Optimizing NEURON Brain Simulator with Remote Memory Access On Distributed Memory Systems

Danish Shehzad
Computer Engineering Department,
Kadir Has University
Istanbul, Turkey
danish.shehzad@stu.khas.edu.tr

Zeki Bozkus
Computer Engineering Department,
Kadir Has University
Istanbul, Turkey
zeki.bozkus@khas.edu.tr

***Abstract*—The Complex neuronal network models require support from simulation environment for efficient network simulations. To compute the models increasing complexity necessitated the efforts to parallelize the NEURON simulation environment. The computational neuroscientists have extended NEURON by dividing the equations for its subnet among multiple processors for increasing the competence of hardware. For spiking neuronal networks inter-processor spikes exchange consume significant portion of overall simulation time on parallel machines. In NEURON Message Passing Interface (MPI) is used for inter processor spikes exchange, MPI_Allgather collective operation is used for spikes exchange generated after each interval across distributed memory systems. However, as the number of processors become larger and larger MPI_Allgather method become bottleneck and needs efficient exchange method to reduce the spike exchange time. This work has improved MPI_Allgather method to Remote Memory Access (RMA) based on MPI-3.0 for NEURON simulation environment, MPI based on RMA provides significant advantages through increased communication concurrency in consequence enhances efficiency of NEURON and scaling the overall run time for the simulation of large network models.[1]**

Keywords—Neuroscience, MPI, Remote Memory Access, Parallelism, NEURON, simulation environment.

INTRODUCTION

Simulations of large scale neuronal networks have become important means for studying brain complex computational behavior. Models of neuronal networks are used to integrate large amount of data and computations on those models are performed using simulation tools to simulate the behavior of brain for processing information. Simulators have also helped for modeling various parts of brain. Through processing of network models better understanding of neuronal networks is developed enabling neuroscientists to virtually observe the behavior of brain and perform experiments along with changing mechanisms on simulating environment. This helps in developing better understanding of brain and can lead to the grounds for eradication of many diseases like epilepsy, Parkinson's disease, etc.

There is wide range of simulators for neuronal simulations. The beneficial feature of the diverse simulation environments are that each simulator has wide-ranging strengths and this diversity results in better development, understanding and simulation of large neuronal models. This diversity also helps the advancement in overall simulation environments enabling to the development of new perspectives in simulation technology. However, the major drawback in existing setup is lack of interoperability because of different type of languages used. These simulation environments and models also helps computer scientists to understand and develop new methodologies for the development of fast and efficient computer architectures that can provide parallel, swift and efficient processing. Few neuronal simulation tools have been developed for simulating neurons and their complex models, leading tools at the current are mainly NEST, NEOSIM and NEURON[1-5].

A. Parallel Simulation in NEURON

For enhancing the performance of NEURON simulator it has been extended to support parallel environments by utilizing the machines in efficient manner[6].NEURON is becoming one of the most suitable tool for building neuron models, managing and using them for solving complex neuronal computational problems. For neuronal simulations running on parallel processors, inter-processor spikes exchange takes considerable portion in total simulation time. The MPI_Allgather method used in Neuron simply uses spike exchange after integrating the cell equations for minimum specific time taken between spike initiation and spike delivery. MPI 2.0 based on two sided communication uses the same technique of first gathering on a processor and then distributing among all other processors in the communication world. Typically MPI_Allgather is an effective method for neural networks given a source cell connected to thousands of target cells in large network models.

2. MPI

Communication between parallel processors running multiple processes takes place through message passing library. Message passing interface is an application programming interface for writing message passing parallel programs which hides the hardware and software details of the underlying system[7]. MPI is implemented as library it enables portable program that can manage to run same program on parallel

978-1-5090-0436-2/15/$31.00 ©2015 IEEE

processors. The communication through MPI can be between two specific processes and is known as point to point communication whereas collective communication involves communication among all processes in process group.

A. MPI Two-Sided Communication

MPI two sided communication provide semantic guarantees implied by the standard and its implantations are subject to various practical constraints. MPI basic communication pattern is based on two calls MPI_Send and MPI_Recv. MPI_Send routines are used for sending messages from origin process to target process and MPI_Recv for receiving messages on target process sent by origin process.

There are certain limitation constraints in two sided communication affecting the programs efficiency. As matching order of Send/Recv and message collections results in restricting choice for hardware message ordering. Also support is required at receiver side for handling message size ambiguity and message probing results in corresponding restraints for buffer allocation or memory registration. Two sided communication (Figure 1) through MPI_Send and MPI_Recv also blocks the communicating processors until the complete transfer of data from sender to receiver takes place entailing synchronization services and data exchange handling.

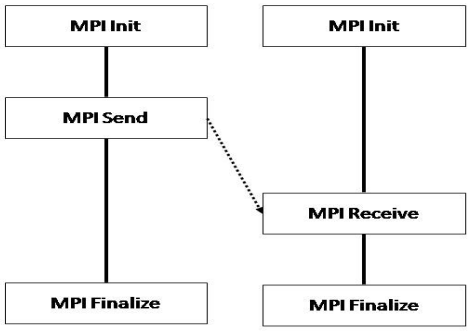

Figure 1. Two Sided Communication in MPI

B. MPI One-Sided Communication

All functions of MPI RMA communication take place in the framework of a window. Window consists of assembly of elements which are defined at the time of creation of window and adjacent area of memory at each process. The exposed memory of a window can be retrieved by using by using one-sided communication functions. The use of MPI-3.0 standard includes three types of one sided communication operations MPI Put, MPI Get, and MPI Accumulate. Transmission of data from origin to target is achieved by using PUT Operation[8]. On contrary to this the get operation receives data from the target window to the origin(Figure 2), accumulate operation combines data into the target from the origin, thus become applicable by using MPI reduction operator which restricts data into the buffer.

The occurrence of communication operations take place in context of either an active target synchronization epoch or a passive target synchronization epoch. All communication operations are non-blocking and are completed at end of synchronization epoch.

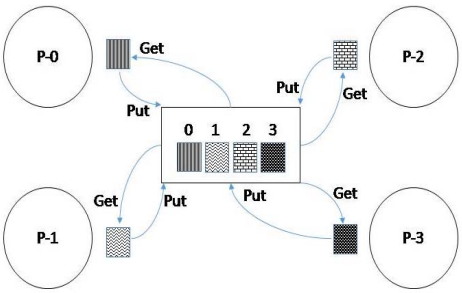

Figure 2.MPI One Sided Communication

C. MPI_Allgather Collective

MPI_Allgather is one of the collective operations for handling complex communications and is built basically on point-to-point communication routines. Collective communication is the process of sending and receiving data among all the processors in MPI communication world(Figure 3). There are other collective communication routines for other message passing libraries but the complete and robust collective communication routines belong to MPI. MPI_Allgather collective routine is based on two main communication steps . In the first step the processes in the MPI communicator's world gathers data from every other processes and along with that distribute its data among all other processes.

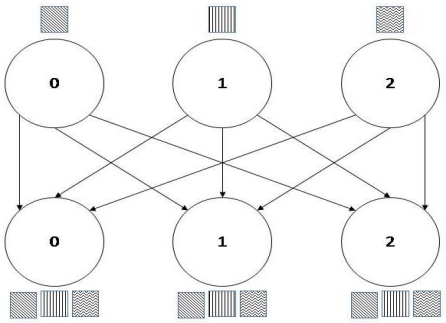

Figure 3.MPI_AllgatherTwo Sided Communication

3. Related Work.

Many studies illustrate that distributing network architecture over multiple processors has advantage of fast processing of data. For instance the scaffold working in NEURON for parallel simulations and performance scaling is achieved by testing the proposed models[9]. Simulations of large spike-coupled neural networks use parallel model for efficient simulation on large computer clusters[10].

Many simulation environments have the capability of provisioning desired functionalities including NEST[11], pGENESIS[12], SPLIT[13], NCS[1], C2 [14]. These simulation environments give many advantages like increase in simulation speed with increase in number of processors, the rate of communication is limited until each processor had very little work to do. Inter-processor spikes exchange is one of the most important factors to be considered in parallel network

simulators. A standard Message Passing Interface (MPI) is widely adopted by most of simulators and functions on use of the non blocking point-to-point message passing utility. NEURON selects basic spike distribution method, which functions to distribute task spikes among all processors[9]. The "Allgather" technique uses MPI_Allgather, and irregularly MPI_Allgatherv but when there are additional spikes to be sent than fit in the fixed size MPI_Allgather storage [15]. The major objective is to get a baseline for future assessment with more advanced point-to-point routing methodologies. For instance, use of NEST, observed that Allgather give improved performance on their 96 core cluster using infiniband switch in comparison with Complete Pair wise Exchange algorithm[10]. The use of MPI_Allgather method increases the communication time along with increase in number of processors.

Table 1: MPI_Allgather time vs Computation time

No. of Processors	Parbulbnet				Parscalebush			
	Total Run Time	MPI_Allgather time	%age MPI_Allgather time	%age Computation Time	Total	MPI_Allgather time	%age MPI_Allgather time	%age Computation Time
10	28.57	5.64	20	80.3	111.56	15.6	13.9	86.1
20	14.22	3.17	22	77.7	41.25	6.97	16.9	83.1
40	9.05	3.03	33	66.5	22.35	6.83	30.5	69.5
50	7.24	2.52	35	65.2	19.44	6.5	33.4	66.6
60	9.13	4.05	44	55.7	16.47	5.92	35.94	64.06
80	7.42	3.44	44	55.6	13.8	5.09	36.8	63.2
100	7.66	3.89	51	49.2	12.2	4.6	37.7	62.3
110	9.45	4.89	52	48.3	12.22	4.93	40.34	59.66
120	8.83	4.83	55	45.3	12.61	5.2	41.23	58.77

4. Proposed Solution.

To analyze the limitations of MPI_Allgather method we performed the simulation test on two published neuronal network models Parbulbnet and parscalebush, exhibiting different spike patterns. These models were downloaded from the ModelDB repository(http://senselab.med.yale.edu) and used parallel models from Netmod [9].

The limitations of MPI_Allgather were observed during gradual increase in number of processors where the size of subnet on single processor become smaller and MPI_Allgather becomes source of communication overhead. This can be seen in the following readings taken for Bush model with 5000 cells and Parbulbnet models(Table 1), where it can be seen that along with increase in number of processors the communication time begin to dominate computation time.

A. Problem Analysis Algorithm

1. Define Master (processor-0)
2. Associate cells to processor
3. for every processer x
4. {
5. Set internal and external connectivity
6. find Max Step Time
7. for nrn_spike_exchange
8. {
9. Get the message for MPI_Allgather
10. Repeat this until t-stop
11. }
12. Integrate the total spike exchange message on processer x
13. }

Get spike message from all processors at processer-0

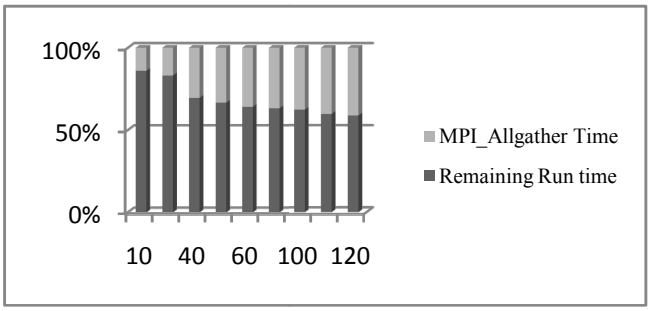

Figure 4.Parscalebush Analysis graph for MPI_Allgather vs Remaining Time

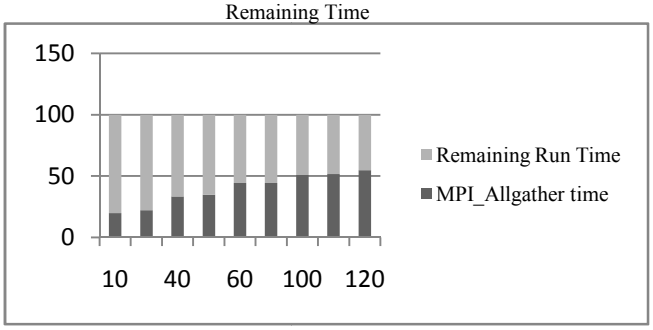

Figure 5. PabulbNet Analysis graph for MPI_Allgather vs Remaining Time

The current MPI_Allgather limitation was analyzed by above steps through making changes in most time consuming portion of nrnmpi.c and mpispike.c. To compute the integrated time few changes in the existing code were made for MPI_Allgather, so that spike exchange time could be revealed out of total run time as shown for Parbush model(Figure 4) and Parbulbnet model(Figure 5). The MPI_Allgather time was compared to the total run time and its limitation was confirmed.

B. Proposed Allgather algorithms with RMA

The normal process as for MPI_Allgather was followed but after the association of cells to processors in our solution RMA_Allgather API resolve around the use of "windows". On each processor a window which is specified region of memory is made available for remote operations by the other MPI processes. Windows are created by collective operation

MPI_win_create, which specifies base address and length of window, all MPI one sided operations are non-blocking.

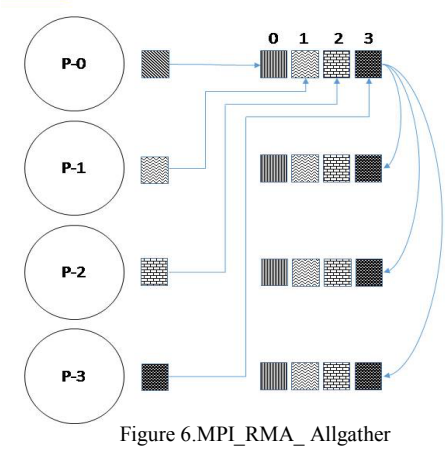

Figure 6.MPI_RMA_ Allgather

In NEURON simulation environment instead of involving both sender and receiver during communication window creation allows one sided communication which diminishes the involvement of both sender and receiver and remote operations are performed on windows which reduce spikes exchange communication overhead. After each specified interval all processes write their portion remotely trhough MPI_Put to the window of master processor(Figure 6), which broadcasts the integrated message to all the processors thus reducing the communication bottle neck of MPI_Allgather.

C. RMA_Allgather Algorithm

1. Define Master (processor-0)
2. Associate cells to processor
3. for every processer x
4. {
5. Set internal and external connectivity
6. find Max Step Time
7. for nrn_spike_exchange
8. {
9. Write all message at processer-0 window by RMA
10. Repeat this until t-stop
11. }
12. }
13. Integrate the total spike exchange message on processer 0
14. processer-0 broadcast the message to every processer window

5. Conclusion

We presented RMA_Allgather method for spikes exchange in NEURON simulation environment for improving its efficiency for simulating large network models. Speed up from parallelizing large network models is found nearly proportional to number of processors but spike exchange time is found inversely affecting run time that along with increasing number of processors exchange time is consuming more time as compared to computation time. To overcome this limitation a new method RMA_Allgather is proposed that will reduce spikes exchange time almost 15% and would be implemented in NEURON simulation environment. We plan to implement this method and explore multiple RMA_Allgather algorithms to improve the performance of NEURON for simulating large network models.

ACKNOWLEDGMENTS

Danish Shehzad and Dr. Zeki Bozkus are funded by the Scientific and Technological Research Council of Turkey (TUBITAK; **114E046**).

REFERENCES

[1] E. C. Wilson, P. H. Goodman, and F. C. Harris Jr, "Implementation of a Biologically Realistic Parallel Neocortical-Neural Network Simulator," in *PPSC*, 2001.

[2] A. Morrison, C. Mehring, T. Geisel, A. Aertsen, and M. Diesmann, "Advancing the boundaries of high-connectivity network simulation with distributed computing," *Neural computation,* vol. 17, pp. 1776-1801, 2005.

[3] A. Delorme and S. J. Thorpe, "SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons," *Network: Computation in Neural Systems,* vol. 14, pp. 613-627, 2003.

[4] N. Goddard, G. Hood, F. Howell, M. Hines, and E. De Schutter, "NEOSIM: Portable large-scale plug and play modelling," *Neurocomputing,* vol. 38, pp. 1657-1661, 2001.

[5] P. Hammarlund, Ö. Ekeberg, T. Wilhelmsson, and A. Lansner, "Large neural network simulations on multiple hardware platforms," in *Computational Neuroscience*: Springer, 1997, pp. 919-923.

[6] M. L. Hines and N. T. Carnevale, "Translating network models to parallel hardware in NEURON," *Journal of neuroscience methods,* vol. 169, pp. 425-455, 2008.

[7] K. Z. Ibrahim, P. H. Hargrove, C. Iancu, and K. Yelick, "An evaluation of one-sided and two-sided communication paradigms on relaxed-ordering interconnect," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, 2014, pp. 1115-1125.

[8] S. Potluri, H. Wang, V. Dhanraj, S. Sur, and D. K. Panda, "Optimizing MPI one sided communication on multi-core infiniband clusters using shared memory backed windows," in *Recent Advances in the Message Passing Interface*: Springer, 2011, pp. 99-109.

[9] M. Migliore, C. Cannia, W. W. Lytton, H. Markram, and M. L. Hines, "Parallel network simulations with NEURON," *Journal of computational neuroscience,* vol. 21, pp. 119-129, 2006.

[10] M. Hines, S. Kumar, and F. Schürmann, "Comparison of neuronal spike exchange methods on a Blue Gene/P supercomputer," *Frontiers in computational neuroscience,* vol. 5, 2011.

[11] M.-O. Gewaltig and M. Diesmann, "NEST (neural simulation tool)," *Scholarpedia,* vol. 2, p. 1430, 2007.

[12] M. Hereld, R. Stevens, J. Teller, W. Van Drongelen, and H. Lee, "Large neural simulations on large parallel computers," *International Journal of Bioelectromagnetism,* vol. 7, pp. 44-46, 2005.

[13] M. Djurfeldt, C. Johansson, Ö. Ekeberg, M. Rehn, M. Lundqvist, and A. Lansner, "Massively parallel simulation of brain-scale neuronal network models," 2005.

[14] R. Ananthanarayanan and D. S. Modha, "Anatomy of a cortical simulator," in *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*,2007,p. 3.

[15] S. Kumar and et al., "The Deep Computing Messaging Framework: Generalized Scalable Message Passing on the Blue Gene/P Supercomputer.," in The 22nd ACM International Conference on Supercomputing (ICS), 2008.