In Proceedings of the 9th Artificial General Intelligence Conference, New York, July, 2016.

For more info see: http://www.personal.psu.edu/lnl

Toward Human-Level Massively-Parallel Neural Networks with Hodgkin-Huxley Neurons

Lyle N. Long
The Pennsylvania State University, University Park, PA USA
lnl@psu.edu

Abstract. This paper describes neural network algorithms and software that scale up to massively parallel computers. The neuron model used is the best available at this time, the Hodgkin-Huxley equations. Most massively parallel simulations use very simplified neuron models, which cannot accurately simulate biological neurons and the wide variety of neuron types. Using C++ and MPI we can scale these networks to human-level sizes. Computers such as the Chinese TianHe computer are capable of human level neural networks.

Keywords: neural networks, neurons, parallel, Hodgkin-Huxley, MPI

1 INTRODUCTION

Artificial intelligence began in roughly 1956 at a conference at Dartmouth University. The participants, and many researchers after them, were clearly overly optimistic. As with many new technologies, the technology was oversold for decades.

Computer processing power, however, has been doubling every two years thanks to Moore's law. In the 1950's one of the main computers was the IBM 701, which could do 16,000 adds/subtracts per second, or 2,000 multiples/divides per second. This is roughly a trillion times smaller than the human brain. As shown in Figure 1, it is more on par with the C. Elegan worm, which is about 1 mm long and has 302 neurons and 6393 synapses [1].

Over a wide range of biological creatures, it is estimated [2,3] that the number of synapses in biological systems can be modeled via:

$$Synapses = 3.7\ Neurons^{1.32} \tag{1}$$

A cockroach has about a million neurons, and using the above formula has about 300 million synapses. A rough estimate is that each synapse can store 1-8 bits and can perform roughly 1-2 operations per second. Thus from these crude estimates the IBM 701 had performance about 10,000 times worse than a cockroach neural system. It is amazing that the term "artificial intelligence" (AI) was coined during this era of horribly low-powered computers. Not until about 1975 did we have a computer on the order of a cockroach, the Cray 1, which had a speed of roughly 160 megaflops. It is not surprising that AI by this time was not taken seriously except in science fiction.

About 20 years later there was the ASCI Red computer with 9298 processors with a terabyte of memory and a speed of 1 teraflop. If this could have been harnessed for

modeling a brain, it would have been on the order of a rat, which has about 200 million neurons.

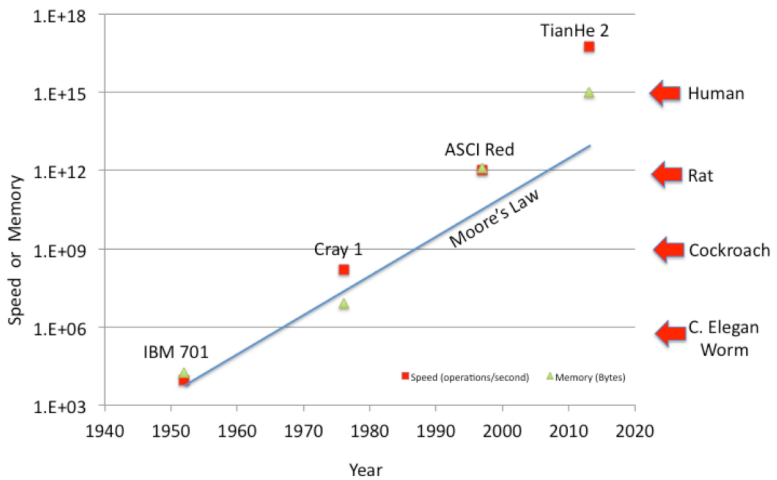

Figure 1. Computers and biological systems speed and memory.

The five largest parallel computers that exist today (which aren't classified) are shown in Table 1 [4]. The TianHe-2 computer in China has more than 3 million processor cores, 1 petabyte of memory, and a peak speed of 55 petaflops.

Table 1. Top five computers in the world, (www.top500.org, Nov. 2015).

Rank	**Name**	**Processor Cores**	**Peak Speed (PetaFlops) (10^{15})**	**Memory (PetaBytes) (10^{15})**	**Power Required (MWatts)**
1	TianHe-2 (China)	3,120,000	55	1.0	17.8
2	Titan (USA-DOE)	560,640	27	0.7	8.2
3	Sequoia (USA-DOE)	1,572,864	20	1.6	7.9
4	K Computer (Japan)	705,024	11	1.4	12.7
5	Mira (USA-DOE)	786,432	10	0.8	3.9

Note that the data in Figure 1 do not follow Moore's law. The increasing numbers of processors makes the trend much faster than Moore's law. Instead of doubling every two years, supercomputer speed doubles about every 1.4 years. Over a 60 year period that leads to about 10,000 times more speed than Moore's law would predict.

The human brain has roughly 10^{11} neurons and 10^{15} synapses. Some estimate that the brain is capable of roughly 10^{14} -10^{15} operations per second, with memory storage of roughly

10^{14} - 10^{15} bytes. Thus the largest computers in the world are now on the same order of magnitude as the human brain in terms of speed and memory. We are very far, however, from replicating the efficiency of the human brain. It only requires about 20 watts and about 1200 cm^3, which is about a million times lower than the supercomputers.

Finally, 60 years after the first AI conference we have computers on the order of the performance of the human brain, even if they are a million times less efficient (in terms of power and space).

The main issues now are algorithms and network structure. We have excellent models of neurons, such as the Hodgkin-Huxley model, but we do not know how the human neurons are wired together, or how carefully we need to match brain architecture.

This paper is an attempt at using efficient and powerful algorithms, together with powerful supercomputers to simulate as many neurons and synapses as possible, and in a scalable manner. The goal is not to simulate the brain, but to develop an engineering system.

There are several computational neuroscience models of neural networks [5-8], but most of these aim for accurate neuroscience simulations. In the work presented here the goal is to perform engineering simulations of massive neural networks for possible applications to complex engineering systems such as cognitive robotics [9]. Riemann et al [10] used 12,000 neurons and 15 million synapses and used 4096 cpus in a Blue Gene P computer. Four seconds of real time took 3 hours of CPU time.

2 HODGKIN-HUXLEY NEURONS

There are numerous models for neurons, as described in [2]. Most of these are very simplified and approximate formulae. As we have shown in previous papers [2, 3], this is a mistaken approach for two reasons:

1. With modern algorithms and computers more accurate models cost almost no more computer time
2. There are typically many orders of magnitude more synapses than neurons in large networks, as shown by equation (1).

Thus accurate neuron models can be used and it is of paramount importance to store and compute the synapse operations extremely efficiently.

Also, it should be mentioned that the neural networks being discussed here are time-dependent spiking (or pulsed) networks. These are quite different than typical rate-based artificial neural networks often used in engineering applications.

In 1952 Hodgkin and Huxley [11] proposed a mathematical model for a neuron. It was used to account for the electric current flow through the surface membrane of a squid giant axon. The Hodgkin-Huxley model is used to explain the different spiking phenomena of a neuron after it is exposed to various current stimulations. In their paper, the effects of different ionic channels to the capacity and resistance of the membrane were incorporated into the model; and empirical curve-fittings were used to generate the

component functions for the equations. The Hodgkin-Huxley (HH) model is one of the most biological plausible models in computational neuroscience, and they won a Nobel prize for their research. Their model is a complicated nonlinear system of coupled ordinary differential equations (ODE) consisting of four equations describing the membrane potential, activation and inactivation of different ionic gating variables respectively.

For many years now researchers have stated that the Hodgkin-Huxley model was far too expensive to use due to its complexity. This is simply not the case, as we showed in [2]. In particular models such as Izhikevich's [12] are not recommended. As shown in [2], it is not as efficient as the author states, nor can it model many types of neurons. The H-H model does not require as much work as people think, and it *can* model many types of neurons. The HH equations are the following differential equations:

$$\frac{d\,u}{d\,t} = E - G\,u \qquad \frac{d\,m}{d\,t} = \propto_m - (\propto_m + \beta_m)\,m$$

$$\frac{d\,n}{d\,t} = \propto_n - (\propto_n + \beta_n)\,n \qquad \frac{d\,h}{d\,t} = \propto_h - (\propto_h + \propto \beta_h)\,h$$

where

$$G = g_{Na} m^3 h + g_K n^4 + g_L \qquad E = g_{Na} m^3 h\, E_{Na} + g_K n^4 E_K + g_L\, E_L + I$$

and where the coefficients and constants are defined as:

$$\alpha_n(u) = \frac{0.1 - 0.01u}{\exp(1 - 0.1u) - 1} \qquad \beta_n(u) = 0.125\exp(-\frac{u}{80})$$

$$\alpha_m(u) = \frac{2.5 - 0.1u}{\exp(2.5 - 0.1u) - 1} \qquad \beta_m(u) = 4\exp(-\frac{u}{18}) \qquad g_{Na} = 120\ ms/cm^2 \quad E_{Na} = 115\ mV$$

$$\alpha_h(u) = 0.07\exp(-\frac{u}{20}) \qquad \beta_h(u) = \frac{1}{\exp(3 - 0.1u) + 1} \qquad g_K = 36\ ms/cm^2 \quad E_K = -12\ mV$$

$$g_L = 0.3\ ms/cm^2 \quad E_L = 10.6\ mV$$

Here *u(t)* is the neuron membrane voltage, parameters g_{Na}, g_K, and g_L are used to model the channel conductances. The additional variables *h*, *m,* and *n* control the opening of the channels. The parameters E_{Na}, E_K, and E_L are the reversal potentials. The term *I* is the input current (from other neurons or some external source), and is typically a function of time.

The HH equations can be solved very efficiently using the exponential Euler method. For an equation of the form

$$\frac{df}{dt} = A - B\,f$$

(note that all four ODE's in the HH equations are of this form) the exponential Euler

method is implemented as

$$f^{n+1} = \left(f^n - \frac{A^n}{B^n} \right) e^{-B^n \Delta t} + \frac{A^n}{B^n}$$

For A and B constant, this is an exact formula. For our purposes we will assume the coefficients change very slowly and can be assumed constant over one time step. Iterations could also be easily performed if necessary, but they are usually not required. Using look-up tables for the coefficients is very effective, since the exponentials are expensive to compute.

3 PARALLEL SOFTWARE IMPLEMENTATION

The software used in these simulations was written in C++ and uses the Message Passing Interface [13]. C++ was used due to its wide acceptance, high performance, efficient memory usage, and powerful modern syntax. MPI was used since it is essentially the only possible approach for massively parallel computers.

One of the difficult aspects of using distributed memory computers, especially when there might be millions of processors, is how to distribute the problem across the processors. This is especially difficult for neural networks, since we have to simulate neurons and synapses and they are connected in very complicated networks.

In the approach used here, the neurons are evenly distributed across the processors using MPI in a single program multiple data (SPMD) approach. Each neuron also has a list of synapses that it is connected to, and each synapse has information on its post-synaptic neuron and its processor number. For the H-H model each neuron stores 19 floats, 4 integers, and a dynamic list of synapses. So the memory used per neuron is (23 + num_synapses) *4 bytes.

While biologically a synapse might store roughly a byte of data, in the computer program each synapse here requires 73 bits (or roughly 5 bytes). The weights are stored as char variables (1 byte), an integer is used to store the post-synaptic neuron number (4 bytes), an integer is used to store the processor on which the post-synaptic neuron exists (4 bytes), and 1 bit is used to store whether it is an input neuron or not. Using a 32-bit integer for the neuron addresses limits the number of neurons per MPI process to 2^{32} (about 4 billion, if they are unsigned ints), which is quite adequate. And using an integer to store processor number also means one could use roughly 4 billion processors. So any of the top five computers in the list above could store roughly as many of these synapses as the human brain (10^{14} - 10^{15}). The amount of memory required by the synapses could be reduced by using a short ints, but they can have maximum values of only 65,536.

Figure 3 shows the number of processors required for a wide range of neurons (and using equation 1 for number of synapses). Figure 4 shows how much memory is required per number of neurons. This shows on a computer such as the TianHe we have enough processors and memory to model human-level neural networks. A computer ten times

larger than the TianHe could model a neural network ten times larger than a human brain, and possibly lead to superintelligence [14].

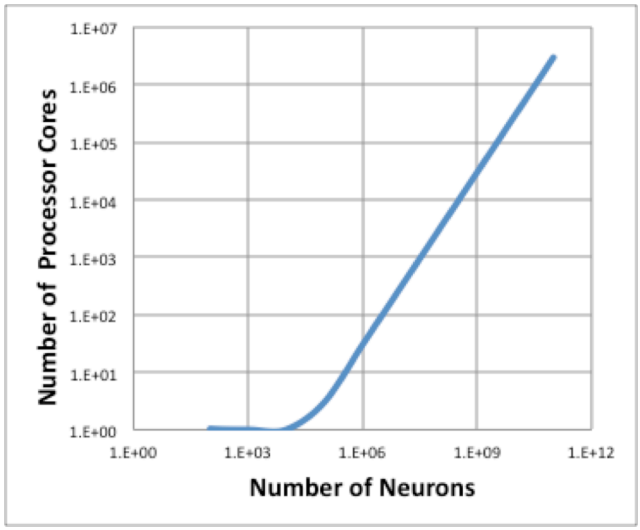

Figure 3. Processors required for a range of neurons.

The other major issue is computer time requirements. As shown in [2] the algorithm for the H-H neurons requires about 69 operations per time step using the exponential Euler method combined with lookup tables for the coefficients. This is only about a factor of two slower than the Izhikevich method, which cannot capture the physics properly or model a wide range of neuron types. A typical time step size for reasonable solutions is about 0.1 mSec. Each processor core of the TianHe-2 computer has a peak speed of about 10 billion operations per second. So for a billion neurons each time step would require about 7 seconds using just one processor (but the machine has 3 million). Also, one second of real time requires roughly 690,000 operations, but we are not interested in real-time neural computing.

Also, we need to consider the *communication* cost of the synapses. When a neuron fires, a pulse is sent to the connecting neurons, and this pulse is weighted by the value of the synapse weight. This can be accomplished with an add operation.process per synapse. So if we have a billion neurons and 1000 synapses per neuron, we'd have 10^{12} synapses. This means we'd have to do 10^{12} operations per time step. Using one processor of the TianHe-2 computer, this would take roughly 100 seconds, which is significantly more than the work required to march the neuron forward in time. As stated earlier, the synapses drive the problem, not the neurons.

The third major issue in using massively parallel computers is the inter-processor communication. Computers such as those shown in Table 1 are hybrid distributed-shared memory machines. Each node of the machine is a shared memory computer, and they are connected via a high-speed network to the other nodes. The networks are often Infiniband networks, or something similar. Communication speeds are often on the order of 100 gigabits/second, with minimum latencies on the order of 1-5 microseconds. A microsecond might sound like a very short period of time, but a 10 gigaflop processor could perform 10,000 operations during a microsecond. So if the processor is sitting idle waiting for data, the performance can be seriously affected. And there is no guarantee you will experience the minimum latency or the maximum bandwidth in practice. And neural networks can require an enormous amount of communication, especially if not done properly.

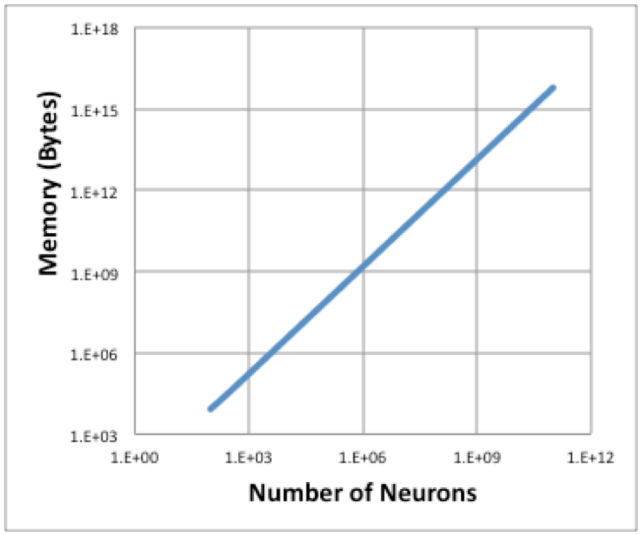

Figure 4. Memory required for a range of neurons.

For the example discussed earlier, with a billion neurons and a trillion synapses, every neuron is connected to 1000 other neurons. If every synapse sends its weight every time step, this would require 10^{12} bytes transmitted each time step. Whether this is feasible depends on the bi-section bandwidth of the supercomputer. Another way to look at this is that the TianHe has 16,000 nodes with 88 Gbytes/node. If each node had 50 billion synapses and had to transmit them each time step, it might take roughly an hour per time step (assuming 100 Mb/sec. connection). So synapse communications need to be handled very carefully to maintain performance.

Fortunately the above scenario is not required. While in a traditional artificial neural network (ANN) using backpropagation, all the synapse weights are involved each sweep

of the network, this is not true in spiking neural networks. In spiking neural networks, the synapse weight only needs to be communicated when the pre-synaptic neuron fires. And in biological systems typically only a few percent of the neurons are active at a time. And in addition, they typically only spike (at most) roughly every 20 time steps. So in effect we might only need to transmit about one in a thousand synapse weight data per step, if programmed properly. So instead of an hour, it might take seconds.

In the code developed here, when a neuron fires, it sends this information to every one of the post-synaptic neurons it is connected to. Some of these neurons might be on other processors, while some might be on the local processor. MPI-3 has many new features, one of which is one-sided communication. Instead of one processor executing a SEND command, and another processor execute a RECEIVE command, a processor can simply do an MPI_PUT and send the signal from one neuron to another, much as a biological neuron does.

The PUT and GET functions are very useful, but an even more appropriate function for sending neural signals is the MPI_ACCUMULATE function. This allows one to put a variable on another processor and have it add the value to the current value on that processor. This is exactly what we need here. For the MPI_PUT and MPI_ACCUMULATE functions it is also necessary to set up "windows," which set up the memory block that is to be shared.

The code here uses a completely unstructured or pointer-based approach. There is a Neuron class, a Synapse class, and a Network class. Each Neuron object has a dynamic list of Synapses, and each of these Synapses connects to one other Neuron. So a Synapse has to store two integers and a byte (the weight). Thus any type of connectivity can be modeled, including, all-to-all networks, convolution networks, recursive networks, or deep networks.

The unstructured or pointer-based approach to network connectivity was chosen for another reason as well. It makes the network very easy to modify (i.e. to add/remove neurons and synapses), as discussed in [15]. One of the biggest issues with neural networks is the "catastrophic forgetting" problem [16]. The current code can add neurons and synapses to handle new situations without affecting the previously trained synapses.

4 CPU TIME ESTIMATES

This computer code has been run on computers at Xsede.org in order to measure CPU time and memory requirements. This is a complicated task since the CPU time depends on the number of neurons, the number of synapses, the firing rates of the neurons, how many neurons are typically firing, the processor speeds, and the inter-processor communication speeds.

Since equation (1) gives an estimate of the number of synapses in biological systems given the number of neurons, it is used to estimate number of synapses in the simulations. This was shown in Figure 1.

We also know that the neurons require 69 floating point operations for time step, and each time step represents 0.1 milliseconds of real time. So one second of real time requires 10,000 steps or 690,000 floating point operations. We also assume a 50 Hz neuron firing rate and at any given time only about 5% of the neurons are firing, which is representative of some biological systems.

Table 2 shows preliminary code performance numbers for up to 2,048 processors. The performance will vary depending on the network connectivity.

Table 2. Preliminary code performance for 300 time steps on Gordon computer at xsede.org.

No. Processor Cores	Total No. Neurons	Total No. Synapses	CPU Time (sec.)	Memory Required (Bytes)
1	$3 * 10^3$	$2 * 10^6$	0.36	$6 * 10^9$
32	$1 * 10^5$	$6 * 10^7$	0.20	$2 * 10^8$
256	$8 * 10^5$	$5 * 10^8$	0.26	$1 * 10^{12}$
2048	$6 * 10^6$	$4 * 10^9$	0.26	$12 * 10^{12}$

5 CONCLUSIONS

Human brain scale simulations are now feasible on massively parallel supercomputers. With careful attention to efficient event-driven programming, table lookups, and memory minimization these simulations can be performed.

The next phase of this research will be incorporating learning. We have implemented back propagation on massively parallel computers in the past [17], and could use that for these networks also. We have also implemented spike time dependent plasticity (STDP) in the past for spiking neural networks [18-20], there are still some issues related to supervised learning using that approach.

References

[1] http://www.wormatlas.org/

[2] Skocik, M.J. and Long, L.N., "On The Capabilities and Computational Costs of Neuron Models," IEEE Trans. on Neural Networks and Learning, Vol. 25, No. 8, Aug., 2014.

[3] Long, Lyle N., "Efficient Neural Network Simulations using the Hodgkin-Huxley Equations," Conference on 60 Years of Hodgkin and Huxley, Trinity College, Cambridge, UK, July 12 - 13, 2012.

[4] www.top500.org

[5] Markram, Henry, et al. "Reconstruction and simulation of neocortical microcircuitry." Cell 163.2 (2015): 456-492.

[6] www.nest-initiative.org

[7] https://www.neuron.yale.edu/

[8] Bower, James M. and Beeman David, *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*, Springer, 2012.

[9] Reimann, Michael W., Costas A. Anastassiou, Rodrigo Perin, Sean L. Hill, Henry Markram, and Christof Koch, "Biophysically Detailed Model of Neocortical Local Field Potentials Predicts the Critical Role of Active Membrane Currents," Neuron, Vol. 79, July 24, 2013.

[10] Kelley, T. D., Avery, E., Long, L. N., and Dimperio, E., "A Hybrid Symbolic and Sub-Symbolic Intelligent System for Mobile Robots," InfoTech@Aerospace Conference, Seattle, WA, AIAA, Reston, VA, 2009, AIAA Paper 2009-1976.

[11] Hodgkin, A.L. and A. F. Huxley, "A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes," Journal of Physiology, vol. 117, pp. 500-544, 1952.

[12] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?," IEEE Transactions on Neural Networks, Vol. 15, No. 5, Sep. 2004, pp.1063-1070

[13] http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf

[14] Bostrom, N., SuperIntelligence: Paths, Dangers, Strategies, Oxfor Press, 2014.

[15] Long, L. N., Gupta, A., and Fang, G., "A computational approach to neurogenesis and synaptogenesis using biologically plausible models with learning ," Frontiers in Systems Neuroscience, Conference Abstract: Computational and Systems Neuroscience (COSYNE) Meeting, Salt Lake City, Utah, Feb. 25-28, 2010.

[16] French, Robert M., "Catastrophic forgetting in connectionist networks," Trends in Cognitive Sciences, Volume 3, Issue 4, 1 April 1999, Pages 128–135.

[17] Long, L.N. and Gupta, A., "Scalable Massively Parallel Artifical Neural Networks", Journal of Aerospace Computing, Information, and Communication (JACIC), Vol. 5, No. 1, Jan., 2008.

[18] Gupta, A. and Long, Lyle N., "Hebbian Learning with Winner Take All for Spiking Neural Networks", IEEE International Joint Conference on Neural Networks (IJCNN), Atlanta, Georgia, June 14-19, 2009.

[19] Long, Lyle N., "An Adaptive Spiking Neural Network with Hebbian Learning." presented at the IEEE Workshop on Evolving and Adaptive Intelligent Systems, Symposium Series on Computational Intelligence, Paris, France, April 11-15, 2011

[20] Long, Lyle N., " Scalable Biologically Inspired Neural Networks with Spike Time Based Learning," Invited Paper, IEEE Symposium on Learning and Adaptive Behavior in Robotic Systems, Edinburgh, Scotland, Aug. 6-8, 2008.