ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 11(39), DOI: 10.17485/ijst/2018/v11i39/130826, October 2018*

Massively Parallel Computational Schemes for Simulating Spiking Neural Networks Using GPU Accelerators

N. Sreenivasa[1] and S. Balaji[2]

[1]Department of Computer Science and Engineering, Jain University, Nitte Meenakshi Institute of Technology, P.O. Box 6429, Yelahanka, Bengaluru – 560064, Karnataka, India; meetcna@yahoo.co.in

[2]Centre for Incubation, Innovation, Research and Consultancy, Jyothy Institute of Technology, Tataguni, Off Kanakapura Road, Bengaluru – 560082, Karnataka, India; drsbalaji@gmail.com

Abstract

Objectives: To review various tools available for simulating Spiking Neural Networks using heterogeneous parallel processing platforms that help to reduce cost, increase the computational speed and also to document/archive lessons learnt. **Methods/Statistical Analysis:** The computational speed is a continuing challenge for simulating genuine spiking neural network models. Understanding of the spiking neural networks is significantly simplified by computer simulators like NEST, GeNN, EDLUT and BRIAN. **Findings**: Simulation is a handy toolkit of scientists and engineers of all disciplines. NEST, GeNN, EDLUT and BRIAN simulators help in achieving better performance not in terms of same kind of processing but with additional special tasks which require more computational power. BRIAN and EDLUT which are hybrid simulators supports both time driven and event driven techniques and outperform when compared to other simulators. **Application/Improvements**: Using BRIAN and EDLUT simulation techniques we can achieve the high performance when compared to other spiking neural simulation techniques.

Keywords: Graphics Processing Unit (GPU) Accelerators, Massive Parallel Computation, Simulation, Spiking Neural Network

1. Introduction

The simulation of Spiking Neural Networks (SNN) is normally a difficult task that involves a large number of calculations, particularly while handling large scales SNNs. The limitations emanate primarily from the demands on the computer resources such as Modern Graphics Processing Unit (CPU) and storage. As SNNs become bigger with added complexity the required processing power increases exponentially. Recent advancements in the designs of SNN simulators facilitate decomposition of computational load into several parts that can beprocessed on many processors. The systematic mapping of computational load to the processing elements reduces the complexity of implementing simulation techniques and improves the efficiency of the simulator with accelerated computational processes. GPUs are low cost supercomputers devoted tohandle massively parallel computations that can be used as computational accelerators[1].

*Author for correspondence

Of late, GPU enabled massively parallel processing has attracted significant interest of the researchers in view of the possibilities of dramatically accelerating computations using off-the-shelf GPU modules. The models or simulations built on GPUs using parallel programming paradigmsresult in accomplishing quicker results in scientific and engineering research. Using Open Computing Language (OpenCL) or Compute Unified Device Architecture (CUDA) solutions to various problems of real world can be implemented easily and run faster compared with multicore or multiprocessor systems[2-3]. Building clusters of heterogeneous computing cores is another promising approach for data processing and parallel computation in enterprise computing.

2. Spiking Neural Networks Modeling

The spiking neural network simulation is close to the realism in third generation of neural networks. In addition to the neuronal and synaptic states, SNNs also integrate the idea of time into the operating model. The idea is that neurons in SNN do not fire at every propagation cycle, but rather fire only when membrane's charge touches a specific neuron[4]. When the neuron fires,it produces the signal which passes to another neuron which in turn, rises or reducesits potentials with respect to this signal. Simulation of SNNs is most inspiring computational job because of its numerical calculations and the simulation time. For a realtime simulation, a large number of numerical calculations needto be massively parallelized. A basic neural network is shown in Figure 1, inwhich the neurons are specified by the label N_j and the presynaptic relations X_i of the neuron are signified as small circles with weight W_i.

The synaptic weight calculation formula is as follows:

$$\text{Synaptic weight}[j] = \sum_{1}^{n} X_i W_i$$

The present activation state in SNN simulation is usually measured as neuron's state, with incoming spikes passing spike values to the next level. Different coding approachesare available for understanding the departing spike chain as real significance number, either trusting on the regularity of spikes, or timing between spikes, and to encode data[5-6]. This type of neural network is used for

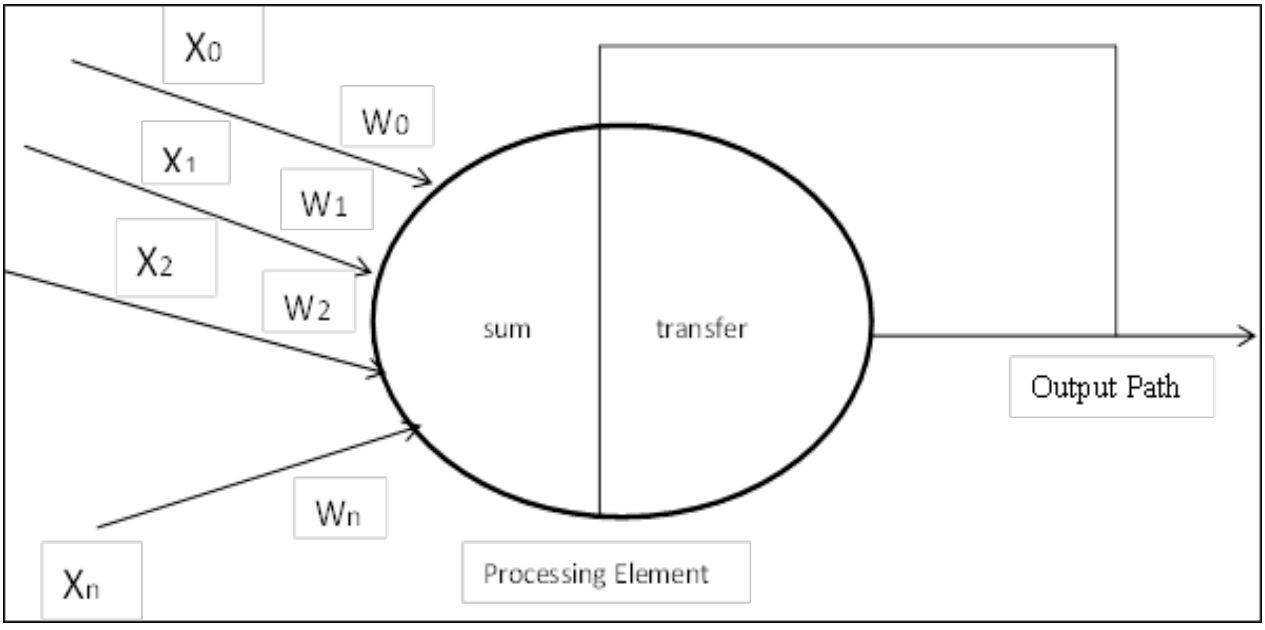

Figure 1. Simplified spiking neural network.

information processing requirements as the technique similar to the customary artificial neural networks. The SNNs can model computer-generated central nervous structure. They can also be used to learn the process of biological neural networks.

The real-worlduse of large scale SNNs is limited due to the exponential increase of computationalprice, which arerelatedtothe accurate simulation of neural models with thecomputational power. As a result, there has been alimiteduse of large scale SNNs to resolve computational jobsdealtbysecond generation neural networks[7-8]. It is challenging to adjust second generation neural models to real time SNNs though it is easy to create and observe its dynamics, but difficult to improve with steady performance. Spiking neural networks acquire biological patterns; it is possiblefor nervous structure to train processes which usuallyoccur in the brain, but they are notentirely understood. Particularly, an SNN simulator permits the study of the theorythat helps in the study of procedures such as:

- Validation of how a theoretical model mimics biological procedure,
- Research of mentalsicknesses by imitating brain sicknesses, and
- Analysis of how the medicine disturbs the brain.

The aim of research involving SNNs is to learn how the human brain functions. Since the beginning of computer science, computers have constantly spent high level of computational power for simulating biological networks, but they are not close to mimic the nature they created. The computational principles of human brain are basically modeled in artificial neural networks which permit there modeling of human capabilities. Thus, networks are used for various machines learning jobs such as pattern recognition and process estimations. The neurons essentiallytransferspikes with other neurons by electrical pulses. The spikes transferred by neurons are very accurate in nature and,they are known as Spike Timing Dependent Plasticity (STDP).There are three different types of spiking neuron models, namely, (i) Integrate and Fire Neuron Model, (ii) The Hodgkin Huxley (H&H) Neuron Model, and (iii) The Izhikevich Neuron (I&N) Model[9-10].

2.1 Integrate and Fire (I&F) Neuron Model

Neuronal dynamics can be conceived as a summation process (sometimes also called integration process) combined with a mechanism that triggers action potentials above some critical value. Neuron models where action potentials are described as events are called Integrate and Fire models, whichhave two separate components that are both necessary to define their dynamics. First is the equation that describes the evolution of the membrane potential $U_i(t)$ and second is the mechanism to generate spikes.

Following are the two ingredients used in Integrate and Fire Neuron Model:

- A linear differential equation to describe the evolution of the membrane potential, and
- A threshold for spike firing; this model is called the 'Leaky Integrate-and-Fire' Model.

2.2 The Hodgkin Huxley(H&H) Neuron Model

The Hodgkin Huxley model or conductance based model is a mathematical model that describes action potentials in neurons that are initiated and propagated in the neurons. It is a set of nonlinear differential equations that approximates the characteristics of excitable cells such as neurons, which is a continuous time model. The neuron layer is signified as a conductance represented by Cm. The motivating gradients initiates the movement of neuronsthat are signified by the dv_m (Membrane Voltage source), whose dv_m value are identified using the ratio of the intra and extracellular absorptions of neurons. Current source I_c is represented by:

$$I_c = C_m(dv_{m/}dt)$$

2.3 The Izhikevich Neuron (I&N) Model

The Izhikevich neuron model simulation is costly and complex to simulate SNNs, because it involves vari-

ous complex differential equations. Equations which are capable to duplicate various rich firing patterns attainable with Hodgkin Huxley neuron model are used to simulate neurons[11]. This simulation model exhibits high performance similar to I&F model.

3. Popular Spiking Neural Network Simulators

3.1 The SNNs Simulating on CPU

A study of SNNs simulation platforms on CPU is reported in literature[12]. Also, the NEURON simulator[13] is widely used because of its support for design and evaluation of different SNN models. The NEURON simulator combines botheventdriven and time driven simulation modes. Furthermore, this simulator is suitable for performing parallel processing on both multi-processorand multicore systemsby distributed processes and threadsin system clusters as per MPI standards. This is available for UNIX, MS Windows and Linux environments. It is also available for IBM Blue Gene and Cray supercomputers.

The Neural Network Simulation Tool (NEST)[14] is developed as the outcome of collective advanced technology projects for the simulation of spiking neural networks. It is intended to simulate the large scale SNNs with the heterogeneity in both synapses and neuron types' simulations. Parallel processing is achieved on multiprocessor systems and cluster of systems through threading and message passing. Simulator outfits time driven simulation method and is available forLinux, UNIX, Mac OS and MS Windows environments. The NEST simulator is released to the scientific community with open source license.

The Brian simulator[15-16] is extremely exile and extensible for the simulation of SNNs on all operating system environments including MS Windows, Linux and Mac OS. This simulator is very popular in research community since it is easy to use and learn and can simulate various spiking neuron models such as H&H and I&F and can be extended to support other models. Brian simulator is coded in Python to take advantage of different techniques coded and released as Python libraries. Some such examples are SciPy and NumPy that can be used for statistical designs and PyLab that can be used for graphical visualization. Python can also be useful in the parallelization processes.

Mvaspike[17] is a general purpose tool for simulating huge and complex spiking neural networks. This simulator is equipped with event driven simulation approach with emphasis on SNNs simulation. A novel balance between simulation efficiency and the modeling liberty is provided using this simulation technique. The simulator is implemented using C++ though the use of the simulator from different programming language environments

Table 1. Select spiking neural network simulators

Feature	BRIAN	NEST	NEURON	NeMo
Time Driven	Yes	Yes	Yes	Yes
Event Driven	Yes	No	Yes	No
GPU	Yes	No	No	Yes
Linux	Yes	Yes	Yes	Yes
Windows	Yes	No	Yes	No
Easy installation	Yes	Yes	No	No

is not difficult. A parallel execution is also available for multi-processorsystems and clusters as shown in Table 1.

3.2 SNNs Simulation Using GPU

We have investigated various platforms used for simulating the SNNs on CPU in the previous section. The good performance NeMo platforms popularly used for the simulation of large scale SNNs[18-19]. NeMo simulator supports various SNN models such as I&F and Izhikevich neuron models enabled by CUDA on GPU and implements an algorithm called scatter gather messaging. The platform supports additional optimization to the unplanned sparse connectivity and to the actual simulations. Furthermore, the simulator supports Matlab, C and Python. NeMo simulator is released with open source license.

The GPU Enhanced Neuronal Networks (GeNN) simulator[20-24] facilitates features to simulate SNNson a GPU equipped system. This simulator is an open source archive developed using CUDA and C/C++acceleratesperformance of the neural simulationusing NVIDIA GPU cards. GeNN simulator is extendable in the sense that every neuron structure can be simulated using this simulator. In GeNN simulation, researchers can host their specific synapse models, neuron models and integration models which are automatically replaced with neural simulation models through code generation. The GeNN simulator is available for Mac OS, Linux and MS Windows environments.

The Myriad[21] simulator focuses on simulations such as H&H neuron models using CUDA on GPU and also supports simulations on clusters. Myriad simulator delivers an extensible and flexible interface with Python, which is later interpreted into a C, based code.

4. Hybrid Simulation of Spiking Neural Networks

The hybrid simulation techniques such as event-driven simulators relatively use simple neural network models defined by various differential equations.These simulations arecarried outfrequentlytoobserve random spike response times. The hybrid simulator such as Event Driven Look up Table (EDLUT),which is an open source simulator, was designed to simulate very large-scale spiking neural networks[22].The look up tables in EDLUT simulator is used to store all expected values of spike fire times.Thus, the total SNN model simulation is coded as aset of classification tables. The tool uses Runge Kutta techniques for the numerical computations andfast simulation of large scale SNNs are feasible in the EDLUT's computational processes. This simulator supports both event driven and time driven techniques thus making this technique as a suitable hybrid technique for simulating the biological spiking neural networks[23].

4.1 Simulation Techniques for Hybrid CPU-GPU Architecture

The EDLUTsimulation can be classified into three computationalsteps:

1. The neuronal dynamics;
2. The spike propagation; and
3. The event of the queue management.

Hence, the EDLUT simulation time isclassified into time neu dynamics, time spike prop, and time queue manage, correspondingly. The simulator uses methods like Integrate and Fire (I&F) Neuron Model and Hodgkin Huxley (H&H) Neuron Model to increase the neuron dynamic computations, which makes EDLUT simulationtoachievegood performance when the neural network integration stage: (i) succeeds over (ii) and, (iii) listed above.

Thetime-driven parallelizing techniques in CPU with GPU improve the performance of the simulation while preserving its correctness and flexibility. The parallelizing technique uses the integration time periods, therefore, gaining more accurate results. Additionally, event-driven simulation techniques are best for the simulationof simple neuron network models with extraordinary performance.

In the computational process, GPUs are known to speed up computations for the time driven simulation techniques because of their parallel design. The GPUs rely on CPU for their scheduling of computations; while

the CPU initiates the simulation,the GPU completes the complete simulation process by executing the scheduled computations. This means that though EDLUT is successively running, GPU evaluates the neural variables in time driven techniques, whereas CPU produces, propagates spikes, and practices learning procedures in both time driven and event driven techniques.

5. Summary and Conclusion

The simulation technique in spiking neural networks plays an important role in modern research whether it is pure or applied sciences. The complexity of biological spiking neural networks, subsystems and interactions among them demand huge computing resources to simulate them. Advancements in heterogeneous parallel processing have helped in handling the complexities of simulating the biological spiking neural systems by utilizing the multicore CPU and many-core GPU architectures. Heterogeneous parallel processing for the simulation of biological systems in medical applications involves developing of algorithms, data structures and other tools to model biological systems and implementing them on heterogeneous parallel processing platforms. The study provides an overview of simulators for spiking neural networks that have good potential for modeling and simulation using heterogeneous parallel processing platforms.

6. References

1. Gerstner W, Kistler W. Spiking Neuron Models. Cambridge University Press; 2002. p. 1–28. https://doi.org/10.1017/CBO9780511815706, https://doi.org/10.1017/CBO9780511815706.002. PMid: 11842739.
2. Hwu WM. GPU Computing Gems Emerald Edition, 1st Edition. Morgan Kaufman; 2011.
3. Carnevale N, Hines M. The NEURON Book. Cambridge University Press; 2006. https://doi.org/10.1017/CBO9780511541612.
4. Izhikevich EM. Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. The MIT Press Cambridge; 2007. PMid: 17220510, PMCid: PMC4437488.
5. Vreeken J. Spiking neural networks: An introduction, Tech. Rep., Artificial Intelligence laboratory, Intelligent Systems Group, University of Utrecht, 2003.
6. Brette R. Simulation of networks of spiking neurons: A review of tools and strategies, Journal of Computational Neuroscience. 2007; 23(3):349–98. https://doi.org/10.1007/s10827-007-0038-6. PMid: 17629781, PMCid: PMC2638500.
7. Brette R, Goodman DF. Simulating spiking neural networks on GPU, Network. 2012; 23(4):167–82. https://doi.org/10.3109/0954898X.2012.730170. PMid: 23067314.
8. Chessa M, Bianchi V, Zampetti M, Sabatini SP, Solari F. Real-time simulation of large-scale neural architectures for visual features computation based on GPU; Network. 2012; 23(4):272–91. https://doi.org/10.3109/0954898X.2012.737500. PMid: 23116085.
9. Anantha Narayanan R, Modha DS. Anatomy of a cortical simulator. SC '07: Proceedings of the 2007 ACM/IEEE Conference on Super Computing; 2007. p. 1–12. https://doi.org/10.1145/1362622.1362627.
10. NEST Simulator. Date accessed: 2016. http://www.nest-initiative.org/.
11. Carlson KD, Nageswaran JM, Dutt N, Krichmar JL. An efficient automated parameter tuning framework for spiking neural networks, Neuroscience. 2014; 8(10):1–16.
12. Fountas Z, Shanahan M. GPU-based fast parameter optimization for phenomenological spiking neural models. International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland; 2015. p. 1–8. https://doi.org/10.1109/IJCNN.2015.7280668.
13. Izhikevich EM. Which model to use for cortical spiking neurons? IEEE Transaction Neural Networking. 2004; 15(5):1063–70. https://doi.org/10.1109/TNN.2004.832719. PMid: 15484883.
14. Hodgkin AL, Huxley AF. A quantitative description of membrane current and its application to conduction and excitation in nerve, Journal of Physiology. 1952; 117(4):500–44. https://doi.org/10.1113/jphysiol.1952.sp004764. PMid: 12991237, PMCid: PMC1392413.
15. Izhikevich EM. Simple model of spiking neurons, IEEE Transaction Neural Networking. 2003; 14(6):1569–72.
16. Nageswaran JM, Dutt N, Krichmar JL, Nicolau A, Veidenbaum AV. A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors, Neural Networking. 2009; 22(5–6):79–800. https://doi.org/10.1016/j.neunet.2009.06.028.
17. Fidjeland A, Roesch E, Shanahan M, Luk W. NeMo: A platform for neural modeling of spiking neurons using GPUs. 20th IEEE International Conference on Application-specific

Systems, Architectures and Processors; 2009. https://doi.org/10.1109/ASAP.2009.24.
18. Garrido A, Carrillo RR, Luque NR, Ros E. Event and time driven hybrid simulation of spiking neural networks, Advances in Computational Intelligence. 2011; 6691:554–61. https://doi.org/10.1007/978-3-642-21501-8_69.
19. Goodman D, Brette R. The Brian simulator, Front in Neuroscience. 2009; 3(2):192–97. https://doi.org/10.3389/neuro.01.026.2009. PMid: 20011141, PMCid: PMC2751620.
20. NEURON Simulator. http://www.neuron.yale.edu/neuron. Date accessed: 20/08/2018.
21. Brian Simulator. http://briansimulator.org/. Date accessed: 10/06/2018.
22. Mvaspike Simulator. http://mvaspike.gforge.inria.fr/. Date accessed: 04/04/2013.
23. NeMo Simulator. http://nemosim.sourceforge.net/. Date accessed: 10/07/2018.
24. GeNN Simulator. http:genn-team.github.io/genn/. Date accessed: 15/07/2018.