CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Human Brain Project

JÜLICH
Forschungszentrum

arbor

Designing a HPC-benchmark for biologically realistic Neuron Simulations

T. Hater, B.F.B. Huisman (Forschungszentrum Jülich)

Computational neuroscience is a major consumer of HPC resources. While point models are highly demanding in terms memory and network capabilities, networks of morphologically realistic neurons require significant computational resources [4, 1, 3]. We present a benchmark for studying the performance of such detailed simulations at large scales and evaluating HPC-architectures to be used in this setting. The basis for this project is Arbor, which has been shown to scale well to large nodes counts when using simple cell models [1]. Here, we extend this to a complex model, that is inspired by a model from the Allen cell database[2]. The goal of this project is twofold, first to understand and improve the performance of Arbor at scale and second to study the suitability of current and emerging HPC architectures for these kinds of workloads.

Where to find. . .

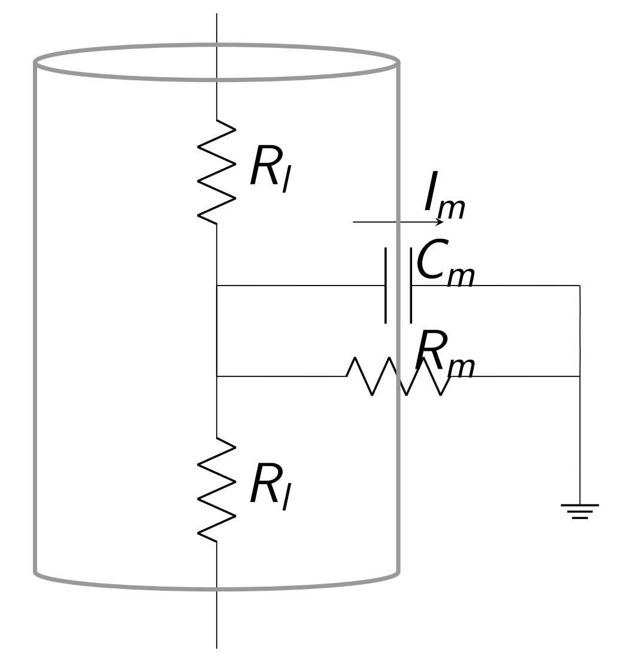
Arbor	arbor-sim.github.io
Contact	t.hater@fz-juelich.de
Poster	github.com/thorstenhater/isc-2023

Detailed Neuron Models

- Cells possess a morphology, split in to compartments.
- Sub-regions are decorated with active components, i.e. ion channels.
- Complex neuron behaviour emerges from the interplay of ion channels.
- Individual ion channels driven by differential equations.
- Potential evolution driven by cable equation

$$\frac{1}{C_m}\partial_t U_m = \partial_x\left(x\partial_x\sigma_l U_m\right) + I_m$$

- Transmembrane current I_m has a leak contribution via R_m and a dynamic component for the ion channels.

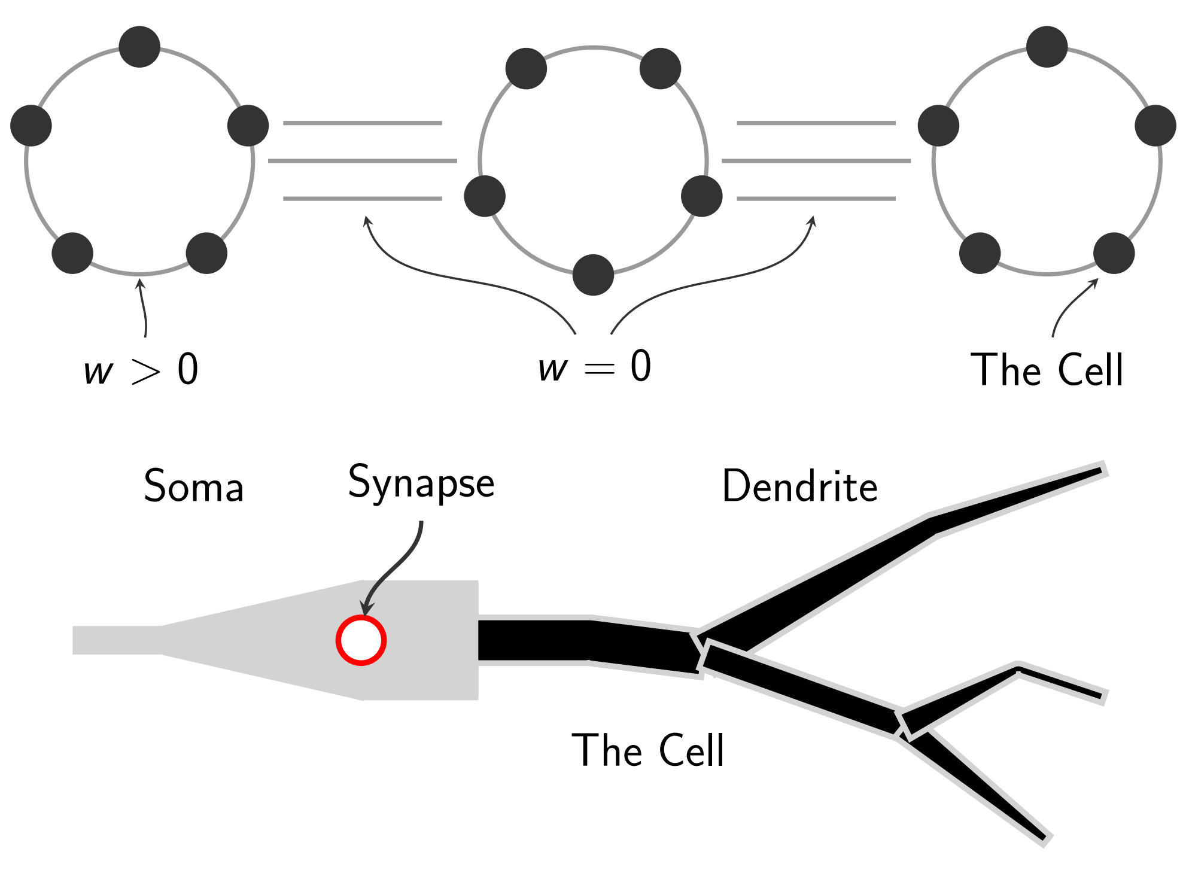


Arbor

- Performance portable library in C++17 for writing neural network simulations.
- Accelerator support for NVIDIA and AMD GPUs.
- Explicit SIMDisation, custom work-stealing thread-pool, and MPI.
- Concurrent communication and computation to hide network latencies.
- High-level Python interface.
- Supports point models and complex cable cells.
- Custom DSL compiler for ion channels.
- Permissive BSD-3 license.

The Cell

- Pseudo-randomly branching tree with tunable
  - depth d
  - branching probability p
  - branch lengths l
  - compartments per branch k
- Two sub-regions with different dynamics
- *Soma:*
  - 6.3 μm sphere at the root of the tree
  - 7 active channels + 1 passive
- *Dendrite:*
  - actual branching structure
  - 4 active channels + 1 passive
- *Synapses:* simple exponential dynamics.
- Dynamics are based on the Allen Cell database [2] and cover relevant protoytypical channels.

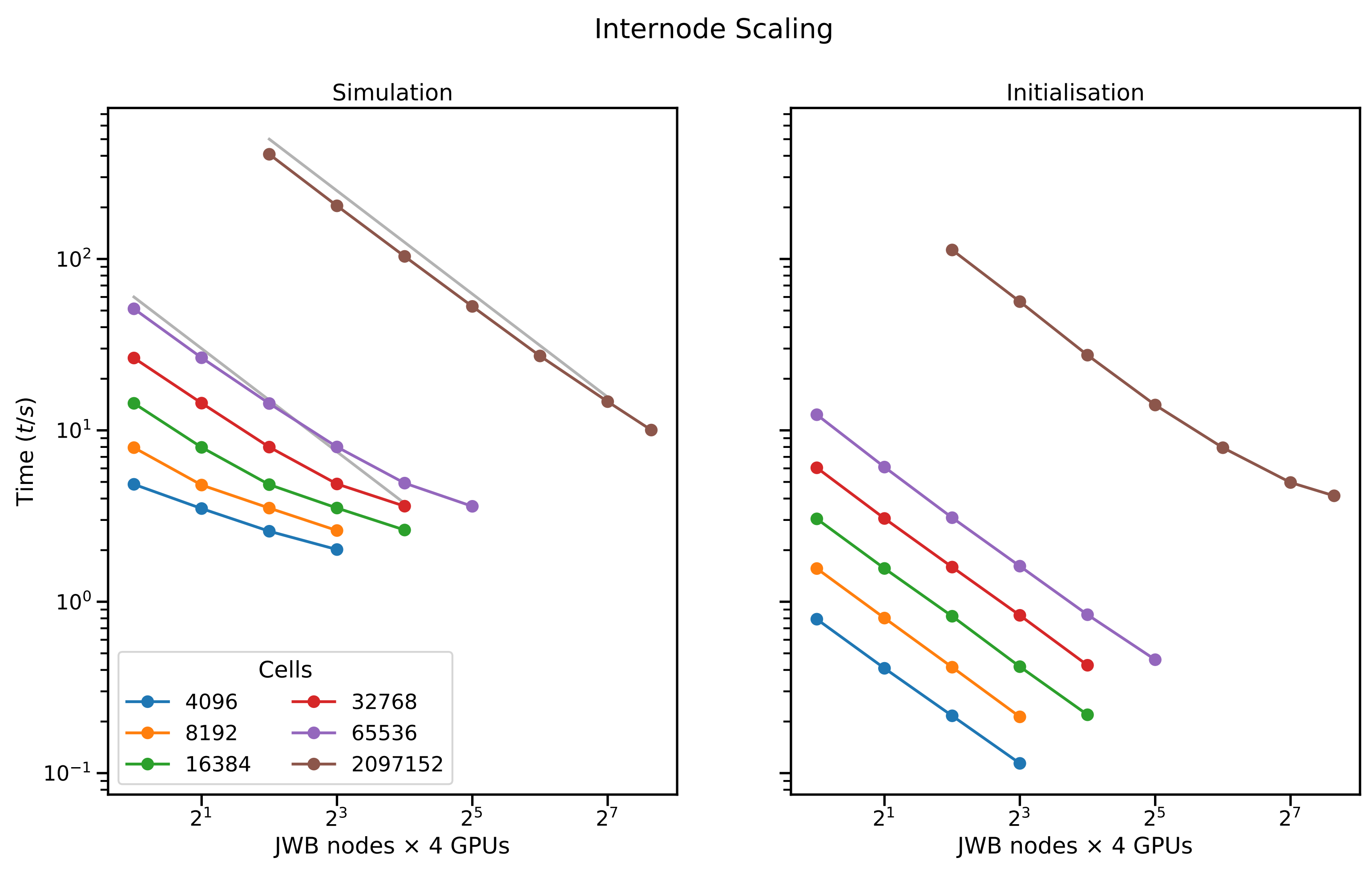


The Network

- N cells in total, subdivided into rings of size n.
- One connection from the soma of cell i to the soma of cell $i + 1 \mod n$
- A spike is injected at $t = 0$ which will propagate indefinitely through each ring.
- A configurable number of additional connections with zero weight are added between rings.
- These stress the system's interconnect, but do not alter the rings' behaviour.
- The combination of steady-state rings and zero-weight external connections allows for a stable and predictable workload.
- The number of spikes generated on all cells is collected to provide a measure of validation.

Results

- We have run this benchmark at varying scales on the JUWELS Booster system at the JSC.
  - 4 × NVIDIA A100 GPU per node.
  - 160 *GB* of GPU memory.
  - 40 *TF*/*s* of double precision compute.
- We chose $d = 10$, $n = 8$, $p = 0.5$, $k = 1$, and $l = 200\,\mu m$ for the parameters.
- Single node profiling shows strong dominance of a single ion channel with 45% time spent. Requires solving a 12 × 12 linear system per compartment.
- Solving the cable equation is the second most relevant cost centre with 15%.
- The workload is limited by the available GPU memory to around 195,000 cells per node.
- (Below) Strong scaling across the GPUs on a single node, showing a small degradation in scaling at 1024 cells per GPU.
- (Right) Strong scaling of different cell counts across up to 256 nodes, efficiency equally falls off at 1024 cells per GPU.



Parallel Efficiency

Efficiency ϵ: the ratio of observed to perfect scaling

$$\epsilon = \frac{T(n_0)}{n \cdot T(n)}$$

For 65536 cells, where $n_0 = 4$ nodes:

Nodes	ϵ_{init}	ϵ_{sim}	ϵ_{total}
8	1.00	1.00	1.00
16	1.03	0.99	0.99
32	1.00	0.97	0.97
64	0.89	0.94	0.93
128	0.71	0.87	0.83
200	0.54	0.81	0.73

Note that the initialisation time is weighted more heavily in ϵ_{total} than one would expect for production use, as the simulated biological time in this benchmark is quite short with 200 *ms*. Typically, the simulated time is of the order of tens of seconds at least.

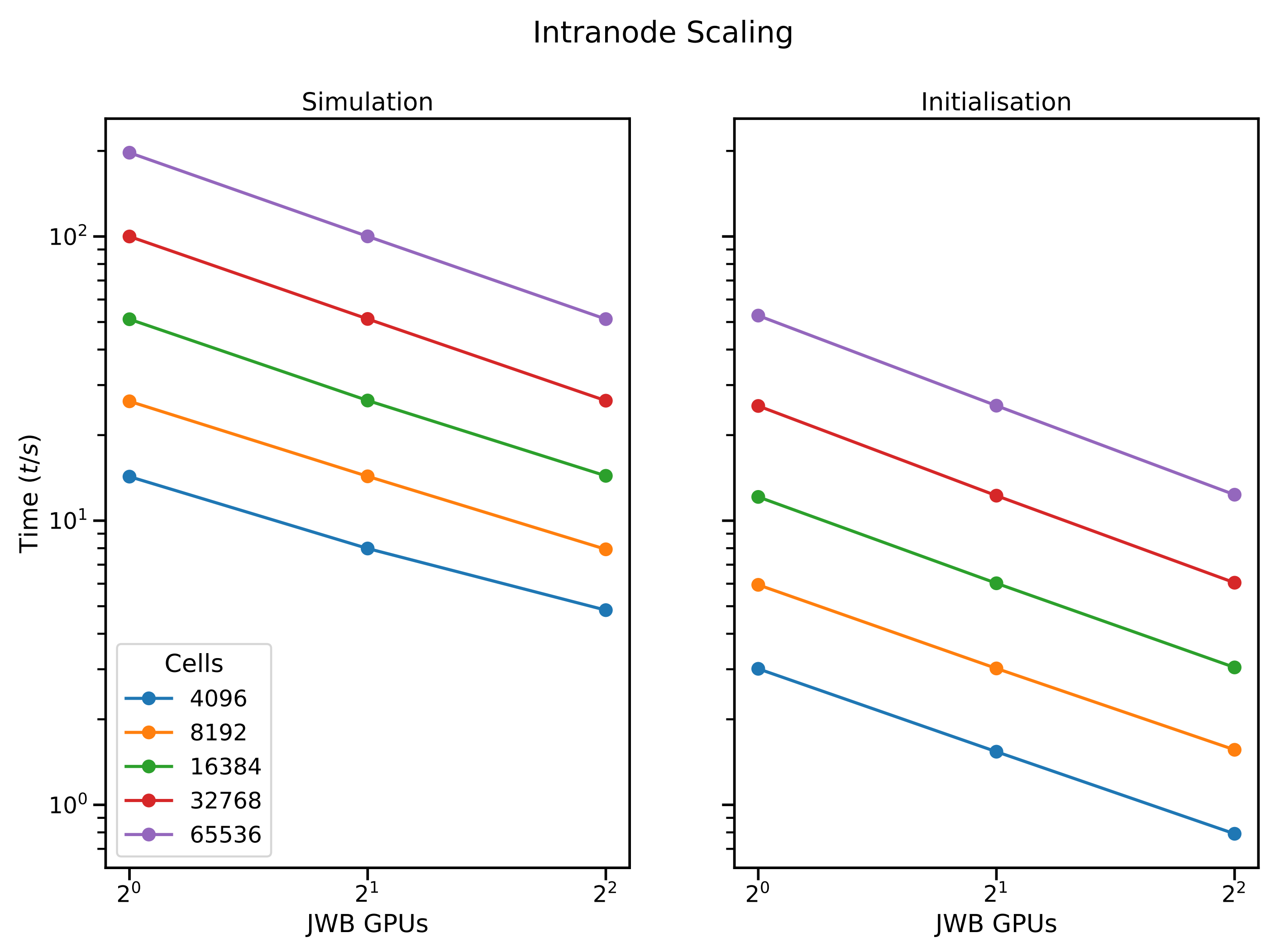
Conclusion

Designing a neuroscience specific HPC benchmark

- Scalable for approximating relevant neuroscience workloads on existing and emerging architectures.
- Workloads can be tuned to stress a mix of system parameters, i.e. computation and interconnect.
- In active use to evaluate existing and upcoming systems at the JSC.
- Allows for studying the performance characteristics of detailed versus point models.

Scaling results for Arbor on JUWELS booster.

- For the showcased workload, Arbor shows scaling up to 256 nodes on JUWELS booster with above 80% efficiency.
- Below 1000 cells per GPU efficiency falls off; indicating the limit of strong scaling.
- Identified some potential areas for improving scalability and overall performance of Arbor.



Future Work

- Extend the node counts to the full JUWELS booster of up to 936 nodes.
- Performing deeper performance analysis of the code under this specific workload.
- We identified a series of potential improvements to Arbor
  - The linear solver used in the ion channels.
  - Arguments to GPU kernels are packed into a single struct, which could lead to performance problems.
  - The DSL compiler for GPU kernels might be improved.
  - Reducing the amount of host/device data transfer.
  - Reducing the memory overhead per cell.
- We plan to evaluate the AMD Instinct and NVIDIA Hopper architectures, as soon as they become available to us. This is both to evaluate these novel designs and to further the development of Arbor.
- Investigate a mixed workload consisting of point neurons located on the CPU and complex models driven by the GPU of the node.

References

[1] N. Abi Akar et al. *Arbor — A Morphologically-Detailed Neural Network Simulation Library for Contemporary High-Performance Computing Architectures.* IEEE, 2019.

[2] Y. N. Billeh et al. *Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex.* Neuron, 2020.

[3] N. T. Carnevale and M. L. Hines. *The NEURON book.* Cambridge University Press, 2006.

[4] M. Gewaltig and M. Diesmann. "NEST (NEural Simulation Tool)". In: *Scholarpedia* (2007).

Acknowledgements

This research has received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements No. 720270 (HBP SGA1), No. 785907 (HBP SGA2), and No. 945539 (HBP SGA3).