Neuromorphic Models on a GPGPU Cluster

Bing Han and Tarek M. Taha, Member, IEEE

Abstract: There is currently a strong push in the research community to develop biological scale implementations of neuron based vision models. Systems at this scale are computationally demanding and have generally utilized more accurate neuron models, such as the Izhikevich and Hodgkin-Huxley models, in favor of the integrate and fire model. This paper examines the feasibility of using a cluster of NVIDIA General Purpose Graphics Processing Units (GPGPUs) for accelerating a spiking neural network based character recognition network based on the Izhikevich and Hodgkin-Huxley models to enable such large scale systems. We utilized a 32 node cluster at NCSA containing an NVIDIA Tesla S1070 GPGPU on each node. Based on a thorough review of the literature, this is the first study examining the use of a cluster of GPGPUs for accelerating neuromorphic models. Our results show that the GPGPU can provide speedups of 24.6 and 177.0 times over a dual core 2.4 GHz AMD Opteron processor for the Izhikevich and Hodgkin-Huxley models respectively. Additionally, the MPI implementations of the models scaled almost linearly, with 16 GPGPUs providing throughputs of 14.1 and 15.9 times that of a single GPGPU for the Izhikevich and Hodgkin-Huxley models respectively. This indicates that clusters of GPGPUs are well suited for this application domain.

I. Introduction

At present there is a strong interest in the research community to model biological scale image recognition systems such as the visual cortex of a rat or a human. Most small scale neural network image recognition systems generally utilize the integrate and fire spiking neuron model. Izhikevich points out however that this model is one of the least biologically accurate models. He states that the integrate and fire model "cannot exhibit even the most fundamental properties of cortical spiking neurons, and for this reason it should be avoided by all means" [15]. Izhikevich compares a set of 11 spiking neuron models [15] in terms of their biological accuracy and computational load. He shows that the five most biologically accurate models are (in order of biological accuracy) the: 1) Hodgkin-Huxley, 2) Izhikevich, 3) Wilson, 4) Hindmarsh-Rose, and 5) Morris-Lecar models. Of these the Hodgkin-Huxley model is the most compute intensive, while the Izhikevich model is the most computationally efficient.

As a result, current studies of large scale models of the cortex are generally using either the Izhikevich [1] or the Hodgkin-Huxley [18] spiking neuron models. One of the main problems with using the Hodgkin-Huxley model for large scale implementations is that it is computationally intensive – each neuron update requires about 1200 flops [15]. The recently published Izhikevich model is attractive for large scale implementations as it is close to the Hodgkin-Huxley model in biological accuracy, but is similar to the integrate and fire model in computational intensity (13 flops).

Given that the visual cortex contains 10^9 neurons, biological scale vision systems based on spiking neural networks would require high performance computation capabilities. GPGPUs have attracted significant attention recently for their low cost and high performance capabilities. These architectures are well suited for applications with high levels of data parallelism, such as biological scale cortical vision models.

In this paper we examine the performance of a spiking neural network based character recognition model on a 32 node GPGPU cluster at the National Center for Supercomputing Applications (NCSA) [24]. MPI implementations of the model were developed to run across multiple GPGPUs on the cluster. The recognition phase of two versions of this model, based on the Izhikevich and Hodgkin-Huxley spiking neuron models, are examined. The model utilizes pulse coding to mimic the high speed recognition taking place in the mammalian brain [7] and spike timing dependent plasticity (STDP) for training [6]. It was trained to recognize a set of 48 24×24 pixel images of characters scaled up to 12288×12288 pixels and was able to recognize noisy versions of these images. The networks tested had over 150 million neurons implemented. An initial version of this model was presented in [4].

The GPGPU examined in this study utilizes the Compute Unified Device Architecture (CUDA) from NVIDIA. This architecture allows code compatibility between different generations of NVIDIA GPGPUs. We compare the performance of this GPGPU against a parallel implementation of the models on a 2.4 GHz AMD Opteron 2216 processor. On both platforms, the full parallelism

This work was supported by an NSF CAREER Award and grants from the US Air Force. This work was also supported in part by a grant of computer time from the National Center for Supercomputing Applications.

H. Bing is with the Electrical and Computer Engineering Department at University of Dayton, Dayton, OH 45469, USA (e-mail: hanbingz@notes.udayton.edu)

T. M. Taha is with the Electrical and Computer Engineering Department at University of Dayton, Dayton, OH 45469, USA (phone: 937-229-3119; fax: 937-229-4529; e-mail: tarek.taha@notes.udayton.edu).

offered by the architectures were utilized (both data and thread level parallelism extensions were used in the code). Our results indicate that the GPGPU can provide speedups of 24.6 and 177.0 times over the CPU for the Izhikevich and Hodgkin-Huxley models respectively. The MPI implementations of the models scaled almost linearly, with 16 GPGPUs providing a throughput of 14.1 and 15.9 times that of a single GPGPU for the Izhikevich and Hodgkin-Huxley models respectively. Based on a thorough review of the literature, this is the first study examining a cluster of GPGPUs for the acceleration of neuromorphic models. Our results indicate that these models are well suited to clusters of GPGPUs.

Section II of this paper discusses background material including a brief introduction to the two spiking network models considered for this study and a discussion of related work. Section III presents the character recognition model and examines its mapping to the GPGPU platform. Sections IV and V describe the experimental setup and results of the model performance, while section VI concludes the paper.

II. Background

A. Hodgkin-Huxley Model

The Hodgkin-Huxley model [12] is considered to be one of the most biologically accurate spiking neuron models. It consists of four differential equations (eq. 1-4) and a large number of parameters. The differential equations describe the neuron membrane potential, activation of Na and K currents, and inactivation of Na currents. The model can exhibit almost all types of neuronal behavior if its parameters are properly tuned. This model is very important to the study of neuronal behavior and dynamics as its parameters are biophysically meaningful and measurable. A time step of 0.01 ms was utilized to update the four differential equations as this is the most commonly used value. Fig. 1 shows the spikes produced with this model.

$$\frac{dv}{dt} = (\frac{1}{C})\{I - g_K n^4 (V - E_K) - g_{Na} m^3 h (V - E_{Na}) - g_L (V - E_L)\} \quad (1)$$

$$\frac{dn}{dt} = (n_\infty(V) - n) / \tau_n(V) \quad (2)$$

$$\frac{dm}{dt} = (m_\infty(V) - m) / \tau_m(V) \quad (3)$$

$$\frac{dh}{dt} = (h_\infty(V) - h) / \tau_h(V) \quad (4)$$

B. Izhikevich Model

Izhikevich proposed a new spiking neuron model in 2003 [14] that is based on only two differential equations (eq. 5-6) and four parameters. The model requires significantly fewer computations than the Hodgkin-Huxley model (13 flops as opposed to 1200 flops), but can still reproduce almost all types of neuron responses that are seen in biological experiments. A time step of 1ms was utilized (as was done by Izhikevich in [14]). Fig. 2 shows the spikes produced with this model.

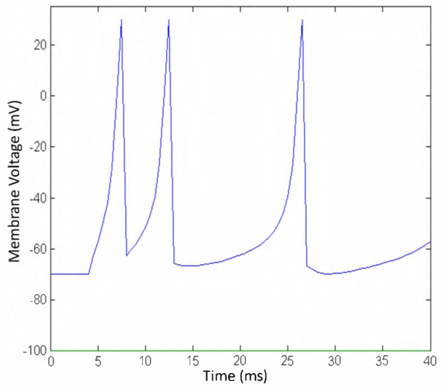


Fig. 1. Spikes produced with the Hodgkin-Huxley model (mV vs ms).

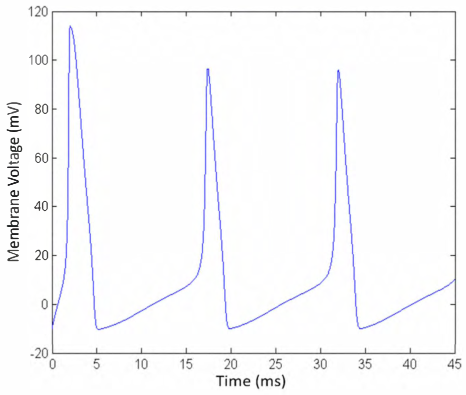


Fig. 2. Spikes produced with the Izhikevich model (mV vs ms).

$$\frac{dV}{dt} = 0.04V^2 + 5V + 140 - u + I \quad (5)$$

$$\frac{du}{dt} = a(bV - u) \quad (6)$$

$$\text{if } V \geq 30 \text{ mV, then } \begin{cases} V \leftarrow c \\ u \leftarrow u + d \end{cases}$$

C. GPGPU Architecture

NVIDIA's CUDA GPGPU architecture consists of a set of scalar processors (SPs) operating in parallel. Eight scalar processors are organized into a streaming multiprocessor (SM), with several SMs on a GPGPU (see Fig. 3). Each SM has an internal shared memory along with caches (a constant and a texture cache), a multi-threaded instruction (MTI) unit, and special functional units. The SMs share a global memory. A large number of threads are typically assigned to each SM. The MTI can switch threads frequently to hide longer global memory accesses from any of the threads. Both integer and floating point data formats are supported in the SPs.

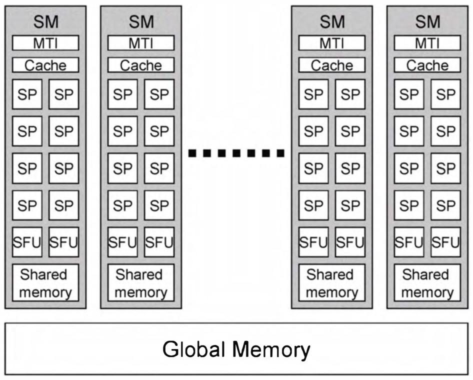


Fig. 3. A simplified CUDA GPGPU architecture. SP=Scalar Processor, SM=Streaming Multiprocessor, SFU=Special Functional Unit, and MTI=Multi-Threaded Instruction Unit.

In this study, a 32 node GPGPU cluster at the NCSA was utilized. Each node in the cluster contained a single NVIDIA Tesla S1070 GPGPU [10]. The Tesla S1070 GPGPU has four 1.3GHz T10 GPGPUs, each with access to 4GB of DDR3 memory (totaling 16 GB). Each T10 GPGPU has 240 processing cores. The system is capable of 3.73 GFLOPs and has a combined memory bandwidth of 408 GB/s. The nodes also contained two 2.4 GHz AMD Opteron 2216 processors.

D. Related work

Several groups have studied image recognition using spiking neural networks. In general, these studies utilize the integrate and fire model. Thorpe developed SpikeNET [7], a large spiking neural network simulation software. The system can be used for several image recognition applications including identification of faces, fingerprints, and video images. Johansson and Lansner developed a large cluster based spiking network simulator of a rodent sized cortex [16]. They tested a small scale version of the system to identify 128×128 pixel images. Baig [2] developed a temporal spiking network model based on integrate and fire neurons and applied them to identify online cursive handwritten characters. Gupta and Long [9] investigated the application of spiking networks for the recognition of simple characters. Other applications of spiking networks include instructing robots in navigation and grasping tasks [20], recognizing temporal sequences [5][13], and the robotic modeling of mouse whiskers [17].

At present several groups are developing biological scale implementations of spiking networks, but are generally not examining the applications of these systems (primarily as they are modeling large scale neuronal dynamics seen in the brain). The Swiss institution EPFL and IBM are developing a highly biologically accurate brain simulation [18] at the subneuron level. They have utilized the Hodgkin-Huxley and the Wilfred Rall [21] models to simulate up to 100,000 neurons on an IBM BlueGene/L supercomputer. At the IBM Almaden Research Center, Ananthanarayanan and Modha [1] utilized the Izhikevich spiking neuron model to simulate 55 million randomly connected neurons (equivalent to a rat-scale cortical model) on a 32,768 processor IBM BlueGene/L supercomputer. Johansson et. al. simulated a randomly connected model of 22 million neurons and 11 billion synapses using an 8,192 processors IBM BlueGene/L supercomputer [8].

A few studies have recently examined the performance of spiking neural network models on GPGPUs. Tiesel and Maida [23] evaluated a grid of 400x400 integrate and fire neurons on an NVIDIA 9800 GX2 GPGPU. In their single layer network, each neuron was connected to its four closest neighbors. This network was not trained to recognize any images. They compared a serial implementation on an Intel Core 2 Quad 2.4 GHz processor to a parallel implementation on an NVIDIA 9800 GX2 GPGPU and achieved a speedup of 14.8 times. Nageswarana et al. [19] examine the implementation of 100K randomly connected neurons based on the more biologically accurate Izhikevich model. They compared a serial implementation of the model on a 2.13 GHz Intel Core 2 Duo against a parallel implementation on an NVIDIA GTX280 GPGPU and achieved a speedup of about 24 times.

III. Network Design And Mapping To GPGPU

A two layer network structure was utilized in this study with the first layer acting as input neurons and the second layer as output neurons (see Fig. 4). Input images were presented to the first layer of neurons, with each image pixel corresponding to a separate input neuron. Thus the number of neurons in the first layer was equal to the number of pixels in the input image. Binary inputs were utilized in this study. If a pixel was “on”, a constant current was supplied to the input, while no current was supplied if a pixel was “off.” The number of output neurons was equal to the number of training images. Each input neuron was connected to all the output neurons.

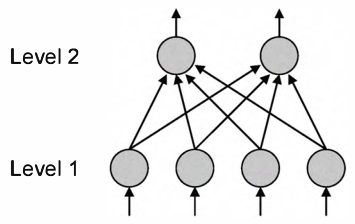


Fig. 4. Network used for testing spiking models.

In the training process, images from the training set were presented sequentially to the input neurons. The weight matrix of each output neuron was updated using the STDP rule each cycle. In the testing phase, an input image is presented to the input neurons and after a certain number of cycles, one output neuron fires, thus identifying the input image. During each cycle, the level one neurons are first evaluated based on the input image and a firing vector is updated to indicate which of the level one neurons fired that cycle. In the same cycle, the firing vector generated in the previous cycle is used to calculate the input current to each level two neuron. The level two neuron membrane potentials are then calculated based on their input current. A level two neuron's overall input current is the sum of all the individual currents received from the level one neurons connected to it. This input current for a level two neuron is given by eq. 7 below:

$$I_j = \sum_i w(i,j)f(i) \tag{7}$$

where

w is a weight matrix in which *w(i,j)* is the input weight from level one neuron *i* to level two neuron *j*.

f is a firing vector in which *f(i)* is 0 if level one neuron *i* does not fire, and is 1 is the neuron does fire.

Two versions of the model were developed where the main difference was the equations utilized to update the potential of the neurons. In one case, the Hodgkin-Huxley equations presented in section 2.1 were used, while in the second case, the Izhikevich equations in section 2.2 were used. The parameters utilized in each case are specified in Appendix I.

In order to achieve high performance from the GPGPU, all the processing cores need to be utilized at the same time. Each streaming multiprocessor can execute a group of threads concurrently (call a warp) on its scalar processors and can easily switch between warps [22]. To hide long memory latencies, it is recommended that hundreds to thousands of threads be assign to the GPGPU. In case one warp is waiting on a memory access, the streaming multiprocessor can easily switch to other warps to hide this latency.

The network utilized in this study has high amounts of parallelism from all the neurons in each layer. Since each neuron is evaluated using a separate thread, a very large number of threads are available for execution. Fig. 5 shows a flowchart of the main components of the algorithm. Three kernels are executed on the GPGPU to update the level 1 and level 2 neurons and their corresponding synaptic currents (level 2 neuron input weights). The check of whether a level 2 neuron fired (and thus categorized an input image) is carried out on the CPU as this is a serial operation. All the neuron parameters are implemented using floating point variables on the GPGPU.

The summation of the neuron weights described in eq. 7 is carried out for one level 2 neuron at a time (kernel 2 in Fig. 5). For a particular level 2 neuron, *j*, all the elements in the masked weight vector *w(i,j)f(i)* have to be added to each other to generate a final input current. This is done by splitting *w(i,j)f(i)* into multiple sub-vectors, with the sum of the all the sub-vectors accumulated in GPGPU shared memory. When all the sub-vectors have been added, parallel reduction [11] is used to generate an input current by adding all the elements of the accumulated vector.

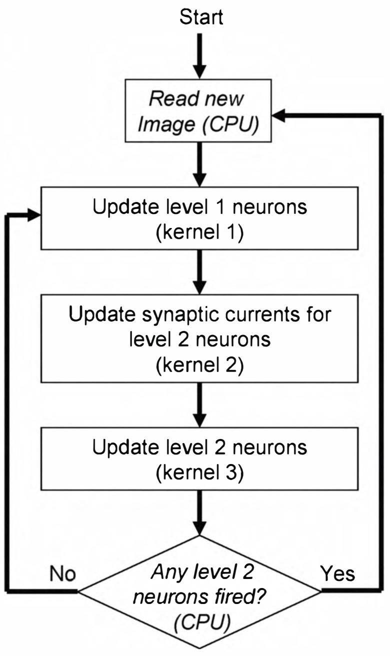

Start
Read new Image (CPU)
Update level 1 neurons (kernel 1)
Update synaptic currents for level 2 neurons (kernel 2)
Update level 2 neurons (kernel 3)
Any level 2 neurons fired? (CPU)
No
Yes

Fig. 5. Algorithm flowchart.

IV. Experimental Setup

The models were run on a 32 node cluster [24]. Each node was a Fedora 10 Linux based PC with two 2.4 GHz dual core AMD Opteron 2216 processor, 8 GB of DRAM, and a NVIDIA Tesla S1070 GPGPU. The model was parallelized using MPI. Multithreaded C implementations of the models (presented initially in [4]) were run on one of the AMD Opteron processors. The code was optimized to use the SSE3 SIMD extensions and utilized the POSIX threads library to run on both cores of the processor. The programs were compiled with -O3 optimizations using gcc. The GPGPU implementation of the model utilized CUDA 2.3 nvcc.

Three networks with varying input image sizes were developed to examine the two spiking neural network models on a single GPGPU (based on [3]). The overall network structure was kept similar to the design shown in

Fig. 4, with two layers of nodes per network. Each of the level 2 neurons was connected to all of the neurons from level 1. Table 1 shows the number of neurons per layer for all the network sizes implemented. In this study all the spiking networks of different sizes were trained to recognize the same number of images. A set of 48 24×24 pixel images were generated initially and were then scaled linearly to the different network input sizes. Fig. 6 shows the training images used. The images represented the 26 upper case letters (A-Z), 10 numerals (0-9), 8 Greek letters, and 4 symbols.

Table 1. Spiking Network Configurations Evaluated

Input image size	768×768	1536×1536	3072×3072
Total neurons	589,872	2,359,344	9,437,232
Level 2 neurons	48	48	48
Level 1 neurons	589,824	2,359,296	9,437,184

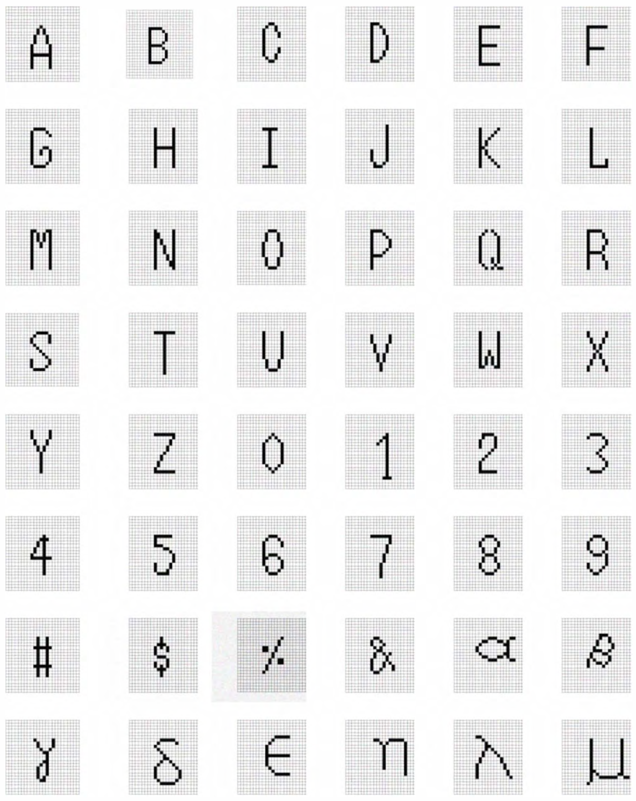


Fig. 6. Training images

V. Results

Both the CPU and the GPGPU versions of the model were able to recognize all the training and an additional set of noisy versions of these images accurately (Fig. 6 and 7), except for noisy image 6. The speedups of the GPGPU implementation of the models on one Tesla S1070 over the AMD processor versions are shown in Table 2. The neurons/s throughput on one Tesla S1070 over the AMD processor is shown in Fig. 8. The Izhikevich model provided a speedup of up to 24.6 times over the CPU, while the Hodgkin-Huxley model provided a speedup of about 177.0 times. The Izhikevich model provided a lower neurons/s throughput as it is less computationally demanding (further discussion below).

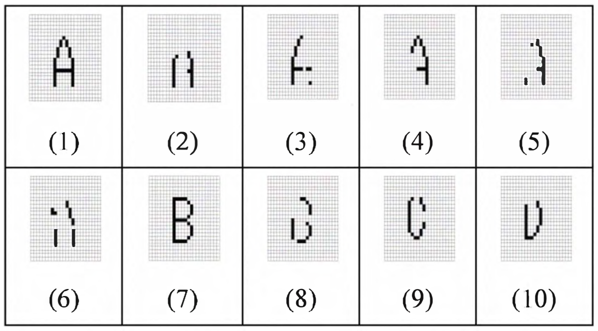
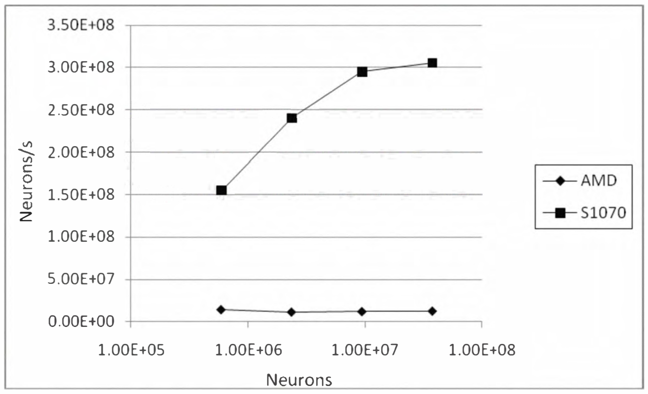


Fig. 7. Additional test images.

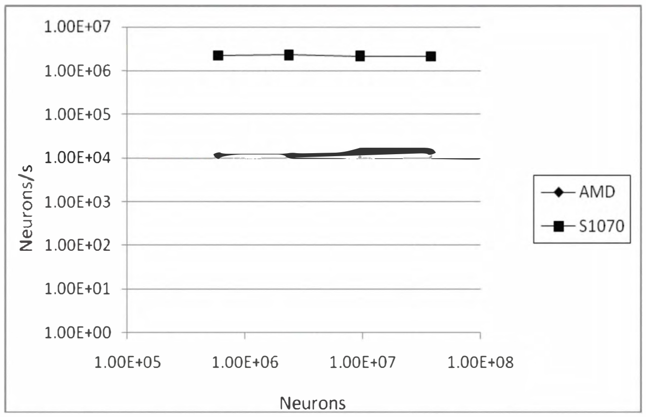
Table 2. Speedup of GPGPU implementations (one Tesla S1070) over CPU implementation (one dual core AMD Opteron 2216 processor) for the training images.

Image size	Speedup	
	Izhikevich	Hodgkin-Huxley
768×768	20.1	199.4
1536×1536	23.7	190.5
3072×3072	24.6	177.0

The raw runtime breakdown (image size of 768×768) of the GPGPU versions of models is shown in Table 3, while a percentage breakdown is shown in Fig. 9. Both level 1 and level 2 neuron computations (kernels 1 and 3 in Fig. 5), along with the level 2 neuron input weight computations (kernel 2), take place on the GPGPU. The rest of the runtime is for: 1) main memory access, including data transfer between the host (CPU) memory and the device (GPGPU) memory, and 2) CPU operations, including initialization and checking all 48 level-two neurons' voltages to find out which one fired (to determine the final recognition result). Given that the number of level 2 neurons does not vary with the input image size, the computation time for the level 2 neurons remains constant for a given model. The runtime for the remaining components do vary with the number of level 1 neurons, and hence the input image size. It is seen in Fig. 9 that the Izhikevich model has a higher percentage of time spent on CPU and memory access, compared to the Hodgkin-Huxley model. This leads to a higher overhead ratio in the Izhikevich model, and thus a lower speedup.



(a)



(b)

Fig. 8. Neurons/s throughput of GPGPU and CPU processors for the (a) the Izhikevich and (b) the Hodgkin-Huxley model.

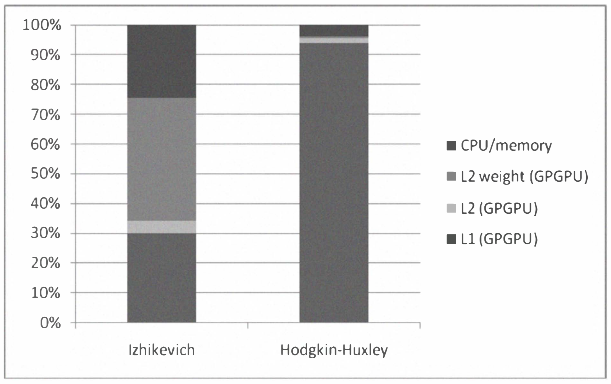


Fig. 9. Timing breakdown by percentage of overall runtime for the two models.

The bytes of data required per flop for the models and the flops achieved in the experiments are listed in Table 4. It also lists the peak capability of the NVIDIA Tesla S1070 utilized (calculated from specification in [10]). The results show that the memory bandwidth requirement for the Izhikevich model is about 19 times the peak capability of the GPGPU, while bandwidth requirement for the Hodgkin-Huxley model is 3.5 times the capability of the GPGPU. This also leads to the speedup of the Izhikevich model to be lower than the Hodgkin-Huxley model (over the AMD processor).

The performance of a single GPGPU is limited by both the number of neurons being processed concurrently and the on-board memory space. To achieve higher performance, a cluster of GPGPUs is utilized for the high performance implementation of larger networks such as 6144×6144, 9216×9216, and 12288×12288. In this paper, an MPI based cluster implementation of the models was run with up to 16 GPGPUs. Both the memory space and the total amount of parallelization increase linearly with the number of nodes utilized in this parallel computation. To make full use of each node, larger images are segmented into multiple sub-images, with the individual sub-images processed concurrently using multiple GPGPUs. The size of each sub-image is equal to the largest size feasible in a single GPGPU. As shown in Table 5 and Fig. 10, a 6144×6144 image is broken into four 3072×3072 segments for both the Izhikevich and Hodgkin-Huxley models. Similarly, image size 9216×9216 is broken into nine 3072×3072 sub-images and processed by nine GPGPUs. The size of each sub image processed in each GPGPU is consistently 3072×3072. Therefore the speedup for larger images is slightly less than the maximum seen for each of the models in the previous results because of task synchronizations and memory bandwidth limitations between host (CPU) memory and device (GPGPU) memory.

Table 3. Timing breakdown (in ms) of the two models.

	Izhikevich	**Hodgkin-Huxley**
L1 neurons (GPGPU)	2.94	958.23
L2 neurons (GPGPU)	0.42	18.24
L2 weights (GPGPU)	4.05	4.39
CPU computation and memory access	2.41	39.76
Total	**9.82**	**1020.62**

Table 4. Model requirements and system peak performance

	Models requirement		**NVIDIA Tesla S1070 peak**
	Izhikevich	**Hodgkin -Huxley**	
Bytes/FLOP	2.11	0.38	0.11

Table 5. Run time (in ms) of MPI based Network Configurations.

Image sizes	**GPGPUs**	**Izhikevich**	**Hodgkin-Huxley**
3072×3072	1	124.86	17838.37
6144×6144	4	128.36	17843.92
9216×9216	9	131.19	17847.06
12288×12288	16	141.74	17910.30

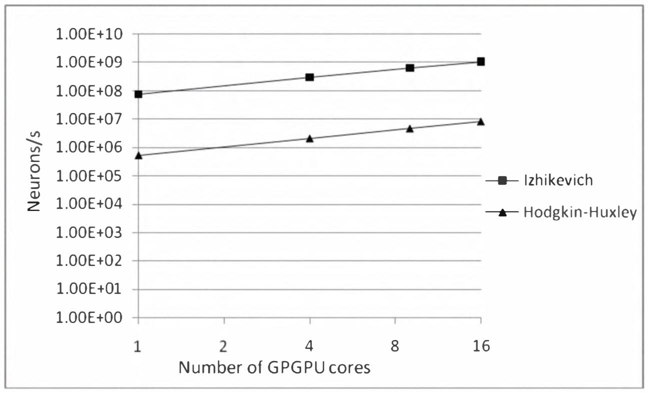


Fig. 10. Neurons per second throughput of the Izhikevich and Hodgkin-Huxley models with variation in the number of GPGPUs utilized through MPI.

The results indicate the 16 GPGPU version had a throughput increase of about 14.1 and 15.9 times over the single GPGPU version for the Izhikevich and Hodgkin-Huxley models respectively. The MPI communication takes place over InfiniBand connections. As shown in Fig. 11, the communication time increased with the number of nodes, but this time was a very small fraction of the overall runtime.

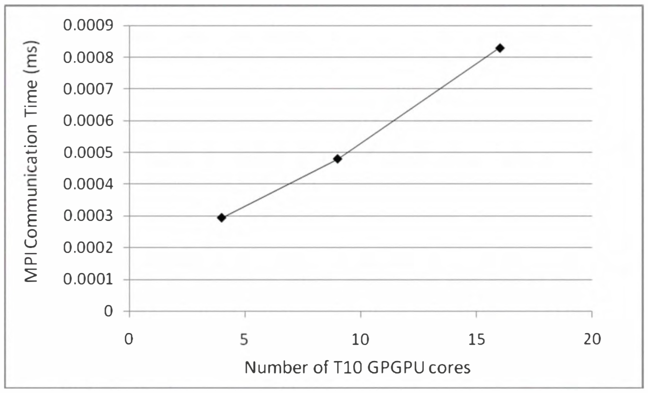


Fig. 11. MPI communication time.

VI. Conclusion

There is a strong effort in the research community recently to develop biological scale implementations of neuron based vision systems. Although the integrate and fire model is typically used in spiking neural network based engineering applications, this model is considered to be significantly inferior to more biologically accurate models. As a result, large scale biological models have tended to utilize the more accurate neuron models, such as the Izhikevich and Hodgkin-Huxley neuron models.

Large scale implementations of biologically accurate neuron models for image recognition applications are computationally demanding, requiring large computational clusters. Graphics processing units provide a low cost, high performance platform for data parallel applications. Given that large neural models have significant parallelism, this paper examined the feasibility of using such hardware for these models. Two versions of an image recognition model (based on the Izhikevich and Hodgkin-Huxley models) were implemented on a cluster of 32 NVIDIA Tesla S1070 based GPGPUs and their performance compared to an AMD Opteron 2216 processor.

Models with 147,456 to 9,437,184 neurons were tested by non-MPI version implementation (based on [3]). The Tesla S1070 GPGPU provided speedups of up to 24.6 and 177.0 times for the Izhikevich and Hodgkin-Huxley models over corresponding implementations on the AMD processor. All available forms of parallelism were utilized on both the GPGPU and the CPU processors. The Izhikevich model provided a lower speedup as it is less computationally demanding (and thus has a higher overhead ratio). These results indicate that GPGPUs can provide a high performance platform for the implementation of biological spiking neuron based pattern recognition models. By using GPGPU clusters, larger networks containing 37,748,784, 84,934,704 and 150,994,992 neurons were implemented. Since the MPI communication time is a very small fraction of the total run time, the neurons per second throughput increases almost linearly with the number of nodes being used.

As future work, we plan to explore acceleration of other spiking neural models and more biologically accurate networks on GPGPUs and GPGPU cluster. Additionally, we plan to examine applications of the GPGPUs and GPGPU cluster based spiking neural network models to other areas, such as biometrics.

Appendix I: Neuron Model Parameters

Hodgkin-Huxley: g_K=36 seimen, g_{Na}=120 seimen, g_L=0.3 seimen, E_K=-12 mV, E_{Na}=115 mV, E_L=10.613 mV, V=-10 mV, V_K=0 mV, V_{Na}=0 mV, V_L=1 mV, time step=0.01 ms.

Izhikevich: Excitatory neurons: a=0.02, b=0.2, c= -55, d=4; Inhibitory neurons: a=0.06, b=0.22, c= -65, d=2, time step=1 ms.

References

[1] R. Ananthanarayanan and D. Modha, “Anatomy of a Cortical Simulator,” Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (Supercomputing 2007), Reno, NV, November 2007.

[2] A. R. Baig, “Spatial-temporal artificial neurons applied to online cursive handwritten character recognition,” in European Symposium on Artificial Neural Networks, April 2004, paper Bruges (Belgium), pp. 561–566.

[3] Bing Han and Tarek M. Taha, "Acceleration of spiking neural network based pattern recognition on NVIDIA graphics processors," Appl. Opt. 49, B83-B91 (2010)

[4] M. A. Bhuiyan, R. Jalasutram, and T. M. Taha, “Character Recognition with Two Spiking Neural Network Models on Multicore Architectures,” in IEEE Symposium on Computational

Intelligence for Multimedia Signal and Vision Processing, April 2009.

[5] D. V. Buonomano and M. M. Merzenich, “A neural network model of temporal code generation and position invariant pattern recognition,” Neural Computation, vol. 11, pp. 103–116, 1999.

[6] Y. Dan and M. Poo, “Spike time dependent plasticity of neural circuits,” Neuron, vol. 44, pp. 23–30, 2004.

[7] A. Delorme and S. J. Thorpe, “SpikeNET: an event-driven simulation package for modelling large networks of spiking neurons,” Network-computation in neural systems, 14(4), 613–627, Nov. 2003.

[8] M. Djurfeldt, M. Lundqvist, C. Johansson, M. Rehn, O. Ekeberg, and A. Lansner, “Brain-scale simulation of the neocortex on the IBM Blue Gene/L supercomputer,” IBM Journal of Research and Development, 52(1-2), 31–41, Jan.-Mar. 2008.

[9] A. Gupta, L. Long, “Character Recognition using Spiking Neural Networks,” International Joint Conference on Neural Networks, Aug. 2007.

[10] Tesla S1070 Specifications, http://www.nvidia.com/object/product_tesla_s1070_us.html

[11] M. Harris, “Optimizing Parallel Reduction in CUDA,”

http://developer.download.nvidia.com/compute/cuda/1_1/Website/projects/reduction/doc/reduction.pdf

[12] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and application to conduction and excitation in nerve,” Journal of Physiology, 117, 500–544, 1952.

[13] T. Ichishita, R. Fujii, “Performance evaluation of a temporal sequence learning spiking neural network,” Proceedings of the 7th IEEE International Conference on Computer and Information Technology, Oct. 2007.

[14] E. M. Izhikevich, “Simple Model of Spiking Neurons,” IEEE Transactions on Neural Networks, vol. 14, no. 6, November, 2003, pp. 1569-1572.

[15] E.Izhikevich, “Which Model to Use for Cortical Spiking Neurons?” IEEE Transactions on Neural Networks, 15(5), 1063-1070, 2004.

[16] C. Johansson and A. Lansner, “Towards Cortex Sized Artificial Neural Systems,” Neural Networks, 20(1), 48–61, Jan. 2007.

[17] K-Team, Inc. Available online: http://www.k-team.com/

[18] H. Markram, “The Blue Brain Project,” Nature Reviews Neuroscience, 7, 153–160, 2006.

[19] J. M. Nageswarana, N. Dutt, J. L. Krichmar, A. Nicolau, A. V. Veidenbauma, “A Configurable Simulation Environment for the Efficient Simulation of Large-Scale Spiking Neural Networks on Graphics Processors,” Neural Networks, vol. 22, pp. 791-800, 2009.

[20] C. Panchev and S. Wermter “Temporal sequence detection with spiking neurons: towards recognizing robot language instructions,” Connect. Sci., 18(1): 1-22, 2006.

[21] W. Rall, “Branching dendritic trees and motoneuron membrane resistivity,” Experimental Neurology, 1, 503–532, 1959.

[22] N. Satish, M. Harris, and M. Garland, "Designing efficient sorting algorithms for manycore GPUs," in IEEE International Symposium on Parallel & Distributed Processing, 2009.

[23] J.-P. Tiesel, A. S. Maida, "Using parallel GPU architecture for simulation of planar I/F networks," in International Joint Conference on Neural Networks, 2009.

[24] V. Kindratenko, J. Enos, G. Shi, M. Showerman, G. Arnold, J. Stone, J. Phillips, W. Hwu, GPU Clusters for High-Performance Computing, In Proc. Workshop on Parallel Programming on Accelerator Clusters - PPAC'09, 2009.