AutoPas: A Library for N-Body Simulations Enabling Optimal Node-Level Performance Through Auto-Tuning

Molecular Dynamics

Applications:
- Chemical Engineering: cavitation, surface tension, droplet coalescence, etc.

Goals:
- High node-level performance in arbitrary scenarios.
- Minimize time to solution.

Main Challenges:
- Drastic impact of simulation variables on time to solution.
- Performance depends on many variables that can change during runtime.

AutoPas Library

AutoPas Vision:
- Base to build full N-Body simulations on top of.
- Manages node-level performance internally via Auto-Tuning.
- Modular C++ template design able to dynamically select optimal SIMD, OpenMP, data structures, etc. at runtime.

Auto-Tuning:
- Instead of the user, the code should find the optimal algorithms.
- Too many combinations possible to test all.
- Too many combinations possible to test all.⇒ Outlook:
  - Performance Modeling, Machine Learning, Bayesian Statistics.
  - Reevaluate combination during runtime and adapt accordingly.
  - Provide flexibility through "strategy" software pattern.

First Results

An early version of AutoPas was coupled with a teaching code by students. This was done to test the performance as well as verifying usability.

Scenario:
- Body-centered cubic lattice with 2M particles.
- Strong scaling on SuperMUC Phase 1.
- 54 Linked Cells per dimension (excl. halo) ≈ 13 particles per cell.

Observations:
- Performance of algorithms varies significantly.
- For every configuration at least one algorithm outperforms the original code.

→ Target for Auto-Tuning!
- AutoPas can offer near optimal scaling.

Motivation

Computational Overhead
- Several data structures, algorithms, parallelizations, etc. available.
- All have scenario dependent strengths.
- Choosing optimal combination not always straightforward.

AutoPas Library

User Code

Particle pairwiseForceFunctor

AutoPas

IteratePairwise()

ContainerSelector.getContainer()

TraversalSelector.getTraversal()

DataLayoutSelector.getLayout()

AutoTuner.iteratePairwise()"=

Traversal.traverseCellPairs()

Container.iteratePairwise()

StdDev 25

Scenario:
- Homogeneity scaling.
- 8M particles.
- Gaussian distribution.
- 100 cells per dim.
- 8 Threads

Observations:
- c08 (8-way coloring)
- Good load balancing.
- sli (1D partitioning)
- Minimal overhead.

References