Multithreaded and Distributed Simulation of Large Biological Neuronal Networks

Jochen Martin Eppler[1,4], Hans Ekkehard Plesser[2], Abigail Morrison[3], Markus Diesmann[3,4] and Marc-Oliver Gewaltig[1,4]

eppler@biologie.uni-freiburg.de, hans.ekkehard.plesser@umb.no, abigail@brain.riken.jp, diesmann@brain.riken.jp, marc-oliver.gewaltig@honda-ri.de

1
Honda Research Institute Europe GmbH
Carl-Legien-Str. 30
63073 Offenbach/Main, Germany
http://www.honda-ri.de

2
Dept. of Mathematical Sciences and Technology
Norwegian University of Life Sciences
1432 Ås, Norway
http://www.umb.no

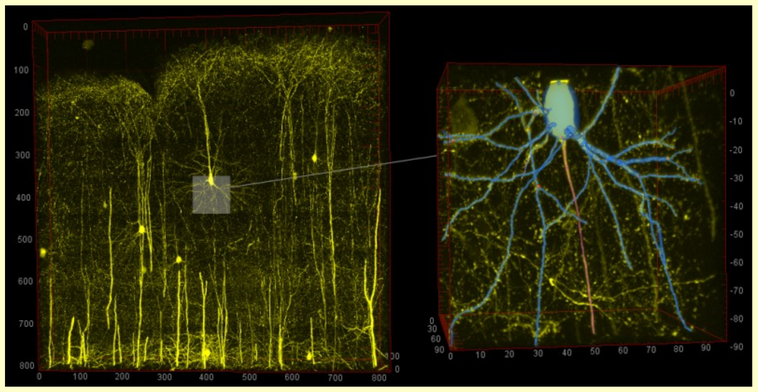
3
RIKEN Brain Science Institute
Wako City
351-0198 Saitama, Japan
http://www.brain.riken.jp

4
Bernstein Center for Computational Neuroscience
Hansastraße 9a
79104 Freiburg, Germany
http://www.bccn.uni-freiburg.de

- The Neural Simulation Tool NEST is a simulation environment for large networks of spiking neurons.
- It is optimized for networks with more than 10^5 neurons and 10^9 connections.
- The key challenges are to represent the network succinctly and to transmit events efficiently.
- NEST uses a hybrid strategy with MPI across cluster nodes and threads on each computer.
- We demonstrate the performance of NEST, comparing different communication algorithms.

Biological Neuronal Networks

One cubic millimeter of cortex contains approximately 10^5 neurons with roughly 10^4 connections each (see figure below). Computational Neuroscience tries to model these networks to understand how macroscopic functions of the brain relate to its neuronal substrate. Neurons communicate by electrical pulses (spikes) over fibers (axons, dendrites) and chemical junctions (synapses). Thus, models of biological neurons typically contain several non-linear differential equations that must be evaluated with high precision. We investigate the temporal dynamics of the spiking (pulse) activity in large networks. This should not be confused with artificial neural networks that focus on learning or function approximation with few (several hundred) idealized threshold units.



Cortical neurons in a fixed, 100 µm-thick brain slice from a transgenic mouse with sparse neuronal GFP expression. Left: Mosaic of 81 high-resolution image stacks acquired on a 2-photon laser-scanning microscope, part of a larger data set spanning several consecutive brain slices. Right: Individual stack from this data set. Image: http://www.advanced-imaging-center.org/aic_partnerView.cfm?KeyID=49.

Modeling in NEST

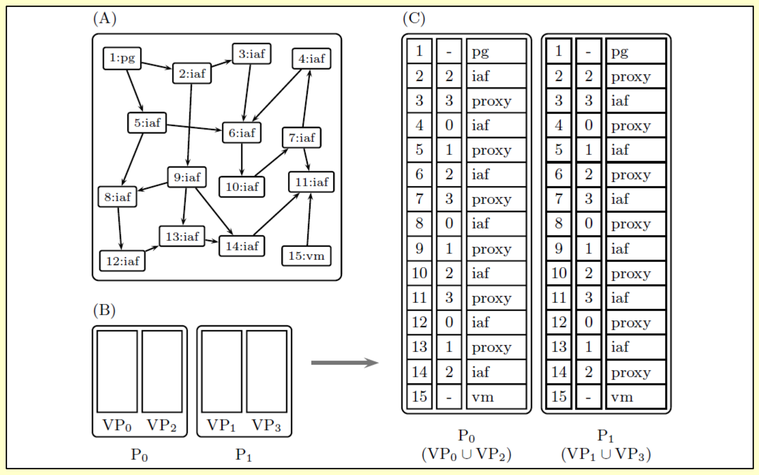
The Neural Simulation Tool NEST [1] is implemented in C++. It can be used either with its own simulation language interpreter (SLI) or by using the bindings for the Python programming language. Simulations in NEST can be seen as the computational counterpart of electro-physiological experiments in Neuroscience. Building a simulation in NEST works as follows:

- Neuronal networks are assembled from a pre-defined set of model neurons and devices to measure spikes and potentials or to stimulate neurons.
- The basic elements can be structured in sub-networks.
- Neurons of different types can be combined in the same network.
- Different synapse types can be used to connect neurons with either static or dynamic parameters (e.g. for learning and adaptation).
- The synapse type determines how an incoming spike affects a neuron.
- After building the network, it can be simulated for a user-defined period of biological time.

The source code for NEST is available under an open source license on the homepage of the NEST Initiative at http://www.nest-initiative.org.

Network Representation in NEST

Conceptually, NEST represents a network as list of nodes, which are either neurons, devices, or sub-networks. Each node has a unique ID and is permanently assigned to one of N_{VP} *virtual processes*. A virtual process (VP) is a POSIX thread in one of N_{MPI} MPI processes. Each process contains N_{Trd} threads, resulting in a total of $N_{VP} = N_{Trd} \cdot N_{MPI}$ virtual processes. Each VP has a complete node list. It contains the nodes that are assigned to this VP and lightweight proxies representing all other nodes. To save memory, the node lists of all VPs in a single process are collapsed into a single list. The following figure illustrates this for a network with 15 nodes, distributed over two processes with two threads each (4 VPs):



(A) The network as directed graph; pg = Poissonian spike generator, iaf = integrate-and-fire neuron, vm = voltmeter. (B) A sketch of four VPs distributed in a modulo fashion onto four processes. (C) The collapsed representation of the network.

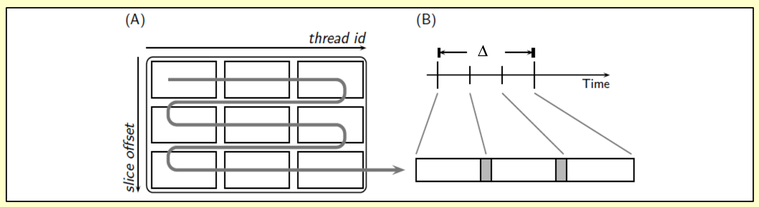
The network connections dominate the memory consumption of a simulation. Thus NEST distributes the connections across the VPs. Each VP only stores those connections that reach nodes this VP.

Scheduling and Communication

NEST evaluates the network model on a time grid $t_i = t \cdot \Delta$, with Δ the shortest transmission delay in the network. A global state transfer function U propagates the system from one state S_i to the next, such that $S_{t+\Delta} \leftarrow U(S_t)$. Starting with t=0, the network state is driven by the following algorithm, :

while $t < T_{stop}$ **do**
  parallel on all *VP* **do**
    deliver all events due
    call $U(S_t)$ for all nodes
  end parallel
  exchange events between VPs
  increment network time: $t \leftarrow t+\Delta$
end while

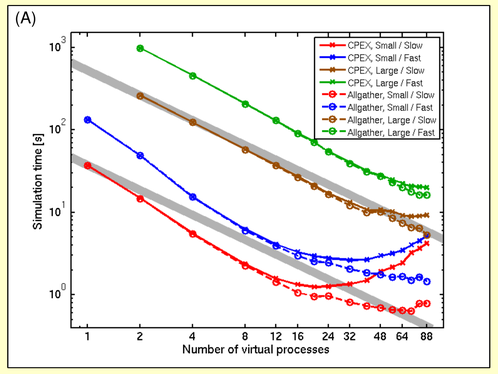
During $U(S_t)$, nodes emit spikes that must be sent to their targets with a delay that depends on the connection. The events are buffered in a thread-safe data-structure (fig. A) until they are delivered. Instead of transmitting the complete event to each target process, it suffices to transmit only the IDs of those nodes that emitted a spike. Each target process then reconstructs the events with their time-stamps, using markers in the transmitted data (fig. B). The reconstructed events are then delivered in parallel by each process.



Performance and Scaling

To test the performance of our algorithms and data structures, we simulated a balanced random network [2] with two sizes (small: 12500 neurons, 15.6 mill. connections; large: 125000 neurons, 156 mill. connections) and two activity levels (slow: 13 Hz; fast: 53 Hz).

For all combinations of network size and activity level, we measured the run-time of a simulation with different numbers of threads and different communication strategies (CPEX [3] vs. MPI_Allgather). The simulations were run on a cluster of 24 Sun Fire X4100 computers, with 2 AMD DualCore Opteron 280 processors (at 2.4 GHz) and 2 GB of RAM per core. All simulations used Scali MPI 4.4.1 over Mellanox Infiniband hardware.

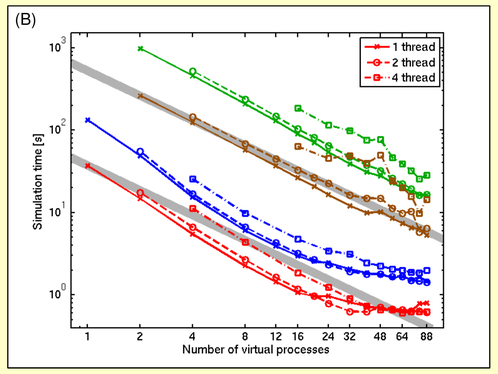


CPEX vs. MPI_Allgather (one thread per process)

Gray lines are linear speed-up.

With increasing N_{VP}, the computational load on each VP decreases, while the communication between them increases. Thus, for large N_{VP}, the communication dominates the run-time of the simulation.

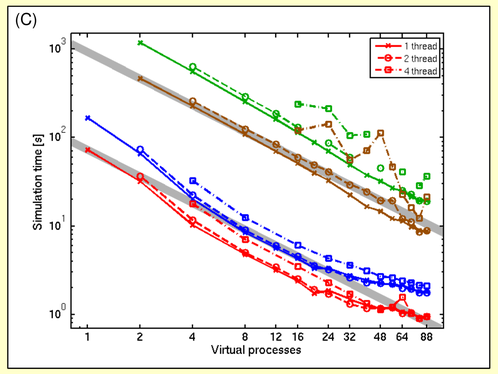
For $N_{VP} > 16$, MPI_Allgather is superior to CPEX.



Threads vs. Processes (low computational load)

Color code as in panel A.

The performance degrades with increasing number of threads, because NEST allocates all memory with its main thread. This is inefficient on NUMA architectures if multiple processes access memory concurrently. To solve this, NEST should allocate memory with the thread that is going to use it.



Threads vs. processes (high computational load)

Color code as in panel (A)

In this simulation, the neurons are driven by random spikes instead of static currents. This leads to higher computational load. As indicated in panel A, scaling is better if the run-time is not dominated by communication

Note that some points are missing due to technical problems on the cluster.

We also compared Scali MPI to MPICH 1.2.7 and OpenMPI 1.2.3 (not shown). The results suggest that the MPI implementation has only little effect on the performance of NEST.

References

[1] Gewaltig, M.-O. and Diesmann, M.: http://scholarpedia.org/article/NEST_(Neural_Simulation_Tool). Scholarpedia 2007.

[2] N. Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. J. Comput. Neurosci. 8 (3), 183–208, 2000.

[3] Tam, A. and Wang, C.: Efficient scheduling of complete exchange on clusters. In: 13th International Conference on Parallel and Distributed Computing Systems (PDCS 2000), Las Vegas, 2000.

Compute facilities kindly provided by the Bernstein Center for Computational Neuroscience Freiburg.

Partially funded by DAAD 313-PPP-N4-lk, EU Grant 15879 (FACETS), and BMBF Grant 01GQ0420 to the Bernstein Center for Computational Neuroscience Freiburg.