Scientific Computing (PSE) Molecular Dynamics
Group D

Worksheet 4

02

SIMULATION TASKS

Task 1 to 3 -Worksheet 4

Thermostats

Integrate system temperature into the simulation to enable heating or cooling

Rayleigh-Taylor instability simulation

Fluid with higher density above fluid with lower density subjected to gravitational forces

Falling drop in liquid simulation

With possibility of saving the current state of molecules to restart a new simulation

TIMELINE

Refactoring / Rewriting of Code

to be continued

Integration of Thermostats

Added the necessary formulas and constraints as well as adapted the XML input to incorporate temperature related parameters

Benchmarking, Linux-Cluster and Testing

Mostly benchmarked on our own local machines as the cluster was a bit more setup work/changes than expected

Due to the following performance enhancing tasks we have rewritten a lot of code using different approaches, firstly to directly improve performance and secondly, to be able to make use of the tools of this week's worksheet.
After that we implemented the new functionality and continued with the simulations and benchmarking

04

PERFORMANCE TUNING AND ENHANCING

Task 4 & 5

VTune and Advisor

Profiling tool for detecting bottlenecks and inefficiencies in the code that can be improved for performance enhancement

Compiler comparison

Compare results before and after optimizations and compare using GNU compiler g++ vs. Intel’s LLVM icpx

Linux Cluster Benchmark

Test out optimized (and unoptimized) code on the CooLMUC Linux cluster for better comparability

Five methods to enable this functionality

```
apply(ParticleContainer<DIMENSIONS> &particles,
    bool useBrown,double brownMotion) {...}
```

```
scaleVelocities
(ParticleContainer<DIMENSIONS> &particles,
        double scalingFactor) {...}
```

```
calculateCurrentTemperature
(ParticleContainer<DIMENSIONS>
      &particles) {...}
```

```
calculateKineticEnergy
(ParticleContainer<DIMENSIONS> &particles) {...}
```

```
initializeVelocities
(ParticleContainer<DIMENSIONS> &particles,
 bool useBrown, double brownMotion) {...}
```

THERMOSTATS

Features

Directly setting a temperature via velocity
Gradual velocity scaling

Parameters of Thermostat

initial temperature of system `t_init`
target temperature `t_target`
maximum temperature delta `delta_t`
frequency of application `nthermostat`

Basic Testing

`test_heating`

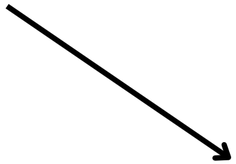
set initial temperature
set higher target temperature

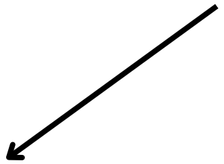
`test_cooling`

set initial temperature
set lower target temperature

`test_holding_temperature`

set initial temperature
and equal target temperature





check if temperature was reached
check if delta was taken into account

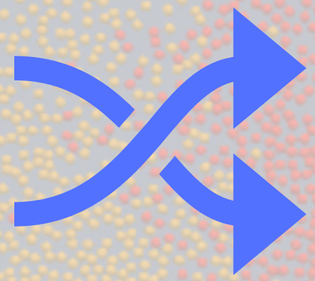
Implementation

Gravitational downwards pull

Using the simple gravity formula of mass times g, additional input parameter in XML to set a gravitational factor for the downwards pull along the y-axis

Possibility to model different fluids

Adapted the input schema and the functionality to enable different sigmas, epsilons and masses and have cuboids or discs with different parameters (which was relatively easy because we firstly implemented it that way by accident last week)



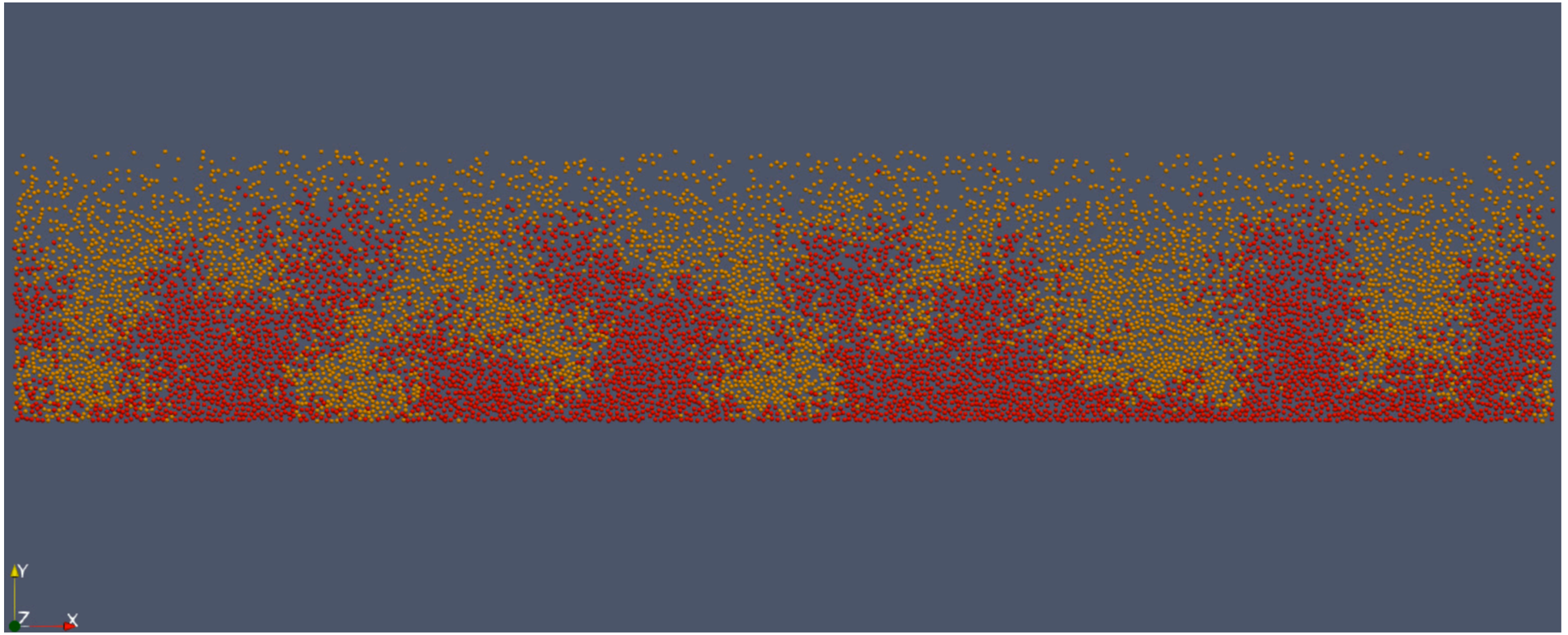
Lorentz-Berthelot mixing rule

Added new force calculation constraint (when molecules are not of the same type)

SIMULATION 1

Colored by type (the denser fluid is orange)

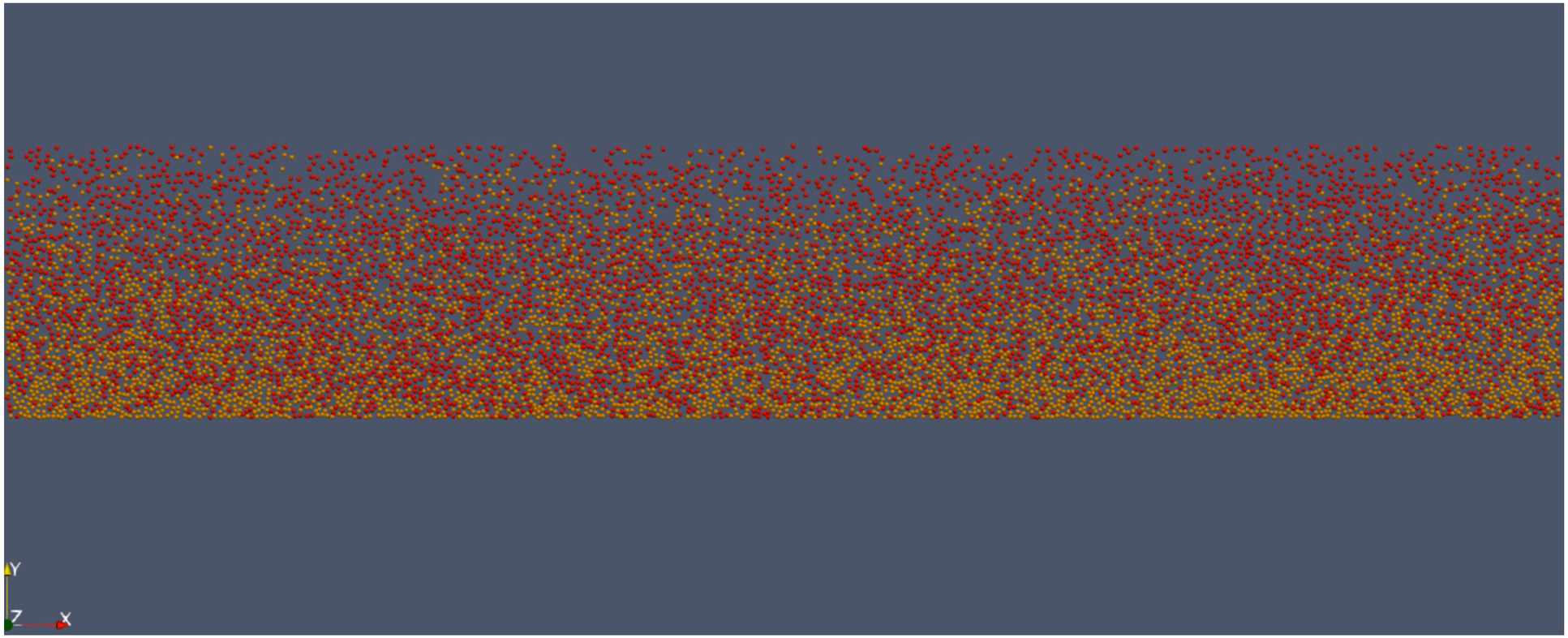
08



SIMULATION 2

Colored by type (the denser fluid is orange)

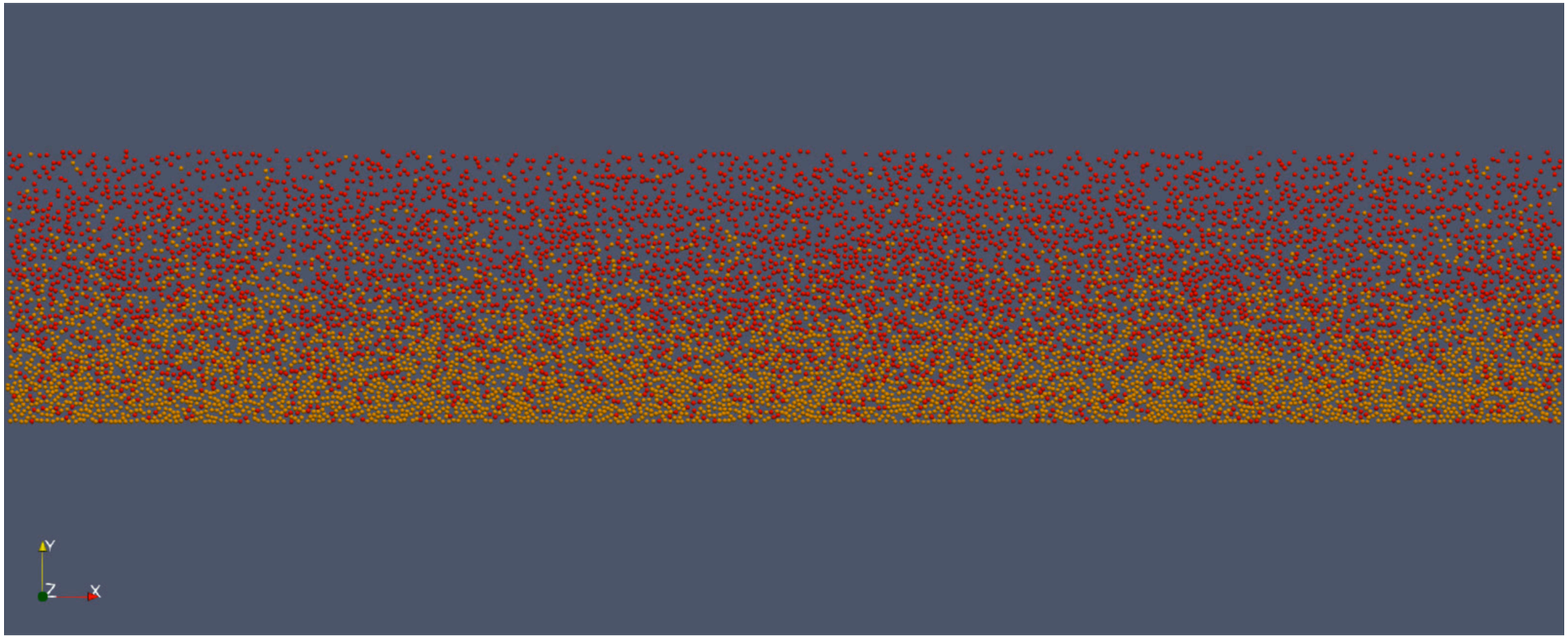
09



SIMULATION 3

Colored by type (the denser fluid is orange)

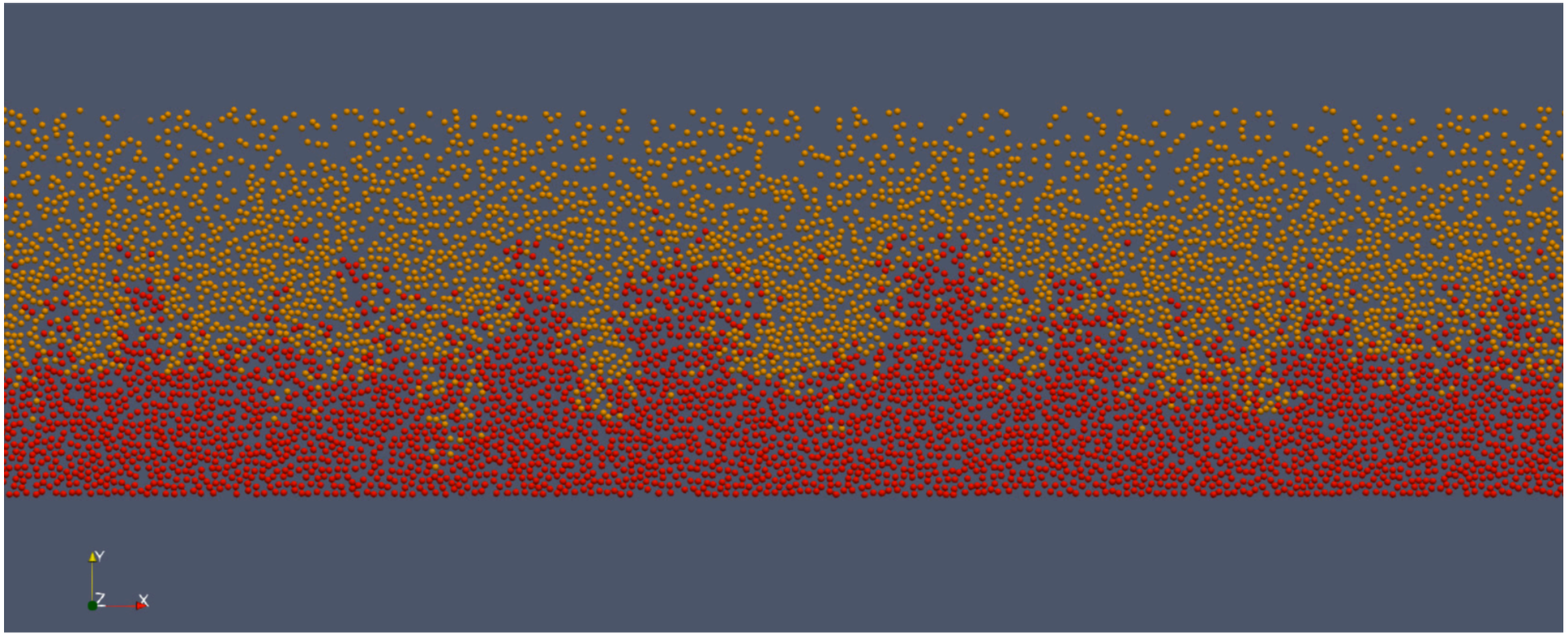
10



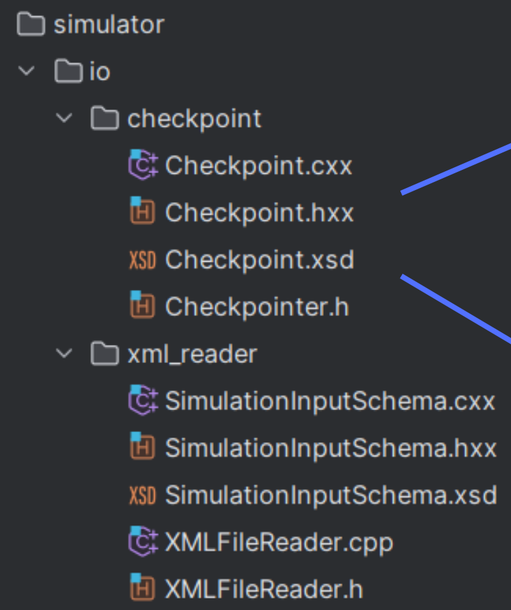
SIMULATION 4

Colored by type (the denser fluid is orange)

11



Checkpointing



Checkpointer

Similar to the Input structure, we created a new XSD schema for a checkpointer that can save the current state of the system

XML output

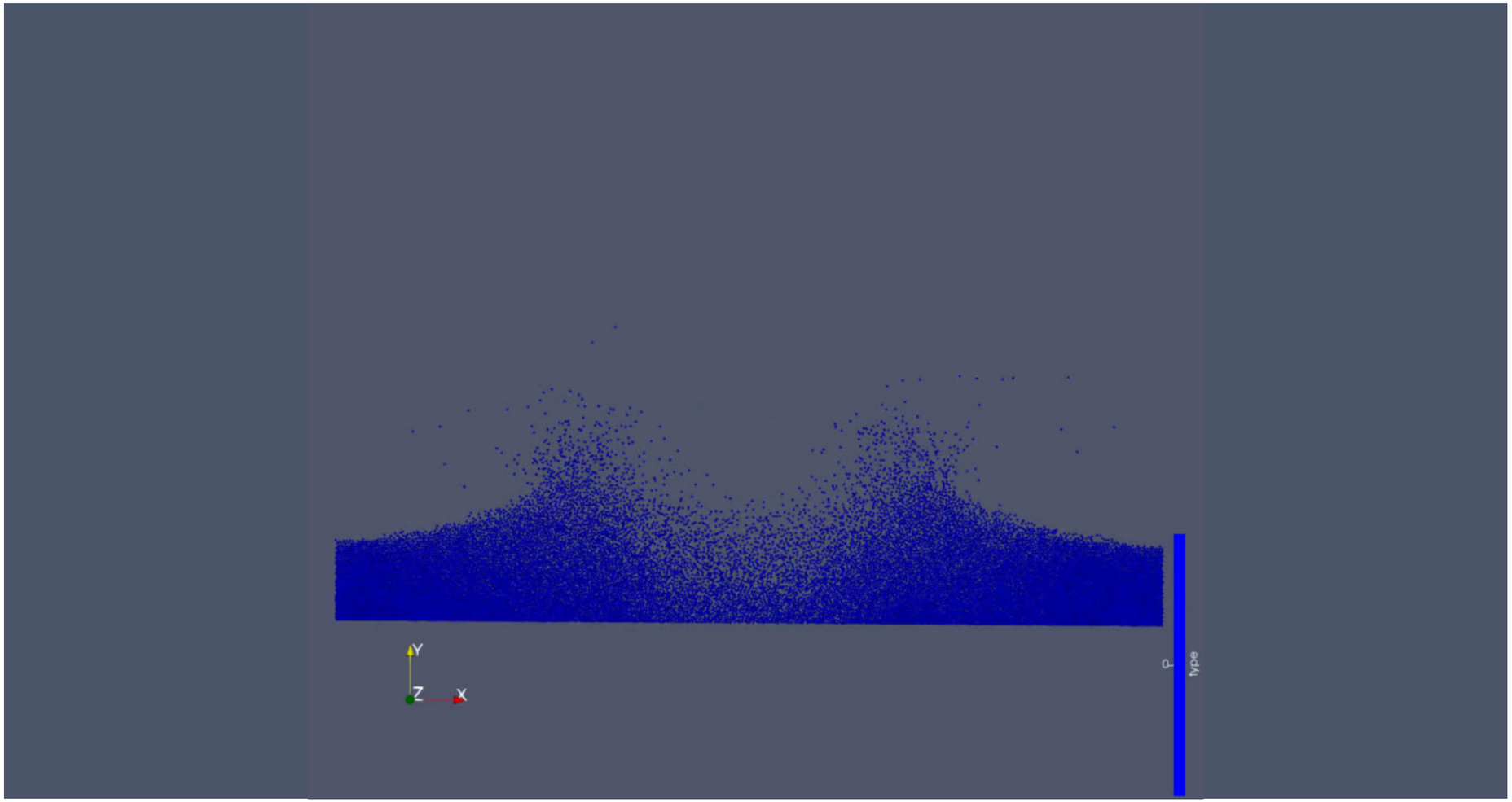
This way the checkpoint can seamlessly be used as a form of input for a follow up simulation that should continue from that checkpoint

```
<xs:complexType>
   <xs:sequence>
     <xs:element maxOccurs="unbounded" name="particle" type="particle"/>
   </xs:sequence>
</xs:complexType>
```

SIMULATION 5

Type as color (only blue here)

13



OPTIMIZATION


```
using double_v = stdx::native_simd<double>;
using double_mask = stdx::native_simd_mask<double>;
using long_v = stdx::native_simd<long>;
using long_mask = stdx::native_simd_mask<long>;
using size_v = stdx::native_simd<size_t>;
```

Vectorization

changed the previous code to accomodate a vectorized approach for better performance and usability

This rather deeply rooted change led to quite some consequences regarding the connected classes, previous tests and configurations

```
template<const size_t DIMENSIONS>
struct VectorizedParticle {
  std::array<double_v, DIMENSIONS> position{};
  std::array<double_v, DIMENSIONS> force{};
  double_v mass{};
  long_v type;
  double_mask active;
```

Dimensions

The use of “DIMENSIONS” allows for a more flexible usage in different simulation scenarios

VTUNE & ADVISOR

Intel VTune
Intel Advisor

VTune

The profiler showed that most computational time is spent on calculate_force() and VTune showed no frontend issues, only backend issues which means the hardware is utilized very well

Advisor

Showed almost no suggestions
Only one advice about unaligned memory access, which after correction turned out to decrease the performance

TIME COMPARISON

Local Hardware: AMD Ryzen 9 7950X3D
16K/32T, 64GB RAM

LOCAL MACHINE

01:54:05 min
Taylor-Rayleigh Instability big

00:10:61 min
WS3 Collision of two bodies

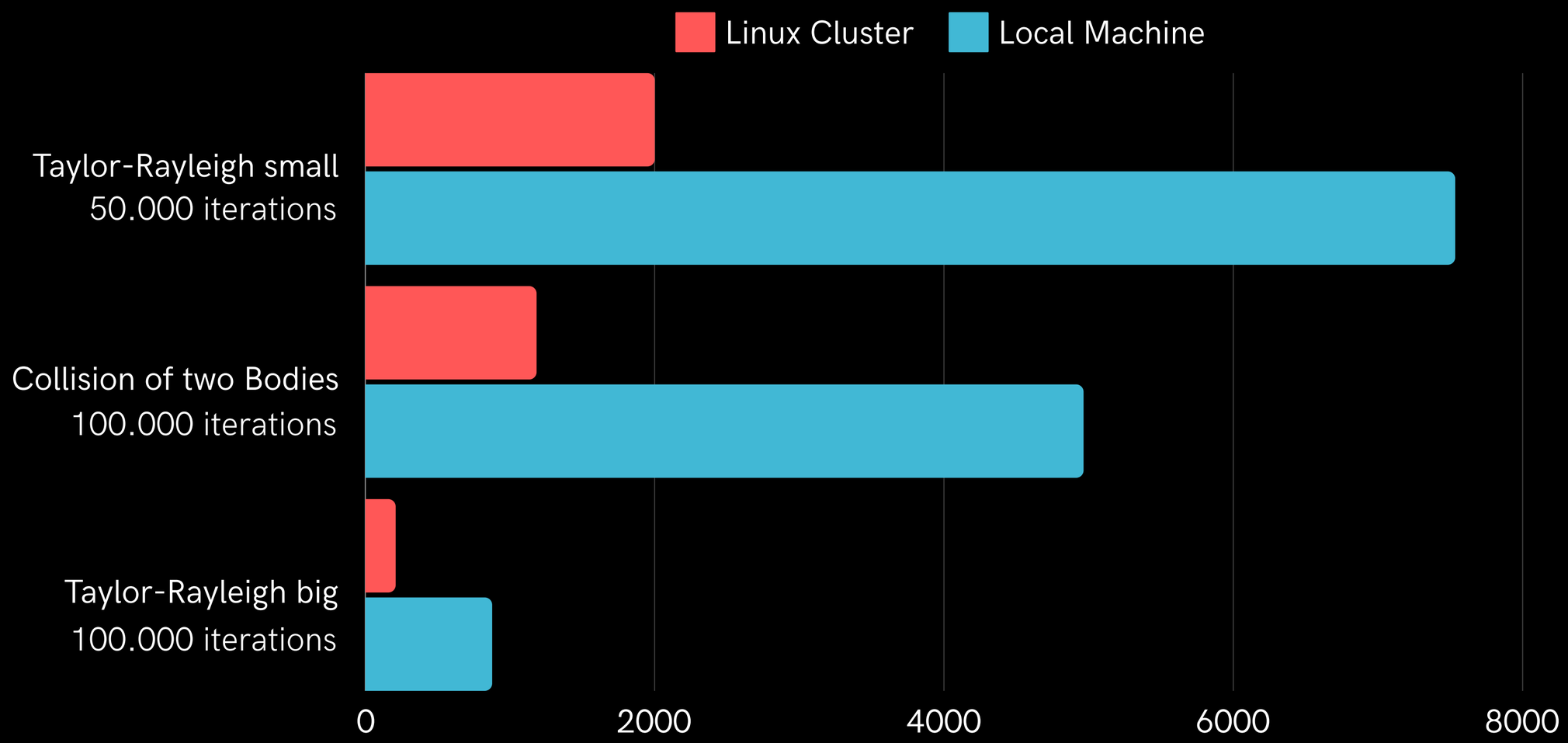
00:08:06 min
Taylor-Rayleigh Instability small

CLUSTER

08:16:46min
Taylor-Rayleigh Instability big

00:33:76 min
WS3 Collision of two bodies

00:39:79 min
Taylor-Rayleigh Instability small



MUPS/s (higher is better)

Local Hardware: AMD Ryzen 9 7950X3D 16K/32T, 64GB RAM

LINUX CLUSTER SERIAL JOB

1000 iterations of the large Taylor-Rayleigh instability simulation

MUPS/s: 236.07
On linux cluster

MUPS/s: 1.122,59
On local machine

TASK 4 BENCHMARK

full simulation of the large Taylor-Rayleigh instability experiment

MUPS/s: 201.42
On linux cluster

MUPS/s: 877.45
On local machine

CONTEST

FUTURE IMPROVEMENTS

PERFORMANCE

Parallelization, use of multi-core and threading which is now coming up with the last worksheet anyway

Maybe try out the usage of GPU hardware with very high parallelization

TESTING

Parallelization tests
Cleaning up test folder, adapting tests to the new structure

REFACTORING

Reduce line of codes by replacing old code with new implementations and deleting old features/implementations

THANK YOU FOR LISTENING

Scientific Computing (PSE) Molecular Dynamics
Group D

Johannes H. | Julius K. | Tim S.