Non-Neural Network Applications for Spiking Neuromorphic Hardware

James B. Aimone, Kathleen E. Hamilton, Susan Mniszewski, Leah Reeder, Catherine D. Schuman, William M. Severa

Abstract—Increasing power costs for large-scale computing in a post-Moores Law system have forced the high-performance computing community to explore heterogeneous systems. Neuromorphic architectures, inspired by biological neural systems, have so far been relegated to auxiliary machine learning applications. Here, we discuss growing research showing the viability of ultra-low-power neural accelerators as co-processors for classic compute algorithms, such as random walk simulations and graph analytics.

1 Introduction

Spiking neuromorphic computers are non-von Neumann computing systems where the architecture is inspired by biological neural systems. They are attractive as a post-Moore's law technology because they can perform certain workloads with significantly less power than traditional architectures. They have recently seen a resurgence due to increasing computational requirements for state-of-the-art deep learning and the need for lower power processing. Particularly in the case of large-scale supercomputing systems, power consumption has and will continue to be a major issue moving forward, as some proposed exascale systems will consume tens to hundreds of megawatts of power [8], [19], so the inclusion of low-power neuromorphic systems is extremely attractive.

Given the incredible amount of industry interest in deep learning, it is perhaps unsurprising that numerous platforms have been developed or specialized for neural networks [11], [14], [16]. However, we concern ourselves specifically with *spiking neuromorphic hardware* which implement a more biologically realistic, dynamic neuron model with sparse, low-precision (spiking) communication [9], [15]. The underlying substrate varies and may be analog or digital [5], [7], [9], [12], [20], [21]. Existing large-scale systems, such as IBM TrueNorth, SpiNNaker, and Intel Loihi can simulate hundreds of thousands or neurons or more in real-time [3], [5], [12].

Since spiking neuromorphic systems are typically spiking neural networks implemented in hardware, neural network-style computations are the workloads that map most naturally to spiking neuromorphic systems, and the vast majority of neuromorphic literature has been focused on this use case [21]. However, there is potential to utilize spiking neuromorphic systems for a variety of non-neural network applications by utilizing the underlying architectural properties of spiking neuromorphic systems. Those properties include: massively parallel computation, collocated processing and memory, simple processing elements that perform specific computations (e.g., integrate-and-fire for neurons, plasticity mechanisms for synapses), very simple communication between components (usually in the form of spikes), event-driven computation, often resulting in low power consumption, stochastically firing neurons for noise, and inherently scalable architectures.

Because of these architectural properties, there is growing interest in applying neuromorphic systems to classic computing challenges, such as matrix operations, random walk simulations, and graph analytics. Here, we discuss several example non-neural network applications that utilize one or more of these properties to implement those applications on spiking neuromorphic systems.

- *K. Hamilton and C. Schuman are with Oak Ridge National Laboratory, Oak Ridge, TN, USA*
- *J. Aimone, L. Reeder, and W. Severa are with Sandia National Laboratories, Albuquerque, NM, USA.*
- *S. Mniszewski is with Los Alamos National Laboratory, Los Alamos, NM, USA.*

PMES Workshop, Dallas, 11 Nov 2018. `http://j.mp/pmes18`

2 Example Non-Neural Network Applications

2.1 Graph Algorithms and Graphical Optimization

The underlying structure of spiking neuromorphic systems are weighted, directed graphs and this leads to a growing application domain: graph algorithms and graphical optimization problems. There have been several approaches for utilizing spiking neuromorphic systems on graph problems and each of the works discussed here takes a radically different approach in mapping a graph algorithm to a neuromorphic system, where some are more straightforward than others. Embedding a graph into a spiking processor could be as simple as directly mapping each vertex to a neuron, and each edge to a synapse. In the context of graph algorithms, synaptic weights can act as indicator flags to identify which edges are active in neuron firing. Finally, the full algorithm implementation may be executed only on the neuromorphic hardware, or it many require a workflow that incorporates a traditional processor.

In [3], [18], an algorithm for single source shortest path finding is given that relies on the event-driven nature of spiking neuromorphic systems as well as synaptic plasticity to give the shortest path routes between nodes. In [4], Guerra and Furber frame vertex coloring as a constraint satisfaction problem (CSP), and demonstrate a solution using spiking neural networks on SpiNNaker. In this work, they give a general approach to mapping CSPs to spiking neural networks, opening up a wider array of potential applications beyond graph algorithms. In [2], Corder, et al., they demonstrate solving the vertex cover problem on TrueNorth by framing the problem as an Ising model. In [6] Hamilton, et al., a fully connected system of spiking neurons is used to identify communities on a graph through spiking label propagation. However this is an example of a workflow that requires heavy use of traditional hardware, since a full spike raster is generated from a spiking neuron network, which is then decoded and analyzed on a CPU.

In [24] graph partitioning (GP) is framed as a Quadratic Unconstrained Binary Optimization (QUBO) or Ising model for solving by the D-Wave quantum annealer. This QUBO approach is not limited to graph algorithms, but opens the door to solving a spectrum of NP-hard optimization problems. In the context of GP, the QUBO formulation as a weight matrix is mapped to neuromorphic hardware (e.g., IBM's TrueNorth) where each node is represented by a neuron and each weight is mapped to a synapse between neurons [13]. Some leakage is added to each neuron and the integrate-and-fire dynamics causes search of the input sampling space till convergence. Feedback from the firing neurons becomes input to related neurons (based on the QUBO matrix) helping steer the search.

Guerra [4] and Jonke [10] have both previously used the stochasticity available in a neuromorphic system to solve optimization problems, such as CSP. A more direct algorithmic approach for solving a GP QUBO was added using a pseudo simulated annealing metaheuristic. Spontaneous Stochastic Leak (SSL) neurons and stochastic leak/decay provide a way to explore the larger sampling space (by adding or removing nodes). Temperature is represented by the probability of firing or leak. The temperature starts at a high value (such as 50%) and is slowly reduced down to a low value (such as 8%). Limitations of the IBM TrueNorth framework required the creation of a separate corelet for each temperature. The resulting converged high energy solutions are good enough to optimal for 2-partitioning of graphs.

2.2 Solving partial differential equations through neural algorithms for Markov random walk

Large scientific computing tasks, often based on systems of partial differential equations (PDEs), have long been one of the driving applications for large HPC systems. While numerical solutions of PDEs are a natural fit for conventional von Neumann architectures, neural approaches have often been seen as non-ideal for the requisite high-precision arithmetic.

Recently, however, spiking neuromorphic approaches have shown early promise on classic scientific computing applications. One example is with diffusion [22]. Diffusion is part of complex computing applications in fields ranging from finance to molecular dynamics. While typically solved as a PDE, diffusion can also be computed as a system of Markov random walks (RWs), with the statistics of the walkers statistically approximating the numerical PDE solution. Notably, the approach in [22] shows that RWs can be implemented in two fundamentally different neural representations. Each of these neural RW algorithms offers distinct scaling trade-offs that, depending on the application, can show complexity advantages when mapped to spiking neuromorphic hardware. In principle, while [22] took advantage of the simple arithmetic and inherently parallel structure of RWs, the benefits of neural hardware for scientific tasks is likely not limited to stochastic methods. Even complex PDEs often aggregate over many relatively simple physical processes acting in parallel. Considering neural hardware as a uniquely structured incredibly parallel architecture may enable new perspectives on how to approach these classes of problems.

2.3 Composite Algorithms through Utility and Numerical Kernels

One challenge facing spiking co-processors is the wildly different and disparate data representations. Conversions between neural representations and non-neural representations is non-trivial, and post-simulation analysis of spiking patterns can be as computationally expensive as the original task. Hence, individual optimized computations are likely insufficient to justify a future neural-heterogeneous HPC platform on their own. However, it is possible and advantageous to provide neural composite algorithms which link spiking neural algorithms with other spiking neural algorithms. In this way, we can avoid the costly conversion and host-side readout.

This requires the development of algorithmic neural building blocks; some useful kernels have been previously discovered. As examples, utility functions such as sorting and finding the maximum/minimum/median can be efficiently computed using spike timing [26], and high fan-in neurons allow for constant-time, subcubic matrix multiplication [17]. Ensuring compatible neural representations accomplishes seamless transitions from one component to another, and spiking neural algorithms (such as those for cross-correlation [23]) may be designed around various codings and tradeoffs (e.g. unary versus temporal coding or time versus space requirements). When combined these networks are capable of computing comprehensive algorithms rather than only simple tasks.

2.4 Scientific Simulations

Clearly a key use case for neuromorphic systems beyond neural networks for machine learning is simulation of biological neural systems for neuroscience purposes [25], [27]. Because of the parallel discrete event system processing implemented on most neuromorphic systems, there is potential to use them for simulating other behaviors as well. A key example in the literature is given by Araujo, et al., in [1], where they show that SpiNNaker can be used to simulate the aggregate motion of a flock of birds, implemented using Boids model. Though there is relatively little work in this space as of yet, we expect that as more neuromorphic systems become widely available, there will be more exploration of discrete event simulations on neuromorphic architectures.

3 Discussion and Conclusions

Unlike quantum computing, which offers potent theoretical advantages despite its practical challenges, neuromorphic computing has received rather limited attention as a post-Moore's Law technology outside of machine learning. Arguably this oversight is due to the historic connections of neural networks; for many applications approximate learned solutions are undesirable, and the perception that neuromorphic computing is inherently low-precision or noisy.

However, as the above applications illustrate, there is no fundamental reason that spiking neuromorphic approaches cannot be used for conventional computing applications. Neuromorphic architectures can be thought of as a specialized parallel computer, albeit one with rather simple nodes and a unique connectivity structure that requires new paradigms in how to think about algorithms. Neural approaches, which provide opportunities for reduced power usage and effectively faster solutions, also introduce a unique blend of complexity trade-offs (i.e., space vs. time vs. power) that correspond to the better understood compute vs. memory tradeoffs in conventional systems.

Finally, this paper has discussed using neuromorphic hardware for non-machine learning, but it is important to note that learning - which will likely be a capability of future neuromorphic hardware [3] - potentially could be used in non-obvious ways to extend the applications mentioned above. A key question for future research in this area is how the adaption processes that are fundamental to biological neurons can be leveraged to reliably extend our computational capabilities for numerical tasks.

ACKNOWLEDGMENTS

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

This paper describes technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

This manuscript has been authored in part by UT-Battelle, LLC, under Contract No. DE-AC0500OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for the United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan.

REFERENCES

[1] R. Araújo, N. Waniek, and J. Conradt, "Development of a dynamically extendable spinnaker chip computing module," in *International Conference on Artificial Neural Networks*. Springer, 2014, pp. 821–828.

[2] K. Corder, J. V. Monaco, and M. M. Vindiola, "Solving vertex cover via ising model on a neuromorphic processor," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.

[3] M. Davies, N. Srinivasa *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[4] G. A. Fonseca Guerra and S. B. Furber, "Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems," *Frontiers in neuroscience*, vol. 11, p. 714, 2017.

[5] S. B. Furber, D. R. Lester *et al.*, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, 2013.

[6] K. E. Hamilton, N. Imam, and T. S. Humble, "Community detection with spiking neural networks for neuromorphic hardware," in *Proceedings of the Neuromorphic Computing Symposium*. ACM, 2017, p. 9.

[7] J. Hasler and H. B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in neuroscience*, vol. 7, p. 118, 2013.

[8] S. Hemmert, "Green hpc: From nice to necessity," *Computing in Science & Engineering*, vol. 12, no. 6, pp. 8–10, 2010.

[9] G. Indiveri, B. Linares-Barranco *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, p. 73, 2011.

[10] Z. Jonke, S. Habenschuss, and W. Maass, "Solving constraint satisfaction problems with networks of spiking neurons," *Frontiers in Neuroscience*, vol. 10, pp. 1–16, 2016.

[11] N. Jouppi, "Google supercharges machine learning tasks with tpu custom chip," *Google Blog, May*, vol. 18, 2016.

[12] P. A. Merolla, J. V. Arthur *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[13] S. M. Mniszewski, G. Kenyon, and A. Sornborger, "LANL neuromorphic overview and update," *ASC Tri-Lab Neuromorphic Meeting*, 2018.

[14] D. Moloney, B. Barry *et al.*, "Myriad 2: Eye of the computational vision storm," in *Hot Chips 26 Symposium (HCS), 2014 IEEE*. IEEE, 2014, pp. 1–18.

[15] E. Neftci, "Data and power efficient intelligence with neuromorphic learning machines," *iScience*, 2018.

[16] K. Ovtcharov, O. Ruwase *et al.*, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Research Whitepaper*, vol. 2, no. 11, 2015.

[17] O. Parekh, C. A. Phillips *et al.*, "Constant-depth and subcubic-size threshold circuits for matrix multiplication," in *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. ACM, 2018, pp. 67–76.

[18] F. J. Ponulak and J. J. Hopfield, "Rapid, parallel path planning by propagating wavefronts of spiking neural activity," *Frontiers in computational neuroscience*, vol. 7, p. 98, 2013.

[19] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Communications of the ACM*, vol. 58, no. 7, pp. 56–68, 2015.

[20] J. Schemmel, D. Briiderle *et al.*, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on*. IEEE, 2010, pp. 1947–1950.

[21] C. D. Schuman, T. E. Potok *et al.*, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[22] W. Severa, R. Lehoucq *et al.*, "Spiking neural algorithms for markov process random walk," *arXiv preprint arXiv:1805.00509*, 2018.

[23] W. Severa, O. Parekh *et al.*, "Spiking network algorithms for scientific computing," in *Rebooting Computing (ICRC), IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[24] H. Ushijima-Mwesigwa, C. F. A. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the d-wave system," in *Proceedings of the Second International Workshop on Post Moore's Era Supercomputing (PMES)*. ACM, 2017, pp. 22–29.

[25] S. J. van Albada, A. G. Rowley *et al.*, "Performance comparison of the digital neuromorphic hardware spinnaker and the neural network simulation software nest for a full-scale cortical microcircuit model," *Frontiers in neuroscience*, vol. 12, 2018.

[26] S. J. Verzi, F. Rothganger *et al.*, "Computing with spikes: The advantage of fine-grained timing," *Neural computation*, pp. 1–31, 2018.

[27] R. M. Wang, C. S. Thakur, and A. van Schaik, "An fpga-based massively parallel neuromorphic cortex simulator," *Frontiers in neuroscience*, vol. 12, p. 213, 2018.