First Impressions from Detailed Brain Model Simulations on a Xeon/Xeon-Phi Node

George Chatzikonstantis
MicroLab-ECE-NTUA, Greece
georgec@microlab.ntua.gr

Dimitrios Rodopoulos*
MicroLab-ECE-NTUA, Greece
drodo@microlab.ntua.gr

Sofia Nomikou
MicroLab-ECE-NTUA, Greece
nomikou@microlab.ntua.gr

Christos Strydis
EMC, Netherlands
c.strydis@erasmusmc.nl

Chris I. De Zeeuw
EMC, Netherlands
c.dezeeuw@erasmusmc.nl

Dimitrios Soudris
MicroLab-ECE-NTUA, Greece
dsoudris@microlab.ntua.gr

ABSTRACT

The development of physiologically plausible neuron models comes with increased complexity, which poses a challenge for many-core computing. In this work, we have chosen an extension of the demanding Hodgkin-Huxley model for the neurons of the Inferior Olivary Nucleus, an area of vital importance for motor skills. The computing fabric of choice is an Intel Xeon-Xeon Phi system, widely-used in modern computing infrastructure. The target application is parallelized with combinations of MPI and OpenMP. The best configurations are scaled up to human InfOli numbers.

CCS Concepts

•**Applied computing** → **Biological networks;**

Keywords

MPI, Neuron Modeling, OpenMP, Performance

1. INTRODUCTION

Neuroscientists introduce workloads of constantly growing complexity in their efforts to reveal details of neuron operation. To this end, software, such as NEURON [13] and NEST [21], has been developed for brain simulation and ported on GPUs and many-core platforms [20, 18]. We present an inferior-olivary (InfOli) simulator based on extended Hodgkin and Huxley (HH) models [14] and its porting on the Intel Xeon-Xeon Phi host-and-coprocessor system [11]. The Xeon Phi coprocessor features up to 61 cores, each with four instruction streams and 512-bit-sized vectorization processing units (VPU), for simultaneous floating-point operations [16]. The system supports traditional parallel coding, such as MPI [23], OpenMP [7] and combinations thereof.

*D. Rodopoulos is also with ESAT-KU Leuven, Belgium

CF'16 May 16-19, 2016, Como, Italy

ACM ISBN 978-1-4503-4128-8/16/05.

DOI: http://dx.doi.org/10.1145/2903150.2903477

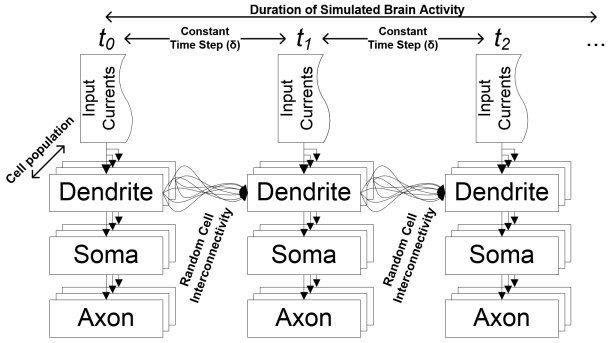


Figure 1: Flowchart of the InfOli simulator

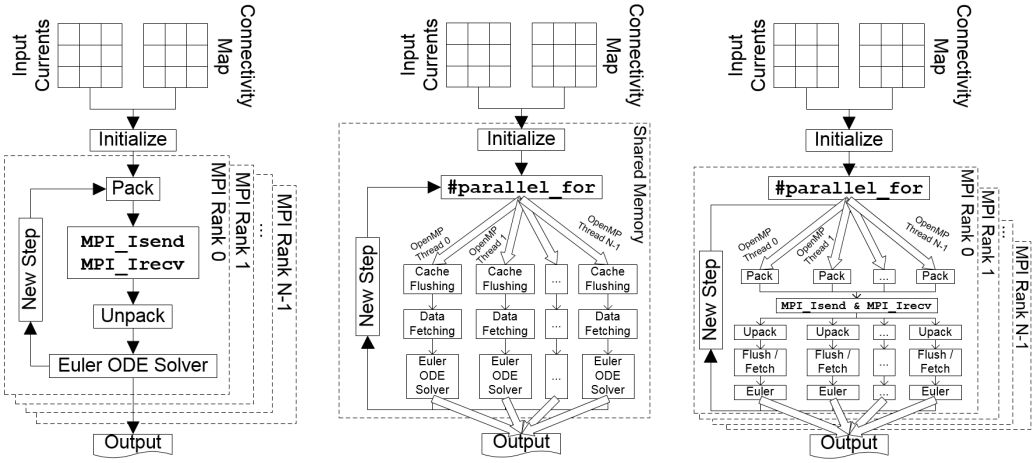
The contributions of the current paper are: (i) Three different implementations are presented for the InfOli simulator (MPI, OpenMP, and hybrid); (ii) Their performance is evaluated natively on Xeon and Xeon Phi platforms and optimal combinations of implementation and platform are identified; and (iii) The best implementation is scaled up to support populations of an entire adult human InfOli [19].

The paper is organized as follows: Section 2 offers insight into neuron modeling and related work on many-core platforms. Section 3 details the parallelization of the InfOli simulator on the Xeon-Xeon Phi system. Section 4 elaborates on the performance measurements of the discussed implementations. Section 5 concludes this work.

2. PRIOR ART AND MOTIVATION

Spiking Neural Networks (SNNs) [5] are categorized as Integrate-and-Fire (I&F) or conductance-based models. I&F are simple models, determining the neuron's response based on a voltage threshold, with variations such as the exponential I&F [4] and the Izhikevich [15] models. HH are prominent conductance-based models, using complex differential equations to expose the neuron's electrochemical properties.

The target model, a tri-compartmental HH extension originally designed by Gruijl et al. [9], is used to describe neurons of the inferior-olivary region [10]. The model, described in Figure 1, consists of the soma, the axon and the dendrite; the soma serves as the neuron's main computational body, the axon connects to other parts of the brain and the dendrite handles inter-neuron connections. The detailed synapses (gap junctions - GJs) in the dendrite present a parallelization bottleneck due to the volume of data exchanged.



(a) MPI (b) OpenMP (c) Hybrid

Figure 2: Flowchart of the implementations discussed in the current paper

To accelerate neuron-model simulations, Fidjeland et al. [12] have deployed 40,000 densely-connected Izhikevich neurons on GPUs. Bhuiyan et al. [3] accelerated up to millions of sparsely-connected Izhikevich- and HH-neurons on various platforms. Choi et al. [6] have developed 1,000 *in silico* spiking Izhikevich neurons on a Xilinx FPGA. Partners of the Human Brain Project have used GPUs and FPGAs for HH-modeling of the human cerebellum [8], scaling up to 400,000 neurons. SNN-simulation toolboxes for FPGAs [24] and GPUs [2] also exist.

We explore the Many Integrated Core (MIC) architecture via a Xeon Phi accelerator card. It features $\geq$ 57 multi-threading cores, along with VPUs that allow Single Instruction Multiple Data (SIMD) execution of FP-operations. The host boots a minimal Linux image on the card, so it can be used as a standalone processor. There exist specialized tools that aid with SIMD execution [17]. Contrary to related work, where a custom implementation of this simulator was ported on a research-grade chip [22], here we evaluate implementations that are generic enough to be platform-agnostic, thus excluding manual vectorization from this work and relying solely on compiler optimizations.

3. IMPLEMENTATION DETAILS

Our InfOli-networks are data-partitionable; each core is assigned the computation of a different part of the network, while core syncing is imposed by GJs, the biological neuron communication mechanism. In Figure 2a, the implementation uses MPI [23], a library that allows data-exchanging between cores over shared memory in our single-node system and over TCP and Infiniband in multi-node systems. In each core, a single unit of execution spawns, called MPI rank, which handles a subset of the neuronal network. To simulate GJs, one option is neuron-to-neuron communication. Assuming a neuron population N, the number of `MPI_Isend` and `MPI_Irecv` pairs required under worst-case conditions (fully-connected network) is $(N-1)^2$. An alternative technique exchanges data in bundles containing all GJ-required data expected by another core. Bundling the data is called *packing*, while *unpacking* is the reverse procedure of extracting the data from received bundles. Assuming k MPI ranks, the worst- case number of `MPI_Isend` and `MPI_Irecv` pairs is $(k-1)^2$, whereas each MPI call exchanges at most N/k more data than the neuron-to-neuron case. Bundle processing imposes timing overheads in each simulation step but outperforms neuron-to-neuron communication.

The OpenMP [7] implementation in Figure 2b uses OpenMP threads as primary units of execution. Via `#pragma omp` directives, the threads compute in parallel different parts of the network. Since memory is shared between the threads, each one can freely access another thread's data. Thus, both computation (i.e. ODE solution) and data-fetching are carried out locally but cache-coherence introduces MESI-protocol-related overheads. Overheads become more pronounced when simulating a small network and creating disproportionately too many OpenMP threads.

In the hybrid implementation of Figure 2c , the cores of a platform are organized into *groups*, which are perceived as MPI ranks and serve as the primary units of execution. Within each group, all cores communicate over shared memory, spawning OpenMP threads for task acceleration. In each group, one "master" core performs single-threaded MPI calls for inter-group data-exchange, whereas packing and unpacking of the data is performed by the OpenMP threads spawned by the entire group. The implementation is logically extensible to multi-node platforms, as long as each node is organized into different groups.

4. PERFORMANCE MEASUREMENTS

We used the Blue Wonder cluster, at the Hartree Center of the Science & Technology Facilities Council (STFC), which features nodes of one Intel Xeon E5-2697v2 processor (dual-socket arrangement) and one Intel Xeon Phi 5110P accelerator. We ran experiments for 5 s of simulated neural activity with a constant simulation step δ= 50 μs natively on the platforms. The network's connectivity map is created by a probability-based generator, similar to the one described in [22]. Compiling has been carried out with the Intel C compiler and performance metrics were collected via the Intel VTune performance analyzer.

In Figure 3, the Xeon Phi underperforms since a strictly MPI-based implementation does not utilize the platform's

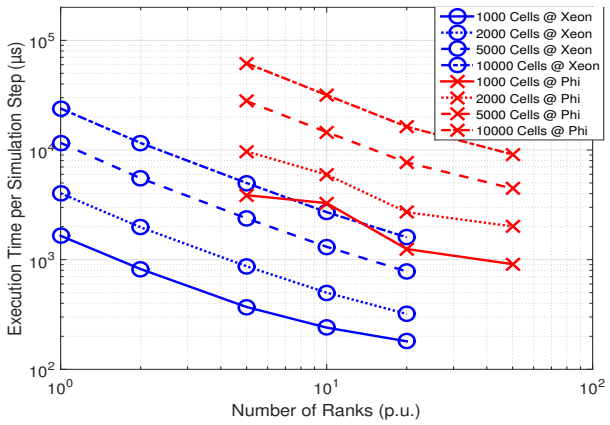


Figure 3: MPI performance on Xeon and Xeon Phi

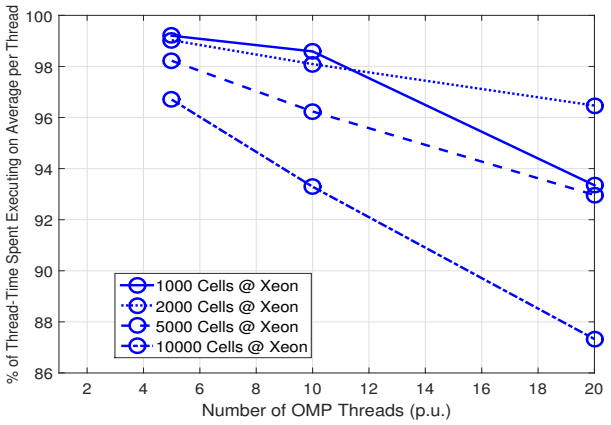


Figure 5: OpenMP thread activity on the Xeon host

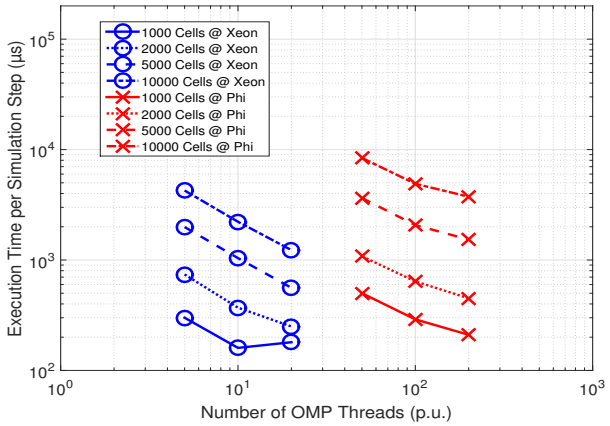


Figure 4: OpenMP performance on Xeon and Xeon Phi

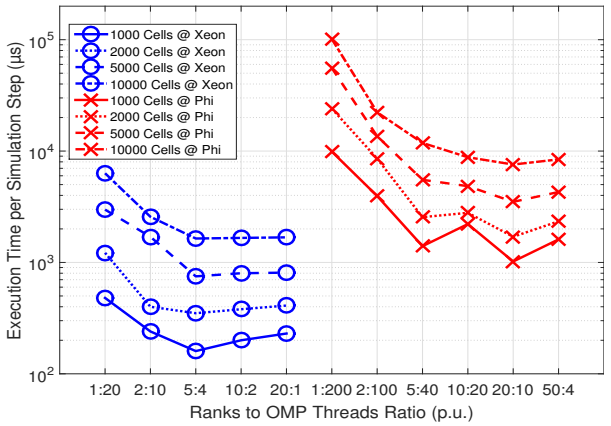


Figure 6: Hybrid performance on Xeon and Xeon Phi

multithreading capabilities; intra-core rank communication forces the MPI implementation to use a single rank per core employed. We observe sub-optimal efficiency when simulating 2,000 or less neurons on the Phi. This behavior suggests that small workloads do not fully exploit the computational resources. This is not observed for larger networks, indicating that the MIC architecture's high memory-bandwidth can satisfy the scaling data-exchanging demands, even when employing 50 MPI ranks.

In Figure 4, OpenMP outperforms MPI on the Xeon Phi by exploiting the accelerator's multithreading resources more aggressively. Whereas the Xeon Phi shows near-linear performance gains when increasing the number of invoked OpenMP threads, solving for small populations does not exhibit consistent scaling for the Xeon host. We hypothesize that the *benefit* of employing large numbers of OpenMP threads lies in reducing each thread's computational burden (each thread solves for less neurons), whereas the *cost* of an OpenMP implementation is related to the overhead of shared-memory operations. On the host, the cost dominates when the neurons-per-thread ratio is low, while the benefit outweighs the cost for larger problem sizes. The claim is supported by Figure 5; Intel defines CPU time as "the amount of time a thread spends executing on a logical processor" [1]. We calculate the mean CPU time per Xeon thread and compare it to the real elapsed time of the workload, thus arriving at the percentage of time spent executing on the processor by the average thread. There is a drastic decrease in thread activity when solving for $1,000$ neurons on 20 Xeon threads, which is a low neurons-per-thread scenario.

In Figure 6, different ratios of MPI ranks to OpenMP threads spawned per rank are used to utilize each target platform at a constant, maximum capacity. We observe that both platforms perform better with balanced MPI-to-OpenMP ratios (5:4 and 20:10 for the Xeon host and the Xeon Phi, respectively). "Middle-of-the-road" approaches appear able to balance the burden of message exchange between a reasonable number of core-groups, while keeping the workload of each group big enough to near-maximally utilize computational resources via OpenMP threads.

In Figure 7, the best configurations scale to larger populations, while keeping an upper bound on execution times set at six hours. The Xeon Phi accelerator cannot compete with the Xeon host for native execution, hinting at a need to perform manual source code vectorization to use more of the accelerator's assets [17], at the cost of increased development time. OpenMP behaves better for the Phi, however a steep performance curve causes message-passing-based implementations to be favoured for networks of $\geq 20,000$ neurons, especially since they are the only option for multi-node systems. The hybrid implementation narrowly outperforms the pure MPI method by using more of the platform's resources, albeit not efficiently due to OpenMP-imposed overheads. On the host, OpenMP remains the optimal implementation

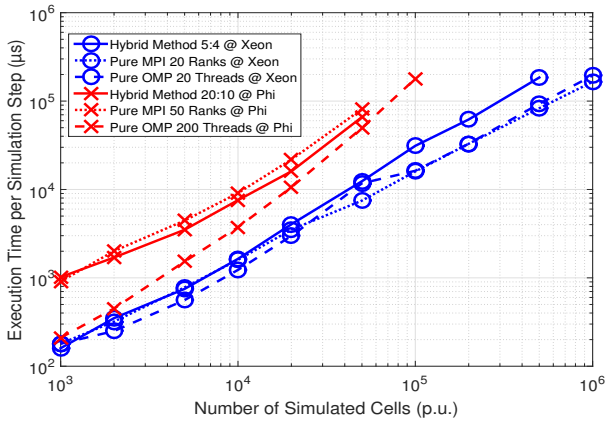


Figure 7: Comparing the best implementations

for smaller networks. For more than 50,000 neurons, pure implementations outperform the hybrid method, since both can fully utilize the Xeon's resources. Finally, MPI slightly outperforms OpenMP for 1 million neurons.

5. CONCLUSIONS

We ported a demanding biological neural-network simulator on a single-node Xeon-Xeon Phi system via three native implementations: an MPI-based one, an OpenMP-based one and a combination of both. The MPI implementation underperforms for the Xeon Phi accelerator since it does not utilize the coprocessor's multithreading resources, but performs well on the host, particularly for more than 10^5 neurons. The hybrid implementation improves on MPI's shortcomings on the accelerator, a feat which is not present in the Xeon host's case due to MPI already utilizing the platform efficiently. OpenMP is the optimal choice on both computing platforms for smaller networks, albeit its performance does not scale linearly in all use-cases. The accelerator's hybrid implementation and the host's pure-MPI programming method rival OpenMP for large networks due to them being extensible to multi-node systems. The Xeon processor exhibits better overall performance than the Xeon Phi, scaling up to a million inferior-olivary nuclei. Their gap in performance may be narrowed by manually vectorizing the source code; when relying on compiler optimizations, we propose that a dual-socket Xeon processor is the preferable choice for single-node simulations of complex brain models.

Acknowledgements

This work is supported by European Commission project H2020–687628–VINEYARD. The STFC Hartree Centre (UK) is acknowledged for providing computational resources.

6. REFERENCES

[1] https://software.intel.com/en-us/node/471922.

[2] Beyeler, M. et al. CARLsim 3: A user-friendly and highly optimized library for the creation of neurobiologically detailed spiking neural networks. In *IJCNN*, pages 1–8, 2015.

[3] Bhuiyan, M. et al. Acceleration of spiking neural networks in emerging multi-core and GPU architectures. In *IEEE IPDPSW*, pages 1–8, 2010.

[4] Brette, R. and Gerstner, W. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637–3642, 2005.

[5] Brette, R. et al. Simulation of networks of spiking neurons: A review of tools and strategies. *J. of Comp. Neuroscience*, 23(3), 2007.

[6] Choi, J. et al. Implementation of hardware model for spiking neural network. In *ICAI*, page 700, 2015.

[7] L. Dagum and R. Enon. Openmp: an industry standard api for shared-memory programming. *IEEE CSE*, 5(1):46–55, 1998.

[8] D'Angelo, E. et al. The human brain project: High performance computing for brain cells hw/sw simulation and understanding. In *DSD*, 2015.

[9] J. R. De Gruijl, P. Bazzigaluppi, M. T. de Jeu, and C. I. De Zeeuw. Climbing fiber burst size and olivary sub-threshold oscillations in a network setting. 2012.

[10] De Zeeuw, C. I. et al. Microcircuitry and function of the inferior olive. *Trends in neurosciences*, 21(9):391–400, 1998.

[11] Fang, J. et al. Test-driving intel xeon phi. In *ACM/SPEC ICPE*, pages 137–148, 2014.

[12] Fidjeland, A. K. et al. . Nemo: a platform for neural modelling of spiking neurons using GPUs. In *IEEE ASAP*, pages 137–144, 2009.

[13] M. L. Hines and N. T. Carnevale. The NEURON simulation environment. *Neural computation*, 9(6):1179–1209, 1997.

[14] A. L. Hodgkin and A. F. Huxley. Propagation of electrical signals along giant nerve fibres. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 140(899):177–183, 1952.

[15] Izhikevich, E. M. et al. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[16] J. Jeffers and J. Reinders. *Intel Xeon Phi Coprocessor High-Performance Programming*. Elsevier, 2013.

[17] Lubin, M. et al. Efficient Software Development: 4 What's New in Intel® Parallel Studio XE 2013 Service Pack. 2013.

[18] M. Migliore, C. Cannia, W. W. Lytton, H. Markram, and M. L. Hines. Parallel network simulations with NEURON. *Journal of Computational Neuroscience*, 21(2):119–129, 2006.

[19] R. D. Monagle and H. Brody. The effects of age upon the main nucleus of the inferior olive in the human. *Journal of Comparative Neurology*, 155(1):61–66, 1974.

[20] Nguyen, H. A. Du et al. Accelerating complex brain-model simulations on GPU platforms. In *DATE*, pages 974–979, 2015.

[21] Plesser, H. E. et al. Nest: the neural simulation tool. *Enc. of Comp. Neuroscience*, pages 1849–1852, 2015.

[22] Rodopoulos, D. et al. Optimal mapping of inferior olive neuron simulations on the single-chip cloud computer. In *IEEE SAMOS*, 2014.

[23] M. Snir. *MPI–the Complete Reference: The MPI core*. MIT, 1998.

[24] Wu, Q. et al. Development of FPGA toolbox for implementation of spiking neural networks. In *CSNT*, pages 806–810, 2015.