Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

NEURON TIME WARP

Mohammand Nazrul Ishlam Patoary
Carl Tropper

School of Computer Science
McGill University
Montreal, Quebec, H3A2A6, CANADA

Zhongwei Lin

College of Computer
National University of Defense Technology
17 Yanwachizheng, Changsha, Hunan, 410073, CHINA

Robert McDougal

School of Medicine
Yale University
New Haven, Connecticut, USA

William W. Lytton, MD

SUNY Downstate Medical Center
450 Clarkson Avenue, MSC 31
Brooklyn, New York 11203, USA

ABSTRACT

Detailed simulation of chemical reactions and the diffusion of ions through a neuronal membrane presents challenges due to the multiple scales at which this occurs, scales that require development and consolidation of a number of different simulation methodologies. In this paper, we describe Neuron Time Warp (NTW), a part of the NEURON project for development of multi-scale tools for simulations of brain parts and brains. NTW relies upon the Next Subvolume Method, a stochastic Monte Carlo algorithm used to simulate chemical reactions within the membrane of a neuron. We make use of a model of a dendrite branch on which to evaluate NTW's performance using MPI and shared memory on a multi-core machine. This work is a first step towards the development of multi-scale simulation models which are capable of portraying the behavior of a neuron with greater fidelity then is possible with differential equation based models alone.

1 INTRODUCTION

The human brain may be viewed as a sparsely connected network of neurons (Carnevale and Hines 2006) containing approximately 10^{14} neurons. A typical neuron is displayed in figure 1. Each neuron receives inputs from thousands of synapses. It also sends outputs to thousands of neurons by means of its axon.

The membrane of a neuron is semi-permeable by virtue of channels which selectively control the flow of ions (primarily sodium, potassium, and calcium) between extracellular and intracellular fluid. Movements of ions through these channels results from diffusion or pumps. The Nernst potential created by different concentrations of these ions on the exterior and interior of the cell provides a battery. Electrical models for neurons (Lytton 2002) can then be constructed using the well-known laws of electricity (Ohm, Kirchkof, capacitance). However, these electrical models only provide a limited view of neuronal activity since there is a complex of chemical species that contribute substantially to the information processing of the neuron, functioning as information 'messengers'. In order to develop realistic models for the electrical response in a neuron, it is necessary to develop more detailed models for the diffusion of these messengers inside the cell. One of the most important messengers is the calcium ion. Calcium ions play a significant role in intracellular signaling.

978-1-4799-7486-3/14/$31.00 ©2014 IEEE

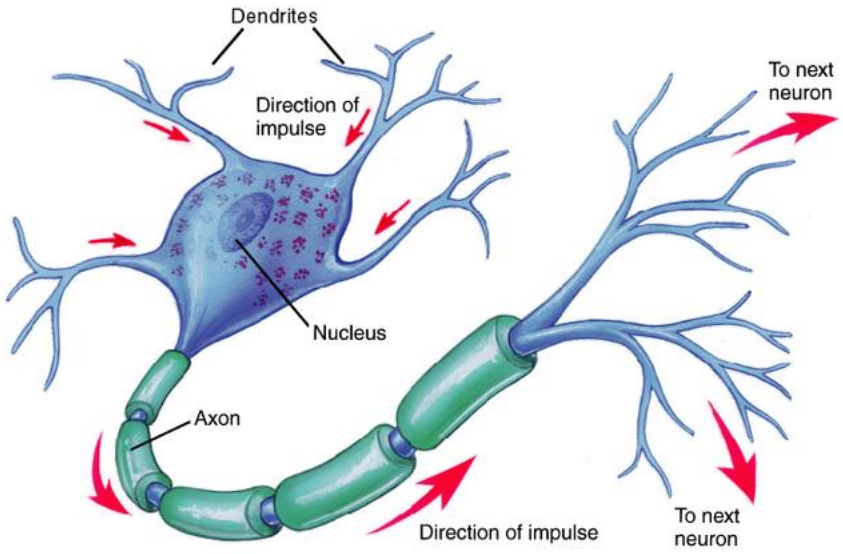

Figure 1: Sketch of a Neuron.

The combination of chemical reactions within a cell with the diffusion of ions can be modeled as a reaction diffusion system and simulated by (parabolic) partial differential equations (Carnevale and Hines 2006). However, a continuous model is not appropriate for a small number of molecules. A stochastic model is a far more realistic and accurate representation (Sterratt, Graham, Gillies, and Willshaw 2011).

In order to express this stochastic behavior, the system consisting of a collection of chemical reactions can be represented by a so-called chemical master equation, which is a probability distribution of the chemical reactants in the system (Sterratt, Graham, Gillies, and Willshaw 2011). In general, it is very difficult to solve this equation. In (Gillespie 1977) Gillespie introduced a Monte Carlo simulation algorithm for this model. Under the assumption that the molecules of the system are uniformly distributed, the algorithm simulates a single trajectory of the chemical system. Simulating a number of these trajectories then gives a picture of the system. The algorithm uses an exponential distribution to compute the time of the next event (i.e.reaction) and employs elementary combinatorics to compute the likelihood of a particular reaction occurring.

Authors (Gibson and Bruck 2000) describe the Next Reaction Method, which attempts to reduce the computation time of the propensities in Gillespie's algorithm. It makes use of a dependency graph among the reactions which is used to identify the reactions which are in need of an update. Gillespie's algorithm then updates the states of all of the reactants. Authors (Cao, Li, and Petzold 2004) modify the Next Reaction Method by re-sorting the event queue in order of the probability of execution of a particular reaction. A number of other efforts aimed at improving the efficiency of the Gillespie algorithm have been made, including (Gillespie 2001), (Takahashi, Kaizu, Hu, and Tomita 2004).

A key assumption in the Gillespie algorithm is that the particles are distributed homogeneously in space. This, however, is not the case in a neuron because ions enter through membranes from the outside, as well as from 'organelles' situated inside of the neuron. Both of these processes lead to inhomogeneities that then lead to internal diffusion of calcium or other chemical species. This diffusive behavior must be included in a realistic neural simulation.

We make use of a Monte Carlo algorithm, the Next Subvolume Method (NSM) (Elf and Ehrenberg 2004). The NSM partitions space into cubes and represents the diffusion of ions between these cubes by events. Other events are used to characterize reactions within the cubes. NSM makes use of the Gillespie

algorithm to compute next event times within the cubes and relies upon the use of a priority queue to determine the next event and diffusion times. Our parallel simulation makes use of the NSM algorithm.

As previously indicated, the number of cells involved in a realistic simulation of a network of neurons is immense. This means that it is necessary to make use of a cluster of computers for such a simulation. Since each cube in the NSM can interact with other cubes, it can be represented by a Logical Process (LP) in PDES, and PDES techniques can be introduced in order to speed up the simulation.

Author (Dematt and Mazza 2008) points out that a conservative synchronization will perform poorly because of the zero-lookahead property of the exponential distribution and the fact that a dependency graph of the reactions is likely to be highly connected and filled with loops. This indicates that an optimistic synchronization such as Time Warp is preferable. Authors (Avril and Tropper 2001) make use of clusters of LPs which are simulated sequentially on individual processors while Time Warp is used to simulate the collection of clusters. XTW (Xu and Tropper 2006) is an outgrowth of CTW and makes use of a multi-level priority queue to schedule events. The cost of event scheduling in XTW is O(1). XTW also makes use of a single rollback message to annihilate all of the messages sent prior to the arrival of a straggler instead of sending individual anti-messages. Our simulator, Neuron Time Warp (NTW), is based upon XTW.

NEURON is a framework for simulating neurons which is used by neuroscientists world-wide. At its heart is a deterministic model for electrical conduction in the neuron. Our work is part of NEURON project and has the long term goal of developing a multi-scale simulation for large scale neuronal networks.

The remainder of this paper is as follows. Section 2 is devoted to previous work. Section 3 contains a description of NTW. Section 4 contains a description of our experiments as well as the results and section 5 contains our conclusions and a road map to our future work.

2 PREVIOUS WORK

A limited amount of work has been done to date applying PDES techniques to stochastic models of neurons. Authors (Jeschke, Ewald, Park, Fujimoto, and Uhrmacher 2008) describe two Time Warp based approaches for parallelizing NSM. Both represent sub-volumes by LPs. Messages between sub-volumes contain the diffusion events. One of the simulators makes use of grid computing while the other makes use of a plug 'n' play simulation environment. Encouraging preliminary results were obtained for the Lotke-Volterra predator-prey model. As pointed out by the authors, a number of areas including window management and state saving remain to be investigated. (Wang, Yao, Zhao, Hou, and Peng 2009) applies a Breathing Time Warp algorithm to a predator prey model, making use of cubes as in (Jeschke, Ewald, Park, Fujimoto, and Uhrmacher 2008). The Gillespie algorithm is not executed within the LPs, casting some doubt upon the algorithm.

3 NTW

The front end for NTW makes it possible for a group of chemical reactions to be input by the user and to be used in the simulation. This enables the user to experiment with different reactions, different concentrations of the molecules and ions involved in the reactions as well as their associated reaction rates. It is also possible to experiment with different diffusion rates into between adjacent cells.

NTW inherited the multi-leveled priority queue from XTW in order to minimize the cost for scheduling events. A history queue, used to record scheduling history, has been added. Every sub-volume (SV) in NSM is an LP in NTW. The reactions and the diffusion between sub-volumes are events in NTW.

3.1 Architecture

The LPs are grouped together into clusters. The clusters can be distributed among separate physical computational units. We currently allocate one cluster per core. Each cluster processes the events belonging to its LPs in increasing time-stamped order. There is a special cluster, called a controller, which is in charge of distributing the simulation workload, computing the GVT, and collecting the simulation results. We

employ Mattern's algorithm to calculate the GVT. Checkpointing is done upon the arrival of a diffusion event at a cluster. Figure 2 describes the main components of a cluster. There are two cluster level event queues, the clEQ and the inputExtEQ. The clEQ is used to sort the events generated by local LPs (i.e. within the same cluster) and its top event is the lowest time stamped event of the cluster. All events from different cluster are temporarily put into inputExtEQ and then forwarded to destination input channel event queue, ICEQ, in time stamp order. The priority SV queue is a container of SVs. The head of the SV queue contains the SV of minimal time of next event of the cluster. In a three-dimensional geometry, an SV can have at most six adjacent SVs, called neighbors. The components of an SV are associated with this observation. Each SV has a SVEQ which is used to find the smallest time-stamped event of the SV, several input channels ICEQ's which are used to hold diffusion events from corresponding neighbors, a processed event queue PEQ, a state saving queue (SQ) and a history queue (HQ). Readers who are interested in the multi-level queuing may turn to (Xu and Tropper 2006) for more details.

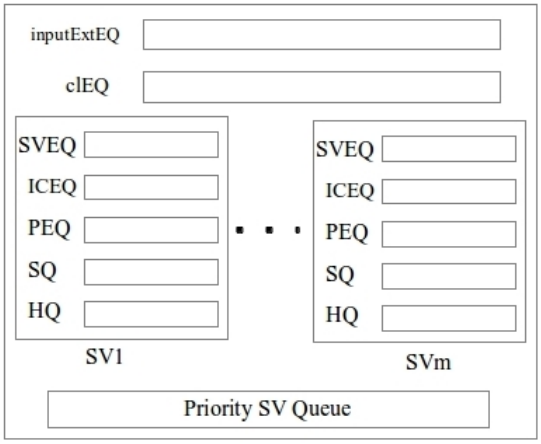

Figure 2: NTW Cluster.

When compared to XTW, the HQ is new. It can be viewed as an array of its neighbors, in which each element of the HQ stores the time stamp of the last event sent to the corresponding neighbor. The HQ is used when the SV needs to be rolled back. For example, suppose that the local virtual time of SV i is 100, its HQ[0] is 80 (this indicates that it did not schedule any event to this neighbor after 80) and HQ[1] is 90. Assume that SV i receives a straggler and needs to roll back to a checkpoint at 88. After retrieving the local virtual time and recovering state it can send rb-messages. It is easy to assert that a rollback-message (rb-message) should be sent to the neighbor defined by HQ[1] whereas we do not need to send an rb-message to the neighbor defined by HQ[0]. In this way, both the communication during the simulation and the cost for rolling back can be reduced.

3.2 Event Flow

The steps for processing events in SVs are determined by NSM. The event set consists of rd-event and diffuse events. In every cluster, the SV's are kept sorted in a binary heap such that the SV for which the first event occurs is at the top. The rd-event is scheduled by the top SV of priority event queue and the diffusion event is scheduled by a neighboring SV which is responsible for the diffusion event, see figure 3.

Upon receiving an event, the major work to perform is finding the host SV and storing the event in the multi-leveled queue, see figure 4.

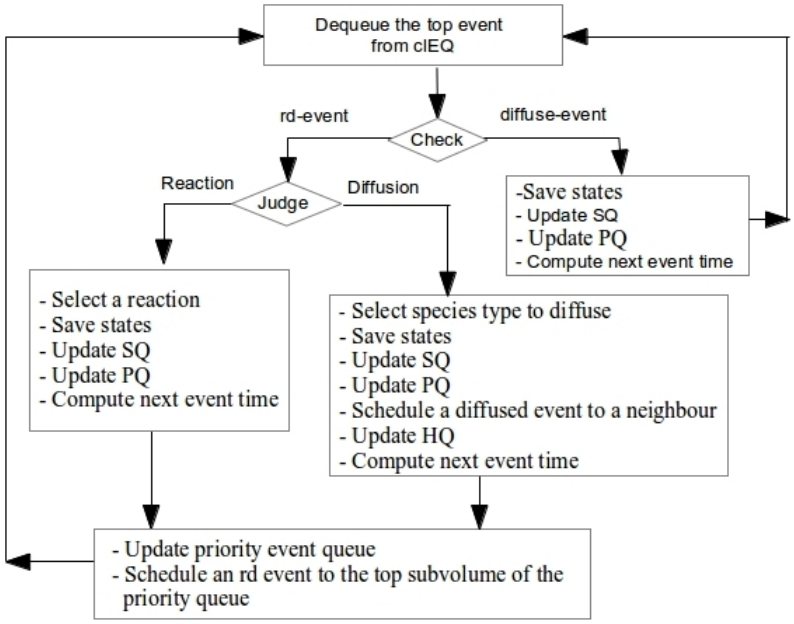

Figure 3: Event Processing.

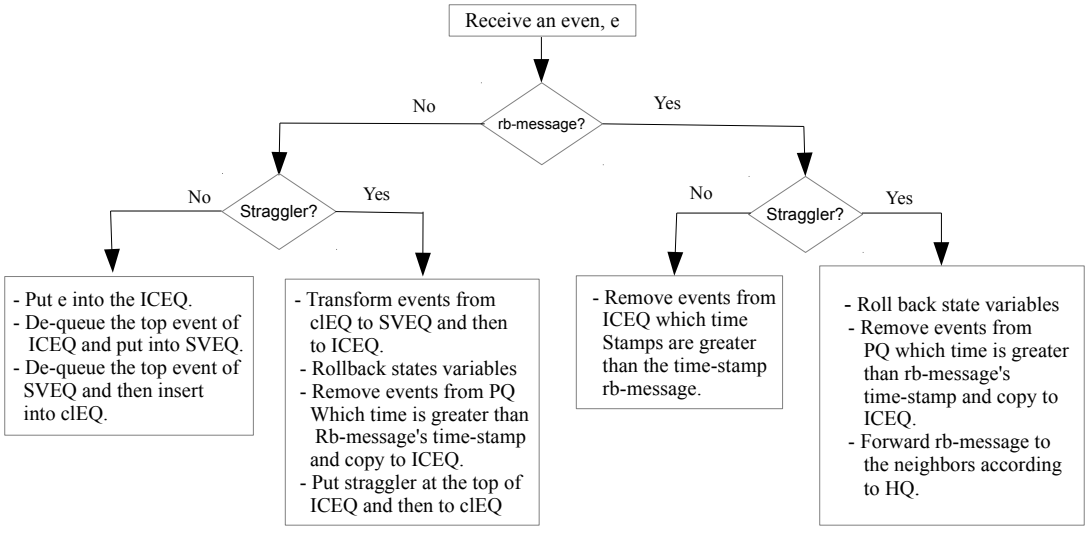

Figure 4: Receiving an Event.

3.3 Shared Memory Communications

Because communication is the main performance bottleneck for PDES applications we employ shared memory to shorten the delay of message passing. We call other LPs located in the same machine a family and give every LP a buffer to receive messages from its family. A semaphore controls access to the buffer. For example, when sending data if LP 0 is about to send data to its (family) LP 2, LP 0 needs to hold the semaphore for LP 2, then find room to write the data, after which it releases the semaphore when writing is finished. When receiving data, LP 0 must first hold the semaphore and then receive the data, after which it releases the semaphore.

4 EXPERIMENTAL WORK AND ANALYSIS

Our experiments made use of the Lotka-Volterra model (Schinazi 1997) to represent chemical reactions and the diffusion of ions taking place on a dendritic branch. The reasons we selected the LV model are that (1) it is simple and we can therefore focus on the simulation and that (2) the LV model is well known so we can verify our experimental results easily.

4.1 Models

The Lotka Volterra model consists of two mobile species, known as the prey and the predator, and 3 reactions shown in figure 5 Each of the species can diffuse from the sub-volumes in which they reside to a (randomly selected) neighboring sub-volume. Their rate of diffusion is governed by a diffusion constant D.

$$R1. \quad prey \xrightarrow{C_1} 2\,prey$$

$$R2. \quad predator + prey \xrightarrow{C_2} 2\,predator$$

$$R3. \quad predator \xrightarrow{C_3} NULL$$

Figure 5: Lotka Volterra Reactions.

The cylinder and branch are basic shapes for modeling neurites in NEURON, so we use a 'Y-shaped' geometry which is taken from a NEURON model to determine the connection of SVs, as shown in figure 6. The Y-Shape geometry consists of 3 cylinders, each 10 microns long and 1 micron in diameter. Each sub-volume is 0.25x0.25x0.25 microns in size, and there are 2766 sub-volumes in total . Initially, each sub-volume had 100 prey and 100 predators. Values for reaction rates C1,C2 and C3 were set to 10, 0.09 and 10 respectively. The diffusion rate Dprey was set to 7.5 $\mu m/ms$ and Dpredator to 5 $\mu m/ms$. To have each cluster involved in simulation own the same number of SVs, we divided the SVs to clusters evenly.

4.2 Verification

The number of species is the state of the Lotka-Volterra model. A portrayal of the interaction of the two species is contained in figure 7. In this figure we plot the population of the prey and the predator for a period of time during which 24,000,000 events were processed. We employ one controller and three workers and use the same initial conditions for all of the sub-volumes. We recorded the state of each sub-volume every 1.0 virtual time units. According to (Wang, Yao, Zhao, Hou, and Peng 2009), we can conclude our experimental results are faithful to Lotka Volterra model.

4.3 Performance

We have two versions of our code, one of which employs MPI alone, while the other makes use of shared memory to pass an event within one physical node. The execution time and the number of rollbacks for

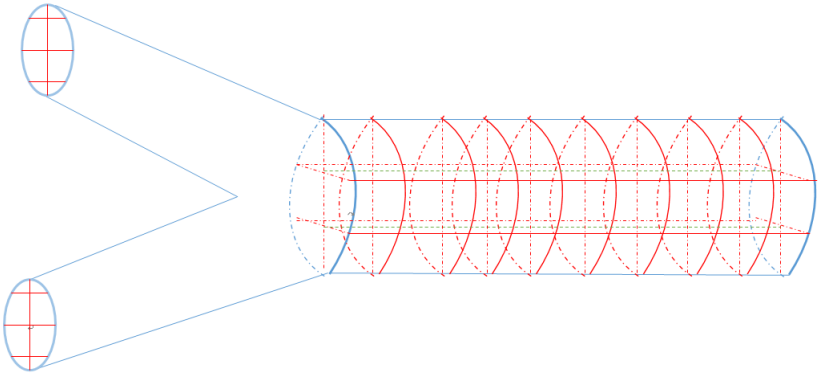

Figure 6: Y-Shaped Geometry.

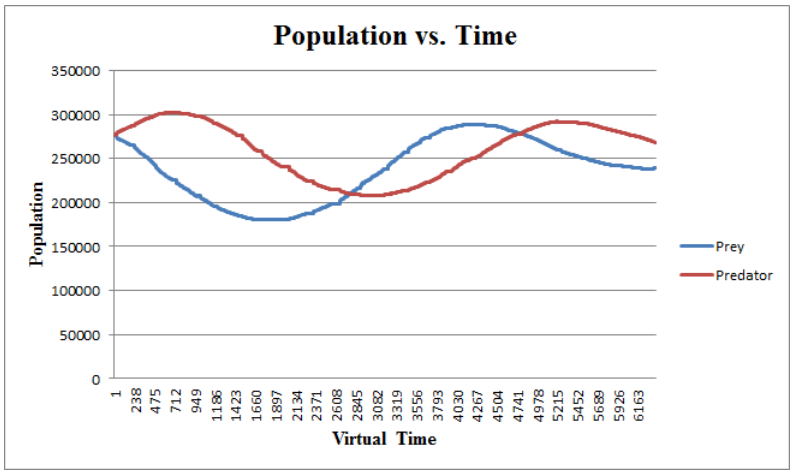

Figure 7: Population of Predator and Prey.

both of these versions are portrayed in figures 8 and 9 on the Y-shaped geometry. The Y-shaped geometry contained 2766 SVs.

Figure 8: Execution Time for Y-Shaped Model.

From these two figures we see that the shared memory version of the simulation time has a lower execution time then the MPI version. The maximum difference of 11% occurs at 4 processors We also see that the MPI version has more rollbacks. The maximum percentage difference occurs with 6 processors, approximately 30%.

In order to determine if shared memory would further improve performance in a larger model we made use of a cuboid of dimension $8\times8\times100$ (thus 6400 SVs in total). The results are depicted in figures 10 and 11 in which we see the expected improvement. There is a 12% maximum difference in the execution time using 4 processors and a maximum difference of 79% in the number of rollbacks using 4 processors.

Diffusion rates affect the probability of a particular type of chemical reaction occurring, thereby altering the state of the system. This motivated us to experiment with different values of the diffusion rate in the Y-Shaped geometry-we doubled the diffusion rate of both of the prey and the predator (Figures 12 and 13).

A similar story plays out in these figures, with the maximum differences for the execution time and the number of rollbacks being 12% and 44% respectively.

These results indicate that better performance is achieved by utilizing shared memory. However, shared memory is not a panacea-the number of rollbacks increases for both approaches after achieving a minimal execution time. This in turn indicates the need for dynamic load balancing and dynamic window management.

5 CONCLUSIONS AND FUTURE WORK

Several observations can be made from the preceding section: (1) NTW scales well with the number of processes used. (2) The use of shared memory improves performance in a multi-core machine. (3) Both dynamic window management and dynamic load balancing are necessary in order to contain the number of rollbacks.

We have previously developed AI based algorithms for dynamic load balancing and window management which will be integrated in NTW. We have also implemented threads in NTW and are investigating the effects on our load balancing algorithms.

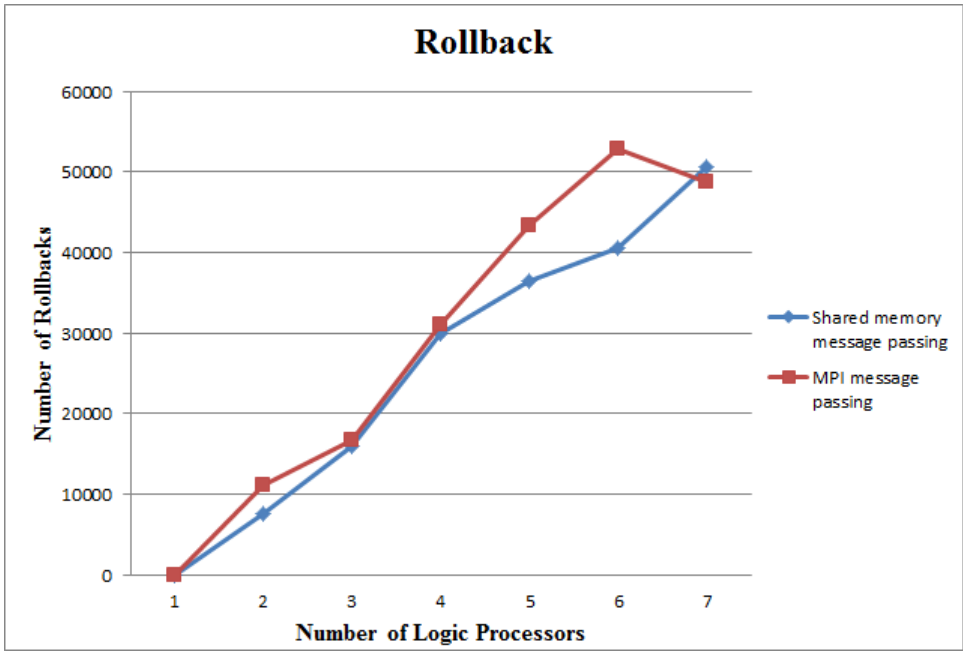

Figure 9: Rollbacks for Y-Shaped Model.

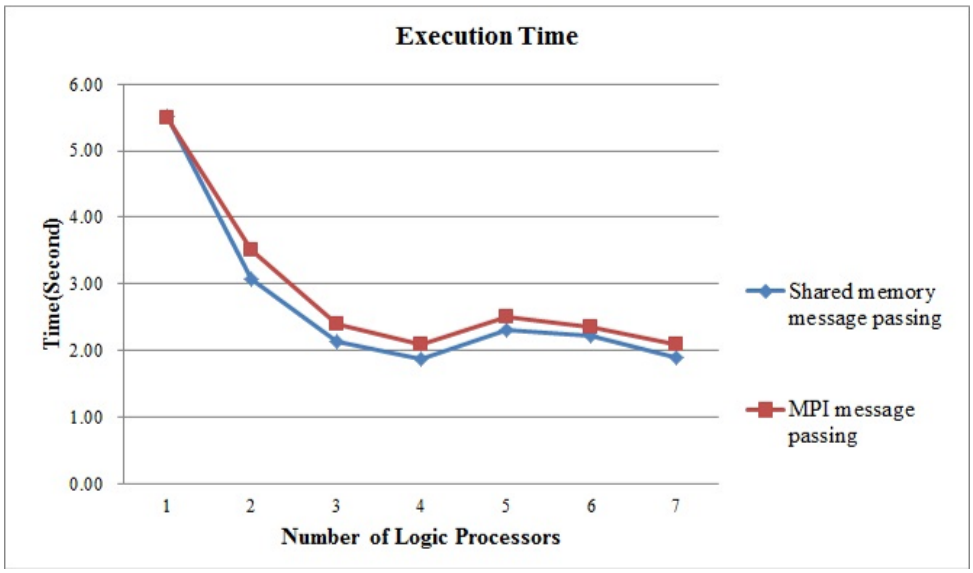

Figure 10: Execution Time for Cuboid Model.

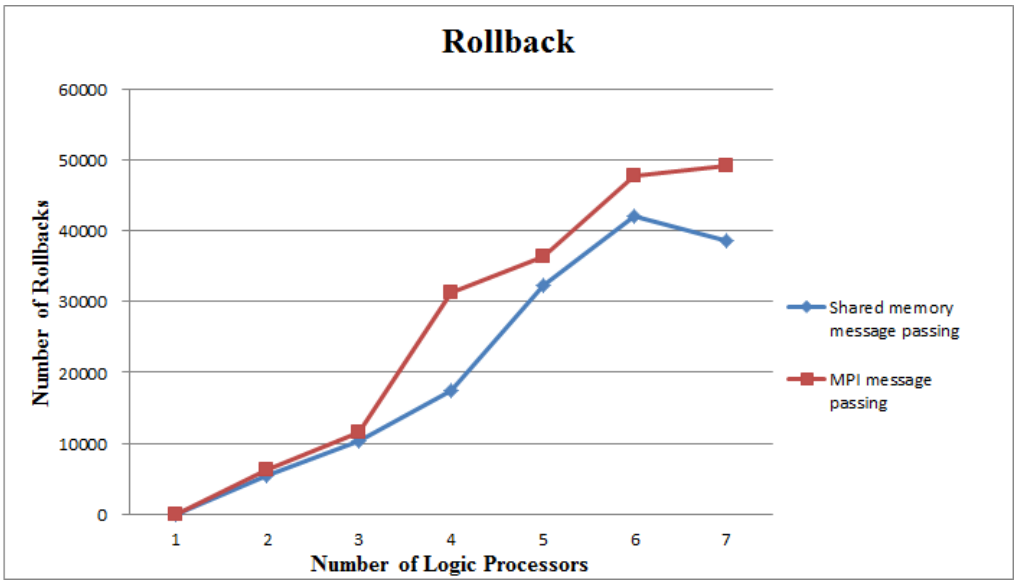

Figure 11: Rollbacks for Cuboid Model.

Figure 12: Execution Time for the Cuboid Model.

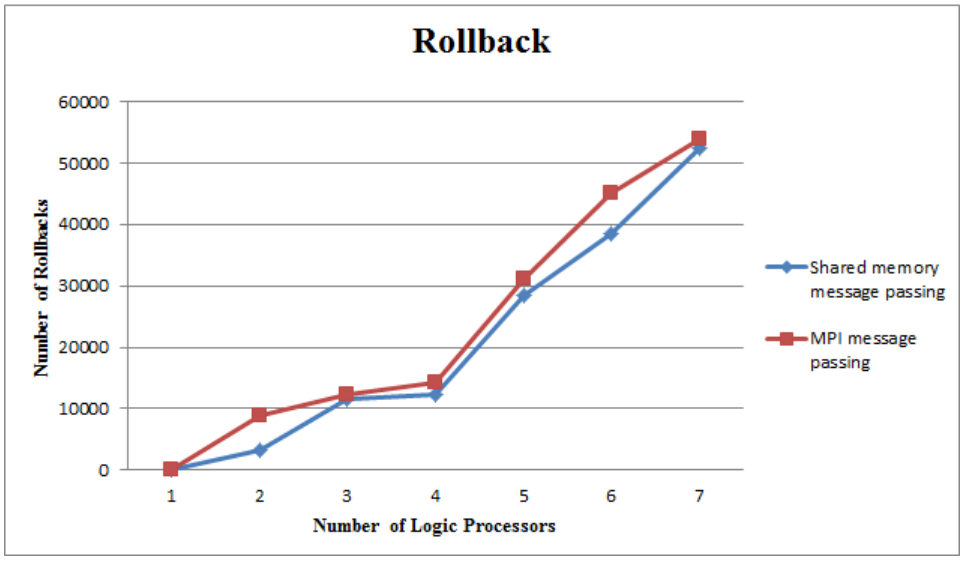

Figure 13: Rollbacks for the Cuboid Model.

Finally, we have integrated our simulator with NEURON (www.neuron.yale.edu). We are coupling our discrete event simulator with the deterministic reaction diffusion described with differential equations, and will investigate Ca wave propagation with this multi-scale combined tool.

This paper described our first steps in the direction of developing parallel multi-scale modeling tools for a neuron. Multi-scale models are needed to complement differential equation-based deterministic models in regions of the neuron which have small numbers of the chemical species being modeled. The stochastic simulation can then portray the stochastic behavior in sections of a neuron where the number of molecules is small, while the paired deterministic simulation portrays the smooth bulk behavior in other parts of the cell where higher concentrations play out across much larger volumes.

ACKNOWLEDGMENTS

This work is funded by China Scholarship Council.

REFERENCES

Avril, H., and C. Tropper. 2001. "On rolling back and checkpointing in time warp". *Parallel and Distributed Systems, IEEE Transactions on* 12 (11): 1105–1121.

Cao, Y., H. Li, and L. Petzold. 2004. "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems". *The journal of chemical physics* 121 (9): 4059–4067.

Carnevale, N. T., and M. L. Hines. 2006. *The NEURON book*. Cambridge University Press.

Dematt, L., and T. Mazza. 2008. "On parallel stochastic simulation of diffusive systems". In *Computational methods in systems biology*, 191–210: Springer.

Elf, J., and M. Ehrenberg. 2004. "Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases". *Systems biology* 1 (2): 230–236.

Gibson, M. A., and J. Bruck. 2000. "Efficient exact stochastic simulation of chemical systems with many species and many channels". *The journal of physical chemistry A* 104 (9): 1876–1889.

Gillespie, D. T. 1977. "Exact stochastic simulation of coupled chemical reactions". *The journal of physical chemistry* 81 (25): 2340–2361.

Gillespie, D. T. 2001. "Approximate accelerated stochastic simulation of chemically reacting systems". *The Journal of Chemical Physics* 115 (4): 1716–1733.

Jeschke, M., R. Ewald, A. Park, R. Fujimoto, and A. M. Uhrmacher. 2008, March. "A Parallel and Distributed Discrete Event Approach for Spatial Cell-biological Simulations". *SIGMETRICS Perform. Eval. Rev.* 35 (4): 22–31.

Lytton, W. 2002. "From Computer to the Brain". *Springer Verlag*.

Schinazi, R. B. 1997. "Predator-prey and host-parasite spatial stochastic models". *The Annals of Applied Probability* 7 (1): 1–9.

Sterratt, D., B. Graham, A. Gillies, and D. Willshaw. 2011. *Principles of computational modelling in neuroscience*. Cambridge University Press.

Takahashi, K., K. Kaizu, B. Hu, and M. Tomita. 2004. "A multi-algorithm, multi-timescale method for cell simulation". *Bioinformatics* 20 (4): 538–546.

Wang, B., Y. Yao, Y. Zhao, B. Hou, and S. Peng. 2009. "Experimental analysis of optimistic synchronization algorithms for parallel simulation of reaction-diffusion systems". In *High Performance Computational Systems Biology, 2009. HIBI'09. International Workshop on*, 91–100: IEEE.

Xu, Q., and C. Tropper. 2006. "Towards large scale optimistic vlsi simulation". *Simulation Modelling Practice and Theory* 14 (6): 695–711.

AUTHOR BIOGRAPHIES

Mohammad Nazrul Ishlam Patoary is a Ph.D. student of School of Computer Science at McGill University. His research area is parallel discrete event simulation. His email is mohammad.patoary@mail.mcgill.ca.

ZHONGWEI LIN is a Ph.D. student of College of Computer at National University of Defense Technology. He received M.S. degree in Computer Science and Technology from National University of Defense Technology. His research area is High Performance Simulation. His email address is zwlin@nudt.edu.cn.

CARL TROPPER is a Professor in the Department of Computer Science at McGill University. His focus over the past several years has been on applying PDES techniques to stochastic discrete event simulations of neurons. His group has developed Neuron Time Warp. His email address is carl@cs.mcgill.ca.

Robert A. McDougal is a Postdoctoral Fellow in Neurobiology at Yale University with a PhD in Mathematics from The Ohio State University. He is a developer for the NEURON simulator and for the SenseLab collection of neuroscience databases. His research focuses on developing computational techniques to study the brain. His email address is robert.mcdougal@yale.edu.

William Lytton is an MD trained at Harvard, Columbia, Alabama, Johns Hopkins, UCSD and Salk Institute. He is a neurologist caring for the indigent at Kings County Hospital, and teaching and researching computational neuroscience at Downstate Medical Center, both in Brooklyn, NY. He is the author of "From Computer to Brain," a basic introduction to computational neuroscience. His research involves the understanding of single-cell and network dynamics with application to neurological and psychiatric disease including epilepsy, stroke and schizophrenia. A major goal of his laboratory (http://it.neurosim.downstate.edu) is the eventual development of personalized medicine through drugs or devices that can be specifically designed to alter functional neurodynamics in the individual patient. His email address is billl@neurosim.downstate.edu.