the temperature distribution, X, can be expressed as

$$X(t,x) = f(x) \cdot g(t) \tag{35}$$

When this is substituted into Eq. (33) and the resulting ordinary differential equation is solved, one obtains the following general solution

$$X(t,x) = e^{-\lambda^2 \alpha t}[Asin(\lambda x) + Bcos(\lambda x)] \tag{36}$$

The constant A, B and λ are evaluated using the pertaining initial and boundary conditions. Upon substitution, one will obtain the following temperature distribution in the x direction

$$X(t,x) = e^{-\lambda^2 \alpha t} sin(\lambda x) \qquad 0 \leq x \leq L,\ t \geq 0 \tag{37}$$

In the above expression λ is constant and equal to π/L.

Repeating the same procedure for Eq.(34), one obtains a similar solution for Y, i.e.,

$$Y(t,y) = e^{-\lambda^2 \alpha t} sin(\lambda y) \qquad 0 \leq y \leq L,\ t \geq 0 \tag{38}$$

Hence, using Eq.(32), the temperature distribution in a cross section of the bar is derived to be:

$$u(t,x,y) = e^{-2(\pi/L)^2 \alpha t} sin(\pi x/L) \cdot sin(\pi y/L) \tag{39}$$

This analytical expression will be used to validate the numerical results of the implementation of our time parallel algorithm on the Intel Touchstone Delta supercomputer.

5. Implementation Results

In order to evaluate the potential of our proposed time parallelism paradigm, a FORTRAN computer code for solving the 2-D heat equation [i.e., Eqs. (29-31)] was written and implemented on the Intel Touchstone Delta multi processor. The numeric nodes of the Delta are i860 microprocessors operating at 40 MHz. These nodes are rated at 33 MIPS, 80(peak) single precision MFLOPS, and 60 (peak) double precision MFLOPS.

On the Delta machine, processors are allocated in terms of a rectangular mesh. In our implementation, a square partition was generally used, in which the number of columns and rows was simultaneously varied from 1 to 10. For the case of 120 processors, a 10×12 rectangle was employed.

The number of time steps K required for our calculation was divided among the N_p processors in the following manner. The p^{th} processor calculates the k^{th} time step, where $k = p + m \times N_p$, $0 \leq m \leq K/N_p$, and $k \leq K$.

For the actual simulations, we selected the Crank-Nicholson scheme, i.e. we set $\beta = 0.5$, and assumed a thermal diffusivity of 0.1. The spatial mesh size and the time step size were chosen as $h = \Delta_x = \Delta_y = 0.1$ and $\Delta_t = 10^{-4}$ respectively. In order to enable a more accurate measurement of the computation time at each processor on one hand, and to allow for potential inaccuracies stemming from the numerical schemes to accumulate, we report results after 5000 iterations, i.e., 0.5 second after experiment start up.

It is important to remember that the bar thickness, L, was divided into $N + 1$ equi-length parts. This results in $N + 2$ equidistant grid points in each spatial direction (see Fig. 2). However, boundary points have a fixed value in this problem. Therefore, there are only N points in each spatial direction at which a computation is performed.

On the Intel Delta, each node program starts by reading the number of mesh points in each spatial direction. Then a file is opened, which is shared between all nodes, and used for recording the initialization and computation time of each node. The values of the parameters α, β, Δ_t, h and K are initialized, the initial temperature distribution and its transform are calculated, and the values of D_ℓ^p and B_ℓ, where $B_\ell = D_\ell^{N_p}$ and $1 \leq \ell \leq N^2$, are computed. This initialization time is a constant, function of the spatial resolution, N, and is measured to be on the average 12, 48, 190, 756 milliseconds, per node, for N equal to 15, 31, 63, 127 respectively.

Following initialization, each processor p first calculates the temperature distribution of the p^{th} time step using Eq. (22) and the D values currently in memory. Then, the value

of D at each node is updated according to the formula

$$D_\ell = D_\ell \times B_\ell \qquad 1 \leq \ell \leq N^2$$

yielding the quantities required for computing the temperature at the $(p + N_p)^{th}$ time step. After processing $O(K/N_p)$ time steps, the measured times for initialization and computation from each node are written onto the common file.

Table 1 shows the total (i.e., initialization plus computation) time, in milliseconds, for four different cases involving different mesh sizes. The first row of this table displays the time achieved with the best serial algorithm (see Section 3) using a single node of the Delta machine. The other rows present the average time per node, calculated according to

$$\tau_{ave} = \frac{1}{N_p} \sum_{p=1}^{p=N_p} \tau_p$$

where τ_p is the total time posted for processor p. The speedup (i.e., best serial algorithm processing time divided by τ_{ave}) achieved as a function of the number of nodes is displayed in Fig. 4.

We observe that, eventhough the serial algorithm has theoretically a lower computational complexity than the time parallel one, it actually required more time. This is due to the fact that the structure of the time parallel algorithm takes full advantage of the vector processing capability of the i860 node, which results in a lower overall computation time.

5. Conclusions

We have presented a novel time parallel algorithm for the solution of linear parabolic partial differential equations. The basic idea is to use a transformation involving the eigenvalue-eigenvector decomposition of coefficient matrices underlying the discretized form of the PDE. Since these matrices are involved in the time stepping iterations, the resulting diagonalization yields a decoupling of the iterations, which in turn allows the solution for all the time steps to be computed in parallel. Because of their highly decoupled structure, our time parallel algorithms can be efficiently implemented on emerging massively parallel MIMD architectures with a minimum of communication and synchronization overhead.

Our method has been implemented on the Intel Touchstone Delta supercomputer. For illustrative purposes, we have demonstrated the potential capabilities of our approach on a two-dimensional heat equation. Our implementation shows that the time parallel algorithms do indeed achieve maximum parallelism in time. In particular, we were able to reach a linear speedup factor of 120 using 120 processors. Our results clearly indicate that, in contradistinction to the general assumption of inherent sequentiality for marching in time, the iterations in Eq. (3) can be more efficiently parallelized in time rather than in space. As a result, even with a limited number of processors, it is more efficient to exploit parallelism in time than in space.

Acknowledgments

This research was carried out at the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology. Support for the work came from the Office of Advanced Concepts and Technology of the National Aeronautics and Space Administration, and from the Office of Basic Energy Sciences of the US Department of Energy through an agreement with NASA.

References

1. Whithman G.B., *Linear and Nonlinear Waves*, John Wiley and Sons, New York, NY (1974).

2. Fijany A. and N. Toomarian, "Fast Time and Space Parallel Algorithms for Solution of Parabolic Partial Differential Equations", Submitted to *IEEE Trans. on Parallel and Dist. Syst.* (January 1993).

3. Fijany A., "Time Parallel Algorithms for Solution of Linear Parabolic PDEs", Proceedings of the 1993 International Conference on Parallel Processing (in press, 1993).

4. Fijany A., "Time Parallel Algorithms for Solution of Linear Parabolic PDEs," Engineering Memorandum, EM 347-93-002, Jet Propulsion Laboratory (February 1993).

5. Fijany A., J. Barhen and N. Toomarian "On the Structure of Time-Parallel Algorithms for Solution of Linear Evolutionary Partial Differential Equations," in preparation.

6. Lelarasmee E., A. Ruheli, and A. L. Sangiovanni-Vincentelli, "The Waveform Relaxation Method for the Time Domain Analysis of Large Scale Integrated Circuits," *IEEE Trans. Computer-Aided Design*, **1**, 131-145 (1982).

7. Saltz J. H. and V. K. Nail, "Towards Developing Robust Algorithms for Solving Partial Differential Equations on MIMD Machines," *Parallel Computing*, **6**, 19-44 (1988).

8. Womble D. E., "A Time-Stepping Algorithm for Parallel Computers," *SIAM J. Sci. Stat. Comput.*, **11** (5), 824-837 (1990).

9. Hackbusch W., "Parabolic Multigrid Methods," Proc. 6th Int. Symp. on Computing Methods in Applied Sciences and Engineering (December 1983).

10. Horton G. and R. Knirsch, "A Time-Parallel Multigrid-Extrapolation Method for Parabolic Partial Differential Equations," *Parallel Computing*, **18**, 21-29 (1992).

11. Strang G. and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ (1973).

12. Chapman, A. J., *Heat Transfer*, Macmillan, New York, NY (1967).

13. Barnett S., *Matrices: Methods and Applications*, Clarendon Press, Oxford, UK (1990).

14. Varga R. S., *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ (1962).

15. Swarztrauber P. N. and R. A. Sweet, "Efficient Subroutines for the Solution of General Elliptic and Parabolic Partial Differential Equations", *Atmospheric Technology*, 79-81 (September 1973).

16. Hockney R. and C. Jesshope, *Parallel Computers.* Adam Hilger Ltd., London, UK (1981).

17. Buzbee B., G. Golub, and C. Nielson, "On Direct Methods for Solving Poisson Equations," *SIAM J. Numer. Anal.*, **7**, 627-656 (1970).

Figure Caption

Fig. 1: Structure of the matrix M obtained using a five point numerical discretization scheme. Elements not shown are zero.

Fig. 2: A two-dimension uniform grid representing the spatial domain.

Fig. 3: Structure of the permutation matrix P. Elements not shown are zero.

Fig. 4: Speedup of the time-parallel algorithm as function of the number of processors for different mesh size.

Table 1: Total execution time, in milliseconds, for different mesh sizes and number of processors employed.

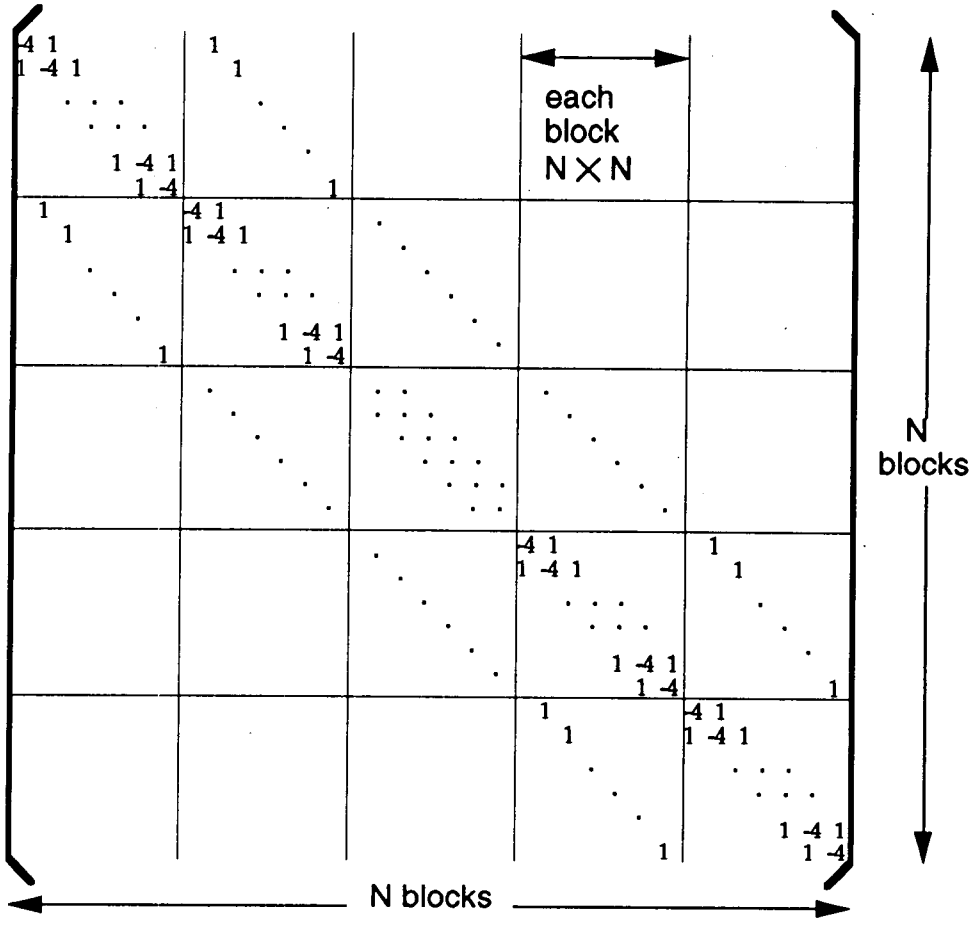

Fig. 1: Structure of matrix M obtained using a five points numerical discretization scheme. Elements not shown are zero.

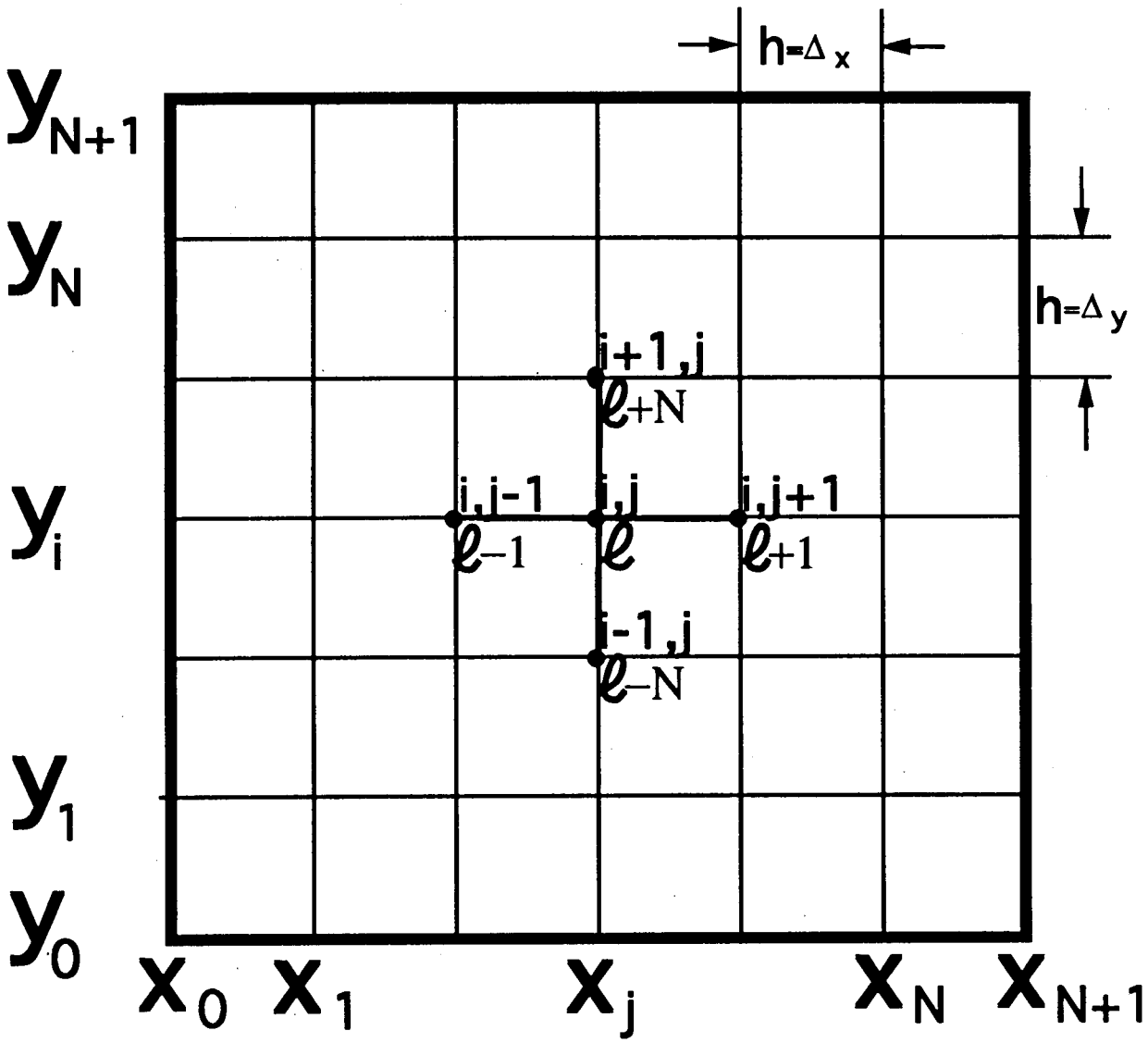

Fig. 2: A two-dimension uniform grid representing the spatial domain.

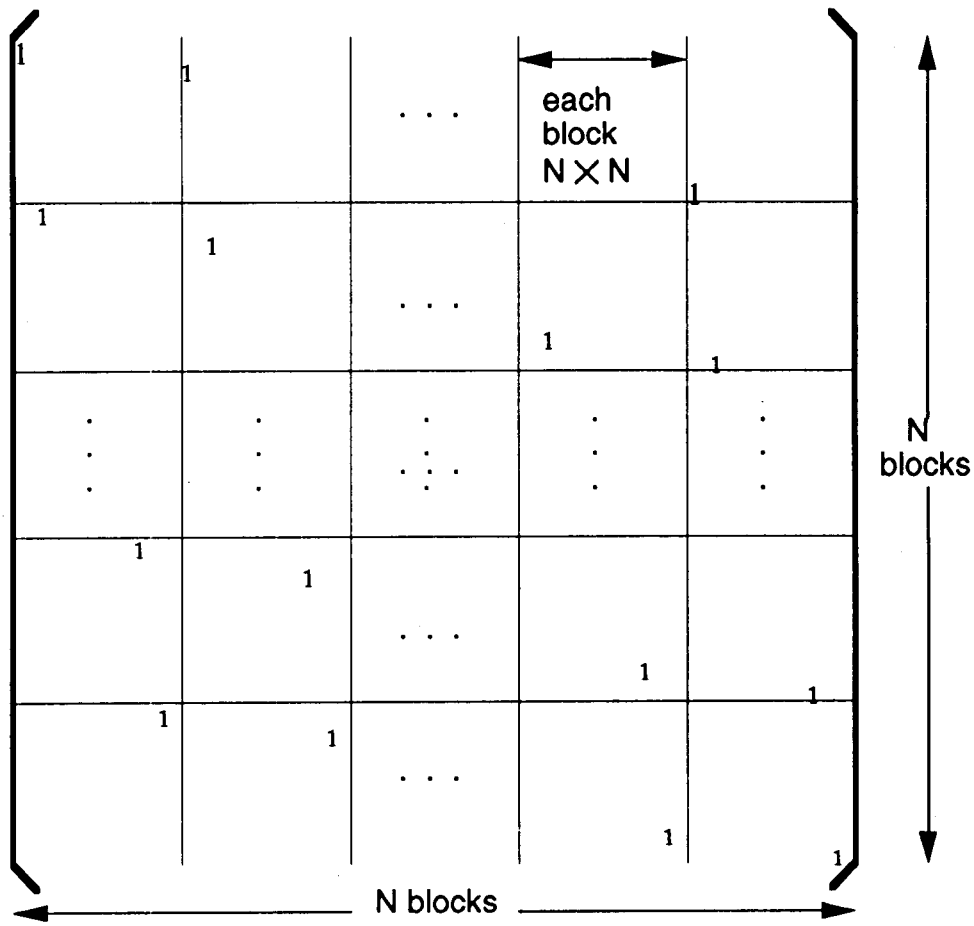

Fig. 3: Structure of the permutation matrix P. Elements not shown are zero

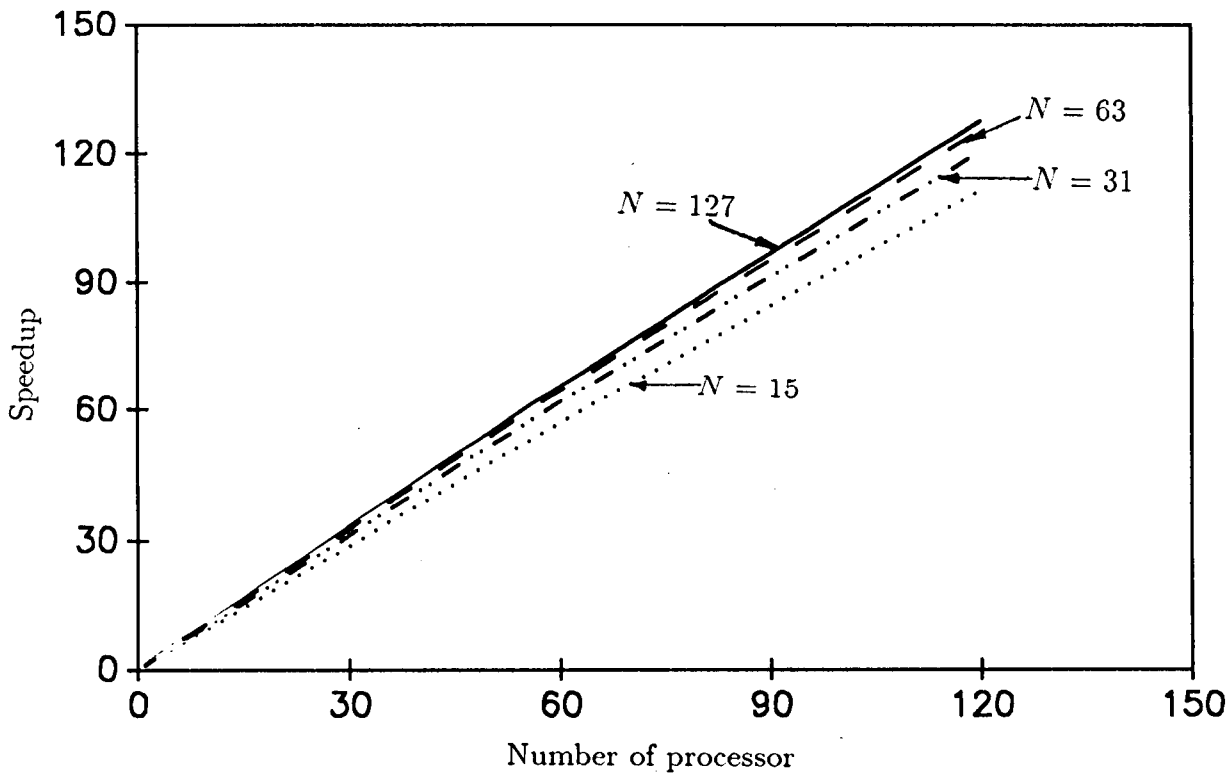

Fig. 4: Speedup of the time-parallel algorithm as function of the number of processors for different mesh size.

Table 1: Total execution time, in milliseconds, for different mesh sizes and number of processors employed

Number of	Dimension of the Mesh			
Processors	15	31	63	127
1	30364	119453	473665	1898395
1	31208	118109	429817	1693977
4	7811	28642	107595	424074
9	3478	12620	47925	188885
16	1962	7119	27040	106576
25	1260	4574	17373	68475
36	879	3191	12123	47784
49	649	2357	8957	35307
64	500	1816	6903	27210
81	389	1447	5500	21680
100	324	1181	4491	17703
120	272	992	3774	14877