BioDynaMo

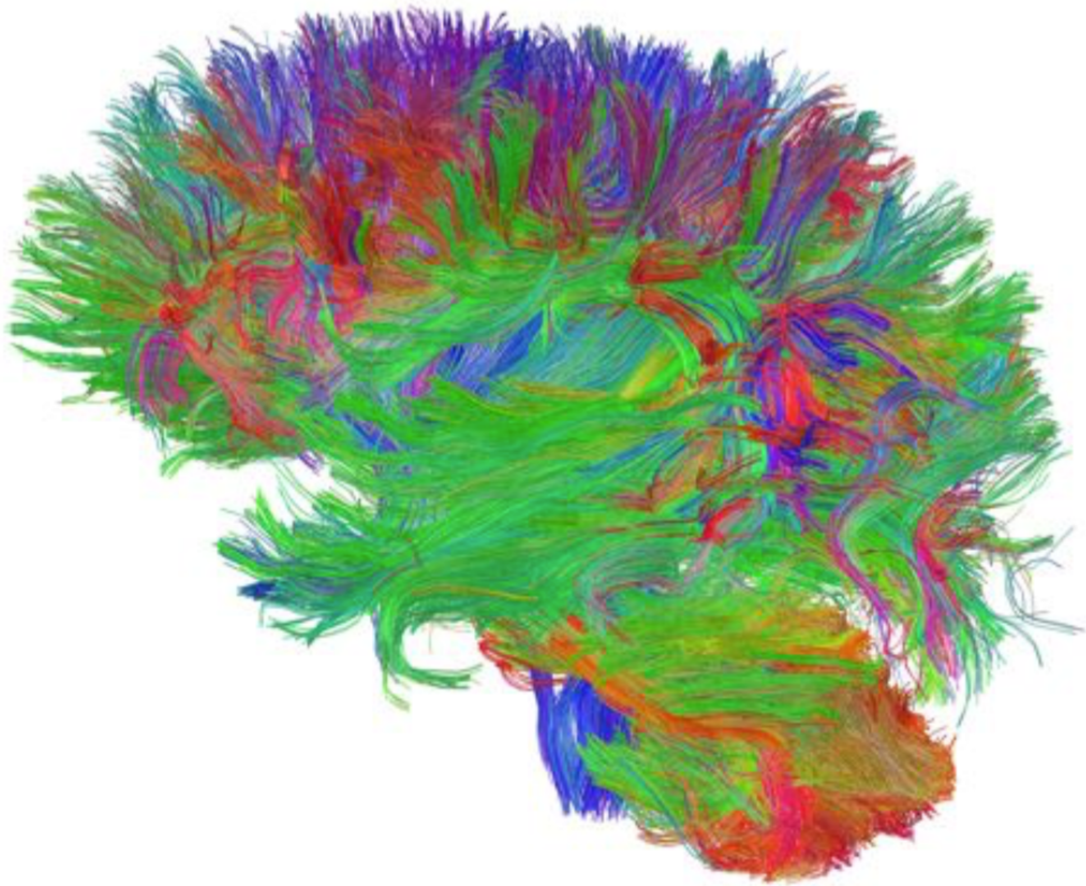
Biology Dynamics Modeller

CERN openlab Technical Workshop

Fons Rademakers

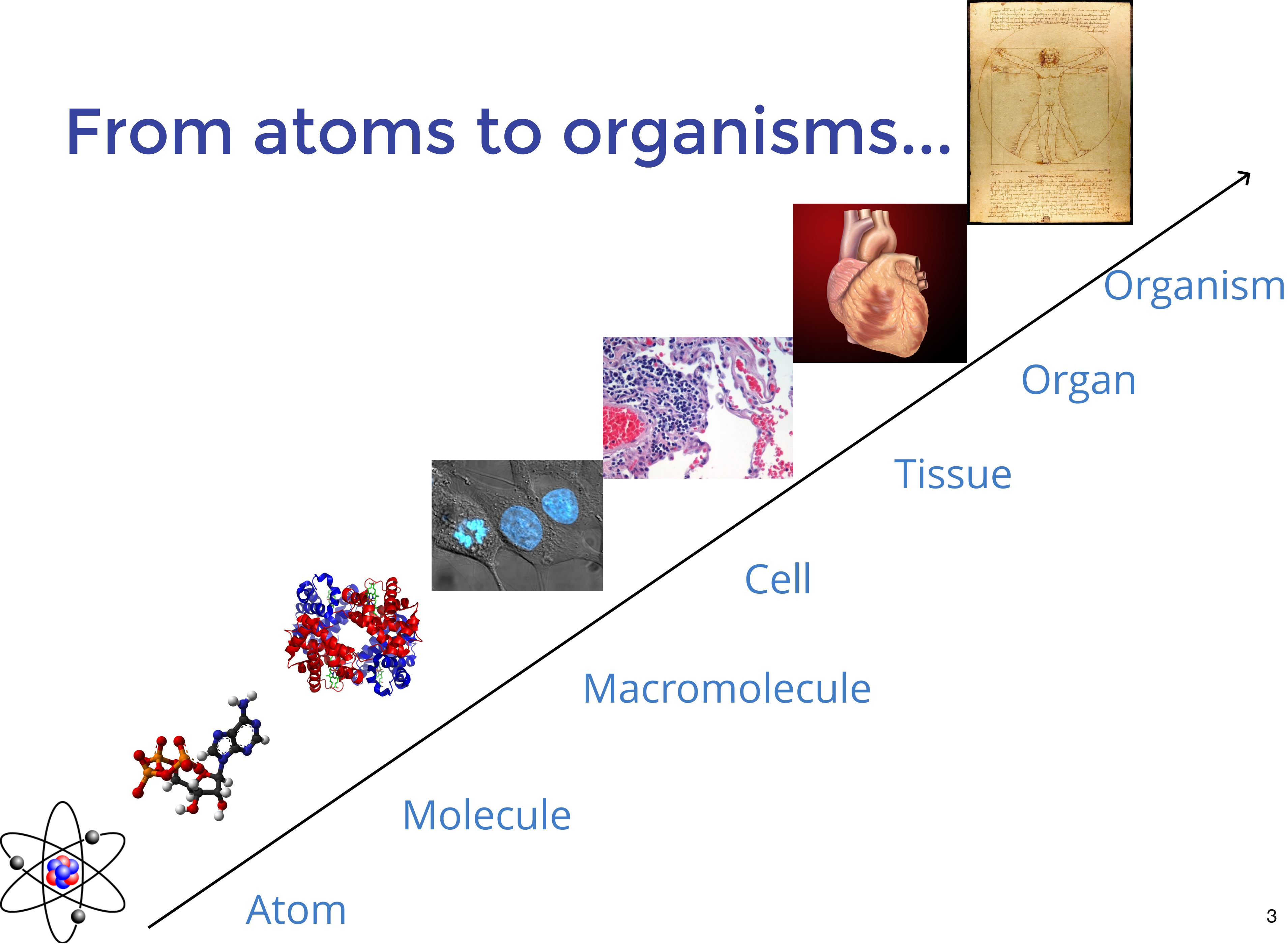
Lukas Breitwieser

Help Life Scientists Understand (Patho)physiological Processes

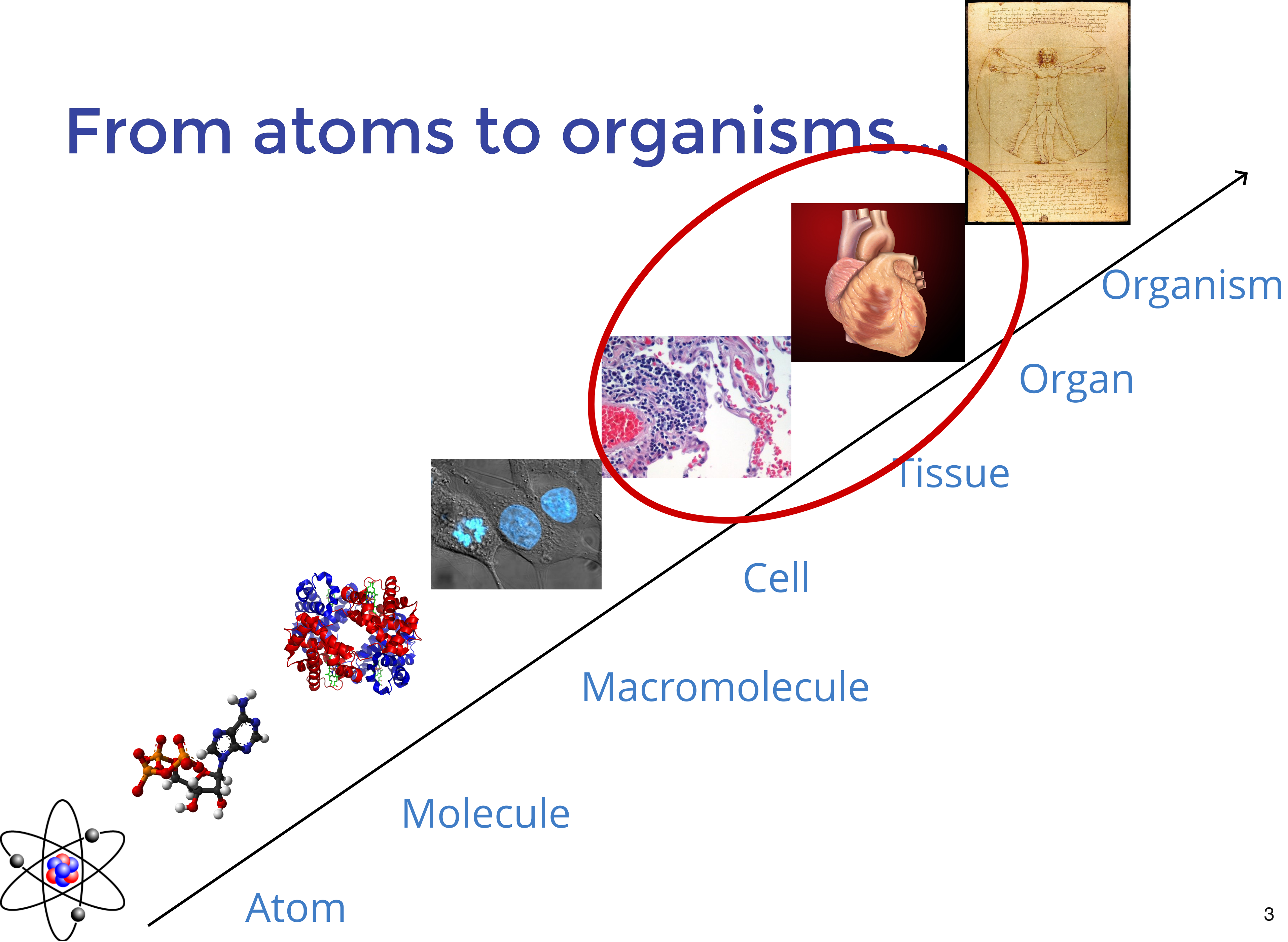


Kaiser; University of Newcastle, UK; www.dynamic-connectome.org

From atoms to organisms...

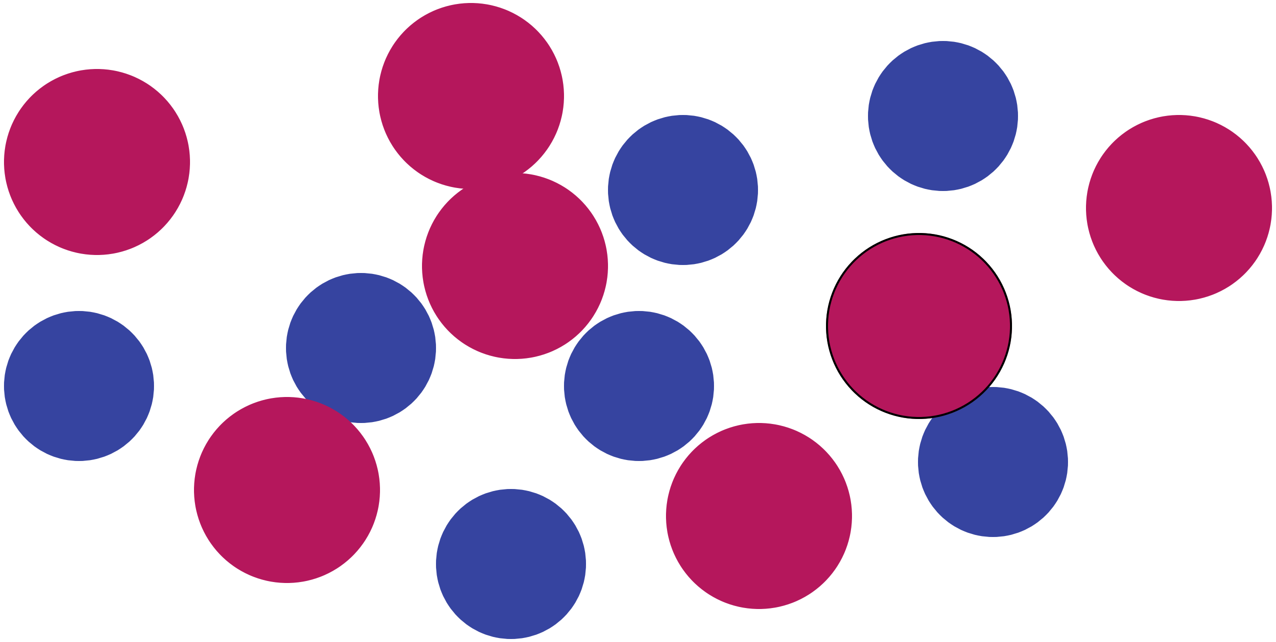


From atoms to organisms…



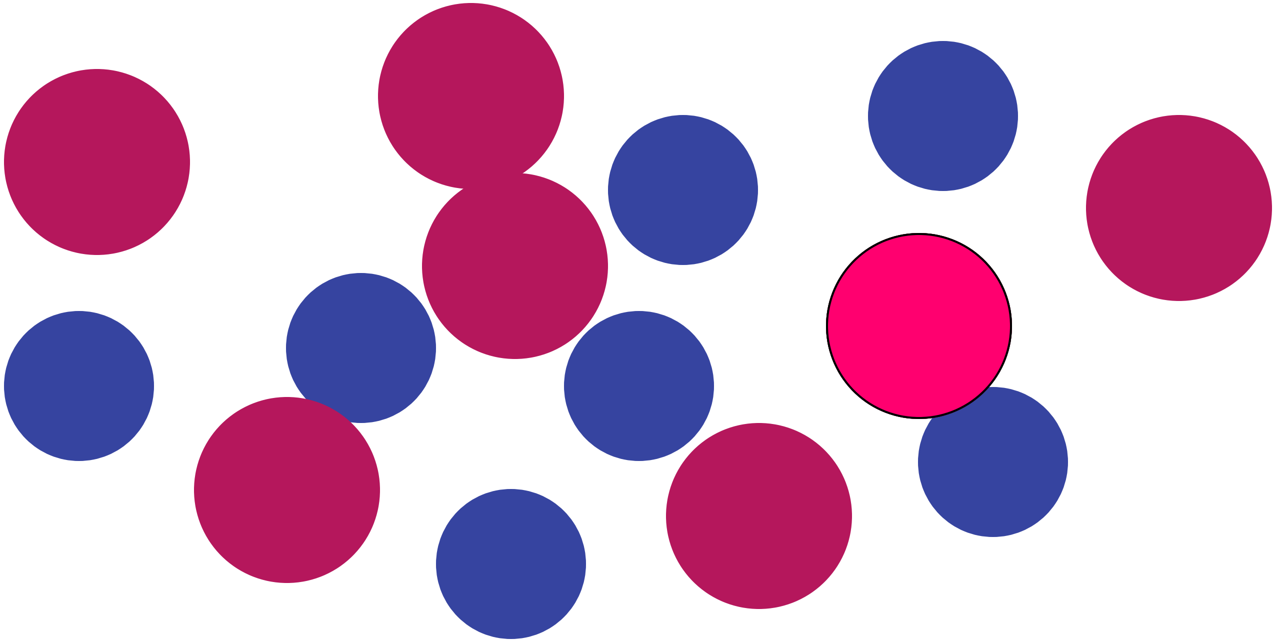
Agent-based simulations

Simulation object = *Agent*



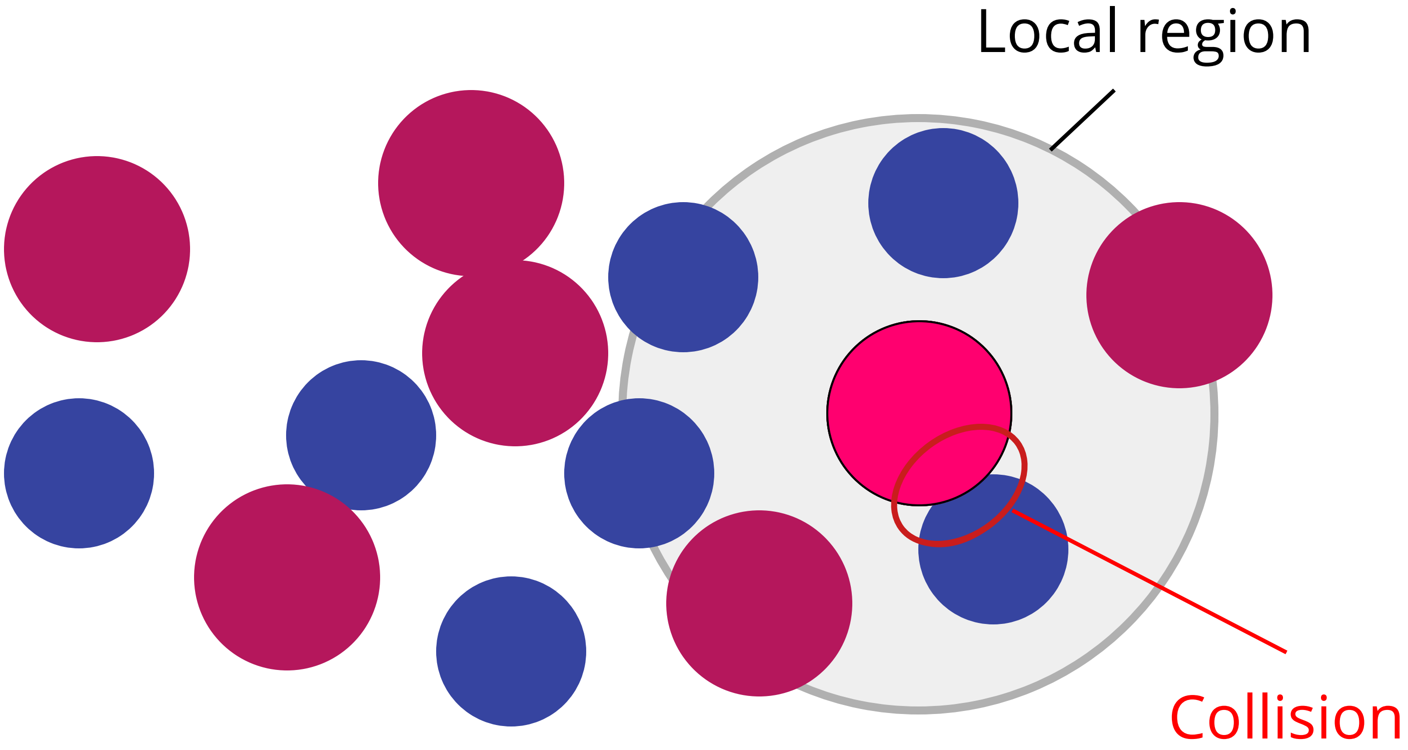
Agent-based simulations

Simulation object = *Agent*

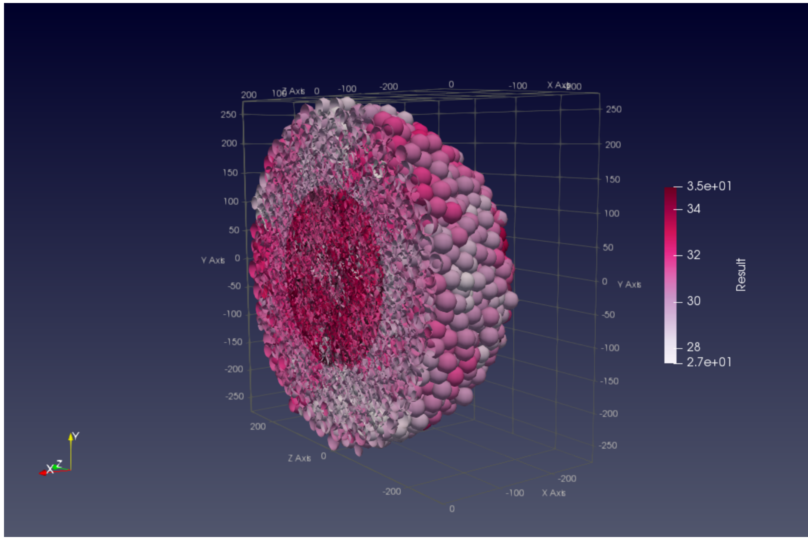


Agent-based simulations

Simulation object = *Agent*

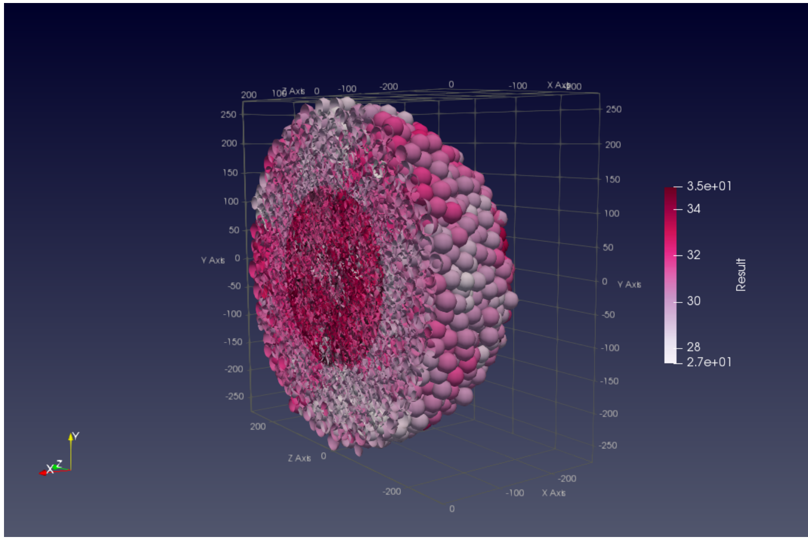


BioDynaMo Design Goals



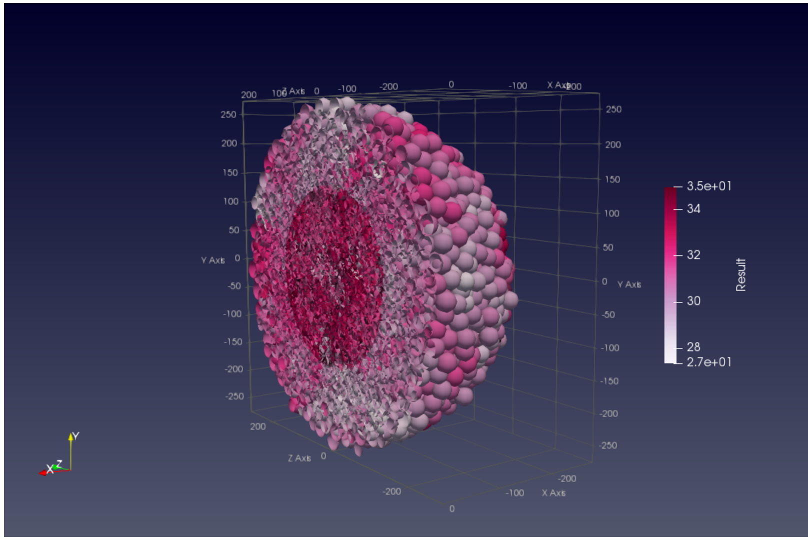
BioDynaMo Design Goals

- Modular system that supports different specialities



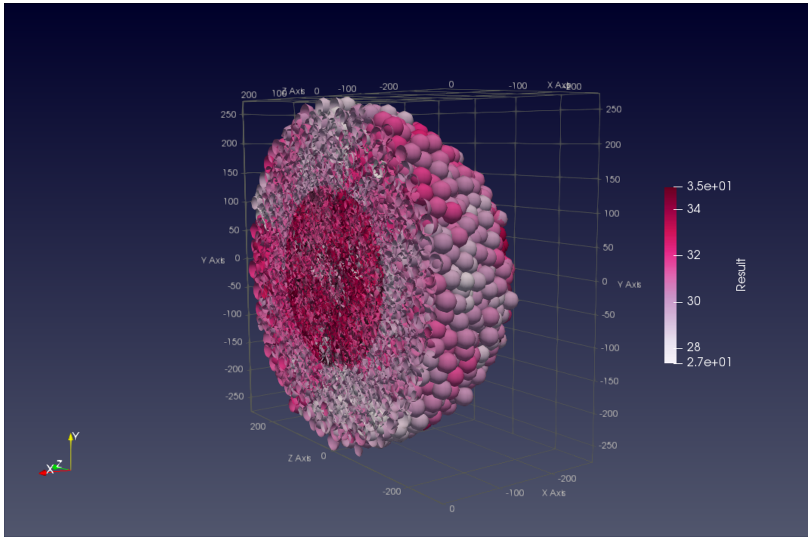
BioDynaMo Design Goals

- Modular system that supports different specialities
- Support large-scale biological simulations



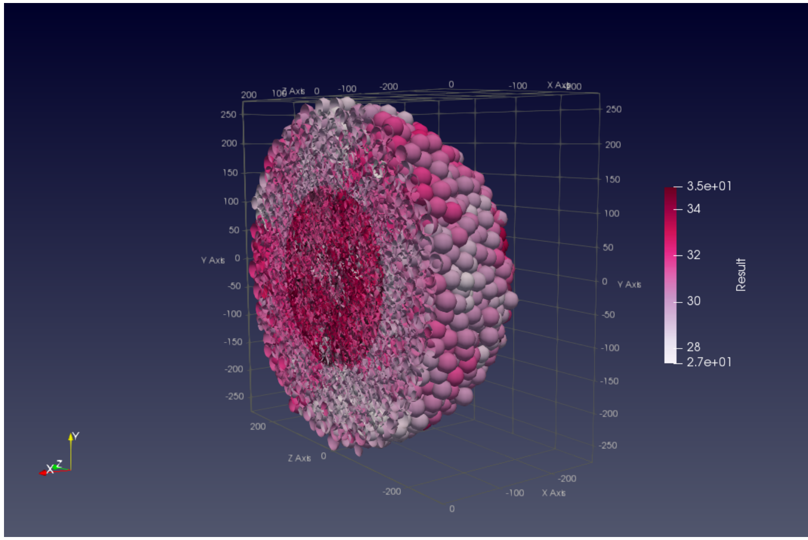
BioDynaMo Design Goals

- Modular system that supports different specialities
- Support large-scale biological simulations
- Hide complexity of distributed computing



BioDynaMo Design Goals

- Modular system that supports different specialities
- Support large-scale biological simulations
- Hide complexity of distributed computing
- Promote reproducibility of results



Large-scale Simulations

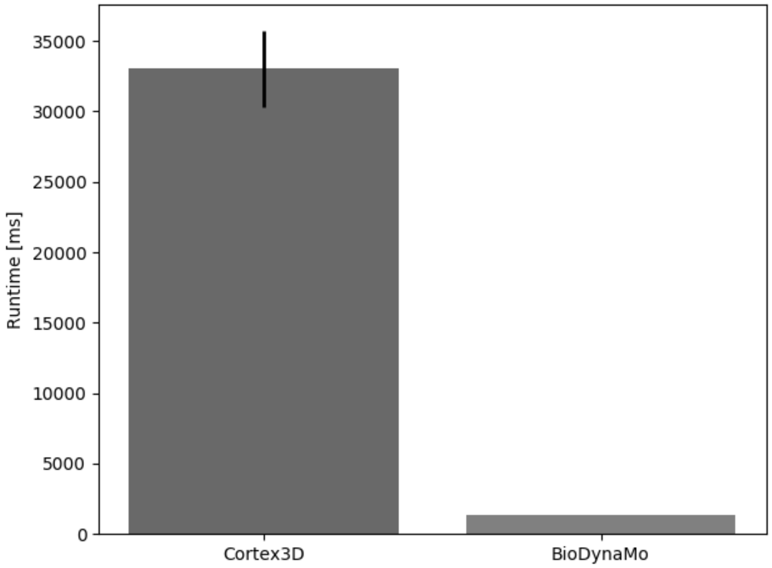
The cerebral cortex consists of ~16 billion neurons

- **Scale up**
  - Efficient use of modern hardware (multi-core CPUs and accelerators)
- **Scale out**
  - Distributed runtime

Preliminary Performance Results

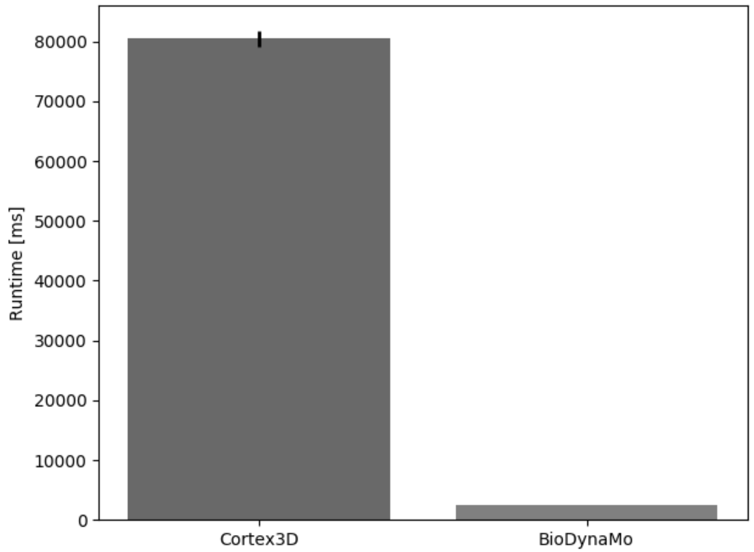
Cell grow and divide

Speedup: 25x



Soma clustering

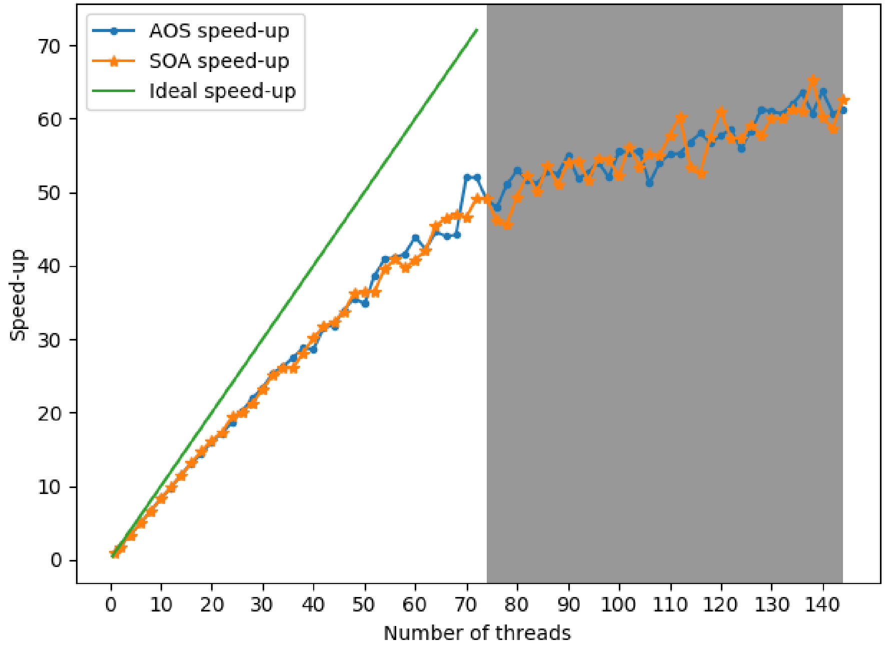
Speed-up: 31x



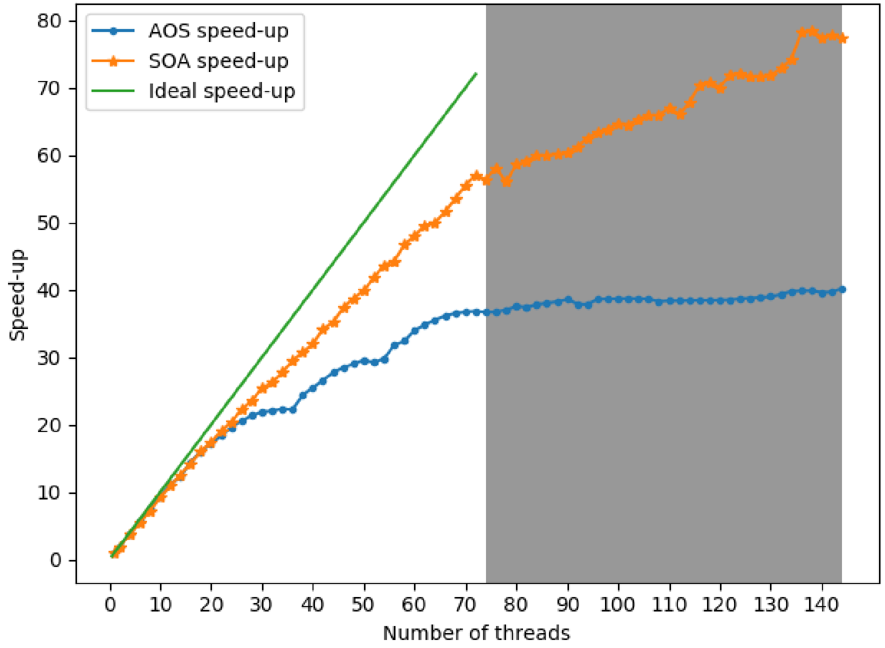
Preliminary Performance Results

AOS vs SOA memory layout

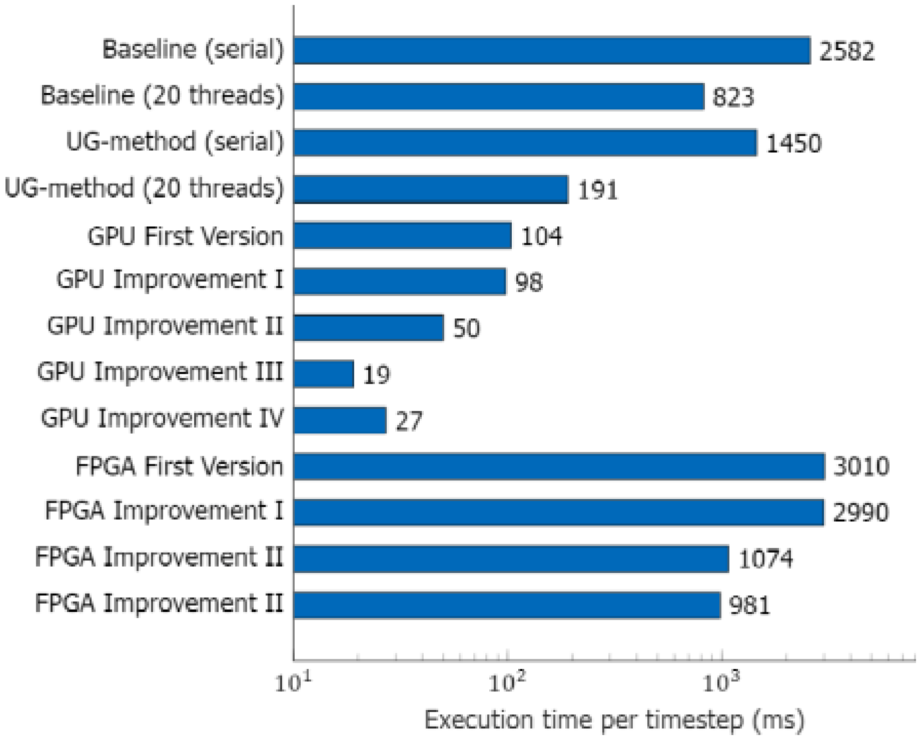
cell grow and divide

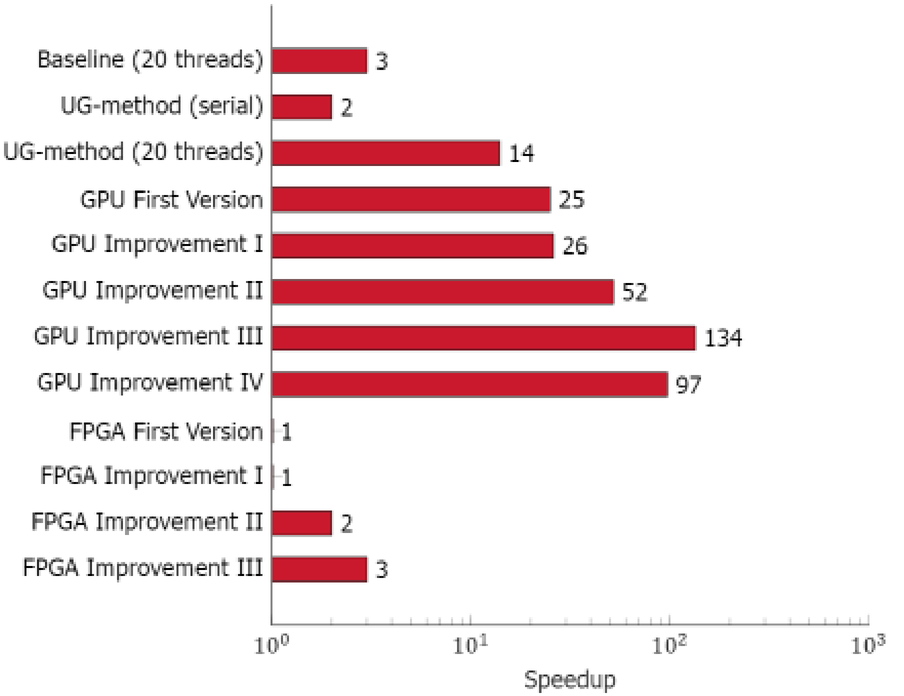


soma clustering

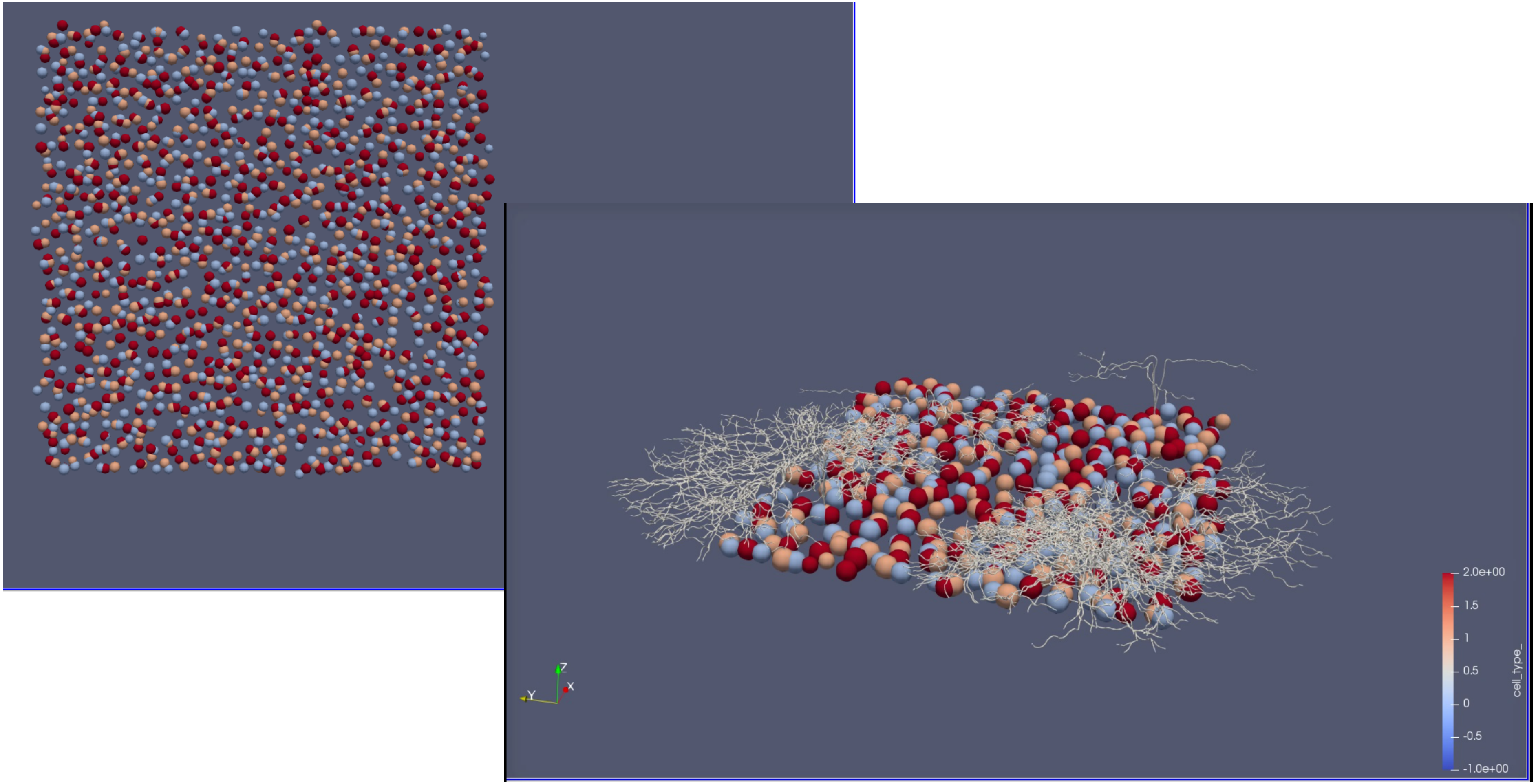


Mechanical Interactions on GPU or FPGA





Retinal Mosaics Use Case



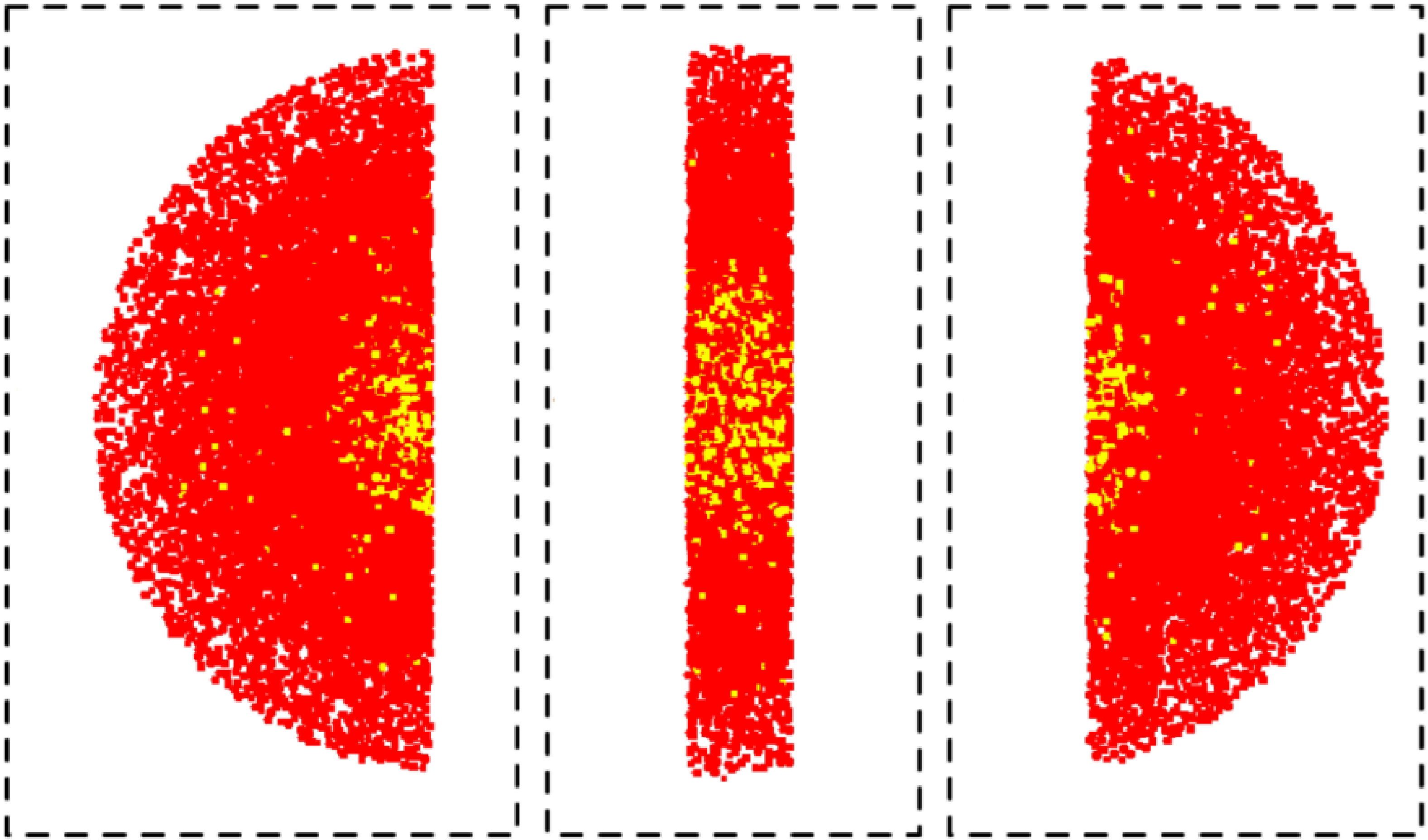
Research performed by Jean de Montigny at the University of Newcastle, UK

Current Status

- **Modular simulation engine**
- **Fully parallelized** with OpenMP
- **GPU & FPGA** implementation for mechanical interactions using CUDA and OpenCL
- First version of **distributed runtime** based on the framework Ray
- ROOT I/O for storage of simulation results and snapshots
- **Visualization** using ParaView and ROOT Eve

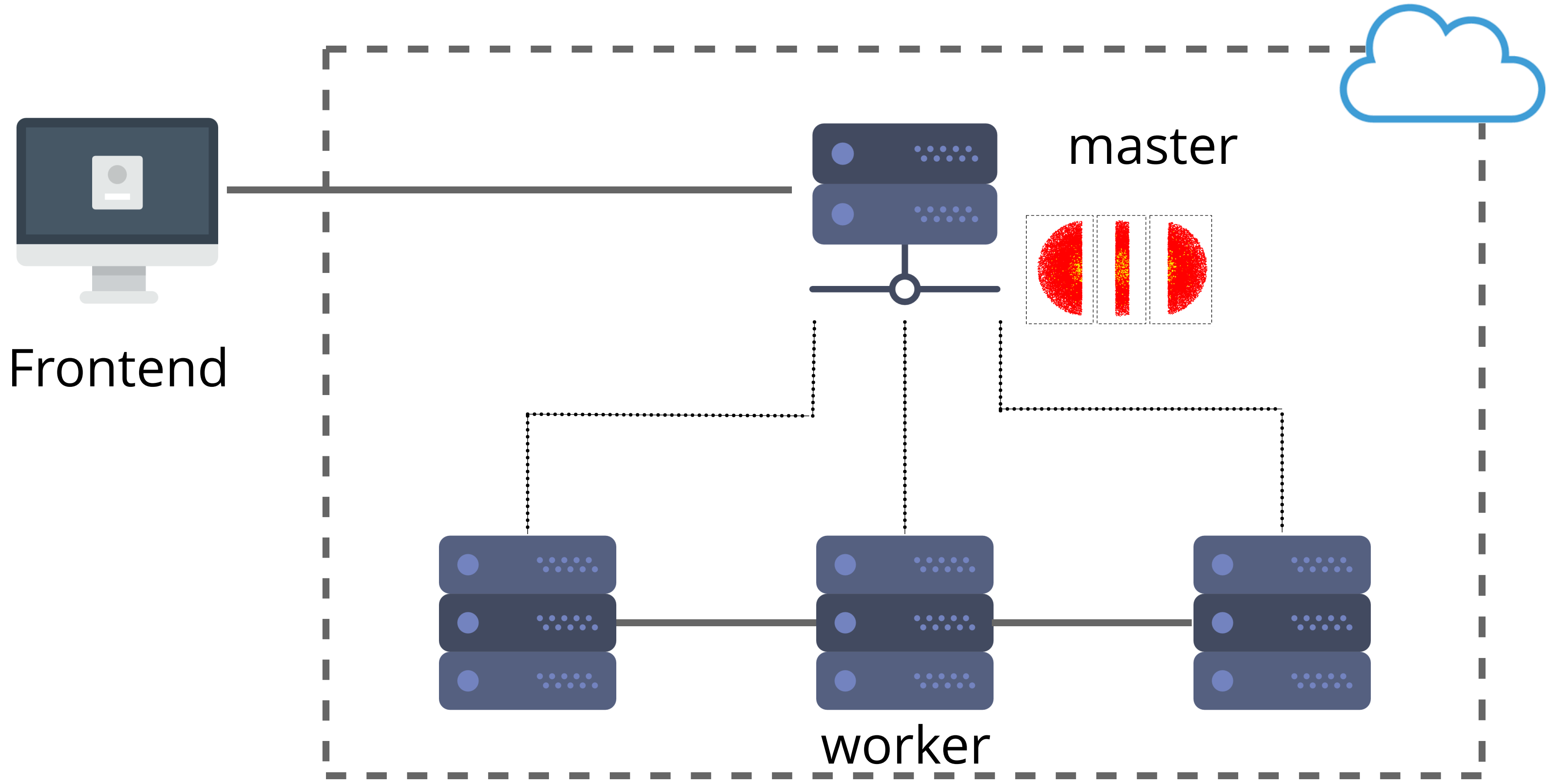
Future Work:
Distributed Runtime

Domain-Decomposition

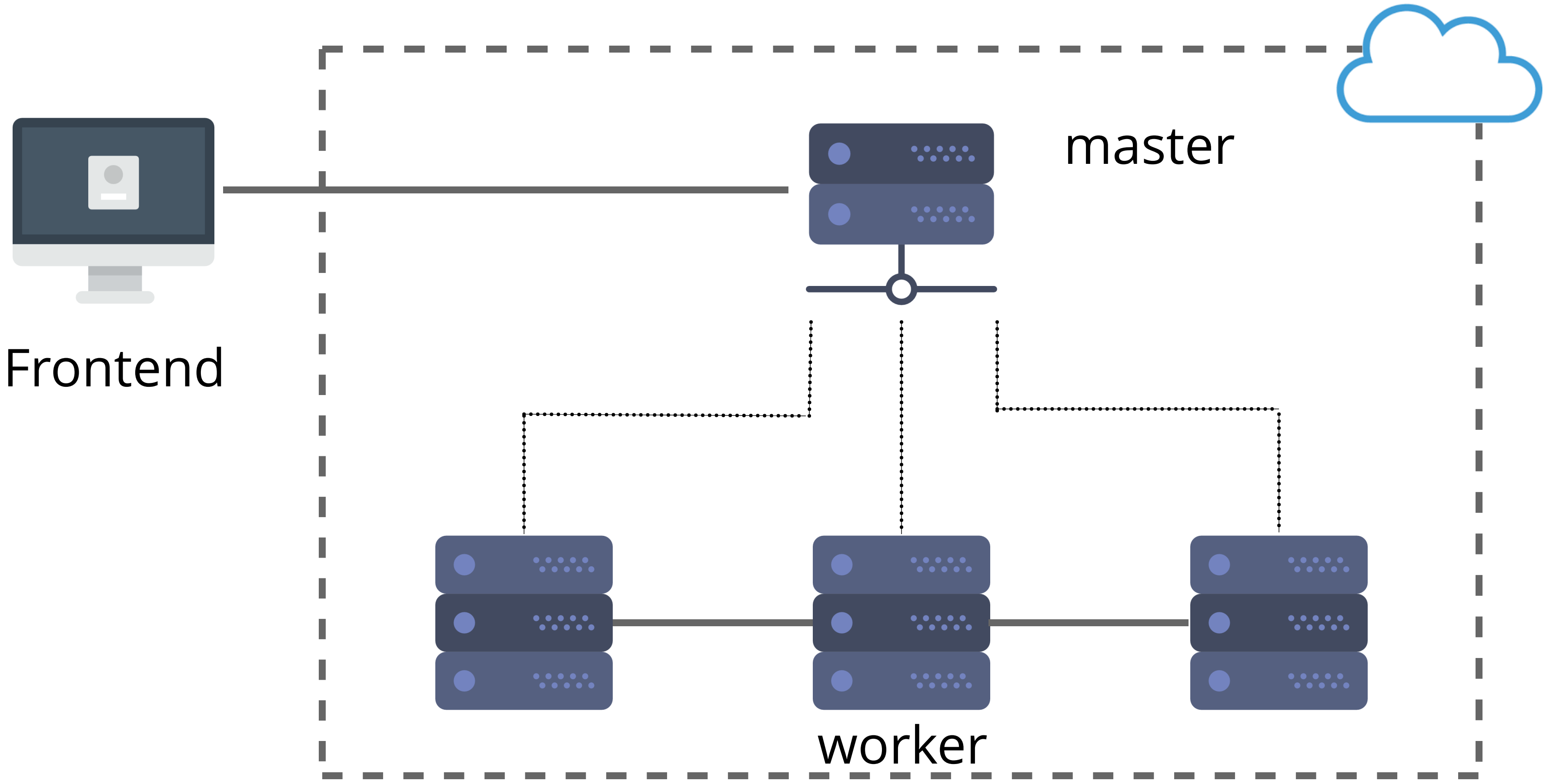


Hauri, Andreas. *Self-construction in the context of cortical growth* . Diss. 2013.

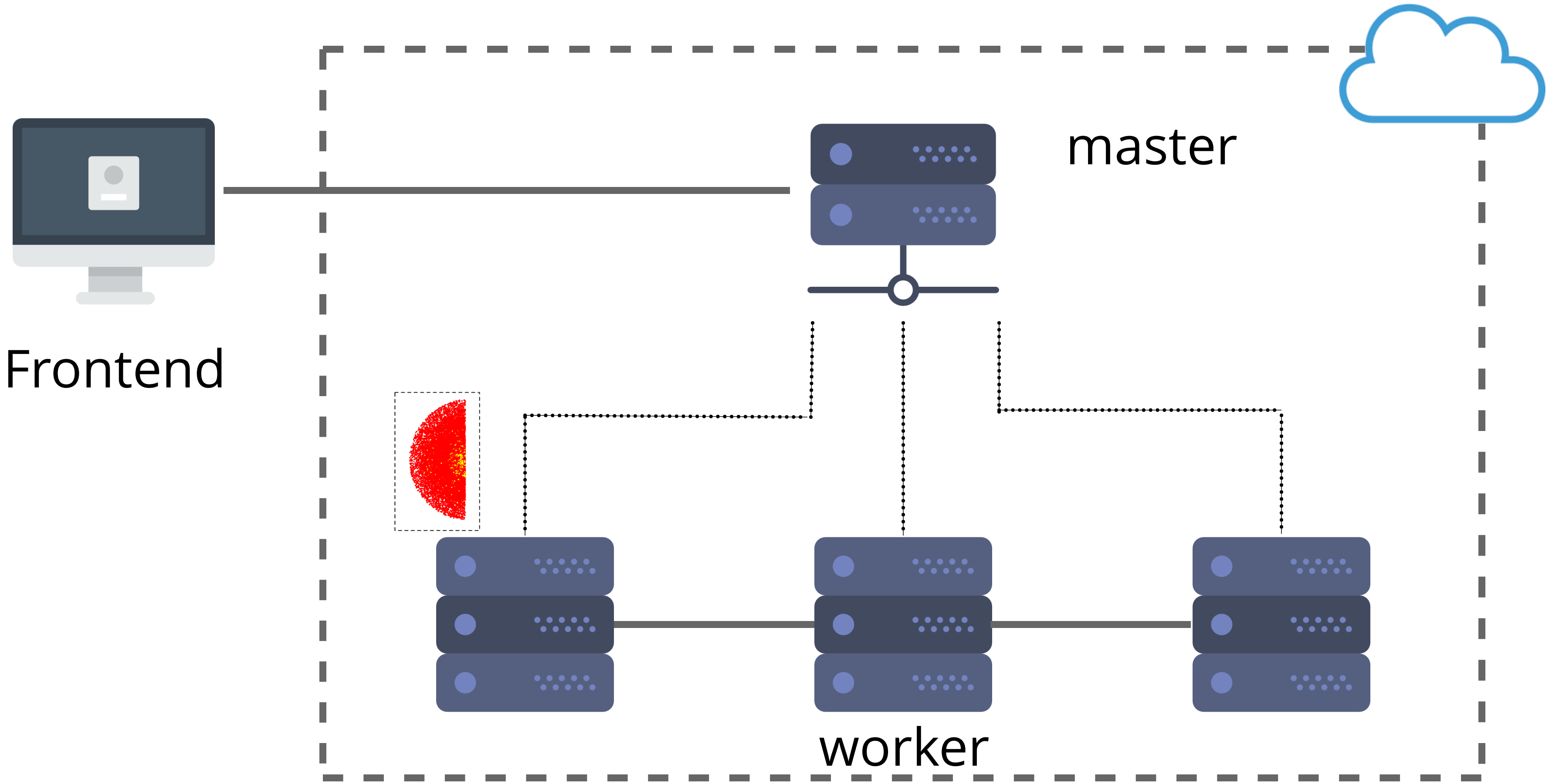
Distributed Runtime



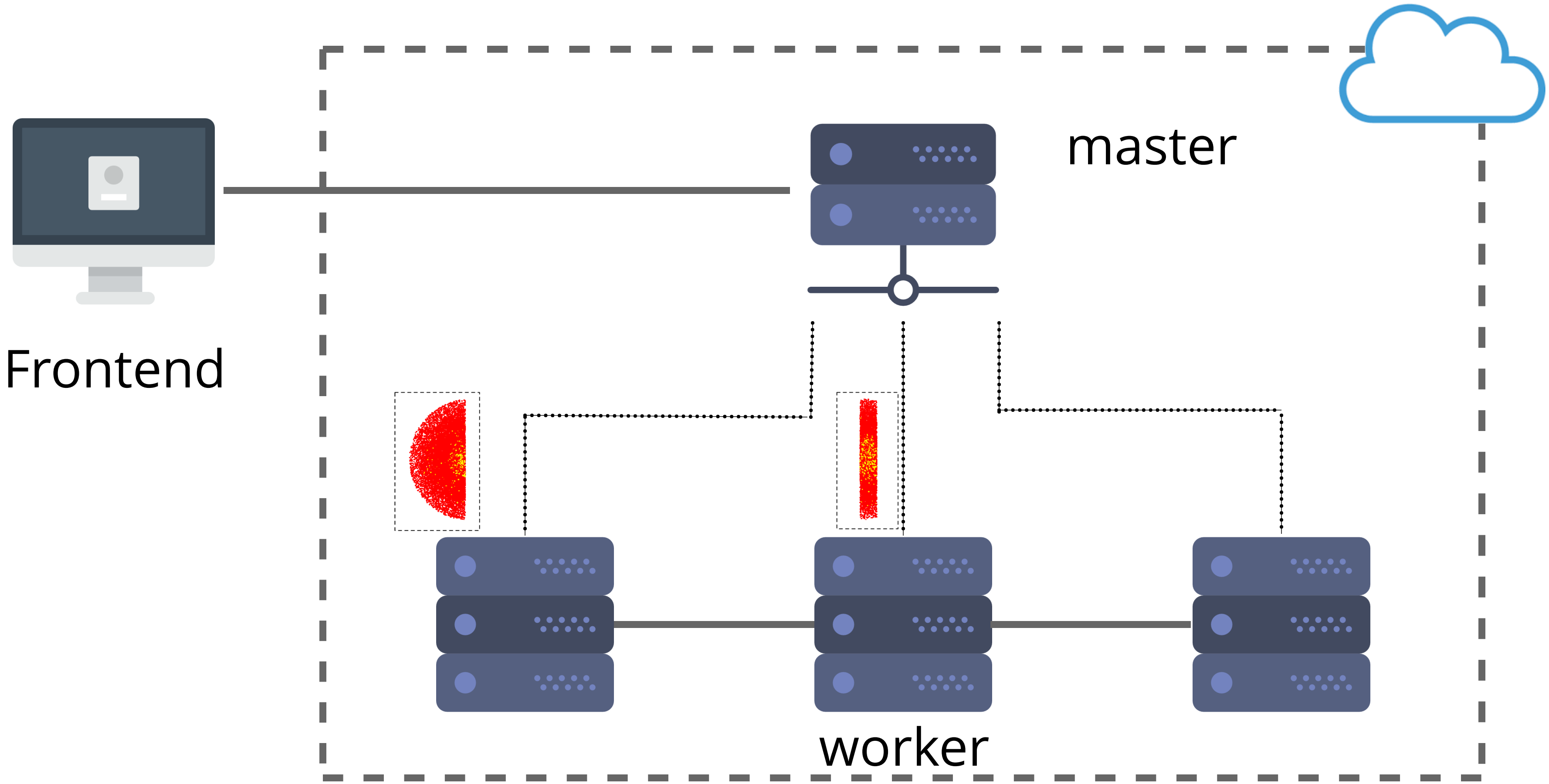
Distributed Runtime



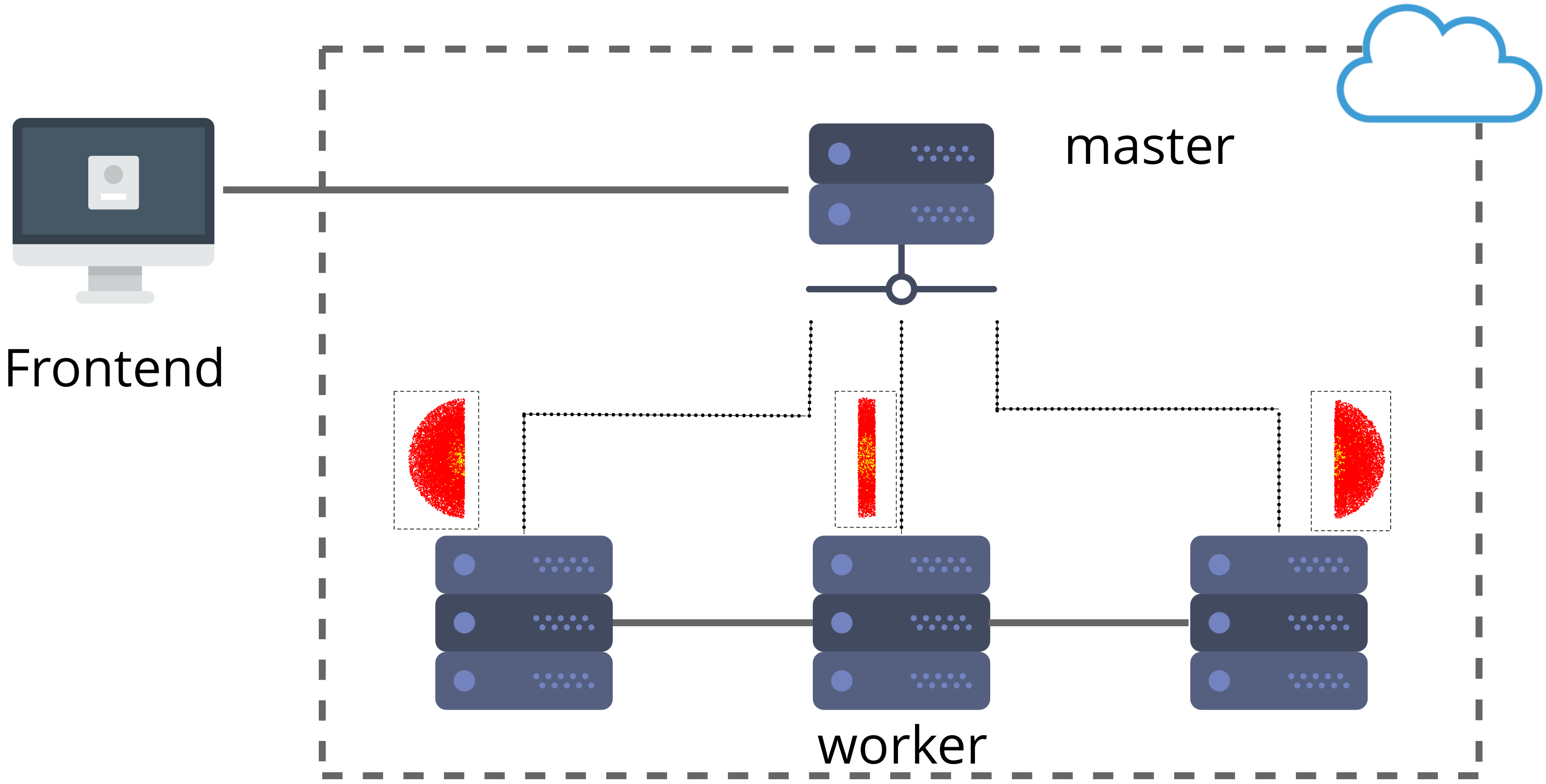
Distributed Runtime



Distributed Runtime



Distributed Runtime



Prototype based on Ray

riselab UC Berkeley

PEOPLE PROJECTS PUBLICATIONS SPONSORS ACADEMICS NEWS EVENTS BLOG JENKINS

Ray: A Distributed Framework for Emerging AI Applications

ROBERT NISHIHARA / DECEMBER 20, 2017 /

The next generation of AI applications will continuously interact with the environment and learn from these interactions. These applications impose new and demanding systems requirements, both in terms of performance and flexibility. In this paper, we consider these requirements and present Ray---a distributed system to address them. Ray implements a dynamic task graph computation model that supports both the task-parallel and the actor programming models. To meet the performance requirements of AI applications, we propose an architecture that logically centralizes the system's control state using a sharded storage system and a novel bottom-up distributed scheduler. In our experiments, we demonstrate sub-millisecond remote task latencies and linear throughput scaling beyond 1.8 million tasks per second. We empirically validate that Ray speeds up challenging benchmarks and serves as both a natural and performant fit for an emerging class of reinforcement learning applications and algorithms.

Published On:

Link: https://arxiv.org/pdf/1712.05889.pdf

Authors: Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, William Paul, Michael Jordan, Ion Stoica

developed by our summer student Nam Nguyen

Challenges

- Alleviate the overheads of distributed execution
  - e.g. (De)serialization
- Performance issues in the cloud
  - Inferior network performance compared to supercomputers
  - Load balancing (heterogeneous computing, runtime variance)
- Fault-tolerance
  - Long running simulations with large number of nodes Will checkpointing be enough?

QUESTIONS?

lukas.breitwieser@cern.ch

CERN openlab