Brain Research Applications on Minsky

OpenPOWER Academia Discussion Group Workshop 2017

Andreas Herten, Forschungszentrum Jülich, 10 November 2017 *Handout Version*

Outline

- Forschungszentrum Jülich, Germany
- Jülich Supercomputing Centre
- POWER Acceleration and Design Centre
- Strong connection to neuroscience (HPCNS)

History of Brain Research

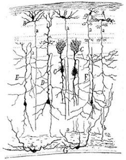
A long way down

JÜLICH FORSCHUNGSZENTRUM

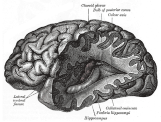
- 1700 BC: Already Egyptians had some knowledge about brain structure
- 17th c.: *Neurology*, status of brain: Thomas Willis (et al)
- 19th c.: Visualization, *neuron doctrine*: Golgi → Ramón y Cajal
- Late 19th c.: neuron electrically excitable
- 20th c.: Brodmann areas (1909); Hodgkin-Huxley model (1952); *neuroscience*
- Today: Brain still not fully decoded
  - Brain atlases in high resolution
  - Models to describe dynamic behavior

  → **Large-scale efforts**

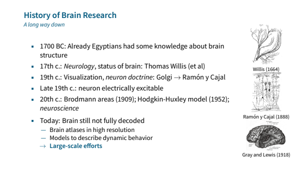
Willis (1664)



Ramón y Cajal (1888)



Gray and Lewis (1918)



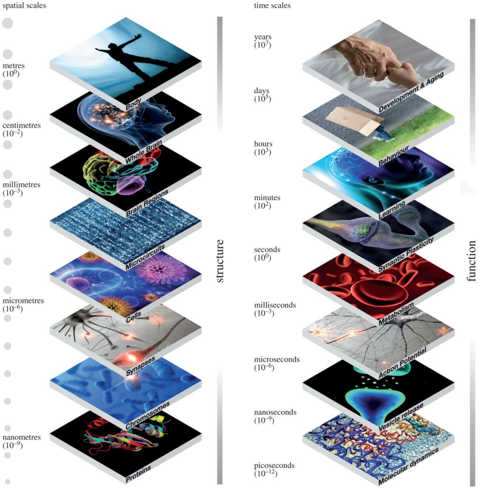
A long way towards understanding of features of the brain

- Egyptians: Drilling hole into skull to cure headaches; brain damage
- Willis: Detailed anatomy of the brain, *Cerebri anatome: cui accessit nervorum descriptio et usus* with detailed drawings; *neurology*
- Cajal uses method of staining brains developed by Golgi (silver chromate) to visualize fibres; start of neural doctrine (=neurons are functional unit of brain)
- Late 19th century: Experiments find that the neuron is electrically excitable
- Brodmann defines areas to create atlas of responsibilities (still used today); Hodgkin-Huxly develop model to describe neurons as electrical circuits with help of giant squid (*action potential*); *neuroscience* is done

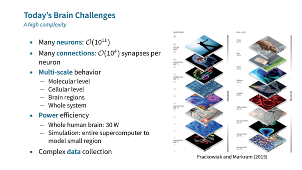
Today’s Brain Challenges

A high complexity

- Many **neurons**: $\mathcal{O}(10^{11})$
- Many **connections**: $\mathcal{O}(10^{4})$ synapses per neuron
- **Multi-scale** behavior
  - Molecular level
  - Cellular level
  - Brain regions
  - Whole system
- **Power** efficiency
  - Whole human brain: 30 W
  - Simulation: entire supercomputer to model small region
- Complex **data** collection



Frackowiak and Markram (2015)



- There are many neurons
- There are many connections between for neuron

→ There are many many connections in total

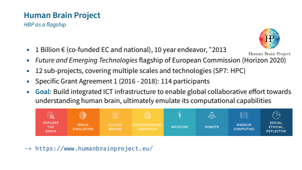
- Different effects for different biological scales: zooming in reveals new features; just like in physics
- Brain runs on amazingly low power footprint
- Data collection is very complex: (some) dynamic studies only with large apparatus; static, but high-res studies only post-mortem, and even then is brain a complex 3D structure

Human Brain Project

HBP as a flagship

- 1 Billion € (co-funded EC and national), 10 year endeavor, *2013
- *Future and Emerging Technologies* flagship of European Commission (Horizon 2020)
- 12 sub-projects, covering multiple scales and technologies (SP7: HPC)
- Specific Grant Agreement 1 (2016 - 2018): 114 participants
- **Goal**: Build integrated ICT infrastructure to enable global collaborative effort towards understanding human brain, ultimately emulate its computational capabilities

→ `https://www.humanbrainproject.eu/`



To understand (*decode*) such a complex organ is an endeavor which needs a large-scale effort → EU Flagship Project

- Funding: 50 % from EU, 50 % from partners
- Sub-projects which focus on different kinds of efforts (mouse brains, theoretical, neuromorphic computing, …, even robotics!)
- Current: Specific Grant Agreement 1

HBP Pilot Systems

HBP & Supercomputers

- Measuring and simulating brain components: computational intensive!

→ High Performance Analytics and Computing Platform (HPAC)

HBP & Supercomputers

- Measuring and simulating brain components: computational intensive!

→ High Performance Analytics and Computing Platform (HPAC)

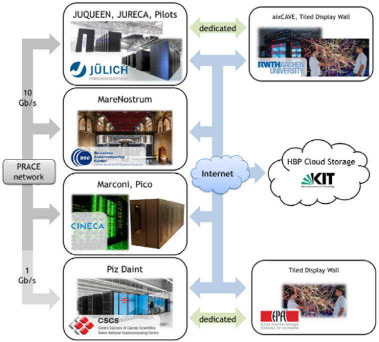
- Already now, HBP needs many computing resource for all the simulations and measurements
  → HPAC
- But simulations will get more and more sophisticated, so demand only increases
- With that large performance need: Special requirements to supercomputers identified
- PCP new way of EU to involve vendors into procurement

HPAC Platform

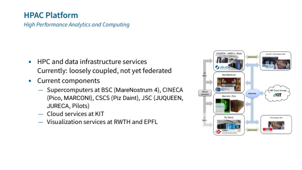
High Performance Analytics and Computing

JÜLICH FORSCHUNGSZENTRUM

- HPC and data infrastructure services
  Currently: loosely coupled, not yet federated
- Current components
  - Supercomputers at BSC (MareNostrum 4), CINECA (Pico, MARCONI), CSCS (Piz Daint), JSC (JUQUEEN, JURECA, Pilots)
  - Cloud services at KIT
  - Visualization services at RWTH and EPFL



Member of the Helmholtz Association



- Largest European supercomputers bundled
- Eventually coupled together
- Currently on the way there, with individual parts finished (collaboratory, UNICORE)

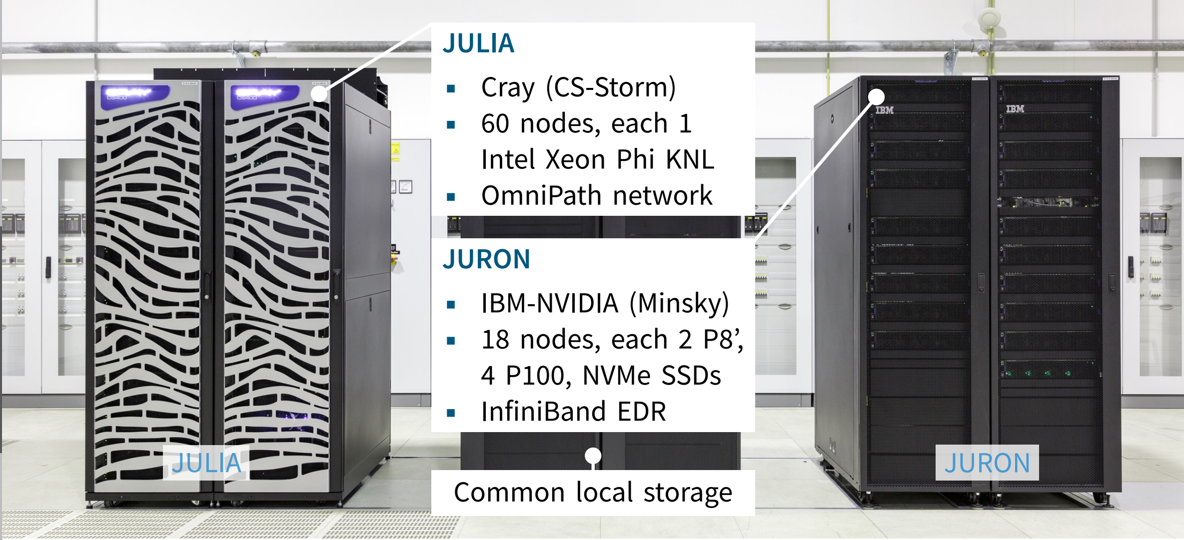
HBP & Supercomputers

- Measuring and simulating brain components: computational intensive!

→ High Performance Analytics and Computing Platform (HPAC)

- Large-scale brain simulations: PFLOP/s, PB
  → Need capability of a supercomputer!
- Special requirements
  - Interactive, scalable visualization (in-situ)
  - Large memory footprint of data (dense memory, fast interconnects)
  - Dynamic resource management, interactive steering
  - Various data sources (eventually: federated services)

→ Pre-Commercial Procurement of two systems: **JULIA** and **JURON**



JULIA

- Cray (CS-Storm)
- 60 nodes, each 1 Intel Xeon Phi KNL
- OmniPath network

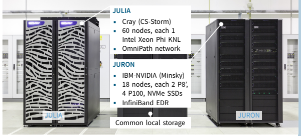
JURON

- IBM-NVIDIA (Minsky)
- 18 nodes, each 2 P8', 4 P100, NVMe SSDs
- InfiniBand EDR

Common local storage

JULIA & **JURON**: Human Brain Project *Prototypes*

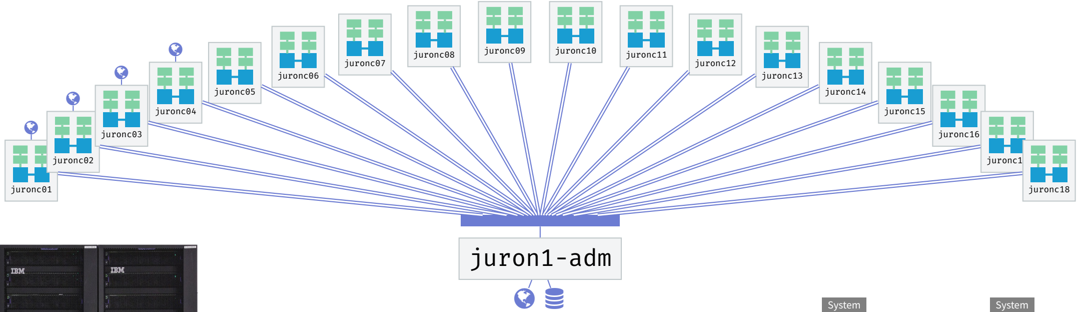
Brain Research Applications on Minsky
└─HBP Pilot Systems
└─JURON
└─JULIA & JURON



JULIA & **JURON**: Human Brain Project *Prototypes*

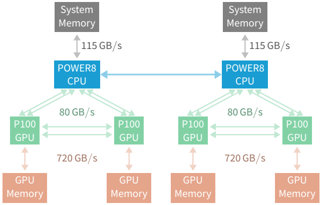
- Two system: Intel-KNL-based JULIA and POWER8'-P100-based JURON
- Match requirements in their own ways
- Middle: A common storage cluster, attached to JURON and JULIA faster than Jülich's global file system GPFS

System Configuration

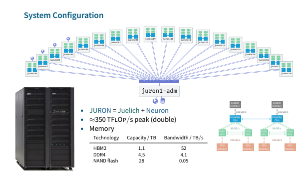


- JURON = Juelich + Neuron
- ≈350 TFLOP/s peak (double)
- Memory

Technology	Capacity / TB	Bandwidth / TB/s
HBM2	1.1	52
DDR4	4.5	4.1
NAND flash	28	0.05



JÜLICH FORSCHUNGSZENTRUM



- JURON's 18 compute nodes, connected via Ethernet and InfiniBand to switch; login node connected via Ethernet to switch
- Login node: access from outside and to Jülich storage resources (+ PCP storage system)
- Also: 4 visualization nodes with direct access from outside
- Capacity and bandwidth are combined values

GPU Hackathon in Jülich

First HBP Applications on JURON

- Eurohack: 1 week of application porting to GPU at JSC in March 2017
- 10 teams; 3 neuroscience

Arbor	Optimizing GPU code of simulation (formerly *NestMC*)
TVB-HPC	First port of back-end to CUDA
The PLI Guys	Build CUDA back-end to Python simulation

  - $>$1000 jobs launched, ⅔ on JURON
  - Every team accelerated code and went home motivated

→ Strong interest in GPU and JURON

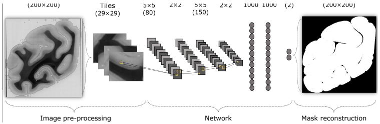
- GPU Hackathon (*Eurohack*) in Jülich; one of the events organized with ORNL all over the world
- Intense work atmosphere, very productive
- Three applications from neuroscience, in the following presented as examples

Applications

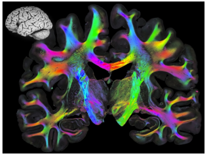
Diverse Set of Applications

Many scales, many steps

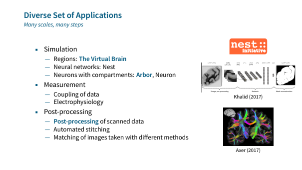
- Simulation
  - Regions: **The Virtual Brain**
  - Neural networks: Nest
  - Neurons with compartments: **Arbor**, Neuron
- Measurement
  - Coupling of data
  - Electrophysiology
- Post-processing
  - **Post-processing** of scanned data
  - Automated stitching
  - Matching of images taken with different methods



Khalid (2017)



Axer (2017)



- Jülich specializes in tools for simulation and post-processing of scanned data
- Strong connection with Supercomputing Centre, specialized interface division: High Performance Computing in Neuro Science (HPCNS) + Simulation Lab Neuro Science (SLNS)
- There's much more to it – 12 sub-projects of HBP

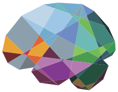
Applications

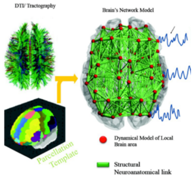
TVB-HPC

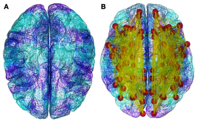
The Virtual Brain

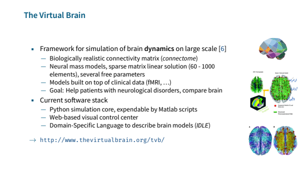
- Framework for simulation of brain **dynamics** on large scale [6]
  - Biologically realistic connectivity matrix (*connectome*)
  - Neural mass models, sparse matrix linear solution (60 - 1000 elements), several free parameters
  - Models built on top of clinical data (fMRI, …)
  - Goal: Help patients with neurological disorders, compare brain
- Current software stack
  - Python simulation core, expendable by Matlab scripts
  - Web-based visual control center
  - Domain-Specific Language to describe brain models (*IDLE*)

→ `http://www.thevirtualbrain.org/tvb/`



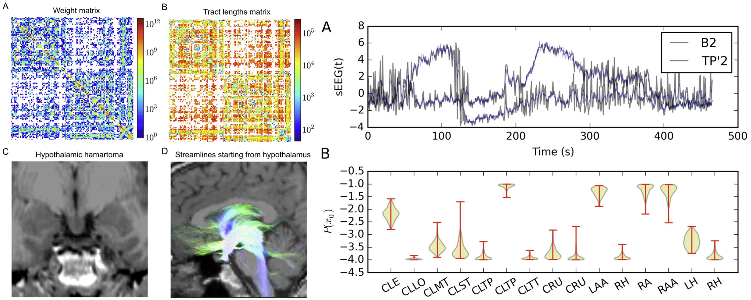






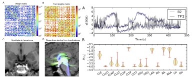
- TVB: software infrastructure to simulate individual brains as models
- Match with measurements; structural vs. functional data
- Eventual goal: simulate a patient's brain in software and guide cure for illness
- Basically written in Python with interfaces to individual Matlab scripts for extension
- View the virtual brain at a web-based applications
- DSL for description of brain models
- Pictures
  1. Logo TVB
  2. Flow of typical brain simulation; input are fibre structures (up, connectome, visualized for example with *Diffusion Tensor Imaging*) and regions of brain
  3. Connectome close-up

Example TVB Science on JURON



- Recently run: Monte Carlo model inference for clinical epilepsy models
  Pictures from "The Virtual Epileptic Patient: Individualized whole-brain models of epilepsy spread" [7]
- Currently single-threaded application; no performance gain yet

Example TVB Science on JURON



- Recently run: Monte Carlo model inference for clinical epilepsy models
  Pictures from “The Virtual Epileptic Patient: Individualized whole-brain models of epilepsy spread” [7]
- Currently single-threaded application; no performance gain yet

- Also TVB currently runs on JURON
- Example epileptic simulation

TVB-HPC

- Traditional approach: Serial computation of models
- TVB-HPC: Fast, parallel back-end (parallel in parameters)
- At GPU hackathon:
  - Optimize specific mass, coupling, post-processing models
  - Study data access issues
  - Learn advanced GPU techniques

  → $20\times$ speedup
- **JURON**: CUDA code for *Kuramoto* model as proof-of-concept
- Since then: Automated code generation

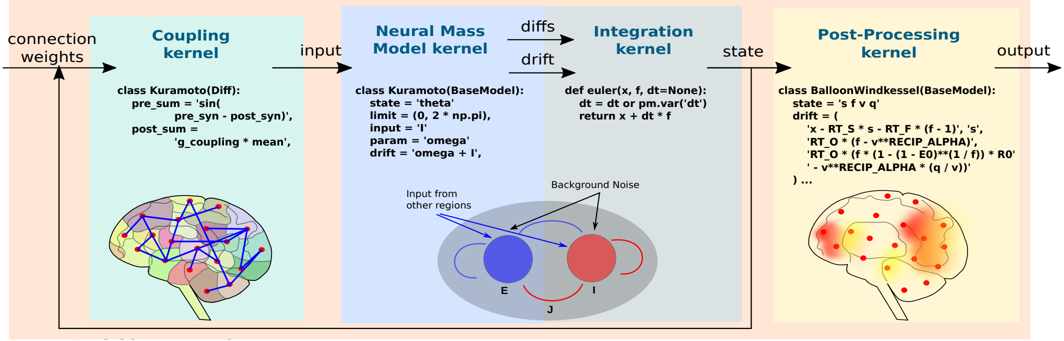
TVB-HPC

- Traditional approach: Serial computation of models
- TVB-HPC: Fast, parallel back-end (parallel in parameters)
- At GPU hackathon:
  - Optimize specific mass, coupling, post-processing models
  - Study data access issues
  - Learn advanced GPU techniques
  - → 20× speedup
- **JURON:** CUDA code for *Kuramoto* model as proof-of-concept
- Since then: Automated code generation

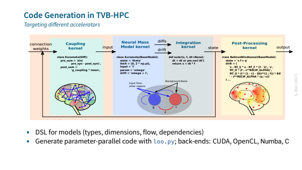
- TVB: Serial simulation of single brain model
- TVB-HPC: Optimized, HPC-targeted simulation of multiple versions of model by simulating many parameters of model at once in parallel
  - First (and at Hackathon): One single model ported to CUDA – very good results
  - Now: Focus on automated code generated

Code Generation in TVB-HPC

Targeting different accelerators



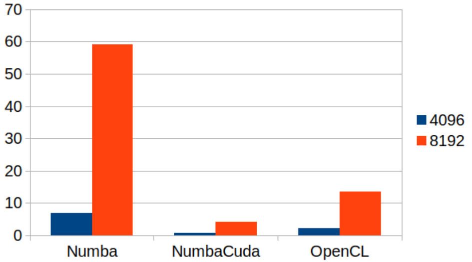
- DSL for models (types, dimensions, flow, dependencies)
- Generate parameter-parallel code with `loo.py`; back-ends: CUDA, OpenCL, Numba, C



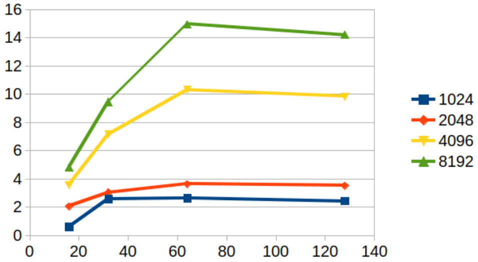
- User writes models in DSL
- Models are most of the time neural mass models, but also coupling and integration kernels (or select one pre-existing for the latter)
- From DSL, loo.py is used to generate to-be-accelerated code

Code Generation in TVB-HPC

Results on JURECA



Test kernel execution time (y/ms) for different targets



Test kernel execution time speedup Numba+CUDA vs. Numba (y) for different number of threads (x)

S. Diaz (2017)

- JURECA node: 2 Intel Haswell CPUs (12 cores), 2 NVIDIA Tesla K80 GPUs
- Numba: Decorator-based auto-acceleration for Python (JIT compilation with `@jit`); different targets

Code Generation in TVB-HPC

Results on JURECA

Generated CPU-targeted code

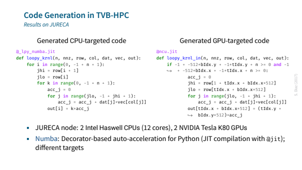
```
@_lpy_numba.jit
def loopy_krnl(n, nnz, row, col, dat, vec, out):
    for i in range(0, -1 + n + 1):
        jhi = row[i + 1]
        jlo = row[i]
        for k in range(0, -1 + n + 1):
            acc_j = 0
            for j in range(jlo, -1 + jhi + 1):
                acc_j = acc_j + dat[j]*vec[col[j]]
            out[i] = k*acc_j
```

Generated GPU-targeted code

```
@ncu.jit
def loopy_krnl_in(n, nnz, row, col, dat, vec, out):
    if -1 + -512*bIdx.y + -1*tIdx.y + n >= 0 and -1
    ↪  + -512*bIdx.x + -1*tIdx.x + n >= 0:
        acc_j = 0
        jhi = row[1 + tIdx.x + bIdx.x*512]
        jlo = row[tIdx.x + bIdx.x*512]
        for j in range(jlo, -1 + jhi + 1):
            acc_j = acc_j + dat[j]*vec[col[j]]
        out[tIdx.x + bIdx.x*512] = (tIdx.y +
        ↪  bIdx.y*512)*acc_j
```

S. Diaz (2017)

- JURECA node: 2 Intel Haswell CPUs (12 cores), 2 NVIDIA Tesla K80 GPUs
- Numba: Decorator-based auto-acceleration for Python (JIT compilation with `@jit`); different targets



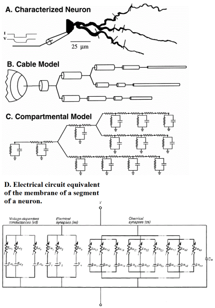
- Current studies with loo.py generation on JURECA, but code runs also on JURON
- Up to 16-fold speed-up for Numba+CUDA vs Numba
- In Numba+CUDA, Numba generates CUDA code in run-time; also here loop.py provides the raw code, which is more targeted towards CUDA already

Applications

Arbor

Arbor Introduction

- TVB: Large scale; effective models; dynamics
- Nest: Biologically correct; point-like neurons; large, spiking networks
- Arbor: Neurons with internal structure, multi-compartment



— Hodgkin-Huxley model: network of neurons as circuit [3]

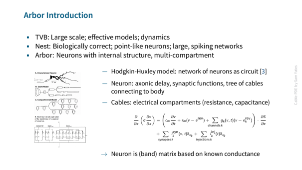
— Neuron: axonic delay, synaptic functions, tree of cables connecting to body

— Cables: electrical compartments (resistance, capacitance)

$$\frac{\partial}{\partial x}\left(\sigma\frac{\partial v}{\partial x}\right) = \left(c_m\frac{\partial v}{\partial t} + r_m(v - e^{\mathrm{rev}}) + \sum_{\text{channels } k} g_k(v,t)(v - e_k^{\mathrm{rev}})\right)\cdot\frac{\partial S}{\partial x} + \sum_{\text{synapses } k} I_k^{\mathrm{syn}}(v,t)\delta_{x_k} + \sum_{\text{injections } k} I_k^{\mathrm{inj}}(t)\delta_{x_k}$$

→ Neuron is (band) matrix based on known conductance

Cable PDE by Sam Yates



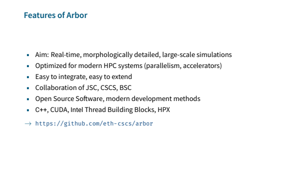
- Zoom in to brain: TVB → Nest → Arbor
- Nest looks at really large networks of point-like neurons and simulates spikes
- Arbor includes now also the internal structure of neurons (neurons are multi-compartment)
- A simplified neuron sketch: Neuron is core, which connects through long axons to the synapses, which are attached to dendrites, of other neurons
- With that a tree of cables is created, first seen by Hodgkin and Huxley
- Cables can be described as partial differential equations of capacitance and resistance
- Describing all cables and further structures leads to a sparse matrix, which has most entries on the main band (but not all)

JÜLICH FORSCHUNGSZENTRUM

Features of Arbor

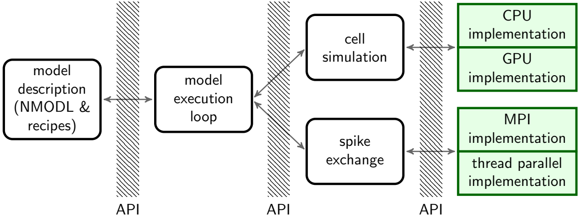
- Aim: Real-time, morphologically detailed, large-scale simulations
- Optimized for modern HPC systems (parallelism, accelerators)
- Easy to integrate, easy to extend
- Collaboration of JSC, CSCS, BSC
- Open Source Software, modern development methods
- C++, CUDA, Intel Thread Building Blocks, HPX

→ https://github.com/eth-cscs/arbor

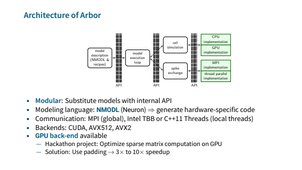


- Other simulators have long history in development and grew historically; → not a-priori well-suited for HPC-like simulations (which are needed to eventually simulate the real brain)
- Arbor effort of JSC, CSCS, BSC to develop a modern simulator, which is built with HPC in mind directly (Arbor was previously called NestMC)
- Open Source Sofware, developed on Github, Continuous Integration
- Many modern technologies

Architecture of Arbor



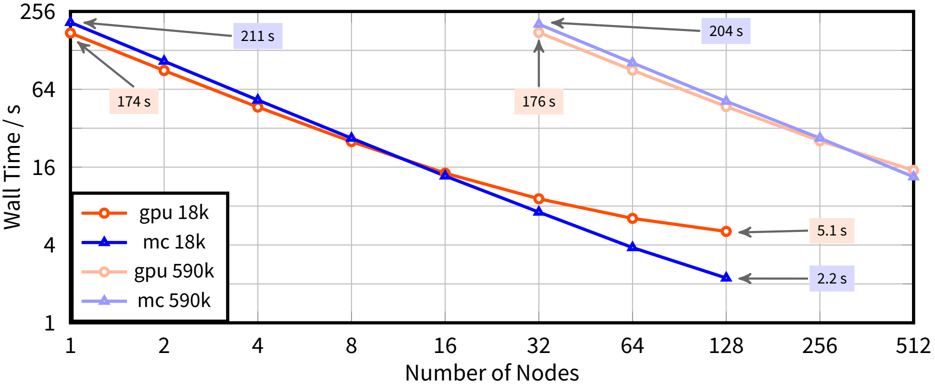
- **Modular**: Substitute models with internal API
- Modeling language: **NMODL** (Neuron) ⇒ generate hardware-specific code
- Communication: MPI (global), Intel TBB or C++11 Threads (local threads)
- Backends: CUDA, AVX512, AVX2
- **GPU back-end** available
  - Hackathon project: Optimize sparse matrix computation on GPU
  - Solution: Use padding → 3× to 10× speedup



- Adopts NEURONs modeling language (NEURON: Another multi-compartment simulator, but also not well-suited for parallel execution w/o modifications)
- Can target GPUs and Xeon Phi (KNLs)
- CPU sorts and packages data closely together, package potentially offloaded to accelerator, solved, and back

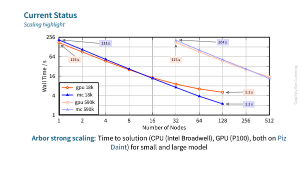
Current Status

Scaling highlight



Graph by Ben Cumming

Arbor strong scaling: Time to solution (CPU (Intel Broadwell), GPU (P100), both on Piz Daint) for small and large model

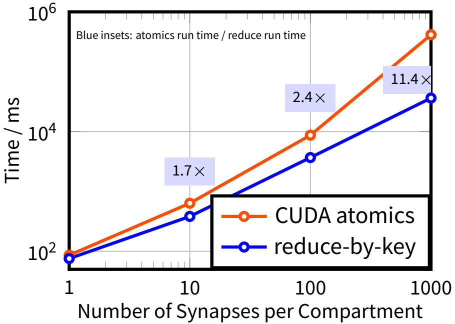


- Currently no JURON plots available, but same/similar tests have been run
- Plot on Piz Daint (which also has P100 GPUs)
- Still more potential within GPUs, actively developed
  - Small model
    - Low number of nodes is better for GPU, because then device can crunch a lot of data (which it is good at)
    - But speed-up eaten up by CPU, which needs to package the large data for the GPU
    - For high number of nodes CPU is faster because the data packages are smaller and can be solved by the CPU without any transfer overheads
  - Large model: pretty much similar

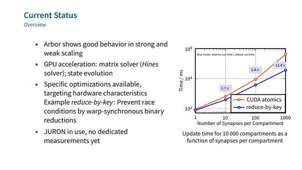
Current Status

Overview

- Arbor shows good behavior in strong and weak scaling
- GPU acceleration: matrix solver (*Hines* solver); state evolution
- Specific optimizations available, targeting hardware characteristics
  Example *reduce-by-key*: Prevent race conditions by warp-synchronous binary reductions
- JURON in use, no dedicated measurements yet



Update time for 10 000 compartments as a function of synapses per compartment



- Much effort in developing all the details
  Example: Hines solver – A solver for mostly-band-matrices; very specific
- Example: Reduce-by-key – Instead of using mutexes to write to common memory location, reduce-by-key uses warp-level binary reductions to increase memory efficiency

Applications

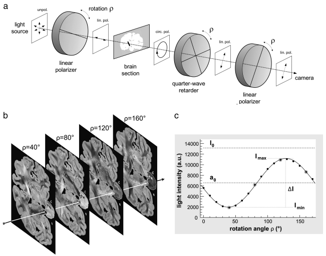
PLI ICA

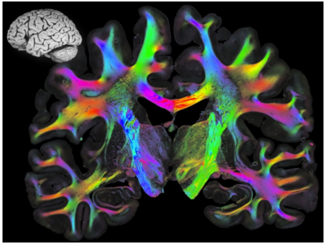
3D PLI ICA

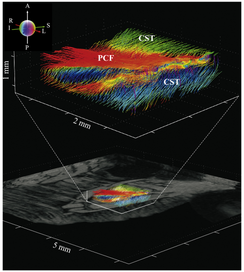
PLI…

- 3D Polarized Light Imaging (PLI): Capture brain slices under polarized light
- Capture at many angles (18, 0° to 170°)
- Myelin around axons refracts light based on inclination to polarization plane

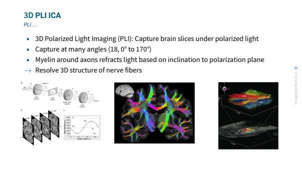
→ Resolve 3D structure of nerve fibers







Images by Axer et al. [8]



- Amount of transmitted light changes (sinusoidally) as function of angle between axon and polarization plane of light; actually light is polarized with filters which change the polarization plane
- Use information of refraction to measure brain in 3D (in slices)

3D PLI ICA

...ICA

JÜLICH FORSCHUNGSZENTRUM

- Independent Component Analysis (ICA): Separate complex signal into components; mixture of sources $\rightarrow$ individual contributions
- Signal-processing method, blind source separation
- Basis: Sinusoidal distribution of measurement basis functions

$\rightarrow$ Identify noise and artifacts in decomposition for removal

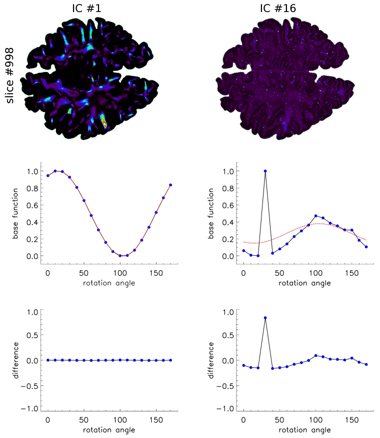
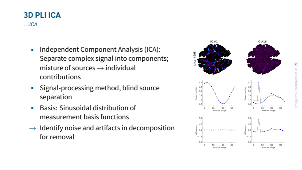


Image by Dammers et al. [9]



- ICA: Method to decompose signal
- Measured signal is overlayed mixture of unknown and independent sources; mixture on sources appear on all measurements (number of measurements needs to be greater than number of sources)
- ICA decomposes mixture into individual sources
- Picture: different decomposition parts – signal (left) vs. noise (right)
  Signal fits very well to a sinus function, noise not that well. Bottom: difference to sinus fit

ICA Challenges and Status

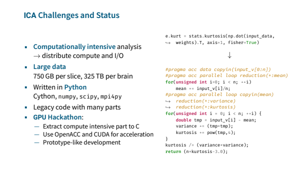
JÜLICH FORSCHUNGSZENTRUM

- **Computationally intensive** analysis
  → distribute compute and I/O
- **Large data**
  750 GB per slice, 325 TB per brain
- Written in **Python**
  Cython, `numpy`, `scipy`, `mpi4py`
- Legacy code with many parts
- **GPU Hackathon**:
  - Extract compute intensive part to C
  - Use OpenACC and CUDA for acceleration
  - Prototype-like development

```
e.kurt = stats.kurtosis(np.dot(input_data,
↪   weights).T, axis=1, fisher=True)
```

↓

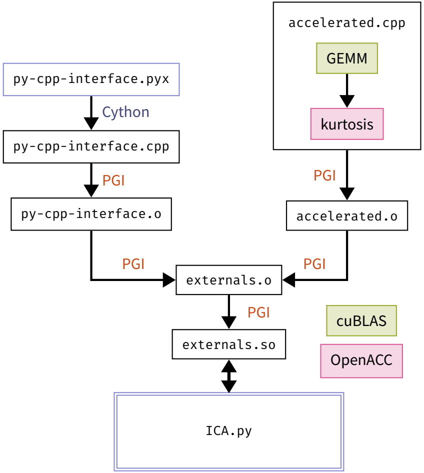
```
#pragma acc data copyin(input_v[0:n])
#pragma acc parallel loop reduction(+:mean)
for(unsigned int i=0; i < n; ++i)
    mean += input_v[i]/n;
#pragma acc parallel loop copyin(mean)
↪   reduction(+:variance)
↪   reduction(+:kurtosis)
for(unsigned int i = 0; i < n; ++i) {
    double tmp = input_v[i] - mean;
    variance += (tmp*tmp);
    kurtosis += pow(tmp,4);
}
kurtosis /= (variance*variance);
return (n*kurtosis-3.0);
```

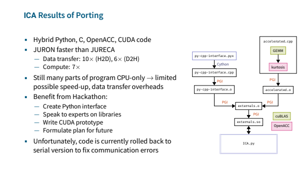


- ICA: Team I co-mentored at GPU hackathon
- Prototype-like development to port the Python application to GPU
- Decided not to use PyCUDA (or similar) but to write C implementations of kernels and add wrappers

ICA Results of Porting

- Hybrid Python, C, OpenACC, CUDA code
- JURON faster than JURECA
  - Data transfer: $10\times$ (H2D), $6\times$ (D2H)
  - Compute: $7\times$
- Still many parts of program CPU-only $\rightarrow$ limited possible speed-up, data transfer overheads
- Benefit from Hackathon:
  - Create Python interface
  - Speak to experts on libraries
  - Write CUDA prototype
  - Formulate plan for future
- Unfortunately, code is currently rolled back to serial version to fix communication errors

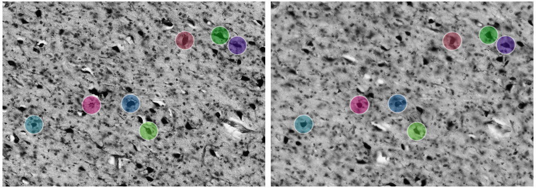




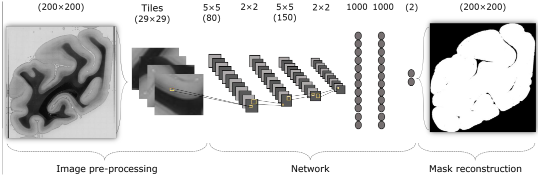
- Eventually: Hybrid code with many programming languages and programming models
- Test-run of C kernel on JURON and JURECA: JURON usually faster
- Still, many parts need to be ported to GPU to benefit from fewer memory copies
- Currently: Fixing MPI deadlock bugs in serial version

Other HBP Applications on JURON

- Only three applications highlighted (TVB-HPC, Arbor, PLI ICA)
- Many applications more on JURON!
  - 2D $\rightarrow$ 3D image registration
  - Multi-scale brain image stitching
  - Pattern recognition



Huysegoms (2017)



Khalid (2017)

Conclusions & Summary

- Brain research exciting also for computational science
- Minsky system **JURON** deployed as pilot supercomputer for Human Brain Project
- System under intensive use (not only by HBP users)
- TVB-HPC and Arbor: Two brain simulation applications operating on different scales
- PLI ICA: Cleanup of scanned images
- Many applications benefit from GPU (and also NVLink)

→ HBP helps to drive development of future supercompu

Thank you for your attention!

a.herten@fz-juelich.de

Appendix

Glossary & References

Glossary I

JÜLICH FORSCHUNGSZENTRUM

API A programmatic interface to software by well-defined functions. Short for application programming interface. 41

Arbor Multi-compartment simulation of neural networks, previously called *NestMC*. 2, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 56, 57

BSC Barcelona Supercomputing Center, a Spanish supercomputing site. 12, 39

CINECA An Italian consortium of universities operating supercomputers. 12

CSCS The national supercomputing centre of Switzerland. 12, 39

CUDA Computing platform for GPUs from NVIDIA. Provides, among others, CUDA C/C++. 19, 29, 31, 33, 39, 41, 52, 54, 60

Glossary II

JÜLICH FORSCHUNGSZENTRUM

DSL A Domain-Specific Language is a specialization of a more general language to a specific domain. 31

EPFL École Polytechnique Fédérale de Lausanne, Switzerland. 12

JSC Jülich Supercomputing Centre, the supercomputing institute of Forschungszentrum Jülich, Germany. 12, 19, 39, 60

JURECA A multi-purpose supercomputer with 1800 nodes at JSC. 33, 34, 54

JURON One of the HBP pilot system in Jülich; name derived from Juelich and Neuron. 19, 27, 28, 45, 54, 56, 57

KIT Karlsruhe Institute of Technology, Germany. 12

Glossary III

JÜLICH FORSCHUNGSZENTRUM

MPI The Message Passing Interface, a API definition for multi-node computing. 41

NVIDIA US technology company creating GPUs. 33, 34, 60

NVLink NVIDIA's communication protocol connecting CPU $\leftrightarrow$ GPU and GPU $\leftrightarrow$ GPU with 80 GB/s. PCI-Express: 16 GB/s. 57, 60

OpenACC Directive-based programming, primarily for many-core machines. 52, 54

OpenCL The *Open Computing Language*. Framework for writing code for heterogeneous architectures (CPU, GPU, DSP, FPGA). The alternative to CUDA. 31

P100 A large GPU with the Pascal architecture from NVIDIA. It employs NVLink as its interconnect and has fast *HBM2* memory. 43

Glossary IV

JÜLICH FORSCHUNGSZENTRUM

Pascal GPU architecture from NVIDIA (announced 2016). 60

RWTH RWTH Aachen University, Germany. 12

Tesla The GPU product line for general purpose computing computing of NVIDIA. 33, 34

TVB-HPC High-Performance Computing sub-project of The Virtual Brain. 29, 30, 31, 32, 33, 34, 35, 56, 57

CPU Central Processing Unit. 33, 34, 43, 54, 60

GPU Graphics Processing Unit. 19, 20, 29, 33, 34, 41, 43, 45, 52, 57, 60

HBP Human Brain Project. 7, 8, 19, 56, 57, 60

Glossary V

ICA Independent Component Analysis. 48, 49, 50, 51, 52, 53, 54, 55, 56, 57

PLI Polarized Light Imaging. 48, 49, 50, 51, 56, 57

TVB The Virtual Brain. 25, 26, 27, 28, 37, 60

References: Images, Graphics I

JÜLICH FORSCHUNGSZENTRUM

[1] T. Willis. *Cerebri anatome: cui accessit nervorum descriptio et usus*. Typis Tho. Roycroft, impensis Jo. Martyn & Ja. Allestry, 1664. URL: https://archive.org/details/cerebrianatomecu00will (pages 3, 4).

[2] S. Ramón y Cajal. *Estructura de los centros nerviosos de las aves*. 1888. URL: https://commons.wikimedia.org/wiki/File:SparrowTectum.jpg (page 3).

[4] H. Gray and W.H. Lewis. *Anatomy of the Human Body*. Lea & Febiger, 1918. URL: http://www.bartleby.com/107/illus739.html (page 3).

[5] Richard Frackowiak and Henry Markram. "The future of human cerebral cartography: a novel approach". In: *Phil. Trans. R. Soc. B* 370.1668 (2015), p. 20140171. DOI: 10.1098/rstb.2014.0171 (page 5).

References: Literature I

[3] A. L. Hodgkin, A. F. Huxley, and B. Katz. "Measurement of current-voltage relations in the membrane of the giant axon ofLoligo". In: *The Journal of Physiology* 116.4 (Apr. 1952), pp. 424–448. DOI: 10.1113/jphysiol.1952.sp004716. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392219/ (pages 3, 37).

[6] Paula Sanz Leon et al. "The Virtual Brain: a simulator of primate brain network dynamics". In: *Frontiers in Neuroinformatics* 7 (2013). DOI: 10.3389/fninf.2013.00010 (page 25).

[7] V.K. Jirsa et al. "The Virtual Epileptic Patient: Individualized whole-brain models of epilepsy spread". In: *NeuroImage* 145 (Jan. 2017), pp. 377–388. DOI: 10.1016/j.neuroimage.2016.04.049. URL: http://www.sciencedirect.com/science/article/pii/S1053811916300891 (page 27).

References: Literature II

JÜLICH FORSCHUNGSZENTRUM

[8] Markus Axer et al. “A novel approach to the human connectome: Ultra-high resolution mapping of fiber tracts in the brain”. In: *NeuroImage* 54.2 (Jan. 2011), pp. 1091–1101. DOI: 10.1016/j.neuroimage.2010.08.075. URL: http://www.sciencedirect.com/science/article/pii/S105381191001178X (page 48).

[9] Jürgen Dammers et al. “Automatic identification of gray and white matter components in polarized light imaging”. In: *NeuroImage* 59.2 (Jan. 2012), pp. 1338–1347. DOI: 10.1016/j.neuroimage.2011.08.030. URL: http://www.sciencedirect.com/science/article/pii/S1053811911009232 (page 50).