Performance Model for Large-Scale Neural Simulations with NEST

Wolfram Schenck[1], Andrew V. Adinetz[2], Yury V. Zaytsev[1], Dirk Pleiter[2], Abigail Morrison[1]

[1] SimLab Neuroscience - Bernstein Facility for Simulation and Database Technology, Institute for Advanced Simulation, Jülich Aachen Research Alliance, Forschungszentrum Jülich, 52425 Jülich, Germany
[2] Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich, 52425 Jülich, Germany
Mail contact: {w.schenck,a.adinets,y.zaytsev,d.pleiter, a.morrison}@fz-juelich.de

- NEST development led by the **"NEST Initiative"** (non-profit organization with international members)
- Best suited for simulations which focus on the dynamics, size, and structure of neural systems
- Suitable both for small-scale (single notebook) and large-scale (supercomputer) simulations
- **Technology:** Implemented in C++, hybrid parallelism (MPI+OpenMP), simulations defined in SLI or Python
- A typical NEST simulation runs in two stages: First the network is wired up ("build stage"), second the dynamics of the whole network is simulated ("simulation stage")

Goals of our Study

Development of a performance model for the simulation stage of NEST

- Collect measurements of NEST runtime performance
- Fit theoretical model to this data

→ Model explains computation time t_{SIM} required by the simulation stage

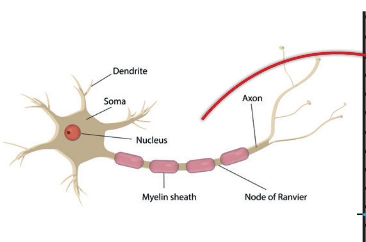
Why Performance Modeling?

- A performance model allows predictions for any simulation size
- A performance model helps to identify spots in the code which require algorithmic improvements

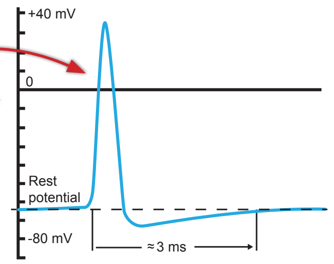
→ An in-depth performance model can guide algorithmic work to strongly improve application performance in various usage scenarios (see Hoefler et al., 2011)

nest:: Neural Simulation Tool

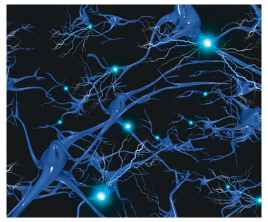
Simulator for large heterogeneous networks of spiking point neurons (and neurons with few compartments) (Gewaltig & Diesmann, 2007)



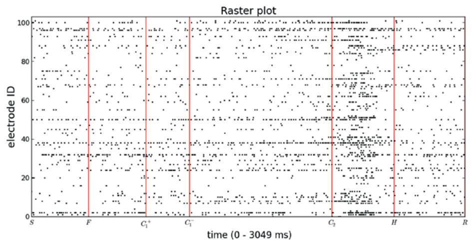
Neuron
(Figure: fotolia.com, © joshya)



Spike



Biological Neural Network
(artistic depiction,
Figure: fotolia.com, © rolffimages)



Spike Raster Plot
(Figure: INM-6, Forschungszentrum Jülich)

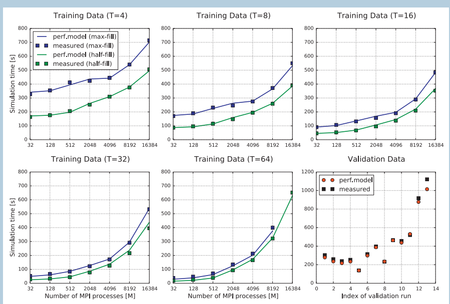
Experimental Design

- **Acquisition of the training data:** Systematic variation of the independent variables M, T, and N_M in a (nearly) full factorial design:
  - $M \in \{32,128,512,2048,4096,8192,16384\}$
  - $T \in \{4,8,16,32,64\}$
  - $N_M \in$ {half-fill, full-fill}

→ Simulation run over 500 ms of *biological time* for each parameter combination, measuring the required simulation time t_{SIM}

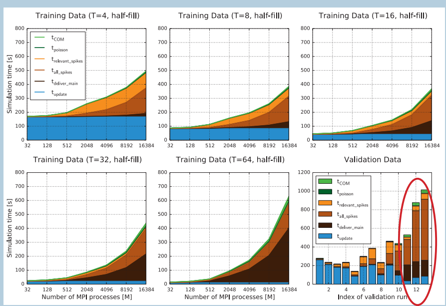
- **Hardware:** Simulations carried out on the **supercomputer JUQUEEN** at Forschungszentrum Jülich, Germany (28672 compute nodes, each equipped with a 16-core PowerPC-A2 CPU and 4-fold SMT [64 hardware threads] and 16 GB of RAM)
- **Model fitting:** Parameters p0,...,p4 of the performance model determined by fitting the estimated values $\hat{t}_{SIM}$ to the measured values t_{SIM}

Comparison of the Measured and Predicted Simulation Times

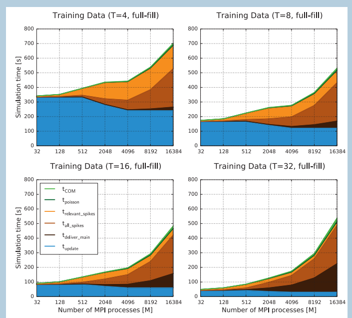


Additional validation runs for varying $M/T/N_M$ combinations (max. relative prediction error: 10 %)

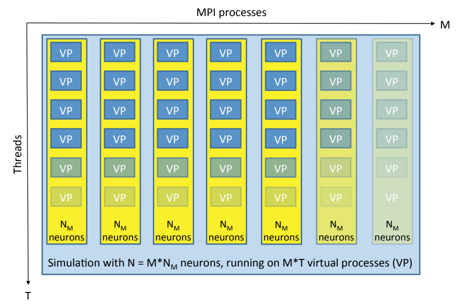
Estimated Contributions to the Simulation Time



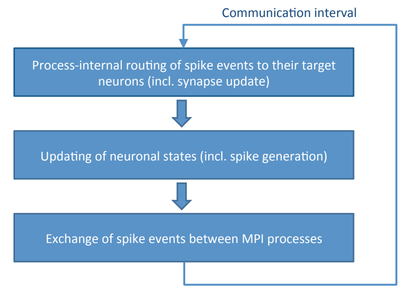
Validation runs on the whole JUQUEEN supercomputer (M=28672 MPI processes)



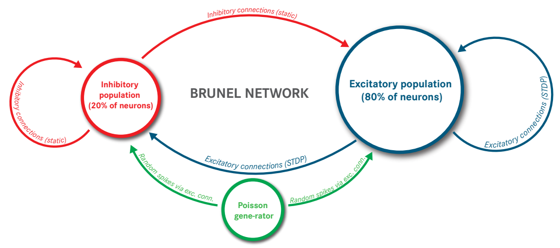
Theoretical Model



Parallel Execution of a NEST Simulation



Main Simulation Loop of NEST



→ Quick convergence to an asynchronous irregular state with mean spike firing rate F per neuron

Random Balanced Network Used in Our Study
(see Morrison et al., 2007)

- Complexities and simulation time components which were identified in the NEST source code:
  - Update of the neuron and synapse states: $t_{update} = p_0 \cdot \frac{N}{MT}$
  - Spike routing:
    - Main loop: $t_{deliver\_main} = p_1 \cdot MT$
    - Checking every spike event for its relevance for the virtual process: $t_{all\_spikes} = p_2 \cdot FN$
    - Processing relevant spikes within each virtual process: $t_{relevant\_spikes} = p_3 \cdot \left(1 - \exp\left(-\frac{K}{MT}\right)\right) FN$
  - Generating random external input (spikes) to neural network: $t_{poisson} = p_4 \cdot \frac{N}{M}$
  - MPI communication: $t_{COM} = p_5 \cdot SM$
  - **Main equation of the performance model** for the prediction of t_{SIM}:

$$\hat{t}_{SIM} = t_{update} + t_{deliver\_main} + t_{all\_spikes} + t_{relevant\_spikes} + t_{poisson} + t_{COM}$$
$$= p_0 \frac{N}{MT} + p_1 MT + p_2 FN + p_3 \left(1 - \exp\left(-\frac{K}{MT}\right)\right) FN + p_4 \frac{N}{M} + p_5 SM$$

$p_0, \ldots, p_4$: Free model parameters (subject to data fitting)
p_5 : Set to a fixed value determined from MPI communication benchmarks

Independent Variables

M	Number of MPI processes
T	Number of threads per process
N_M	Memory fill factor (specifies number of neurons per process)
N	Overall number of neurons (**$N = N_M * M$**)
K	Overall fan-in per neuron (fixed value: **K = 11,250**)

Model Parameters Depending on Network Dynamics

F	Mean spike frequency per neuron (ca. 7 Hz in all our simulation runs)
S	Size of the MPI send buffer per process

Results

- Coefficient of determination on the training data after model fitting: $R^2 = 99.6$ %
- Coefficient of determination on an independent data set with various values for $M/T/N_M$ (called "validation runs"): $R^2 = 98.4$ %

→ For small simulations (small M) and small thread numbers (T), the time for neuron and synapse update dominates

→ For large simulations and large thread numbers, the simulation times increase, which is mainly caused by the process-internal spike routing

Conclusions

- **General:**
  - Main complexities derived from NEST source code can explain runtime behavior
  - Successful extrapolation to new data
- **For further NEST development:**
  - For small simulations (e.g., on workstations or small compute servers): Optimize the update steps in NEST
  - For large simulations on supercomputers: Modify process-internal spike routing considerably

Acknowledgements

This work was funded by the Helmholtz Association through the Portfolio Theme "Supercomputing and Modeling for the Human Brain" and supported by the VSR computation time grant JINB33 on the supercomputer JUQUEEN at the Jülich Supercomputing Centre. We warmly thank Markus Diesmann (INM-6, Foschungszentrum Jülich) for his general support, Maximilian Schmidt (INM-6, Foschungszentrum Jülich) for the validation data, and Nadine Daivandy (JSC, Foschungszentrum Jülich) for the graphical layout and design of this poster.

References

M. O. Gewaltig & M. Diesmann (2007), NEST (Neural Simulation Tool), Scholarpedia 2(4): 1430

T. Hoefler, W. Gropp, W. Kramer & M. Snir (2011), Performance Modeling for Systematic Performance Tuning, State of the Practice Reports (SC 11): 6:1-6:12

A. Morrison, A. Aertsen & M. Diesmann (2007), Spike-Timing Dependent Plasticity in Balanced Random Networks, Neural Computation 19(6): 1437-1467

S. Kunkel, T. C. Potjans, J. M. Eppler, H. E. Plesser, A. Morrison & M. Diesmann (2012), Meeting the memory challenges of brain-scale network simulation, Frontiers in Neuroinformatics 5: article 35