Simulation of Biological Neural Microcircuits on Multi-Core Systems

Ioan Lucian Muntean, Marius Joldos
Computer Science Department
technical University of Cluj-Napoca
Romania
Email: {Ioan.Lucian.Muntean, Marius.Joldos}@cs.utcluj.ro

Abstract

Our research focuses on the identification and quantification of the impact that multi-core parallelization strategies have on the stability of the result of spiking neural networks simulations. We investigated OpenMP-based implementations of the Spike Response Model and Spike Time-Dependent Plasticity for studying behaviors of biological neurons and synapses. The underlying neural microcircuits have small-world topologies. The simulation strategy is a synchronous one. The software development methodology we follow makes use of systematic unit testing and continuous integration, giving us a way to verify various perturbations of simulation results. We carried out investigations on systems having different multi-core processors. The processing speed (spikes/second) of our simulator scales well with the number of cores, but the parallel efficiency is moderate when all cores of the system are used in the simulation (0.57 for 12 cores e.g.). The primary outcomes of this work are twofold: One the one hand, the proposed parallel simulation strategies show a dynamic behavior unaltered by the use of multi-core specific technologies. On the other hand, we analyze issues met in our approach to multi-core simulations.

1. Introduction

In many engineering and scientific areas, numerical simulation has established as a valuable investigation instrument, in addition to theory and experiment. Moreover, there are areas/domains, such as computational neuroscience (CNS), in which the simulation became the first choice for the investigation of scientific hypotheses, complementing the invasive nature of in-vivo brain experiments.

The use of multi-core systems for CNS research potentially addresses the high computational demands challenge of these simulations. Nevertheless, we face a slow adoption of multi-core technologies in simulation applications where the dynamics behavior of neural microcircuits is critical. The main reason for that is the almost chaotic nature of the behavior of brain microcircuits that needs to be captured in the computer simulations as well. Therefore, slight changes of the simulation setup (changes in input parameters, topological properties of the microcircuits, numerical methods etc.) could change dramatically the result of the simulation leading to false scientific hypotheses. Thus, our research focuses on the identification and quantification of the impact that multi-core parallelization strategies have on the stability of the result of spiking neural networks simulations. The software development methodology we follow addresses the problem of validating complex codes for computational sciences stated in [1], standing for CNS simulations as well. Our approach makes use of systematic unit testing and continuous integration, giving us a way to verify various perturbations of simulation results. Furthermore, we employ additional simulation tools such as NEURON [2], wide-spread in the CNS community, for constructing reference scenarios against which we compare our results.

In [3], we proposed a framework for the multi-core simulations of biological neural microcircuits and presented the first results obtained for the parallelization of the Hodgkin-Huxley [4] neuron model with OpenMP. Here, we extend the framework with OpenMP-based implementations of the Spike Response Model (SRM) and Spike Time-Dependent Plasticity (STDP) for studying behaviors of biological neurons and synapses. The underlying neural microcircuits have small-world topologies. The simulation strategy is a synchronous one. The approach for dynamics analysis undertaken here is based on the Liapunov exponent method. We carried out investigations on systems having different multi-core processors (Intel Q8400, E5507, and X5680). The processing speed (spikes/second) of our simulator scales well with the number of cores, but the parallel efficiency is moderate when all cores of the system are used in the simulation (0.57 for 12 cores e.g.). The primary outcomes of this work are twofold: On the one hand, the proposed parallel simulation strategies show a dynamic behavior unaltered by the use of multi-core specific technologies. On the other hand, we analyze issues met in our approach to multi-core simulations.

The remaining of this paper is structured as follows: Next section presents other scientific work related to parallel investigations of biological neural circuits. In Sec. 3 we introduce elements of computational neuroscience needed for understanding our work. Our approach to multi-core simulation and the Neurosim simulator we use as framework for this research are discussed in Sec. 4. The results are presented in Sec. 5. Section 6 concludes the paper and outlines potential continuations of this work.

2. Related work

The dynamics of biological neural microcircuits has been studied in works such as [5] and [6], emphasizing the importance of almost chaotic behavior of the brain models in the development of new, robust and real-time computational models. In [7] is proposed a model for the assessment of the
Parallel simulations of biological neural models have been carried out with different technologies and with different goals in mind. In [9] there have been presented strategies for the parallelization of biophysically realistic neural simulations based on the compartmental technique. The multi-core approach uses the Native POSIX Thread Library implementation of the POSIX threads, in order to overcome limitations of OpenMP. The neural models Hodgkin-Huxley and Izhikevich have been employed in [10] many-core-based image recognition with spiking neurons (on GPUs). The network topology was rather simple (two-layered with 9216 to 576000 level 1 neurons and 48 level 2 neurons). Many-core-aware implementations of neuromorphic computational models are presented in [11] (for a cluster of Sony PlayStation3) and in [12] (with CUDA on GPUs).

The Neocortex simulator [13] provides a good base for CNS research, esp. in the areas where the dynamics behavior of microcircuits is very important (resonance or information processing). The topology of the investigated neural microcircuits has small world characteristics. We aim at integrating our parallel implementations in the Neocortex simulator.

3. Simulation of Neural Microcircuits

The simulation of spiking neural microcircuits with biological relevance is a complex task. It involves models where the propagation of spikes takes place sequentially for each connection (synapse) of each neuron, once the spiking threshold is passed. The advantage of the clock-driven approach lays in the simplicity of the handling of neuron and synapse updates. The downside resides especially in considering the same simulation time scale for the entire microcircuit (typically 1 ms). In the brain, processing takes place continuously and, thus, this strategy might be inaccurate for certain investigations.

3.2. Models for Neuron and Synapses

The reader is invited to consult [15] for a detailed presentation of the two models introduced here. We only point out a few elements of these models that impact most their simulation. The notations are taken from [15].

The Spike Response Model (SRM). SRM is a generalization of the leaky Integrate-and-Fire model. Its parameters depend on the time since the last post-synaptic spike. The state of a neuron is solely described by its membrane potential $u_i$, expressed at the time $t$ as an integral over its past activity. An action potential (spike) occurs (is fired) when $u_i$ passes some threshold value.

Considering that the neuron $i$ fired its last spike at time $t_i$, the Eq. 1 gives the evolution of $u_i$ after the firing:

$$u_i(t) = \eta(t - t_i) + h(t|t_i),$$

where

$$h(t|t_i) = \sum_j w_{ij} \sum_{f=1}^{F_{ext}} \epsilon_{ij}(t - t_i, t - t_{ij}^f) + \int_0^{\infty} \kappa(t - t_i, s) F_{ext}(t - s) ds. \quad (2)$$

$t_{ij}^f$ are firing times of presynaptic neurons $j$ and $w_{ij}$ is the synaptic efficacy. The kernel $\epsilon_{ij}(t - t_i, s)$ depends on $s = t - t_{ij}^f$ and gives the evolution in time a postsynaptic potential determined by the firing of a presynaptic neuron $j$ at time $t_{ij}^f$. The last term of the expression, $\kappa$, describes the effects of an external driving current $F_{ext}$.

The evaluation of the membrane potential $u_i$ in the simulation is quite expensive due to the exponential expressions of $\eta$, $\epsilon$, and $\kappa$.

Spike Time-Dependent Plasticity. According to this model, the synapse efficacy $(w_{ij})$ depends on temporal correlations between presynaptic and postsynaptic firing times. The dependency of the synapse’ strength change is expressed in the equation below:

$$\frac{d}{dt} w_{ij}(t) = a_0 + S_j(t) \left[ a_1^{pre} + \int_0^{\infty} a_2^{pre,post}(s) S_i(t - s) ds \right] + S_i(t) \left[ a_1^{post} + \int_0^{\infty} a_2^{post,pre}(s) S_j(t - s) ds \right]. \quad (3)$$

The simulation of neural microcircuits with biophysical details of neurons, synapses, and interconnect topology are taken into account, making their in silico investigation computationally expensive. In this section, we briefly introduce such simulation artifacts that are important to the work presented in our paper.

3.1. Simulation Strategy

We have considered here the synchronous (clock-driven) strategy for simulating spiking neural networks. The key points of this approach are [14]:

- the simulation time advances synchronously for each neuron;
- the dynamics of every neuron is updated at every time step;
where $a_0$ is an activity independent term, $a_1^{pre}$ and $a_2^{pre}$ are non-Hebbian terms modeling the long-term potentiation and long term depression of synapses, resp. $S^p_j(t)$ and $S^j(t)$ are pre- and postsynaptic spike trains, resp. The model parameters $a_0$, $a_1^{pre}$, $a_2^{post}$, and all kernels $a_3^{pre,post}$ and $a_4^{pre,post}$ depend on $w_{ij}$.

Here, we have considered an exponential learning window $W(s)$ given by the expression below:

$$W(s) = \begin{cases} A_pos \exp[\frac{s}{\tau_1}] & \text{for } s < 0, \\ A_neg \exp[-s/\tau_2] & \text{for } s > 0, \end{cases}$$

which also defines the kernels $a_3^{post,pre}$ and $a_4^{pre,post}$.

As in the case of SRM, the STDP calculation is dominated by the evaluation of exponential functions. Furthermore, these models relay on the activity history of neurons, thus requiring that the simulation manages historical data such as the spikes.

### 3.3. Dynamics Analysis

The dynamics of a system (brain microcircuits e.g.) can exhibit different behaviors such as ordered, periodic, chaotic, or random. There are several metrics for analyzing dynamics of brain microcircuits: the Liapunov exponent, system’s entropy, fractal dimension, or the NM separation [6]. We briefly introduce here the Liapunov exponent method. More detailed descriptions of these methods as well as the relation between system’s dynamics and their computability can be found in [16]. In [17], and [18] such metrics are considered for reasoning about the chaoticity of the brain, in particular.

The Liapunov exponent $\lambda$ is computed according to Eq. 4.

$$\delta x(t) = \delta x(0) \exp(\lambda t)$$

Here, $\delta x$ is the time-dependent distance between the trajectories of the simulation at $t$ and represents the time in which the system evolved. If $\lambda \geq 0$, the system exhibits a chaotic behavior, whereas a constant $\lambda$ indicates constant distances between the trajectories, meaning that the system is periodic.

We consider different ways of defining the trajectory of a spiking neural network simulation:

- the activity in the microcircuit at every time step of the simulation (i.e. spikes and firing times);
- the strength of synapses in the microcircuit at every time step;
- both the activity of the neurons and the synaptic changes at every time step.

If $N$ is the number of neurons and $p$ the one of the synapses, the simulation trajectory has $N \times p$, and $N \times p$ dimensions, resp., making the analysis of the dynamics behavior of the microcircuit a complex task by itself.

### 4. The Multicore-enabled Simulator

Within the frame of parallel investigation of the dynamics behavior of neural microcircuits we have developed the Neurosim SNN simulator. In this section we introduce the reader with Neurosim and with our approach to parallel investigations of the models described in Sec. 3.

#### 4.1. Overview of Neurosim

Neurosim is a spiking neural network simulator designed to allow for different combinations of simulation models and parallelization strategies and technologies. Figure 1 depicts the multi-layer organization of this research code.

The top level layer is focused on providing services for data analysis (dimensionality reduction features (based on the Principal Component Analysis), viewers (spike maps, voltage plots), dynamics analysis (the Liapunov exponent method)). The strategy layer implements clock-driven simulation kernels with OpenMP (OpenCL, CUDA and Cilk+ implementations are under development). The models for neurons and synapses cover different levels of biological details (from the very detailed Hodgkin-Huxley one to the Spike Response Model). Configuration management allows for switching between models and parallelization technologies (either at compile-time or at runtime). A suite of network handling and simulation instrumentation modules form the tools layer.

This SNN simulator is intended for use in the grid-based framework for dynamics analysis of biological neural microcircuits presented in [3]. There, simulation results from Neurosim are compared with reference results obtained with other community tools such as NEURON and Neocortex, where sequential processing is employed.

#### 4.2. Parallelization Approach

The approach undertaken to the OpenMP-based simulation of the SRM and STDP models takes into account the potential for parallel processing of the membrane potential of neurons.
(Eq. 1). In a time step \( t \), \( u \) could be evaluated in parallel for each neuron of the microcircuit. Furthermore, the calculation of the synaptic efficacy \( w_{ij} \) could be processed in parallel for each of the synapses of a neuron. Between the time steps, there is an implicit synchronization specific to the clock-driven simulation strategy (see Sec. 3).

The usage of the spike history in simulations involving these two models introduces an additional need for synchronizing the access to the storage location of the spikes of a neuron. Such a situation occurs when a pre-synaptic fires and the spike is propagated in the spike list of the current neuron, which, in turn, is calculating the synapse strength based on the spike time correlation. A convenient way to prevent this happening is to use synchronization locks or thread-safe data structures. As such, this reduces the concurrency potential of the simulation method.

Due to the recurrent nature of the small-world topology of our microcircuits, it practically makes little sense to use a topology-aware scheduling of workload to threads (such as allocation of all neurons from partition with dense connections to the same thread).

4.3. Systematic Development

In order to prepare robust parallel implementations for the integration with other simulation tools such as the Neocortex simulator, we rely on the methodology of our approach (introduced in [3]).

Unit tests cover the verification of the data structures employed in the simulation. The correctness of the algorithm implementations is harder to assess, considering the intensive usage of random numbers in biological neural simulations. Their verification with unit tests is based on the one hand on average values and, on the other hand, on controlling the conditions of the pseudo-random number generators. The execution with multiple thread-based simulation scenarios is tested against reference values accumulated during several time steps.

Scenarios constructed with simulation tools such as NEURON are used as reference in the verification process. Fresh deployments of Neurosim, regression tests for tracking changes that affect one or more of its modules are managed within a continuous integration setup. The setup uses the Hudson [19] integration server and Subversion as source code and scenario repository.

5. Results

5.1. Simulation Setup

The results presented here have been computed on three different multi-core systems:

- **b05.ici:** Intel Xeon X5680, 3.33 GHz, L3 cache 12 MB;
- **acalgrid:** Intel Xeon E5507, 2.26 GHz, L2 cache 4 MB;
- **s10-2:** Intel Core 2 quad Q8400, 2.66GHz, L2 cache 4 MB.

The simulator was compiled with the GNU C++ compiler v4.4, having OpenMP support.

In Table 1 are listed the network configurations used for obtaining the results presented in this section. In addition to these configurations, smaller networks have been used for the dynamics analysis (800 neurons organized in three partitions).

### TABLE 1. Neural microcircuit topologies employed in simulations (N - Neurons, p - Synapses)

<table>
<thead>
<tr>
<th>Alias</th>
<th>#N</th>
<th>#p</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>5000</td>
<td>1000</td>
</tr>
<tr>
<td>S2</td>
<td>5000</td>
<td>1500</td>
</tr>
<tr>
<td>S3</td>
<td>5000</td>
<td>2000</td>
</tr>
<tr>
<td>S4</td>
<td>5000</td>
<td>2500</td>
</tr>
<tr>
<td>S5</td>
<td>7500</td>
<td>1000</td>
</tr>
<tr>
<td>S6</td>
<td>7500</td>
<td>1500</td>
</tr>
<tr>
<td>S7</td>
<td>7500</td>
<td>2000</td>
</tr>
<tr>
<td>S8</td>
<td>7500</td>
<td>2500</td>
</tr>
<tr>
<td>S9</td>
<td>10000</td>
<td>1000</td>
</tr>
<tr>
<td>S10</td>
<td>10000</td>
<td>1500</td>
</tr>
<tr>
<td>S11</td>
<td>10000</td>
<td>2000</td>
</tr>
<tr>
<td>S12</td>
<td>10000</td>
<td>2500</td>
</tr>
</tbody>
</table>

5.2. Dynamics Analysis

Figure 2 shows the spike map difference between four simulations of the same scenario (800 neurons, 300 – 400 synapses/neuron, three partitions). The three plots were obtained by eliminating the spike times from the last three simulations that coincided with the ones in the reference (first) simulation. The dots plotted in Fig. 2 represent neuron firing (spiking) times. The stimulus was applied in the form of Poisson spike trains for a duration of 10 ms. The results were obtained on acalgrid105.

![Fig. 2. Difference in the neuronal activity between four simulations of the same scenario (simulation time 50 ms, 800 neurons in the network). Dots stand for spikes present only in one of the simulations.](image)

Results show that especially in the initial phase of the simulation the difference in activity is very high. \( nz \) accounts for the number of non-zero values in the difference set. The dynamics of the simulated behavior is thus rather regular once the external stimulus is suppressed. Our parallel implementations show thus an unaltered dynamics behavior.

5.3. Scalability Study

Figure 3 shows results obtained with the simulation of a neural microcircuit on the system b05.ici.

![Fig. 3. Difference in the neuronal activity between four simulations of the same scenario (simulation time 50 ms, 800 neurons in the network). Dots stand for spikes present only in one of the simulations.](image)
The main observation derived from these results is that the processing rate (spikes/seconds) of the simulator scales linearly with the number of threads up to 12. This corresponds to the number of physical cores of the system. Beyond this threshold, the spike rate growth is slower. The parallel efficiency obtained with this scenarios was about 0.57 for 12 threads and 0.38 for 24, when hyper threading was activated.

We next investigated the impact of the block wise processing of synapses and of neurons on the spike rate processing of the simulator. The update of the neurons was computed in parallel, in sets of neurons of different sizes. The computation of STDP implies the update of the plasticity for each synapse. Since the neurons included in our microcircuits have high connectivity, we organized the processing of the synapses in blocks, to foster compiler-based optimizations and a better usage of the cache memory. In Fig. 4 are presented the results obtained for the microcircuit with 5000 neurons and 2500 synapses per neuron. The synapse blocks had 10, 20, 50, 100, and 250 elements resp. Each of the four threads used in the analysis processed sets of neurons of size 2, 4, 10, 16, and 1250 resp.

The main observations derived from the results in Fig. 4 are:

- We have found no unique combination of block sizes for the processing of synapses and neurons. Best results have been obtained blocks of 250 synapses and 10 neurons per thread.
- The use of blocks of 100 synapses led, in average, to very good spike rates for all neuron block sizes.

The analysis of the other microcircuits listed in Table 1 exhibited a similar behavior.

5.4. Experience Report

We have used a range of compiler and code optimizations to speed up computations with our simulator. Furthermore, we investigated various thread scheduling and locking possibilities of OpenMP, leading to good scaling of the processing speed of the simulator. Nevertheless, we believe that in order to achieve better performance, more efficient data structures, hardware-aware and algorithm specific (such as for handling spike history), need to be elaborated.

Verification of the correctness of the sequential and multi-core implementations with unit tests is very convenient for a couple of reasons: a wide range of configuration and simulation scenarios can be described and computed as unit tests; the use cases of the simulation code get documented with concrete and working scenarios; the confidence in the quality of the simulation results is strengthened. Nevertheless, we found difficult to employ the unit tests for scenarios based on random activity, esp. choosing acceptable values for the accumulated ones.

Accommodating code specific to different multi-core and many-core technologies (OpenMP, OpenCL, CUDA) within the same simulator needs to be carefully handled. On the one hand, the performance of the program can be affected (statically or dynamically linking against the right libraries e.g.). On the other hand, the maintenance of the build and deployment scripts becomes more complex. The adoption of the continuous integration approach requires disciplined and systematic coding procedures. Nevertheless, we find this additional effort to be mandatory for developing good computational science codes.

6. Conclusions

In this paper we have investigated the dynamics of the OpenMP-based simulation of SRM and STDP. The methodological approach to code development followed here uses
extensively unit testing and continuous integration. The proposed parallel simulation strategies show a dynamic behavior unaltered by the use of multi-core specific technologies. The results of the scalability study show that the achieved parallel efficiency is good but it still needs to be improved. As such, a potential direction for further research is the design of data structure more suitable for the simulation of SRM and STDP models that maintain the unaltered dynamics behavior of results achieved so far. Another interesting continuation of this work would be the parallel dynamics investigation of event-driven simulation strategies.

Acknowledgements

This work was partly supported by the financing contract POSDRU/89/1.5/S/62557.

References