Load Balancing and Auto-Tuning for Heterogeneous Particle Systems using ls1 mardyn

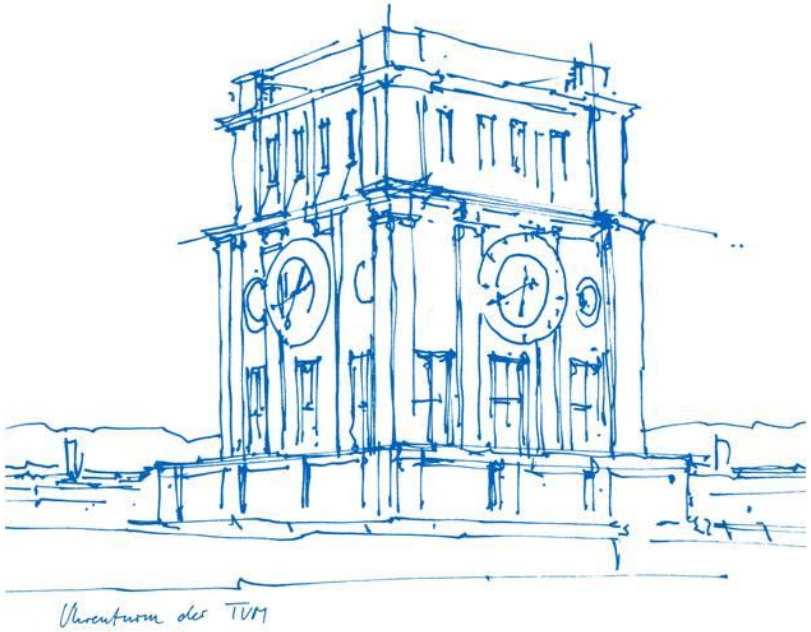
Steffen Seckler

Technical University of Munich

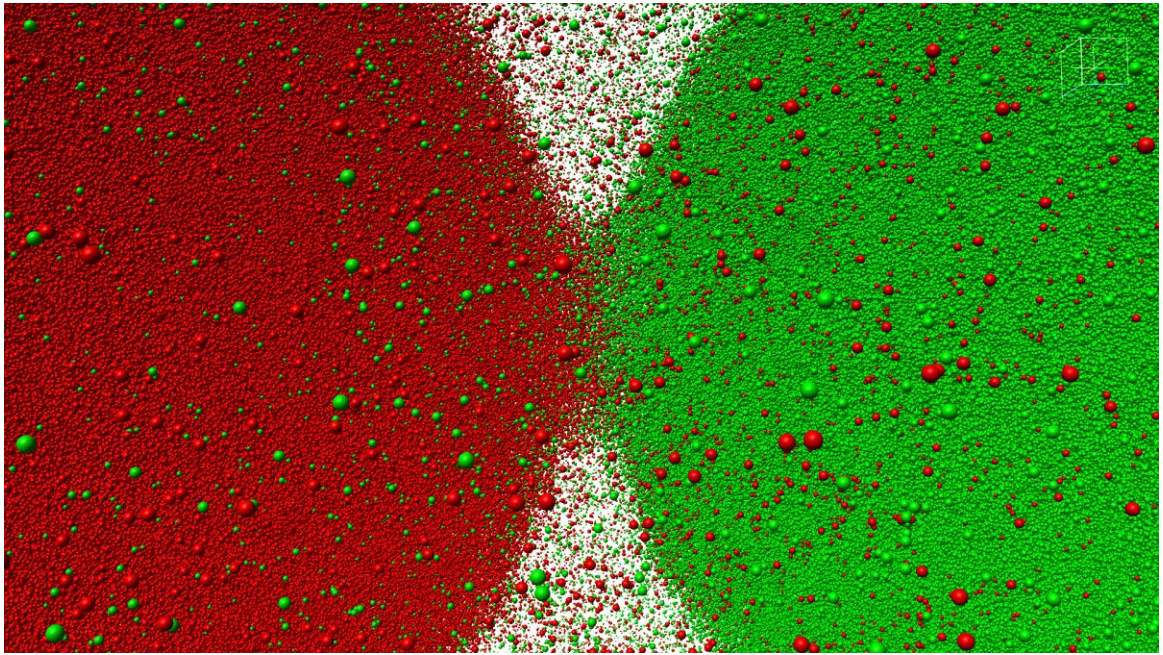
Department of Informatics

Chair of Scientific Computing in Computer Science

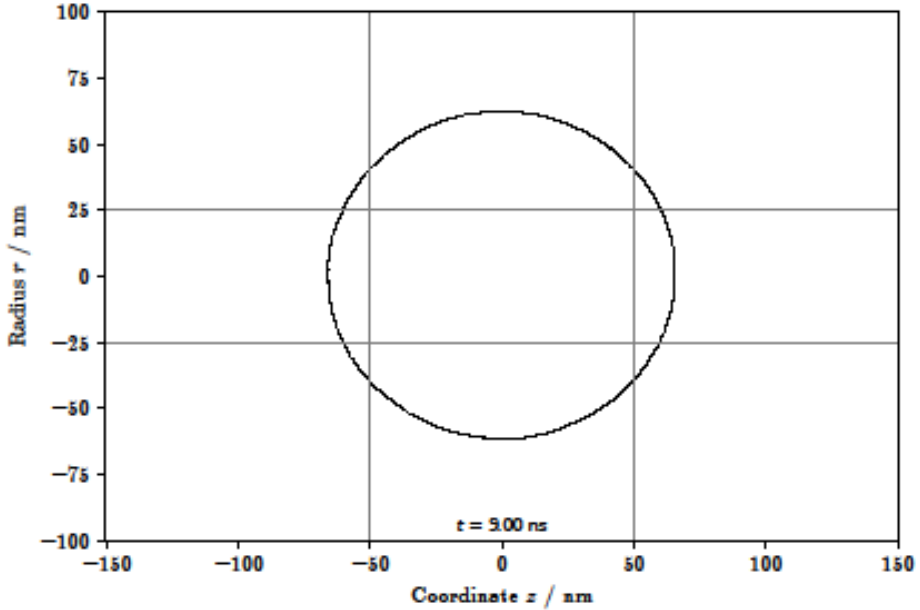
Stuttgart, 07.10.2019



Motivation – Droplet Coalescence



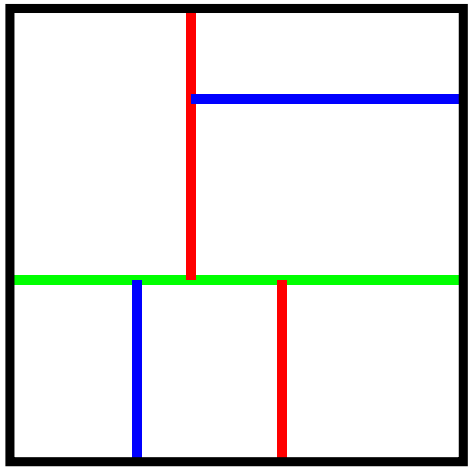
Motivation – Droplet Coalescence

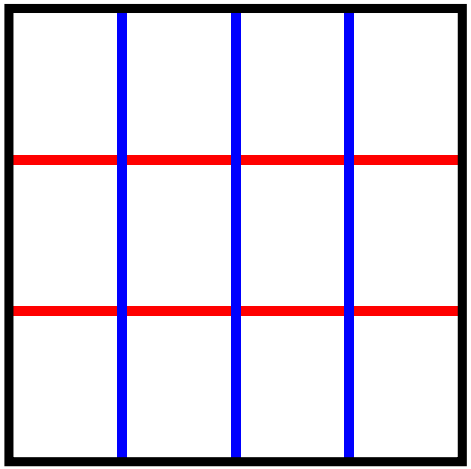


Load Balancing Techniques in ls1 mardyn

Observations:

- Large load imbalances for droplet coalescence
- ls1 mardyn can use k-d trees (kdd) to distribute the domain (cell-based)





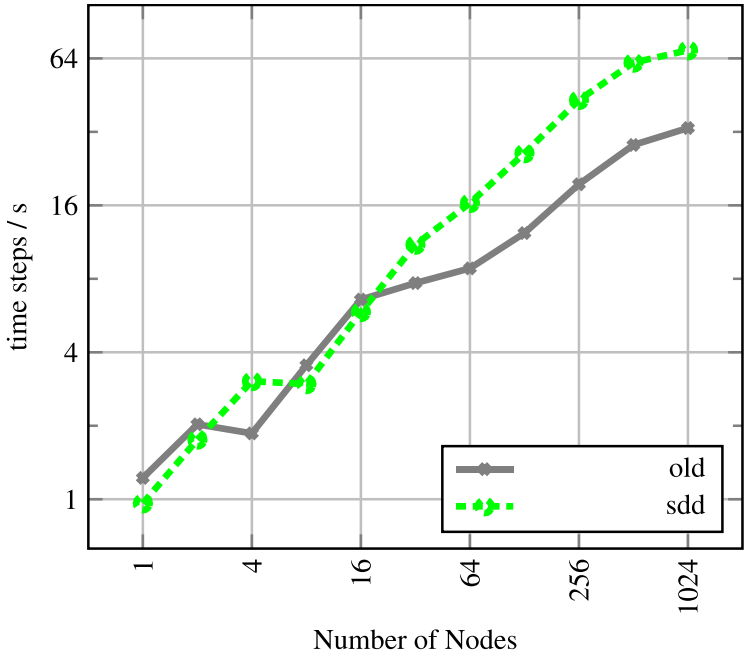
What people use:

- No load balancing: Cartesian grid (sdd)

Motivation – Droplet Coalescence

Observations:

- Bad scaling
- K-d trees worse than Cartesian domain decomposition

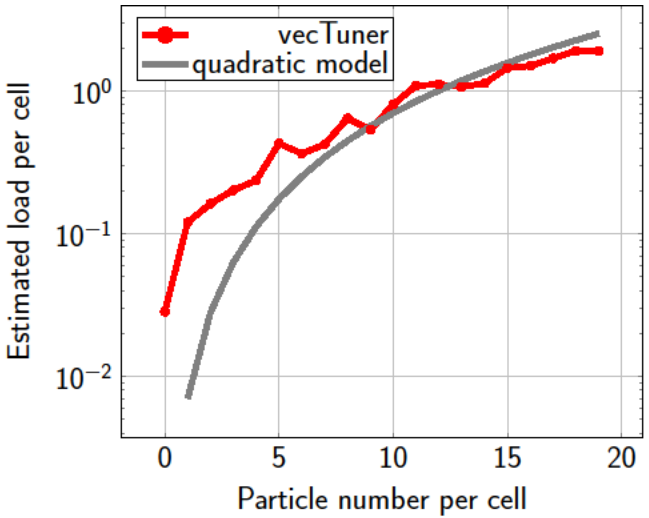


Load Estimation – Not only Load Balancing

Reasons:

- The load was distributed properly!
- BUT: The assumed load was not correct!

- Quadratic model: assume load per cell $L_c = N^2$, N … particle count
- Iteration over MANY empty cells also expensive!
- VecTuner (new): measure using single cells at startup (no caching effects, …)



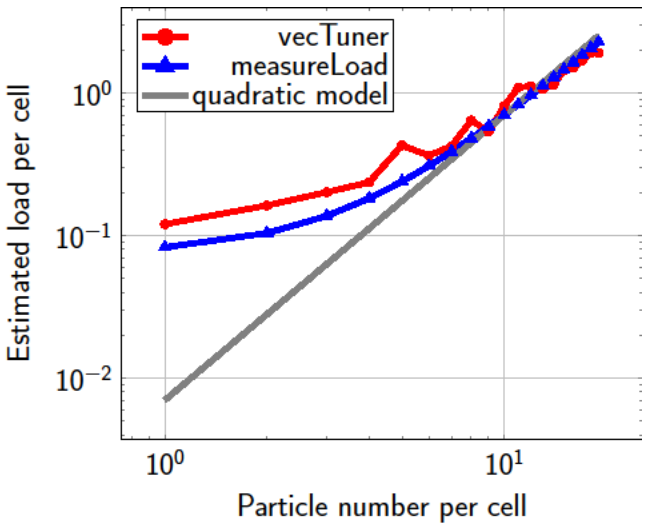
Load Estimation – Not only Load Balancing

Can we somehow measure the load for each cell appropriately (caching effects, …)?

$$T_i = \sum_{c \in C} n_{i,c} \cdot C_c$$

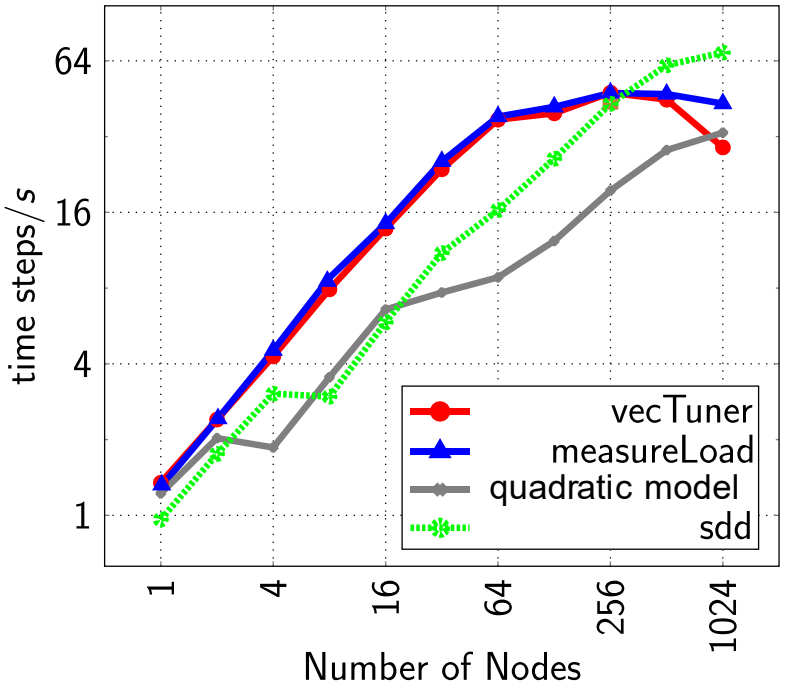
T_i, … measured Time on process i
C_c, … cost for each cell with type c
$n_{i,c}$, … number of cells with type c on process i

With $C_c = C(n) = a \cdot n^2 + b \cdot n + c, \quad a, b, c > 0$

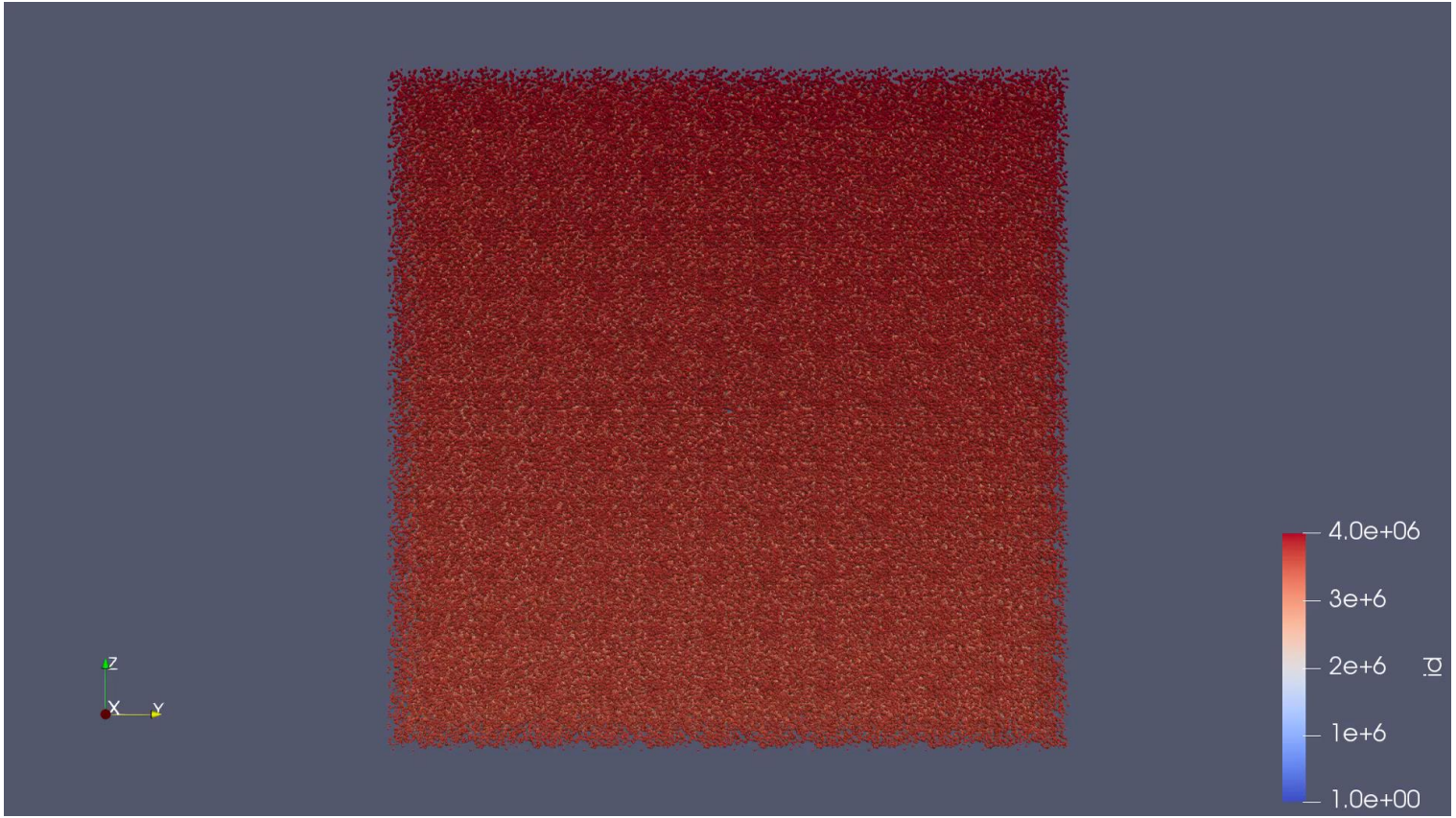


Load Estimation – Not only Load Balancing

- vecTuner and measureLoad were outperforming quadratic model
- Sdd still faster for small scenarios
- Now limited by load balancing, not load estimation



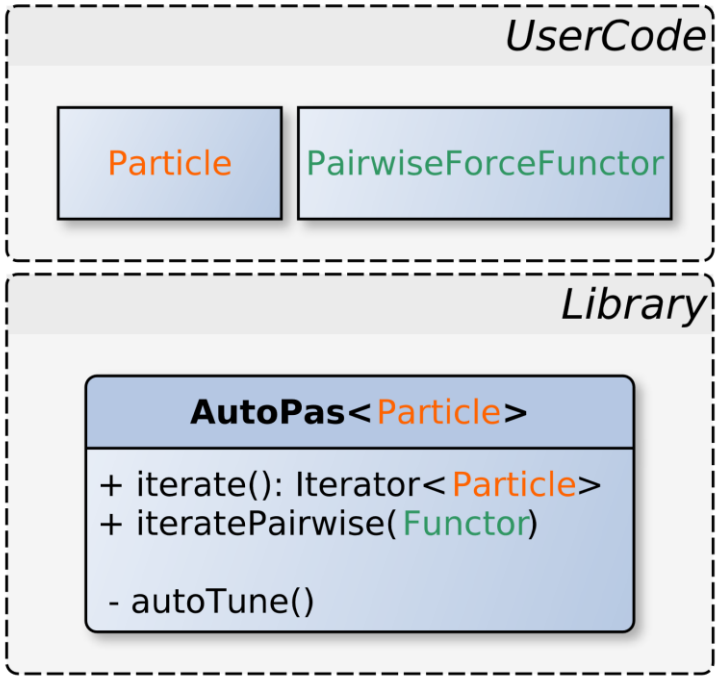
AutoPas



AutoPas

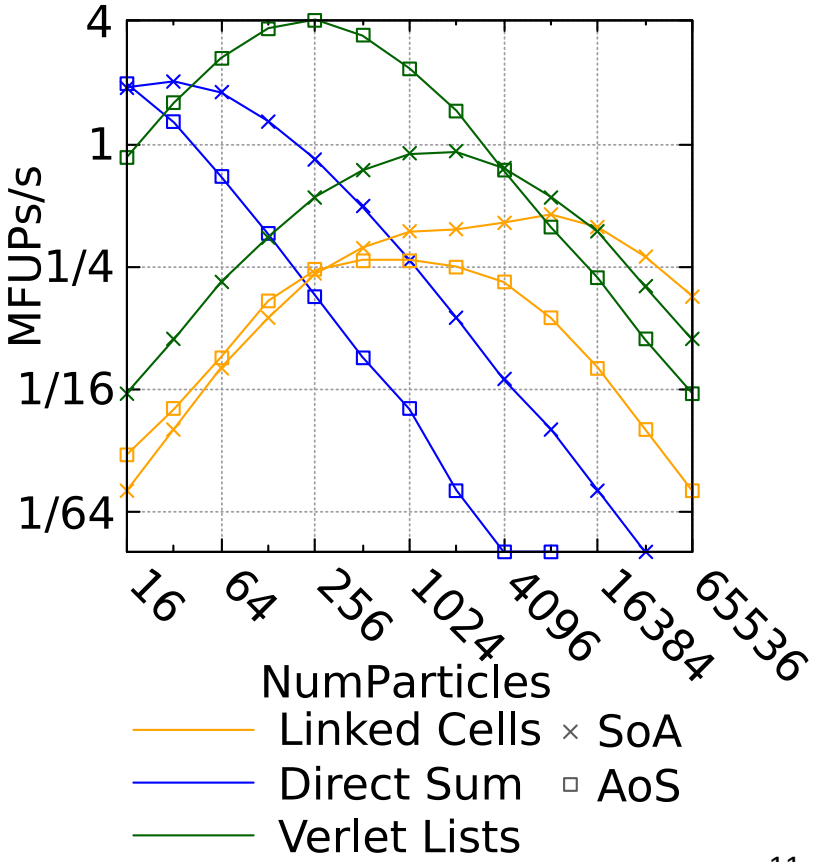
- Node-Level C++ library
- User defines:
  - Properties of particles
  - Force for pairwise interaction
- AutoPas provides:
  - Containers, Traversals, Data Layouts, …
  - Dynamic Tuning at run-time
- Black Box container

➔ General base for N-Body simulations



AutoPas

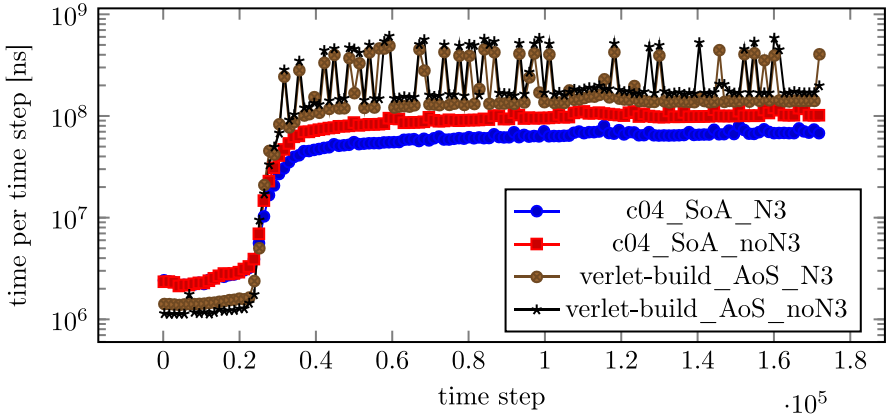
- Every container has advantages
- Linked Cells benefits most from SoA
- Currently implemented:
  - 7 different Containers
  - 17 Traversals
  - SoA, AoS (exp. support: Kokkos, Cuda)
- Currently auto-tuning mainly using brute-force search
  - We are working on more advanced auto-tuning techniques (ML, Bayesian Opt.)



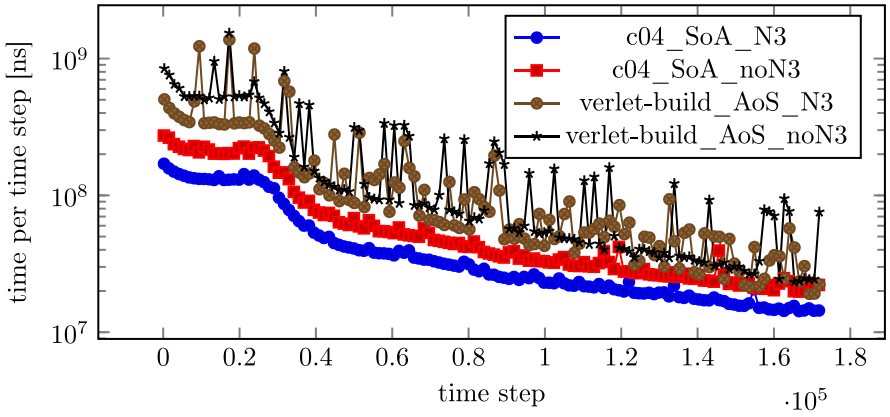
AutoPas in ls1 mardyn

- AutoPas is fully integrated in ls1 mardyn
- Using a verlet-like interface
  - Neighborlists and container only updated every N time steps
  - Particles are only moved between different MPI ranks every N time steps

- Right: without LB, exploding liquid



(a) Rank 0

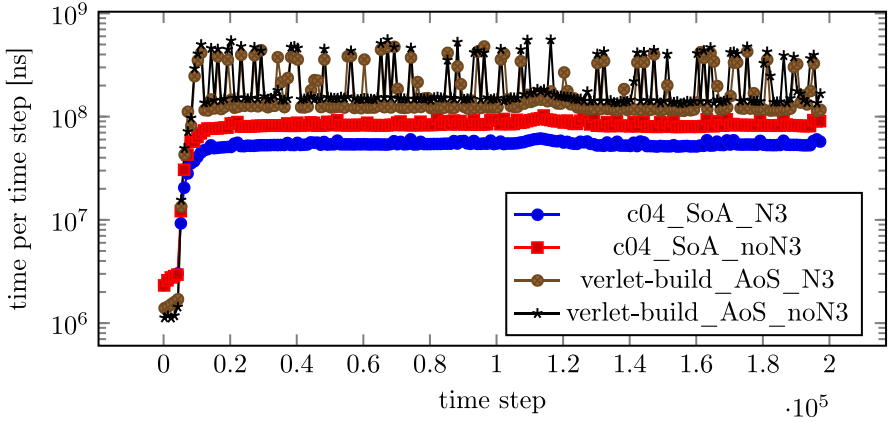


(b) Rank 1

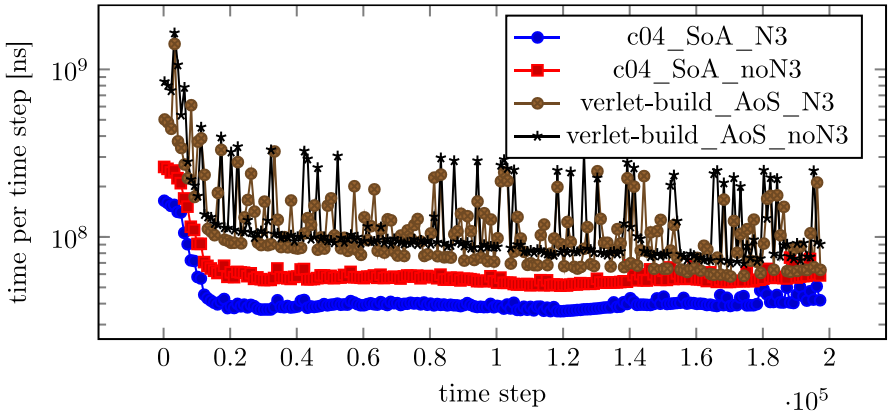
AutoPas in ls1 mardyn

- Support for LB using ALL load balancing library developed at JSC
  - Uses staggered grids
  - Diffusive LB
  - Load = time per process

- Right: with LB, exploding liquid



(a) Rank 0



(b) Rank 1

Thank you!

- AutoPas is developed
  - on GitHub
    - https://github.com/AutoPas/AutoPas/
  - in the context of the TaLPas (BmBF) project
    - www.talpas.de

- ls1 mardyn is developed
  - on GitHub
    - https://github.com/ls1mardyn/ls1-mardyn
  - See: www.ls1-mardyn.de

Questions?

Backup Slides