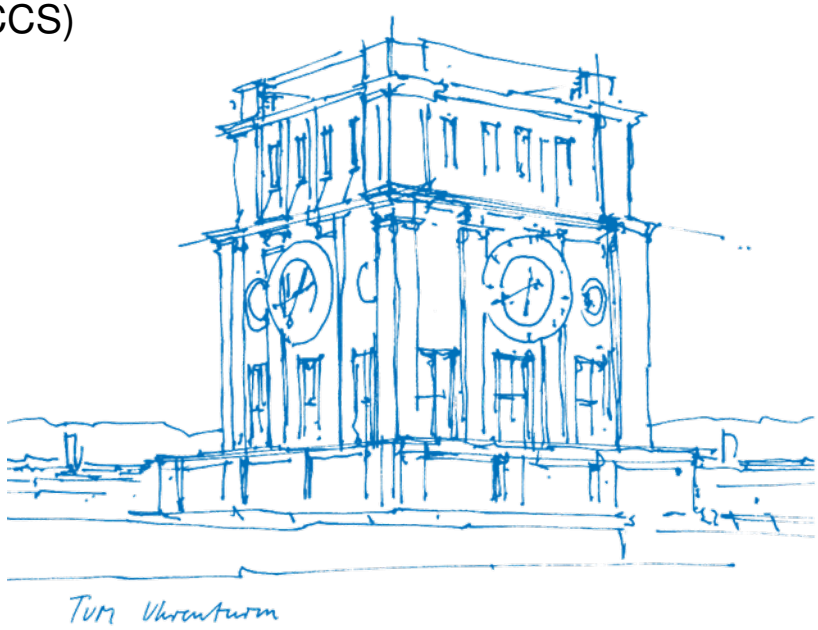
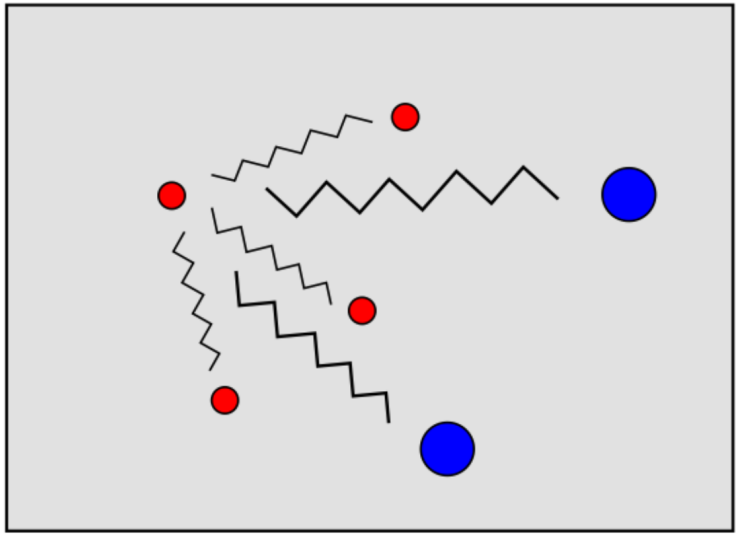
Self Optimizing Particle Simulations: Combining Expert Knowledge with Learning

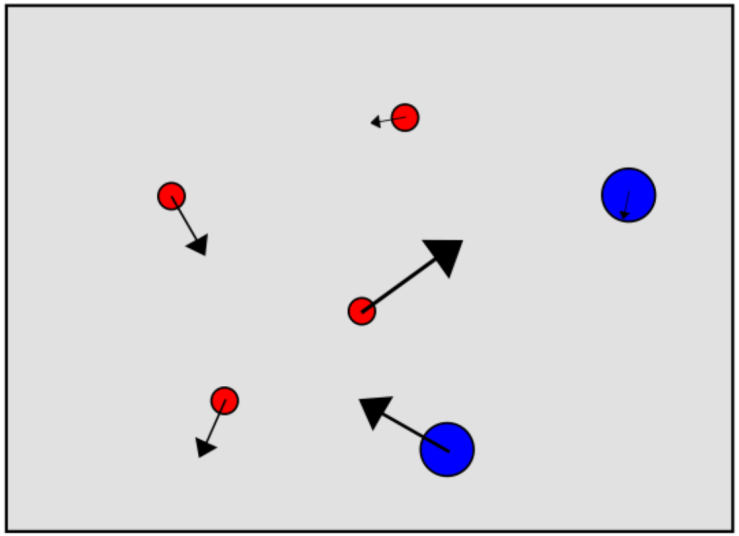
Fabio Gratl, Hans-Joachim Bungartz, Philipp Neumann
Technical University of Munich
TUM School of Computation, Information and Technology (CIT)
Chair of Scientific Computing in Computer Science (SCCS)
Garching, 02.05.2024



Particle Simulations: Core Ideas

- Model N particles
- Compute particle interactions
- Propagate according to Newton's Laws of Motion





$\Rightarrow$ How?

Moving Through Time

- Acceleration of one particle i according to Newton's 2nd Law:

$$\vec{a}_i = \dot{\vec{v}}_i = \ddot{\vec{r}}_i = \frac{\vec{F}_i}{m_i} \tag{1}$$

- Can be transferred to a system of $2 \cdot d \cdot N$ ODEs:

$$\dot{\vec{r}}_i = \vec{v}_i \tag{2}$$

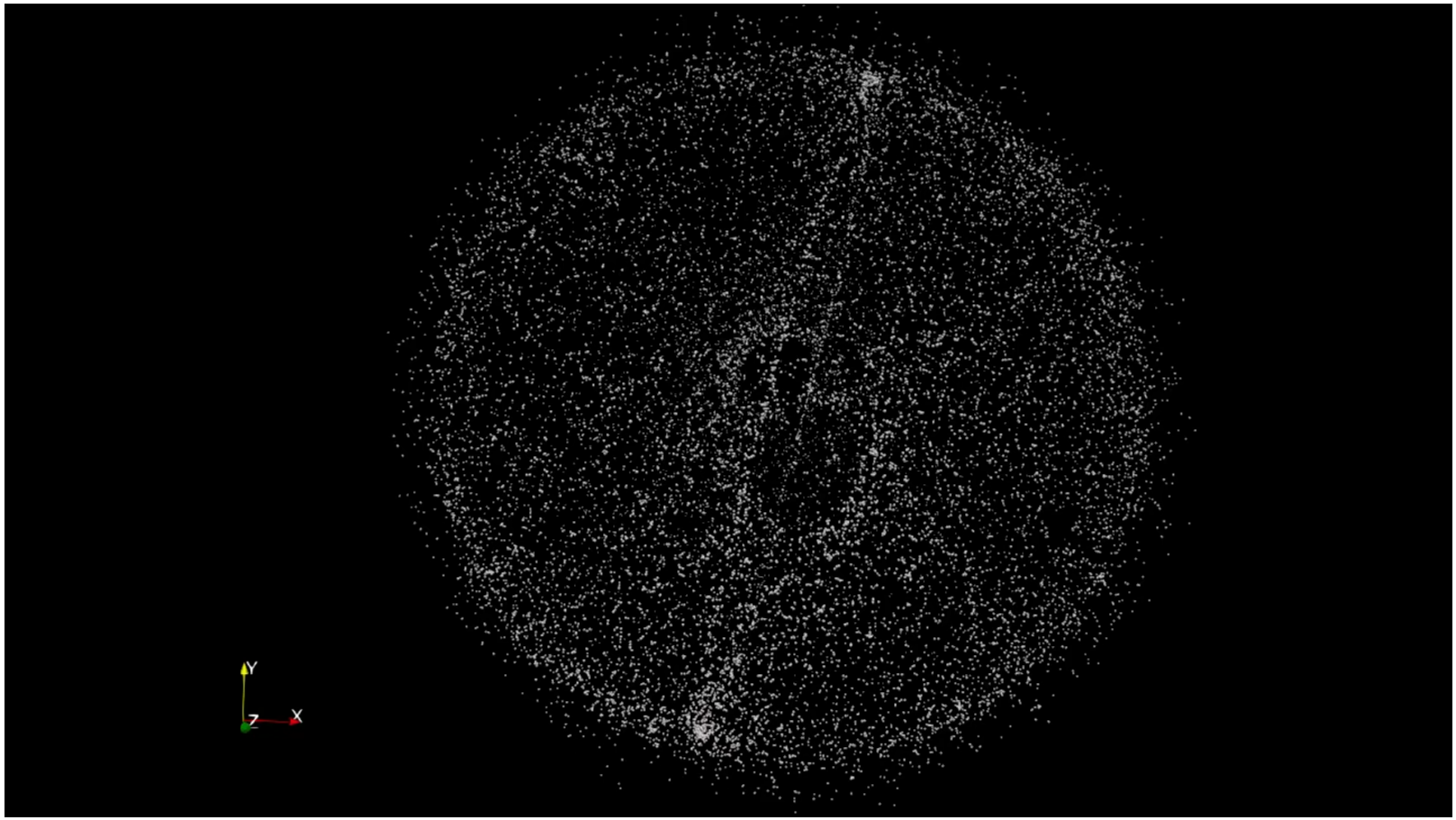
$$\dot{\vec{v}}_i = \vec{a}_i \tag{3}$$

- Requirements to the integrator:
  - Computationally cheap
  - Close to "true solution" $\Rightarrow$ Positions, System Energy, ...?
- Example: Velocity-Störmer-Verlet:

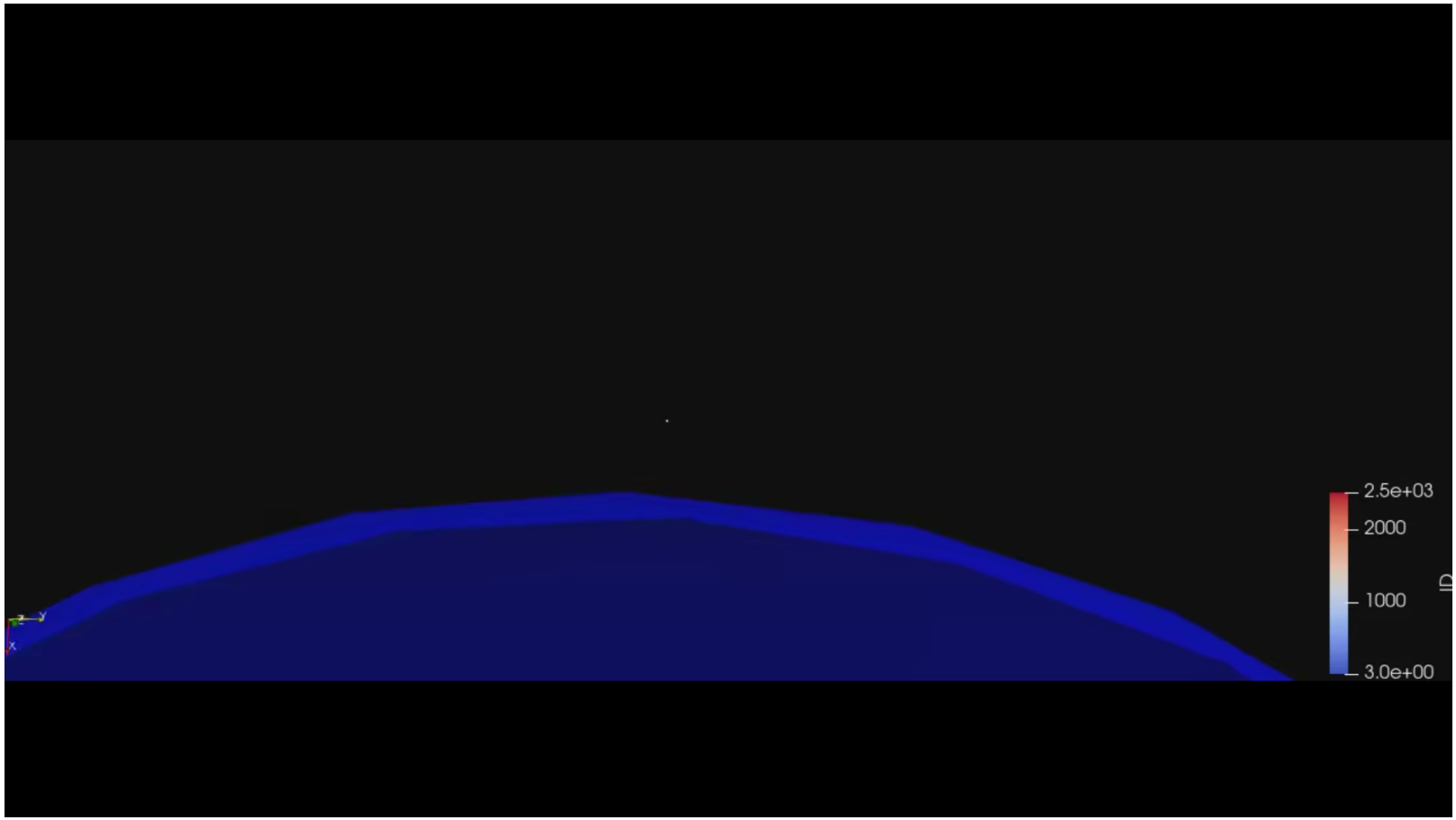
$$r_i(t_{n+1}) = r_i(t_n) + \Delta t \cdot v_i(t_n) + (\Delta t)^2 \frac{F_i(t_n)}{2m_i} \tag{4}$$

$$v_i(t_{n+1}) = v_i(t_n) + \Delta t \frac{F_i(t_n) + F_i(t_n + 1)}{2m_i} \tag{5}$$

Application: Space Debris

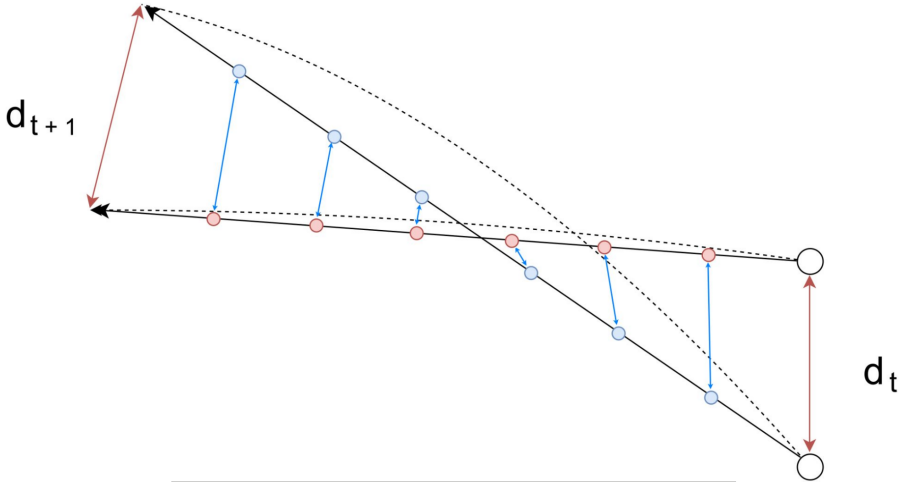


Application: Space Collisions

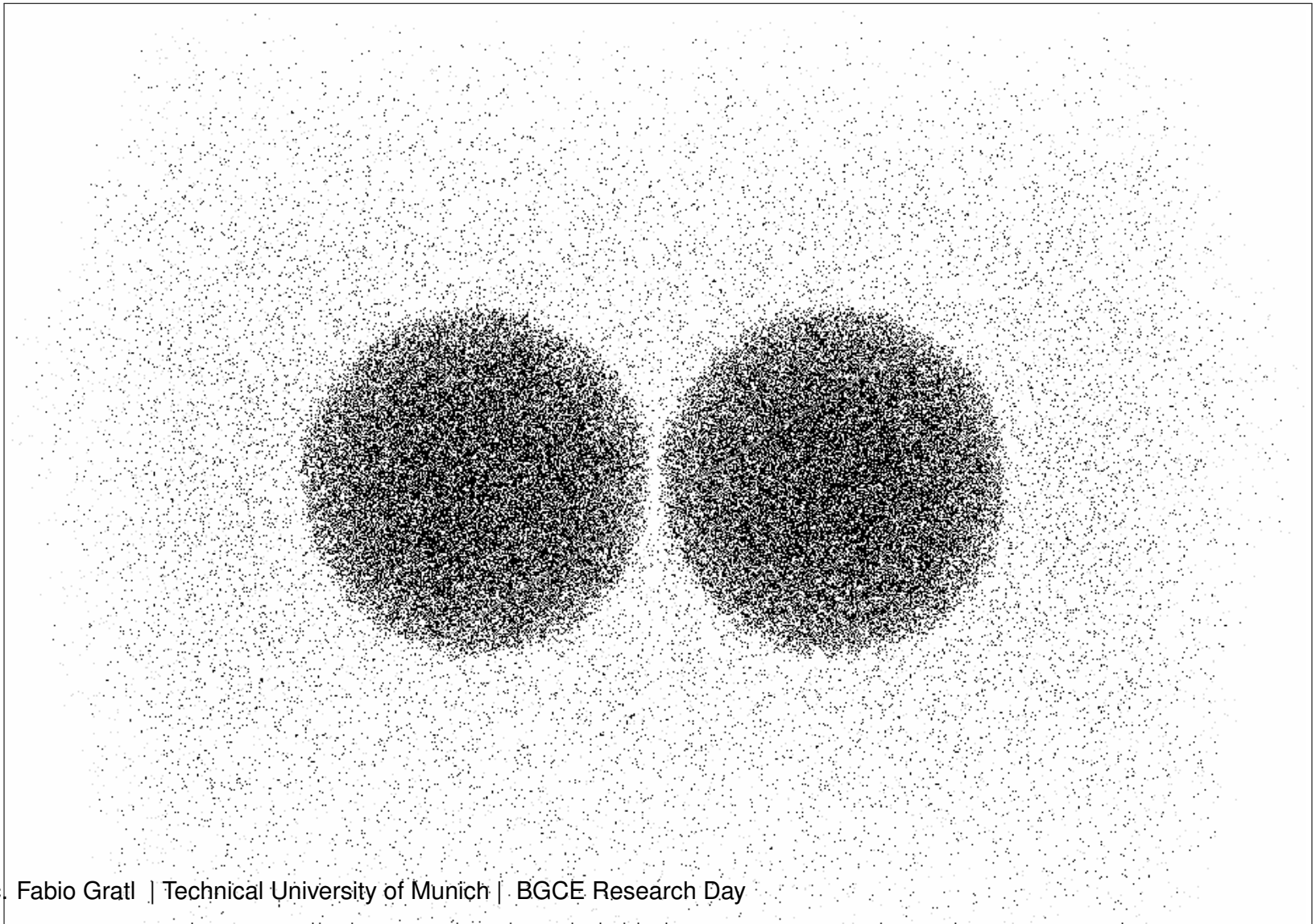


LArge-scale Deterministic Debris Simulation

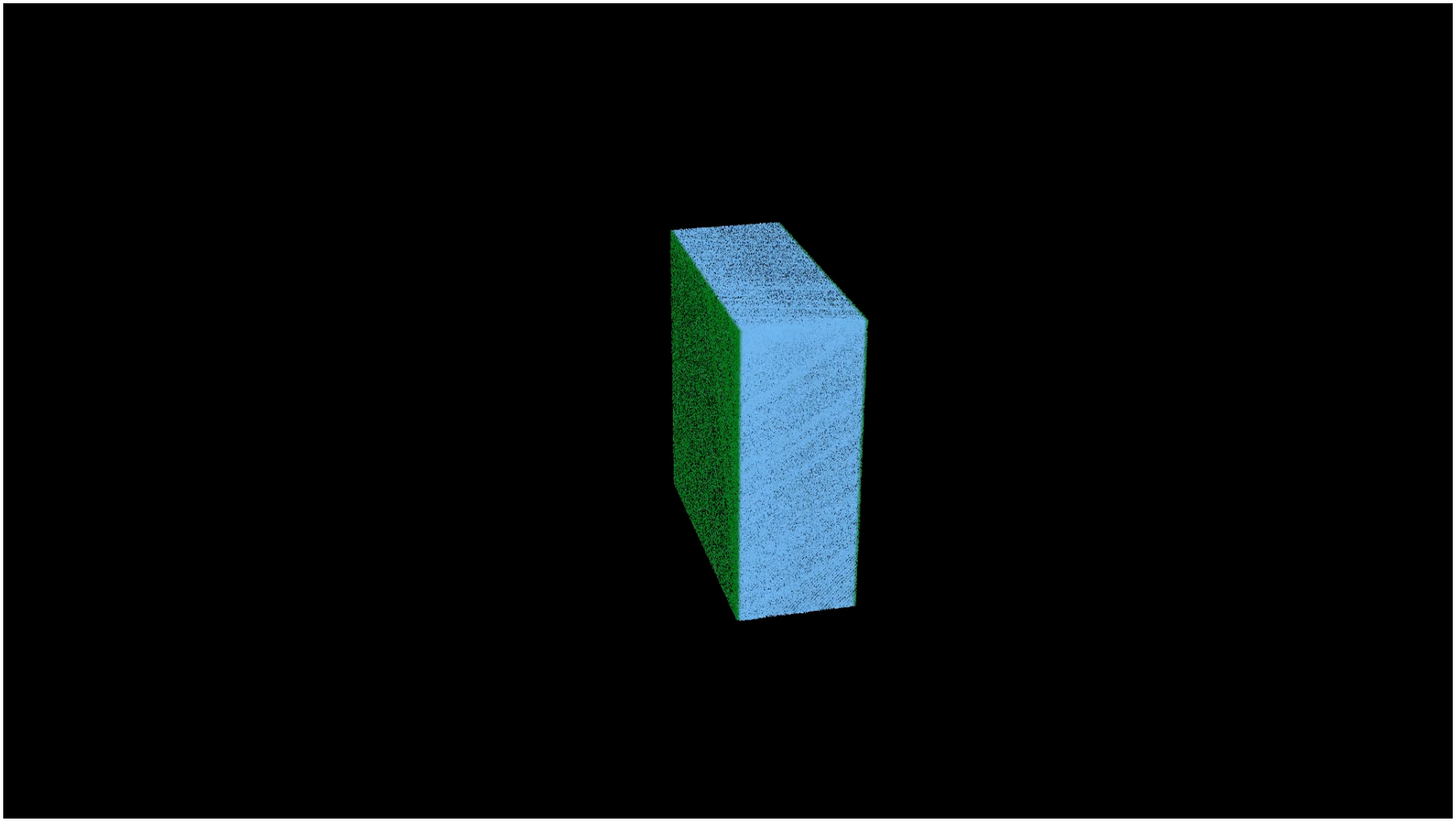
- Simulation of satellites as particles
- Trajectory modelling
- Newton's Laws of Motion
- Yoshida integrator
- Propagation perturbations
- Sub-timestep collision tracking
- N-Body problem $\Rightarrow O(N^2)$
- NASA Breakup Model
- https://github.com/esa/LADDS



Application: Molecular Dynamics Droplets



Application: Molecular Dynamics Exploding Liquid



Molecular Dynamics - Short Range

- Here: small rigid molecules as points without geometry
- Simulation of movement of particles
- Computation of pairwise forces
- Newton's Laws of Motion
- LeapFrog integrator
- N-Body problem $\Rightarrow O(N^2)$
- Force cut-off $\Rightarrow O(N)$

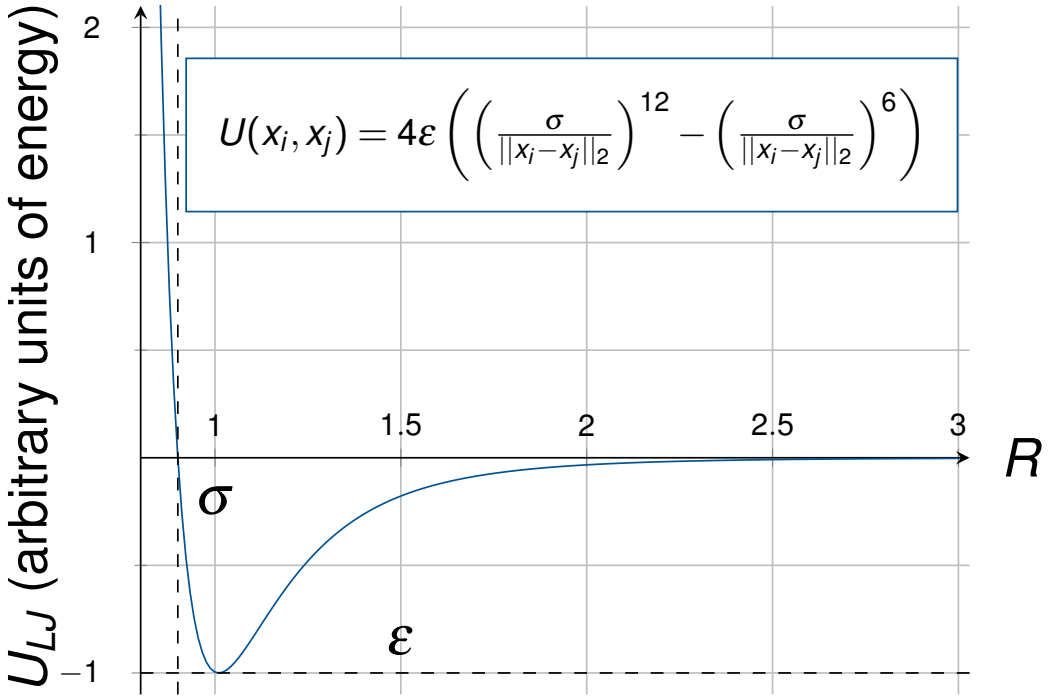
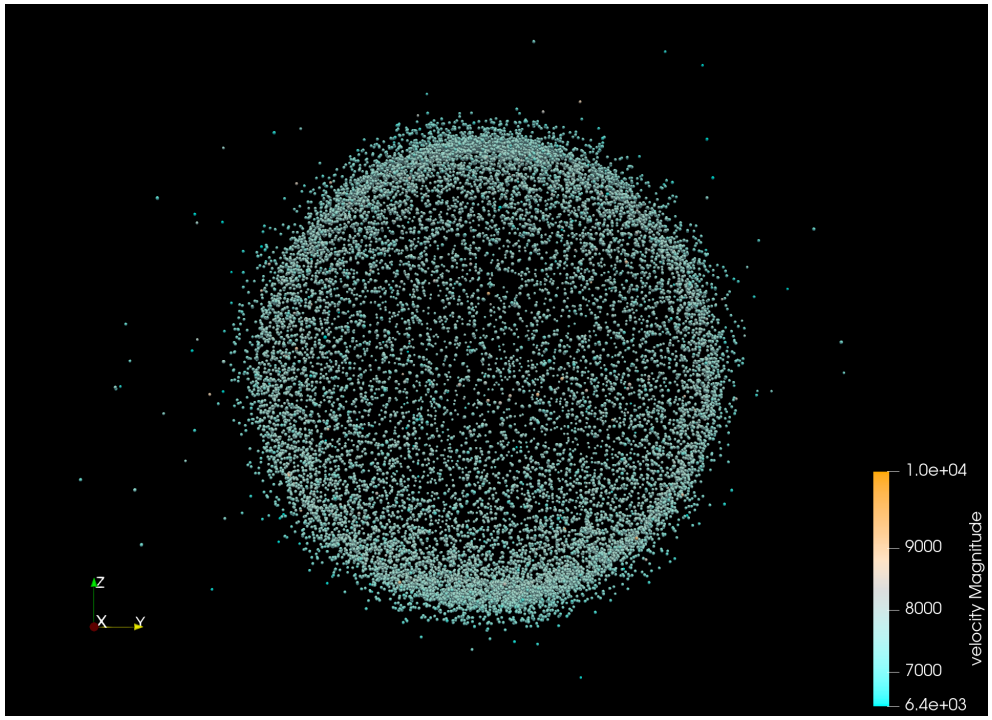


Figure: Lennard Jones Potential for $\varepsilon = 1$ and $\sigma = 0.9$

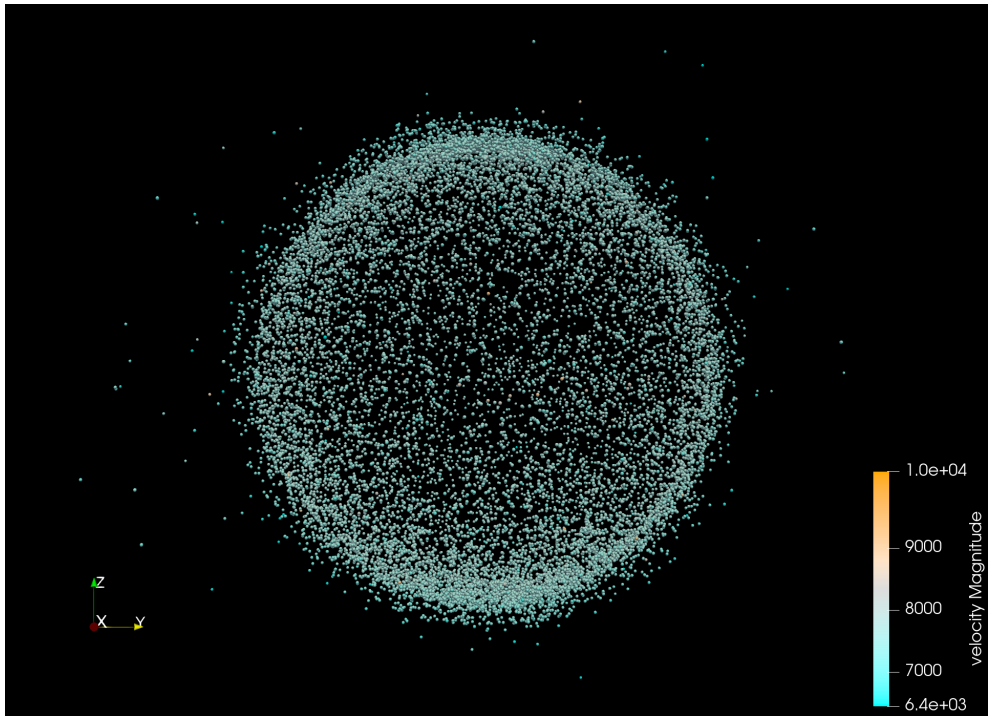
Particle Simulation Challenges

- Total number of particles
- Particle density
- (In-)Homogeneity
- Systems changing over time
- Many possible algorithms
- Arbitrary simulation types:
  MD, SPH, DEM, ...
- Overall goal:
  Minimize time to solution!

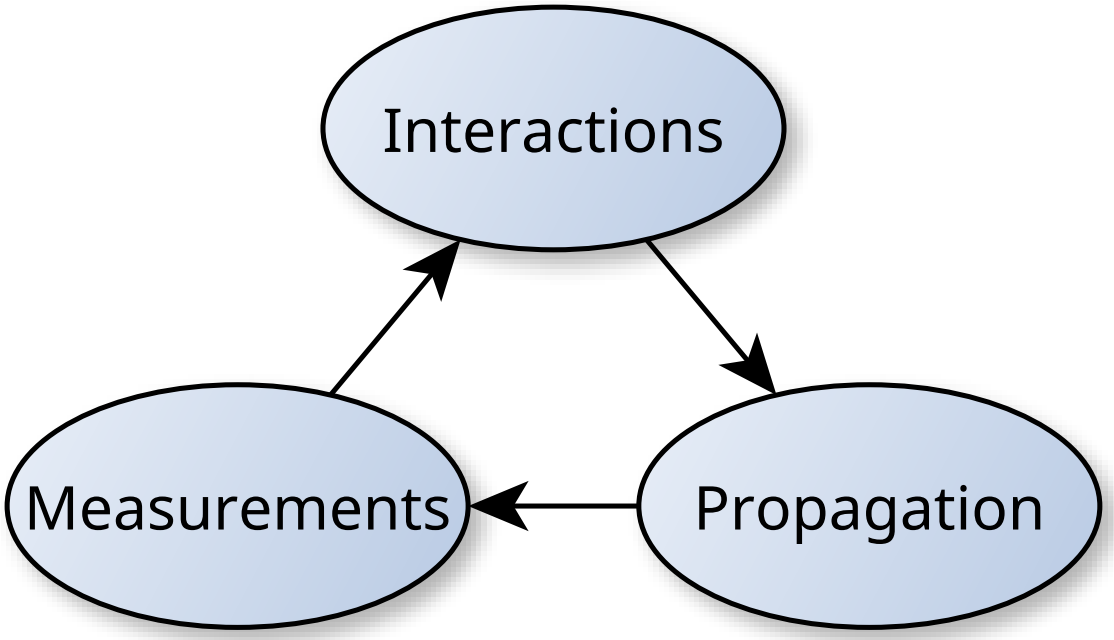


Particle Simulation Challenges

- Total number of particles
- Particle density
- (In-)Homogeneity
- Systems changing over time
- Many possible algorithms
- Arbitrary simulation types:
  MD, SPH, DEM, ...
- Overall goal:
  Minimize time to solution!
  Or energy!



Particle Simulation Abstraction

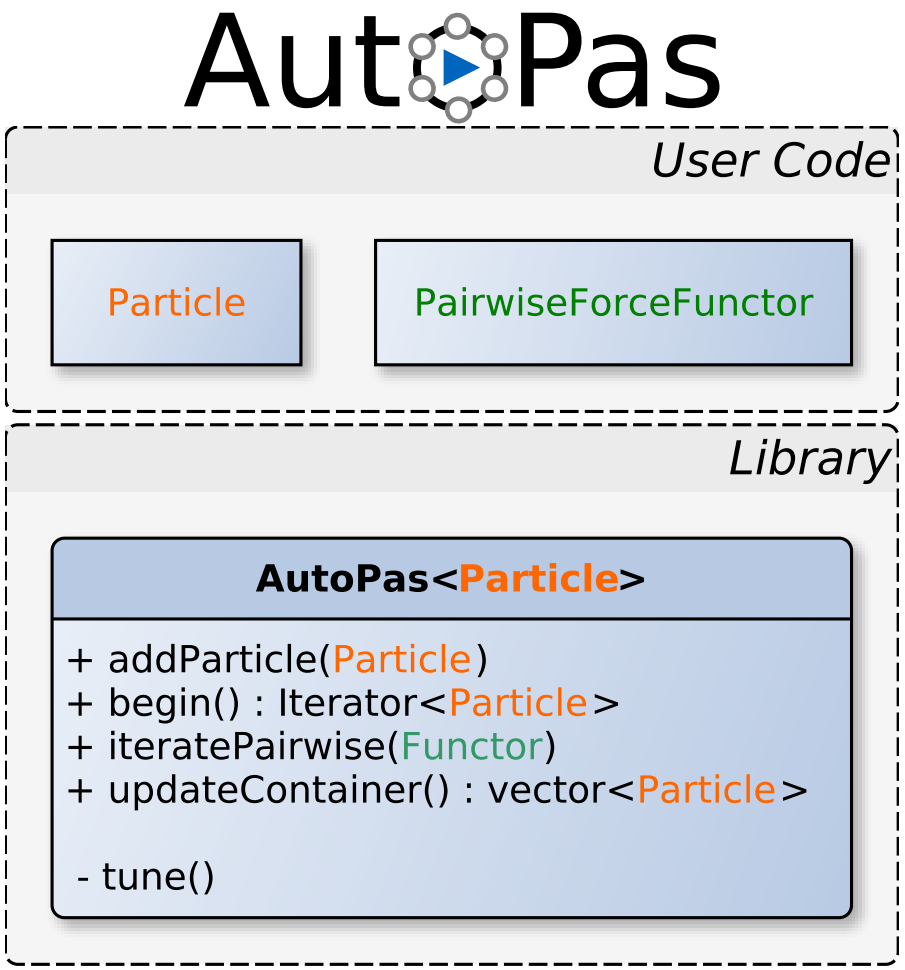


Introducing AutoPas

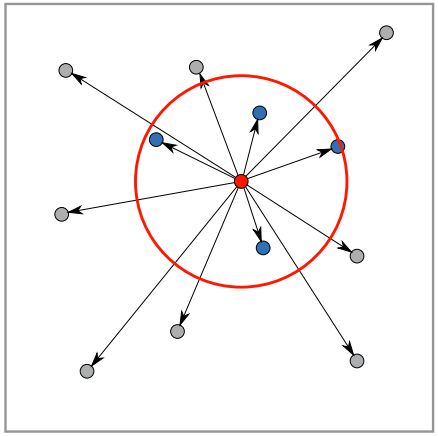
- Node-Level `C++17` library
- Black-box particle container
- Facade-like software pattern
- User defines:
  - Properties of particles
  - Force for pairwise interaction
- AutoPas provides
  - Containers, Traversals, Data Layouts, ...
  - Dynamic Tuning at run-time

⇒ General base for N-Body simulations

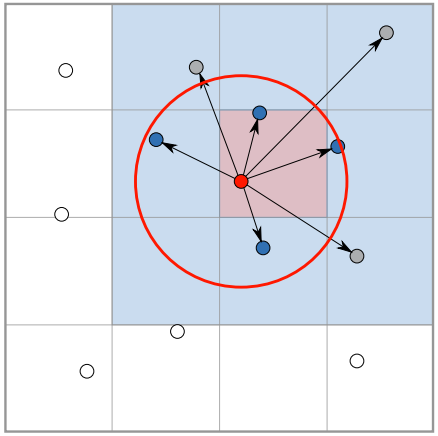
https://autopas.github.io/



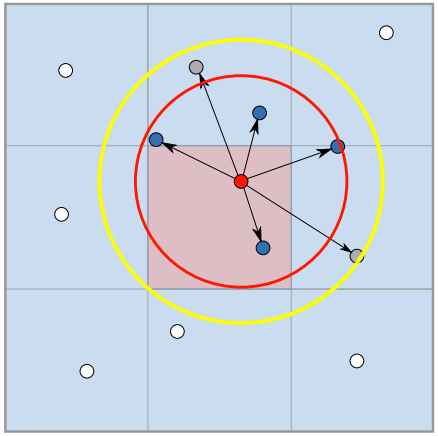
Algorithmic Choices



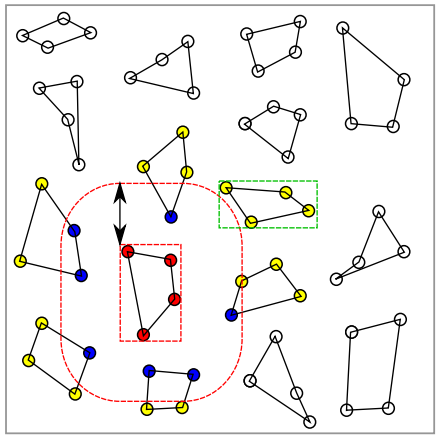
Direct Sum



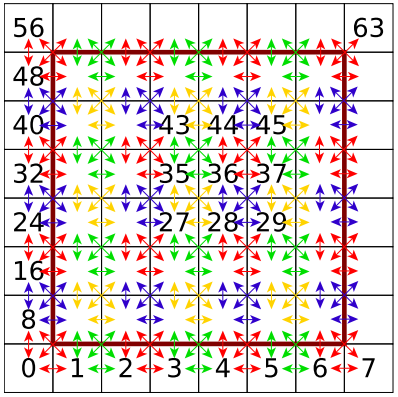
Linked Cells



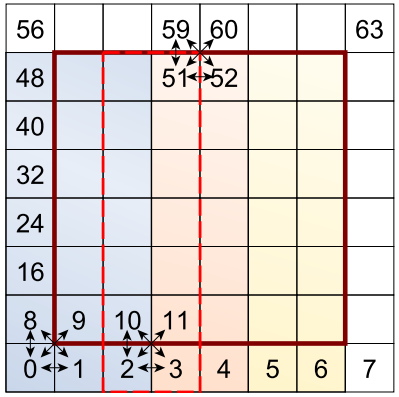
Verlet Lists



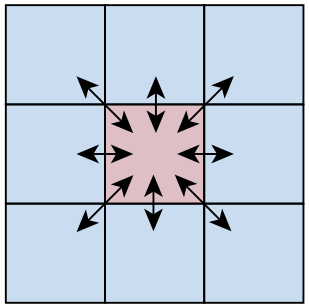
Verlet Cluster Lists



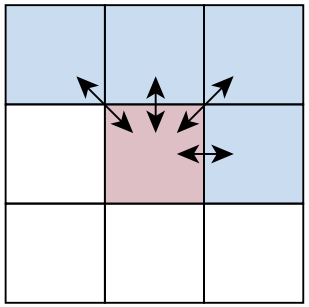
Coloring



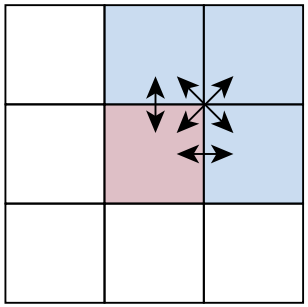
Slicing



C01



C18

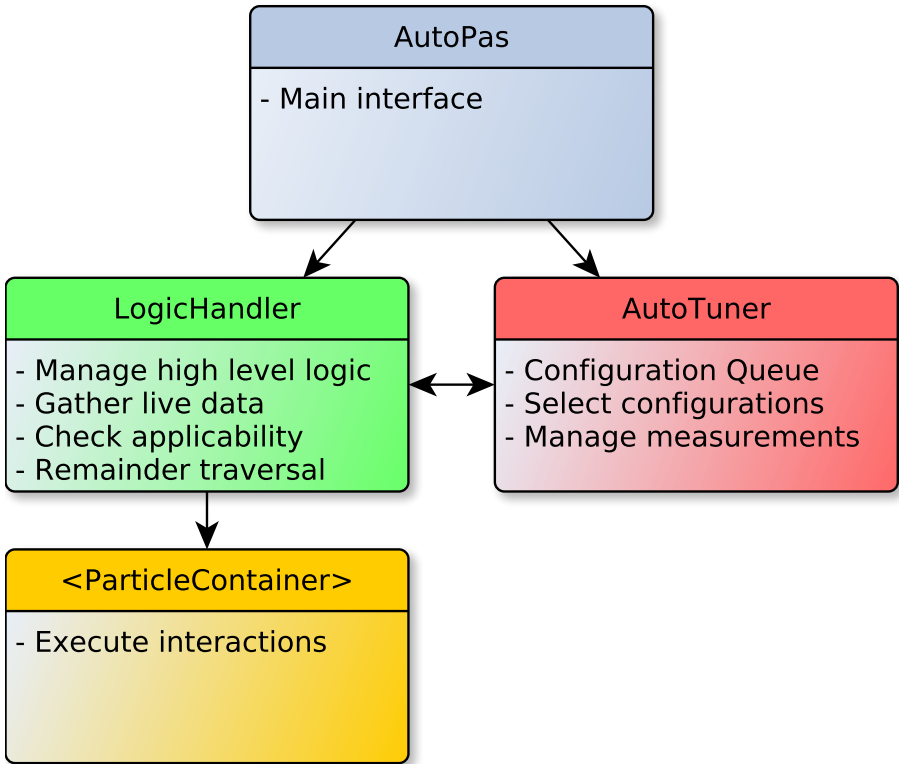


C08

Stencils

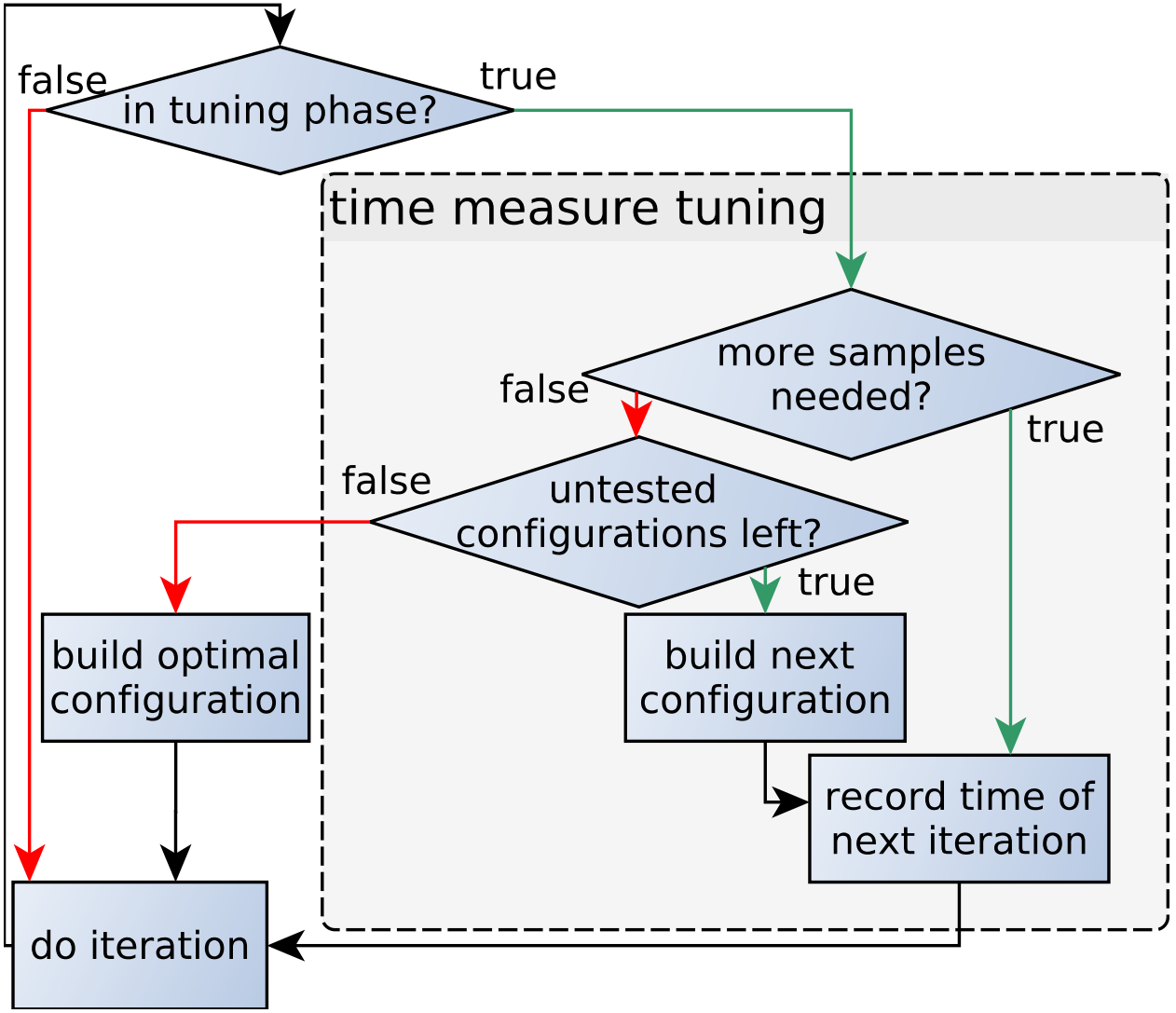
AutoPas Tuning Workflow

- Simple user interface
- Abstraction layer for non-critical common functionality
- Highly specialized data containers for particle interactions



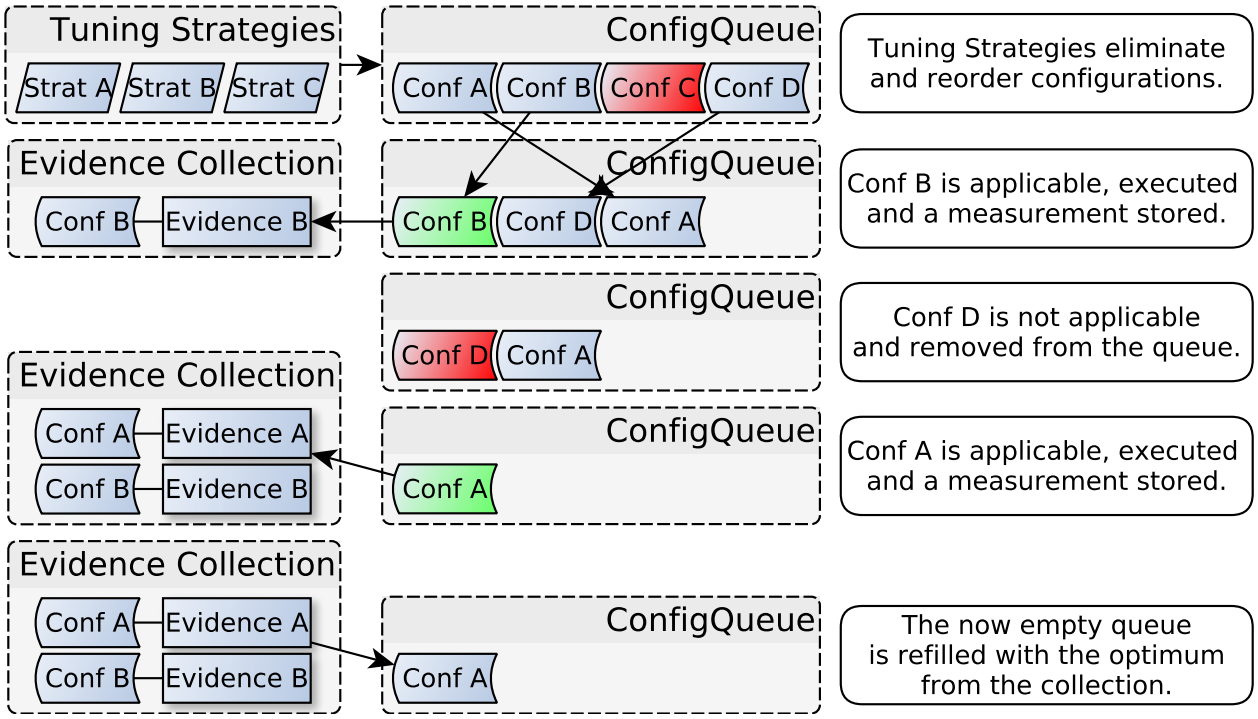
Tuning Cycle

- Common interfaces for containers, traversals, etc.
- Repeated periodically
- User can restrict search space

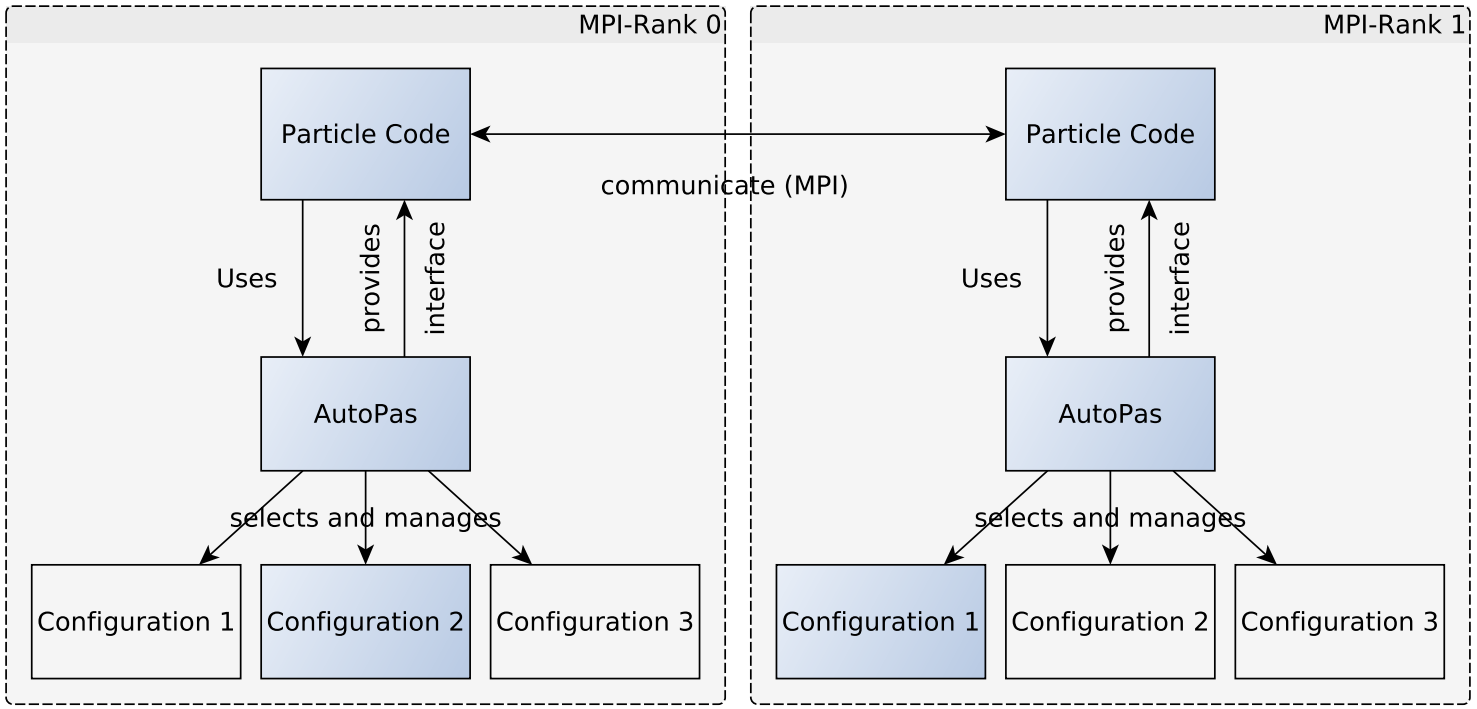


AutoPas Tuning Pipeline

- Modular design
- Easily extensible with new:
  - ⇒ Configurations
  - ⇒ Tuning strategies
  - ⇒ Tuning metrics

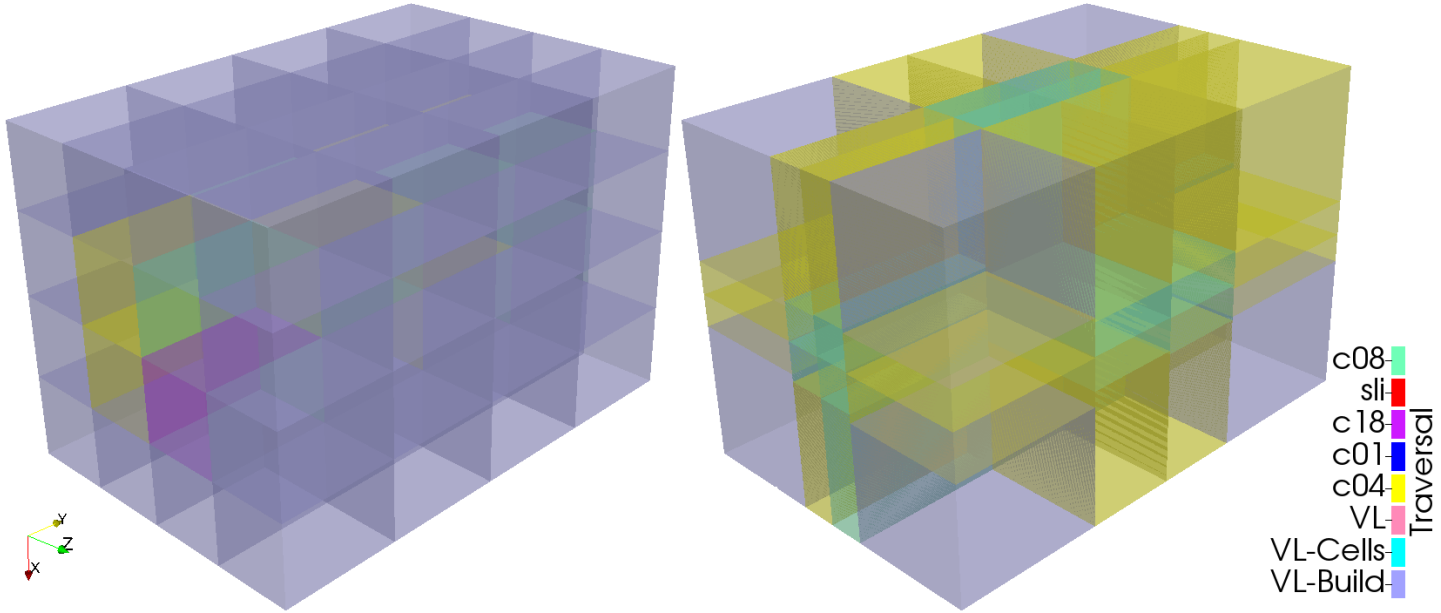


AutoPas in MPI parallel Software

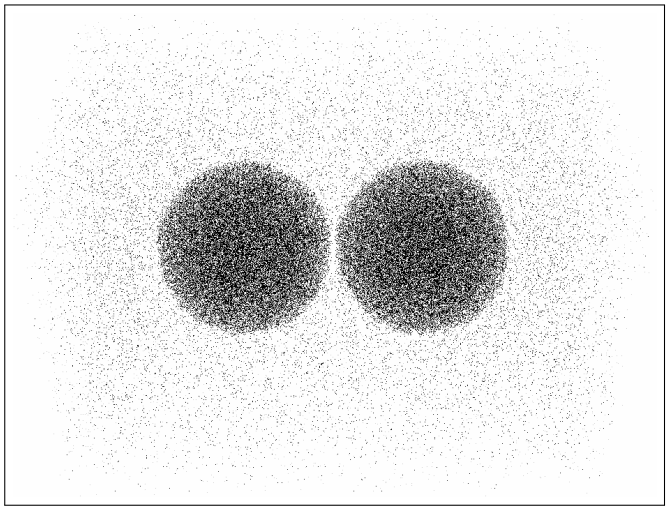


- One AutoPas instance per Rank.
- Independent tuning!
- Interfaces independent of algorithms.
- AutoPas acts as black-box container.

Speedups from Dynamic Tuning is ls1 mardyn

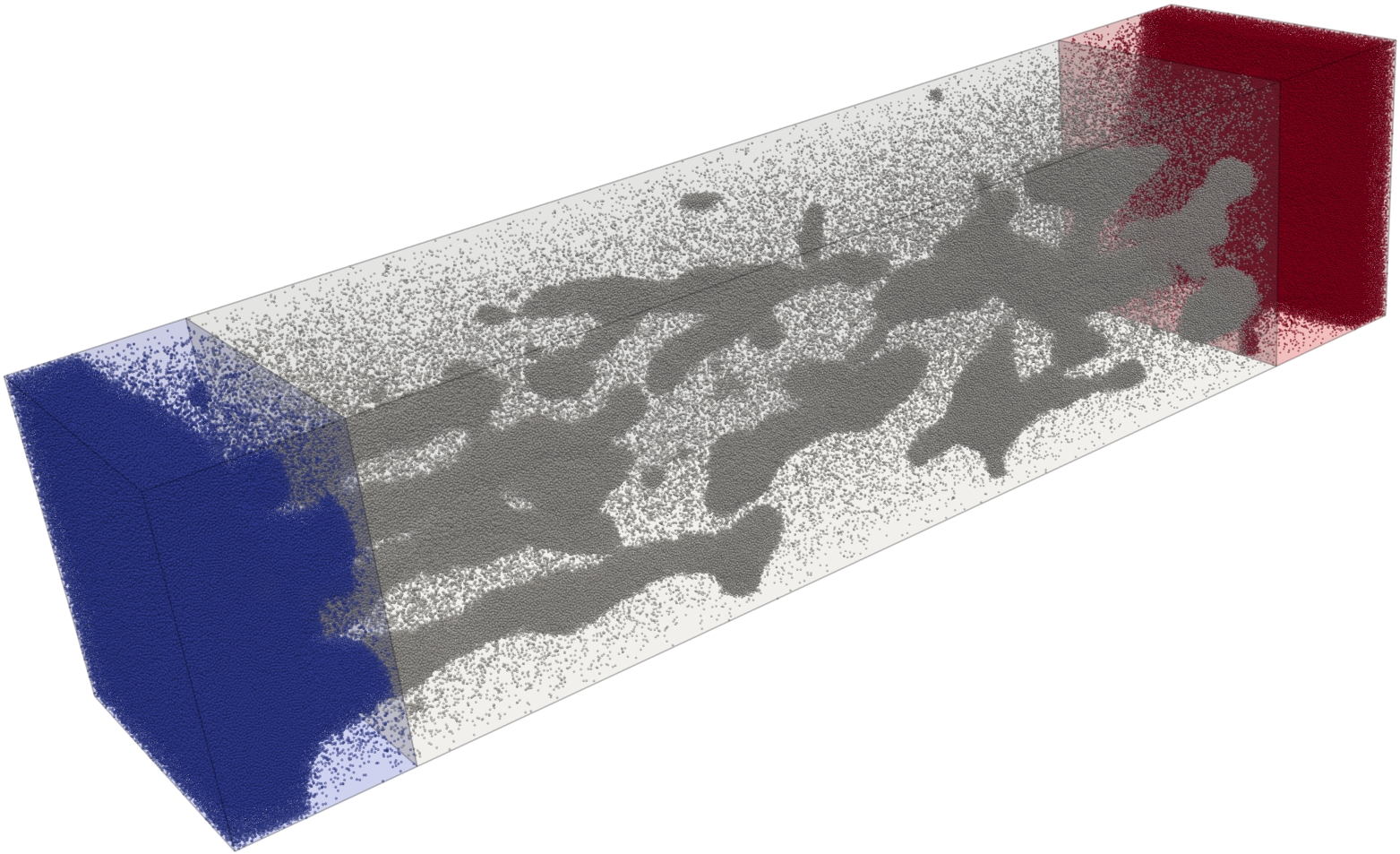


- 3M Particles over 64 Ranks
- Auto-Tuning + Diffusive Loadbalancing
- Speed-up about 50% vs vanilla ls1

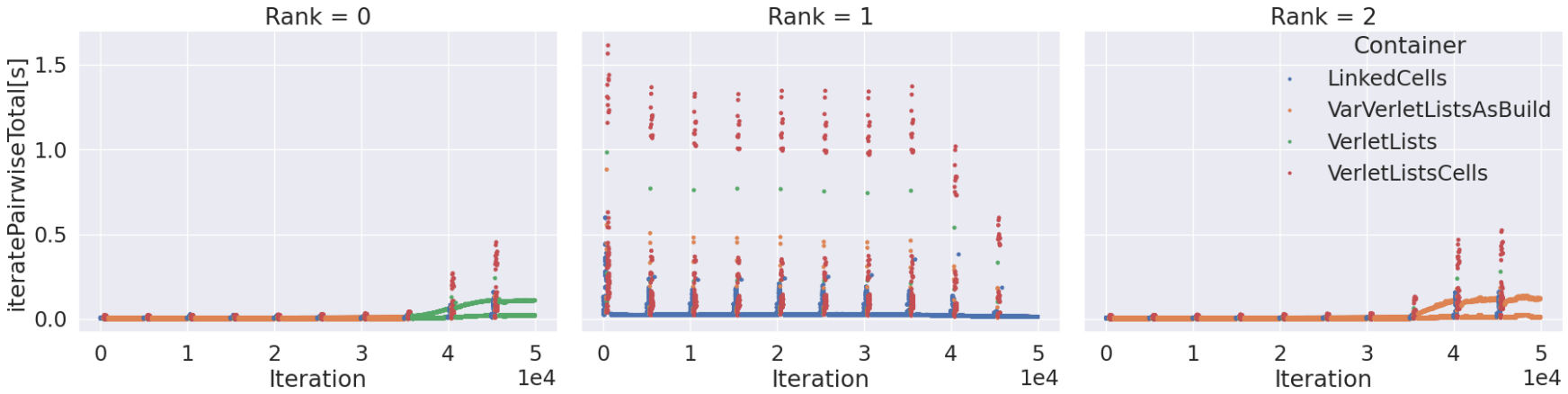


Seckler, S., Gratl, F., Heinen, M., Vrabec, J., Bungartz, H. J., & Neumann, P. (2021). AutoPas in ls1 mardyn: Massively parallel particle simulations with node-level auto-tuning. Journal of Computational Science, 50, 101296.

ls1 mardyn + AutoPas $\Rightarrow$ MPI + Tuning



Iteration Performance over Time - Full Search



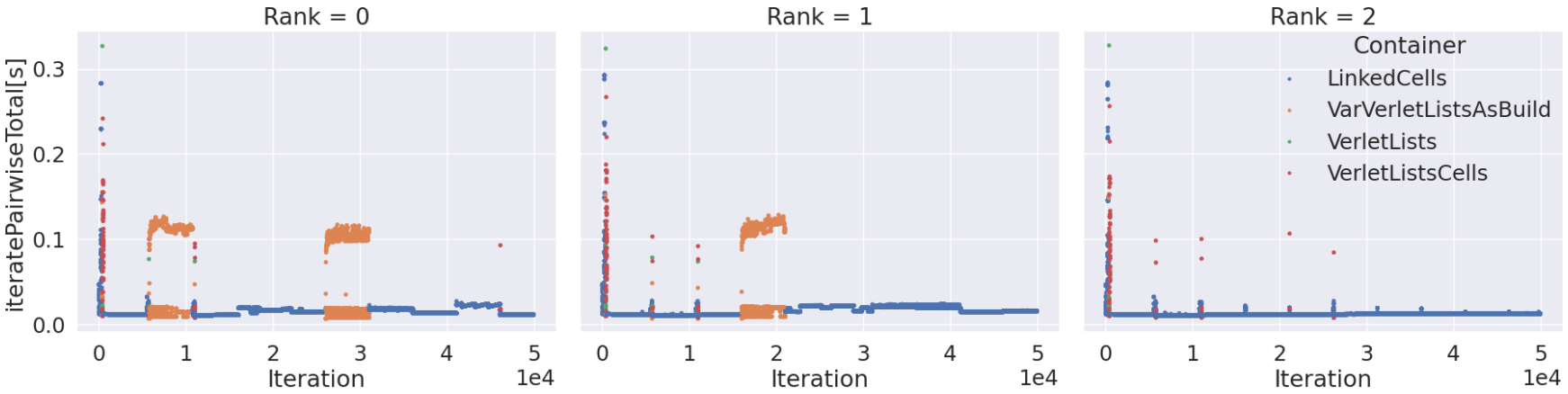
- Only first 50k iterations of 280k
- Exhaustive search through 92 configurations
- Tuning phases become extremely expensive
- Highest outliers are Verlet list rebuilds

Rule Based Tuning

```
if numParticles > 1000:
  [container="LinkedCells"] >= [container="DirectSum"];
endif
```

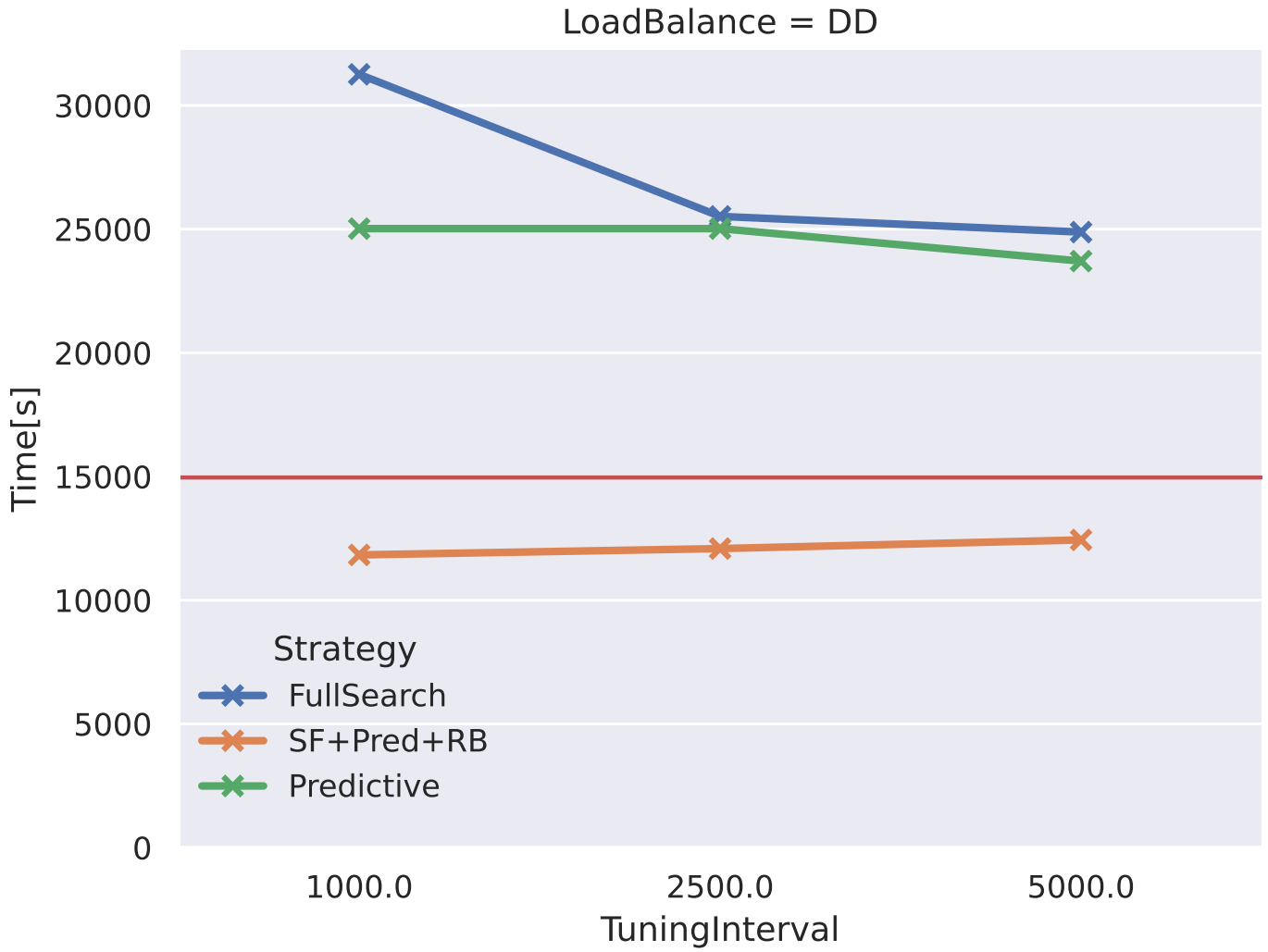
- Domain Specific Language
- Filters based on pattern matching
- Can make use of dynamic live data
- Implemented via virtual machine

Iteration Performance over Time - Smart



- Y-axis 5x shorter!
- Apply expert rules to avoid inefficient configurations from the start
- Apply learning to only test promising configurations

Combined Smart Tuning Outperforms Everything



Summary

- AutoPas enables automated algorithm selection on node level.
  - Independent tuning of all MPI ranks.
  - Optimal algorithm choices also for non-expert users.
- Black Box particle container facilitates development of new applications.
  - Adapts autonomously to changing requirements.
  - No software expert knowledge needed.
- Speed up automated algorithm selection by combination of knowledge and learning.