Parallelization of 2-D IADE-DY Scheme on Geranium Cadcam Cluster for Heat Equation

Simon Uzezi Ewedafe
Computing & IT. Baze University, Abuja
Baze University, Abuja
Abuja, Nigeria

Rio Hirowati Shariffudin
Institute of Mathematical Sciences
Universiti Malaya
Kuala Lumpur, Nigeria

***Abstract*—A parallel implementation of the Iterative Alternating Direction Explicit method by D'Yakonov (IADE-DY) for solving 2-D heat equation on a distributed system of Geranium Cadcam cluster (GCC) using the Message Passing Interface (MPI) is presented. The implementation of the scheduling of n tri-diagonal system of equations with the above method was used to show improvement on speedup, effectiveness, and efficiency. The Master/Worker paradigm and Single Program Multiple Data (SPMD) model is employed to manage the whole computation based on the use of domain decomposition. The completion of the execution can need task recovery and favorable configuration. The above mentioned details consist of a main report about the numerical validation of the parallelization through simulation to demonstrate the proposed method effectiveness on the cluster system. It was found that the rate of convergence decreases as the number of processors increases. The result of this paper suggests that the 2-D IADE-DY scheme is a good approach to solving problems, particularly when it is simulation with more processors.**

Keywords*—Parallel Implementation; Heat Equation; SPMD; IADE-DY; Domain Decomposition

I. INTRODUCTION

Software programmers developing parallel application do focus on some challenges in the area of parallel computing. According to [18] there are theoretical challenges such as task decomposition, dependence analysis, and task scheduling. Then there are practical challenges such as portability, synchronization, and debugging. An alternative and cost effective means of achieving a comparable performance is by way of distributed computing, using a system of processors loosely connected through a local area network [3]. For a global computational task with other processors, relevant data need to be passed from processors to processors through a message passing mechanism [7, 11, 28, 22]. There is greater demand for computational speed and computations must be completed within reasonable time period by using multiple processors on a single problem, hence, the demand for faster processors has been growing rapidly, which can only be met by the sue of parallel computers for grand challenge problems [19, 30] and [4].

There are a number of important unresolved questions concerning multiprocessor computers, among these issues are: should they consist of a few, rather powerful processors or many very much less powerful processors, or something in between? According to [16] there is a natural expectation that the multiprocessors with a few, powerful processors will have an MIMD architecture, and that the others will have SIMD architecture. Parallelization of heat equation has been proposed by [3], and recent developments have included a number of different applications [5, 2]. Another issue is the communication among the processors. How is the memory connected to the processors, and how are these processors connected to each other? The model proposed in this paper enhances overlap communication and computation to avoid unnecessary synchronization; hence, the method yields significant speedup by the use of the non-blocking communication.

While the theoretical properties of the 2-D IADE-DY algorithm employing the master/worker paradigm and SPMD model are promising, achieving good performance in practice can be challenging. In reference to [2] this is due to fundamental tradeoff between the reduction of the time required for an inherently sequential part of the algorithm, and an increase in the number of the iterations required to converge. Previous analysis of the IADE scheme in the literature did not consider the efficient parallelization and scheduling of tasks to improve scalability. Sequential numerical methods for solving time dependable problems have been explored extensively [25, 30].

A number of software tools have been developed for parallel implementation, MPI [19] is chosen since it has a large user group. The objective of our parallel focus is to improve performance. Due to our objective, parallelizing code has traditionally been paired with general code optimizations for performance, especially in the scientific and engineering area [18].

The main contribution of this paper is to present a detailed study of the parallelization using the 2-D IADE-DY algorithm employing master/worker paradigm and SPMD model to enhance overlapping communication with computation on the GCC cluster system running MPI that result in significant improved speedup, effectiveness, and efficiency across varying mesh sizes. The Master/Worker paradigm and SPMD model is employed to manage the whole computation based on the use of domain decomposition. The completion of the execution can need task recovery and favorable configuration. Our results demonstrate two properties that make this approach attractive for the platform of GCC: overlap communication and computation, and ability to arbitrary use various varying mesh sizes. The distribution done in the GCC reduces the memory pressure on the master while preserving parallel efficiency.

To obtain results with sufficient accuracy for the numerical prediction of the scalable parallel implementation of the AGE, IADE and ADI algorithm, fine discretization of the domain would be necessary. Due to the limitation in both processing element power and memory on sequential architectures and the dimension of full scale utility, only coarse grids are possible. A confine enhancement may be achieved if a domain decomposition method is used to allow locally refined meshes. The paper is organized as follows: section 2 emphasizes on previous related work, section 3 introduces the model for the 2-D heat equation and method and the 2D-IADE-DY scheme. Section 4 and 4 introduces the performance analysis and numerical experiment. Finally, a conclusion is included in section 6.

II. Previous Work

Parallelization of Partial Differential Equations (PDE) by time decomposition was first proposed by [24]. The motivation for the paper was to achieve parallel real-time solutions. Recent improvements have included a number of different applications [5], and [2] emphasizes the scheduling of tasks in the Para real algorithm. The importance of loop parallelization and loop scheduling has been extensively studied [1]. This work is distinct while promoting flexibility, and applies standard parallel concepts. Several approaches to solving heat equation have been carried out in [6, 25, 26, 27] and [13, 29, 32]. We have applied the 2-D IADE-DY scheme by simulation to schedule the n tri-diagonal system of equations with the above method used to show improvement on speedup, effectiveness, and efficiency. Reference [10] and [12] show speedup and efficiency, while comparing to our results generated using GCC, the GCC results show better conformity to linearity for speedup and closeness to unity for efficiency than [10] as applied to the simple method using MPI. In [20], the unconditional stability of the alternating difference schemes has similarity to our scheme. Our implementation compared to [26] and [27, 6] is a way of proofing stability and convergence in the GCC cluster system. We also note the various constant improvements on speedup, effectiveness, and efficiency analysis carried out in [33h] using the overlapping domain decomposition method. However, [32] proposed a generalized speedup formula as the ratio of the parallel to sequential speed. As in relation to the performance strategies implementation, a thorough study of speedup models together with their advantages is implemented in [30, 9, 28] show the same conformity to our implementation, but here we were able to achieve unity conformity in the message passing mechanism.

III. The Model Problem

The problem that is of interest to us is the heat equation in 2-dimension. We assume that the heat will spread within the field based on a dynamics described in [27, 31] and the Alternating Group Explicit [13] method by the following:

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + h(x,y,t), (x,y,t) \in R \times (0,T], \quad (3.1)$$

with the initial condition,

$$U(x,y,0) = F(x,y), (x,y,t) \in R \times \{0\}, \quad (3.1a)$$

and $U(x,y,t)$ is specified on the boundary of $R, \partial R$ by

$$U(x,y,t) = G(x,y,t), (x,y,t) \in \partial R \times (0,T], \quad (3.1b)$$

where for simplicity we assume that the region *R* of the *xy*-plane is a rectangle. Consider the two-dimensional heat (3.1) with the auxiliary conditions (3.1a) and (3.1b). The region R is a rectangle defined by

$$R = \{(x,y) : 0 \le x \le L, 0 \le y \le M\}.$$

At the point $P(x_i, y_j, t_k)$ in the solution domain, the value of $U(x,y,t)$ is denoted by $U_{i,j,k}$ where $x_i = i\Delta x,\ y_j = j\Delta y \ \ for\ \ 0 \le i \le (m+1), 0 \le j \le (n+1)\ \ and$ $\Delta x = L/(m+1),\ \Delta y = M/(n+1).$ The increment in the time t, Δt is chosen such that $t_k = k\Delta t \ \ for\ k = 0,1,2,\ldots$ for simplicity of presentation, we assume that m and n are chosen so that $\Delta x = \Delta y$ and consequently the mesh ratio is defined by $\lambda = \Delta t/(\Delta x)^2$.

A. The IADE-DY and DS-MF

By fractional splitting, each time step in the double sweep methods is split into two steps of size $\Delta t / 2$. The horizontal sweep advances from t_k to $t_{k+1/2}$ by using a difference approximation that is implicit in only the x-direction. Specifically, past values in the y-direction along the grid line $x = x_i$ are used, to yield the intermediate value $u_{i,j,k+1/2}$. Then, in the vertical sweep from $t_{k+1/2}$ to t_{k+1} , the solution is obtained by using an approximation implicit in only the y-direction and uses past values in the x-direction along the grid line $y = y_j$, to yield the final value $u_{i,j,k}$.

At the $(k+1/2)$ time level method, the solution of (3.1) uses a backward-difference approximation.

$$u_{i,j,k+1/2} - u_{i,j,k} = \frac{\lambda}{2}\delta_x^2 u_{i,j,k+1/2} + \frac{\lambda}{2}\delta_y^2 u_{i,j,k} \quad (3.2)$$

Where $\delta_x \ and\ \delta_y$ are the usual central difference operators in the *x* and *y* coordinates respectively.

$$u_{i,j,k+1} - u_{i,j,k} = \frac{\lambda}{2}(u_{i-1,j,k+1/2} - 2u_{i,j,k+1/2} + u_{i+1,j,k+1/2}) + \frac{\lambda}{2}(u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}) \quad (3.3)$$

$$-\frac{\lambda}{2}u_{i-1,j,k+1/2}+(1+\lambda)u_{i,j,k+1/2}-\frac{\lambda}{2}u_{i+1,j,k+1/2}=\frac{\lambda}{2}u_{i.j+1,k}+(1-\lambda)u_{i,j,k}+\frac{\lambda}{2}u_{i,j-1,k} \tag{3.4}$$

From (3.4), for $j=1,2,\ldots,n$

$$i=1:(1+\lambda)u_{1,j,k+1/2}-\frac{\lambda}{2}u_{2,j,k+1/2}=\frac{\lambda}{2}u_{o,j,k+1/2}+\frac{\lambda}{2}u_{1,j+1,k}+(1-\lambda)u_{1,j,k}+\frac{\lambda}{2}u_{1,j-1,k} \tag{3.5}$$

$$i=2,3,\cdots,m-1:-\frac{\lambda}{2}u_{i-1,j,k+1/2}+(1+\lambda)u_{i,j,k+1/2}-\frac{\lambda}{2}u_{i+1,j,k+1/2}=\frac{\lambda}{2}u_{i,j+1,k}+(1-\lambda)u_{i,j,k}+\frac{\lambda}{2}u_{i,j-1,k} \tag{3.6}$$

$$i=m:-\frac{\lambda}{2}u_{m-1,j,k+1/2}+(1+\lambda)u_{m,j,k+1/2}=\frac{\lambda}{2}u_{m+1,j,k+1/2}+\frac{\lambda}{2}u_{m,j-1,k}+(1-\lambda)u_{m,j,k}+\frac{\lambda}{2}u_{m,j+1.k} \tag{3.7}$$

let $a=1+\lambda,\ b=c=-\frac{\lambda}{2}$. Equation (3.5) – (3.7) can be written in a more compact matrix form as:

$$Au_j^{(k+1/2)}=f_k,\qquad j=1,2,\ldots,n. \tag{3.8}$$

where

$$u=(u_{1,j},u_{2,j},\ldots,u_{m,j})^T,\quad f=(f_{1,j},f_{2,j},\ldots,f_{m,j})^T$$

$$f_{1,j}=\frac{\lambda}{2}u_{1,j+1,k}+(1-\lambda)u_{1,j,k}+\frac{\lambda}{2}(u_{1,j-1,k}+u_{0,j,k+1/2})$$

$$f_{i,j}=\frac{\lambda}{2}u_{i,j+1,k}+(1-\lambda)u_{i,j,k}+\frac{\lambda}{2}u_{i,j-1,k}\quad i=2,3,\ldots,m-1$$

$$f_{m,j}=\frac{\lambda}{2}u_{m,j+1,k}+(1-\lambda)u_{m,j,k}+\frac{\lambda}{2}(u_{m,j-1,k}+u_{m+1,j,k+1/2}) \tag{3.9}$$

at the (k+1) time level, (3.1) is approximated by,

$$u_{i,j,k+1}-u_{i,j,k+1/2}=\frac{\lambda}{2}\delta_x^2u_{i,j,k+1/2}+\frac{\lambda}{2}\delta_y^2u_{i,j,k+1} \tag{3.10}$$

$$u_{i,j,k+1}-u_{i,j,k+1/2}=\frac{\lambda}{2}(u_{i-1,j,k+1/2}-2u_{i,j,k+1/2}+u_{i+1,j,k+1/2})+\frac{\lambda}{2}(u_{i,j-1,k+1}-2u_{i,j,k+1}+u_{i,j+1,k+1}) \tag{3.11}$$

$$-\frac{\lambda}{2}u_{i,j-1,k+1}+(1+\lambda)u_{i,j,k+1}-\frac{\lambda}{2}u_{i,j+1,k+1}=\frac{\lambda}{2}u_{i-1,j,k+1/2}+(1-\lambda)u_{i,j,k+1/2}+\frac{\lambda}{2}u_{i+1,j,k+1/2} \tag{3.12}$$

from (3.4), for i = 1, 2, . . . , m.

$$j=1:(1+\lambda)u_{i,1,k+1}-\frac{\lambda}{2}u_{i,2,k+1}=\frac{\lambda}{2}u_{i-1,1,k+1/2}+(1-\lambda)u_{i,1,k+1/2}+\frac{\lambda}{2}u_{i+1,1,k+1/2}+\frac{\lambda}{2}u_{i,0,k+1} \tag{3.13}$$

$$j=2,3,\cdots,n-1:$$

$$-\frac{\lambda}{2}u_{i,j-1,k+1}+(1+\lambda)u_{i,j,k+1}-\frac{\lambda}{2}u_{i,j+1,k+1}=\frac{\lambda}{2}u_{i-1,j,k+1/2}+(1-\lambda)u_{i,j,k+1/2}+\frac{\lambda}{2}u_{i+1,j,k+1/2} \tag{3.14}$$

$$j=n:-\frac{\lambda}{2}u_{i,n-1,k+1}+(1+\lambda)u_{i,n,k+1}=\frac{\lambda}{2}u_{i-1,n,k+1/2}+(1-\lambda)u_{i,n,k+1/2}+\frac{\lambda}{2}u_{i+1,n,k+1/2}+\frac{\lambda}{2}u_{i,n+1,k+1} \tag{3.15}$$

let $a=1+\lambda,\ b=c=-\frac{\lambda}{2}$. Equations (3.13) – (3.15) can be displayed in a more compact matrix form as:

$$Bu_i^{(k+1)}=g_{k+1/2},\qquad i=1,2,\ldots,m \tag{3.16}$$

where

$$u_i^{(k+1)}=(u_{i,1},u_{i,2},\ldots u_{i,n})^T,\quad g=(g_{i,1},g_{i,2},\ldots,g_{i,n})^T$$

$$g_{i,1}=\frac{\lambda}{2}u_{i+1,1,k+1/2}+(1-\lambda)u_{i,1,k+1/2}+\frac{\lambda}{2}(u_{i-1,1,k+1/2}+u_{i,o,k+1})$$

$$g_{i,j}=\frac{\lambda}{2}u_{i+1,1,k+1/2}+(1-\lambda)u_{i,j,k+1/2}+\frac{\lambda}{2}u_{i-1,j,k+1/2}\quad j=2,3,\cdots,n-1$$

$$g_{i,n}=\frac{\lambda}{2}u_{i+1,n,k+1/2}+(1-\lambda)u_{i,n,k+1/2}+\frac{\lambda}{2}(u_{i-1,n,k+1/2}+u_{i,n+1,k+1}) \tag{3.17}$$

B. IADE-DY

The matrices A and B are respectively tridiagonal of size (*mxm*) and (*nxn*). Hence, at each of the (*k* + ½) and (*k* + 1) time levels, these matrices can be decomposed into $G_1 + G_2 - \frac{1}{6}G_1G_2$, where G_1 *and* G_2 are lower and upper bidiagonal matrices given respectively by

$$G_1 = [l_i, 1], \quad \text{and} \quad G_2 = [e_i, u_i], \tag{3.18}$$

where

$$e_1 = \frac{6}{5}(a-1), u_i = \frac{6}{5}b,$$
$$e_{i+1} = \frac{6}{5}(a + \frac{1}{6}l_i u_i - 1),$$
$$l_i = \frac{6c}{6-e_i}(e_i \neq 6)\ i = 1,2,\cdots,m-1$$

hence, by taking *p* as an iteration index, and for a fixed acceleration parameter *r > 0*, the two-stage IADE-DY scheme of the form,

$$(rI + G_1)u^{(p+1/2)} = (rI - gG_1)(rI - gG_2)u^{(p)} + hf \quad \text{and}$$
$$(rI + G_2)u^{(p+1)} = u^{(p+1/2)} \tag{3.19}$$

can be applied on each of the sweeps (3.2) and (3.10). By carrying out the relevant multiplications in (3.19), the following equations for computation at each of the intermediate levels are obtained:

(i) at the $(p+1/2)^{th}$ iterate,

$$u_1^{(p+1/2)} = \frac{1}{\hat{d}}(s_1 \hat{s} u_1^{(p)} + w_1 \hat{s} u_2^{(p)} + hf_1)$$
$$u_i^{(p+1/2)} = \frac{1}{\hat{d}}(-l_{i-1}u_{i-1}^{(p+1/2)}v_{i-1}s_{i-1}u_{i-1}^{(p)} +$$
$$(v_{i-1}w_{i-1} + s_i \hat{s})u_i^{(p)} + w_i \hat{s} u_{i+1}^{(p)} + hf_i), \tag{3.20}$$
$$i = 2,3,\cdots,m-1$$
$$u_m^{(p+1/2)} = \frac{1}{\hat{d}}(-l_{m-1}u_{m-1}^{(p+1/2)}v_{m-1}s_{m-1}u_{m-1}^{(p)}$$
$$+ (v_{m-1}w_{m-1} + s_m \hat{s})u_m^{(p)} + hf_m)$$

Where,

$$g = \frac{6+r}{6},\ h = \frac{r(12+r)}{6},\ d = 1+r,$$
$$s = r - g,\ s_i = r - ge_i,\ i = 1,2,\cdots,m$$

and $v_i = -gl_i,\ w_i = -gu_i \quad i = 1,2,\ldots,m-1.$

(ii) at the $(p+1)^{th}$ iterate,

$$u_m^{(p+1)} = \frac{u_m^{(p+1/2)}}{d_m},$$
$$u_i^{(p+1)} = \frac{1}{d_i}(u_i^{(p+1/2)} - u_i u_{i+1}^{(p+1)}, \tag{3.21}$$
$$where\ d_i = r + e_i,\ i = m-1,\cdots,2,1$$

IV. Performance Analysis and Parallel Algorithm

All experiment were performed on the GCC of 8 nodes with Gigabit Ethernet interconnect. Each node consists of dual core processors (3.0GHZ) with 16 GB of RAM. The MPI implementation was implemented in C/MPI. A parallel platform design to run numerical application has to be efficient [8]. The platform contains more computations on large set of varying mesh sizes, and its evaluation has to be large to benchmarking. Performance concerns not only the cost of functions of the schemes, but resource accesses and code placement on computing resources [8]. Making declaration for placement of data at the beginning of computation, it does not accept any perturbation. The 2D IADE-DY scheme is extremely tested using the GCC cluster system for its implementation. The objective is to evaluate the overhead it introduces and its ability to exploit the inherent parallelism of an iterative computation as stated in [18]. The scalability across varying number of processors and mesh sizes is observed. To obtain any speedup we need convergence in fewer than N iterations. The closer the coarse propagator is to the fine propagation, the faster will be the convergence. If they are too similar, then the sequential part of the algorithm will significantly degrade the speedup. A simple speedup analysis according to [2] produces the following:

$$\varphi = \frac{N}{Nr(K+1) + K}, \tag{4.1}$$

Where *r* is the ration of the time taken by coarse propagation to fine propagation over the same time interval, *K* is the number of iterations required for convergence, and communication overhead is ignored.

In the limit $r \to 0$, $\varphi \to \frac{N}{K}$, therefore, the efficiency will be $\frac{1}{K}$. Full efficiency can be achieved if the algorithm converges in one iteration. To make *r* smaller, the coarse propagator must be less than accurate due to larger time step or coarse spatial grid, which in turn requires more iteration to converge [17]. As treated in [14, 15], the algorithm for the scheme is performed on a distributed memory system of p processors, assumes that each processors initially stores *n = N/p* objects distributed over the entire physical domain. In the

first iteration of the algorithm, the domain is decomposed into two sub-domains so that the difference between the sums of the weight of the sub-domain is as small as possible. Then the same process is applied to two sub-domains in parallel, and process is repeated recursively, for log p iteration. In other words, during iteration *i*, $1 \le i \le \log p$, the p processors are group into 2^{i-1} groups of $p/2^{i-1}$ processors each. At the beginning of the iteration, the problem domain is already partitioned into 2^{i-1} sub-domains and the objects in each sub-domain are stored in single group of processors. At the end of the iteration, each processor group is divided into two groups, and the corresponding sub-domain is divided into two sub-groups with the object in one sub-domain residing in one half the processors and the other objects in the other sub-domain residing in the other half of processor. Data parallelism originated the SPMD [23]. Thus, the finite difference approximation can be treated as a SPMD problem; essentially the same computation must be performed for multiple data sets. The multiple data are different parts of the overall grid, each sent to a different computer node (processor). The main issues that arise in parallelizing a finite difference grid are: the determination of how best to partitioned the grid among processors, and how to pass instructions about grid boundaries from node to node.

The domain decomposition is used to distribute data between different processors, in order to minimize the idle time static load balancing is used to distribute the data such that each processor gets almost the same number of computational points. The partitioning and load balancing is done in the pre-processing stage, wherein, separate grid files are generated for each processor along with other necessary information about partitioning. Thus there is no need to allocate any extra storage or scatter the grid data when the parallel program is executed. At the end of the parallel computation each process writes the output into separate files suitable for verification.

V. Numerical Experiments and Discussion

The algorithm was tested on the 2-D heat equation and the application of the above mentioned algorithm is now demonstrated on meshes of 100x100, 200x200 and 300x300 respectively. Tables 1 – 3, show the various performance timing. Consider the 2-D parabolic equation of the form:

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \frac{\partial U}{\partial t} \tag{5.1}$$

The boundary conditions and initial condition posed are:

$$\left.\begin{aligned} U(0,y,t) &= 0 \\ U(1,y,t) &= 100 \\ U(x,0,t) &= 0 \\ U(x,1,t) &= 100 \end{aligned}\right\} \quad t \ge 0 \tag{5.1a}$$

$$\begin{aligned} U(x,y,0) &= Sin(\pi x)Sin(\pi y) \\ (0 \le x \le 1,\ 0 \le y \le 1) \end{aligned} \tag{5.1b}$$

The exact solution is given by

$$U(x,y,t) = e^{-\pi t} Sin(\pi x)Sin(\pi y).$$

TABLE I. 100x100 meshes with MPI

Scheme	N	T_w	T_m	T_{sd}	S_{par}	E_{par}
	1	334.3	8.6	3.3	1.000	1.000
	2	285.6	8.5	3.2	1.516	0.758
	3	198.7	8.5	3.1	2.172	0.724
IADE	4	141.6	8.5	3.1	2.764	0.691
	5	128.1	8.5	3.1	3.165	0.633
	6	108.5	8.5	3.1	3.588	0.598
	7	98.1	8.5	3.1	3.787	0.541
	8	83.8	8.5	3.1	4.144	0.518

TABLE II. 200x200 meshes with MPI

Scheme	N	T_w	T_m	T_{sd}	S_{par}	E_{par}
	1	486.4	14.9	5.8	1.000	1.000
	2	392.8	14.7	5.6	1.782	0.891
	3	308.5	14.7	5.6	2.562	0.854
IADE	4	286.7	14.7	5.6	3.244	0.811
	5	203.1	14.7	5.6	3.91	0.782
	6	189.6	14.7	5.6	4.302	0.717
	7	163.5	14.7	5.6	4.795	0.685
	8	142.9	14.7	5.6	4.952	0.619

TABLE III. 300x300 meshes with MPI

Scheme	N	T_w	T_m	T_{sd}	S_{par}	E_{par}
	1	685.4	18.6	9.5	1.000	1.000
	2	536.8	18.5	9.3	1.798	0.899
	3	482.1	18.5	9.3	2.658	0.886
IADE	4	413.8	18.5	9.3	3.476	0.869
	5	386.9	18.5	9.3	4.24	0.848
	6	251.8	18.5	9.3	4.926	0.821
	7	210.1	18.5	9.3	5.572	0.796
	8	189.6	18.5	9.3	5.976	0.747

Here, we observed in our experiments designed to test the effectiveness of our approach that as the mesh size increases, the execution time increases as well with a proportionate decrease in time as processors increases for three mesh sizes in Tables 1 – 3. T_w is the time for the worker, T_m is the master time, T_{sd} is the worker domain decomposition time for worker allocation, S_{par} is the speedup, and E_{par} is the efficiency. This phenomenon shows that as the number of processors increases, though it might lead to a decrease in execution time but will get to a point that increasing the processors will not have much impact on total execution time. The time spend in data exchange will be significant compared to the time spend in computation and the parallel efficiency goes down. Hence, when the number of processors increases, balancing the number of computational cells per processors will become a difficult task due to significant load imbalance. When the number of processors increases, execution time suddenly increases for certain number of processors mesh sizes. This gain is due to the uneven distribution of the computational cells when a large number of processors are used, execution time had a very small change due to domain decomposition influence on performance in parallel computation. The larger the mesh sizes show that up to certain number, the speedup improvement is near linearity. The performance begins to degrade with an effect caused by increase in communication overheads.

The problem size is scaled up following the memory-bounded constraint. This phenomenon is well under expected since the implicit replacement has a very low computation overhead as implemented on the three problems. However, these jumps in communication time which are relatively larger than the others are mainly caused by the architecture of the communication between the processors, that is, due to the underlying machine architecture not the algorithm. This rate of performance decrease is fairly shown for parallel computing, especially for experiments conducted under non-dedicated environments which show that the proposed algorithm scales well. Our experiment shows reliability by conforming to convergence, and how memory is been distributed to access main data. This step is made possible by the Master/Worker computation process.

VI. Conclusion

We have explained in this paper how the role of GCC in the parallelization of the 2-D IADE-DY scheme is a good approach to solving problems, particularly when it is simulation with more processors. The objective is to present a design of paradigm adapted architecture for distributed computation, because they depend on empirical concern (data and code). The algorithm presented shows significant improvement when implemented on the above number of processors. In addition to the ease of use compared to other common approaches, the results shoe negligible overhead with effective load scheduling which produce the expected inherent speedups. It was also confirmed that the domain decomposition, and the use of SPMD are important and this is easy with our parallel platform of GCC. The performance of the 2D IADE-DY with the parallel paradigm is in many cases superior. As the number of processors increases, the bottleneck of parallel computation appears and the global reduction consumes a large part of time, then the improvement becomes significant.

References

[1] J. Aguilar, E. Leiss, 'Parallel loop scheduling approaches for distributed and shared memory system,' Parallel Process Letter 15 (1 – 2), 131 – 152, 2005.

[2] E. Aubanel, 'Scheduling of tasks in the parareal algorithm,' Parallel Computing 37 (3), 172 – 182, 2011.

[3] W. Barry, A. Michael, 'Parallel programming techniques and application using networked workstation and parallel computers,' Prentice Hall, New Jersey, 2003.

[4] D. Callahan, K. Kennedy, 'Compiling programs for distributed memory multiprocessors,' Journal of supercomputer 2, pp 151 – 169, 1988.

[5] E. Celledoni, T. Kvamsdal, 'Parallelization in time for thermo-viscoplastic problems in extrusion alluminium,' Int'l Journal for numerical methods in engineering 75 (5), 576 – 598, 2009.

[6] H. Chi-Chung, G. Ka-Kaung, 'Solving partial differential equations on a network of workstations,' IEEE, pp 194 – 200, 1994.

[7] P.J Coelho, M.G Carvalho, 'Application of a domain decomposition technique to the mathematical modeling of utility boiler' Journal of numerical methods in eng., 36 pp 3401 – 3419, 1993.

[8] D. Cyril, M. Fabrice, 'Jacobi computation using mobile agent,' Int'l Journal of Computer Science & Information Technologies, 1 (5), 392 – 401, 2010.

[9] D'Ambra P., M. Danelutto, S. Daniela, L. Marco, 'Advance environments for parallel and distributed applications: a view of current status,' Parallel Computing 28, pp 1637 – 1662, 2002.

[10] H.S Dou, Phan-Thien, 'A Domain decomposition implementation of the simple method with PVM,' Computational Mechanics 20 pp 347 – 358, 1997.

[11] F. Durst, M. Perie, D. Chafer, E. Schreck, 'Parallelization of efficient numerical methods for flows in complex geometries,' Flow simulation with high performance computing I, pp 79 – 92, Vieweg, Braunschelweig, 1993.

[12] J.H. Eduardo, M.A. Yero, H. Amaral, 'Speedup and scalability analysis of master-slave application,' 2007.

[13] D.J. Evans, M.S. Sahimi,'The alternating group explicit iterative method for parabolic equations I: 2-dimensional problems, *Intern. j. compt. math*, Vol. 24, (1988) pp. 311-341

[14] S. U. Ewedafe, R. H. Shariffudin, 'Armadillo generation distributed system with geranium cadcam cluster for solving 2-d telegraph problem,' Intern. j. compt. math, vol. 88, 589 – 609, 2011.

[15] S. U. Ewedafe, R. H. Shariffudin, 'Parallel implementation of 2-d telegraphic equation on MPI/PVM cluster,' Int. j. parallel prog, 39, 202 – 231, 2011.

[16] A. Fatoohi, E.G. Chester, 'Implementation of an ADI method on parallel computers,'' Journal of scientific computing 2 (2), 1987.

[17] M. J. Gander, S. Vandewall, 'Analysis of the parareal time-parallel time-integration method,' SIAM jour. on scientific computing 29 (2), 556 – 578, 2007.

[18] N. Giacaman, O. Sinnen, 'Parallel iterator for parallelizing object-oriented applications,' Intl journal of parallel programming, 39 (2) 223 – 269, 2011.

[19] W. Groop, E. Lusk, A. Skjellum, 'Using MPI, portable and parallel programming with the message passing interface,' 2nd Ed., Cambridge MA, MIT Press, 1999.

[20] Y. Guangwei, H. Xudeng, 'Parallel iterative difference schemes based on prediction techniques for Sn transport method,' Applied numerical mathematics 57, 746 – 752, 2007.

[21] M. Gupta, P. Banerjee, 'Demonstration of automatic data partitioning for parallelizing compilers on multi-computers,' IEEE trans. parallel distributed system, 3, vol. 2, pp 179 – 193, 1992a.

[22] K. Jaris, D.G. Alan, 'A High-performance communication service for parallel computing on distributed systems,' Parallel computing 29, pp 851 – 878, 2003.

[23] H. Laurant, 'A method for automatic placement of communications in SPMD parallelization,' Parallel computing 27, 1655 – 1664, 2001.

[24] J. L. Lions., Y. Maday, G. Turinki, 'Parareal in time discretization of PDE,' Comptes, rendus de lacadimie des sciences – series 1 – mathematics 332 (7), 661 – 668, 2011.

[25] A. R. Mitchell, G. Fairweather, 'Improved forms of the Alternating direction methods of Douglas, Peaceman and Rachford for solving parabolic and elliptic equations, 'Numer. maths, 6, 285 – 292, 1964.

[26] J. Noye, 'Finite difference methods for partial differential equations,' Numerical solutions of partial differential equations. North-Hilland publishing company, 1964.

[27] D.W Peaceman, H.H Rachford, 'The numerical solution of parabolic and elliptic differential equations,' Journal of soc. indust. applied math. 8 (1) pp 28 – 41, 1955.

[28] L. Peizong, Z. Kedem, 'Automatic data and computation decomposition on distributed memory parallel computers,' ACM transactions on programming languages and systems, vol. 24, number 1, pp 1 – 50, 2002.

[29] M. S. Sahimi, E. Sundararajan, M. Subramaniam, and N. A. A Hamid, 'The D'Yakonov fully explicit variant of the iterative decomposition method,' *International journal of computers and mathematics with applications*, 42, 1485 – 1496, 2001.

[30] V. T Sahni, 'Performance metrics: keeping the focus in routine. IEEE parallel and distributed technology, Spring pp 43 – 56, 1996.

[31] G.D Smith, 'Numerical solution of partial differential equations: finite difference methods 3rd Ed.,' Oxford oniversity press New York, 1985.

[32] X.H Sun, J. Gustafson, 'Toward a Better Parallel Performance Metric,' Parallel Computing 17, 1991.

[33] M. Tian, D. Yang, 'Parallel finite-difference schemes for heat equation based upon overlapping domain decomposition,' Applied maths and computation, 186, pp 1276 – 1292, 2007.