Bringing high performance neural simulation to the desktop with BrainGrid

Michael Stiber [a], Fumitaka Kawasaki [a], Maxfield Strange [a], Andrew Watson [a]

[a] Computing and Software Systems Division, School of STEM
University of Washington Bothell
Bothell, WA, USA
stiber@uw.edu – depts.washington.edu/biocomp

Neuroscience has benefitted as much as any scientific endeavor from becoming a computational science. Of course, computers have played a major role in neuroscience for decades (such as in [1]), but modern hardware now presents researchers with access to inexpensive desktop high-performance computing capabilities that rival that of recent-vintage supercomputers (for example, the NVIDIA Tesla K80 graphics processor has almost 5,000 processors with an aggregate performance of nearly 9 teraflops) for costs that range from the hundreds to a few thousand dollars.

Taking advantage of this computing power, however, is problematic. General-purpose simulation environments, such as Neuron [2] and GENESIS [3], focus primarily on supporting high-level, physiological descriptions of cells and networks and, as such, target single-processor platforms (whose performance characteristics have been flattening out in recent years) or networked clusters (which are expensive, difficult to maintain, and unlikely in general to provide significant performance increase). Other, special-purpose simulators targeting graphics processing units (GPUs), such as [4], have limited flexibility and would require significant GPU-oriented software development for most computational neuroscience investigations. Generally speaking, developing non-trivial GPU programs can take weeks to months. Moreover, while validation of simulation software is difficult in general, it is even more so for parallel hardware.

The BrainGrid simulation framework [5] has been developed to help researchers take advantage of inexpensive, modern multiprocessor hardware to either significantly speed up large and long-duration simulations or enable simulations that are impractical on general-purpose hardware, either singly or as clusters. This framework targets three pain points in such work: (1) time and difficulty in developing GPU code, (2) difficulty in validating correctness of parallel code, and (3) difficulty in gaining significant performance increases from parallel hardware, especially given the idiosyncrasies of neural simulation algorithms.

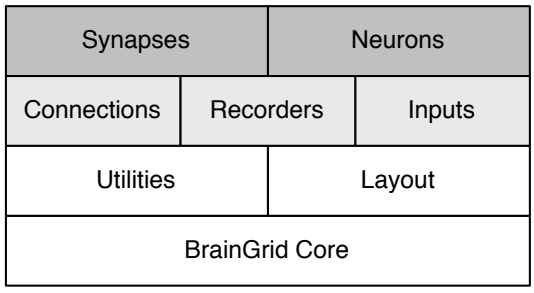

Figure 1: BrainGrid architecture. Dark grey: modules that require some coding to create new mathematical models; light grey may need additional code for specialized learning rules, stimulus protocols, or data collection.

Figure 1 shows this framework's structure. We assume that investigators intend to write their own simulation code. The BrainGrid framework isolates investigator code to the smallest possible context, often part of a single function, and provides coding patterns to further reduce the need to write code from scratch and simplify programming. It includes subsystems optimized for GPU-based neural simulations. Validation is facilitated by pre-structuring code so that patterns that are efficient on the GPU will run on an ordinary, single processor CPU. As a result, code can be written and validated first in a familiar CPU environment and then migrated to a GPU, with only minor changes, minimizing situations in which bugs can arise and maximizing performance. This framework inverts the usual approach to easing GPU software development, in which the GPU programming environment is made to look like the CPU environment. As a result, BrainGrid can achieve speedups in excess of 20X on six-year-old GPU technology [6] (more than 40X on current vintage hardware), as opposed to two to three times using others' methods. In addition, because the elements of the framework that optimize neural simulation algorithms on GPU hardware are part of the simulator core, existing models can take advantage of software and hardware performance improvements without need for modification.

BrainGrid's utility has been demonstrated in simulations of development and bursting in cortical cultures that involved 10,000 neurons, more than 450,000 synapses, and 600 million time steps [7], reducing what would have been impractically long 6-9 month simulations to 3-4 days' duration. BrainGrid is under active development by software engineers and is made available with an open source license. An extension to BrainGrid, the BrainGrid workbench, will use software engineering best practices to facilitate more rigorous testing and indicate when changes to software may invalidate the results of previous simulations.

Keywords: simulation, tools, high performance computing

References

[1] Perkel, D.H., Moore, G.P., and Segundo, J.P. (1963) Continuous-time simulation of ganglion nerve cells in *aplysia*. in *Biomedical Sciences Instrumentation*. Plenum Press, New York.

[2] Carnevale, N.T. and Hines, M.L. (2006) *The NEURON Book*. Cambridge University Press, Cambridge, UK.

[3] J. M. Bower and D. Beeman. (1998) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System*, Second edition, Springer-Verlag, New York.

[4] Minkovich, K., Thibeault, C. M., O'Brien, M. J., Nogin, A., Cho, Y., and Srinivasa, N. (2014). HRLSim: a high performance spiking neural network simulator for GPGPU clusters. *IEEE Transactions on Neural Networks and Learning Systems* 25(2): 316-331.

[5] The BrainGrid Project. (2016) http://github.com/UWB-Biocomputing/BrainGrid

[6] Kawasaki, F. and Stiber, M. (2012) Accelerating large-scale simulations of cortical neuronal network development. *Tech. Rep. UWB-CSS-12-01*, University of Washington Bothell Computing and Software Systems Program, Bothell, WA. http://www.uwb.edu/css/about/faculty/tech-reports

[7] Kawasaki, F. and Stiber, M. (2014) A simple mode of cortical culture growth: burst property dependence on network composition and activity. *Biological Cybernetics*, 108: 423–443.