Scalable simulation of rate-coded and spiking neural networks on shared memory systems

Helge Ülo Dinkelbach (helge-uelo.dinkelbach@informatik.tu-chemnitz.de)

Julien Vitay (julien.vitay@informatik.tu-chemnitz.de)

Fred H. Hamker (fred.hamker@informatik.tu-chemnitz.de)
Chemnitz University of Technology - Artificial Intelligence Lab
Chemnitz 09107, Germany

Abstract

The size and complexity of the neural networks investigated in computational neuroscience are increasing, leading to a need for efficient neural simulation tools to support their development. Several neuro-simulators have been developed over the years by the community, all with different scopes (rate-coded, spiking, mean-field), target platforms (CPU, GPU, clusters) or modeling principles (fixed model library, code generation). We compare here the current version of the neuro-simulator ANNarchy against other state-of-the-art simulators on rate-coded and spiking benchmarks with a focus on their parallel performance.

Keywords: Rate-coded neurons, Spiking neurons, Parallel Computing

Introduction

Several simulators are available for the simulation of spiking networks, such as NEST (Gewaltig & Diesmann, 2007), Auryn (Zenke & Gerstner, 2014), Brian2 (Stimberg, Goodman, Benichoux, & Brette, 2014) and GeNN (Yavuz, Turner, & Nowotny, 2016). There are however only a few supporting rate-coded networks: ANNarchy (Vitay, Dinkelbach, & Hamker, 2015), NEST (Hahne et al., 2017) and recently Draculab (Verduzco-Flores & De Schutter, 2019). Several publications have highlighted the computational differences between the simulation of rate-coded and spiking neural networks (Brette et al., 2007; Brette, 2015; Blundell et al., 2018). These implementations can differ dependent on the chosen target platform, i.e. on CPUs or GPUs (Brette & Goodman, 2012). Some neuro-simulators such as NEST, ANNarchy, Brian2, and GeNN use code generation to bring these implementations under one interface, see (Blundell et al., 2018) for a recent review. Alternatively, specialized simulators with predefined neural models such as Auryn benefit from a high optimization level for specific operations, e.g. random number generation and vectorization (Zenke & Gerstner, 2014). To our knowledge, NEST and ANNarchy are the only simulators covering both rate-coded and spiking networks. In this article, we compare the parallel performance on shared-memory systems of NEST, ANNarchy, Brian2, Auryn and Brian2GeNN (Stimberg, Goodman, & Nowotny, 2018) on rate-coded and spiking benchmarks.

ANNarchy

ANNarchy is an equation-oriented neuro-simulator written in Python, able to simulate both rate-coded and spiking networks using code generation in C++ and CUDA. Its interface and main concepts are presented by Vitay et al. (2015). Since this publication, ANNarchy has been extended to allow the simulation of spiking networks on GPUs. The parallel implementation for rate-coded and spiking networks has been largely improved, especially through the use of ring buffers for delayed event-driven synaptic transmission.

Models for benchmarks

We investigate balanced recurrent networks using 1) a stochastic linear rate-coded neuron model and 2) a conductance-based integrate-and-fire spiking neuron model. In each case, the population consists of N neurons divided into 80% excitatory and 20% inhibitory neurons (Dale's law). The neurons are reciprocally and randomly interconnected with a fixed number of connections per neuron for the rate-coded benchmark and a fixed probability for the spiking one.

Linear rate-coded benchmark

The linear rate-coded benchmark is a reimplementation of the NEST model lin_rate_ipn as described by Hahne et al. (2017). The firing rate r_i of a neuron depending on its inputs r_j is described by the following stochastic differential equation (SDE):

$$\tau \frac{dr_i}{dt} + r_i = \sum_{j=1}^{C} w_{ij} r_j + \sqrt{\tau} dW \quad (1)$$

whereas C is the total number of incoming connections. Each neuron is connected to 10% of the other neurons. The connection weights w_{ij} are fixed, with excitatory weights set at $\frac{0.1}{\sqrt{N}}$ and inhibitory ones at $\frac{-0.5}{\sqrt{N}}$. The pre-synaptic firing rates r_j are delayed by 5 ms to simulate synaptic delays. dW represents a stochastic Wiener process which is implemented using a normal distribution: $dW = \sigma \cdot \mathcal{N}(\mu, 1)$ where $\mu = 0$, $\sigma = 5$ and $\tau = 10$ ms. The Euler-Maruyama numerical method is used to discretize the SDE with a step size of $dt = 0.1$ ms.

COBA benchmark

The COBA model proposed by Brette et al. (2007) is a benchmark for spiking networks based on the work of Vogels and Abbott (2005). The network consists of 4000 neurons (3200

excitatory and 800 inhibitory). The neuron model is described by an ordinary differential equation for the membrane potential v_i for each neuron:

$$\tau \frac{dv_i}{dt} = (E_L - v_i) + g_i^{exc}(E_{exc} - v_i) + g_i^{inh}(E_{inh} - v_i) + I \quad (2)$$

and the two conductance channels g_i^{exc} and g_i^{inh}:

$$\tau_{exc} \frac{dg_i^{exc}}{dt} = -g_i^{exc} \quad (3)$$

$$\tau_{inh} \frac{dg_i^{inh}}{dt} = -g_i^{inh} \quad (4)$$

All neurons are randomly connected to each other with a probability of 0.02. All parameters were set according to Brette et al. (2007) and Vogels and Abbott (2005).

Performance comparison

We compare the recent versions of NEST 2.16.0, ANNarchy 4.6.8, Brian2 2.2.2.1, Brian2GeNN 1.3.1 and Auryn 0.8.2. The collection of implementation codes used in this paper can be found online[1]. The simulations were performed on a dual socket system with Intel Xeon X5660 (6 cores with 2,8 GHz) and a NVIDIA Kepler K20m, running Linux Mint 19, g++ 7.4 and CUDA SDK 9.0.

COBA benchmark

Fig. 1 shows the normalized computation time in seconds as a function of the number of threads for NEST (blue line), ANNarchy (orange line), Auryn (green line) and Brian2 (red triangle). It also depicts a comparison of the GPU performance of ANNarchy (diamond) and Brian2GeNN (dot). We used Auryn to generate and store the connectivity matrices and loaded them for the other simulators to ensure the comparability of the networks. The simulation was performed for ten seconds of simulated biological time.

NEST is the only simulator in this comparison using a Runge-Kutta 4 method to solve the ODEs as the Euler implementation patch used by Zenke and Gerstner (2014) and Vitay et al. (2015) is not working with the recent version of NEST. One can assume that this different numerical method is responsible for a large fraction of the observed difference with the other simulators. As in our earlier benchmark (Vitay et al., 2015), the Auryn simulator performs best in this setup, which might be due to the vectorized evaluation of the neural equations. While the scalability of NEST and Auryn is almost linear, the scalability of ANNarchy is limited mainly because of two reasons: 1) only the update of the neural equations is parallelized with OpenMP, synaptic transmission being inefficiently parallelized when using arrays (NEST uses object-oriented representations of synapses) and 2) with increasing number of threads, inter-CPU communication for sharing the neural and synaptic variables will increase.

[1] www.github.com/hdinkelbach/codes_ccn_2019

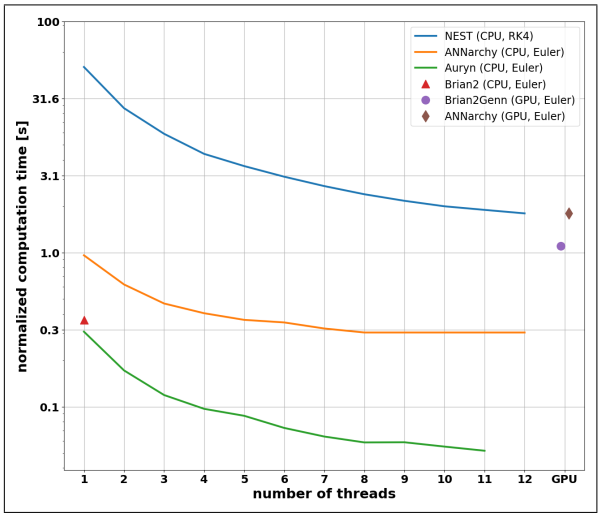

Figure 1: Simulation time (logarithmic scale) for the COBA benchmark as a function of the number of threads.

GPU implementations typically perform very well for large arrays and high computational load. While this can be easily ensured for the update of neural equations, the opposite applies for the event-driven transmission as only a few neurons are active in each time step. Consequently, the overhead for launching computation kernels on the device or data transfers from or to the device can be a strong performance limiter. The investigated network is rather small and relatively low active with an average activity of approximately 20 Hz, leading to a lower performance on GPUs, which is in line with findings by Yavuz et al. (2016) or Stimberg et al. (2018).

Linear rate-coded benchmark

Fig. 2 shows the single-threaded computational time in seconds as a function of the number of neurons using NEST (blue line), ANNarchy CPU (orange line) and ANNarchy GPU (brown line). The network was run for 100 ms of simulated biological time. NEST was used without activated waveform-relaxation technique to be comparable to ANNarchy. NEST and ANNarchy implementations using a single thread shows a similar scaling behavior depending on the network size. The GPU implementation shows a lower increase for small networks, what is an indicator for a high transfer overhead, the computational kernels being bound to the memory bandwidth rather the computation itself. This is most likely a consequence of uncoalesced memory access patterns due to the small number of connections per neuron, as we already previously observed (Dinkelbach, Vitay, Beuth, & Hamker, 2012).

We investigated also the scalability of the parallel implementations of ANNarchy and NEST using openMP. The results are shown in Fig. 3. For this benchmark, we selected the 4000 neurons configuration and tested the performance using 1 up to 12 threads. We observe a similar situation as in the COBA benchmark: NEST (blue) achieves a nearly lin-

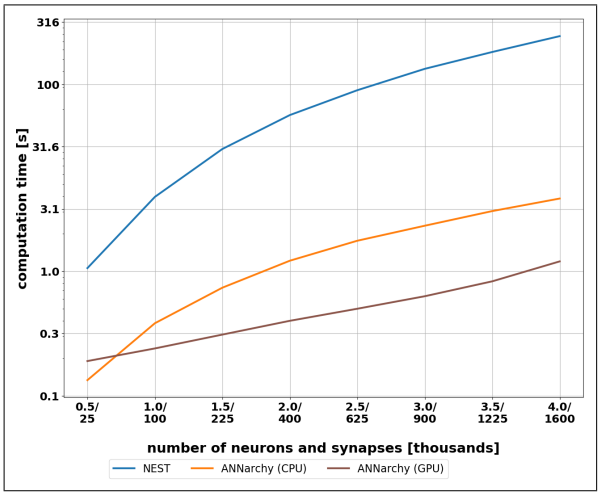

Figure 2: Simulation time (logarithmic scale) for the linear rate-coded benchmark as a function of network size. We compare NEST using a single thread (blue) and ANNarchy using either a single thread (orange) or single GPU (brown).

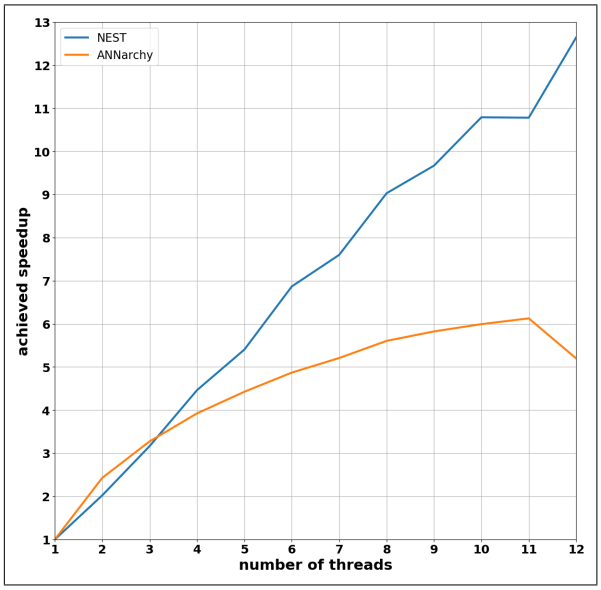

Figure 3: Strong scaling experiment using a linear rate-coded network with 4000 neurons for NEST and ANNarchy.

ear speedup while the speedup of ANNarchy (orange) scales sub-linear from 4 threads on. This might be due to false sharing issues in the openMP parallelization. As for the spiking networks, we assume that the inter-CPU communication for the shared arrays is a performance limiter in comparison to the object-oriented implementation of NEST. A second possible factor is that the random number generation is not parallelized in ANNarchy. For the investigated setup, ANNarchy remains faster than NEST for 12 threads, but one can expect that NEST will later outperform ANNarchy using more threads due to the better scalability of NEST.

Discussion

ANNarchy allows an easy definition of rate-coded and spiking models via an equation based interface to allow simulation on multi-core CPUs and GPUs. We presented a comparison of two benchmark-models using different state-of-the-art simulators. For spiking neural models, NEST and Auryn achieve a good scalability and clearly prove their suitability for shared memory systems. While ANNarchy achieves a good performance for small number of threads, the scalability is limited for higher numbers. This opens space for improvement especially on multi-core systems.

The differences obtained by the different simulators can be explained by two main factors: the evaluation of neural updates and the synaptic transmission. When the neural variables are stored in continuous arrays, their state updates can be vectorized, leading to an improved performance factor in the currently available multi-core CPUs. However, these arrays are shared among multiple cores or even CPUs, what potentially limits the possible speedup. Comparatively, object-oriented approaches as in NEST cannot benefit directly from vectorization but the objects are handled individually can reduce the inter-core/-CPU communication. A similar situation can be observed for synaptic transmission in combination with sparse connectivity. If pre- or post-synaptic neurons of adjacent synapses are not close to each other, there is a high probability that array-based structures will load more data than needed. Which then, in case of multiple threads, leads to an increased inter-core/-CPU communication. In contrast to object-oriented neurons where only the data of the corresponding neuron is loaded. We discussed this effect in our earlier work for synaptic transmission in rate-coded networks (Dinkelbach et al., 2012).

Another key factor concerns the data structures used for synaptic transmission. While in rate-coded networks it is straightforward to organize the connectivity matrices with one row per post-synaptic neuron (each weighted sum of inputs only needs to access the corresponding row, where weights are stored continuously in memory), spiking networks can be also implemented using one row per pre-synaptic neuron: spikes are sent to the efferent neurons. The choice of this structure can influence the performance as demonstrated by Stimberg et al. (2018).

ANNarchy was originally designed as a rate-coded simulator which was then extended to spiking networks, while NEST went the other way. We draw the same conclusion as Hahne et al. (2017): the two paradigms, rate-coded and spiking, require different techniques for communication and computation. This makes the implementation within a unified simulation environment a challenging task, where code generation might prove very useful.

Acknowledgments

This work was supported by Deutsche Forschungsgemeinschaft (DFG) in the projects "Auto-tuning for neural simulations on different parallel hardware" (DFG HA2630/9-1) and "Computational Connectomics" (DFG HA2630/11-1).

References

Blundell, I., Brette, R., Cleland, T. A., Close, T. G., Coca, D., Davison, A. P., ... Eppler, J. M. (2018). Code Generation in Computational Neuroscience: A Review of Tools and Techniques. *Frontiers in Neuroinformatics*, *12*. doi: 10.3389/fninf.2018.00068

Brette, R. (2015). Philosophy of the Spike: Rate-Based vs. Spike-Based Theories of the Brain. *Frontiers in Systems Neuroscience*, *9*, 1–14. doi: 10.3389/fnsys.2015.00151

Brette, R., & Goodman, D. F. M. (2012). Simulating spiking neural networks on GPU. *Network: Computation in Neural Systems*, *23*(4), 167–182. doi: 10.3109/0954898X.2012.730170

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., ... Destexhe, A. (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, *23*(3), 349–398. doi: 10.1007/s10827-007-0038-6

Dinkelbach, H. Ü., Vitay, J., Beuth, F., & Hamker, F. H. (2012). Comparison of GPU-and CPU-implementations of mean-firing rate neural networks on parallel hardware. *Network: Computation in Neural Systems*, *23*(4), 212–236. doi: 10.3109/0954898X.2012.739292

Gewaltig, M.-O., & Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, *2*(4), 1430.

Hahne, J., Dahmen, D., Schuecker, J., Frommer, A., Bolten, M., Helias, M., & Diesmann, M. (2017). Integration of Continuous-Time Dynamics in a Spiking Neural Network Simulator. *Frontiers in Neuroinformatics*, *11*. doi: 10.3389/fninf.2017.00034

Stimberg, M., Goodman, D. F. M., Benichoux, V., & Brette, R. (2014). Equation-oriented specification of neural models for simulations. *Frontiers in Neuroinformatics*, *8*, 6. doi: 10.3389/fninf.2014.00006

Stimberg, M., Goodman, D. F. M., & Nowotny, T. (2018). Brian2GeNN: a system for accelerating a large variety of spiking neural networks with graphics hardware. *bioRxiv*. doi: https://doi.org/10.1101/448050

Verduzco-Flores, S., & De Schutter, E. (2019). Draculab: A Python Simulator for Firing Rate Neural Networks With Delayed Adaptive Connections. *Frontiers in Neuroinformatics*, *13*, 1–15. doi: 10.3389/fninf.2019.00018

Vitay, J., Dinkelbach, H. Ü., & Hamker, F. H. (2015). ANNarchy: a code generation approach to neural simulations on parallel hardware. *Frontiers in Neuroinformatics*, *9*. doi: 10.3389/fninf.2015.00019

Vogels, T. P., & Abbott, L. F. (2005). Signal Propagation and Logic Gating in Networks of Integrate-and-Fire Neurons. *Journal of Neuroscience*, *25*(46), 10786–10795. doi: 10.1523/JNEUROSCI.3508-05.2005

Yavuz, E., Turner, J., & Nowotny, T. (2016). GeNN: a code generation framework for accelerated brain simulations. *Scientific reports*, *6*, 18854. doi: 10.1038/srep18854

Zenke, F., & Gerstner, W. (2014). Limits to high-speed simulations of spiking neural networks using general-purpose computers. *Frontiers in Neuroinformatics*, *8*. doi: 10.3389/fninf.2014.00076