INTRODUCTION TO ARBOR

What’s new and demonstration

June 10, 2021 | Brent F. B. Huisman | Jülich Supercomputing Centre

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).

WHO IS ARBOR?

Open development style through Github

- Issues, PR workflow, Discussions, Slack/Gitter
- Code quality: PR review, unit testing, CI at Github and CSCS

Core contributors

- Ben Cumming
- Nora Abi Akar
- Stuart Yates
- Anne Küsters
- Thorsten Hater
- Brent Huisman

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

Website: `arbor-sim.org`

ARBOR STATUS

- Latest release: v0.5. v0.6 imminent.
- 42 Github forks
- 1300+ commits to main branch
- loc: C++ header: 68k, C++: 68k, Python: 16k, reStructuredText: 8k
- 24 contributors, from 9+ institutions

Ongoing collaborations:

- FIPPA - extend Arbor by key plasticity processes to simulate and analyze the long-term adaptive dynamics of large-scale, morphologically-detailed neuronal networks
- Arborio - large-scale model of the inferior olive of the cerebellar complex as a case study
- LFPy - investigating Arbor as possible backend
- Co-simulation - Nest, Elephant, TVB

BUILDING ARBOR

Computational model of Neurons

- Arbor simulates networks of multi-compartment neurons
- Neurons: approximated by axonal delay, synaptic functions and a set of cables connected in a tree.
- Cables: characterized as electrical compartments (frustums) composed of ion channels, cable resistance and capacitance.
- Neurons represented as sparse, close-to-band matrices to be solved (e.g. by Hines solver) against known current states due to synaptic conductance.
- Network and spike exchange between neurons at synapses are represented by concatenations of matrices.

USER DESIGN

Describe the neuroscience first ...

Cells

- Cells represent the smallest model to be simulated
- Cells are the smallest unit of work distributed across processes
- Types:
  - Cable cells
  - Leaky integrate-and-fire cells
  - Spiking cells
  - (Benchmark cells)

Recipes

- Recipes describes models in a cell-oriented manner and supplies methods to:
  - Map global cell identifier gid to cell type
  - Describe cells (Cable cell 123, what is it’s morphology?)
  - List all connections from other cells that terminate on a cell
- Advantage: parallel instantiation of cell data

USER DESIGN

... then translate it into execution.

Cell groups

- Cell groups represent a collection of cells of the same type together with implementation of their simulation
- Partitioning into cell groups provided by decomposition
- A **simulation** manages instantiation of model and scheduling of spike exchange as well as integration for each cell group

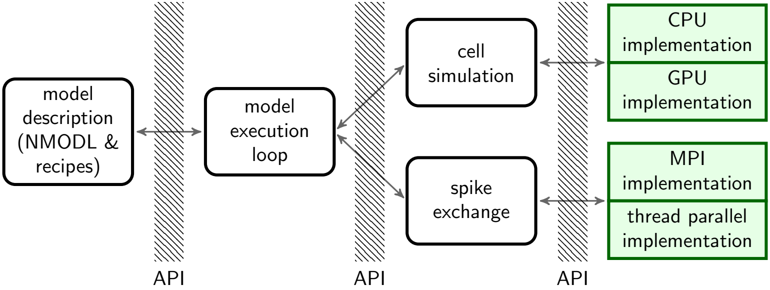
Mechanisms

- In a recipe, mechanisms are specifications of ion channel and synapse dynamics. Advantage: parallel instantiation of cell data.

  Implementations of mechanisms:
- Hand-coded for CPU/GPU execution
- A translator (modcc) compiles a subset of NMODL to architecture-optimized vectorized C++ or CUDA
- Soon: Arblang

INTERNAL DESIGN

Programming interface ensures extensibility



- Components can be substituted according to the internal API.

DESIGN

Summary

Arbor internal model:

- Internal API uncouples model description, execution, spike exchange and cell simulation
- Computational work is hidden in pluggable backends, allowing automatic generation for different architectures

User models are composed of:

- Cells representing the smallest unit of computation
- Recipes representing a parallelizable set of neuron construction and connections
- Mechanisms representing ion channel and synapse dynamics

INTEROPERABILITY, PORTABILITY AND EXTENSIBILITY

Why are they relevant to a computational neuroscientist?

- Why a library?
  - Makes Arbor interoperable with other tools and your way of working.
- What is portability?
  - Write your science and let Arbor worry about how to efficiently execute it.
- What is performance portability?
  - You can extend Arbor to take advantage of new hardware without having to modify your scientific description. Your scientific description will continue to work.

WHAT'S NEW?

- Expanded set of tutorials
- CI significantly expanded
  - Automated building of Python and Spack packages
  - Soon: Ebrains CD
- File format compatibility: cell morphologies
  - SWC
  - NeuroML
  - Neurolucida ASCII
  - Arbor Cable Cell
- Arbor GUI
  - Focussed on cell design, decoration
  - Can run single cell model simulations

WHAT'S COOKING?

- Arbor mechanism description language (codename ARBLANG)
  - Declarative
  - Simple, extendable and maintainable
  - Capable of performing powerful optimizations on the source code
  - Make support in other simulators easy, make Arblang translatable to/from similar DSLs (eg: NEUROML/LEMS, NMODL, SBML)
- Crack the nut of distributed gap junctions
  - MSc will study Wave Relaxation method this summer
  - Arborio is investigating *multi*-GPU cell groups
- Implement synaptic plasticity, synaptic scaling, and structural plasticity
  - FIPPA
- LFP estimation
  - Arborio

DEMO

- To the demo!
- `docs.arbor-sim.org/en/latest/tutorial/network_ring.html`

A BRIEF HISTORY OF ARBOR

- CSCS were working with EPFL on HPC-ification Neuron, Moose, Brian
- Ben Cumming pitched the idea of a clean break to FZJ, where Alex Peyser picked it up
  - Fundamental to Arbor and the investment of CSCS and FZJ in it: better use of computational resources
  - To our knowledge, Arbor is one of the few HBP-grown projects!
- Nest-MC (multi-compartment Nest) is born, unrelated to Nest however
  - initial commit: december 3, 2015
- September 28, 2017: renamed to Arbor
- October 12, 2018: first formal release, v0.1
- April 1, 2020: v0.3 release with Python wrapper
- July 15, 2020: Brent Huisman hired as Product Owner
- May 15, 2021: arbor-sim.org acquired