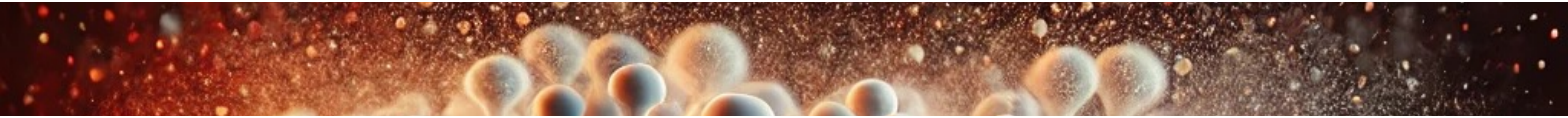
PSE Molekulardynamik
Sheet 4: Thermostats, Rayleigh-Taylor instability and “falling drops”



Group A: Daniel Schade, Ashutosh Solanki, Robin Cleve

21.06.2024

Table of contents

1. **Thermostats**
2. **Rayleigh-Taylor instability**
   - 2.1 Periodic boundaries
   - 2.2 Further steps
3. **Simulation of a falling drop - Liquid**
4. **Performance Measurement and Profiling**
5. **Tuning the sequential Performance**

Thermostats

- Calculate temperature of the system for all particles:

$$T = \frac{\sum_i m_i |\mathbf{v}_i|^2}{\#\text{dimensions} \times \#\text{particles}}$$

- Rescale velocity for all particles:

$$\mathbf{v}_i \leftarrow b \cdot \mathbf{v}_i, \quad \text{where} \quad b := \sqrt{\frac{T_{\text{new}}}{T_{\text{current}}}}$$

- Apply gradually, limit temperature change of the system by a certain value
- **Temperature of System with with static particles is not defined**

➡ Initialize system with Brownian motion!

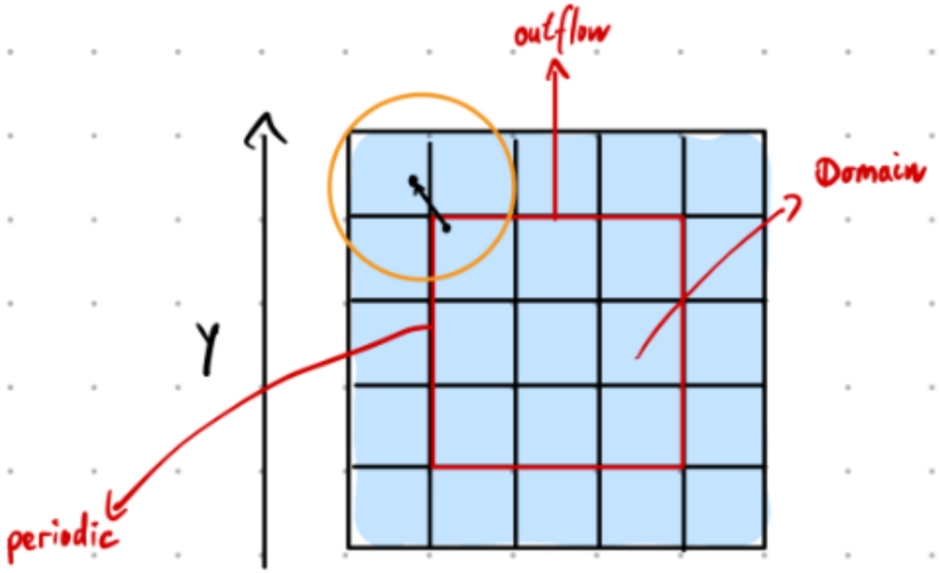
Rayleigh-Taylor instability

Periodic boundaries

- **Main challenges:**
  1. Teleporting of particles that left the domain

     -> “easier part” of the implementation
  2. Calculate forces for particles close to the border of the domain

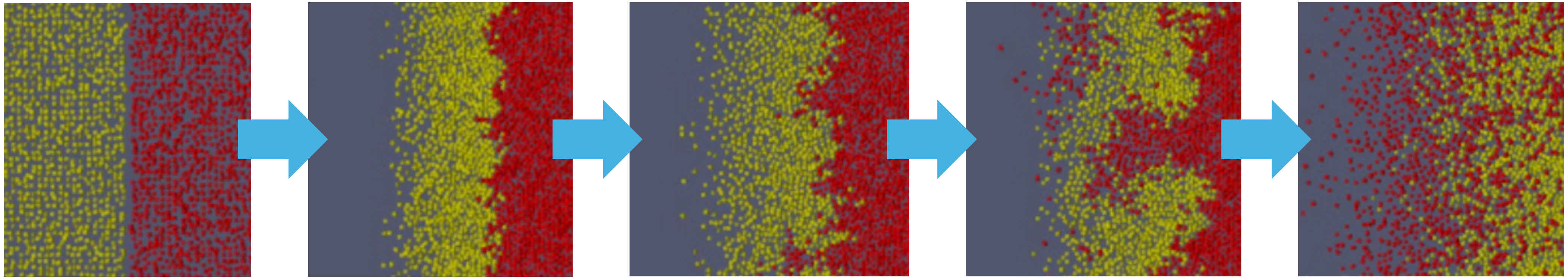
     -> Interpret cell of the opposite side as halo cell
- **General approach:**
  - split domain up in Sides, Edges, Corners
- **Edge case:**
  - Particle in the boundary cell corner
  - Which boundary did it cross
  - For now it depends on the processing order



Rayleigh-Taylor instability

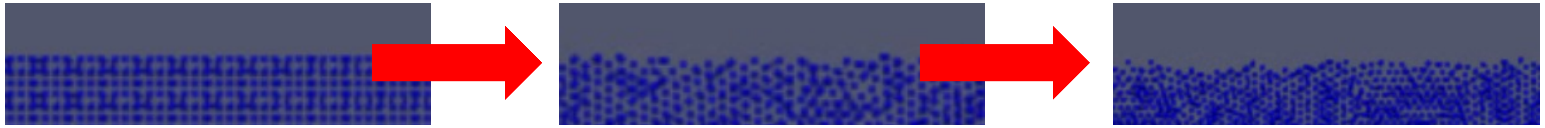
Further steps

1. Gravity
   - Add force along the y-axis to each particle
2. Lorentz-Berthelot Mixing Rules
   - Epsilon and Delta can now be configured in each particle individually
   - Adapt Leonard Jones force
3. Simulation

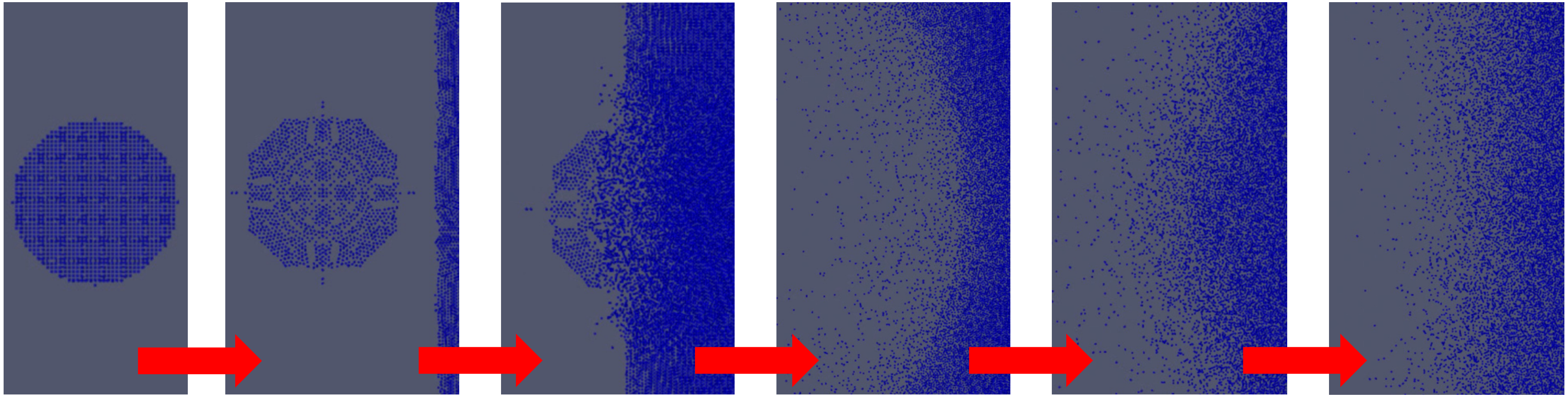


Simulation of a falling drop - Liquid

Prepare liquid:

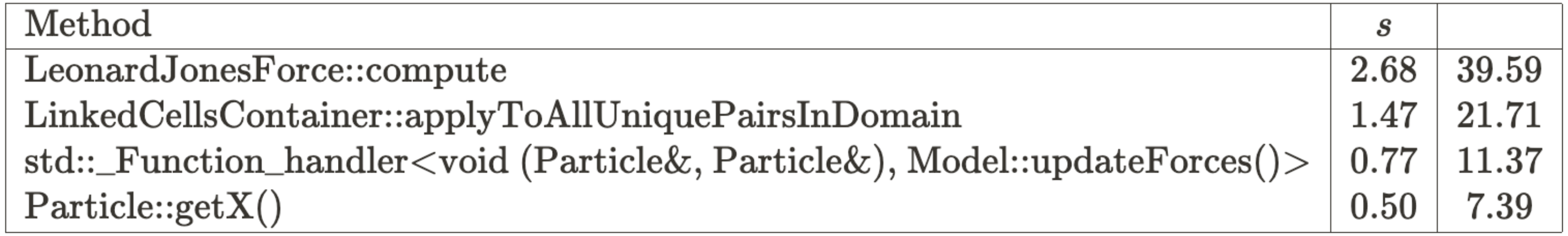


→ “checkpoint 1”



Performance Measurement and Profiling

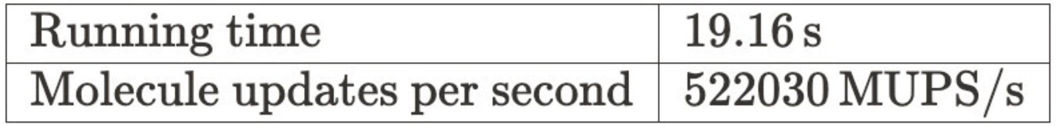
- These methods consume the most runtime, 80.06% of total execution time



Method	*s*	
LeonardJonesForce::compute	2.68	39.59
LinkedCellsContainer::applyToAllUniquePairsInDomain	1.47	21.71
std::_Function_handler<void (Particle&, Particle&), Model::updateForces()>	0.77	11.37
Particle::getX()	0.50	7.39

- Observations:
  - Iterating over cells of the linked cells container consume a lot of time, as well as the
  - getter method for the position of molecule

- Linux cluster - Fist 100 iterations of the Rayleigh-Taylor instability of task 2



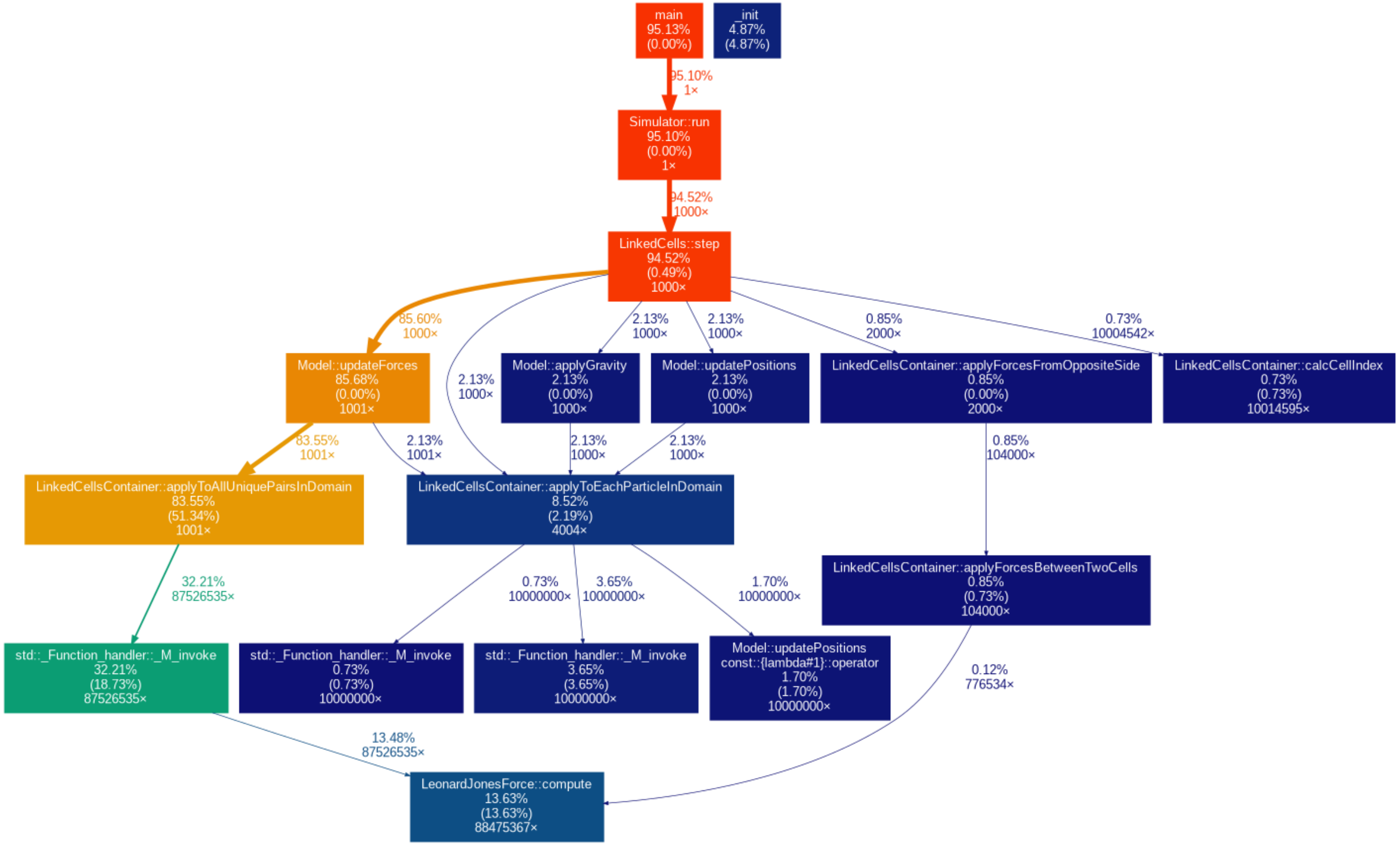
Running time	19.16 s
Molecule updates per second	522030 MUPS/s

Tuning the sequential performance

- Leonard-Jones force between two particles
  - Calculate difference in distance of the particles only once
  - **Speed up by 1.2s!**
- Reduce getter calls
  - establishing friend classes and adding multiple compiler flags (e.g. -Ofast)
  - **Speed up by 2.04s!**
- Conclusion
  - Only constant speed up, time complexity of program stays the same
  - More complex changes, like caching Leonard-Jones parameters, may be implemented in the following weeks

Tuning the sequential performance

Graph after second optimization step



main
95.13%
(0.00%)

_init
4.87%
(4.87%)

95.10%
1×

Simulator::run
95.10%
(0.00%)
1×

94.52%
1000×

LinkedCells::step
94.52%
(0.49%)
1000×

85.60%
1000×

2.13%
1000×

2.13%
1000×

2.13%
1000×

0.85%
2000×

0.73%
10004542×

Model::updateForces
85.68%
(0.00%)
1001×

Model::applyGravity
2.13%
(0.00%)
1000×

Model::updatePositions
2.13%
(0.00%)
1000×

LinkedCellsContainer::applyForcesFromOppositeSide
0.85%
(0.00%)
2000×

LinkedCellsContainer::calcCellIndex
0.73%
(0.73%)
10014595×

83.55%
1001×

2.13%
1001×

2.13%
1000×

2.13%
1000×

0.85%
104000×

LinkedCellsContainer::applyToAllUniquePairsInDomain
83.55%
(51.34%)
1001×

LinkedCellsContainer::applyToEachParticleInDomain
8.52%
(2.19%)
4004×

LinkedCellsContainer::applyForcesBetweenTwoCells
0.85%
(0.73%)
104000×

32.21%
87526535×

0.73%
10000000×

3.65%
10000000×

1.70%
10000000×

0.12%
776534×

std::_Function_handler::_M_invoke
32.21%
(18.73%)
87526535×

std::_Function_handler::_M_invoke
0.73%
(0.73%)
10000000×

std::_Function_handler::_M_invoke
3.65%
(3.65%)
10000000×

Model::updatePositions
const::{lambda#1}::operator
1.70%
(1.70%)
10000000×

13.48%
87526535×

LeonardJonesForce::compute
13.63%
(13.63%)
88475367×