Parallel Domain Decomposition for 1-D Active Thermal Control Problem with PVM

Simon Uzezi Ewedafe
Department of Computing
The University of the West Indies,
Mona Kingston 7, Jamaica

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

***Abstract*—this paper describes a 1-D Active Thermal Control Problem (1-D ATCP) with the use of Stationary Iterative Techniques (Jacobi and Gauss-Seidel) on the discretization of the resulted matrices. Parallelization of the problem is carried out using Domain Decomposition (DD) parallel communication approach with Parallel Virtual Machine (PVM) to enable better flexibility in parallel execution, and greater ease of parallel implementation across the different domain of block sizes. We described the parallelization of the method by the use of Single Program Multiple Data (SPMD) technique. The 1-D ATCP is implemented on a parallel cluster (Geo Cluster), with the ability to exploit inherent parallelism of the computation. The parallelization and performance strategies are discussed, and results of the parallel experiments are presented.**

***Keywords*—1-D ATCP; Stationary Techniques; SPMD; DD; PVM**

I. INTRODUCTION

Researchers' efforts in parallel computing have focused on improving various aspects of communication performance in parallelism. The goal is to provide a study on the impact of communication performance for parallel applications. The focus is on high-performance cluster architecture, for which a fast active message-passing layer to a low latency, high bandwidth network is provided [20]. Applications are most sensitive to overhead, and some are hyper-sensitive to overhead in that the execution time increases with overhead at a faster rate that the message frequency would predict [2]. Developments in computer architectures have shown that parallel computer systems are a viable way to overcome major hardware design challenges relating to energy-consumption, and thermal constraints, while offering high computational peak performance [17]. Many applications are also sensitive to message to message rate, or to message transfer bandwidth, but the effect is less pronounced than with overhead. Cluster applications have more processor cores to manage, and exploit the computational capacity of high-end machines providing effective and efficient means of parallelism even as the challenges of providing effective resources management grows. It is a known fact that high capacity computing platform are expensive and are characterized by long-running, high processor-count jobs [3]. Developing parallel applications have its own challenges in the field of parallel computing. Regarding [12], there are theoretical challenges such as task decomposition, dependence analysis, and task scheduling.

Asynchronous message driven execution is a convenient and effective model for general purpose parallel programming. The flow of control in the message-driven systems is dictated by the arrival of messages. In Charm++ based message-driven applications [14], the problem to be solved is decomposed into collections of communicating parallel objects, providing the opportunity for easy overlap of communication with computation [7], and run time-level optimization such as automatic load balancing. In the loosely-coupled style, the assembly of separate parallel modules in a single application only requires interfaces, rather than the deeper structural changes or non-overlapped time, or processor division that might be required in the SPMD models [18]. In cases where data the concise expression of the algorithm, the code needed to explicitly communicate this data is a message-driven style, and this can dominate the structure of the program, and overwhelm the programmer as in [18]. For a global task with other processors relevant data needs to be passed from processor to a processor through a message-passing mechanism [19, 13], since there is greater demand for computational speed and the computations must be completed within reasonable time period. Design and analysis on finite difference DD for 2-D Heat Equations have been discussed in [24], and the parallelization for 3-DTEL on a parallel virtual machine with DD in [8] show effective load scheduling over various mesh sizes, which produce the expected inherent speedups.

This paper concern the estimated peak junction temperature of semiconductor devices during the manufacturing process, which is part of the thermal control systems [23] that treats the sequential algorithm of Parabolic Equation in solving thermal control process on printed circuit board. Temperature control of high-power microprocessor devices during testing is very important to determine the device performance [10]. The device manufacturer specifies a temperature and allowed deviation from it during the testing procedure (e.g., 85^oc – $o/+2^oc$). The test process applies computer-controlled electrical signals to the device, and for high-power devices, this may result in die-average power density in the range of $100KW/m^3$ [6]. High power microprocessor devices are also subjected to a classification test to determine the effective operating speed of the device. During this classification test, the goal of control is to keep the temperature of the die at a single set temperature where the device power is varied between 0% to 100% power in a predetermined test sequence [21]. An alternative approach to thermal management in automatic testing equipment was proposed and tested by [10]. In their approach, the surface of the device under test is heated with laser radiation while simultaneously cooled by forced convention. A significant complication in this scheme arises from the time required for temperature signal propagation from the device package surface to the die, upon which the power actually dissipated. However, [21] analyzed the effect of the conduction time lag on the control power for sinusoidal die power. Minimization of the required laser power, which can amount to hundreds of watts or more, is of great importance to limit both the electrical power consumed by the control system and the added load on the test facility's cooling system. Hence, [6] extended the analysis of [21] to multi-frequency wave forms, with the aid of determining optimal control power for multi-frequency test power sequences. They show that the profile control calculation with specified die temperature tolerance in [21] is not suitable for non-sinusoidal die power profiles, and they develop a new approach for the situation. All the high performance electronic devices are subjected to a 100% functional test prior to being shipped by the manufacturers [21].

The theoretical properties of the 1-D stationary techniques with the DD parallel communication approach with PVM to enable better flexibility in parallel execution, and greater ease of parallel implementation across the different domain of block sizes**,** and employing SPMD technique is emphasized in this paper. The 1-D ATCP is implemented on the Geo Cluster with the ability to exploit inherent parallelism. Previous works on 1-D stationary techniques on ATCP did not consider the parallel communication and DD approach on PVM to determine the temperature distribution. Our programming style allows the application programmer to specify the program organization in a clear and readable program code.

This paper presents a systematic methodology for investigating the effects of measuring variations in communication performance on the temperature distribution and in-depth study of application sensitivity to overhead and bandwidth in quantifying application performance in respond to changes in communication performance of the ATCP. In general, we use the above efforts to improve communication performance in Geo cluster architecture. Our results demonstrated flexibility in parallel execution, and greater ease of implementation. On the other hand, to improve the understanding of the stability of ATCP, this paper aims to confirm simulation to give increase confidence in the reality of the system. It also provides useful stationary techniques for evaluating thermal control problems even to higher dimensional problems.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 introduces the model for the 1-D ATCP and the stationary schemes. Section 4 introduces the parallel design and implementation details, including the PVM. Section 5 introduces the results of the experiments for the parallelization on Geo cluster. Section 6 gives the conclusion.

II. Related Work

The implementation of sequential algorithm in solving ATCP on printed circuit board with numerical finite difference method to design the discretization of the Partial Differential Equations (PDE) was implemented in [23]. The implementation design approaches are either passive or active controlled. The passive controlled design is suitable for low to medium power dissipation, while the active thermal control system is suitable for industrial processing, and testing of die. In testing package high-power integrated circuits, active thermal control is useful in providing die-level temperature stability [6]. They consider active test power sequences that contain many frequencies at various phase angles, each contributing to the temperature of the die. They develop a method of temperature variation of the die, and a method of controlling multiple frequency test sequences subject to a finite temperature tolerance. Sweetland et al. [21] presented an active thermal control of distributed-parameter system; with application to testing of packaged integrated circuit devices, requiring the die-level temperature be modulated to account for the distributed thermal capacitance and resistance of the device packaging. They also demonstrated fundamental limits of temperature control for typical devices under test conditions. Parallelization by time decomposition was first proposed by [16] with motivation to achieve parallel real-time solutions, and even the importance of loop parallelism, loop scheduling have been extensively studied [1]. Programming on heterogeneous many-core systems using explicit platform description to support programming by [17] constituted a viable approach for coping with power constraints in modern computer architectures, and can be formed across the whole computing landscape to high-end supercomputers and large-scale data centers. The stationary iterative techniques have been developed to solve various problems in PDEs [4], been widely used for solving algebraic systems resulting from the use of finite difference methods in several scientific and engineering applications. Chi-chung et al. [5] used a network of workstations as a single unit for speeding up computationally intensive applications as a cost-effective alternative to traditional parallel computers. The platform provided a general and efficient parallel solution for time-dependent PDE. However, [22] proposed an efficient parallel finite-difference scheme for solving Heat Equations numerically. They based it upon the overlapping DD method. Ewedafe and Rio [8] parallelized a 3-D ADI scheme using DD method with the SPMD technique on a MPI platform of clusters. On the same investigation, [9] used a numerical iterative scheme to solve 2-D Bio-Heat on MPI/PVM cluster systems with the SPMD technique. The Geo cluster are designed for application running on distributed memory clusters which can dynamically and statically calculate partition sizes based on the run-time performance of applications. We use the stationary iterative techniques (Jacobi and Gauss-Seidel) on the resulted matrices from the discretization of the 1-D ATCP model. Parallelization of the problem is carried out using DD parallel communication approach with PVM. The parallelization strategy and performance are discussed, and results of the parallel experiments are presented.

III. The Model Problem

The mathematical model for the 1-D ATCP follows the 1-D model of [21]. For the transient response, the physical model of the device is reduced to 1-D model of the form:

$$\frac{\partial^2 v(x,t)}{\partial x^2} = \frac{1}{(b_t)}\frac{\partial v(x,t)}{\partial t} \tag{3.1}$$

where b_t is the thermal diffusivity (that measure the ability of a material to conduct thermal energy relative to its ability to store thermal energy), and $b_t = \alpha = \dfrac{k}{\rho c_p}$,

and k is the thermal conductivity $(W/(m.k))$, ρ is the density (kg/m^3), and c_ρ is the specific heat capacity $(J/(kgk))$ while ρc_p together can be considered the volumetric heat capacity $(J/(m^3k))$. Hence,

$$\begin{gathered} b_t\frac{\partial^2 v(x,t)}{\partial x^2} = \frac{\partial v(x,t)}{\partial t} \\ \text{let } v(x,t) = V, \text{then we have} \\ b_t\frac{\partial V}{\partial x^2} = \frac{\partial V}{\partial t} \end{gathered} \tag{3.2}$$

We can then solve (3.2) by extending the 1-D explicit finite difference method to the above, Eq. (3.2) becomes:

$$\begin{gathered} \frac{\partial V}{\partial t} = V_t = \frac{V_{i,j+1} - V_{i,j}}{\Delta t} \\ \frac{\partial^2 V}{\partial x^2} = V_{xx} = \frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{(\Delta x)^2} \end{gathered} \tag{3.3}$$

applying eq. (3.3) on Eq. (3.2), the temperature of the explicit node is given by:

$$\frac{V_{i,j+1} - V_{i,j}}{\Delta t} = b_t\left(\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{(\Delta x)^2}\right)$$

rearranging we have:

$$V_{i,j+1} = \frac{bt\Delta t}{(\Delta x)^2}\left(V_{i+1,j} - V_{i-1,j}\right) + \left(1 - 2b_t\frac{\Delta t}{(\Delta x)^2}\right)V_{i,j} \tag{3.4}$$

although an implicit scheme of the Crank-Nicolson (C-N) unconditionally stable scheme cab be applied to Eq. (3.2) as seen below.

A. C-N Implicit Scheme on 1-D ATCP

When the C-N implicit scheme is used we write using the same finite difference scheme:

$$\frac{V_{i,j+1} - V_{i,j}}{\Delta t} = \frac{b_t}{2}\left(\frac{\begin{gathered}V_{i+1,j+1} - 2V_{i,j+1} + V_{i-1,j+1} \\ + V_{i+1,j} - 2V_{i,j} + V_{i-1,j}\end{gathered}}{(\Delta x)^2}\right) \tag{3.5}$$

and rearranging to give:

$$\begin{gathered} -b_t\frac{\Delta t}{(\Delta x)^2}V_{i-1,j+1} + \left(2 + b_t\frac{\Delta t}{(\Delta x)^2}.2\right)V_{i,j+1} \\ -b_t\frac{\Delta t}{(\Delta x)^2}V_{i+1,j+1} = b_t\frac{\Delta t}{(\Delta x)^2}V_{i-1,j} + \\ \left(2 - b_t\frac{\Delta t}{(\Delta x)^2}.2\right)V_{i,j} + b_t\frac{\Delta t}{(\Delta x)^2}V_{i+1,j} \end{gathered} \tag{3.6}$$

let $b_t\dfrac{\Delta t}{(\Delta x)^2} = r$, then we have:

$$\begin{gathered} -rV_{i-1,j+1} + (2+2r)V_{i,j+1} - rV_{i+1,j+1} = \\ rV_{i-1,j} + (2-2r)V_{i,j} + rV_{i+1,j} \end{gathered} \tag{3.7}$$

here, we have a tridiagonal system of equations which can be solved with the stationary iterative techniques.

B. Stationary Techniques on 1-D ATCP

A system of linear algebraic equations can be sparse and banded. We will typically employ the concise notation

$$Au = b \tag{3.8}$$

to represent such systems and the focus of this section is the study of methods for efficiently solving equation (3.8) on parallel computers. We begin with the decomposition

$$A = D - E - F, \tag{3.9}$$

in which D is the diagonal of A, $-E$ is the strict lower part and $-F$ is the strict upper part. It is always assumed that the diagonal entries of A are all nonzero. The Jacobi iteration determines the ith component of the next approximation so as to annihilate the ith component of the residual vector.

Thus,

$$(b - Ax_{k+1}) = 0 \tag{3.10}$$

however, recall Eq. (3.8) and note that iterative methods for solving this system of linear equations can essentially always be expressed in the form:

$$U^{(n+1)} = GU^{(n)} + K \tag{3.11}$$

where n is an iterative counter and G is the iteration matrix, it is related to the system matrix A by

$$G = I - Q^{-1}A$$

where I is the identity matrix and Q is generally called the splitting matrix. The Jacobi scheme can be constructed as follows. Firstly, decompose A as in Eq. (3.9), substitute into (3.8) to obtain:

$$(D-L-U)U=b, \ or \ DU=(L+U)U+b \qquad (3.12)$$

hence, introducing iteration counter, (3.12) becomes

$$U^{(n+1)}=D^{-1}(L+U)U^{n}+D^{-1}b \qquad (3.13)$$

from Eq. (3.13) $L+U=D-A$, so
$D^{-1}(L+U)=I-D^{-1}A.$

Thus, D is the splitting matrix and Eq. (3.13) is in the form (3.11) with

$$G\equiv D^{-1}(L+U)=1-D^{-1}A, \ k=D^{-1}b \qquad (3.14)$$

hence, in matrix terms the definition of the Jacobi method can be expressed as

$X^{(k+1)}=D^{-1}(L+U)x^{(k)}+D^{-1}b$ as in Eq.(3.13)

where

$$x_i=\frac{1}{a_{i,j}}(b_i-\sum_{j\neq i}a_{i,j}x_j) \qquad (3.15)$$

suggesting an iterative method defined by

$$x_i^{(k+1)}=\frac{1}{a_{i,j}}(b_i-\sum_{j\neq i}a_{i,j}x_j^{k}) \qquad (3.16)$$

Consider again the linear equations; if we proceed as with the Jacobi method but not assume that the equations are examined one at a time in sequence, and that previously computed results are used as soon as they are available we obtain the Gauss-Seidel (GS):

$$X_i^{(k)}=\frac{1}{a_{i,j}}\left[b_i-\sum_{j<i}a_{i,j}x_j^{(k)}-\sum_{j>i}a_{i,j}x_j^{(k-1)}\right] \qquad (3.17)$$

the computation appear to be serial, since each component of the new iterate depends upon all previously computed components. The update cannot be done simultaneously as in the Jacobi method. Secondly, the new iterate $x^{(k)}$ depends upon the order in which the equations are examined. The GS is called the "Successive Displacement" to indicate the dependence of iterates on the ordering. A poor choice of ordering can degrade the rate of convergence.

IV. Parallel Implementation, Design and Analysis

A. The Parallel Platform

The implementation was done on Geo Cluster consisting of 16 Intel Celeron CPU J 1900 at 1.99GHz quad core, and 495.7GB of Disk type. PVM [11] is a software system that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resource. It supports program executed on each machine in a user-configurable pool, and present a unified, general, and powerful computational environment for concurrent applications. The program, written in Fortran, C, or C++, are provided access to PVM through calling PVM library routines for functions such as process initiation, message transmission and reception, and synchronization via barriers or rendezvous. PVM is ideally suited for concurrent applications composed o many interrelated parts, and is very useful for the study of large-scale parallel computation.

B. Parallel Implementation and DD

Partitioning strategy simply divides the problem into parts. Most partitioning formulations, however, require the results of the parts to be combined to obtain the desired result. Partitioning can be applied to the program data i.e. dividing the data and operating upon the divided data concurrently. This is called data partitioning or DD. Decomposition into fixed chunks per processing element. When the domain is splinted, each block is given an identification number by a "master" task, which assigns these sub-domains to "slave" tasks running in individual processors. In order to couple the sub-domains' calculations, the boundary data of neighboring blocks have to be interchanged after each iteration. The calculations in the sub-domains use the old values at the sub-domains' boundaries as boundary conditions. DD is used to distribute data between different processors; the static load balancing is used to maintain same computational points for each processor. Data parallelism originated the SPMD [15], thus, the finite difference approximation used in this paper can be treated as an SPMD problem. The SPMD model contains only a single program with multiple data and each process will execute the same code. To facilitate this within a single program, statements need to be inserted to select which portions of the code will be executed by each processor. The copy of the program is started by checking pvm_parent, it then spawns multiple copies of itself and passes then the array of tids. At this point, each copy is equal and can work on its partition of the data in collaboration with other processes. In the master model, the master program spawns and direct a number of slave program which perform computations. Any pvm task can initiate processes on the machine. The master calls pvm_mytid, which as the first pvm call, enrolls this task in the pvm system. It then calls pvm_spawn to execute a given number of slave programs on other machines in pvm. Each slave calls pvm_tid to determine its task id in the virtual machine, and then uses the data broadcast from the master to create a unique ordering from 0 to nproc minus 1. Subsequently, pvm_send and pvm_recv are used to pass messages between processors. When finished, all pvm programs call pvm_exit to allow pvm to disconnect any sockets to the process and keep track of which processes are currently running. A master program wakes up worker programs, assign initial data to the workers and let them work, receive results from workers, update and display them. The worker program works like this: receive initial data from master, exchange the edges data with the next door workers, and send the result to the master.

C. Parallel Communication with PVM

The activities of communication are slow compare to computation, often by order of magnitude. The communication costs are measured in terms of latency and bandwidth. The communication costs comprised of the physical transmission costs and the protocol overhead. The overhead is high in heterogeneous networks, where data may have to be converted to another format. The communication cost can be reduced but not avoided. The non-blocking communication is employed across SPMD to reduce the problem of blocking. Factors considered in communication include; Cost of communications which implies overhead and required synchronization between tasks, latency versus bandwidth, synchronization communication, and efficiency of communication. We have our grid distributed in a block fashion across the processors, the values for the ghost cells are calculated on neighboring processors and sent using PVM class. Synchronization is the coordination of parallel tasks associated with communication. A routine that returns when the transfer has been completed is a synchronous message passing and performs two actions: transferring of data then synchronous processes through send / receive operations

D. Speedup and Efficiency with Effectiveness

Speed-up and efficiency are commonly used to measure the performance of a parallel code. The runtime of the original serial code is used as a measured of the runtime on one processor. In this context, runtime can be defined as the time that has elapsed from the first moment when the first processor actually begins execution of the program to the moment when the last process executes its last statement. In this present code, time is measured after the initialization of PVM, and before the domain decomposition. The time measurement ends after the writing of the last result, just before finalizing PVM. Only the timing of the day is considered. The parallel Speed-up (Sp) is the ratio of the runtime on one processor t1 to the runtime on P processor tp.

$$Sp = \frac{t1}{tp} \tag{4.1}$$

The parallel efficiency Ep is the ratio of the parallel Speed-up Sp to the number of processors P.

$$Ep = \frac{Sp}{p}, \tag{4.2}$$

another intuitive aspect for the performance but also a value easy to obtain for optimizing the efficiency of the parallel code is the communication time.

Hence, effectiveness is given as

$$L_n = S_n/(nT_n) = E_n/T_n = E_n S_n/T_1 \tag{4.3}$$

which clearly shows that L_n is a measure of both speedup and efficiency. Therefore, a parallel algorithm is said to be effective if it maximizes L_n and hence $L_n T_1 = S_n E_n$.

V. Results and Discussion

Consider the Telegraph Equation of the form:

$$\frac{\partial^2 v(x,t)}{\partial x^2} = \frac{1}{(b_t)}\frac{\partial v(x,t)}{\partial t} \tag{5.1}$$

the boundary condition and initial condition posed are:

$$\begin{aligned} &V(0,t) = h(t), \\ &V(1,t) = k(t), \qquad 0 \le t \le T \\ &\frac{\partial V}{\partial x}(x,0) = f(t), \quad 0 \le t \le 1 \\ &V(x,0) = f(x)\ (0 \le x \le 1) \end{aligned} \tag{5.2}$$

A. Parallel Efficiency

The speedup and efficiency obtained for various sizes, for 100 mesh size to 300 mesh size, are listed in Tables I to III, and the effectiveness of the model is shown in Table IV. The Tables show the parallel time decreasing as the number of processors increase for using the stationary iterating schemes. The speedup and efficiency versus the number of processors are shown in Fig. 1 and Fig. 2, respectively. The results in the Tables show that the parallel efficiency increases with increasing grid size, and decreases with the increasing block number for given grid size. As the number of processors increase, though this leads to a decrease in execution time, but a point is reached when the increased processors will not have much impact on total execution time. Hence, when the number of processor increase, balancing the number of computational cells per processors will become a difficult task due to significant load imbalance. Performance begins to degrade with an effect caused by the increase in communication overhead as the mesh increases. The gain in increasing execution time for certain mess sizes is due to uneven distribution of the computational cell, and the execution time has a very small change due to DD influence on performance in parallel computation.

TABLE I. The Total Time T, The Parallel Speed-Up Spar and the Efficiency Epar for a Mesh of 100

Scheme	*N*	*S_{par}*	*E_{par}*
	1	*1.000*	*1.000*
	2	*0.953*	*0.477*
	6	*1.048*	*0.175*
Jacobi	*8*	*1.296*	*0.162*
	10	*1.564*	*0.156*
	12	*1.831*	*0.153*
	14	*2.094*	*0.150*
	16	*2.313*	*0.145*
	1	*1.000*	*1.000*
	2	*1.108*	*0.554*
	6	*1.216*	*0.203*
GS	*8*	*1.397*	*0.175*
	10	*1.645*	*0.165*
	12	*2.109*	*0.176*
	14	*2.271*	*0.162*
	16	*2.526*	*0.158*

TABLE II. The Total Time *T*, the Parallel Speed-Up *S_{PAR}* and the Efficiency *E_{PAR}* for a Mesh of 200

Scheme	*N*	*S_{par}*	*E_{par}*
	1	*1.000*	*1.000*
	2	*1.074*	*0.537*
	6	*1.265*	*0.211*
Jacobi	*8*	*1.532*	*0.192*
	10	*1.848*	*0.185*
	12	*2.226*	*0.186*

	14	*2.584*	*0.185*
	16	*2.928*	*0.183*
	1	*1.000*	*1.000*
	2	*1.205*	*0.603*
	6	*1.328*	*0.221*
GS	*8*	*1.612*	*0.202*
	10	*1.924*	*0.192*
	12	*2.497*	*0.208*
	14	*2.703*	*0.193*
	16	*3.034*	*0.190*

TABLE III. THE TOTAL TIME *T*, THE PARALLEL SPEED-UP S_{PAR} AND THE EFFICIENCY E_{PAR} FOR A MESH OF 300

Scheme	*N*	S_{par}	E_{par}
	1	*1.000*	*1.000*
	2	*1.348*	*0.674*
	6	*1.665*	*0.278*
Jacobi	*8*	*2.038*	*0.255*
	10	*2.606*	*0.261*
	12	*3.115*	*0.260*
	14	*3.665*	*0.262*
	16	*4.190*	*0.262*
	1	*1.000*	*1.000*
	2	*1.441*	*0.721*
	6	*1.813*	*0.302*
GS	*8*	*2.176*	*0.272*
	10	*2.747*	*0.275*
	12	*3.419*	*0.285*
	14	*3.857*	*0.276*
	16	*4.317*	*0.270*

TABLE IV. EFFECTIVENESS OF THE VARIOUS SCHEMES WITH PVM FOR 300 MESH SIZE

	N	*PVM T(s)*	L_n
	1	*178.5*	*0.56*
	2	*132.4*	*0.51*
Jacobi	*6*	*107.2*	*0.26*
	8	*87.6*	*0.29*
	10	*68.5*	*0.38*
	12	*57.3*	*0.45*
	14	*48.7*	*0.54*
	16	*42.6*	*0.55*
	1	*213.7*	*0.47*
GS	*2*	*148.3*	*0.46*
	6	*117.9*	*0.26*
	8	*98.2*	*0.28*
	10	*77.8*	*0.35*
	12	*62.5*	*0.46*
	14	*55.4*	*0.50*
	16	*49.5*	*0.55*

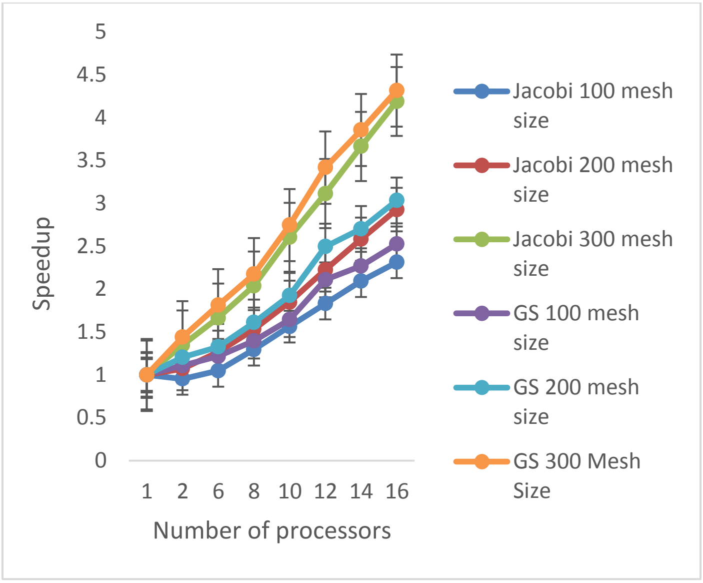

Fig. 1. Speedup versus the number of processors for mesh 100,200 and 300

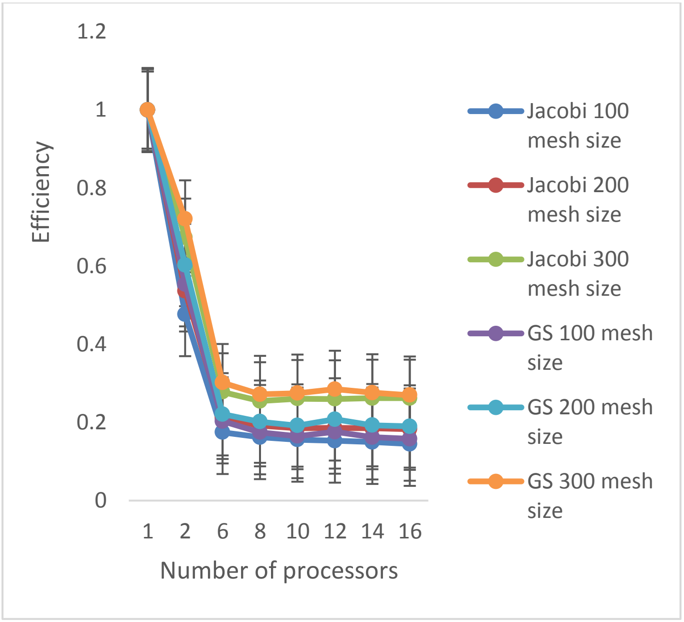

Fig. 2. Parallel efficiency versus the number of processors for mesh 100, 200 and 300

The numerical efficiency includes the DD efficiency and convergence rate behavior. The DD efficiency includes the increase of floating point operations induced by grid overlap at interfaces and the CPU time variation generated by DD techniques. , the performance is strongly dependent on the load condition of the nodes, and the distributed platform allows efficient overlapping of computation and communication. It is pointed out that when the computational domain are partitioned over the processors, especially for small domains, the execution time and speed-up seems not noticeable, but with larger mesh sizes the effect of parallelization become noticeable. The result presented in the Tables and Figures show that PVM is both powerful and economically distributed processing tool.

VI. CONCLUSION

In this paper, we consider the combination of the stationary iterative techniques on the resulted matrices from the discretization process of the 1-D ATCP model with the SPMD parallel domain decomposition technique, which sacrifices some flexibility of a parallel platform with the general-purpose message-driven execution. To demonstrate the use of the stationary techniques on the above-mentioned model, we present example of the model with initial and boundary conditions implemented on different mesh sizes. The temperature control in a distributed parameter thermal system has been extended to parallel implementation on cluster systems in one dimension. The results presented in this paper show the study of the parallel domain decomposition on a parallel cluster and analysis for the above model with PVM. The system allows a parallel collection of overlapping communication. Computational results obtained have clearly shown the benefits of parallelization. The DD greatly influences the performance of the 1-D ATCP model on the parallel clusters. However, we are interested in improving the algorithms used in this paper, and also to test the improved algorithms on increase number of processors thereby exploits the communication of data transfer across multiple clusters.

VII. FUTURE WORK

The parallel domain decomposition for 1-DATCP with the use of Stationary Iterative Techniques on Geo Cluster Systems using PVM employing the SPMD technique has been carried out. This paper has shown the ability to exploit inherent parallelism of the computation. We suggest future work to be carried out on the 1-D ATCP model employing the used of Iterative Alternating Direction Implicit (IADE) method. Parallel implementation on the scheme could use the Input File Affinity Measure on a tightly coupled and loosely coupled distributed environment with dynamic allocation of a task with varying mesh sizes. Another avenue for improvement would be to make the parallel implementation adapt a unique pattern of predictable based knowledge of the algorithms on problem domain in question.

ACKNOWLEDGMENTS

The University of the West Indies Mona Campus awarded the Initiative Grant that motivated this study.

REFERENCES

[1] J. Aguilar, E. Leiss, 'Parallel Loop Scheduling Approaches for Distributed and Shared Memory System', Parallel Process Letter 15 (1 – 2), 2005, pp. 131 – 152

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

[3] W. Barry, A. Michael, '*Parallel Programming Techniques and Application using Networked Workstation and Parallel Computers*' 2003, Prentice Hall, New Jersy

[4] R. L. Burden, J. D. Faires 'Numerical Analysis 5th ed.' PWS Publishing Company, Boston, 1993

[5] H. Chi-chung, Ka-keung C., G. Man-Sheung Yuen, M. Hamdi, Solving PDE on Network of Workstation. IEEE, 1994, pp194 – 200

[6] C. R. Christopher, J. H. Lienhard, 'Active Thermal Control of Distributed Parameter Systems Excited at Multiple Frequencies' Journal of Heat Transfer, 2005

[7] S. U. Ewedafe, H. S. Rio, 'Armadillo Generation Distributed Systems & Geranium Cadcam Cluster for solving 2-D Telegraph Equation' Int'l Jour. of Computer Mathematics, 88, Issue 3, 2011, 589 – 609

[8] S. U. Ewedafe, H. S. Rio, 'On the Parallel Design and Analysis for 3-D ADI Telegraph Problem with MPI' Int'l Jour. Of Advanced Compt. Sci. and Applications, Vol. 5, No. 4, 2014

[9] S. U. Ewedafe, H. S. Rio, 'Domain Decomposition of 2-D Bio-Heat Equation on MPI/PVM Clusters' Int'l Journal of Emerging Trends of Technology in Compt. Science, Vol. 3, Issue 5, 2014

[10] P. Fahnl, A. C., Lienhard V., J. H, A. H. Slocum 'Thermal management and Control in Testing Packaged Integrated Circuit (IC) Devices' Proc. 34th Inter Society Energy Conversion Conference' 1999

[11] A. Geist, A. Beguelin, J. Dongarra, 'Parallel Virtual Machine (PVM)' Cambridge MIT Press

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

[13] K. Jaris, D.G. Alan, 'A High-Performance Communication Service for Parallel Computing on Distributed System', Parallel Computing 29, 2003, pp 851 – 878

[14] L. V. Kale, S. Krishnan, 'Charm++ Parallel Programming with Message-Driven Objectives, in: G. V. Wilson, P. Lu (Ed.)' Parallel Programming using C++, MIT Press, 1966, pp 175 - 213

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

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

[17] S. Martin, S. Benkner, S. Pllans, 'Using Explicit Parallel Description to Support Programming of Heterogeneous Many-Core Systems', Parallel Computing 38 (2012), pp 52 – 65

[18] P. Miller, A. Becker, L. Kale, 'Using Shared Arrays in Message-Driven Parallel Programs' Parallel Computing 38 (2012), pp 66 - 74

[19] Peizong L., Z. Kedem, 'Automatic Data and Computation Decomposition on Distributed Memory Parallel Computers' ACM Transactions on Programming Languages and Systems, vol. 24, number 1, 2002, pp 1 – 50

[20] P. Richard, M. Amin, E. David, E. Thomas, *'The work of Parallelism*' Computer Science University of California Berkeley CA 94720

[21] M. Sweetland, V. J. H. Lienhard, 'Active Thermal Control of Distributed Parameter System with Application to Testing of Packaged (IC) Devices' ASME Journal of Heat Transfer, 2003, pp 165 – 174

[22] M. Tian, D. Yang, 'Parallel Finite Difference Schemes for Heat Equations Based on Overlapping Domain Decomposition", Applied Maths & Compt, Issue 18, 2007, pp 1276 – 1292

[23] S. Zarith, A. Ghaffar, N. Alias, F. Sham, A. Hassan, H. Hassan, 'Sequential Algorithms of Parabolic Equation in Solving Thermal Control Process on Printed Circuit Board' Jour. Fundamental Science Issue 4, 2008, pp 379 - 385

[24] W. Zheng-Su, Z. Baolin, C. Guang-Nan, 'Design and analysis for finite difference DD for 2-D heat equation', ICA3PP – 02, IEEE Computer Society