Parallel Simulation of Three–Dimensional Bursting with MPI and OpenMP*

S. Tabik[1], L.F. Romero[2], E.M. Garzón[1] and J.I. Ramos[2]

[1] Depto de Arquitectura de Computadores y Electrónica, Universidad de Almería, 04120 Spain

[2] Room I-320, E.T.S. Ingenieros Industriales, Universidad de Málaga, 29080 Spain

Abstract. This work presents a mathematical model and its parallel implementation via two parallel paradigms for the simulation of three–dimensional bursting phenomena. The mathematical model consists of four nonlinearly coupled partial differential equations and includes fast and slow subsystems. The differential equations have been discretized by means of a linearly–implicit finite difference method in equally–spaced grids. The resulting system of equations at each time level has been solved by means of an optimized Preconditioned Conjugate Gradient (PCG) method. The proposed mathematical model has been implemented via: (1) a message passing paradigm based on the standard MPI and (2) a shared address space paradigm based on SPMD OpenMP. The two implementations have been evaluated on two current parallel architectures, i.e., a cluster of biprocessors Xeon and an SGI Altix 3700 Bx2 based on Itanium. It is shown that better performance and scalability are obtained on the second platform.

1 Introduction

Bursting phenomena occur in physiology and biology. For example, neurons communicate by firing and transmitting action potentials. Usually, action potentials occur in a periodic fashion, as in response to a constant applied current of sufficiently large magnitude. In addition, many cell types, e.g., pancreatic β-cells, exhibit more complex behavior characterized by brief bursts of oscillatory activity interspersed with quiescent periods during which the cell membrane potential changes slowly. Models of bursting electrical activity can be classified into two main groups. The first and earliest one was based on the assumption that bursting was caused by an underlying slow oscillation in the intracellular Ca^{2+} concentration [1, 2]; however, recent experiments indicate that this assumption is not entirely correct and, as a consequence, models relying on alternative mechanisms have been developed [3]. The different known bursting mechanisms can be classified into three main groups. In type I, bursts arise from hysteresis and bistability as in the pancreatic β–cell model. In type II, bursts arise from an underlying slow oscillation, while, in type III, bursting arises from a subcritical

* This work was supported by the Ministerio de Educación y Ciencia of Spain under Projects TIN2005-00447, FIS2005–03191 and Fondos FEDER.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 106–113, 2006.

Hopf bifurcation [1, 2, 4, 5]. This classification is by no means complete, for it is based on knowledge acquired with nonlinear ordinary differential equations. In extended systems, i.e., spatio–temporal systems, one can easily show that bursting occurs through one of the three types mentioned above provided that the system is homogeneous. However, when the system is not initially homogeneous and the diffusion coefficients are not nil, the homogeneous bursting solution may bifurcate and result in spatio–temporal nonhomogeneities and/or quenching.

The objective of this paper is several–fold. First, a three–dimensional model of bursting consisting of four, nonlinearly coupled partial differential equations is proposed. This model has been selected so that it exhibits bursting under homogeneous conditions, and both bursting and quenching for non–homogeneous initial conditions. The source or reaction terms of these partial differential equations can be classified into two subsystems exhibiting fast and slow behavior. The accurate simulation of the fast processes demands the use of small time steps, while the extended system considered here requires the use of sufficiently small spatial step sizes in order to accurately resolve the steep gradients of the dependent variables.

Second, a parallel implementation of the finite difference equations resulting from the discretization of the three–dimensional bursting model equations is developed and analyzed using two parallel paradigms: (1) a message passing paradigm where all communications between processors are established via MPI and (2) a shared address space paradigm based on SPMD OpenMP. An efficient parallel conjugate gradient (CG) solver has been used to solve the system of linear equations. Both parallel implementations are then evaluated on two parallel platforms: (1) a cluster of biprocessors Intel(R) Xeon(TM) and (2) a SGI Altix 3700 Bx2 system based on Itanium.

2 Formulation and Numerical Method

Bursting in extended systems has been simulated by means of the following nonlinearly coupled system of partial differential equations

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{D}\left(\frac{\partial^2 \mathbf{U}}{\partial x^2} + \frac{\partial^2 \mathbf{U}}{\partial y^2} + \frac{\partial^2 \mathbf{U}}{\partial z^2}\right) + \mathbf{S}(\mathbf{U}) \tag{1}$$

where $\mathbf{U} = (u, v, w, p)^T$, $\mathbf{D}$ is a diagonal matrix with components equal to D_u, D_v, D_w and D_p, t is time, x, y and z are Cartesian coordinates, $\mathbf{S} = (S_u, S_v, S_w, S_p)^T$ is the vector of the source/reaction terms given by

$$\begin{cases} S_u = f(u) - v - gw(u - u_0) \\ S_v = \frac{1}{5}(v_\infty(u) - v) \\ S_w = f_w(w) + \alpha_w(p - 0.3) \\ S_p = \beta_p((1 - p)H(u) - p) \end{cases} \tag{2}$$

and

$$\begin{cases} f(u) = 1.35u(1 - \frac{u^2}{3}) \\ f_w(w) = -0.2(w - 0.2)(w - 0.135)(w - 0.21) \\ v_\infty(u) = tanh(5u) \\ H(u) = \frac{3}{2}(1 + tanh(5u - 2.5)) \end{cases} \tag{3}$$

with $u_0 = 2$, $\alpha_\omega = 0.002$, $\beta_z = 0.00025$ and $g = 0.73$. The functions $f(u)$ and $f_\omega(\omega)$ have three zeros each, whereas $v_\infty(u)$ and $H(u)$ are monotonically increasing functions of their arguments.

Eq. 1 is a simplified model of bursting electrical activity in cells, where u and w are the fast system and represent the currents of activated and voltage–dependent channels, respectively, whereas w and p constitute the slow subsystem and represent a current and its activation, respectively.

Eq. 1 have been solved in a parallelepiped $\Omega \equiv [-L_x, L_x] \times [-L_y, L_y] \times [-L_z, L_z]$ and homogeneous Neumann boundary conditions have been used on all the boundaries. The initial conditions used in the computations are

$$u = -2.5, v = -0.2, \omega = -0.5, z = 0.5 \text{ in } \omega$$
$$u = 2.5, v = -0.2, \omega = 0.5, z = -0.5 \text{ in } \Omega - \omega$$

where $\omega = [-\frac{L_x}{2}, \frac{L_x}{2}] \times [-\frac{L_y}{2}, \frac{L_y}{2}] \times [-\frac{L_z}{2}, \frac{L_z}{2}]$. These initial conditions were selected so that, under homogeneous conditions, Eq. 1 exhibits bursting oscillations. By discretizing the time variable in Eq. 1 by means of the (second–order accurate) Crank-Nicolson technique, one can obtain a nonlinear system of elliptic equations at each time step. These elliptic equations can be linearized with respect to time in order to obtain a system of linear elliptic equations at each time step, i.e., the nonlinear terms $\mathbf{S}^{n+1}$ are approximated by means of the second-order accurate terms $\mathbf{S^n} + \mathbf{J}^n \Delta\mathbf{U}$ where $\mathbf{J}$ denotes the Jacobian of the source terms, the superscript n denotes the n–th time level, i.e., $t^n = n\Delta t$, $n = 0, 1, 2, 3, \ldots$, Δt is the time step, and $\Delta\mathbf{U} \equiv \mathbf{U}^{n+1} - \mathbf{U}^n$. The resulting system of linear elliptic equations was discretized in space by means of second–order accurate central finite difference formulae in a regular grid of $N_x \times N_y \times N_z$ points, and the resulting system of linear algebraic equations can be written as

$$\begin{array}{l} (\mathbf{I} - \frac{k}{2}\mathbf{J}^n_{i,j,k})\Delta\mathbf{U}_{i,j,k} \\ -\alpha_x(\Delta\mathbf{U}_{i+1,j,k} - 2\Delta\mathbf{U}_{i,j,k} + \Delta\mathbf{U}_{i-1,j,k}) \\ -\alpha_y(\Delta\mathbf{U}_{i,j+1,k} - 2\Delta\mathbf{U}_{i,j,k} + \Delta\mathbf{U}_{i,j-1,k}) \\ -\alpha_z(\Delta\mathbf{U}_{i,j,k+1} - 2\Delta\mathbf{U}_{i,j,k} + \Delta\mathbf{U}_{i,j,k-1}) = \mathbf{T}^n_{i,j,k} \end{array} \tag{4}$$

where $\mathbf{I}$ is the identity or unit matrix, $\alpha_x = \frac{k}{2\Delta x^2}\mathbf{D}$, $\alpha_y = \frac{k}{2\Delta y^2}\mathbf{D}$, $\alpha_z = \frac{k}{2\Delta z^2}\mathbf{D}$, Δx, Δy and Δz are the grid spacings in the x, y and z directions, respectively, and

$$\begin{array}{l} \mathbf{T}^n_{i,j,k} = 2\alpha_x(\mathbf{U}_{i+1,j,k} - 2\mathbf{U}_{i,j,k} + \mathbf{U}_{i-1,j,k}) \\ +2\alpha_y(\mathbf{U}_{i,j+1,k} - 2\mathbf{U}_{i,j,k} + \mathbf{U}_{i,j-1,k}) \\ +2\alpha_z(\mathbf{U}_{i,j,k+1} - 2\mathbf{U}_{i,j,k} + \mathbf{U}_{i,j,k-1}) + \mathbf{S}^n_{i,j,k} \end{array} \tag{5}$$

Since the finite difference discretization of a system of $N_e(= 4)$ partial differential equations results in $N_e \times N_x \times N_y \times N_z$ nodal variables and algebraic equations,

a main issue when solving systems of linear algebraic equations such as Eqs. 4 is the ordering of the equations and variables. A natural ordering of the grid points and blocking of nodal variants has been chosen here to obtain a well structured matrix, and because of its better cache behavior. For this ordering, Eq. 1 can be expressed in matrix form as $Ax = b$, where A is an heptadiagonal block matrix, with 4×4 blocks, because $N_e = 4$ in Eq 1, and $x = \Delta\mathbf{U}$. We have used the conjugate gradient method with Jacobi preconditioner to solve the linear system of equations $Ax = b$. The sequential algorithm can be described as follows:

Initialize the source/reaction terms according to Eqs. 2 and 3.
Initialize the solution U^0 according to the initial conditions considered.
do n=1,2,3,...
- Update the matrix A and right–hand–side b according to Eqs. 4 and 5, respectively.
- Solve the system of equations $A \cdot \Delta U = b$.
- Update the solution $U^n = U^{n-1} + \Delta U$.

end do

Preliminary profiling of the serial simulation has shown that solving the system of equations is the most computationally consuming step. Therefore, the efficiency of the whole code depends mainly on the efficiency of the solution of the system of linear equations. In addition, the storage format of the matrix has a large influence on the performance of the solver used [12]. For this reason, an optimized preconditioned Conjugate Gradient (PCG) method for banded matrices, designed by Ortigosa et al. [7], has been used. The optimized PCG is based on a maximum exploitation of the locality of data and message-computation overlap. Moreover, The matrix has been stored using a compressed diagonal format which takes into account the sparsity pattern of the Jacobian term, and the implementation of the sparse matrix-vector product includes a consequent hand-unrolled product which exploits the pipelined floating point units.

In this paper, all computations have been performed on (coarse) $51 \times 51 \times 51$ and (fine) $101 \times 101 \times 101$–point equally-spaced meshes. The coarse mesh represents the largest step size for which accurate results can be obtained, and the time step is equal to 0.0001 s .

Some sample results corresponding to $L_x = L_y = L_z = 15$ are presented in Fig. 1 which shows the time history of u at two monitor points: one (identified with asterisks) in ω and the other (identified with squares) in $\Omega - \omega$. This figure shows that, for the initial conditions considered in this paper, the time history of u in $\Omega - \omega$ exhibits an oscillatory behavior characterized by periods of intensive variations in u starting from $t = 0$, whereas the time history of u in ω takes a while before it starts oscillating. The oscillations observed in Fig. 1 are typical of relaxation oscillations.

Fig. 1 also shows that the oscillations of u in ω and $\Omega - \omega$ are not in phase, but nearly coincide at about $t = 1370$ a.u.; thereafter, the one that was lagging overcomes the other, and, after some time, both time histories reach the same slowly varying value. Similar trends to those shown in Fig. 1 have also been observed at other six monitoring locations.

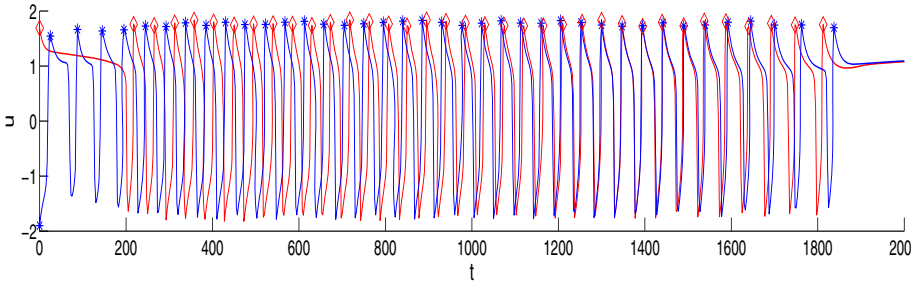

Fig. 1. u versus time t (in a.u.), in ω ($*$) and $\Omega - \omega$ ($\diamond$)

Although not shown here, three–dimensional, time–dependent visualizations of u indicate that there are inward and outward propagating waves in ω before the outward ones propagate in $\Omega - \omega$, i.e., there is an interplay between the values of u in ω and $\Omega - \omega$ caused by the diffusion of the dependent variables. This back–and–forth play between the inner and outer domains is a consequence of the initial conditions, the source/reaction terms, and the diffusion coefficients of Eqs. 1.

3 Parallel Implementation

CLUsters of MultiProcessors (CLUMPs) have become today's dominant platforms. Such architectures support three parallel programming paradigms, i.e., message passing paradigm, shared address space paradigm and hybrid paradigm. Here, we propose two parallel implementations of the three–dimensional bursting model described above; the first is based on the message passing paradigm using MPI, while the second is based on a shared address space using OpenMP. Both implementations are based on SPMD style, whereby each processor works on its own sub-domain.

3.1 MPI Code

At each time-step, each processor updates its sub-matrix and local right–hand–side. Then, the parallel solution of the system $A \cdot \Delta U = b$ is performed by the PCG solver as in [7]. Using a Jacobi preconditioner, each iteration of the PCG method includes two inner products and one sparse matrix vector product; therefore, three communications and their corresponding synchronizations are needed. In this implementation, computations and communications have been overlapped using asynchronous messages in order to minimize the communications overhead, specially in the matrix vector product. In this way, after the iterations of the PCG are completed, a message with the boundary values of U is sent to the neighboring processors and this is overlapped with the last update of ΔU.

3.2 OpenMP Code

We have chosen the SPMD style instead of the loop level one because it reduces overhead and results in better scalability [10]. In this implementation, only one

parallel section covers the whole dynamic extent of the code and, therefore, only OpenMP synchronization directives have been used. The ordering of the computations employed in the MPI implementation to minimize the communication overhead is also employed here to reduce the waits in the synchronizations points. For the inner product, the synchronization events have been implemented by using counters protected by lock variables. Additional flags have been included in order to grant permission for accessing shared data. As soon as a processor has computed the data on its borders, it enables the flag. If other processor requires these data, it waits for this flag to be enabled. Reset of counters and flags has been carefully implemented by means of odd and even sense-reversing flags to enhance performance and avoid data race conditions. In both massage-passing and shared space address paradigms, global barriers have been avoided to minimize waiting times.

4 Evaluation of the Parallel Implementation

We have assessed the performance of both parallel codes for the coarse and fine grids on the architectures described below:

- A cluster of Intel(R) Xeon(TM) biprocessors at 3.06 GHz with 2 GB RAM and 512 KB cache. The nodes of the cluster are interconnected via two gigabit Ethernet networks: one for data (NFS) and the other for computation.
- An SGI Altix 3700 Bx2 of 64 processors at 1600 MHz Intel Itanium 2 Rev with 128 GB RAM and 6 MB L3. The Altix 3700 computer system is based on a Distributed Shared Memory architecture [8] and uses a cache–coherent Non-Uniform Memory Access (NUMA) where the latency of processors to access the local memory is lower than the latency to access the global (or remote) memory [9].

Preliminary evaluation has shown that for both codes, the computational cost is similar for each time step; therefore, in this section, we show and analyze the average execution time for only one time step.

On the cluster of biprocessors, the parallel performance and the scalability of the code based on MPI paradigm are better when the dimension of the matrix is large as shown in Table 1. One of the underlying reasons is the increase in the computation–communication ratio, i.e., a finer grid or a large matrix involves more computation, but also requires more communications. The evaluation of the code based on OpenMP cannot be completed in more than two processors; therefore, no significative conclusions can be made for this case.

On a SGI Altix 3700 Bx2 system, both paradigms yield very good performance for both coarse and fine grids as shown in Table 2. This is due to the high speed interconnection technology used in this architecture [9] and to its good shared memory features. Besides, both codes includes pipelining techniques, which allow to reach very good performance on VLIW architectures such those based on Itanium [12]. Moreover, the scalability for both codes is slightly better for the coarse grid; this can be explained by two reasons, first, the good behavior of the

Table 1. Performance and speedup of the parallel simulation per time step on a cluster of biprocessors, for the coarse and fine grids

	coarse grid				fine grid			
	openMP		MPI		openMP		MPI	
proc	runtime (s)	speedup	runtime (s)	speedup	runtime (s)	speedup	runtime (s)	speedup
1	0.61	1.00	0.61	1.00	6.20	1.00	6.20	1.00
2	0.50	1.20	0.58	1.05	5.05	1.23	3.02	1.99
4			0.32	1.88			1.64	3.66
6			0.24	2.52			1.26	4.76
8			0.20	3.03			1.03	5.83

Table 2. Performance and speedup of the parallel simulation per time step on a SGI Altix 3700 Bx2, for the coarse and fine grids

	coarse grid				fine grid			
	openMP		MPI		openMP		MPI	
proc	runtime (s)	speedup	runtime (s)	speedup	runtime (s)	speedup	runtime (s)	speedup
1	0.258	1.00	0.258	1.00	2.001	1.00	2.001	1.00
2	0.135	1.91	0.135	1.91	1.010	1.98	1.025	1.97
4	0.070	3.68	0.072	3.59	0.591	3.39	0.855	2.36
6	0.047	5.50	0.044	5.91	0.407	4.91	0.459	4.43
8	0.035	7.44	0.035	7.42	0.2789	6.19	0.336	6.01

memory management, since the percentage of data that fits in the cache memory in this case is higher than that for the fine grid; second, both implementations use a reorganization of the computations in order to minimize the communications overhead (in MPI-code) and the synchronizations overhead in (OpenMp-code), the penalty of this reordering on the locality accesses is higher for the fine grid.

Finally, it should be mentioned that the performance of both sequential codes on the platform based on Itanium is 2 to 3 times better than that on the platform based on Xeon, and this is in accord with the results of the Standard Performance Evaluation Corporation (SPEC) [11].

5 Conclusions

A mathematical model for three–dimensional bursting phenomena and two parallel implementations of it have been presented. The model is described by four, nonlinearly coupled partial differential equations which have been discretized by means of a second–order accurate, linearly–implicit finite difference method in equally–spaced grids. The resulting system of linear algebraic equations at each time level has been solved by means of the PCG solver optimized for banded matrices and implemented using two parallel paradigms: (1) a message

passing paradigm with a message-computation overlap and (2) a SPMD OpenMP style. Both implementations have been evaluated on two parallel platforms: a DSM platform based on Itanium and a cluster of biprocessors Xeon. The performance of both implementations depends on the mesh size, the parallel paradigm used and the architecture of the platform. It has been shown that both parallel paradigms are suitable for the platform based on Itanium, especially for coarse grids, whereas for clusters based on Xeon processors, only with fine grids and using the message passing paradigm one can achieve the expected performance.

References

1. Rinzel, J. and Lee, Y. S., Dissection of a model for neuronal parabolic bursting, *J. Math. Biol.* **25** (1987) 653–675.
2. Rinzel, J., Electrical excitability of cells, theory and experiment: review of the Hodgkin–Huxley foundation and an update, *Bull. Math. Biol.* **52** (1990) 5–23.
3. Smolen, P. and Keizer, J., Slow voltage inactivation of Ca^+2 currents and bursting mechanisms for the mouse pancreatic β–cell, *J. Membr. Biol.* **127** (1992) 9–19.
4. Bertram, R., Buttle, M. J., Kiemel, T. and Sherman, A., Topological and phenomenological classification of bursting oscillations, *Bull. Math. Biol.*, **57** (1995) 413–439.
5. Keener, J. and Sneyd, J., *Mathematical Physiology*, Springer, New York, 1998.
6. Ramos, J. I., Linearization methods for reaction–diffusion equations: Multidimensional problems. *Appl. Math. Comput.* **88** (1997) 225-54.
7. Ortigosa, E. M., Romero, L. F. and Ramos, J. I., Parallel scheduling of the PCG method for banded matrices rising from FDM/FEM, *J. Parall. Distr. Comput.* **63** (2003) 1243 - 1256.
8. Protic, J., Tomasevi, M. and Milutinovic, V., *Distributed Shared Memory: Concepts and Systems*, Wiley, New York, 1997.
9. Dunigan, T., Vetter, J. and Worley, P., Performance evaluation of the SGI Altix 3700, In *Proc. of the IEEE Int. Conf. Parallel Proc., ICPP*, 231–240, 2005.
10. Krawezik, G. and Cappello, F., Performance comparison of MPI and OpenMP on shared memory multiprocessors, *Concurr. Comput.: Practice and Experience*, **18** (2006) 29–61.
11. http://www.spec.org/
12. Mellor–Crummey, J. and Garvin, J., Optimizing sparse matrix–vector product computations using unroll and jam. Int. J. High Perfor. Comput. Applic., 18 (2004) 225–236.