

Towards Extreme-Scale Agent-Based Simulation with BioDynaMo

Lukas Breitwieser, ACAT 2022

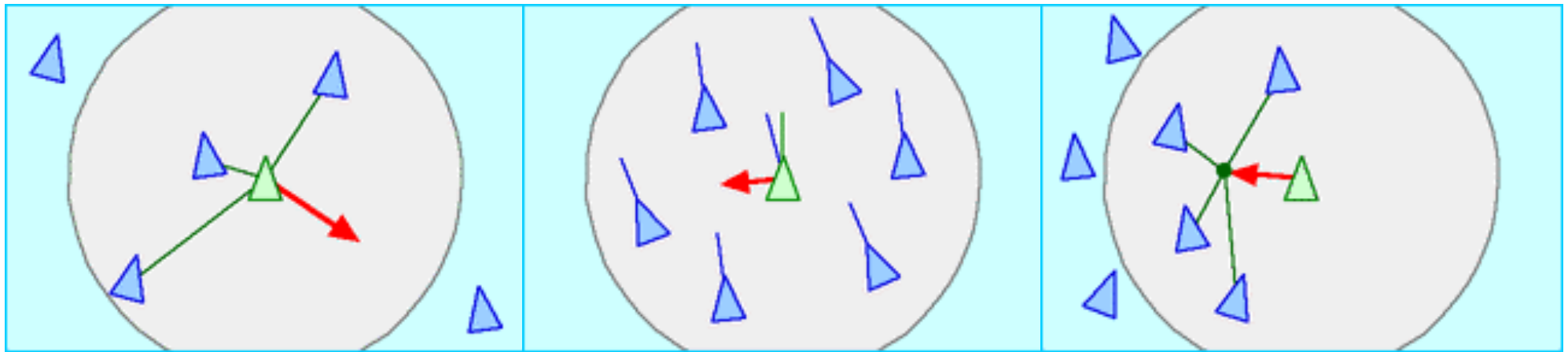
Introduction to Agent-Based Simulation

Modeling complex systems – e.g. a swarm of birds



The agent-based model

- Agent: bird
 - position
 - velocity
 - shape
- Behaviors:

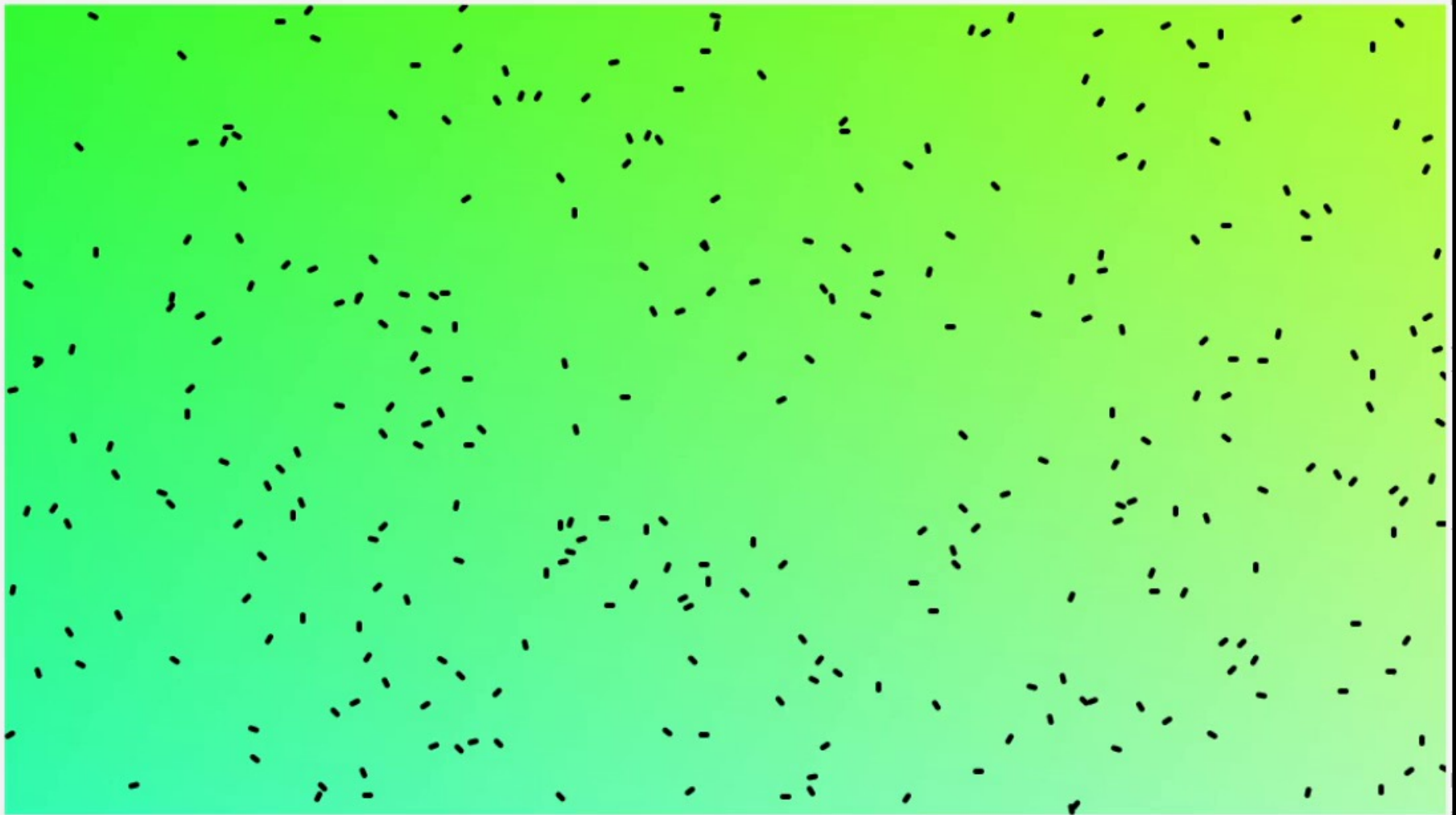


Separation

Alignment

Cohesion

Agent-based simulation

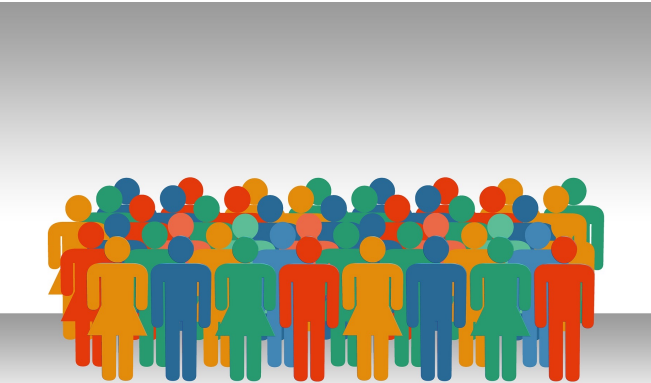
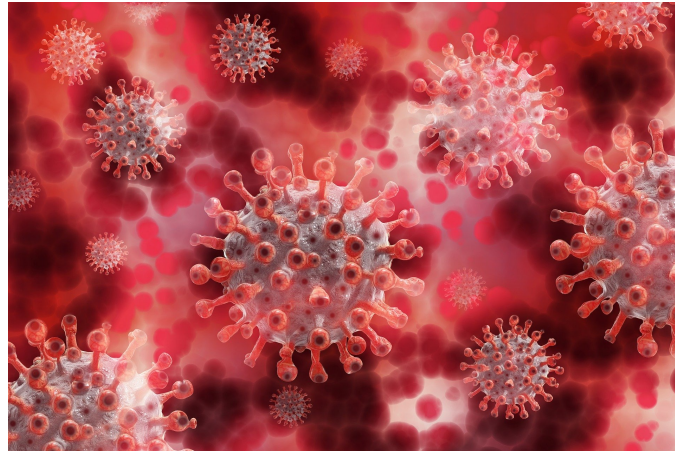
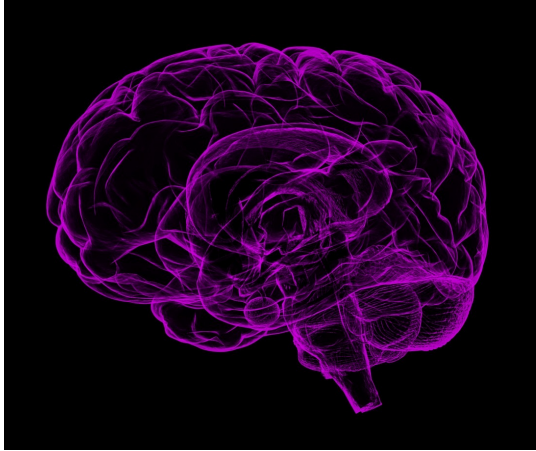


Start Step Boids Number: 300 - + ☐ Walls Background: Greenish ▾

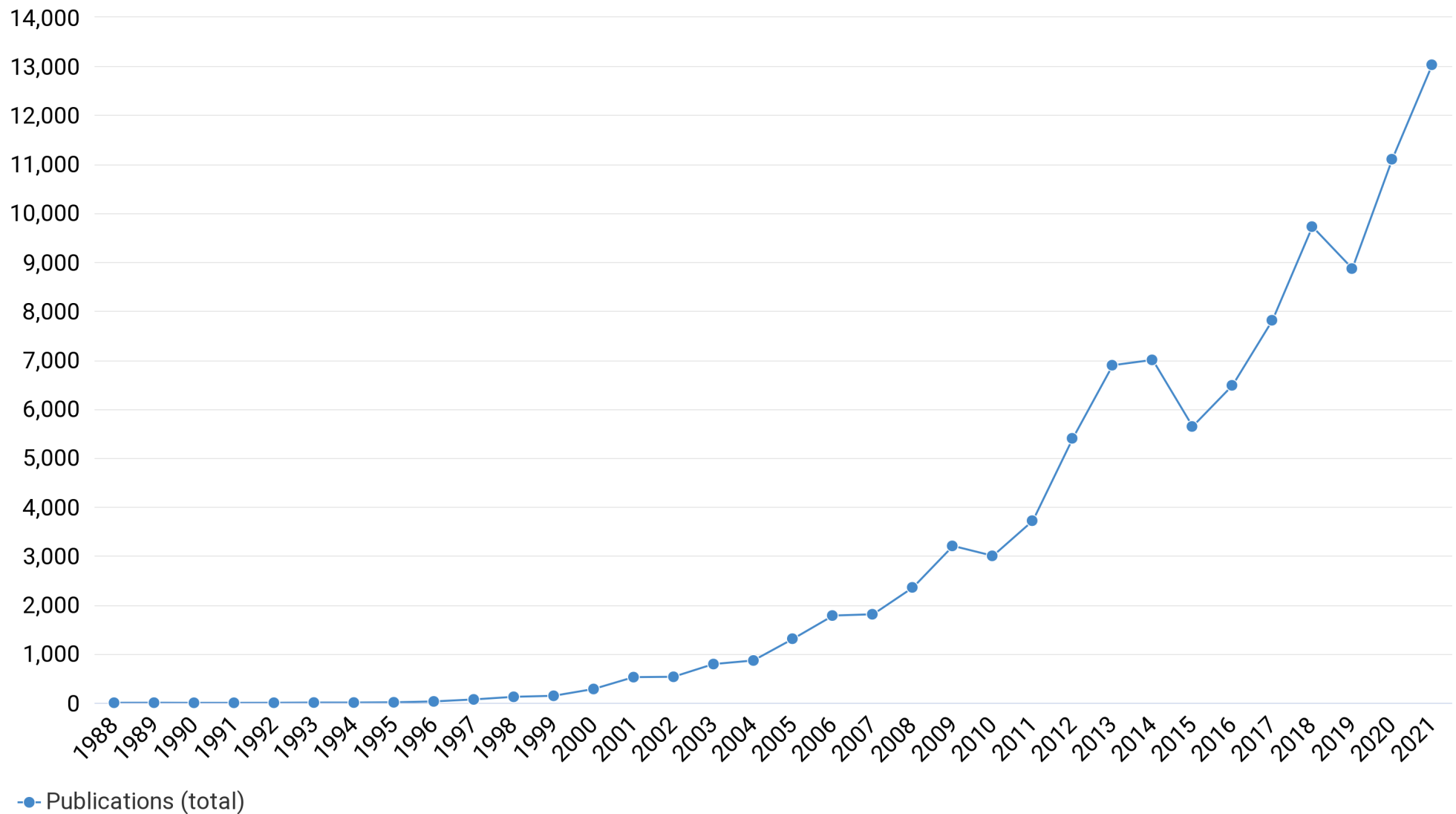
Boids Rules: ☐ Avoid ☐ Align ☐ Cohesion ☐ FoV Dead Angle 60 - + Speed: 4.5 - +

Mouse Mode: ☒ None ☐ Scary ☐ Attractive ☐ Predator

Agent-based simulation is very versatile



Rising Number of Publications in this field



Source: <https://app.dimensions.ai>

Exported: October 14, 2022

Criteria: "agent-based model" OR "agent-based simulation" in full data.

© 2022 Digital Science and Research Solutions Inc. All rights reserved. Non-commercial redistribution / external re-use of this work is permitted subject to appropriate acknowledgement. This work is sourced from Dimensions® at www.dimensions.ai.

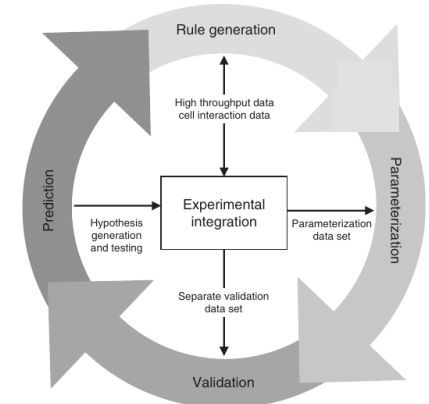
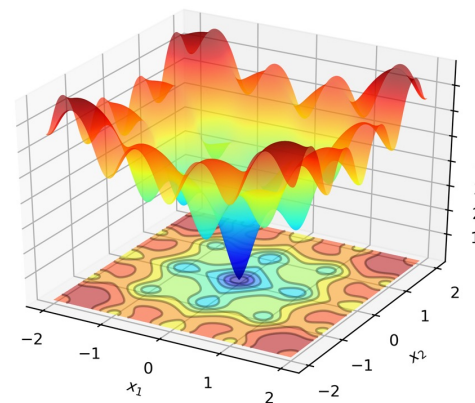
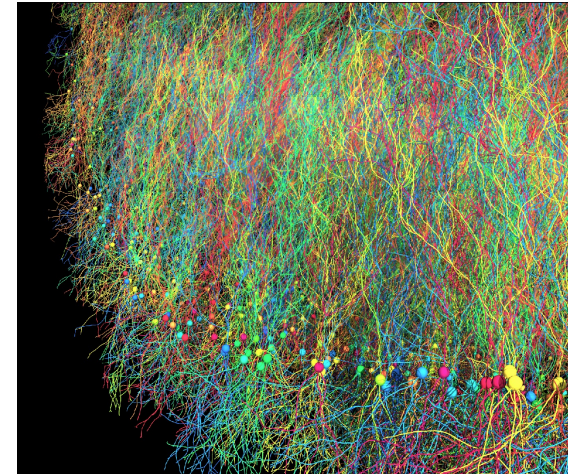
Performance considerations

The problem

Existing simulation platforms do not always take full advantage of modern hardware.

Impact of low performance

- Limitation of the size and complexity of models
- Longer development time
- Limited capability to explore parameter space
→ less optimal solution
- Increased cost



Our solution: BioDynaMo



BioDynaMo is a **modular and high-performance agent-based** simulation platform written in C++.

<https://biodynamo.org>

Developed by the BioDynaMo collaboration:



and other universities:



CERN Knowledge Transfer



Features and abstraction layers

Simulation

- **Agent geometry:** sphere, cylinder
- **Agents:** Cell, NeuronSoma, NeuriteElement
- **Behaviors:** Secretion, Chemotaxis, Proliferation, GeneRegulation
- Extracellular diffusion
- Agent interaction force

Simulation

BioDynaMo's model building blocks

- Generation of agent populations
- Agent reproduction & mortality
- Environment search
- Multi-scale simulations
- Dynamic scheduling
- Statistical analysis
- Parameter management
- Parameter optimization
- Hierarchical model support
- Hybrid-modeling
- Space boundary conditions

BioDynaMo's high-level features

- Parallelism & thread-safety
- Performance optimizations
- GPU support
- Visualization
- Web-based interface
- Backup & restore of simulations
- Quality assurance infrastructure

BioDynaMo's low-level features

OpenMP

ROOT

ParaView

Others

Libraries

Linux / MacOS

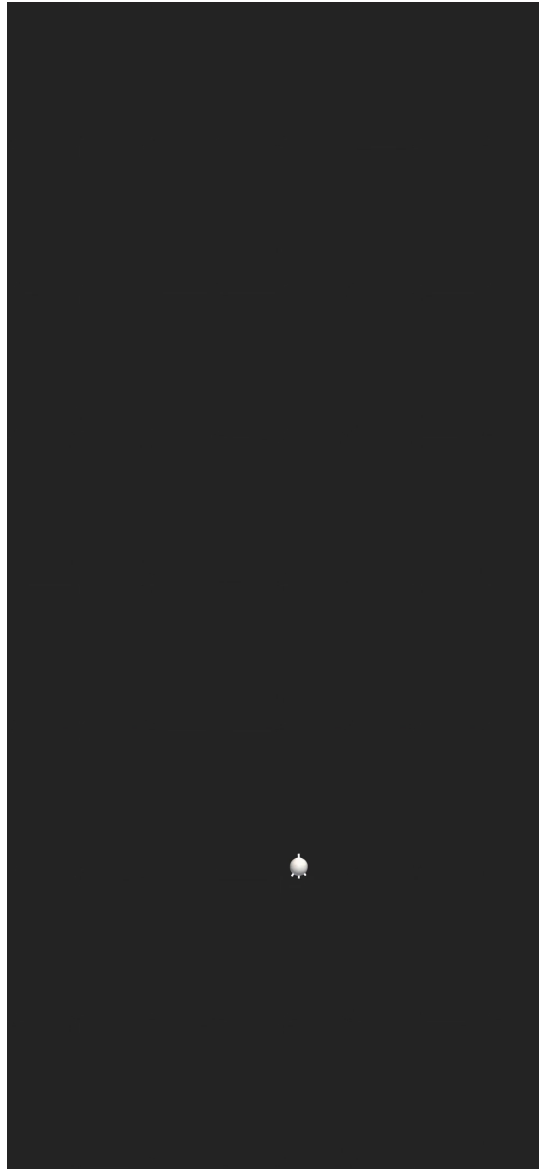
Operating System

(Multi-core) CPUs

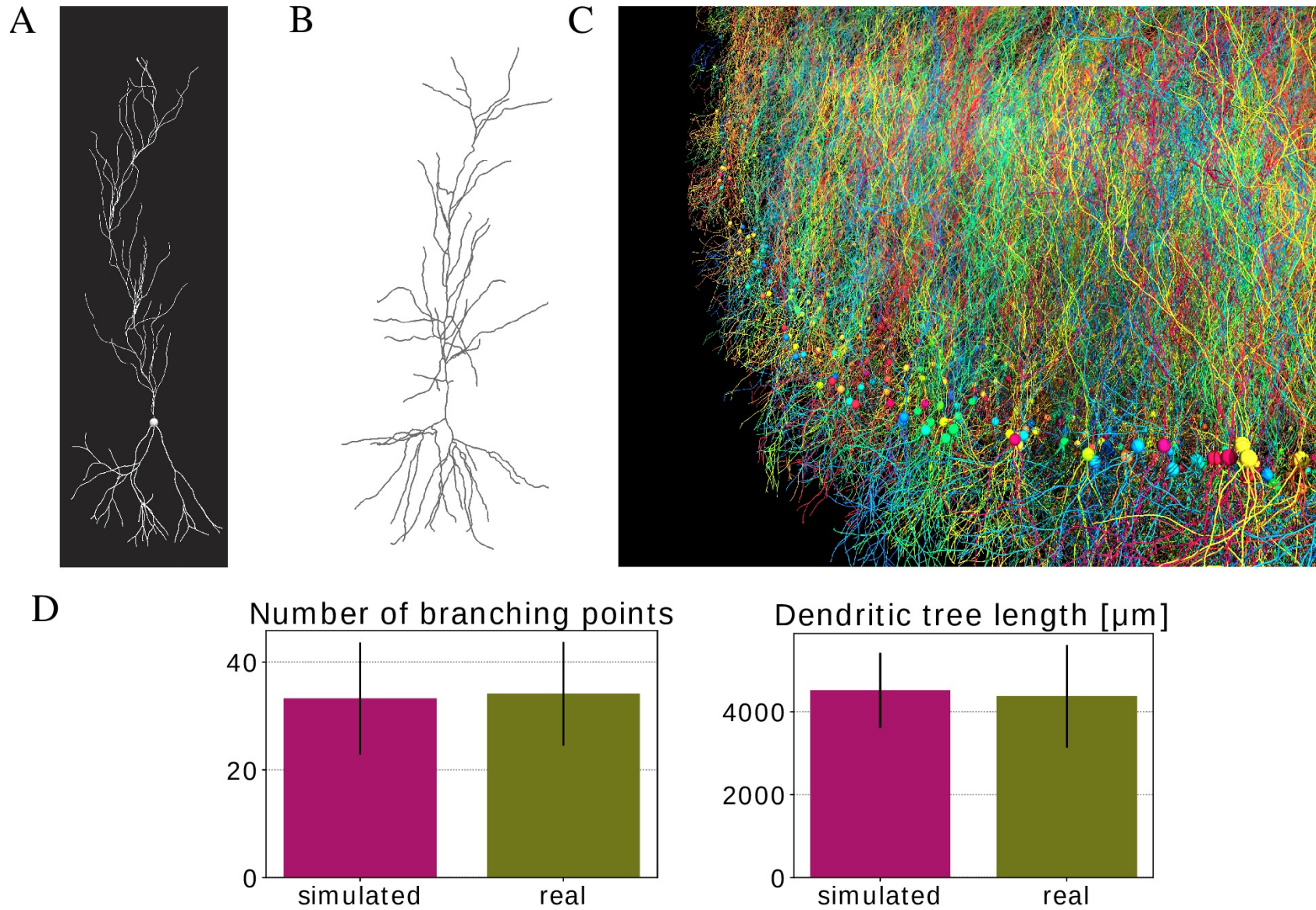
GPU

Hardware

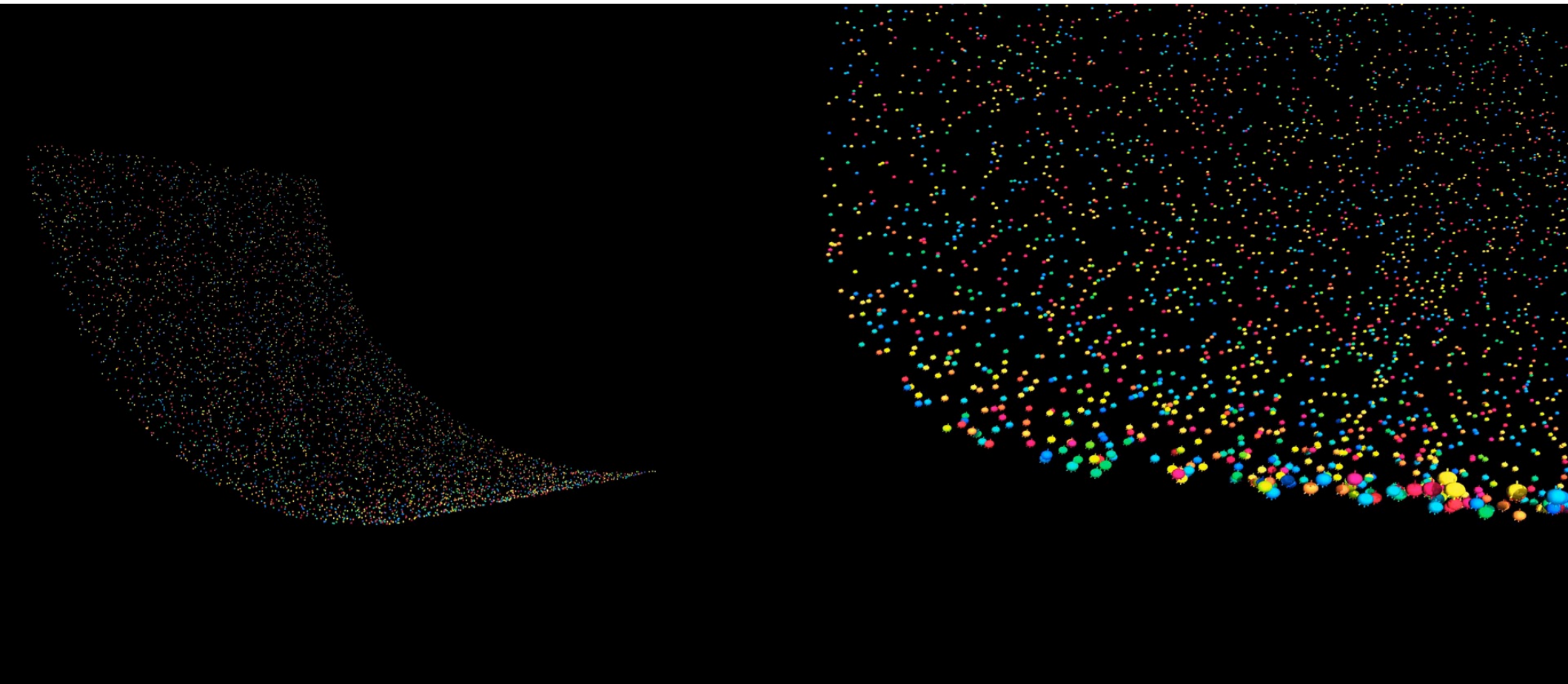
Demo: Neuroscience use case



Demo: Neuroscience use case



Demo: Neuroscience use case



Performance Challenges and Improvements

Maximize parallelization

- Optimized algorithm to search for neighbors
- Parallelize the addition and removal of agents

Efficient thread synchronization during agent updates

Minimize data transfers and memory access latency

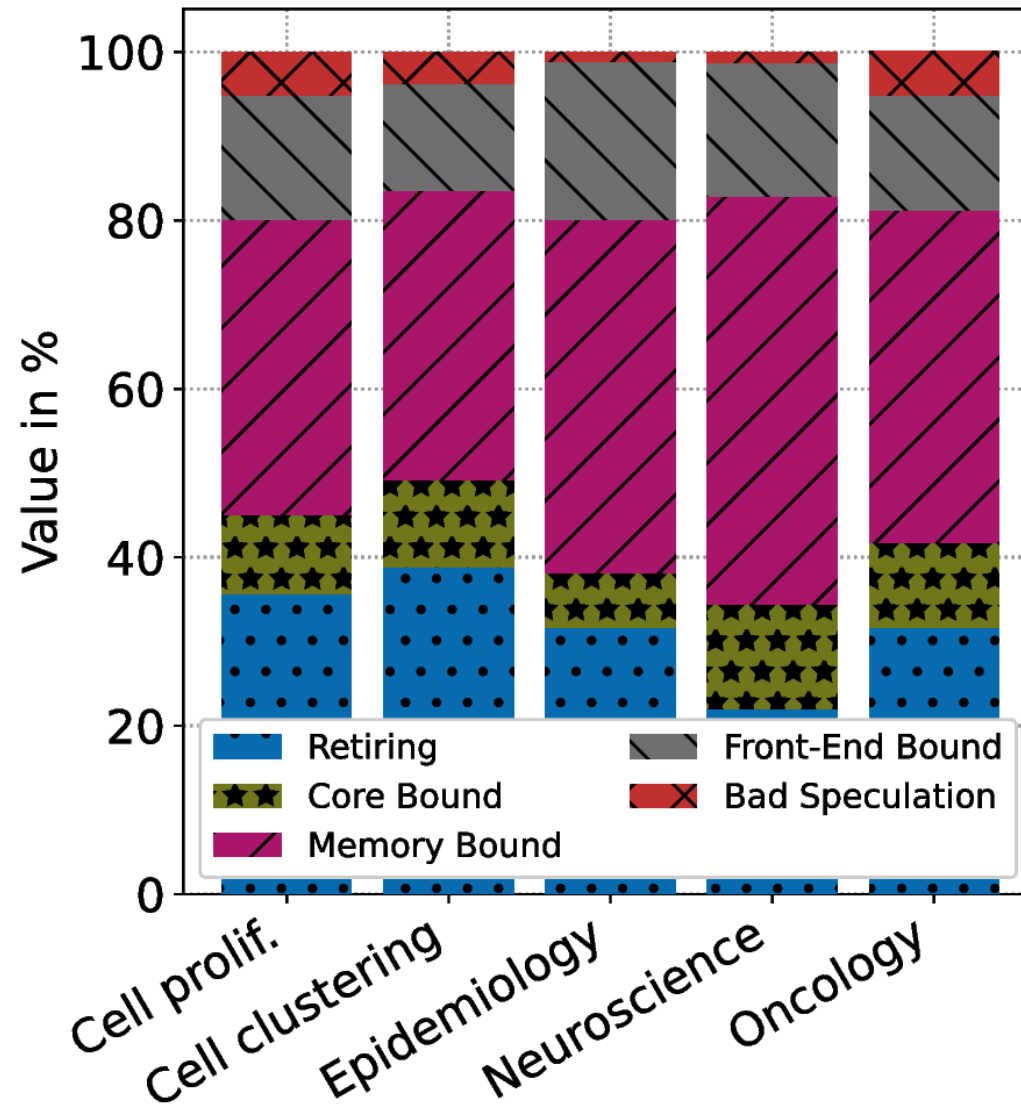
- **NUMA-aware iteration**
- **Agent Sorting and Balancing**
- **Pool-based memory allocator**

Avoid unnecessary work

- Pair-wise force calculation for static regions

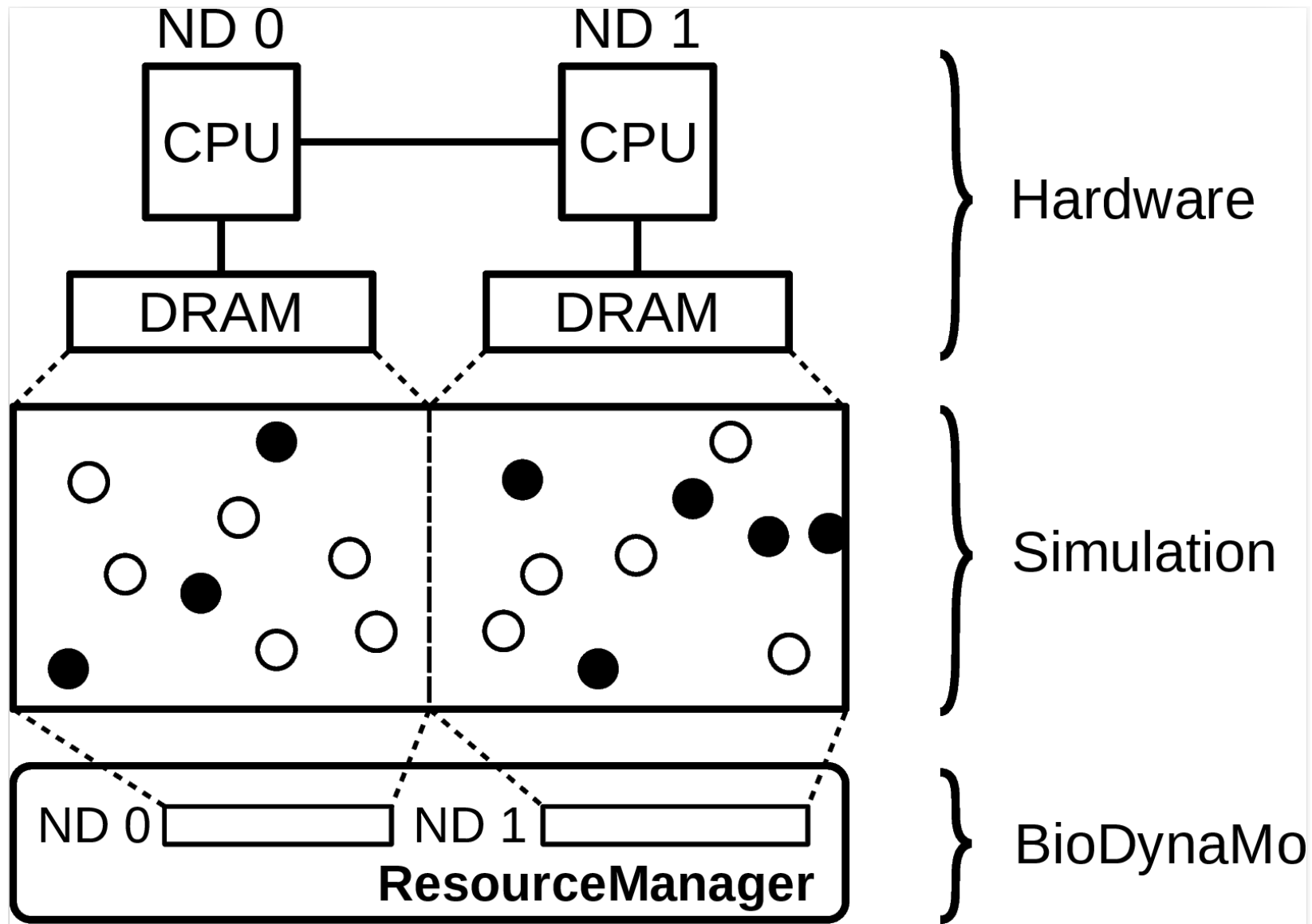
Offload computation to the GPU

Challenge: Agent-based workload is memory-bound



Minimize Memory Access Latency

NUMA-aware iteration



Agent sorting and balancing mechanism

A Agents in 3x3 grid

x \ y	0	1	2
0		a	e
1	b	c,d	f
2	g,h,i		j

B Grid box indices

x \ y	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

C Morton order of 4x4 grid

x \ y	0	1	2	3
0	0	1	4	5
1	2	3	6	7
2	8	9	12	13
3	10	11	14	15

D Determine offsets

		x[0,3] y[0,3]															
		x[0,1] y[0,1]		x[2,3] y[0,1]		x[0,1] y[2,3]		x[2,3] y[2,3]									
box Morton code	0-3	4	5	6	7	8	9	10	11	12	13	14	15				
box counter	0	4	5	5	6	6	7	8	8	8	9	9	9				
offset	0	0	0	1	1	2	2	2	3	4	4	5	6				
found gap	T	F	F	T	F	T	F	F	T	T	F	T	T				

offsets {0,0}{5,1}{6,2}{8,4}

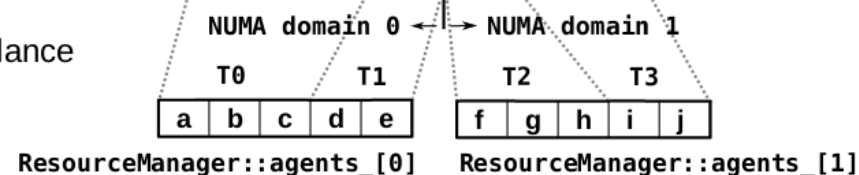
E Determine Morton order

index	0	1	2	3	4	5	6	7	8
+ offsets					0	1	2		4
= Morton order	0	1	2	3	4	6	8	9	12

F Partition

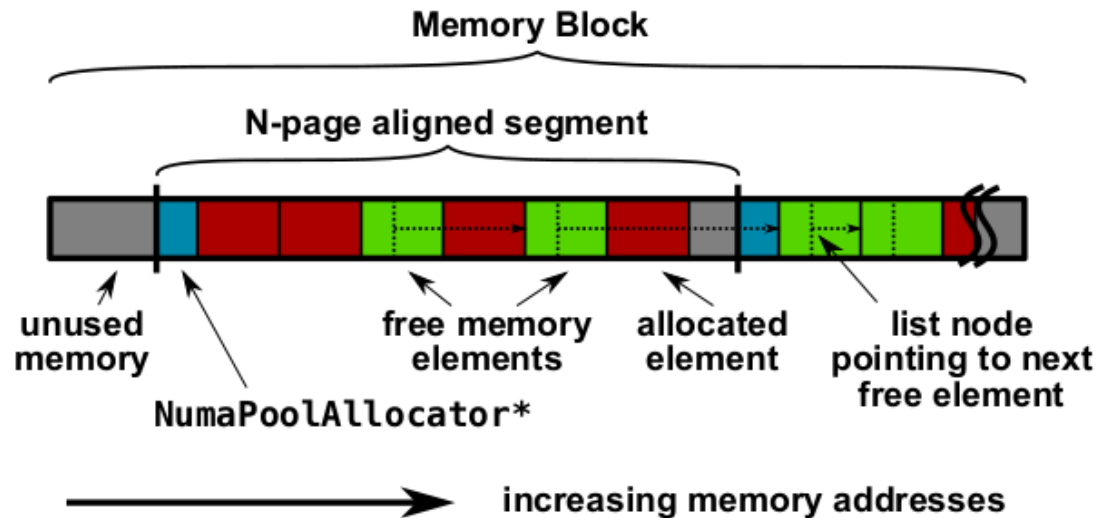
agents per box	0	1	1	2	1	1	3	0	1
prefix sum	0	1	2	4	5	6	9	9	10

G Sort and balance

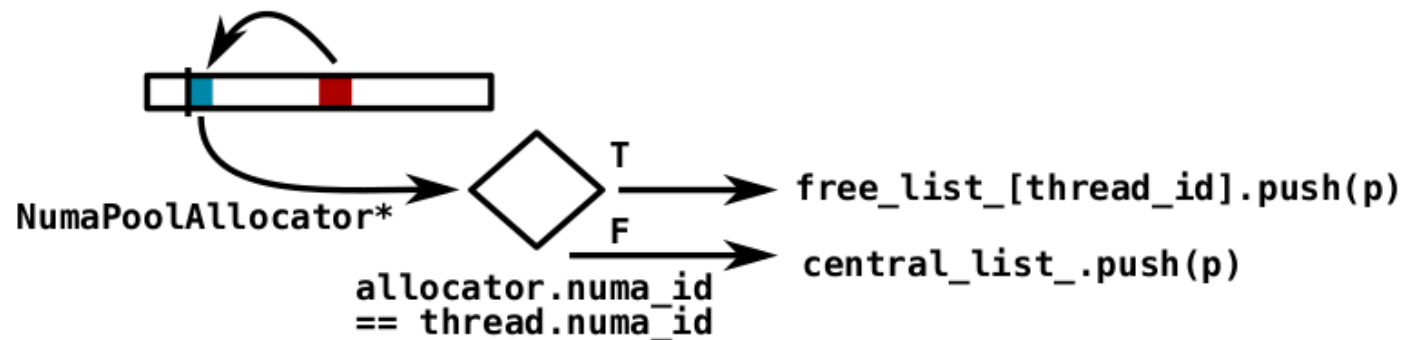


BioDynaMo memory allocator

A Layout

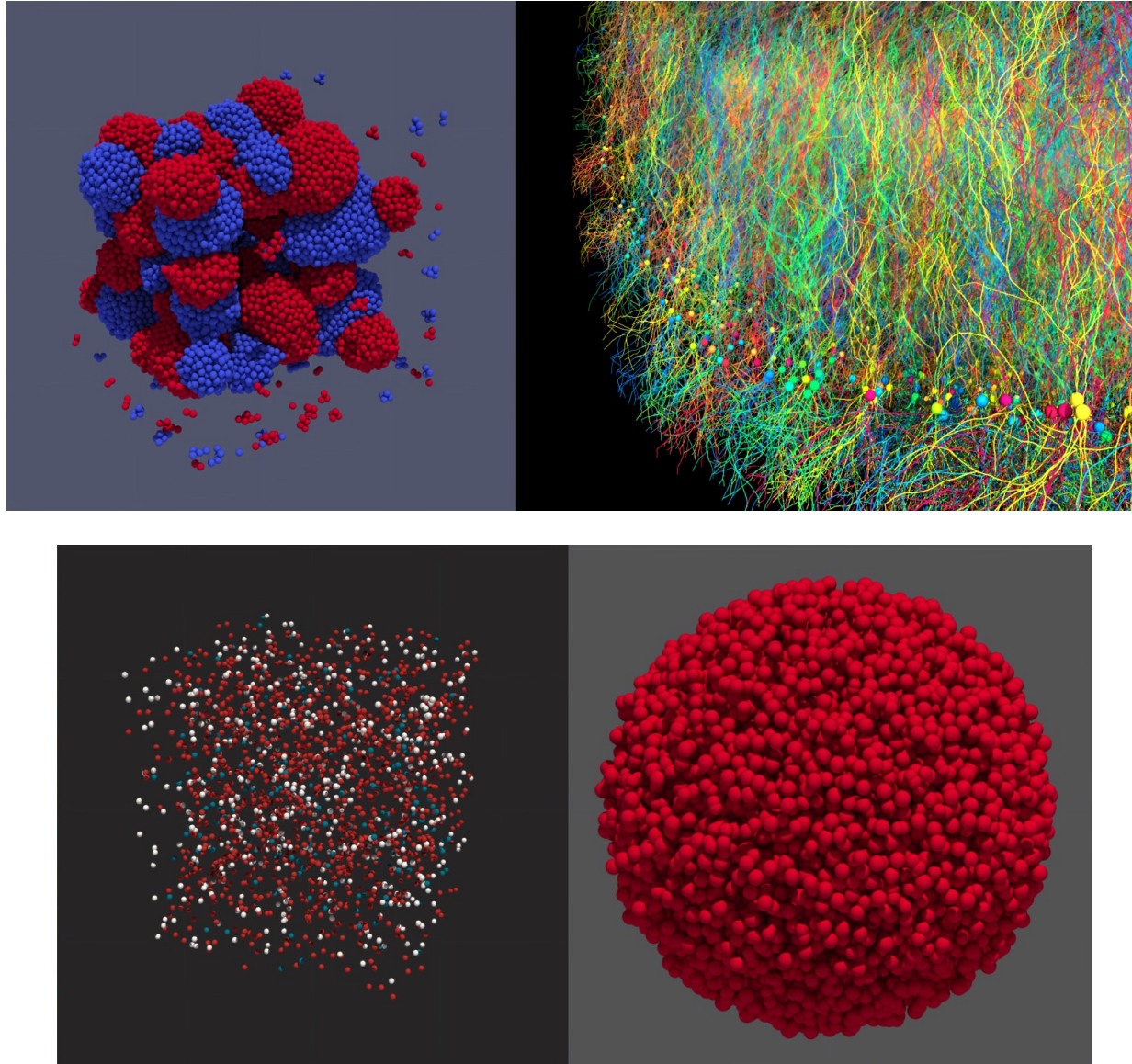


B Deallocation



Performance Evaluation

Benchmark simulations



Benchmark simulations characteristics

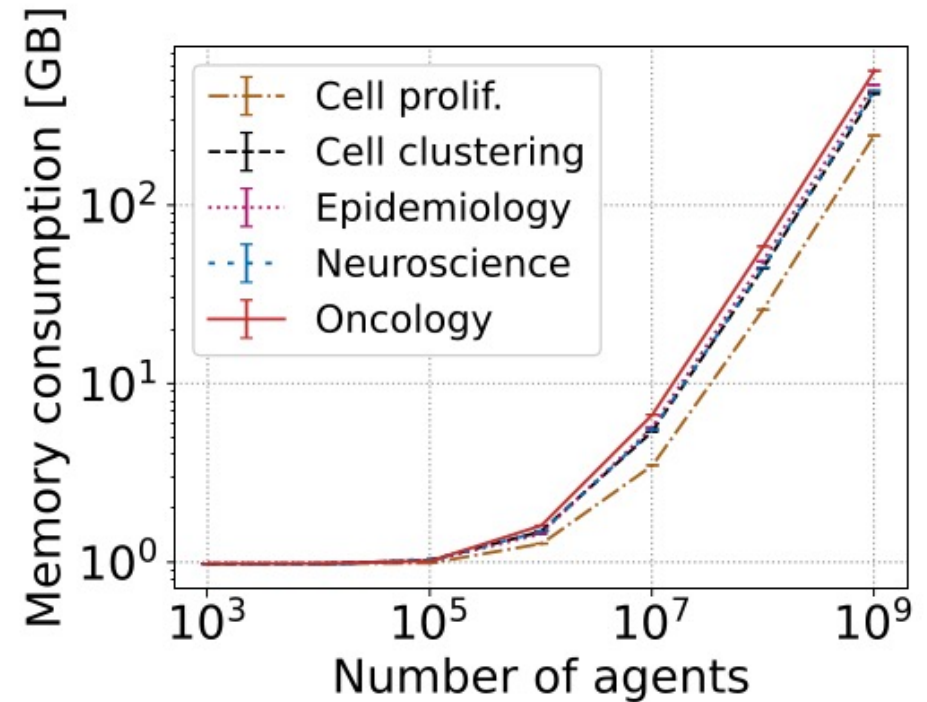
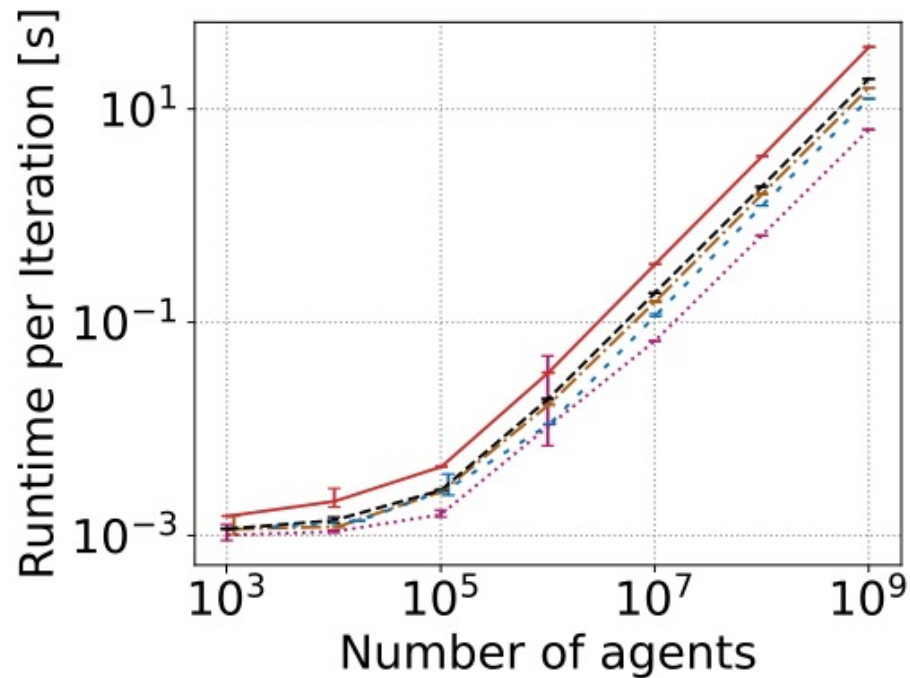
Characteristic	Cell proliferation	Cell clustering	Epidemiology use case	Neuroscience use case	Oncology use case
Create new agents during simulation	X			X	X
Delete agents during simulation					X
Agents modify neighbors				X	
Load imbalance			X	X	
Agents move randomly			X		X
Simulation uses diffusion		X		X	
Simulation has static regions				X	
Number of iterations	500	1000	1000	500	288
Number of agents (in millions)	12.6	2	10	9	10
Number of diffusion volumes	0	54m	0	65k	0

Benchmark hardware

TABLE II: Benchmark hardware

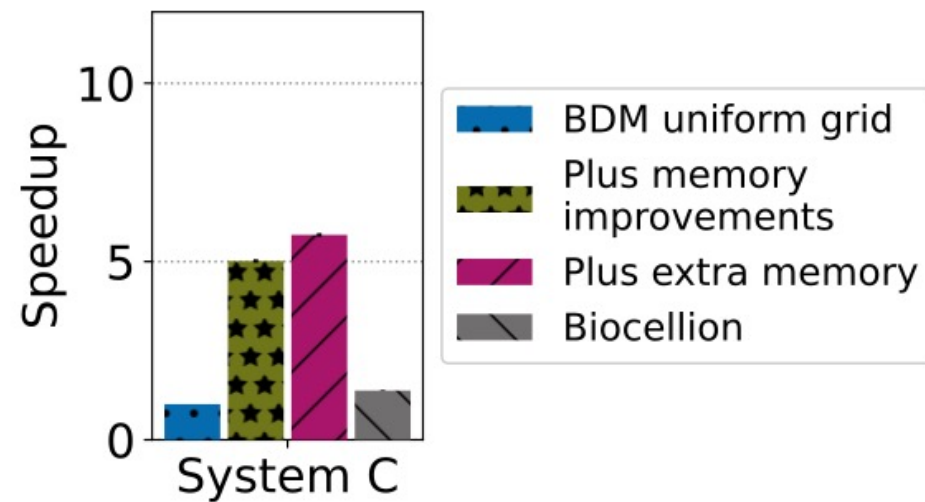
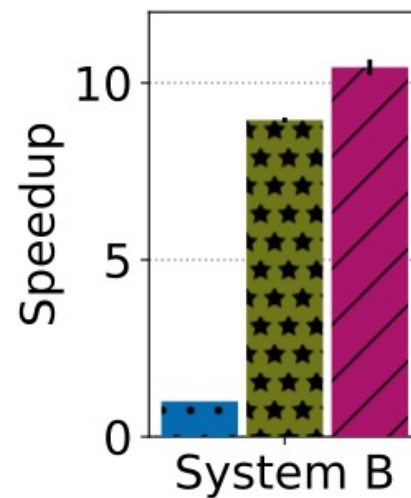
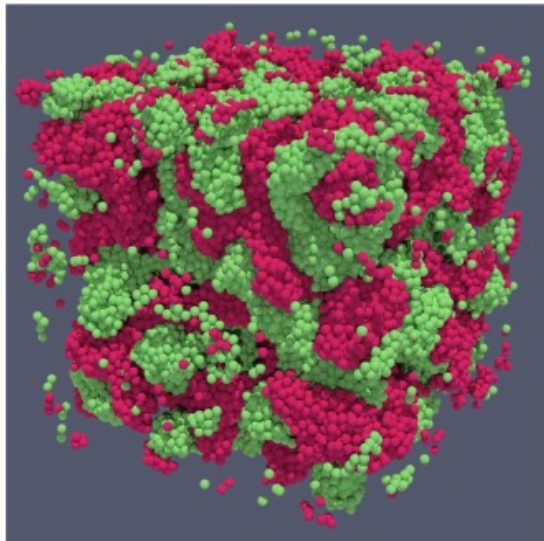
System	Main memory	CPU	OS
A	504 GB	Server with four Intel(R) Xeon(R) E7-8890 v3 CPUs @ 2.50GHz with a total of 72 physical cores, two threads per core and four NUMA nodes.	CentOS 7.9.2009
B	1008 GB		
C	62 GB	Server with two Intel(R) Xeon(R) E5-2683 v3 CPUs @ 2.00GHz with a total of 28 physical cores, two threads per core and two NUMA nodes.	CentOS Stream 8

Runtime and Memory Complexity

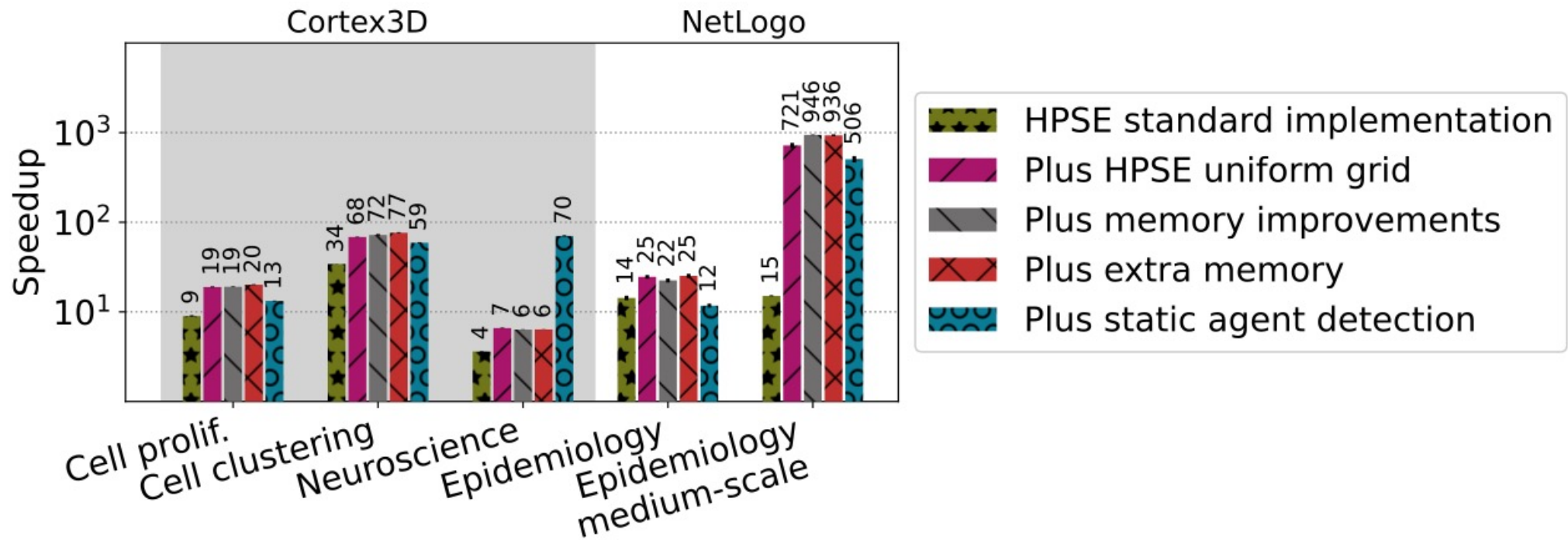


Comparison with Biocellion

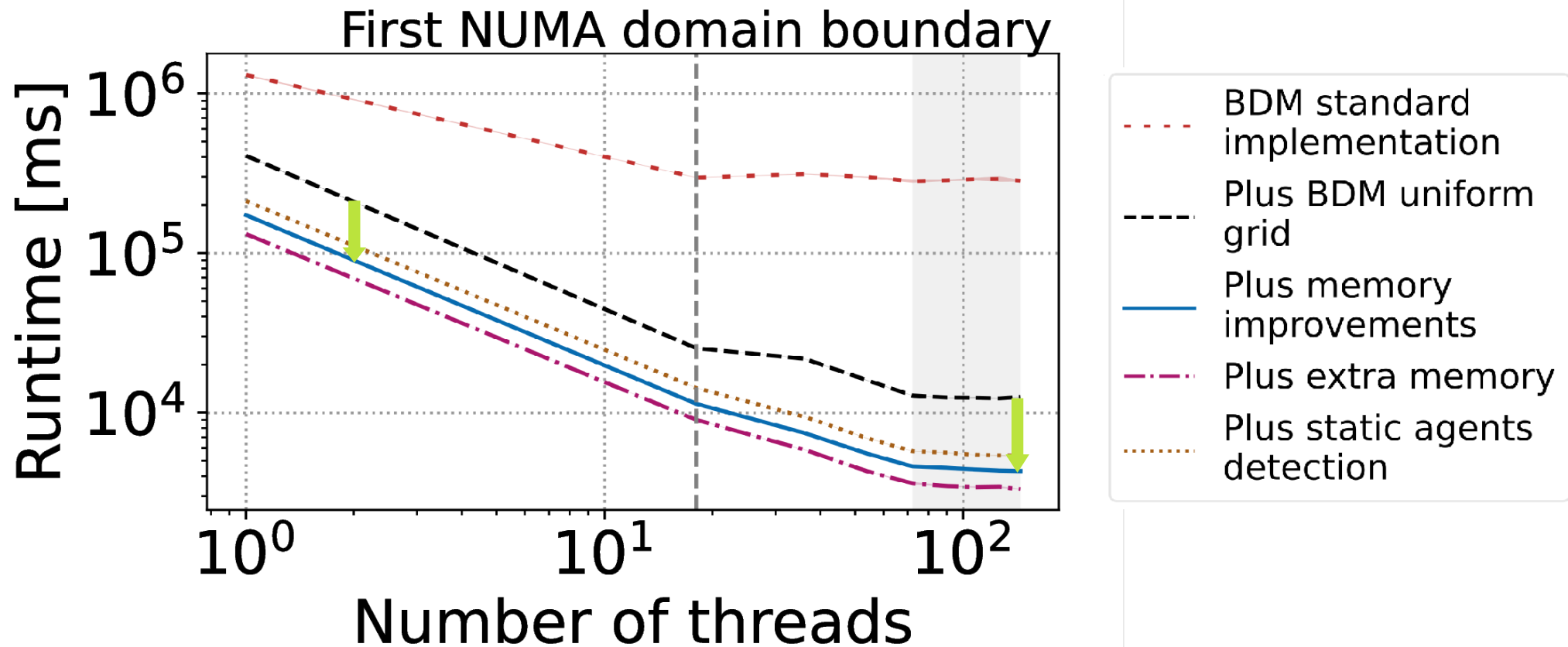
- Single-node 16 CPU cores; 13.4 million cells
→ BioDynaMo is **4.15x faster**
- Biocellion: 21 nodes, 672 CPU cores, 281 million cells
BioDynaMo: one node, 72 CPU cores
→ same runtime, but **9.3x fewer CPU cores** used



Comparison with Cortex3D and NetLogo



Strong scaling



Publications about BioDynaMo

2021

- Lukas Breitwieser et al. **BioDynaMo: a modular platform for high-performance agent-based simulation.** In: Bioinformatics, 2021. DOI: [10.1093/bioinformatics/btab649](https://doi.org/10.1093/bioinformatics/btab649).
- Ahmad Hesam et al. **"GPU Acceleration of 3D Agent-Based Biological Simulations.** In: arXiv, 2021. [arXiv:2105.00039v1](https://arxiv.org/abs/2105.00039v1).

2017

- Roman Bauer et al. **The BioDynaMo project: experience report.** In: Advanced research on biologically inspired cognitive architectures, 2017. DOI: [10.4018/978-1-5225-1947-8.ch006](https://doi.org/10.4018/978-1-5225-1947-8.ch006).

Publications using BioDynaMo

2022

- Marios Demetriades et al. **Interrogating and Quantifying In Vitro Cancer Drug Pharmacodynamics via Agent-Based and Bayesian Monte Carlo Modelling.** In: Pharmaceutics 14(4), 2022. DOI: [10.3390/pharmaceutics14040749](https://doi.org/10.3390/pharmaceutics14040749).
- K. Gazeli et al. **Interrogating an in silico model to determine helium plasma jet and chemotherapy efficacy against B16F10 melanoma cells.** In: Applied Physics Letters, 120(5), 2022. DOI: [10.1063/5.0077694](https://doi.org/10.1063/5.0077694)
- Nicolo Cogno et al. **A 3D Agent-Based Model of Lung Fibrosis.** In: Symmetry, 14(1), 2022. DOI [10.3390/sym14010090](https://doi.org/10.3390/sym14010090)

2021

- Jean de Montigny et al. **Retinal self-organization: a model of RGC and SAC mosaic formation.** In: bioRxiv, 2021. DOI: [10.1101/2021.10.22.465398](https://doi.org/10.1101/2021.10.22.465398).
- Jean de Montigny et al. **An in silico hybrid continuum-/agent-based procedure to modelling cancer development: Interrogating the interplay amongst glioma invasion, vascularity and necrosis.** In: Methods 185, 2021. DOI: [10.1016/j.ymeth.2020.01.006](https://doi.org/10.1016/j.ymeth.2020.01.006).

Summary

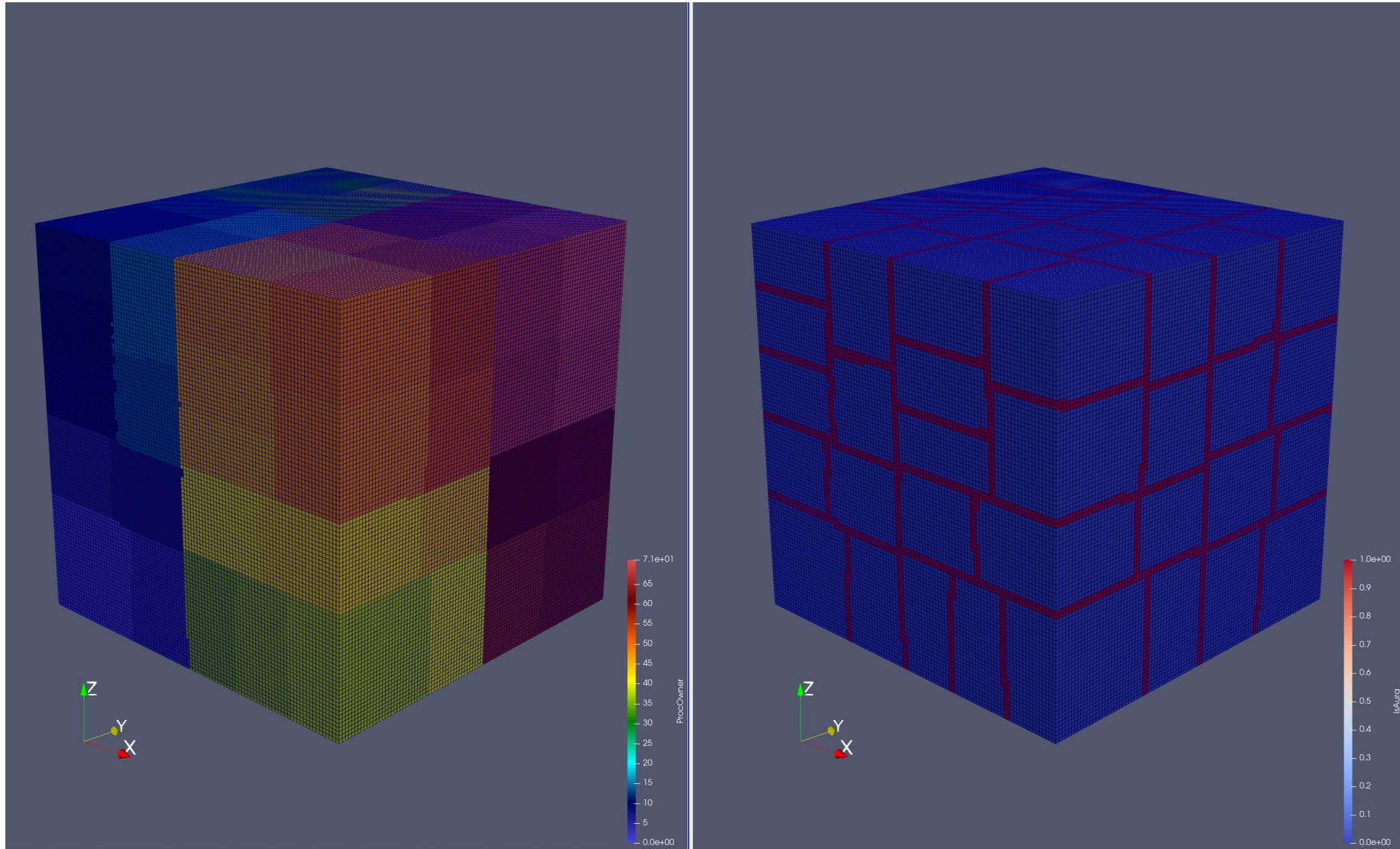
- Agent-based simulation can be used to model many complex systems
- BioDynaMo is up to **three orders of magnitude faster** than state-of-the-art tools.
- These improvements allow BioDynaMo simulating **billions of agents** on a single server.
- BioDynaMo is currently being used in:
 - neuroscience
 - oncology
 - epidemiology
 - cryobiology
 - socioeconomics
 - finance
 - ...
- BioDynaMo is **open-source** and we would be very happy to welcome new users and contributors.

Thank you for your attention!

Lukas.Breitwieser@cern.ch

Backup Slides

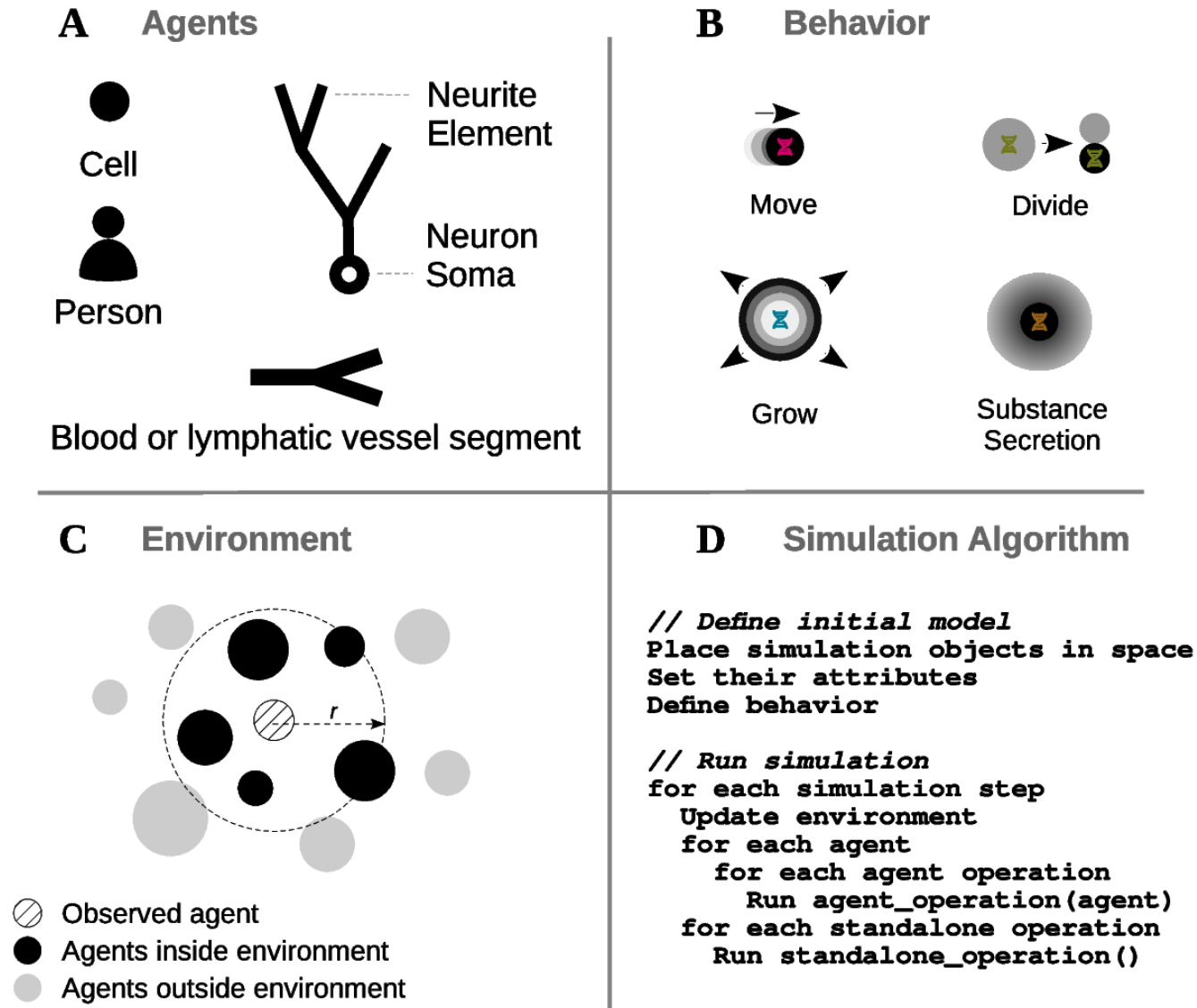
Distributed simulation engine



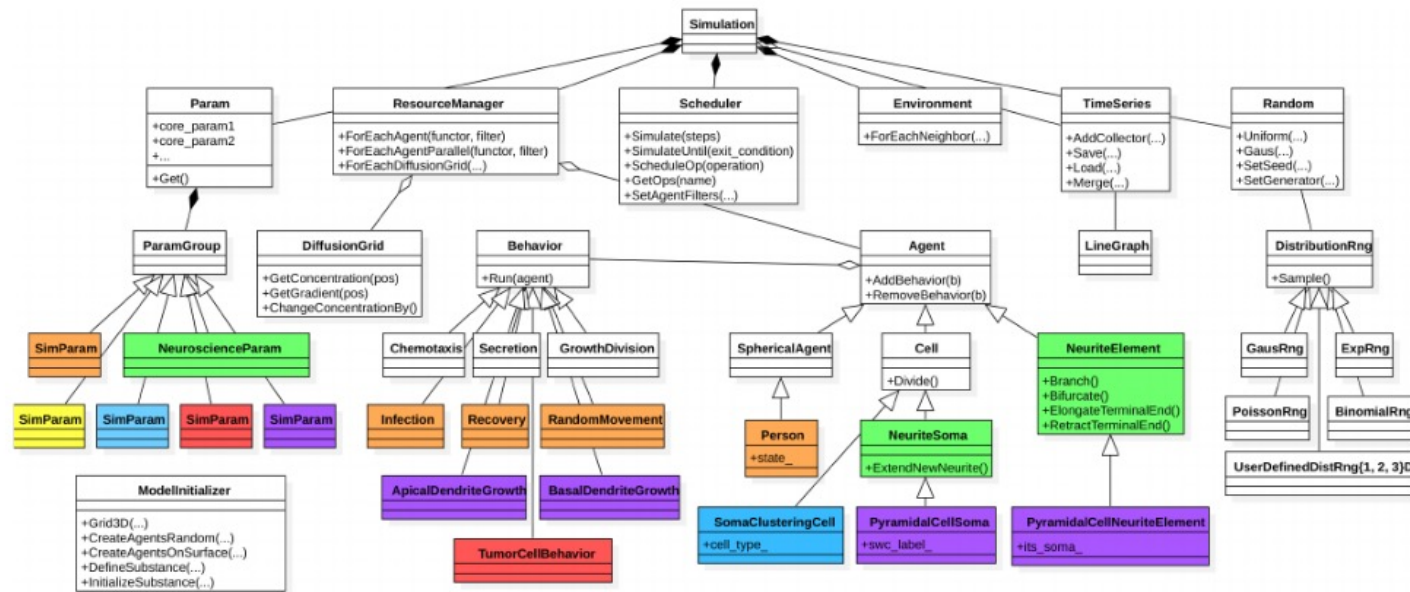
Agent-based simulation algorithm

```
1 ModelInitialization()
2 for  $i \in iterations$  do
3     for  $op \in pre\_standalone\_operations$  do
4         |  $op()$ ;
5     end
6     wait()
7     parallel for  $a \in agents$  do
8         | for  $op \in agent\_operations$  do
9             | |  $op(a)$ ;
10            end
11        end
12    for  $op \in standalone\_operations$  do
13        |  $op()$ ;
14    end
15    wait()
16    for  $op \in post\_standalone\_operations$  do
17        |  $op()$ ;
18    end
19 end
```

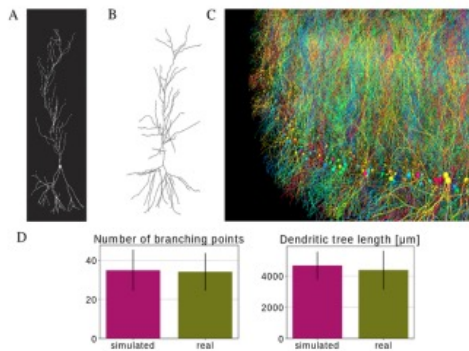
Important building blocks



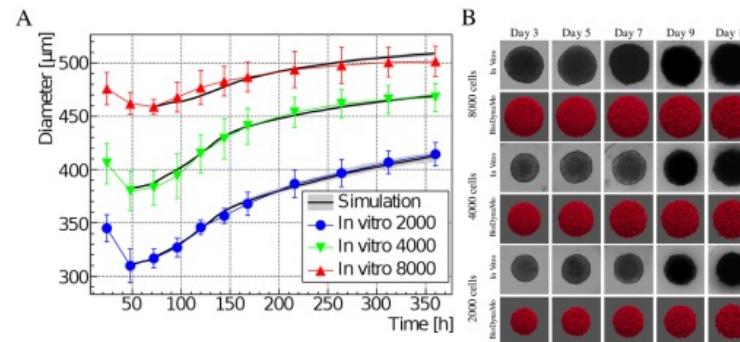
Modular software design



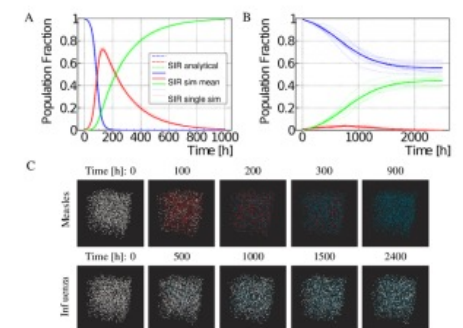
Neuroscience use case



Oncology use case

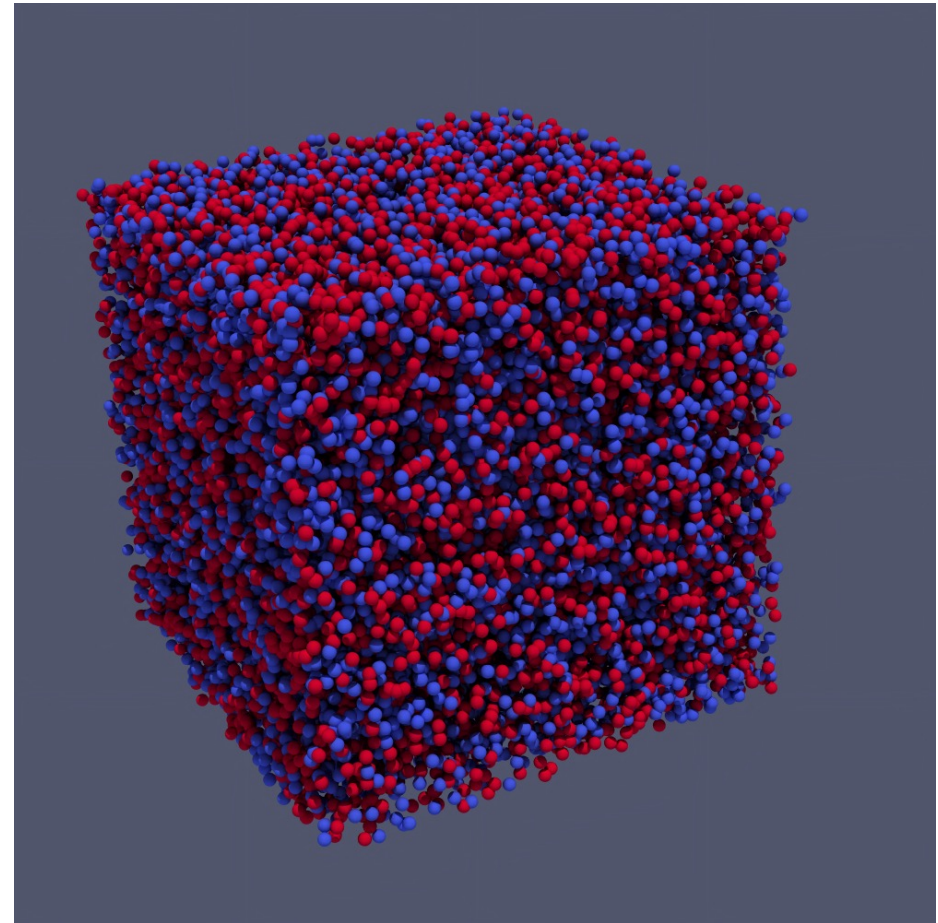


Epidemiology use case

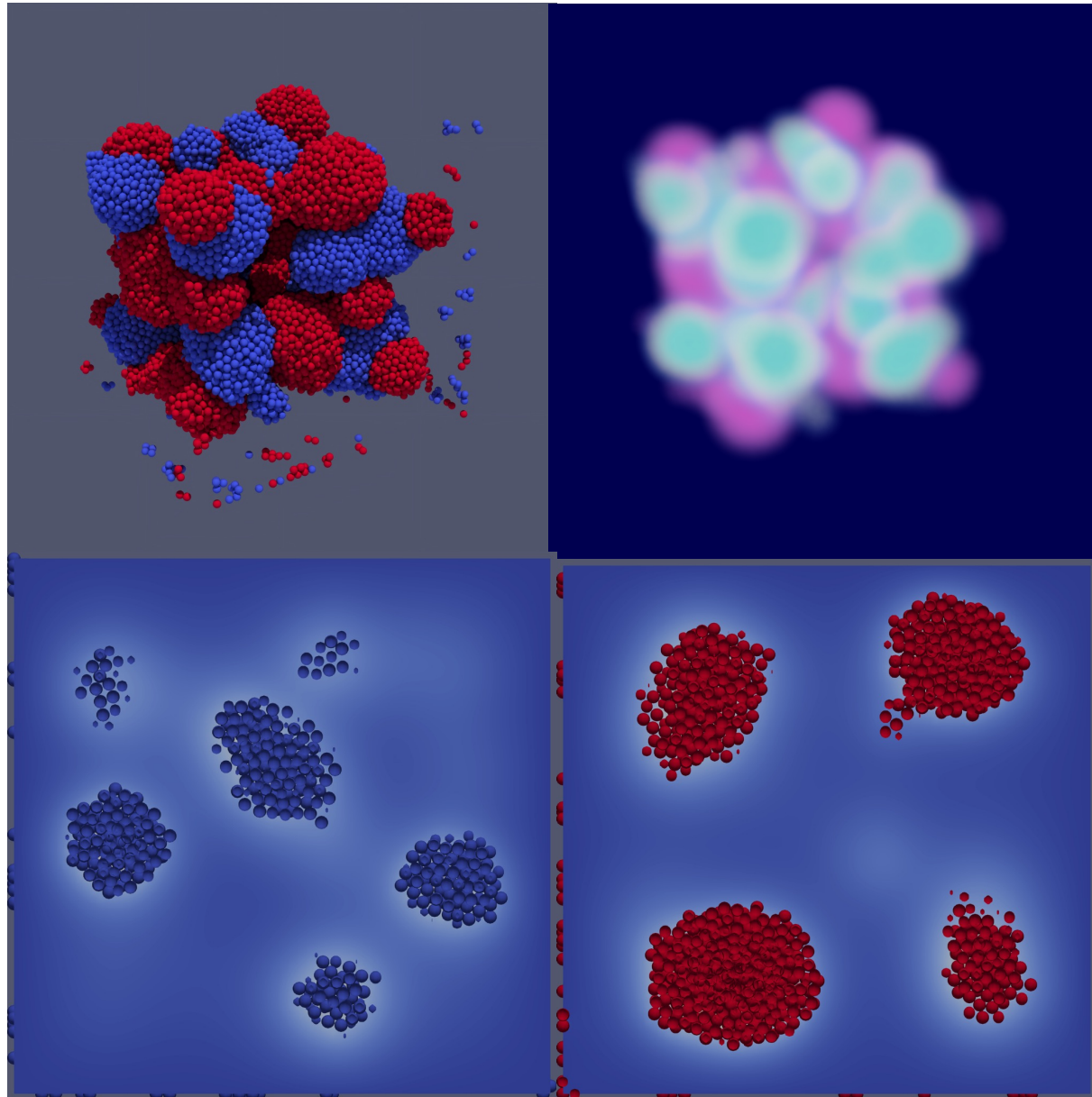


Cell clustering model

- Agent: Cell
 - Spherical shape
 - cell type
- Behaviors
 - Secrete a substance into the extracellular matrix
 - Follow the concentration gradient (chemotaxis)
- Initial condition
 - Randomly distributed in 3D space



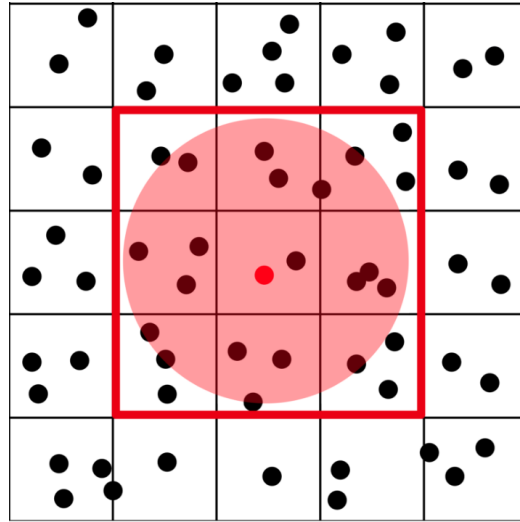
Cell clustering result



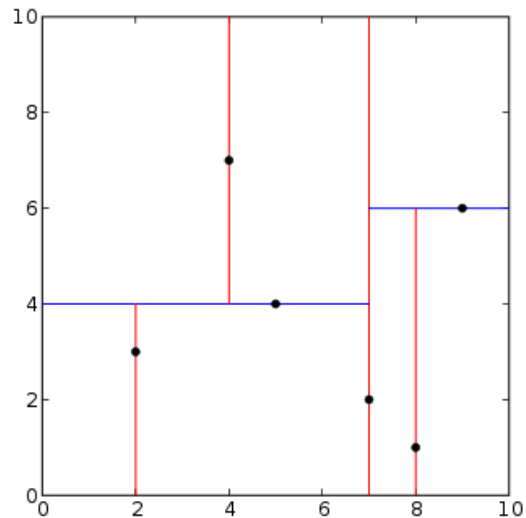
Maximize parallelization

Optimized uniform grid to search for neighbors

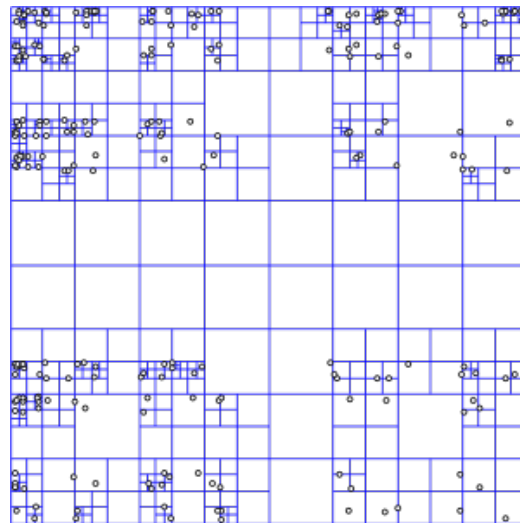
Uniform Grid



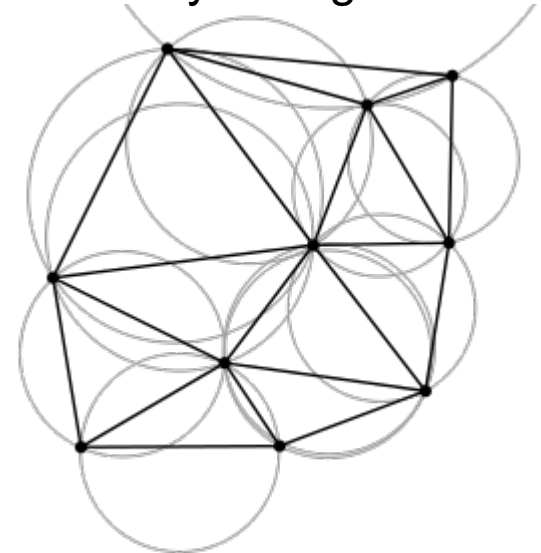
Kd-Tree



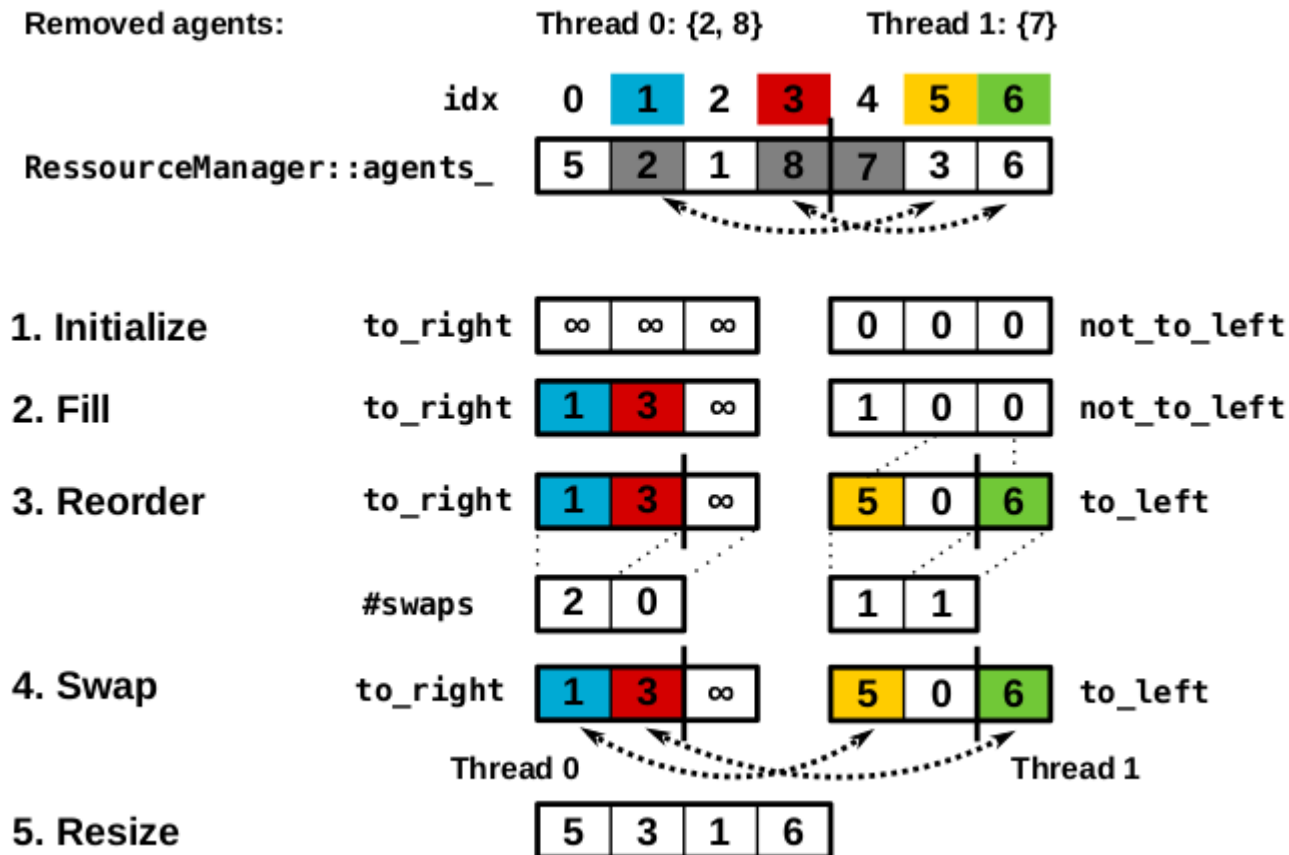
Quadtree



Delaunay Triangulation



Parallel agent removal mechanism



Optimize Thread-Synchronization

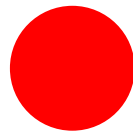
Thread-synchronization (TS) during agent-updates

Algorithm 1: Agent-based simulation algorithm

```
1 ModelInitialization()
2 for  $i \in \text{iterations}$  do
3   for  $op \in \text{pre\_standalone\_operations}$  do
4     |  $op()$ ;
5   end
6   wait()
7   parallel for  $a \in \text{agents}$  do
8     for  $op \in \text{agent\_operations}$  do
9       |  $op(a)$ ;
10    end
11  end
12  for  $op \in \text{standalone\_operations}$  do
13    |  $op()$ ;
14  end
15  wait()
16  for  $op \in \text{post\_standalone\_operations}$  do
17    |  $op()$ ;
18  end
19 end
```



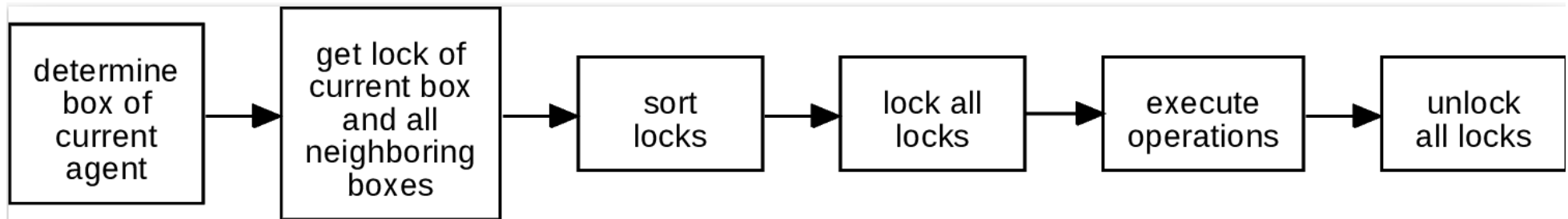
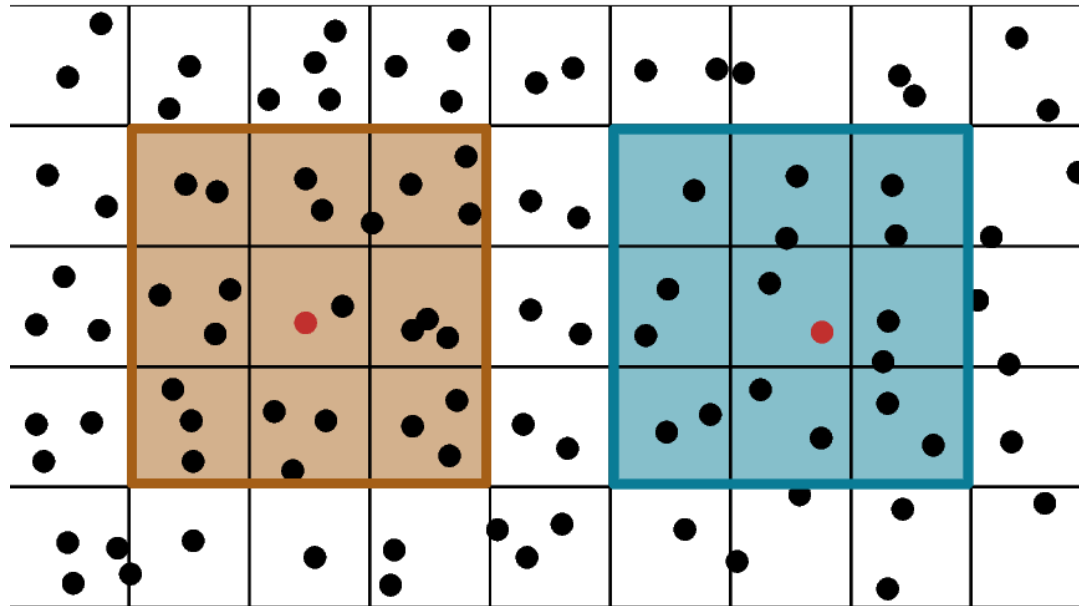
T0



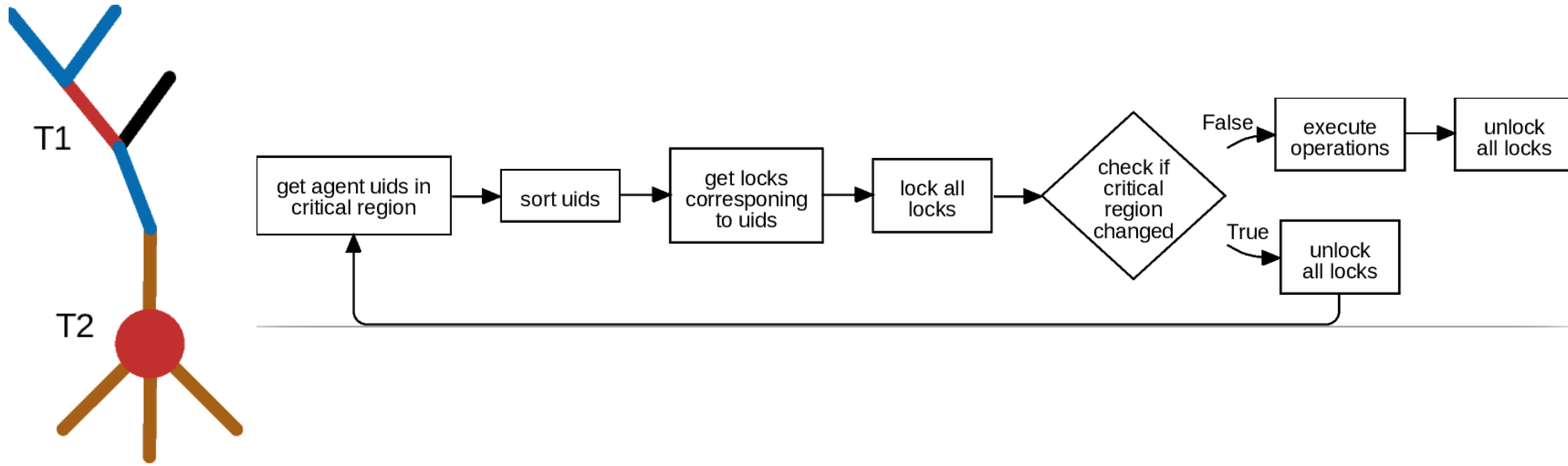
T1

- Only necessary if agents modify their local environment.
 - Two agents (updated by two different threads) could attempt to modify the same neighbor.
- BioDynaMo provides two TS mechanisms
 - Automatic TS
 - User-defined TS

Automatic thread-synchronization



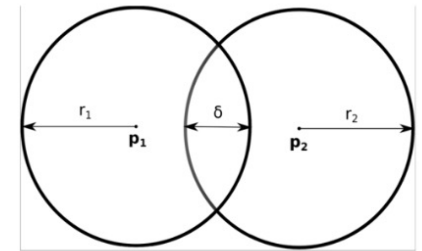
User-defined thread-synchronization



```
void NeuronSoma::CriticalRegion(std::vector<AgentPointer<>>* aptrs) {
    aptrs->reserve(daughters_.size() + 1);
    aptrs->push_back(Agent::GetAgentPtr<>());
    for (auto& daughter : daughters_) {
        aptrs->push_back(daughter);
    }
}
```

BioDynaMo's GPU capabilities

- Operations can have implementations for different compute targets (CPU, GPU, and FPGA).
- If an operation has multiple implementations, the scheduler decides which one to use.
- Currently, BioDynaMo provides a GPU operation to calculate mechanical forces between spheres.



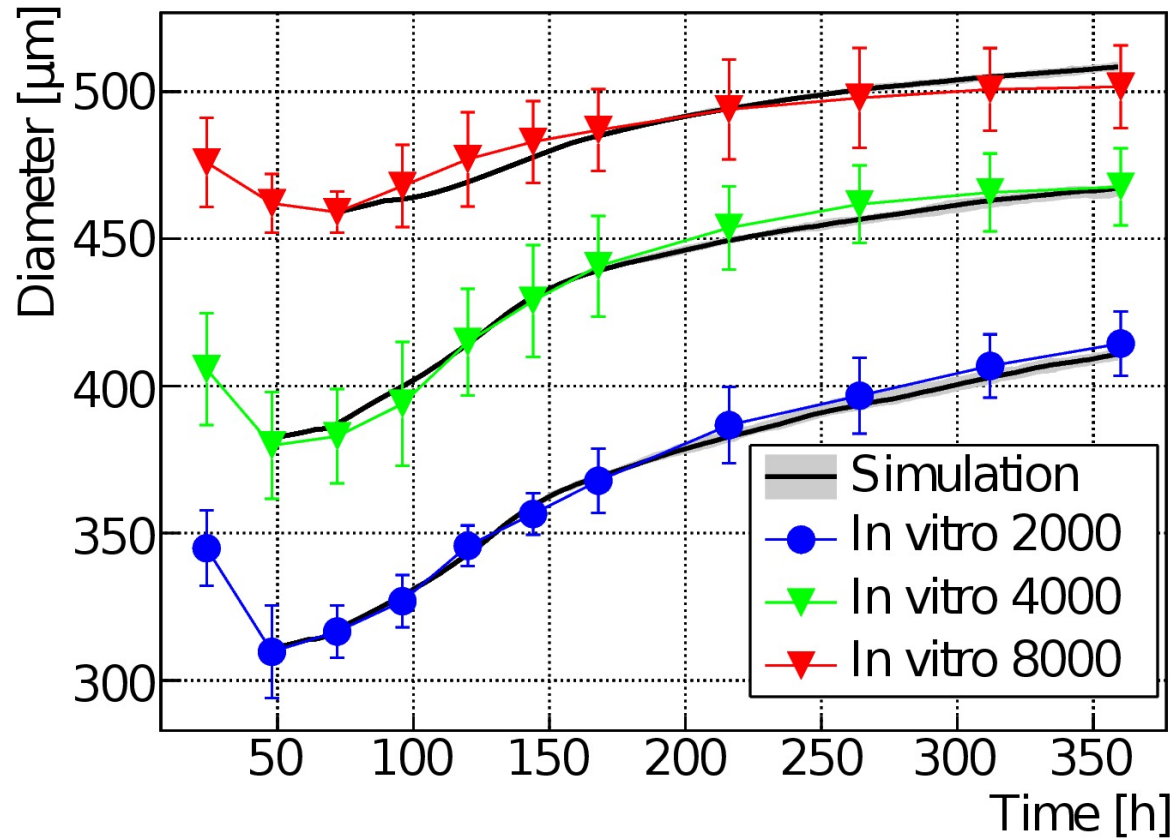
$$\begin{aligned}\delta &= r_1 + r_2 - \|p_1 - p_2\| \\ r &= \frac{r_1 \cdot r_2}{r_1 + r_2} \\ \mathbf{F} &= (\kappa \cdot \delta - \gamma \cdot \sqrt{r \cdot \delta}) \cdot \frac{p_1 - p_2}{\|p_1 - p_2\|}\end{aligned}$$

Collision force computation

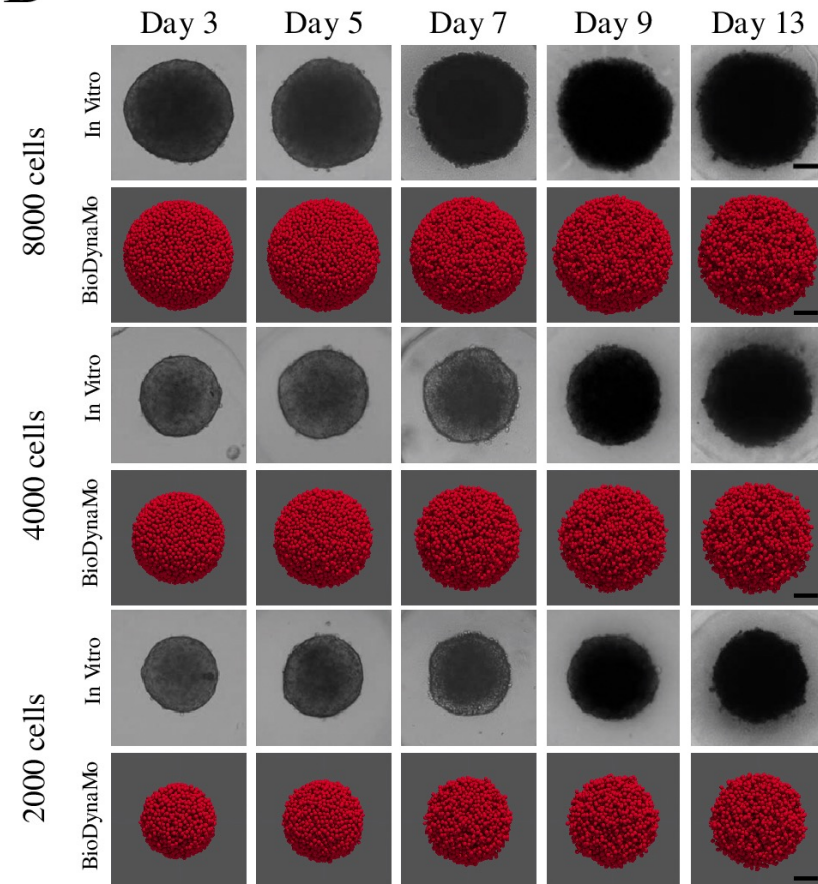
(Ongoing) Use Cases

Oncology use case

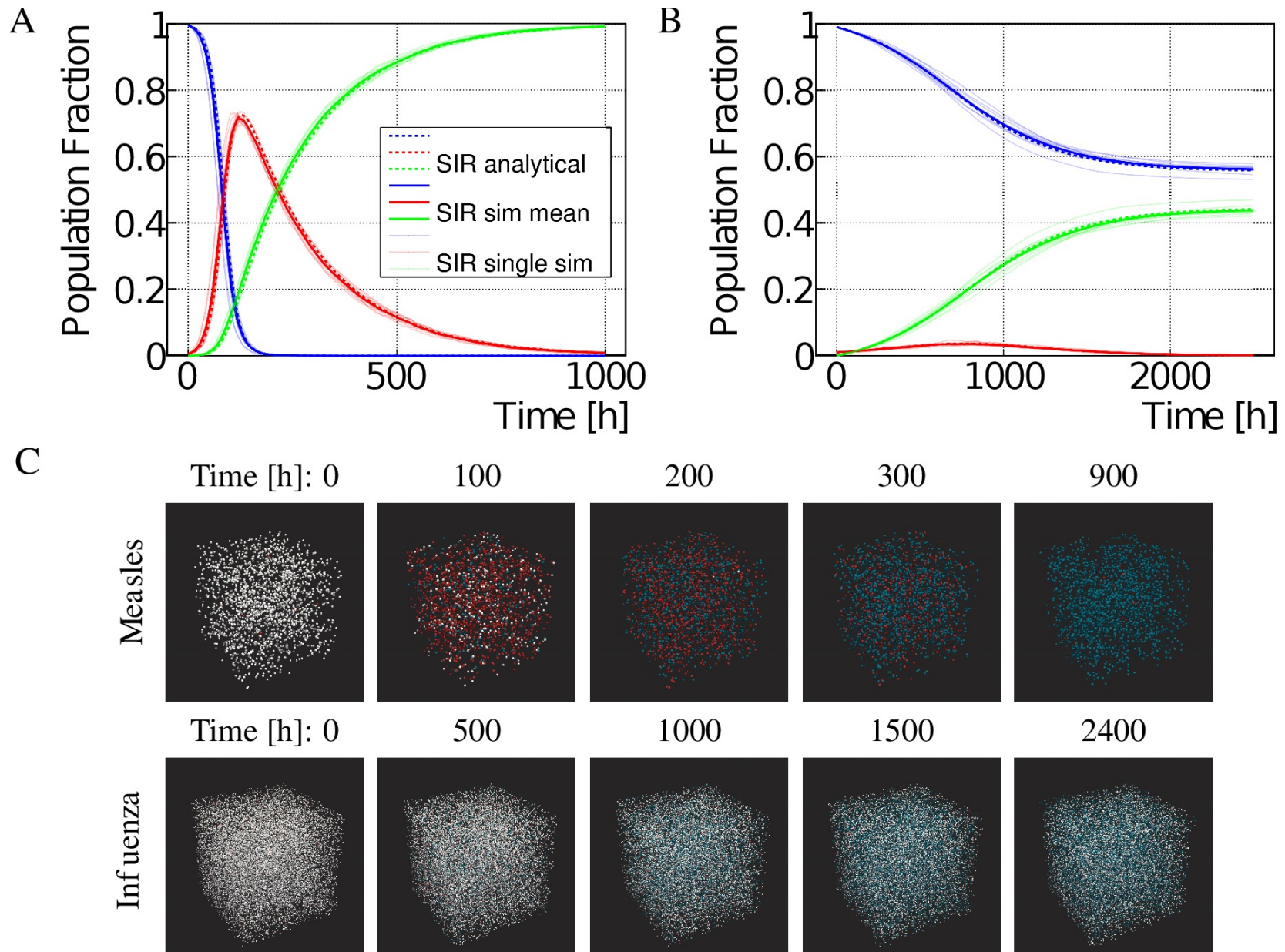
A



B



Epidemiology use case



Cancer modeling article

Methods 185 (2021) 94–104



Contents lists available at ScienceDirect

Methods

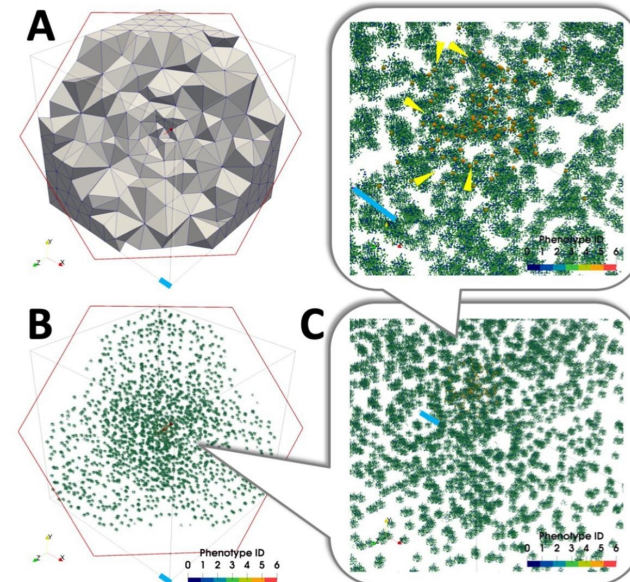
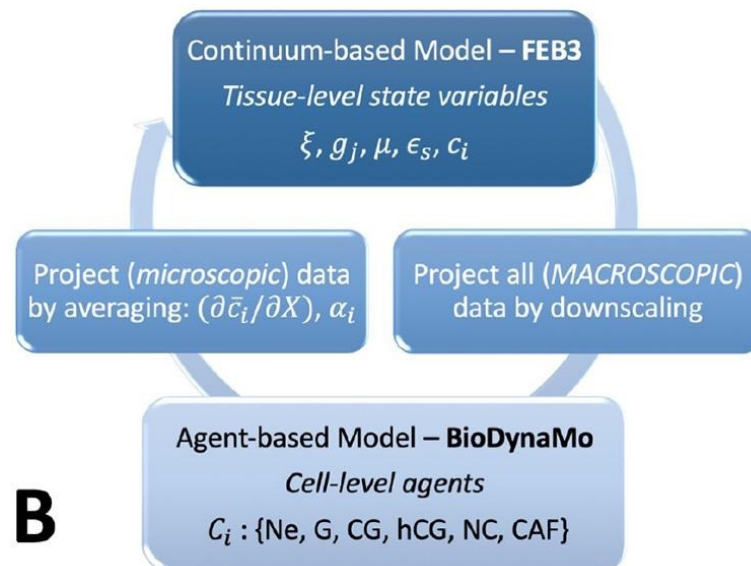
journal homepage: www.elsevier.com/locate/ymeth



An *in silico* hybrid continuum-/agent-based procedure to modelling cancer development: Interrogating the interplay amongst glioma invasion, vascularity and necrosis

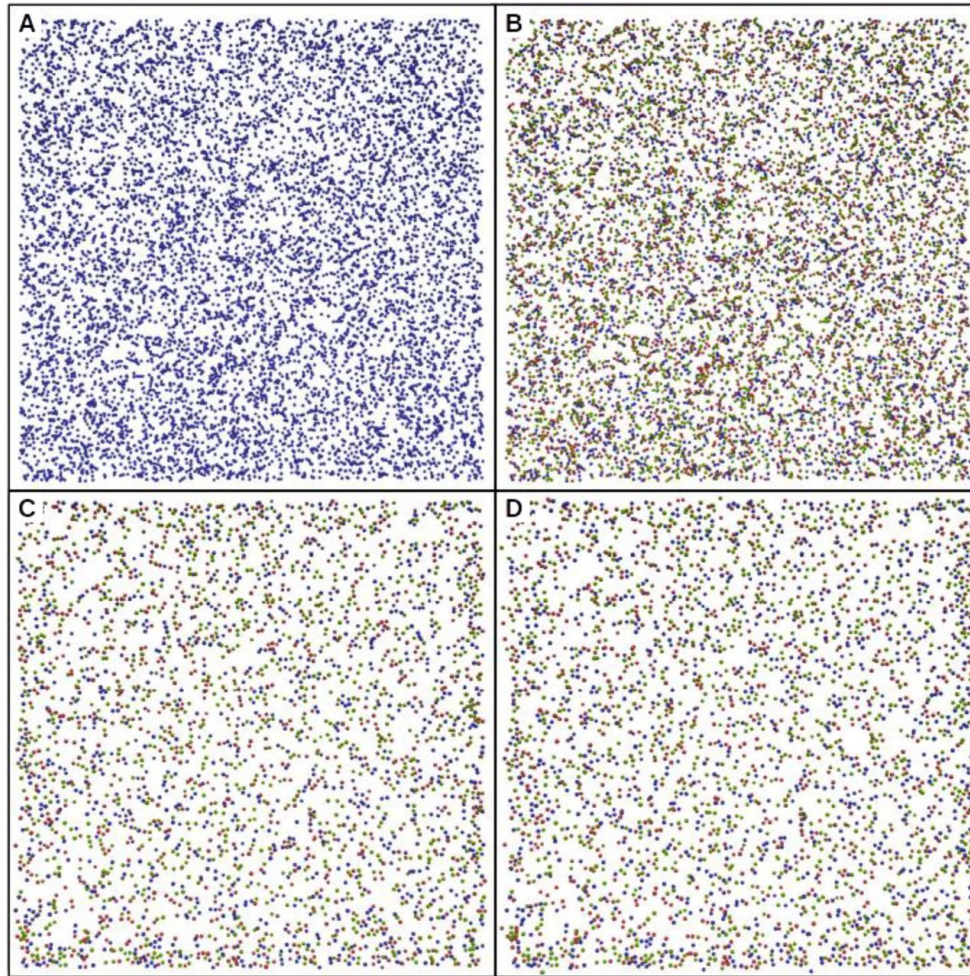


Jean de Montigny^a, Alexandros Iosif^b, Lukas Breitwieser^{c,d}, Marco Manca^e, Roman Bauer^{f,a}, Vasileios Vavourakis^{b,g,*}



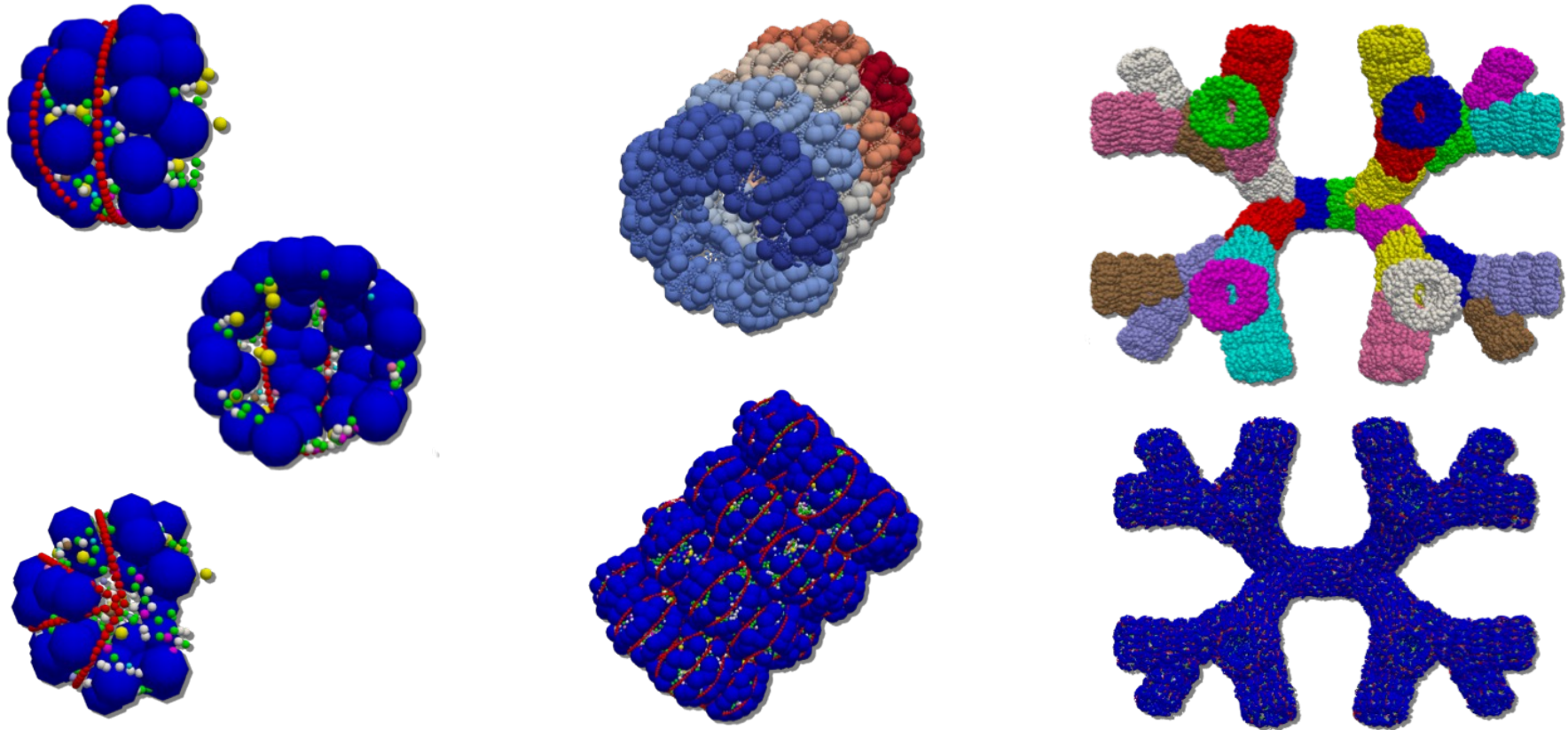
Retinal self-organization

Understand the mechanisms of cells self-organization during early development which is pivotal for their function.



Radiation-induced lung injury simulation


Simulate onset of radiation pneumonitis and/or lung fibrosis in normal tissue after exposition to thoracic irradiation.



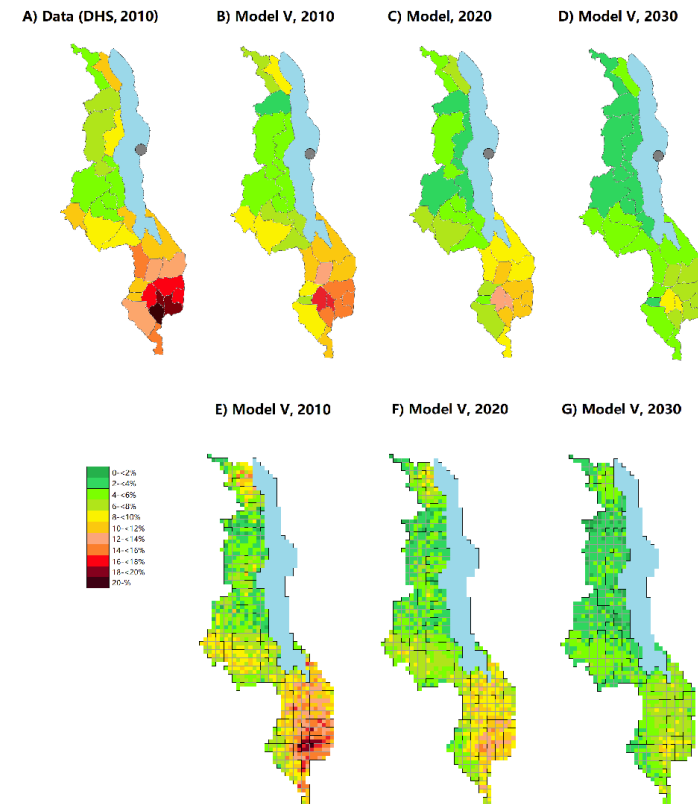
Spatial Spread of HIV in Malawi

- Collaboration with UniGE
- Original simulation written in R (Runtime: ~5.5h)
- Goal: speed up execution time
- Preliminary runtime with BioDynaMo: less than **2 minutes**
- Further work needed to make models equivalent

The spatial spread of HIV in Malawi: An individual-based mathematical model

 Janne Estill, Wingston Ng'ambi, Liudmila Rozanova, Olivia Keiser

doi: <https://doi.org/10.1101/2020.12.23.20248757>



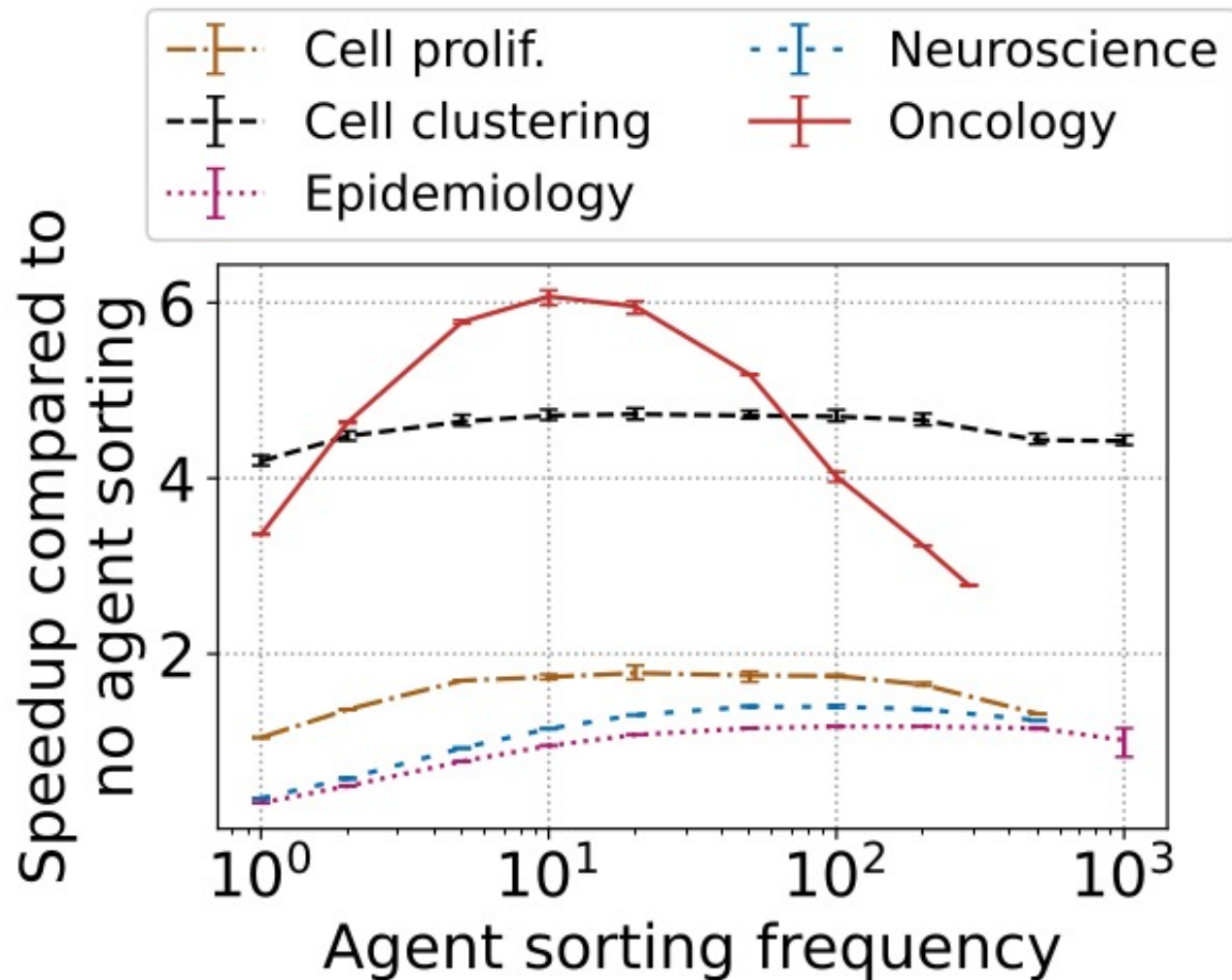
Evaluation

Use cases performance data

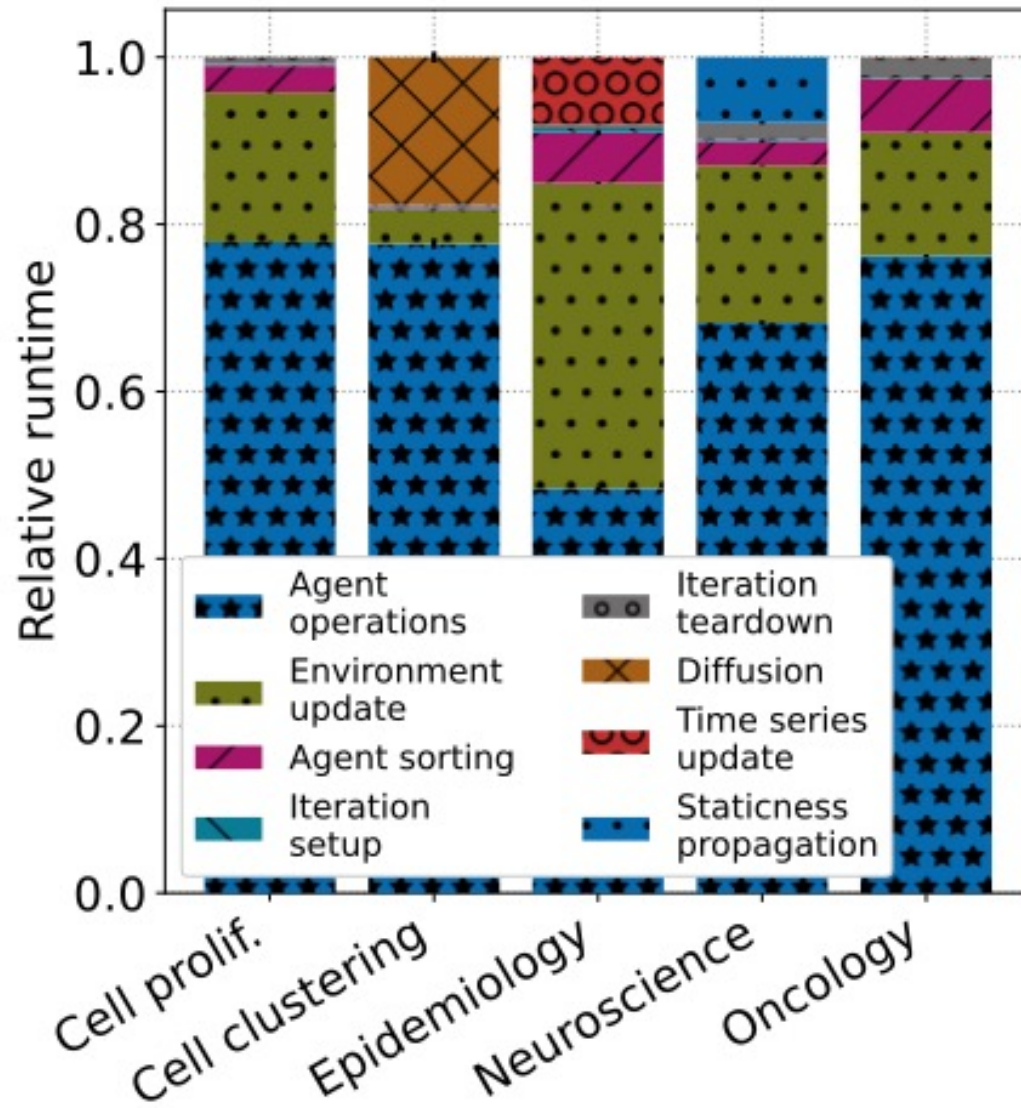
Table 6. **Performance data.** The values in column "Agents" and "Diffusion volumes" are taken from the end of the simulation. Runtime measures the wall-clock time to simulate the number of iterations. It excludes the time for simulation setup and visualization.

Simulation	Agents	Diffusion volumes	Iterations	System (Table 5)	Physical CPUs	Runtime	Memory
Neuroscience use case							
Single (Figure 4A in the main manuscript)	1 494	250	500	A	1	0.16 s	382 MB
				D	1	0.12 s	479 MB
Large-scale (Figure 4C in the main manuscript)	9 036 986	65 536	500	A	72	35 s	6.47 GB
				D	2	11 min 28 s	5.37 GB
Very-large-scale	1 018 644 154	5 606 442	500	B	72	1 h 24 min	438 GB
Oncology use case (Figure 5 in the main manuscript)							
2000 initial cells	4 177	0	312	A	1	1.05 s	382 MB
				D	1	0.832 s	480 MB
4000 initial cells	5 341	0	312	A	1	1.76 s	382 MB
				D	1	1.34 s	480 MB
8000 initial cells	7 861	0	288	A	1	3.27 s	384 MB
				D	1	2.60 s	482 MB
Large-scale	1 000 3925	0	288	A	72	1 min 42 s	7.42 GB
				D	2	43 min 56 s	5.84 GB
Very-large-scale	986 054 868	0	288	B	72	6 h 21 min	604 GB
Epidemiology use case (Figure 6C in the main manuscript)							
Measles	2 010	0	1000	A	1	0.53 s	381 MB
				D	1	0.42 s	479 MB
Seasonal Influenza	20 200	0	2500	A	1	16.41 s	383 MB
				D	1	16.40 s	479 GB
Medium-scale (measles)	100 500	0	1000	A	72	1.36 s	1 GB
Large-scale (measles)	10 050 000	0	1000	A	72	59.19 s	5.87 GB
				D	2	19 min 18 s	5.41 GB
Very-large-scale (measles)	1 005 000 000	0	1000	B	72	2 h 0 min	495 GB
Soma clustering (Figure 2)	32 000	1 240 000	6 000	A	72	12.91 s	1.02 GB
				D	2	2 min 7 s	522 MB

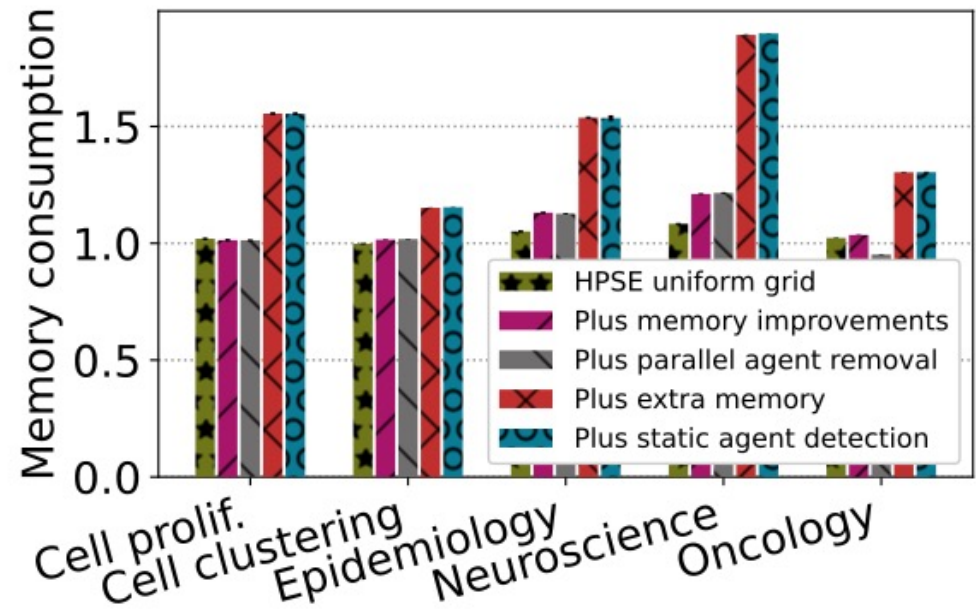
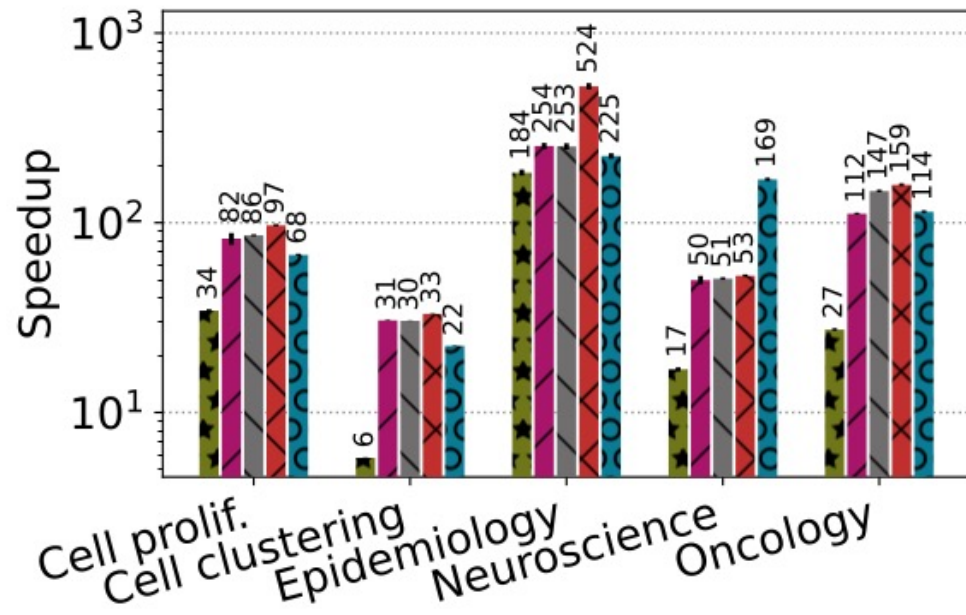
Agent sorting and balancing



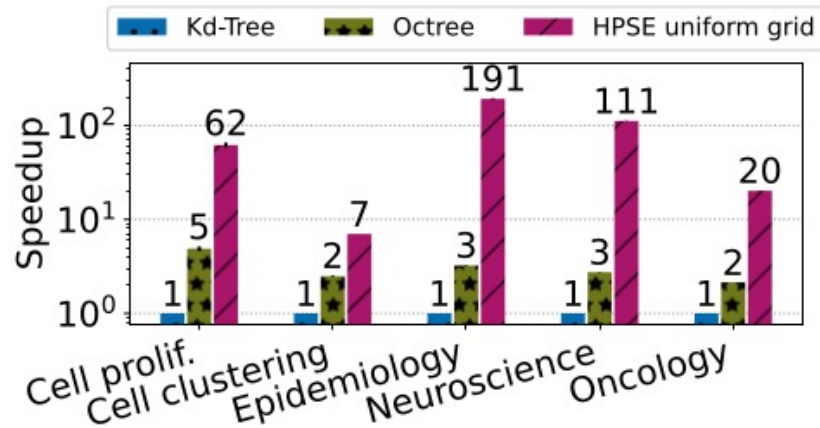
Operation breakdown



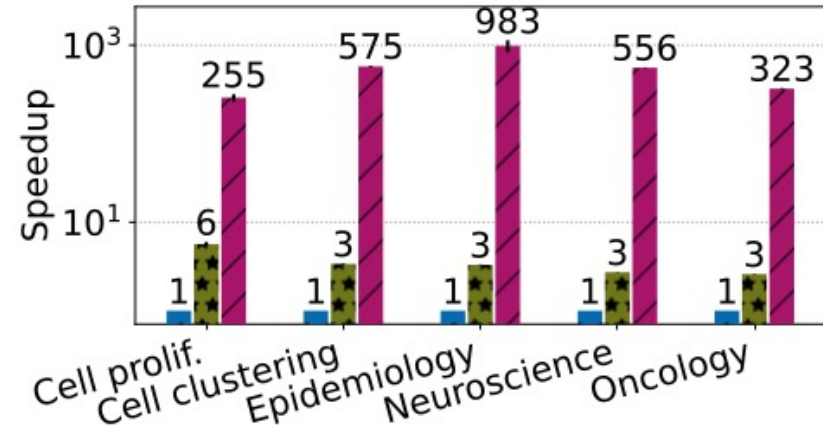
Optimization overview



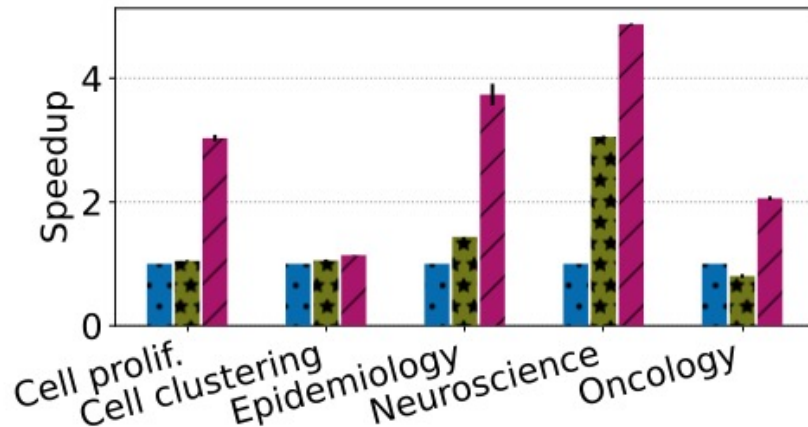
Environment algorithm comparison



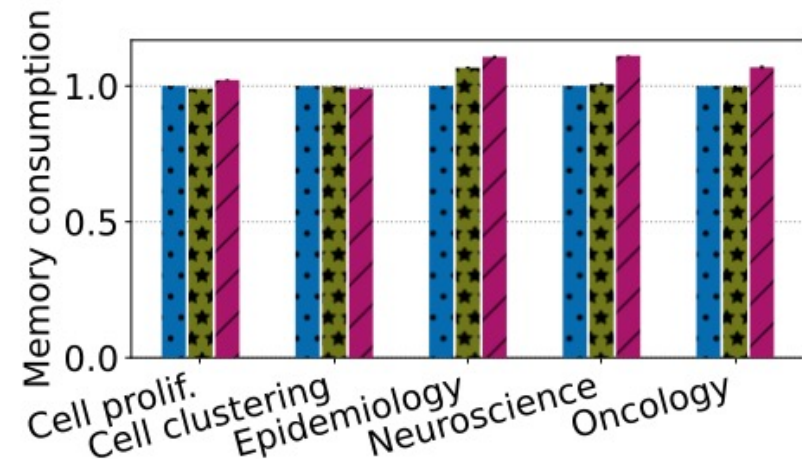
(a) Whole simulation



(b) Build time



(c) Search time (indirect)



(d) Memory consumption

Memory allocator comparison

