On the parallelization of bio-heat transfer problem using input file affinity measure with MPI

S. U. Ewedafe[1] & R. H. Shariffudin[2]
[1]*Department of Computing, Baze University, Abuja, Nigeria*
[2]*Faculty of Science, Institute of Mathematical Sciences, University of Malaya, Malaysia*

Abstract

In this paper, a method based on applying Successive Over-Relaxation (SOR) to find the solution of 2-Dimensional Bio-Heat Transfer Problem (2DBHTP) on a distributed computing environment using Input File Affinity Measure (I_{aff}) with Message Passing Interface (MPI) is introduced. A finite difference discretization is developed to solve the 2DBHTP since 2DBHTP requires the evaluation of temporal and spatial distributions of temperature. The platform gives us better predictions of the effects of thermal physical properties of temperature distribution. This work proposes a parallel overhead with overlapping communication in its implementation using MPI. Based on the implementation of the MPI distributed computing environment, we present a performance improvement strategy running on the master-slave platform. Parallel comparisons of the method were experimentally evaluated, and parallel results show speedup and efficiency on a various number of mesh sizes. The method used combines elements of numerical stability and parallel algorithm design.
Keywords: 2DBHTP, SOR, I_{aff}, MPI, speedup, discretization, finite difference, spatial temperature, parallel overhead, performance.

1 Introduction

Computing infrastructures are reaching an unprecedented degree of complexity. First, parallel processing is coming to mainstream, because of the frequency and power consumption wall that leads to the design of multi-core processors. Second, there is a wide adoption of distributed processing technologies because of the deployment of the Internet and consequently large-scale grid and cloud infrastructures. All these combined together make programming of computing infrastructure a very difficult challenge. Programmers have to face both parallel and distributed programming paradigms when designing an application and several software codes that are executed on various computing resources spread over the Internet within a grid or cloud-base infrastructure [1]. In the world of parallel computing MPI is the de facto standard for implementing programs on multiprocessors. To help with program development under a distributed computing environment a number of software tools have been developed. MPI [2] is chosen here for the parallelization. Many applications are “embarrassingly” parallel and require minimal performance out of MPI. These applications exploit coarse grain parallelism and communicate rarely.

However, program development for distributed memory parallel computers is time-consuming and error prone, as the programmer is forced to manage both parallelism and communication. Distributed systems can increase application performance by a significant amount and the incremental enhancement of a network-based concurrent computing environment is usually straightforward because of the availability of high bandwidth networks [3, 4]. The natural programming style under a distributed system is therefore the Multiple Instructions Multiple Data (MIMD) [3–6]. The basic idea is to split a program into a few smaller tasks and to allocate these tasks to several processors to be executed simultaneously. Thus the total execution time can be reduced to just a fraction of that of a uniprocessor computer.

Studying Bio-heat transfer in the human body has been a hot topic and is useful for designing clinical thermal treatment equipments, for accurately evaluating skin burn and for establishing thermal protections for various purposes. The Bio-heat transfer is the heat exchange that takes place between the blood vessels and the surrounding tissues. Monitoring the blood flow using the techniques has great advantage in the study of human physiological aspects. This requires a mathematical model which relates the heat transfer between the perfuse tissue and the blood. The theoretical analysis of heat transfer design has undergone a lot of research over the years, from the popular Penne’s bio-heat transfer equation proposed in 1948 to the latest one proposed by Deng and Liu [7]. Many of the Bio-heat transfer problem by Penne’s account for the ability of the tissue to remove heat by diffusion and perfusion of tissue by blood. Predictions of heat transport have been carried out by Chinmay [8] and Liu and Xu [9]. The major concern in the modeling of the Bio-heat is the accurate continuum representation of the heat transfer in the living tissue incorporating the effect of blood flow, due to the presence of two heat sources. Penne’s assumes that for heat transfer to take place there must be two heat sources, as in heat produced by the metabolism and the heat transfer from the blood flow surrounding tissue at each point of the forearm. Effects of thermal properties and geometrical dimensions on the skin burn injuries have been discussed. [9, 10] proposed a comparison of 1D and 2D programmed for predicting the state of skin burn. Liu *et al.* [11] used a finite difference method to solve the BHTP − a triple-layered skin structure composed of epidermis, dermis, and subcutaneous.

Dai and Zhang [12] developed a three-level unconditional stable finite difference scheme and used a domain decomposition strategy for solving the 1-D BHTP for the same three-layered skin structure.

In this paper, a method based on applying SOR to find the solution of 2DBHTP on a distributed computing environment using I_{aff} with MPI is implemented. We also assess parallel performance improvement in terms of speedup and efficiency. The rest of the paper is organized as follows: Section 2 introduces the model for the 2DBHTP. Section 3 gives the parallel implementation using I_{aff}. Section 4 introduces the results and discussion. Finally, a conclusion is included in Section 5.

2 Bio-heat model for 2-d

Modern clinical treatments and medicines such as cryosurgery, cryopreservation, cancer hyperthermia, and thermal disease diagnostics, require the understanding of thermal life phenomena and temperature behaviour in living tissues [13]. The well-known Penne's equation and the energy balance for a control volume of tissue with volumetric blood flow and metabolism yields the general Bio-heat transfer equations. ρ , c_p are densities and specific heat of tissue, ω_b and c_b are blood perfusion rate and specific heat of blood, q_m''' is the volumetric metabolic heat generation, U_a is the arterial temperature, U is the nodal temperature. The bio-heat problem is given as:

$$\rho c_p \frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \omega_b c_b (U_a - U) + q_m''', \quad 0 \le x \le 1, 0 \le y \le 1, t > 0$$

$$\frac{\partial U}{\partial t} = \frac{1}{pc_p}\frac{\partial^2 U}{\partial x^2} + \frac{1}{\rho c_p}\frac{\partial^2 U}{\partial y^2} + \frac{\omega_b c_b}{\rho c_\rho} U_a - \frac{\omega_b c_b}{\rho c_\rho} U + \frac{q_m'''}{pc_p}, \quad (1)$$

This further simplifies into the form:

$$\frac{\partial U}{\partial t} = \frac{1}{pc_p}\frac{\partial^2 U}{\partial x^2} + \frac{1}{\rho c_\rho}\frac{\partial^2 U}{\partial y^2} - \frac{\omega_b c_b}{\rho c_\rho} U + \frac{\omega_b c_b}{\rho c_\rho}(U_a + \frac{q_m'''}{\omega_b c_b}) \quad (2)$$

Assume q_m''' to be constant, and denote $U^{\oplus} = U_a + \frac{q_m'''}{\omega_b c_b}$, $b = \frac{\omega_b c_b}{\rho c_p} (> 0)$ and $c = \frac{1}{\rho c_p} (> 0)$. We can obtain the simplified form of the 2-D Penne's equation with the initial and boundary conditions given below:

$$\frac{\partial U}{\partial t} = c\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) - bU + bU^{\oplus}, \quad (3)$$

with initial condition

$$U(x, y, 0) = f(x, y) \tag{4}$$

and boundary conditions

$$\left.\begin{array}{l} U(0, y, t) = f_1(y, t),\ U(1, y, t) = f_2(y, t) \\ U(x, 0, t) = f_3(x, t),\ U(x, 1, t) = f_4(x, t) \end{array}\right\} \tag{5}$$

When the explicit scheme is used, we write using the same finite-difference scheme:

$$\frac{U_{i,j}^{n+1} - U_{i,j}^{n}}{\Delta t} = c\left(\frac{U_{i+1,j}^{n} - 2U_{i,j}^{n} + U_{i-1,j}^{n}}{\Delta x^2} + \frac{U_{i,j+1}^{n} - 2U_{i,j}^{n} + U_{i,j-1}^{n}}{\Delta x^2}\right) - bU_{i,j}^{n} + bU^{\oplus},$$

The temperature of the node in the scheme formulation takes the form:

$$\begin{aligned} U_{i,j}^{n+1} = c\frac{\Delta t}{\Delta x^2}\left(U_{i+1,j}^{n} + U_{i-1,j}^{n} + U_{i,j+1}^{n} + U_{i,j-1}^{n}\right) - \\ \left(1 - 4c\frac{\Delta t}{\Delta x^2} - b\Delta t\right)U_{i,j}^{n} + b\Delta t U^{\oplus} \end{aligned} \tag{6}$$

2.1 Stationary methods for 2-d bio-heat equation

If we use the central differences for both U_{xx} and U_{yy}, and the forward difference for U_t, into (3) and let $\Delta x^2 = \Delta y^2 = \Delta^2$ we have:

$$\begin{aligned} U_{i,j}^{n+1} = U_{i,j}^{n} + \frac{c\Delta t}{\Delta^2}\left(U_{i+1,j}^{n} + U_{i-1,j}^{n} + U_{i,j+1}^{n} + U_{i,j-1}^{n} - 4U_{i,j}^{n}\right) \\ - b\Delta t U_{i,j}^{n} + b\Delta t U^{\oplus} \\ i = 1\ldots, n,\ j = 1\ldots, m \end{aligned} \tag{7}$$

Let $\frac{c\Delta t}{\Delta^2} = Fo$, hence

$$\begin{aligned} U_{i,j}^{n+1} = U_{i,j}^{n} + Fo\left(U_{i+1,j}^{n} + U_{i-1,j}^{n} + U_{i,j+1}^{n} + U_{i,j-1}^{n} - 4U_{i,j}^{n}\right) \\ - b\Delta t U_{i,j}^{n} + b\Delta t U^{\oplus} \end{aligned}$$

It is stable in one spatial dimension (1-D) only if $\Delta t / \Delta^2 \le 1/2$. In two dimensions (2-D) this becomes $\Delta t / \Delta^2 \le 1/4$. Suppose we try to take the largest possible time step, and set $c\Delta t = \Delta^2 / 4$. Then eqn (7) becomes:

$$U_{i,j}^{n+1} = \frac{1}{4}\left(U_{i+1,j}^{n} + U_{i-1,j}^{n} + U_{i,j+1}^{n} + U_{i,j-1}^{n}\right) - b\Delta t U_{i,j}^{n} + b\Delta t U^{\oplus} \qquad (8)$$

thus the algorithm consists of using the average of U at its four nearest neighbor points on the grid (plus contribution from the source). If we are proceeding along the rows, incrementing j for fixed i, we have the formula eqn (8) as:

$$U_{i,j}^{n+1} = \frac{1}{4}\left(U_{i+1,j}^{n} + U_{i-1,j}^{n+1} + U_{i,j+1}^{n} + U_{i,j-1}^{n+1}\right) - b\Delta t U_{i,j}^{n} + b\Delta t U^{\oplus},$$

$$i = 1\ldots,n,\ j = 1\ldots,m \qquad (9)$$

This method is also slowly converging and only of theoretical interest, but some analysis of it will be instructive. If we have approximate values of the unknowns at each grid point, this equation can be used to generate new values. We call $U^{(n)}$ the current values of the unknowns at each iteration k and $U^{(n+1)}$ the value in the next iteration. We define a scalar $\omega_n (0 < \omega_n < 2)$ and apply eqn (9) to all interior points $(i,\ j)$. Hence, we have:

$$U_{i,j+1}^{n+1} = \varpi * GS + (1-\varpi)U_{i,j}^{n} \qquad (10)$$

where GS is the calculated value of the Gauss-Seidel method and ϖ is omega with values ranging from $0 < \varpi < 2$.

3 Parallel implementation

3.1 The cluster system

The implementation is done on a distributed computing environment (Armadillo Generation Cluster) consisting of 48 Intel Pentium at 1.73GHZ and 0.99GB RAM. Communication is through a fast Ethernet of 100 MB per second running Linux. The cluster performance has high memory bandwidth with a message passing supported by MPI [14]. We concentrate on basic message operations: blocking send, blocking receives, non-blocking send, and non-blocking receive. Note that MPI provides a rather comprehensive set of messaging operations. MPI primitive communication operation is the blocking send to blocking receive. At each time step we have to evaluate v^{n+1} values at $'\mathrm{lm}'$ grid points, where $'\mathrm{l}'$ is the number of grid points along the x axis. Suppose we are implementing this method on an $\mathrm{R}\times\mathrm{S}$ mesh connected computer. Denote the workers by $\mathrm{P}_{\mathrm{i1,j1}}$:$\mathrm{i1} = 1,2,\ldots,\mathrm{R}$ and $\mathrm{R} < \mathrm{l}$, $\mathrm{j1} = 1,2,\ldots,\mathrm{S}$ and $\mathrm{S} < \mathrm{M}$. Let $\mathrm{L}_1 = \left[\frac{1}{\mathrm{R}}\right]$ and $\mathrm{M}_1 = \left[\frac{\mathrm{M}}{\mathrm{S}}\right]$ where $[\ ]$ is the smallest integer part. Divide the $'\mathrm{lm}'$ grid points into $'RS'$ groups so that each group contains at most $(L_1+1)(M_1+1)$ grid points and at least L_1M_1 grid points. Denote these groups by $G_{i1j1}: i1 = 1,2,\ldots,R,\ j1 = 1,2,\ldots,S$. Design G_{i1j1}, such that it contains the following grid points

$$G_{i1j1} = \left\{ \begin{aligned} &(X_{(i1-1)+1}, Y_{(j1-1)+j}): i = 1,\ 2,\ \cdots,\ L_1 \text{ or } L_i + 1 \\ &\qquad\qquad\qquad\qquad j = 1,\ 2,\ \cdots,\ M_1 \text{ or } M_1 + 1 \end{aligned} \right\}$$

Assign the group G_{i1j1}, to the workers $P_{i1,j1}: i_1 = 1,2,\ldots,R$, for, $j_1 = 1,2,\ldots,S$. Each worker computes its assigned group $v_{i,j}^{n+1}$ values in the required number of sweeps. At the $(p+1/2)^{th}$ sweep the workers compute $v_{i,j}^{(p+1/2)th}$ values of its assigned groups. For the $(p+1/2)^{th}$ level the worker P_{i1j1} requires one value from the worker P_{i1-1j1} or P_{i1+1j1}, worker. In the $(p+1/2)^{th}$ level the communication between the workers is done row-wise. After communication between the workers is completed then each worker P_{ij} computes the $v_{i,j}^{p+1/2}$ values. For the $(p+1)^{th}$ sweep each worker P_{i1j1} requires one value from the P_{i1-1j1} or P_{i1+1j1} worker. Then each worker computes the values $v_{i,j}^{(p+1)th}$ of its assigned group. Statements need to be inserted to select which portions of the code will be executed by each processor. We focus our evaluation on MPI because it serves as an important foundation for a large group of applications.

3.1.1 The I_{aff}

With reference to [15], the I_{aff} is discussed. The platform introduces each task going through three phrases during execution of a parameter-sweep application: (1) an initialization phase. The duration of this phase is equal to t_{init}. This phase includes the overhead incurred by the master to initiate a data transfer to a slave, (2) a computational phase. The duration of this phase is equal to t_{comp}. Any additional overhead related to the reception of input files by a worker node is also included in this phase and (3) a completion phase, where the output file is sent back to the master and the master task is completed. The duration of this phase is equal to t_{end}. Therefore, the initialization phase of one slave can occur concurrently with the completion phase of another worker node. However, the total execution time of a task is equal to

$$t_{total} = t_{init} + t_{comp} + t_{end} \tag{11}$$

One processor is the master and the other processors are workers. P_{eff} is the effective number of processors needed to run an application with no idle periods on any worker processor. A processor may have idle periods if:

$$t'_{comp} < (P-1)t_{init} \tag{12}$$

P_{eff} is then given by the following equation:

$$P_{eff} = \left\lfloor \frac{t'_{comp}}{t_{init}} + 1 \right\rfloor \tag{13}$$

the total number of tasks to be executed on a processor is at most

$$M = \left\lceil \frac{T}{P} \right\rceil \tag{14}$$

for a platform with P_{eff} processors, the upper bound for the total execution time (makespan) will be

$$\lceil t_{makespan} \rceil = M(t_{init} + t'_{comp}) + (P-1)t_{init} \tag{15}$$

the second term in the right hand side of eqn (15) shows the time needed to start the first (P – 1) tasks in the other P – 1 processors. If we have a platform where the number of processors is larger than P_{eff} the overall makespan is dominated by communication times between the master and the workers. We then have

$$\lceil t_{makespan} \rceil = MPt_{init} + t'_{comp} \tag{16}$$

the set of eqn (12)–(14) will be considered in subsequent sections of this paper. It is worth noting that eqn (15) is valid when workers are constantly busy, either performing computation or communication. Eqn (16) is applicable when workers have idle periods, i.e., are not performing either computation or communication. eqn (12) occurs mainly in two cases:

For very large platforms (P large).

For applications with small $\frac{t_{comp}}{t_{init}}$ ratio, such as fine-grain applications.

In order to measure the degree of affinity of a set of tasks concerning their input files, the concept of input file affinity is introduced. Given a set of *E* of tasks, composed of R tasks, $E = \{T_1, T_2, \ldots, T_R\}$, and the set *Q* of the *X* input files needed by the tasks belonging to group *E*, $Q = \{f_1, f_2, \ldots, f_x\}$.

4 Results and discussion

4.1 Benchmark problem

We implement the SOR technique on the 2DBHTP. We assume a platform composed of variable number of heterogeneous processors. The solution domain was divided into rectangular blocks. The experiment is demonstrated on meshes of 100×100, 200×200 and 400×400, respectively. Tables 2–4 show the various performance timing.

$$\frac{\partial U}{\partial t} = c\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) - bU + bU^{\oplus}, \tag{17}$$

The boundary condition and initial condition posed are:

$$\left.\begin{array}{l} U(0,y,t) = 0 \\ U(1,y,t) = 0 \\ U(x,0,t) = 0 \\ U(x,1,t) = 0 \end{array}\right\} \quad t \geq 0 \tag{18}$$

The cell size was chosen as $\Delta x = \Delta y.$ the values of the physical properties in our test cases are chosen to be $\rho = 1000 kg/m^3$, $c = c_b = 4200 J/kg^0 c$, $w_b = 0.5 kg/m^3$, temperature is set to be $U_0 = 12^0 c$. Table 1 provides a comparison of the accuracy of the methods under consideration in terms of absolute error.

Table 1: Sequential result for 2DBHTP with various schemes.

Method	GS	SOR
Av. Abs. Err.	10.5×10^{-4}	8.7×10^{-4}
RMS	5.6×10^{-4}	4.3×10^{-7}
It	11	9
Δx	1×10^{-1}	1×10^{-1}
Δt	2×10^{-4}	2×10^{-4}
λ	5×10^{-1}	5×10^{-1}
t	1.8×10^{-3}	1.8×10^{-3}
eps	1×10^{-4}	1×10^{-4}

The experiment is demonstrated on meshes of 200×200 and 300×300 for block sizes of 100 and 200, respectively, both for MPI. The tables show the various performance timings.

4.1.1 Parallel efficiency

The speed-up and efficiency obtained for various sizes of 200×200 to 300×300 are for various numbers of sub-domains; from B = 100 to 200 as listed in Tables 1–4. In these tables we listed the wall (elapsed) time for the master task, T_W, the master CPU time, T_M, the average worker computational time, T_{SC} and the average worker data communication time T_{SD} all in seconds. The speed-up and efficiency versus the number of processors are shown in Fig. 1 and Fig. 2 respectively, with block number B as a parameter. The results show that the parallel efficiency increases with increasing grid size for a given block number

Table 2: A mesh of 300×300, with B = 50 blocks and $Niter$ = 100 for MPI.

Schemes	***N***	T_w	T_m	T_{sd}	T_{sc}	***MPI***		
						T	S_{par}	E_{par}
	1	*945*	*46*	*27*	*819*	*593*	*1.000*	*1.000*
	2	*652*	*45*	*25*	*572*	*364.3*	*1.628*	*0.814*
	4	*617*	*45*	*25*	*289*	*255.21*	*2.324*	*0.581*
	8	*524*	*45*	*25*	*229*	*208.47*	*2.845*	*0.356*
SOR	*16*	*382*	*45*	*25*	*152*	*183.0*	*3.241*	*0.203*
	20	*262*	*45*	*25*	*107*	*143.68*	*4.128*	*0.206*
	24	*195*	*45*	*25*	*89*	*122.92*	*4.825*	*0.201*
	30	*133*	*45*	*25*	*31*	*111.46*	*5.321*	*0.177*
	38	*92*	*45*	*25*	*20*	*101.12*	*5.924*	*0.156*
	48	*83*	*45*	*25*	*10*	*94.97*	*6.245*	*0.130*

Table 3: A mesh of 300×300, with B = 100 blocks and $Niter$ = 100 for MPI.

Schemes	***N***	T_w	T_m	T_{sd}	T_{sc}	***MPI***		
						T	S_{par}	E_{par}
	1	*1326*	*68*	*30*	*1109*	*773.7*	*1.000*	*1.000*
	2	*921*	*66*	*28*	*984*	*440*	*1.758*	*0.879*
	4	*875*	*66*	*28*	*752*	*285.14*	*2.713*	*0.678*
	8	*704*	*66*	*28*	*561*	*235.35*	*3.287*	*0.411*
SOR	*16*	*562*	*66*	*28*	*329*	*194.51*	*3.978*	*0.249*
	20	*431*	*66*	*28*	*284*	*161.28*	*4.797*	*0.240*
	24	*349*	*66*	*28*	*201*	*138.74*	*5.577*	*0.232*
	30	*298*	*66*	*28*	*162*	*114.49*	*6.758*	*0.225*
	38	*211*	*66*	*28*	*144*	*99.25*	*7.795*	*0.205*
	48	*184*	*66*	*28*	*94*	*95.12*	*8.134*	*0.169*

Table 4: A mesh of 300×300, with B = 200 blocks and $Niter$ = 100 for MPI.

Schemes	***N***	T_w	T_m	T_{sd}	T_{sc}	***MPI***		
						T	S_{par}	E_{par}
	1	*2431*	*97*	*62*	*1499*	*1144*	*1.000*	*1.000*
	2	*1764*	*95*	*60*	*1103*	*644.03*	*1.776*	*0.888*
	4	*987*	*95*	*60*	*865*	*387.76*	*2.950*	*0.738*
	8	*834*	*95*	*60*	*679*	*347.57*	*3.291*	*0.411*
SOR	*16*	*698*	*95*	*60*	*482*	*234.23*	*4.884*	*0.305*
	20	*573*	*95*	*60*	*321*	*214.14*	*5.342*	*0.267*
	24	*432*	*95*	*60*	*294*	*201.76*	*5.670*	*0.236*
	30	*359*	*95*	*60*	*210*	*165.13*	*6.928*	*0.231*
	38	*286*	*95*	*60*	*194*	*144.46*	*7.919*	*0.208*
	48	*231*	*95*	*60*	*121*	*134.70*	*8.493*	*0.177*

using MPI and decreases with the increasing block number for given grid size. In Tables 1–4, the master time T_M is constant when the number of processors increases for a given grid size and number of sub-domains. The master program is responsible for (1) sending updated variables to worker (T_1), (2) assigning task to worker (T_2), (3) waiting for the worker to execute tasks (T_3), and (4) receiving the results (T_4).

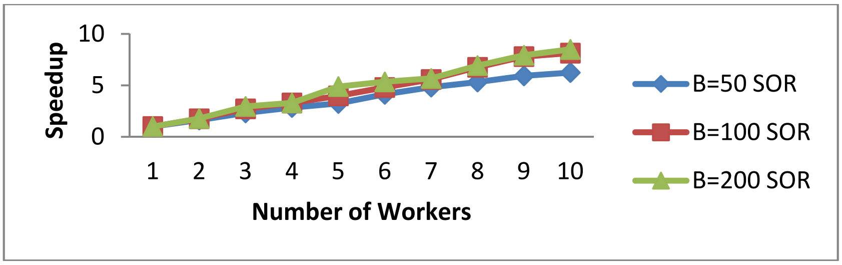

Figure 1: Speed-up versus the number of workers for various block sizes. Mesh 300×300 MPI.

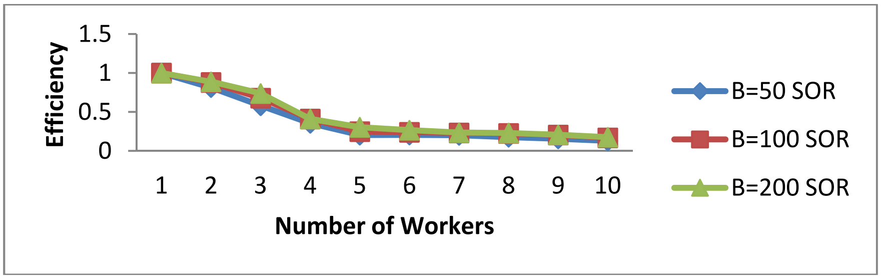

Figure 2: Parallel efficiency versus the number of workers for various block sizes. Mesh 300×300 MPI.

As the total number of processors increase, the bottleneck of parallel computers appears and the global reduction consumes a large part of time.

5 Conclusions

A thorough analysis of the 2DBHTP using SOR iterative technique and I_{aff} on master-worker paradigm was made, which shows that significantly better efficiency can be obtained. The efficiency is accomplished by overlapping computation of the sequential and parallel phases of the algorithm. The I_{aff} on master-worker paradigm has been emphasized. Experiments with 2DBHTP revealed that the rate of convergence decreases as the number of processor increases for various mesh sizes. Our goal is shown on the interleaving of communication/computation that is at the core of the MPI performance, and the proposed methods combine elements of numerical stability and parallel algorithm design without increasing computational costs. Here, MPI is used for its performance and portability. On the basis of the current parallelization strategy, more sophisticated models can be attacked efficiently.

References

[1] Hinde, L. B. Perez, C. and Priol, T., Extending Software Component Models with the Master-Worker Paradigm, *Parallel Computing* 36, pp. 86–103, 2010.

[2] Foster, I. Geist, J. Groop, W. and Lust, E., Wide-Area Implementations of the MPI, *Parallel Computing* 24 pp. 1735–1749, 1998.

[3] Callahan D, and Kennedy K., Compiling Programs for Distributed Memory Multiprocessors, *Journal of Supercomputer* 2, pp. 151–169, 1988.

[4] Evans D. J., and Hassan B.: Numerical Solution of the Telegraph Equation by the AGE Method Int'l Journal of Computer Mathematics (2003) **80** (10) 1289–1297.

[5] Chypher R., Ho A., *et al*., Architectural Requirements of Parallel Scientific Applications with Explicit Communications *Computer Architecture*, 12 pp. 2–13, 1993.

[6] Peizong L., and Kedem Z., Automatic Data and Computation Decomposition on Distributed Memory Parallel Computers *ACM Transactions on Programming Languages and Systems*, 24(1), pp. 1–50, 2002.

[7] Deng, Z. S. and Liu, J., Analytic Study on Bio-Heat Transfer Problems with Spatial Heating on Skin Surface or Inside Biological Bodies, *ASME Journal of Biomechanics Eng*., 124 pp. 638–649, 2002.

[8] Chinmay, M., Bio-Heat Transfer Modeling, *Infrared Imagine*, pp. 15–31, 2005.

[9] Liu, J. and Xu, L. X., Estimation of Blood Perfusion Using Phase Shift Temperature Response to Sinusoidal Heating at Skin Surfaces, *IEEE Trans. Biomed. Eng*., 46, pp. 1037–1043, 2001.

[10] Rubinsky, B. *et al*., Analysis of a Steufen-Like Problem in a Biological Tissue around a Cryosurgical Problem. *ASME J. Biomech. Eng*., 98, pp. 514–519, 1976.

[11] Liu J., Chen X. and Xu L., New Thermal Wave Aspects on Burn Evaluation of Skin Subjected to Instantaneous Heating, *IEEE Trans. Biomed. Engrg.*, 46, pp. 420–428, 1999.

[12] Dai W. Z. and Zhang J., *A Three Level Finite Difference Scheme for Solving the Pennes's Bio-Heat Transfer in a triple layered Skin Structure*, Technical Report No. 343, Department of Computer Science, University Science, University of Kentucky Lexingtn, KY 835, 2002.

[13] Jennifer, J. Z. *et al*., A Two Level Finite Difference Scheme for 1-D Penne's Bio-Heat Equation, 2002.

[14] Groop, W., Lusk, E. and Skjellum, A., Using MPI, Portable and Parallel Programming with the Message Passing Interface, 2nd Ed., Cambridge MA, MIT Press, 1999.

[15] Fabricio, D. S. Senger, H., Bag of Task running on Master-Slave with Input File, *Parallel Computing* 35, pp. 57–71, 2009.