2 Neuroscience with NEST, Arbor and Elephant

Hans Ekkehard Plesser, Susanne Kunkel, Håkon Mørk
Norges miljø- og biovitenskapelige universitet, NMBU, Norway

hans.ekkehard.plesser@nmbu.no

2.1 Introduction

The long-term goal of the neuroscience work in DEEP-EST is to provide an optimised setup for the integrated simulation and analysis of large-scale brain activity[21]. Such in situ analysis is essential to facilitate the interactive investigations of brain dynamics, where scientists can observe network activity while a simulation is running and interact with it to ensure that dynamics stay within relevant regimes. In DEEP-EST, our focus was on simulations of functional models of brain structure using the NEST simulator[22] combined with two types of in situ analysis: computation of electrical local field potentials using the Arbor[23] and HybridLFPy packages[24] on the one side, and statistical analysis of spike activity using the Elephant package[25] on the other.

2.2 Application structure

2.2.1 NEST

NEST is a simulation code for the investigation of the dynamics of brain-scale neuronal network models, as for example the recently published multi-area model[26]. NEST operates on the level of resolution of neurons and synapses, where neurons are brain cells connected to each other by synapses.

The simulator considers brain tissue as an abstract assembly of nodes (neurons) and connections (synapses) or, in other words, a directed graph. The neurons in these simulations are point neurons, i.e. the state of a node changes according to a set of

[21] Suarez, E. et al. (2021), „Modular Supercomputing for Neuroscience“, *Lecture Notes in Computer Science,* 2019 BrainComp Conference, Cetraro, Italy Springer International Publishing, 10.1007/978-3-030-82427-3_5

[22] http://www.nest-simulator.org/

[23] Akar, NA (2018) arXiv:1901.07454 [q-bio.NC]

[24] Hagen E et al. (2016) *Cerebral Cortex*, 26(12) pp. 4461–4496.

[25] http://elephant.readthedocs.io/

[26] Schmidt M et al. (2018) Brain Struct Funct 223: 1409.

ordinary differential equations (ODE), without taking into account the complete morphology of the cell.

The interaction between nodes is mediated by stereotyped events in the form of delayed delta pulses. These so-called action potentials (or spikes) are emitted by the nodes (neuronal activity) and propagated along the connections. The interaction strength (synaptic weight) can either be static or dynamic (synaptic plasticity) and depends on the activity of the two neurons joined by the connection.

NEST does not implement a specific network model but provides the user with a range of neuron and synapse models and efficient routines to connect them to complex networks with on the order of ten thousand incoming and outgoing connections for each neuron. Concrete network models and the corresponding simulation experiments are specified by model description scripts. These scripts are written either in NEST's built-in simulation language SLI (based on PostScript) or using the Cython-based Python interface PyNEST[27,28], with PyNEST being the default interface.

A published example of a large-scale network model is the multi-area model[26], which was relevant also in the context of the DEEP-EST project. It is the first multi-scale model of vision related brain areas and comprises approximately 4 million neurons and 6000 incoming synapses per neuron, where neurons emit on average 14.6 spikes/s. Each individual area is represented by a modified version of the Potjans-Diesmann model[29], a microcircuit model corresponding to a cortical network under a surface of 1 mm^2. The microcircuits representing the areas differ in neuron numbers and connection probabilities. The minimal synaptic transmission delay in the network is 0.1 ms biological time, i.e., the time simulated in the biological system. This requires frequent MPI communication of spikes (every 0.1 ms biological time). In terms of wallclock time, MPI communication occurs at approximately 10–30 ms intervals, depending on the activity level in the neuronal network. Due to long transients in the network dynamics the model needs to be simulated for 100 s biological time.

The NEST code base is open source and under continuous development in order to enable the investigation of novel models and theories in Computational Neuroscience on the one hand, and to meet the requirements of new computer hardware on the other hand. Since release 2.16, the NEST 5th generation simulation kernel (5G)[30] is included, which achieves excellent scaling with respect to memory usage and good scaling with respect to runtime on the largest supercomputers currently available for academic

[27] Eppler, JM et al. (2008) Front. Neuroinform. 2:12.

[28] Zaytsev YV and Morrison A (2014) Front. Neuroinform. 8:23.

[29] Potjans TC and Diesmann M (2014) Cereb. Cortex 24, 785–806.

[30] Jordan J et al. (2018) Front. Neuroinform. 12:2.

research. The key step from the previous kernel used in NEST releases 2.6.0–2.14.0 to the 5G kernel is a new connectivity representation and spike exchange scheme using directed communication based on MPI_Alltoall().

2.2.2 Arbor/HybridLFPy

Arbor simulates compartmental neuron models. This means that the spatial structure of each neuron is represented as a spherical cell body (soma), to which an arbitrary number of dendritic trees are attached. Each dendritic tree consists of segments, i.e. tubes or cables, of a given length and radius; in the simulation, each segment is represented by a configurable number of compartments. Each segment is either connected to one other segment at each of its ends (linear cable) or to several segments at its far end (branching point; far end: end pointing away from the soma). Electric currents flow along the cables formed by the dendritic tree. This current flow is described by ordinary differential equations, with one set of equations for each compartment, coupled to neighbouring compartments. The main task of Arbor is to solve the resulting system of ODEs; this task is highly amenable to vectorisation. In addition, Arbor also transmits spikes between neurons via synapses; this mechanism is of lesser importance for our purposes because HybridLFPy is based on simulating the dynamics of disconnected compartmental neurons based on spike input generated by NEST.

HybridLFPy computes mesoscopic electrical brain signals, called local field potentials (LFPs) based on the network dynamics simulated using NEST. Specifically, spike trains generated by neurons in a NEST simulation, using highly connected point neurons are fed into detailed models of unconnected neurons simulated using Arbor to compute the electrical currents passing through the cell membrane at different locations. From these currents, HybridLFPy then computes the LFP at different locations in a piece of brain tissue using electrostatic principles.

2.2.3 Elephant (ASSET)

Elephant is a pure Python library for the statistical analysis of spike activity of neurons. It can be installed using standard Python distribution tools. Elephant implements a wide and growing range of analysis methods. We focus mainly on the calculation of cross-correlations between spike trains and the detection of repeated patterns of spike activity across groups of neurons, so-called synfire chains.

Cross-correlations are detected using standard approaches, either implemented directly in Python or using NumPy convolution algorithms. Except for possible thread-parallelisation provided by the NumPy convolution implementation, cross-correlation algorithms are purely serial at present.

Detection of synfire chains uses the ASSET algorithm[31] in an optimised version[32], replacing the non-optimised version currently included in the release version of Elephant. The optimised algorithm uses MPI4Py for parallelisation.

2.3 Application mapping

Traditionally, NEST simulations have two distinct phases: a network construction (build) phase and a simulation phase. The key part of the build phase is the construction of network connectivity, i.e., building in largely random order a hierarchical data structure representing connections between neurons; each connection is represented only on the thread managing the connection's target neuron.

During the simulation phase, differential equations for the individual neurons are updated and spikes emitted according to a threshold criterion. Information on emitted spikes is exchanged between MPI processes and threads in steps of the minimal synaptic delay in the network, which is the maximum interval permitted by causality. Spikes are delivered to target neurons in parallel, each virtual process being responsible for delivery to the set of neurons it manages. This delivery process entails essentially random accesses to the connectivity data structure.

For the fifth generation (5G) kernel, we distinguish a third phase, called initialization phase, which comprises all necessary initialization processes at the beginning of a NEST simulation before the actual simulation takes place. In the NEST 5G kernel (NEST release 2.16), connectivity information, which is available only on the postsynaptic side after the build phase, needs to be transferred to the presynaptic side in order to enable directed communication of spikes during simulation. The transfer of connectivity data involves at least one round of MPI_Alltoall() communication, which makes the initialization phase a non-negligible component.

In the benchmarks hpc_benchmark.sli and hpc_mam_benchmark.sli, build phase and initialization phase take up a significant amount of the total runtime as the neuronal networks are simulated only for one second of biological time. In simulations of the multi-area model, build phase and initialization phase require only a small fraction of the total runtime as the network is simulated for 100 s of biological time.

To enable the interaction of NEST with Arbor/HybridLFPy (see Figure 2.1), a small fraction of the connectivity details of the multi-area network, which is available after the build phase of NEST, needs to be communicated, where HybridLFPy maps the connectivity to the detailed neuron models.

[31] Torre E et al. (2016) PLoS Comput Biol 12(7): e1004939.

[32] Canova C et al. (2017) ASSET for JULIA: executing massive parallel spike correlation analysis on a KNL cluster. Poster presented at HBP Summit 2017.

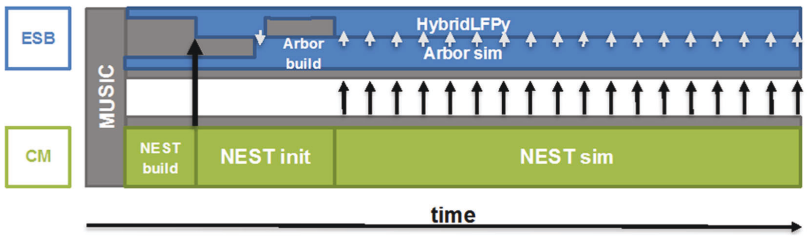

Figure 2.1: Schematic workflow of NEST and Arbor/HybridLFPY in the MSA

During the simulation phase NEST needs to communicate spikes from a fraction of the neurons of the multi-area model to Arbor or Elephant. Communication takes place frequently and is coordinated by the MUSIC library (see Figure 2.1 and Figure 2.2). We estimate that the total amount of data that needs to be communicated from CM to ESB or DAM in each communication round is negligible (about 1 kB if we assume communication every 0.1 ms of simulated time).

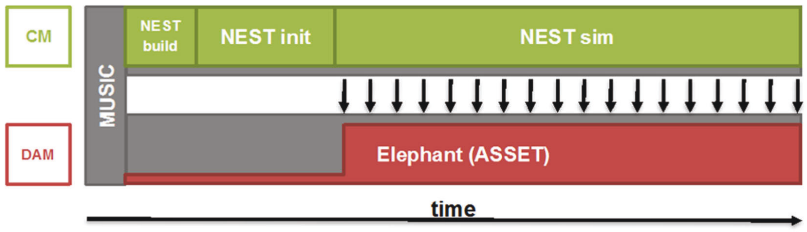

Figure 2.2: Schematic workflow of NEST and Elephant (ASSET) in the MSA

NEST (on CM) and Arbor/HybridLFPy (on ESB) start to run at the same time. While NEST constructs neurons and connections, Arbor instantiates neuron models. After the build phase of NEST, detailed connectivity information about the multi-area network is available. HybridLFPy requires part of this connectivity data in order to map the incoming connections of selected point-neurons simulated in NEST to their compartmental counterparts simulated in Arbor. Based on that, Arbor can build connections to the neuronal compartments.

After the communication of connectivity data from CM to ESB, NEST enters the initialization phase, which does not necessarily end at the same time as the Arbor build phase. The simulation phases of both NEST and Arbor follow, where Arbor relies on frequent spike input from NEST.

During the simultaneous simulation phases of NEST and Arbor, full network activity of the multi-area model is simulated in NEST and spikes from the previously selected fraction of the network are frequently communicated to Arbor running on the ESB using the MPI-based MUSIC library. The spatially detailed (compartmental) neuron models simulated in Arbor consume the spikes according to the mapping created by HybridLFPy.

Locally on the ESB HybridLFPy requires frequent information about ionic currents into and out of the neuronal compartments simulated in Arbor in order to predict the LFP signals and their development over time.

Elephant is fed with spikes from selected populations of the multi-area model using the MUSIC library to coordinate MPI communication (see Figure 2.2). Therefore, NEST (on CM) and the Python script that applies the necessary Elephant functions to the incoming spike trains (on DAM) start to run at the same time but the Python script needs to wait with the analysis until NEST reaches the simulation phase and produces spikes.

We expect that in simulations of the multi-area model this initial idle time of Elephant will be irrelevant as neither build nor initialization time, but the actual simulation time, dominates the total runtime of NEST.

The simulation of the multi-area model with NEST is run on the CM using a hybrid parallelisation scheme combining MPI and OpenMP threads. CM is optimal for NEST, because NEST's irregular memory access patterns perform optimally on CPUs with large, low-latency RAM and because NEST does not benefit from vectorisation.

Selected neurons of the multi-area network are simulated in greater detail with Arbor running on the ESB, because Arbor requires considerably more compute power relative to memory, since Arbor simulation does not require full network connectivity information. Arbor benefits significantly from vectorisation using AVX2, AVX512, and GPGPUs; it uses hybrid parallelisation combining MPI and C++11 threads or Intel TBB.

Analysis of spike trains recorded from selected populations of the multi-area model is carried out by Elephant, which runs on the DAM.

2.4 Porting experience

Porting the code to the different DEEP-EST modules has been straightforward for all three applications (NEST to the CM, Arbor to the ESB, and Elephant to the DAM). There were, in particular, no issues with porting Arbor to the ESB as GPU support was already in place.

To use the workflows described above, we needed to implement communication back ends in Arbor and NEST. We had suggested earlier to use the MUSIC library for the communication within the NEST-Arbor coupling. More careful analysis of the interaction between NEST and MUSIC as part of this project revealed that use of MUSIC for MPI communication between NEST and Arbor would impose frequent synchronisation of threads in MPI-OpenMP hybrid NEST simulations. To avoid this, we decided to implement NEST-Arbor coupling directly via MPI instead of using MUSIC as an intermediary. The mapping of neuron identities between NEST and Arbor, which MUSIC would have provided, was ensured through proper simulator scripting.

NEST, Arbor and Elephant could be installed and run out-of-the box using standard compiler and build tools available after we had familiarized ourselves with the software environment on the DEEP-EST system, with an effort off less than 0.5 Person Month (PM). Basic interfacing NEST and Elephant via MUSIC including minor bug fixes took also about 0.5 PM. The NEST-Arbor interface was implemented in collaboration with the Arbor development team; NMBU contributed roughly half of the effort (3 PM).

2.5 Scalability

Both NEST and Arbor have already been shown to scale well on modern supercomputers[33],[34] (Figure 2.3 and Figure 2.4). With the 5th generation simulation kernel, the communication scheme for the exchange of spikes between MPI processes was changed from `Allgather()` to `Alltoall()`, allowing each MPI process to send spikes only to the MPI processes that host the targets. To this end, the connection infrastructure of NEST was redesigned. Arbor has been developed considering support for GPUs and explicit vectorization from the very outset.

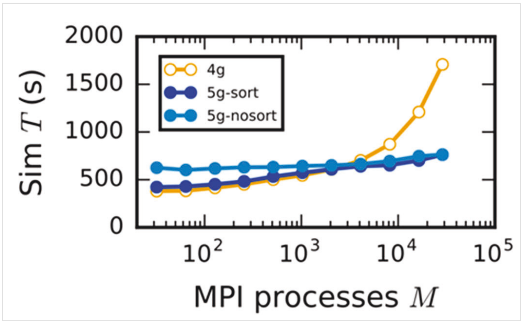

Figure 2.3: Simulation time for NEST running the HPC benchmark[33] on JUQUEEN; shown for previous kernel (4g) and new kernel with optimizations for small-scale to medium-scale regime (5g-sort) and without the optimizations (5g-nosort). Adapted from Figure 7C in[33])

[33] Jordan, J. et al. (2018) doi:10.3389/fninf.2018.00002

[34] Akar, N. A. et al (2019) doi: 10.1109/EMPDP.2019.8671560

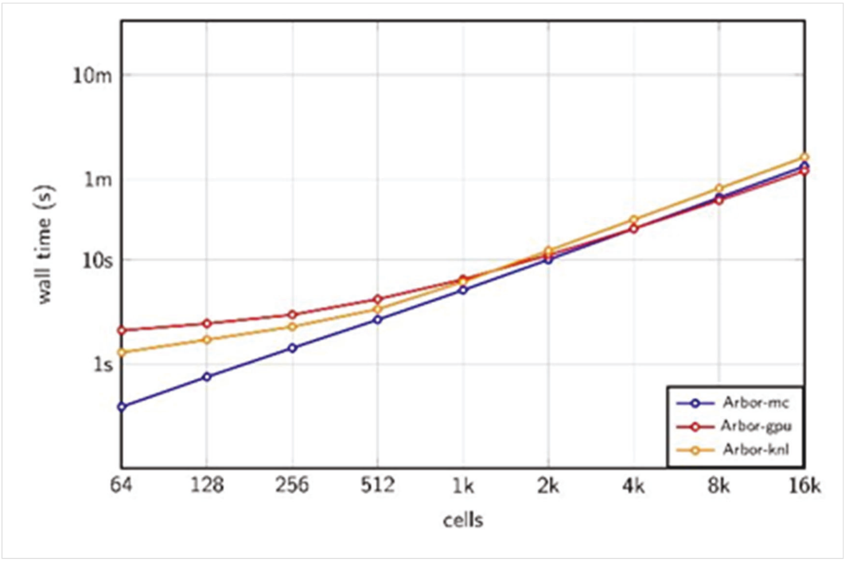

Figure 2.4: Performance of Arbor (based on [34]): Single node wall time of Arbor running on Piz Daint multicore, GPU and Tave KNL

The new NEST kernel shows good weak-scaling behaviour on modern supercomputers (5g-nosort, Figure 2.3, adapted from Figure 7C in Jordan et al. 2018[33]). We go from 32 MPI processes to about 32,000 MPI process, while increasing the problem size therefore in weak scaling by a factor of 1000, and keep the runtime nearly constant. For large numbers of MPI processes, the 5g kernel shows much better scaling behaviour and a decrease in runtime by more than 55% for simulations on the full JUQUEEN[35] system compared to the previous kernel (4g). The S-shaped trend of the simulation time observed for the new NEST kernel (5g-sort) can be explained as follows: For a smaller number of MPI processes, an additional reduction in memory usage is achieved by optimizations for the small-scale to medium-scale regime (compare 5g-sort: small-scale optimizations enabled to 5g-nosort: small-scale optimizations disabled). The optimizations exploit the lesser degree of distribution of each neuron's outgoing connections across processes in the regime up to few thousands of MPI processes,[33]). As gradually the optimizations get less effective with increasing numbers of MPI processes, due to an increasing degree of distribution of connections across processes, simulation times also increase. Note that this effect on scalability in the small-scale to medium-scale regime can be observed in all scaling measurements for NEST shown in this deliverable as in all cases the optimizations

[35] M. Stephan, J. Docter, JUQUEEN: IBM Blue Gene/Q Supercomputer System at Jülich Supercomputing Centre, *Journal of large-scale research facilities*, 1, A1 (2015)

were enabled (5g-sort). In the large-scale regime the outgoing connections of each neuron are fully distributed such that the optimizations for the small-scale to medium-scale regime no longer play a role. The simulation time increases slowly in this large-scale regime.

Arbor's single node performance has been analysed using a randomly connected network benchmark employing CSCS' Piz Daint multicore, GPU and KNL clusters. For more than 4000 cells the GPU is utilized enough to run the benchmark more efficiently in terms of the wall time than on multicore or KNL (Figure 2.4; based on Akar et al. 2018[34]), Table 3 and Fig 4).

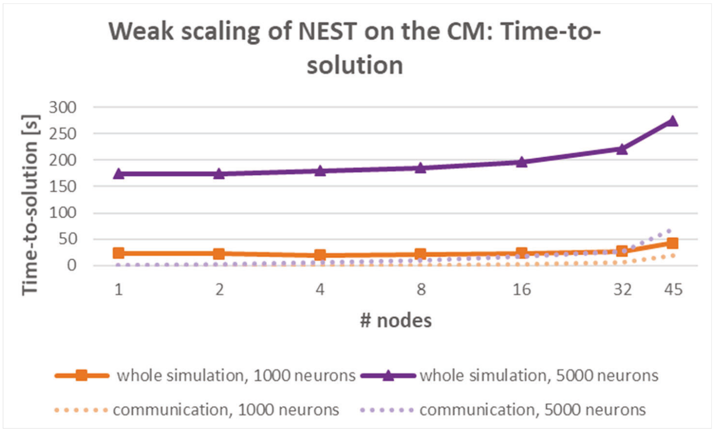

Figure 2.5: Time-to-solution for NEST running the HPC benchmark on the CM: Simulation time and contribution of MPI communication

Within this document we show some results obtained on the DEEP-EST system. Figure 2.5 shows a weak scaling of the HPC benchmark using NEST on the Cluster Module (CM) (1 MPI process per node and 24 threads per MPI process). Figure 2.6 shows the corresponding parallel efficiency. The benchmark network model includes plastic synapses, which need to be updated whenever they transmit a neuronal signal thereby causing workload in addition to neuronal updates. The minimum simulation time (among at least 5 repetitions) and the time spent communicating spikes across MPI processes vs. number of compute nodes is shown for a test case with 1000 and 5000 neurons per thread and 11,250 synapses per neuron[36].

[36] Benchmarks simulated with NEST@da46542 (with timers and optimization for small-scale regime)" for 1000 and 5000 neurons per thread

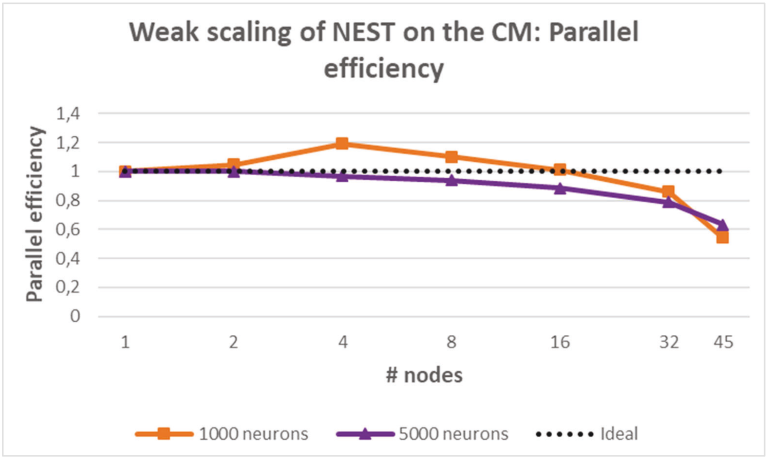

Figure 2.6: Parallel efficiency for NEST running the HPC benchmark on the CM

Figure 2.7 shows the mean simulation time for the 5000 neurons case but with subtracted communication time, which allows for a comparison with measurements obtained using the NEST dry-run mode. A dry-run simulation is carried out by one MPI process emulating the input from other MPI processes, which enables predictions for large-scale simulations. For all simulation time plots lower is better. As NEST optimizations for the small-scale and medium-scale regime were enabled, we observe the typical increase in simulation time described above (c.f. 5g-sort, Figure 2.3), for node counts of 64 and above.

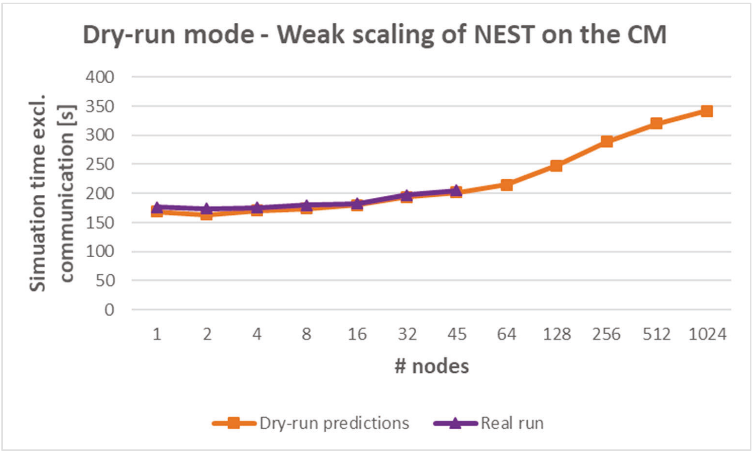

Figure 2.7: Simulation time for NEST running the HPC benchmark on the CM: Dry-run prediction (excluding communication time)

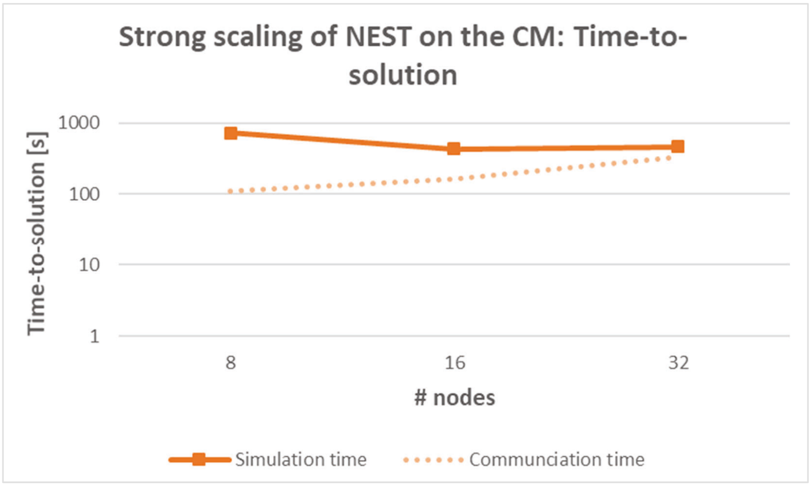

Figure 2.8: Simulation time for NEST running the MAM benchmark on the CM

Figure 2.8 shows a strong scaling of the multi-area model (MAM) benchmark using NEST on the CM (1 MPI process per node and 24 threads per MPI process). Figure 2.9 shows the corresponding parallel efficiency. We have developed the MAM benchmark in this project to provide a scalable benchmark network model with easily controllable parameters and stable dynamics that captures the main performance-relevant features of the multi-area model[37] such as short synaptic transmission delays requiring frequent communication. The benchmark network model consists of 4 million neurons and 5,625 synapses per neuron, where all synapses are static (no additional workload due to synaptic plasticity). The simulations scale well between 8 and 16 MPI processes, but communication time dominates the simulation time at 32 MPI processes. This is due to more frequent communication and less workload compared to the NEST HPC benchmark. The effect of the NEST optimizations for the small-scale and medium-scale regime also plays a role but cannot be distinguished from the other factors.

[37] Schmidt, M. et al (2018) doi. org/10.1371/journal.pcbi.1006359

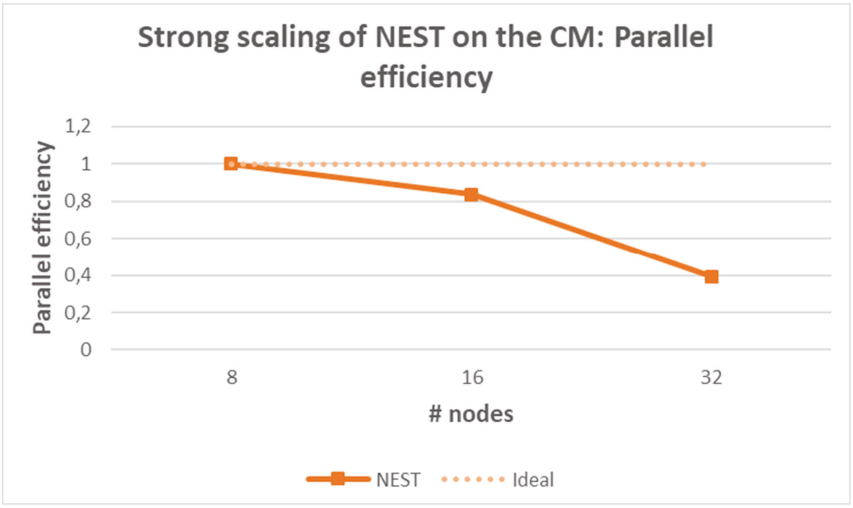

Figure 2.9: Parallel efficiency for NEST running the MAM benchmark on the CM

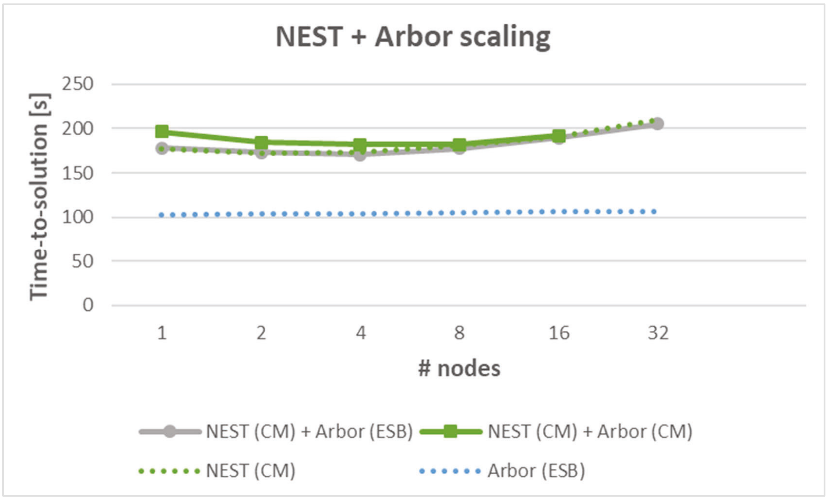

Figure 2.10: Weak scaling time-to-solution for the combined NEST and Arbor simulations

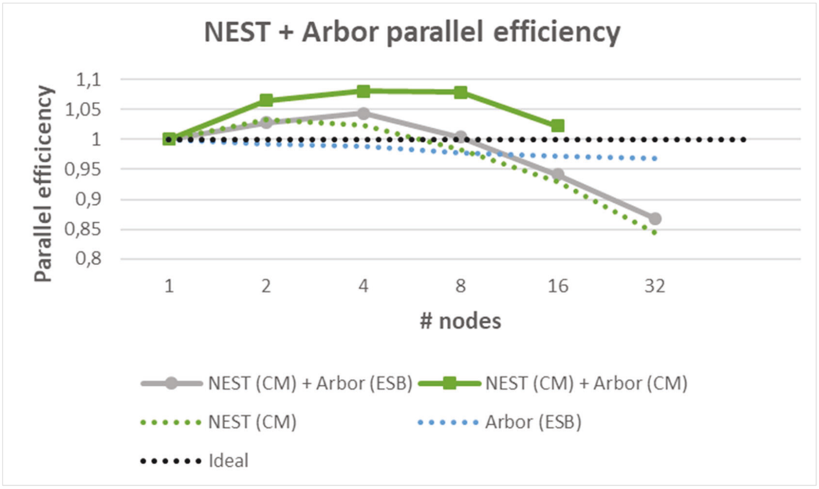

Figure 2.11: Weak scaling parallel efficiency for the Nest - Arbor coupling

Figure 2.10 and Figure 2.11 show the weak scaling behaviour of combined NEST-Arbor simulations[38]. In the smallest case (1 node), NEST simulates 120,000 point neurons while Arbor simulates 1% of this number, i.e., 1,200 compartmental neuron, each on a single compute node. The neuron numbers are scaled linearly with the number of compute nodes. We consider two different configurations: NEST running on the CM and Arbor on the ESB (grey) and NEST and Arbor both running on the CM (green). On the ESB, Arbor uses the GPU on each node, while on the CM Arbor runs 24 threads per node using AVX512. For comparison, we also show the simulation times for the NEST part only (dotted green) and the Arbor part only (dotted blue). Note that for ESB-only and CM-only cases experiments were limited to 16 nodes for each of the programs due to the limited number of nodes.

The underlying Arbor simulations scale perfectly on the ESB when run alone (dotted blue), while the simulation time for pure NEST simulations on the CM (dotted green) scales reasonably well. The combined NEST-Arbor simulation run on CM and ESB (grey) requires essentially the same time as the NEST simulation alone, indicating that the MSA allows us to extend the NEST simulation to a co-simulation without runtime penalty. Executing NEST and Arbor on the CM only leads to increased runtimes (green), indicating the benefit of combining CM and ESB. We also find that co-simulation on CM and ESB reduces energy consumption, see Figure 2.15.

[38] NEST@abc4e0b78

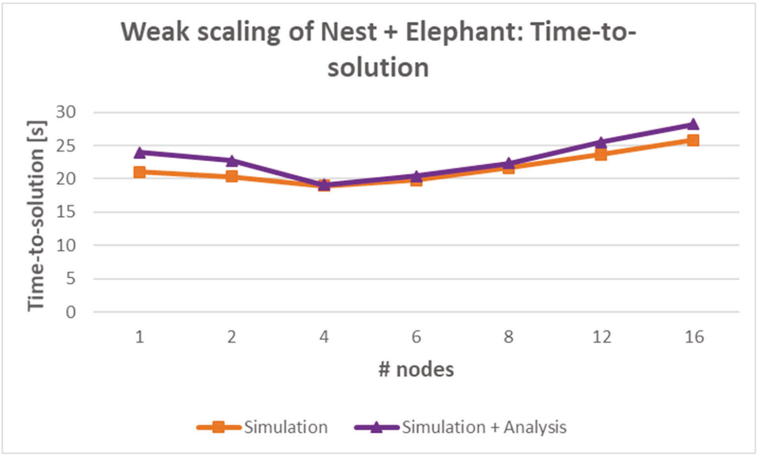

Figure 2.12: Weak scaling of NEST simulation on CM feeding Elephant weak/ensemble-scaling: HPC Benchmark using NEST on CM and analyses with Elephant running on DAM

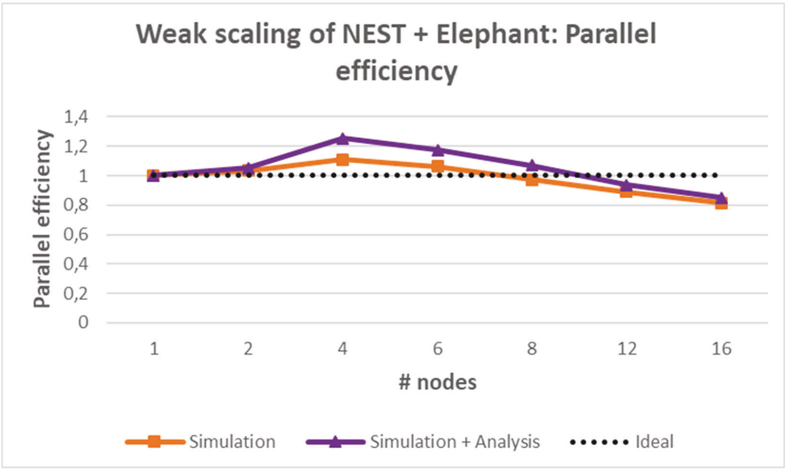

Figure 2.13: Parallel efficiency for the weak scaling NEST + Elephant run

Figure 2.12 shows an example of a NEST simulation running on the CM and sending data for analysis in Elephant on the DAM via MUSIC[39]. Figure 2.13 shows the parallel efficiency. Simulation time and parallel efficiency are shown as function of number of CM nodes used and network size scales linearly with the number of nodes, with 24 MPI processes running on each node (to accommodate MUSICs proper support for

[39] NEST@7616f3eb with bugfix; Elephant v 0.1.0 under Python 3.6.8; MUSIC@8c6b77a57 with path for ParaStationMPI.

threading; approximately 940 neurons per process). The analysis is performed on a single DAM node running two Python processes: one performing ASSET analysis exploiting the GPU and the other performing cross-correlation analysis. Comparison of simulation without spike transfer to the DAM (orange) and simulation with analysis on the DAM (purple) shows that the overhead for analysis is small (approximately 10%) and that, while not perfect, simulation time is roughly in agreement with a weak scaling regime.

2.5.1 Our path to Exascale

Above we discussed to what extent the applications can scale at the moment. The following subsections will outline our path to Exascale

2.5.1.1 What are the limitations – Can they be fixed?

The most visible performance limitation in our work is the relatively poor weak-scaling performance of NEST on the CM for large numbers of neurons as shown in Figure 2.5, which also affects the run time of co-simulations running NEST on the CM and Arbor on the ESB as shown in Figure 2.10. In part, this weak scaling is a consequence of the optimisations for small to medium scale simulations of the NEST 5g kernel, which exploit the lesser degree of distribution of each neuron’s outgoing connections across MPI processes in this regime. As the number of processes increases the exploitation potential decreases rendering the optimisations less and less effective. The optimisations reduce the total simulation times in this regime but due to the gradual decrease in effectiveness distort the observed scaling behaviour on smaller systems such as the existing CM; scaling behaviour of large-scale simulations is not affected by this. Further optimisation will focus on simulation on Exascale systems with an aim at reducing overall communication requirements by introducing support for local connectivity: in real neuronal circuits, a large fraction of the connections are local, but this locality is not yet exploited in NEST or Arbor to minimize communication.

2.5.1.2 How to use future Exascale systems

Exascale computers will be required to allow full-scale simulations of models of primate brains at the resolution of individual neurons. Only Exascale systems will provide the memory necessary to represent the connectivity in networks at the scale of entire brains, the computing power needed to advance the dynamics of neurons, and the interconnects to facilitate signal exchange between neurons. Using a network with highly simplified structure, we demonstrated the feasibility of simulating networks on the size of a cat brain on a major Petascale computer (K, JUQUEEN[40]). Since then,

[40] Kunkel, S. et al. (2014) doi: 10.3389/fninf.2014.00078

we have made important steps in resource efficient dry-run benchmarking[41,42] and directed communication[33]. Dedicated efforts as part of the DEEP-EST project have reduced spike-delivery times[43], addressing a key performance bottleneck. Parallel activities in the EC ICT Flagship *Human Brain Project*[44] focused on reducing the times required to construct networks with realistic complexity in parallel and to further optimise communication schemes for Exascale systems. This work will be pursued in collaboration with Japanese colleagues, which will allow actual experiments on the largest available pre-Exascale system, Fugaku, later in 2021.

2.5.1.3 Where did the DEEP-EST project help on the way to Exascale?

Comprehensive performance profiling allowed us to identify crucial performance bottlenecks in spiking network simulations. Network models with realistic degree (in/out-degree of $O(10^4)$ per neuron) and complexity characteristic of brain networks are represented in the simulator as large adjacency lists which are traversed in random order due to the stochastic activity in network models. This leads to unpredictable memory access patterns and thus inefficient caching. As part of our activities in the DEEP-EST project, we were able to develop new spike-delivery techniques improving caching performance and thus overall simulation performance[43]. The success of the new spike-delivery algorithm was rather unexpected as the memory bottleneck imposed by local spike routing has long been considered insuperable in neuronal network simulation technology. The techniques are not specific to the NEST simulator for which we have developed them, but are applicable to other simulators for pulse-coupled networks with high connection degrees as well. We consider this a generally useful contribution to large-scale network simulation.

Beyond this surprising success and the resulting benefit for the NEST users, our work contributes indirectly to the development of neuromorphic systems. The technology for simulations of spiking neuronal networks on conventional computer architectures informs and inspires the design of neuromorphic systems, and it constitutes an important reference benchmark for such systems regarding accuracy, energy consumption and speed. Mitigating the von Neumann bottleneck in spiking network simulations on conventional architectures by latency-hiding techniques challenges neuromorphic systems. The effective use of such techniques indicates that the memory bottleneck can likely be overcome by many-core systems as naturally each core needs to oversee a decreasing amount of memory.

[41] Kunkel, S., and Scheck, W. (2017) doi: 10.3389/fninf.2017.00040

[42] Kunkel, S., and Scheck, W. In preparation.

[43] Pronold, J. et al. In preparation.

[44] https://www.humanbrainproject.eu, Specific Grant Agreements 2 (2018–2020) and 3 (2020–2023).

2.6 Energy consumption

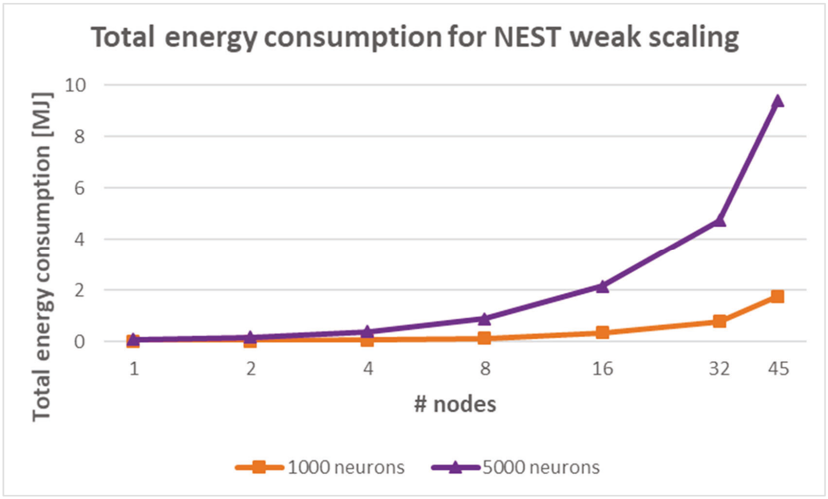

Figure 2.14: Total energy consumption for NEST running the HPC benchmark on CM for two different network sizes

Figure 2.14 shows the total energy consumption from the benchmark runs shown in Figure 2.5 and Figure 2.15 shows the total energy consumption for the NEST-Arbor co-simulations shown in Figure 2.10. While timings shown in Figure 2.5 and Figure 2.10 show the actual simulation time (propagation of network state), the total energy consumption includes the time required for network construction and initialization before the simulation.

For the pure NEST simulation on the CM we observe linear scaling as expected up to 32 nodes followed by a noticeably superlinear increase when simulating on 45 nodes. For the co-simulation, scaling is nearly perfectly linear when combining up to 32 nodes each on CM and ESB. Co-simulations on the CM alone were only performed up to 16+16 nodes, the limit set by the DEEP-EST system size at present, with higher energy consumption when using only the CM, especially for 16+16 nodes case. This indicates energy efficiency gains from the modular system architecture.

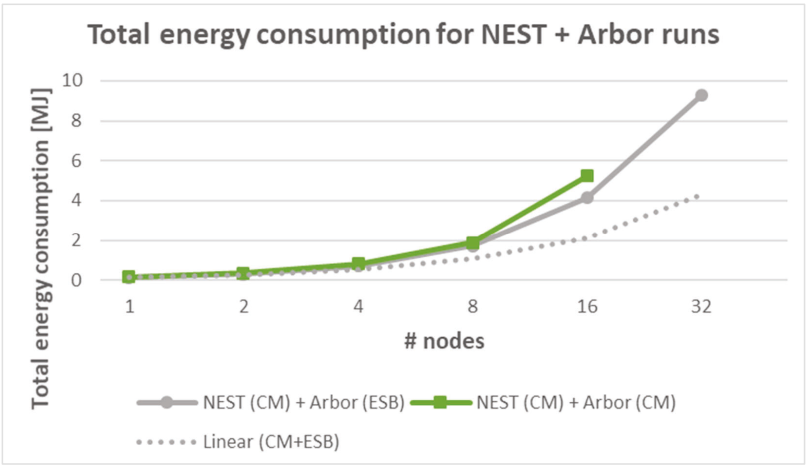

Figure 2.15: Total energy consumption for NEST and Arbor co-simulation running on CM and ESB (grey) and on the CM alone (green). The number of nodes is per simulator, i.e., 16 nodes means NEST running on 16 nodes (always CM) and Arbor running on 16 different nodes (either ESB or CM)

2.7 Performance comparison

After over three years of development, this subsection compares our current application status with their status at the start of the DEEP-EST project.

2.7.1 New spike-delivery algorithm

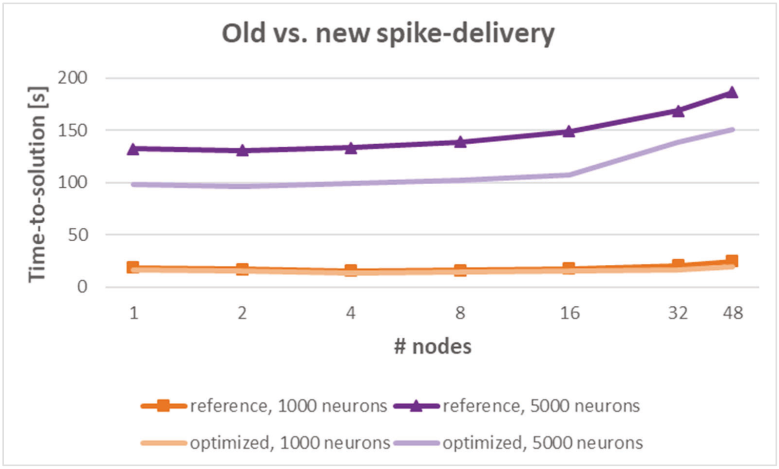

Figure 2.16: Simulation time reduction for new spike-delivery algorithms

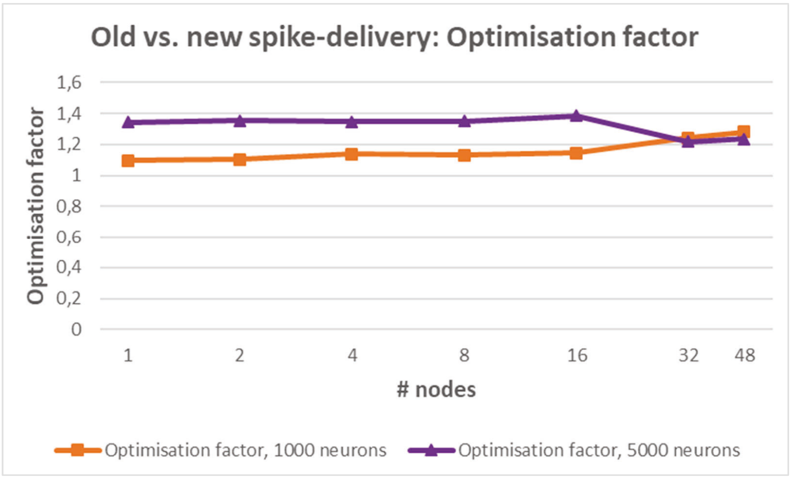

Figure 2.17: Optimisation factor gained by the new spike delivery

Sparse connectivity combined with irregular spiking activity leads to a practically random memory-access pattern during spike delivery. Seemingly this is a worst-case situation for the von Neumann architecture, where for any computation the content of a respective memory unit needs to be transported to the central processing unit and the result needs to be transported back. In weak-scaling spike delivery dominates the simulation time[33]. To overcome the memory bottleneck, we have rearranged the elementary algorithmic steps required to deliver the incoming, essentially random spike data to the process-local targets, such that they can be more efficiently processed by conventional computer hardware. The redesign also includes common latency-hiding techniques such as software prefetching and software pipelining. Figure 2.16 shows a significant reduction in simulation time as a result of new algorithms for spike-delivery in NEST, when comparing the HPC benchmark (2 MPI process per node with 12 threads per process and 1000 or 5000 neurons per thread) and the same case with the new spike-delivery enabled[45]. Figure 2.17 shows the optimisation factor gained by employing the new spike delivery. Note that in this version of the HPC benchmark all synapses were static (no additional workload due to synaptic plasticity), which allowed us to better expose the memory bottleneck.

[45] Optimized spike delivery: HPC-Benchmark simulated with NEST@8f1b08c (with timers and optimization for small-scale regime); reference data simulated with NEST@8897668.

2.8 Conclusion

Running brain-model simulations on Exascale computers to explore brain dynamics at the scale of full brains is a major challenge in computational neuroscience and in simulation technology. The focus put in the DEEP-EST project on improving application scaling and performance has allowed us to test new techniques and to understand factors affecting performance of simulators, especially NEST, even better. This has driven the development of more efficient spike-delivery techniques and the development of an advanced dry-run mode. The latter allows benchmarking of large parallel simulations on a small subset of the relevant system and presents an approach also viable for other, comparable tools.

In situ processing of spike data generated by large-scale brain simulations will be essential as networks are scaled up, since storing a raw spike during simulation and re-loading it for analysis becomes infeasible. Coupling NEST-Arbor and NEST-Elephant enables combining on the one hand simulations at different levels of description and on the other hand simulations and analysis. Our work in the DEEP-EST project in this area has shown that a hybrid approach of distributing different parts of a workflow across different modules of a MSA clearly holds potential.