WHAT IS ARBOR?

Arbor is a library for implementing performance portable network simulations of multi-compartment neuron models.

- Simulate large networks of morphologically-detailed, spiking neurons
- Library: you control your program/workflow. Interoperable.
- Portable: scientific description is separate from execution instructions. E.g. run one scientific description on laptop CPU, GPU cluster or future hardware.
- *Performance* portable: add optimized backends for new computer architectures. Currently supported:
  - Distributed parallelism using MPI
  - CUDA backend for NVIDIA and AMD GPUs
  - Vectorized backends for x86-64 (KNL, AVX, AVX2) and Arm64 (NEON, SVE) intrinsics
- Executes on all HPC systems in the HBP (and outside).

WHAT IS ARBOR? (2)

Repo: `github.com/arbor-sim/arbor`, website: `arbor-sim.org`

- Latest release: v0.6
- 48 Github forks, 69 Github stars
- 1400+ commits to main branch
- loc: C++: 157k, Python: 13k, reStructuredText: 21k
- 26 contributors, from 9+ institutions

Core contributors

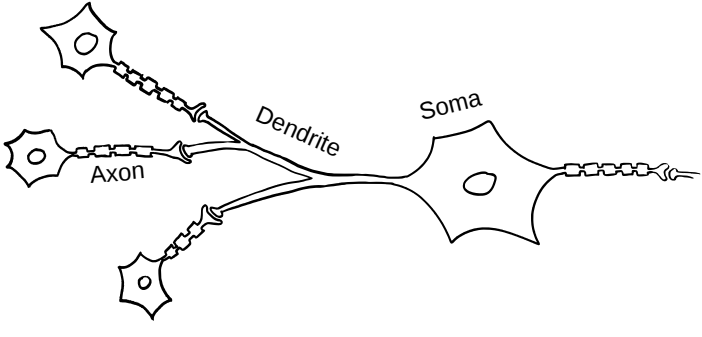
- Ben Cumming, Nora Abi Akar, Fabian Bösch, Simon Frasch, Lukas Drescher
- Anne Küsters, Thorsten Hater, Brent Huisman

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

Modeling

Neuron



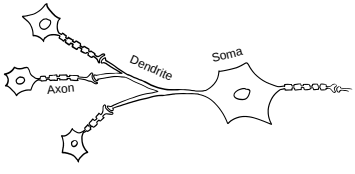
[1]

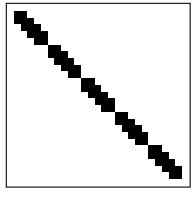
- Dendrite: 1D electric flux on tree structure
- Soma: Emits spikes if the voltage rises over a threshold
- Axon: The output for the spike signal modeled by a time delay

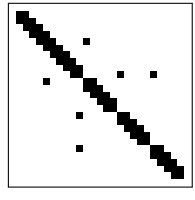
1D flux with varying properties in a tree structure

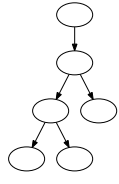
Modeling [2]

Discretization









- Each branch: subdivide each branch → tridiagonal matrix
- With branching: almost tridiagonal → Hines matrix

Which Solver?

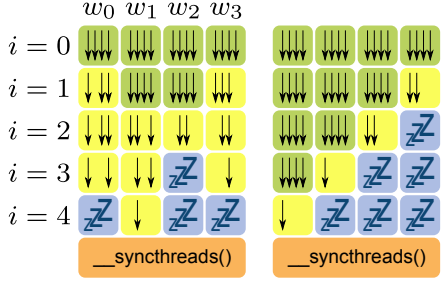
- Tridiagonal matrix: Thomas algorithm
- Hines matrix: tree solver, starting at the leaves

We've got Hines matrices and know how to solve them, let's make it fast!

GPU Recap

Many Threads, grouped in blocks. Good for parallelization:

- enough threads and blocks
- latency hiding, memory access
- low divergence
- independent tasks

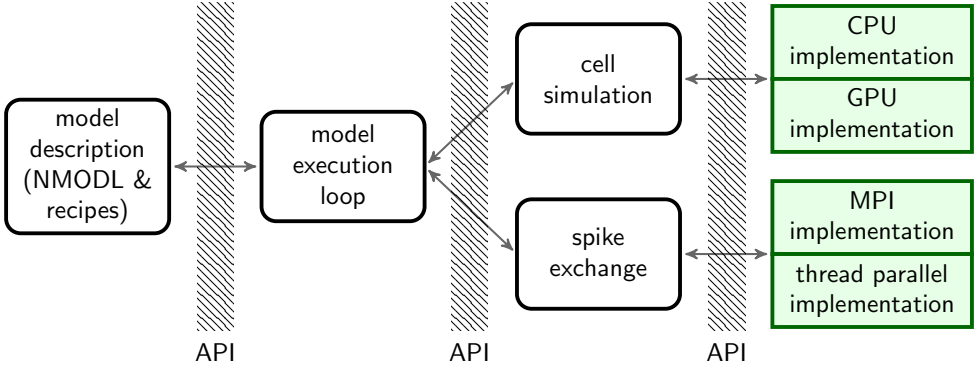


kernel

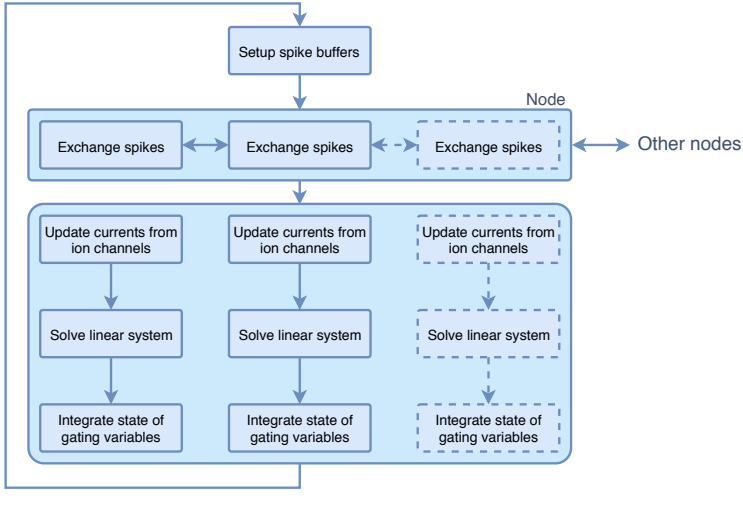
```
int count = counts[threadIdx.x];
for (int i = 0; i < count; ++i) {
    // smart stuff
}
__syncthreads();
```

ARBOR DESIGN

- Modular: components can be substituted according to internal API
- Internal API: 'thin' API; type parameterization allows components to determine low-overhead API data structures



CELL SIMULATION TIMELOOP



WRAP UP

Questions?

- Web: `arbor-sim.org`
- Docs: `docs.arbor-sim.org`
- Community: `github.com/arbor-sim/arbor/discussions`
- Chat: `gitter.im/arbor-sim/community`

Acknowledgements: This research has received funding from the European Unions Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 720270 (Human Brain Project SGA1), Specific Grant Agreement No. 785907 (Human Brain Project SGA2), and Specific Grant Agreement No. 945539 (Human Brain Project SGA3).

EBRAINS

Human Brain Project