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

Helge Ülo Dinkelbach, Julien Vitay and Fred H. Hamker

Department of Computer Science, Chemnitz University of Technology, Germany

Professur Künstliche Intelligenz

TECHNISCHE UNIVERSITÄT CHEMNITZ

Overview

We compare ANNarchy - 4.6.8 with current versions of other neural simulation tools:

- Auryn - 0.8.2
- NEST - 2.16.0
- Brian2 - 2.2.2.1
- Brian2GeNN - 1.3.1

We investigate the performance on non-plastic recurrent networks (80% excitatory, 20% inhibitory neurons) using:

- stochastic linear rate-coded neuron model with a fixed number of connections per neuron (10%) comparable to [1]
- conductance-based integrate-and-fire neuron and a fixed connection probability (2%) comparable to [2, 3]

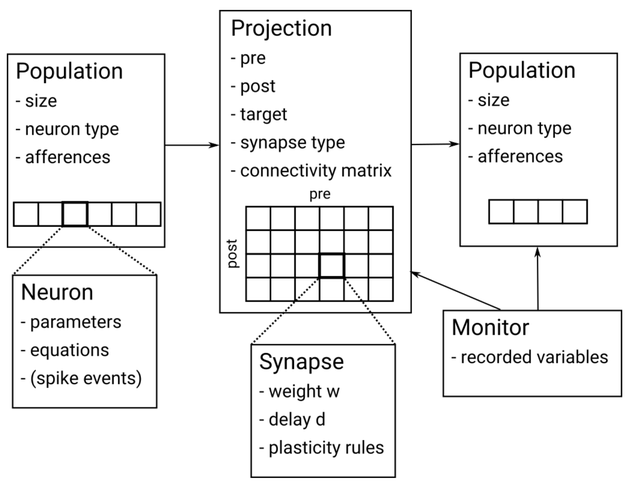
ANNarchy neural simulator [3]

ANNarchy is a Python based framework using code generation based on equation-like definitions of biologically-inspired neural networks:

- Easy definition of rate-coded and spiking models
- High degree of freedom for definition of activation function or learning mechanisms
- Different integration methods (Euler, exponential, Midpoint, Runge-Kutta 4 is planned)

```python
from ANNarchy import *
setup(dt=0.1, paradigm="cuda")

COBA = Neuron(
    parameters="""
        El = -60.0          : population
        Vr = -60.0          : population
        Erev_exc = 0.0      : population
        Erev_inh = -80.0    : population
        Vt = -50.0          : population
        tau = 20.0          : population
        tau_exc = 5.0       : population
        tau_inh = 10.0      : population
        I = 20.0            : population
    """,
    equations="""
        tau * dv/dt = (El - v) + g_exc * (Erev_exc - v)
                    + g_inh * (Erev_inh - v ) + I

        tau_exc * dg_exc/dt = - g_exc
        tau_inh * dg_inh/dt = - g_inh
    """,
    spike = "v > Vt",
    reset = "v = Vr",
    refractory = 5.0
)

P = Population(geometry=4000, neuron=COBA)
Pe = P[:3200]
Pi = P[3200:]
P.v = Normal(-55.0, 5.0)
P.g_exc = Normal(4.0, 1.5)
P.g_inh = Normal(20.0, 12.0)

Ce = Projection(pre=Pe, post=P, target='exc')
Ce.connect_fixed_number_pre( 75, weights=0.6)
Ci = Projection(pre=Pi, post=P, target='inh')
Ci.connect_fixed_number_pre( 15, weights=6.7)

compile()

simulate(1000.0, measure_time=True)
```

- Code generation allows to adjust both runtime configuration and used data structures



References

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

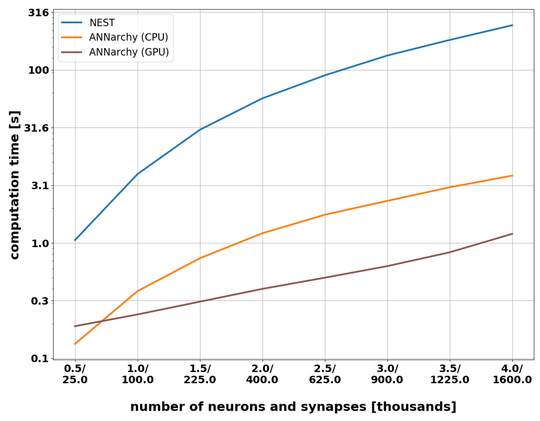
[2] Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., . . . Destexhe, A. (2007).Simulation of networks of spiking neurons: A review of tools and strategies. Journal of Computational Neuroscience, 23(3), 349–398. doi: 10.1007/s10827-007-0038-6

[3] Vitay J, Dinkelbach H. Ü. and Hamker F. H. (2015). ANNarchy: a code generation approach to neural simulations on parallel hardware. Frontiers in Neuroinformatics. 9:19, 10.3389/fninf.2015.00019

[4] Dinkelbach, H. Ü., Vitay, J., Beuth, F. and 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.

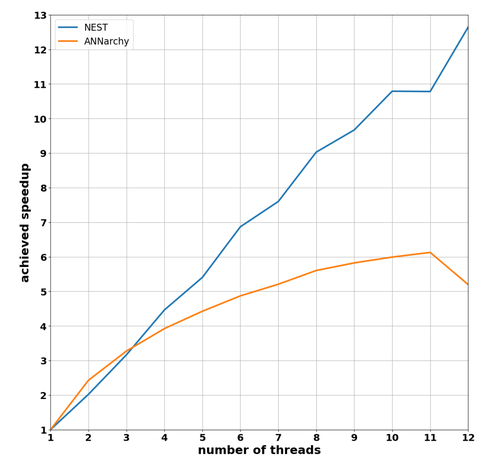
Linear rate-coded benchmark

Single-thread computation time as a function of network size



- ANNarchy is faster than NEST using single thread
- GPU implementation requires higher number of elements to be efficient, as shown in earlier experiments [4]

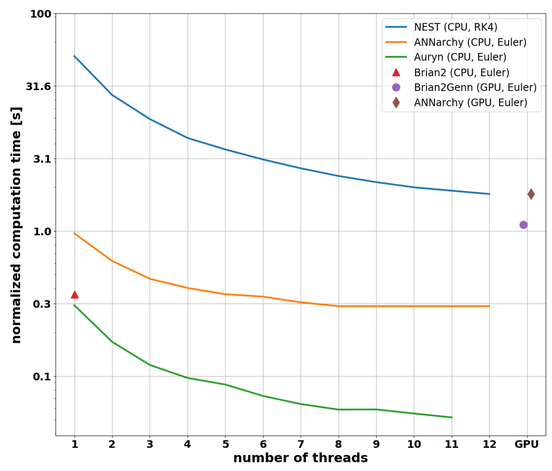
Scalability for 4,000 neurons and $1,6 \cdot 10^6$ synapses on CPUs



- NEST achieves higher scalability
- ANNarchy scales sub-linear for more than 4 threads
- Array-based implementations as in ANNarchy are more likely impaired by false sharing than object-oriented implementations as in NEST

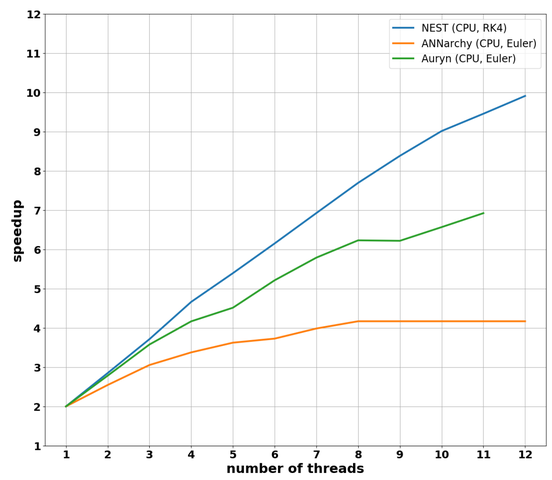
Spiking benchmark

Computation times for 4,000 neurons on CPUs and GPUs



- Auryn achieves the best results using single- and multi-core
- NEST has a higher computation time due to a more expensive numerical method
- The GPU implementations require larger networks to be efficient

Scalability for the CPU implementations



- NEST achieves the best scalability
- Scalability of ANNarchy is limited by connectivity representation
- Array-based approaches (Auryn and ANNarchy) achieve lower scalability compared to object-oriented implementation (NEST)

Conclusion

ANNarchy was originally designed as a rate-coded simulator which was then extended to spiking networks, while NEST went the other way. Other simulators like Auryn and Brian2 are focused on the development of spiking neural networks. We draw the same conclusion as [1]: 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.

Acknowledgement

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

SPP 2041 Computational Connectomics

DFG

helge-uelo.dinkelbach@informatik.tu-chemnitz.de