Introduction: Performance and scalability

The performance and scalability of simulations in Morpheus heavily depend on the type of (multi-scale) model that is being simulated. It is therefore difficult to make general statements on the computational efficiency. However, we can test the performance on a set of “benchmark” models that form the modules from which more complex model can be constructed.

We have tested the performance of ODE lattices, reaction-diffusion (PDE) models and cellular Potts models, using the available Example models. The results show the execution time and memory consumption for these models as well as their scalability in terms of problem size and scalability in terms of efficiency of multi-threading.

Performance measurements

To quantify performance, we measured the following aspects for each simulation:

- **Execution time** in terms of the (wall time), using the C+ + function `gettimeofday()` available in `<sys/time.h>`. The execution time does not include the time needed for initialization, analysis and visualization.
- **Memory usage** in terms of the physical memory (RAM) used by the simulation, using the resident set size (RSS) from the `/proc/self/stat` pseudo-file.

Scaling with problem size

We investigated the scalability with respect to problem size to see how performance in terms of the execution time (wall time) and memory usage (RAM) scales with increasing population size or lattice size.

We calculate and plotted both the execution time and memory usage in:

- **Absolute** terms: execution time in seconds (sec) and memory in megabytes (MB).
- **Relative** terms: execution time and memory per cell or lattice site in millisecond (msec) or kilobyte (kB).

Performance in absolute sense provides a sense of the problems sizes that are practically manageable within certain time and memory constraints.

Performance in relative sense shows the scalability of the simulation for problem sizes. Ideally, the performance per cell or lattice site stays constant or decreases with increasing problem sizes.

Scalability of multi-threading

We have also measured the scalability with respect to the number of openMP threads to see how the performance scale with the number of concurrent threads.

We measured the execution time (wall time) for each of the simulation run on in 1, 2, 4, 6 threads. Comparison of these execution times shows the speed-ups that can be achieved by adding concurrent

threads.

Methods

Benchmark models

The models used in performance tests are available as Example models:

- ODE lattices: Lateral Signaling
- Reaction-diffusion (PDE): Activator-Inhibitor (2D)
- Cellular Potts models (CPM): Cell Sorting (2D)
- Multi-scale (CPM+PDE): Vascular Patterning

The models are run without analysis and visualization tools and execution time is measured from StartTime to StopTime. The time for initialization is excluded since this vanishes for large jobs.

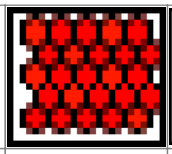
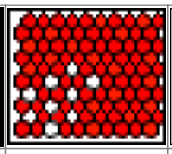
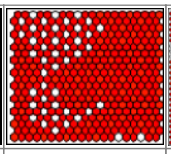
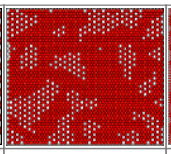
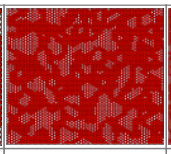
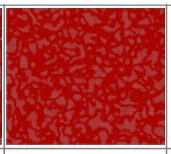
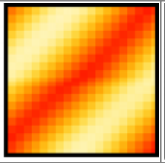
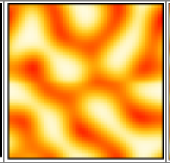
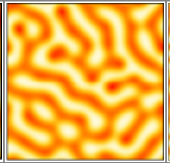
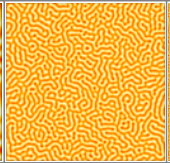
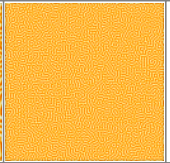
Hardware

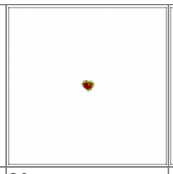
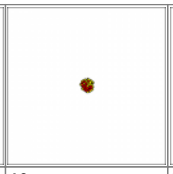
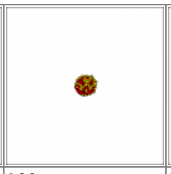
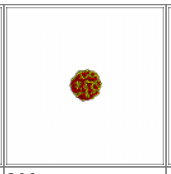
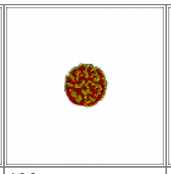
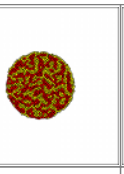
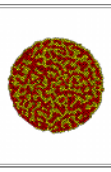
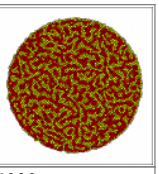
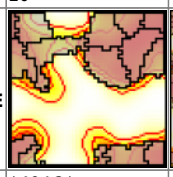
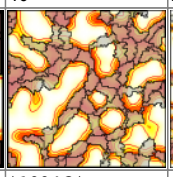
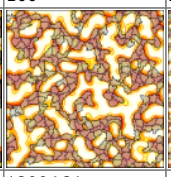
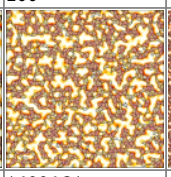
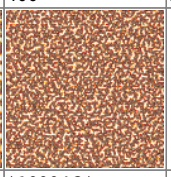
All simulations were performed on a Intel Core i7-860 vPro.

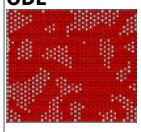
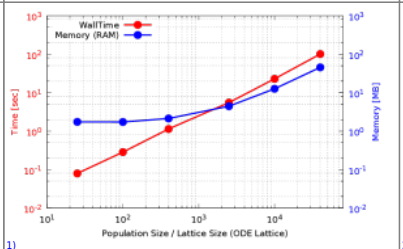
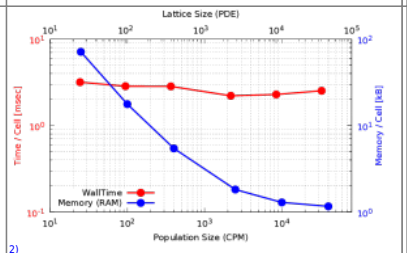
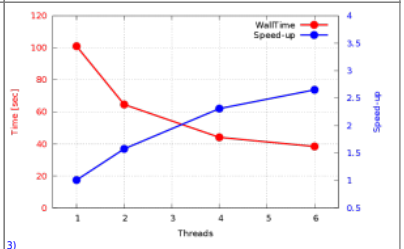
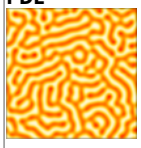
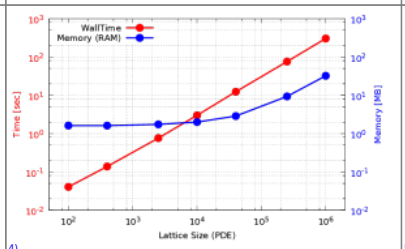
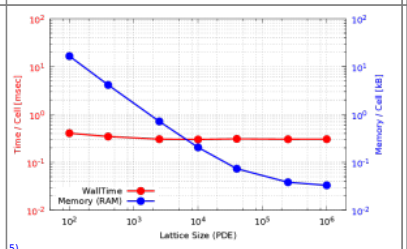
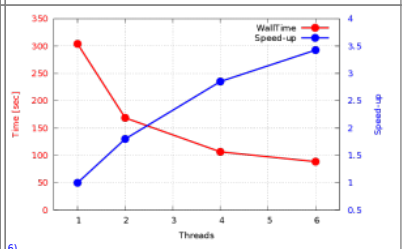
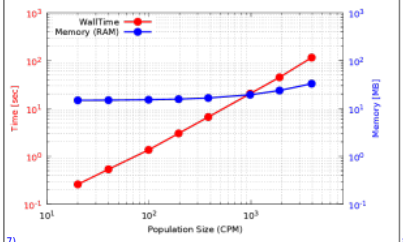
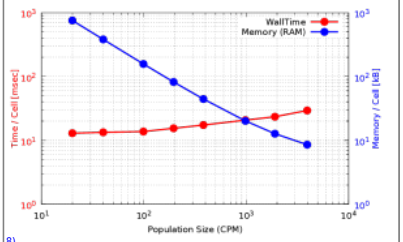
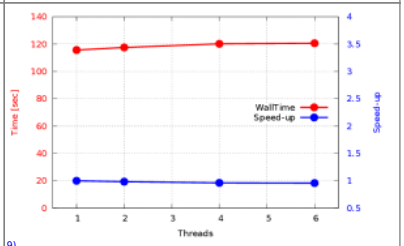
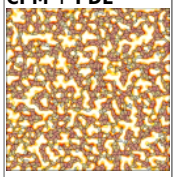
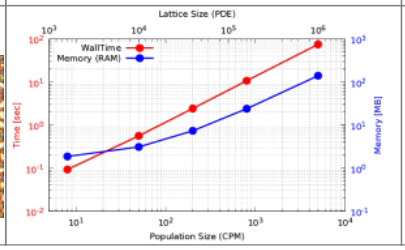
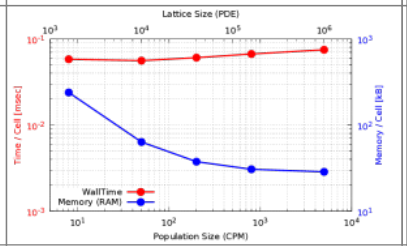
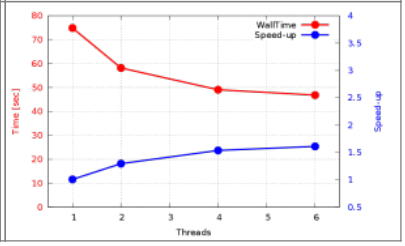
Hardware specification

# of Cores	4
# of Threads	8 (hyperthreading)
Clock Speed	2.8 GHz
Cache	8 MB
Memory	20 GB

Results

ODE								
Cells	25	100	400	2500	10000	40000		
PDE								
Lattice	10^2	20^2	50^2	100^2	200^2	500^2	1000^2	

CPM								
Cells	20	40	100	200	400	1000	2000	4000
CPM + PDE								
Lattice	40^2	100^2	200^2	400^2	1000^2			
Cells	8	50	200	800	5000			

	Problem size (absolute)	**Problem size (relative)**	**Multi-threading**
Description	Total execution time (red) and memory usage (blue) of simulation, excl. initialization and visualization	Execution time (red) and memory usage (blue), relative to number of cells and/or lattice sites	Execution time and speed-up as a function of number of openMP threads
ODE 	 1)	 2)	 3)
PDE 	 4)	 5)	 6)
CPM	 7)	 8)	 9)
CPM + PDE 			

1)

Execution time is linearly with number of cells. Total memory usage smaller than 20Mb.

2)

Exec. time per cells is approx. constant. Contribution of memory overhead decreases with problem size.

3)

Reasonable scalability of multithreading due to concurrent processing of intracellular ODEs and `NeighborReporters` for each cell.

4)

Execution time is linearly with number of lattice sites. Total memory usage smaller than 20Mb.

5)

Exec. time per lattice site is approx. constant. Contribution of memory overhead decreases with

problem size.
6)

Good scalability of multithreading, esp. for 2 or 4 threads, due to parallelization of reaction step by domain decomposition of lattice along y-axis. Parallelization of `Diffusion` is only done for large 3D lattices.
7)

Execution time is almost linearly with number of CPM cells. Small memory footprint, despite `edgelist` tracking.
8)

Exec. time per CPM cell is almost constant, although performance decreases for larger systems. Decrease of memory usage per cell is here mostly due to use of large lattice in all cases.
9)

Parallel processing is not available for CPM simulations. Therefore, multithreading does not results in speed-up. Instead, the multithreading overhead even slightly decreases performance.