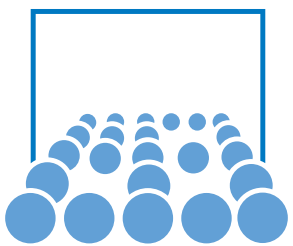
PRACE Scientific Conference 2013

591 TFLOPS Multi-TRILLION Particles Simulation on SuperMUC

W. Eckhardt[TUM], A. Heinecke[TUM],
R. Bader[LRZ], M. Brehm[LRZ], N. Hammer[LRZ], H. Huber[LRZ], H.-G. Kleinhenz[LRZ],
J. Vrabec[ThEt], H. Hasse[LTD], M. Horsch[LTD],
M. Bernreuther[HLRS], C. W. Glass[HLRS], C. Niethammer[HLRS],
A. Bode[TUM,LRZ], H.-J. Bungartz[TUM,LRZ]

June 16, 2013



Outline

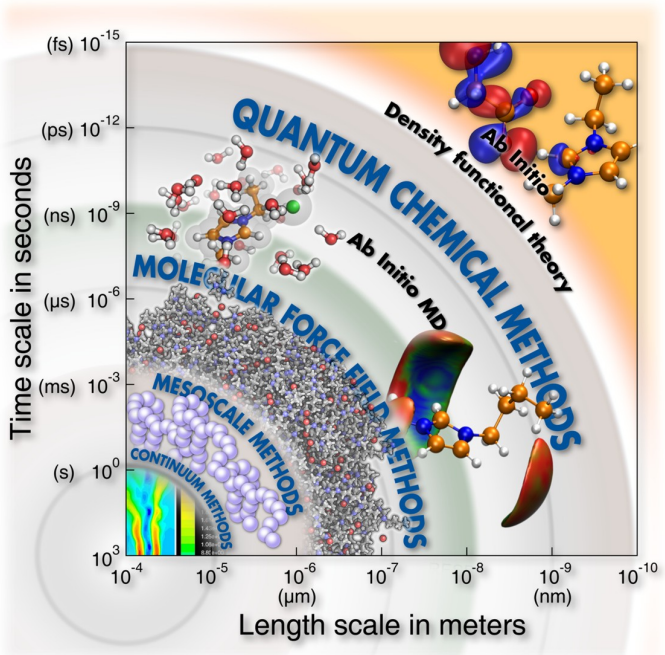
Applications of Interest

Applications in Process Engineering:

- Study of nano-scale flows
- Study of interfacial phenomena
- Size dependence of interfacial quantities such as the surface tension
- Behavior of droplets, interactions between droplets, nucleation
- Mixing behavior of fluids

Several simulations methods available:

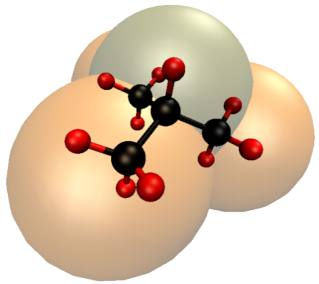
- Simulation methods vary in their level of detail
- The more detail, the more predictive power
- Force field methods are favorable with respect to scaling and materials behavior



Molecular Dynamics Simulations

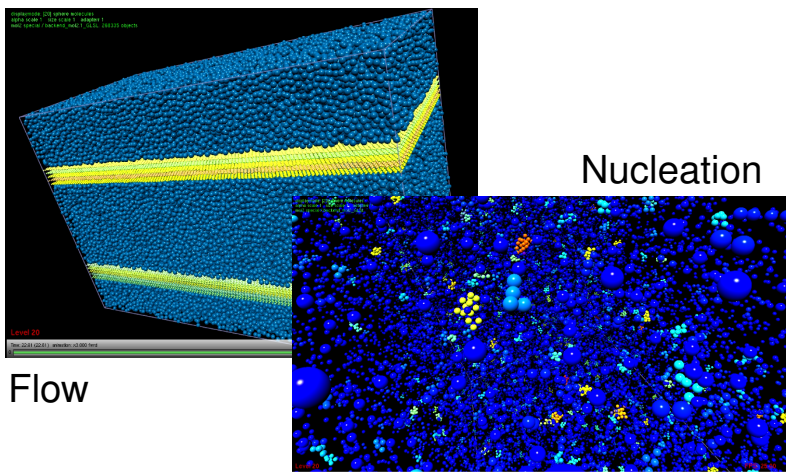
Force field based methods:

- Classical models for molecular interactions (parameters have a physical interpretation)
- Contain all thermodynamic properties (static, dynamic, surface properties)
- Straightforwardly applicable to mixtures
- Excellent predictive power and technical accuracy
- Directly applicable for studies of fluids



Massively parallel molecular dynamics code *ls1 mardyn*:

- Arbitrary mixtures of rigid molecules
- “Large” systems, “long” time scales
- Written in object-oriented C++, parallelized with MPI
- Lennard-Jones potential, electrostatic interactions



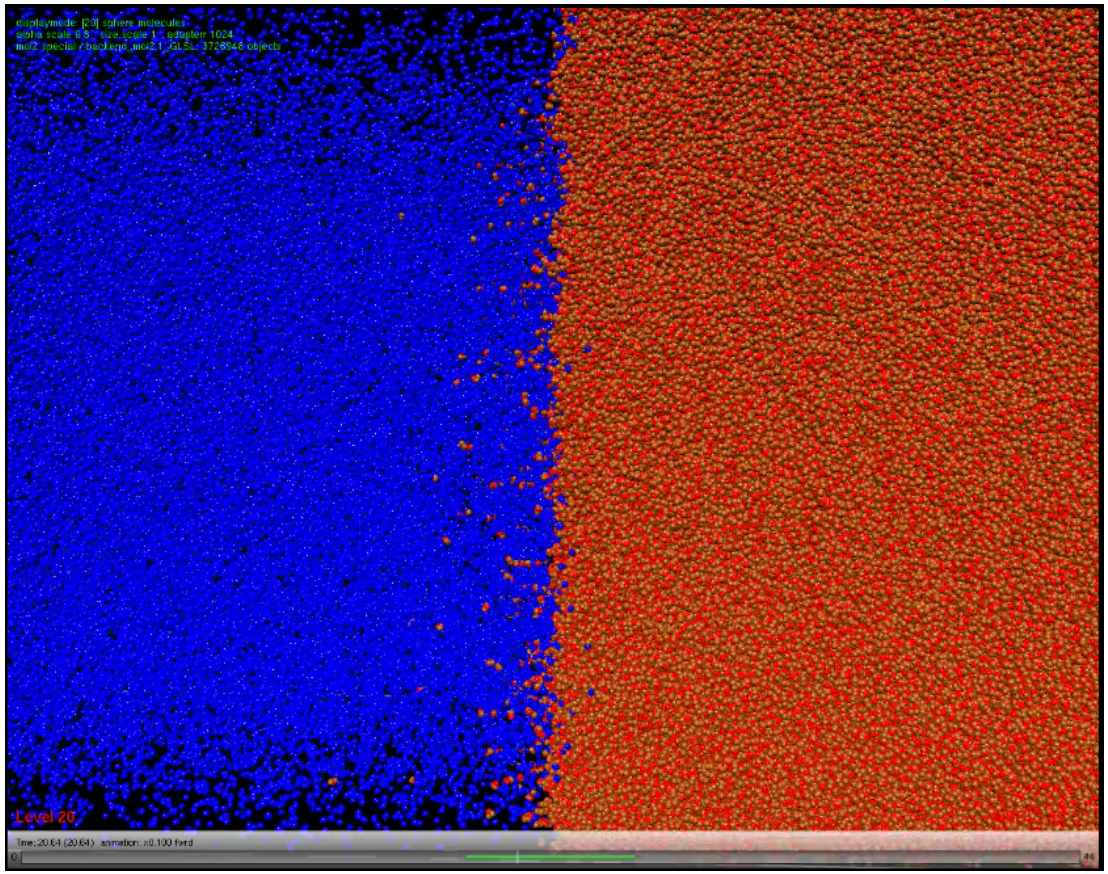
Flow

Nucleation

Example: Diffusion

gaseous (nitrogen) + liquid phase (acetone): $T = 350$ K, $p = 3$ MPa, $N = 1,000,000$

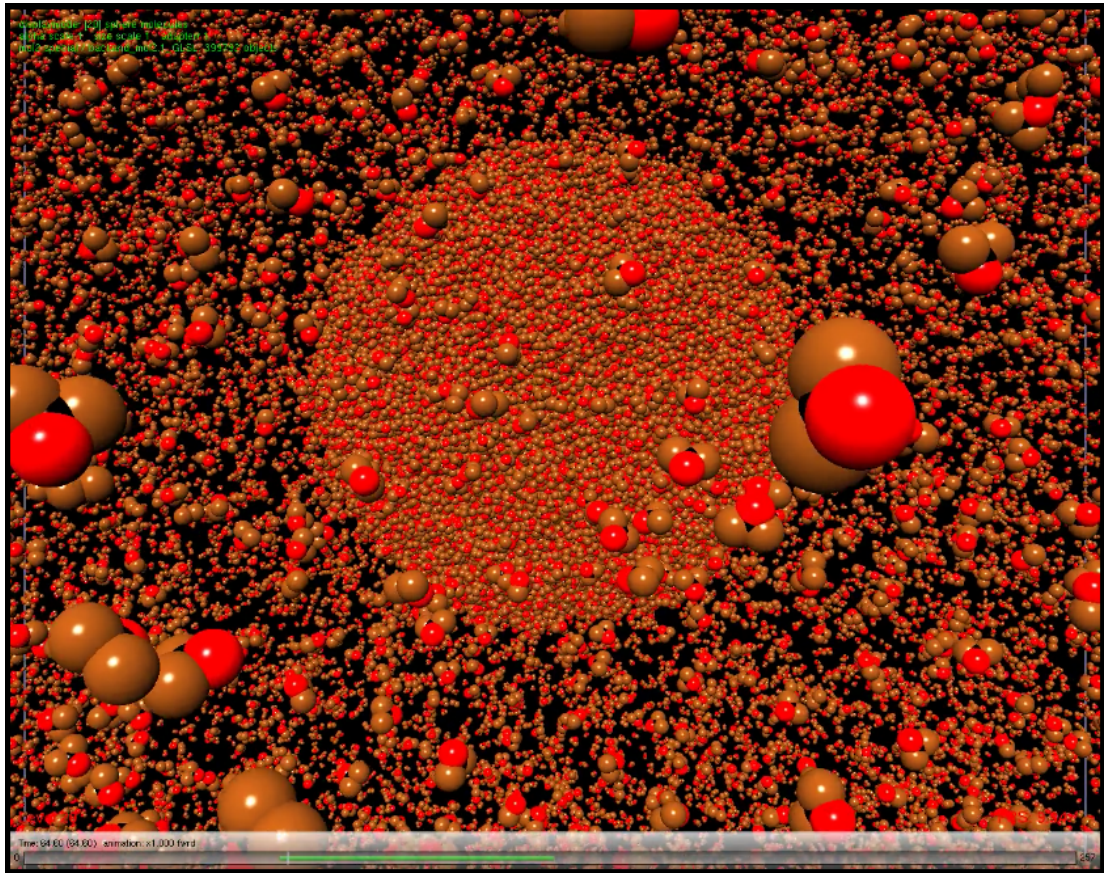
Mutual diffusion



Example: Vapor-liquid equilibrium

Drop of acetone in its vapor phase: $T = 350$ K, $N = 1,000,000$

Calculation of properties of the curved surface, e.g. surface tension



Example: Collisions of Nanoscale Droplets

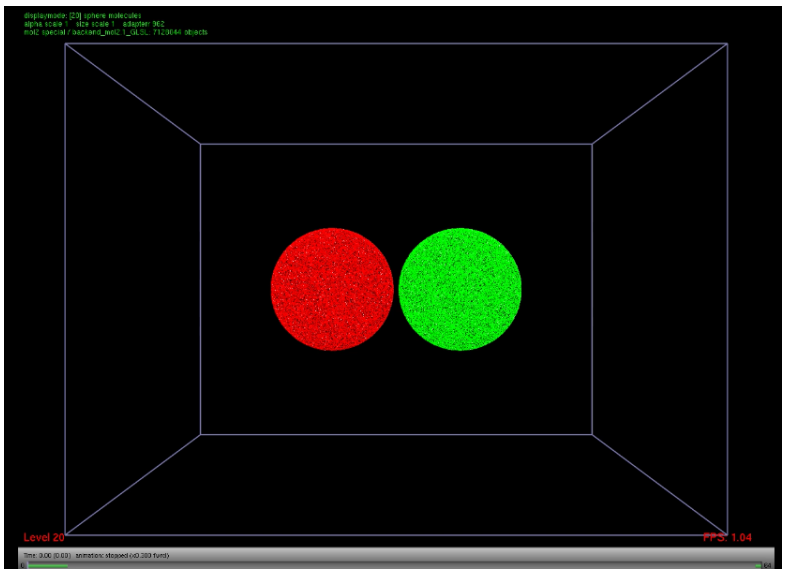
Substances modelled by 1 Lennard-Jones site, truncated and shifted:

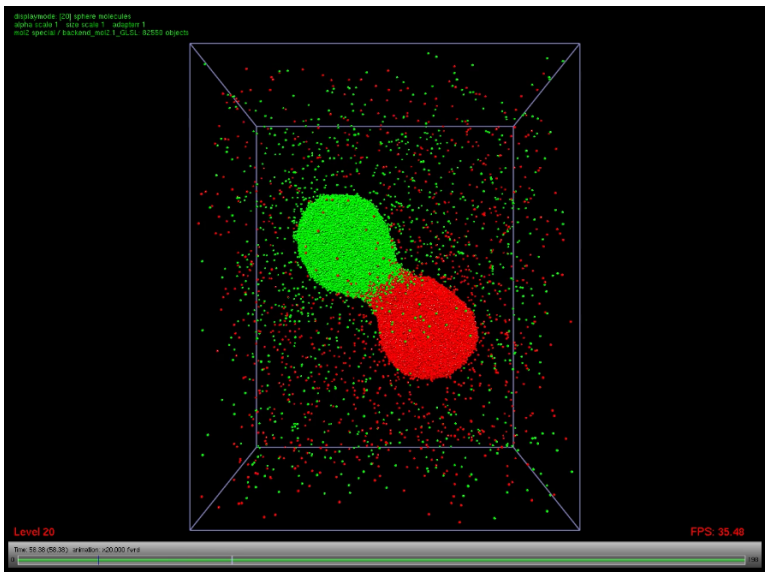
- Ar / Kr / Xe
- Methane

and interaction of two droplets of different kind, e.g. Argon $\Longleftrightarrow$ Methane

Collision of droplets in vacuum:

- Create equilibrated initial configuration seperately
- Place droplets in vacuum
- Set initial velocities

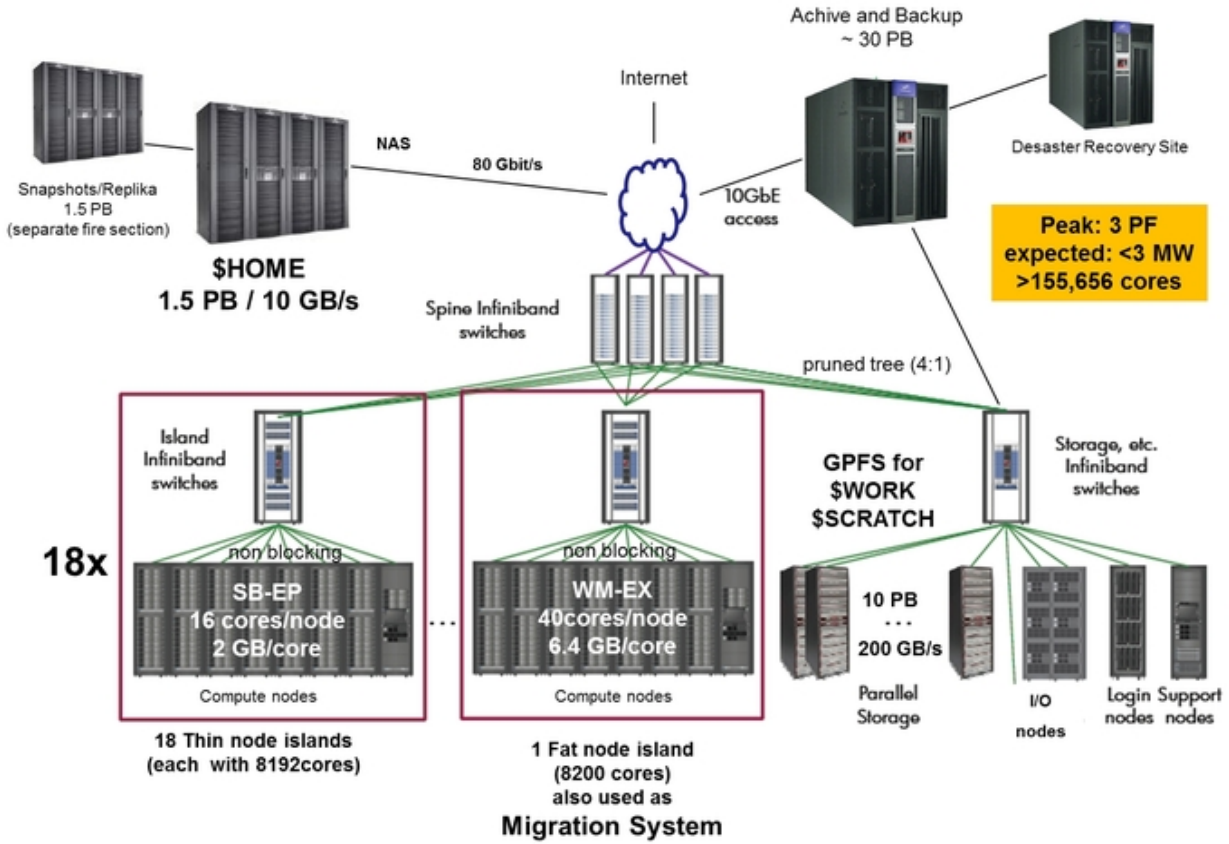




Outline

SuperMUC - System Topology

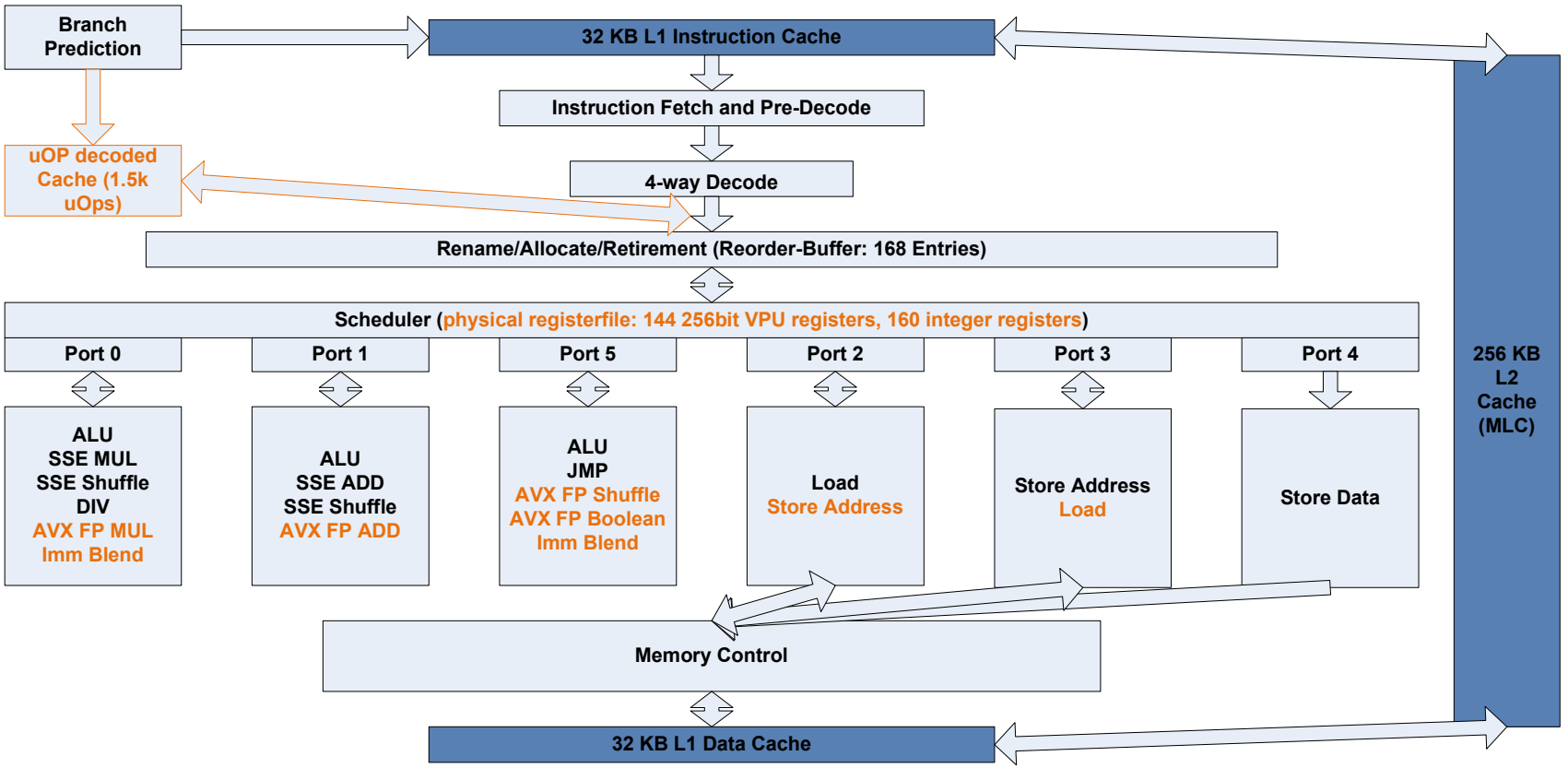
- IBM System x iDataPlex, $>$ 155.000 cores in total (Sandy Bridge-EP + Westmere)
- Theoretical double precision peak performance of 3 PFLOPS
- Novel warm water cooling



http://www.lrz.de/services/compute/supermuc/systemdescription/

Intel Sandy Bridge EP Processor Architecture

- μOp decoded cache; optimal for small kernels
- Features Advanced Vector Extensions: AVX128 and AVX256
- Hyperthreading technology to increase core utilization



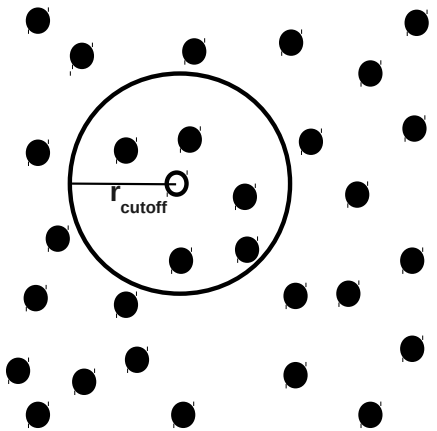
Outline

Molecular Dynamics Simulation

Discrete particles i and j with position x and velocity $\dot{x}$ interact via potential $U(r_{ij})$:

- Truncated and shifted Lennard-Jones-12-6 potential $U_{LJ}\left(r_{ij}\right)$ with potential parameters ϵ and σ:

$$U_{LJ}\left(r_{ij}\right) = 4\epsilon \cdot \left(\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right).$$



- Total effective force on a molecule i:

$$F_i = \sum_{j \in particles, j \neq i} F(r_{ij}).$$

- Perform time integration: $F = m \cdot \ddot{x}$
- Calculate statistical data:

Potential energy:

$$U_{pot} = \sum_i \sum_{j \neq i} U(r_{ij}).$$

Virial pressure:

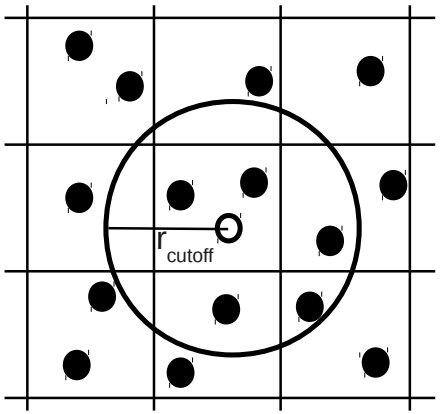
$$p = \frac{1}{2} \sum_i \sum_{j \neq i} r_{ij} \cdot F_{ij}.$$

Linked-Cell Algorithm

The Lennard-Jones potential is evaluated explicitely up to a **cutoff radius** r_c
$\Rightarrow$ Consider only local neighbors of a molecule

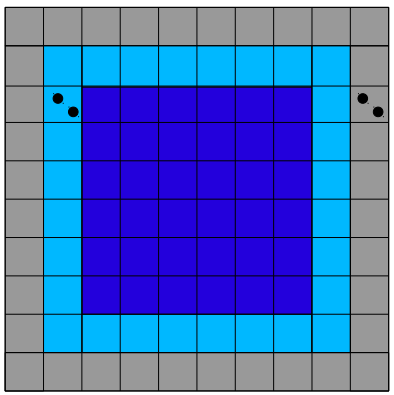
Linked-Cells-Algorithm:

- Subdivide domain in cells of edge-length $l = r_c$
- Neighboring molecules are contained in same or adjacent cells:
  $\Rightarrow$ Linear runtime of neighbor search
- Inherently cache-friendly

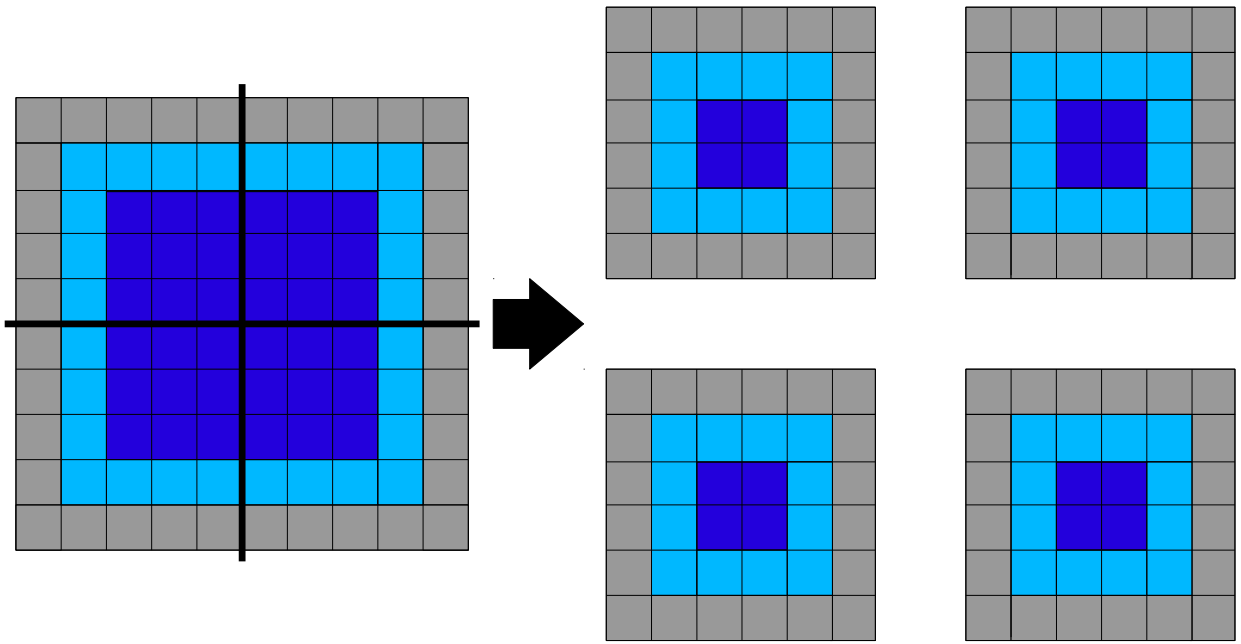


Linked-Cells-Algorithm with periodic boundary conditions:

Extend Linked-Cells structure by one cell layer to replicate particles from opposite boundary



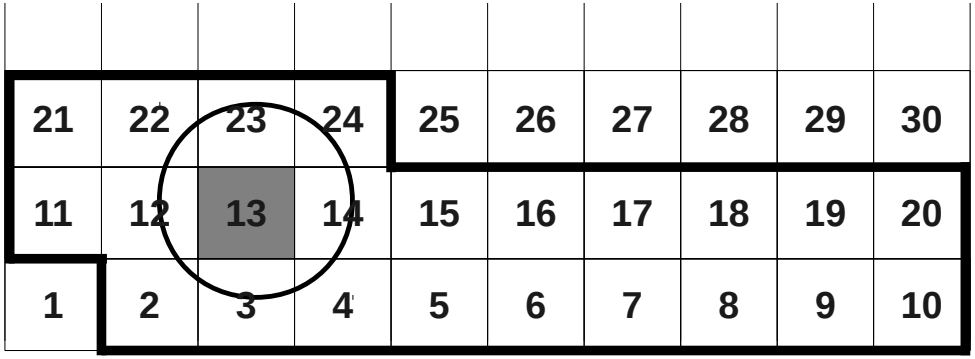
Parallelization of the Linked-Cell Algorithm



- Communication along spatial dimensions
  (6 instead of 26 communication partners)
- Non-blocking, overlapping MPI Send/Receive:
  - Start receive operation for both neighbors along spatial dimension
  - Start send operation
  - Wait until all operations finished
- AllReduce operations for global values / statistics

Outline

Memory-efficient Implementation of Linked-Cells



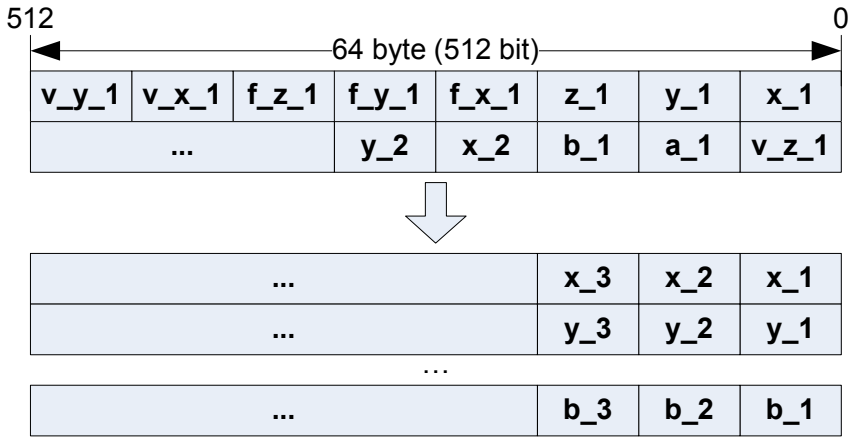
- Cells are traversed by sliding window in FIFO-order
- For each particle outside the sliding window, store only

Poxition $\vec{x}$	3 * 4 B = 12 B
Velocity $\vec{v}$	3 * 4 B = 12 B
Identifier id	8 B
Total	32 B

- For each molecule inside the sliding window, store additionally the force
  $\Rightarrow$ minor overhead in terms of memory
- Perform time integration, when cell moves out of sliding window

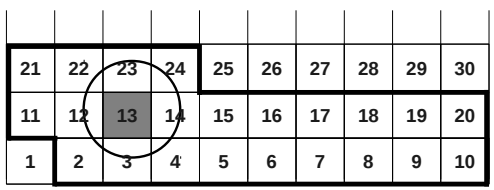
Implementation of the Compute Kernel

The software is written in object-oriented C++ (`Array of Structures`)
$\Rightarrow$ trade-off between cache-efficiency and vectorization



- Copy members needed to temporary `Structure of Arrays` on per–cell basis
- Vectorize over molecules per cell, not spatial coordinates, to fully exploit vector length
- Convert back

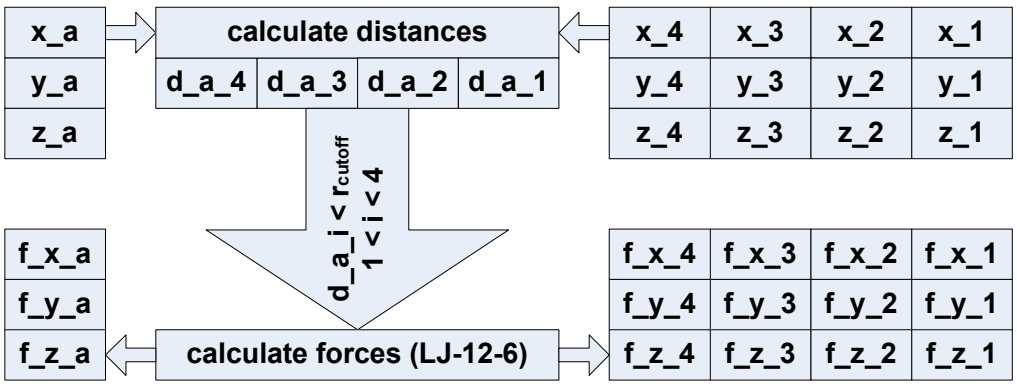
- SoA nicely integrates with sliding window traversal
  $\Rightarrow$ negligible memory overhead



Implementation of the Compute Kernel cont'd.

Implementation with intrinsics using single-precision AVX128 instructions:

- AVX256 only of advantage for large cutoff radii, i.e. long arrays per cell
- Better performance on AMD's Interlagos



Compute 4 interactions in parallel, mask pairs where distance $d\_a\_i > r_c$

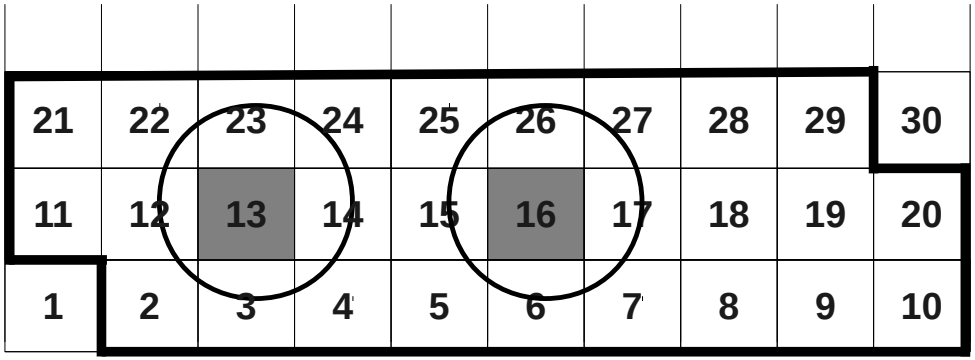
- Large number of elements masked $\Rightarrow$ gather / scatter would be beneficial

Lack of instruction-level parallelism in kernel:

- Multiplications dominate
- Multiplications show true dependencies: $U(r_{nm}) = 4\epsilon \cdot \left(\left(\frac{\sigma}{r_{nm}} \right)^{12} - \left(\frac{\sigma}{r_{nm}} \right)^{6} \right)$

Light-weight Shared-memory Parallelization

The Lennard-Jones kernel inhibits efficient use of superscalarity and pipelining
$\Rightarrow$ make use of Hyperthreading to increase efficiency



Extend window and introduce synchronization barrier

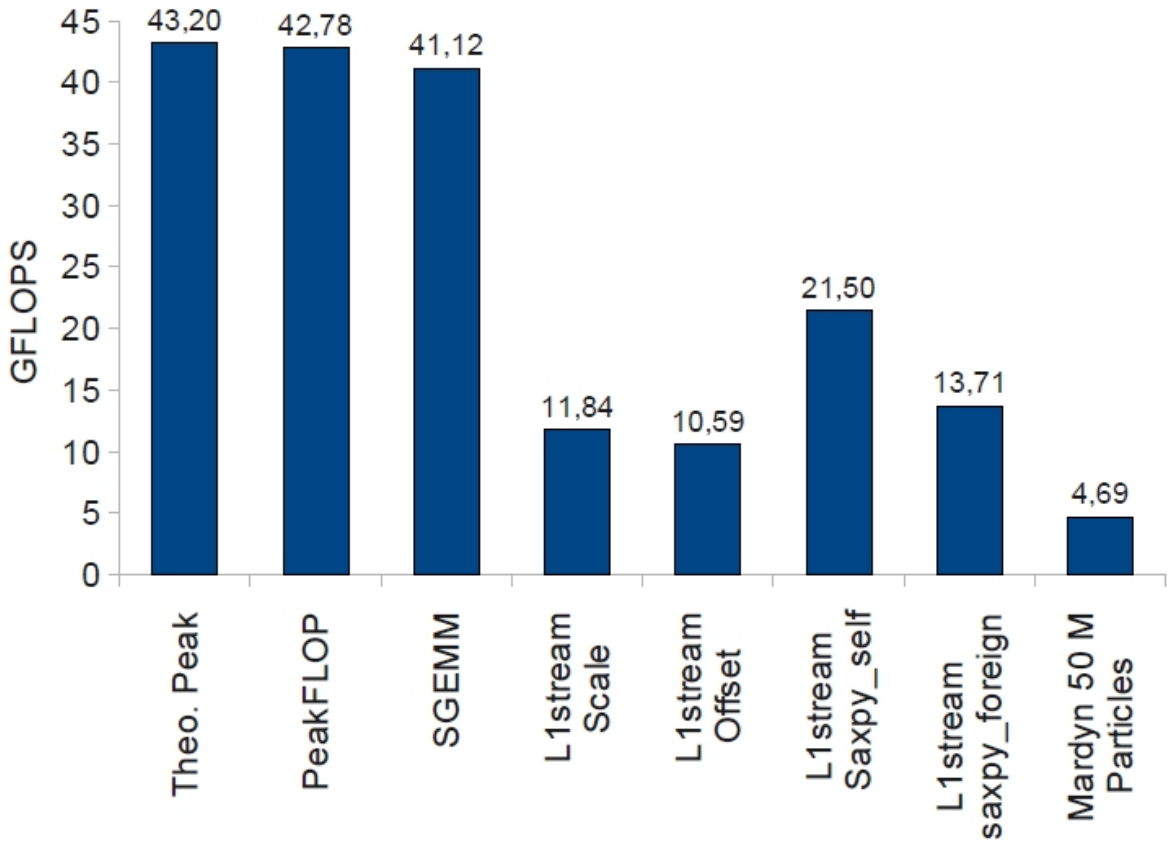
Preprocessing (converting particle data to SoA) and postprocessing (time integration) have to be executed sequentially by master thread

Not scalable, but $\Rightarrow$ for Hyperthreading optimal performance gain of 12 %:

- Cheap synchronization
- True sharing in private L1/L2 cache

Outline

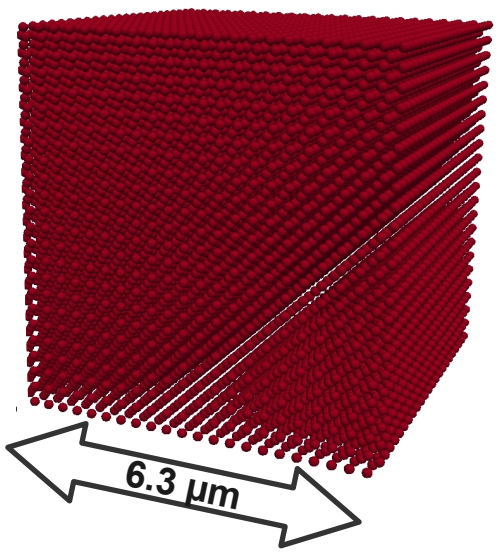
Single-core Performance



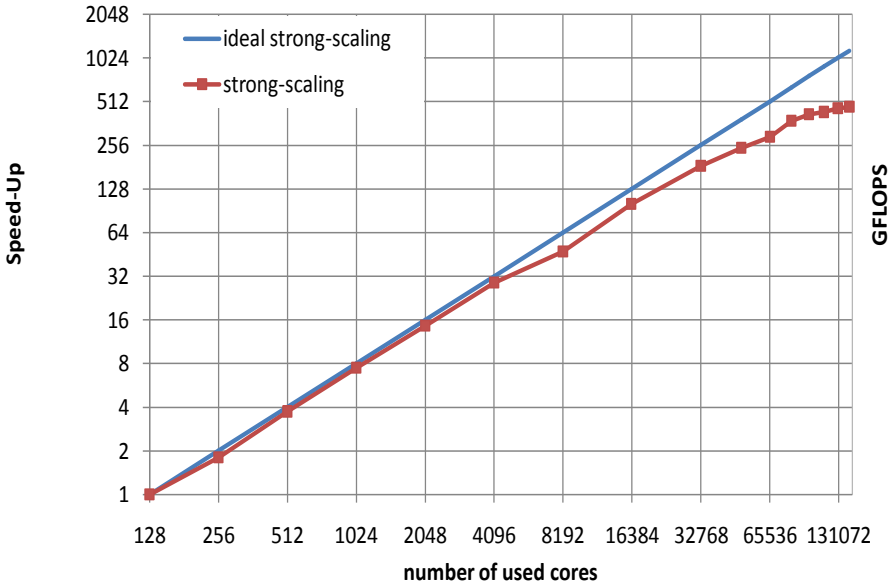
This corresponds to a single-core performance of 10.9 %

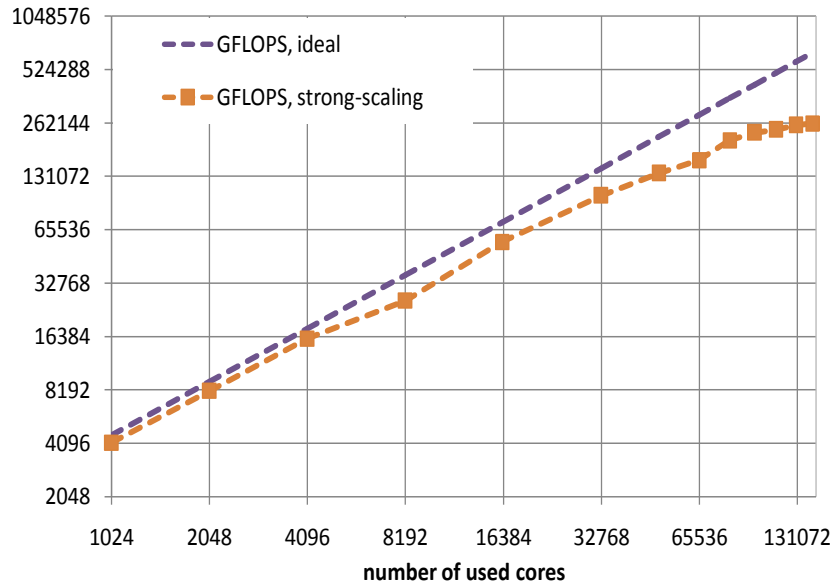
Experiment Setup

- Single-site Lennard-Jones particles setup on a body-centered cubid lattice
- Number density $\rho\sigma^3 = 0.78$, cut-off radius $r_c = 3.5\sigma$
- Strong scaling: $4.8 \cdot 10^9$ particles
- Weak scaling: $4.52 \cdot 10^8$ per node
- Largest simulation on 9126 nodes with $4.125 \cdot 10^{12}$ particles
- In case of liquid krypton: cube with edge-length $l = 6.3\,\mu$m



Scalability Results

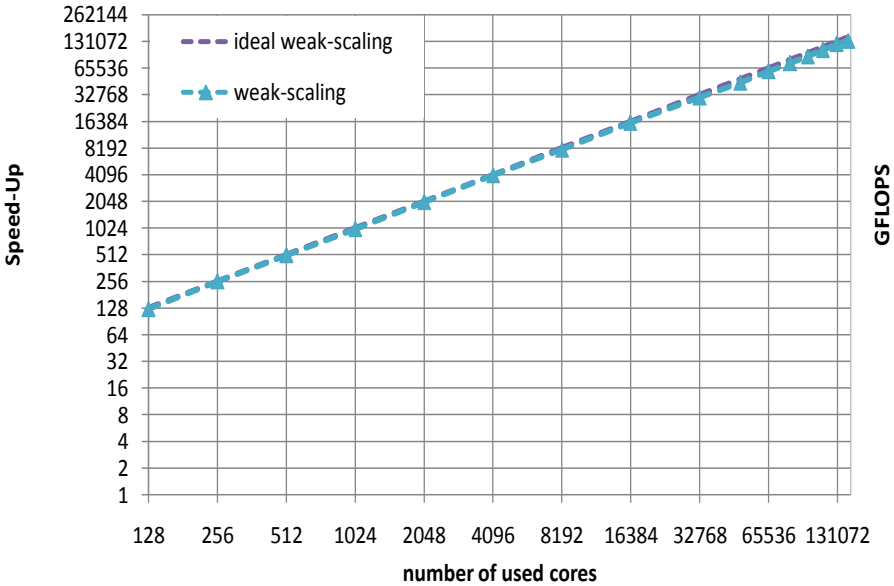


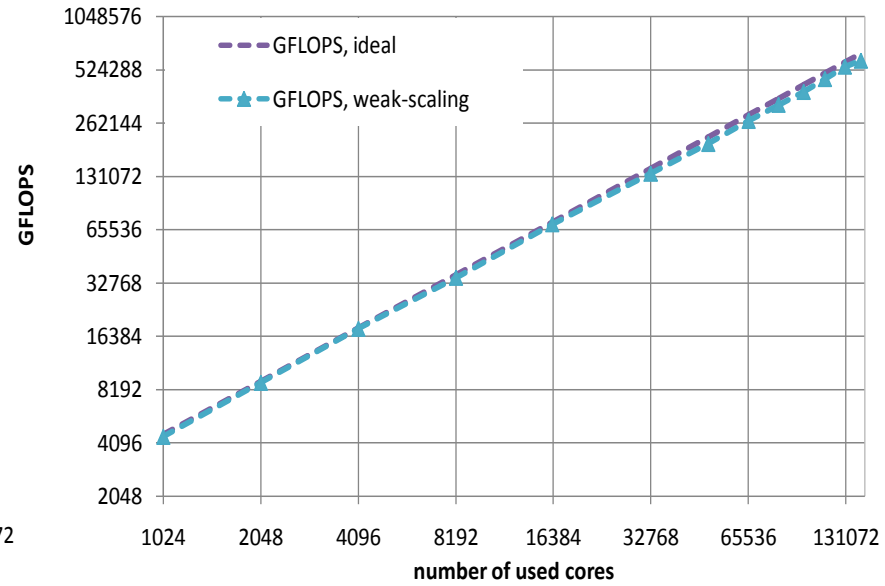


Strong scaling:

- $4.8 \cdot 10^9$ particles
- Particles per process for 146016 cores: $\approx$ 33000
- Peak performance of 260 TFLOPS on 146016 cores
- Runtime of 0.15 s / iteration
- Parallel efficiency of 42 % on 146016 cores (292032 threads) compared to 128 cores (256 threads)

Scalability Results





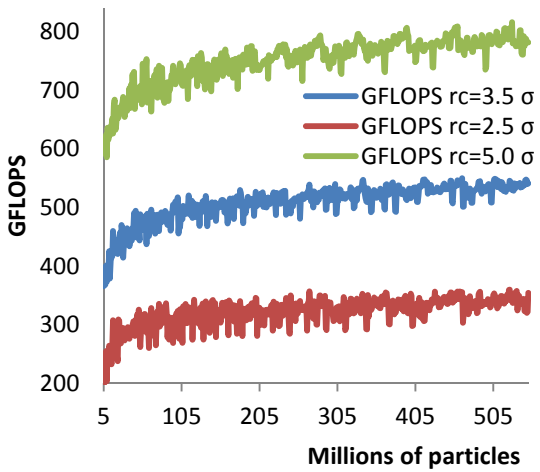
Weak scaling:

- $4.52 \cdot 10^8$ particles / node
- Peak performance of 591 TFLOPS on 146016 cores
- Parallel efficiency of 86.3 % on 146016 cores (292032 threads) compared to 1 core (2 threads),
  i.e. 9.4 % peak performance
- Maximum number of $4.125 \cdot 10^{12}$ particles with runtime of 40s / iteration

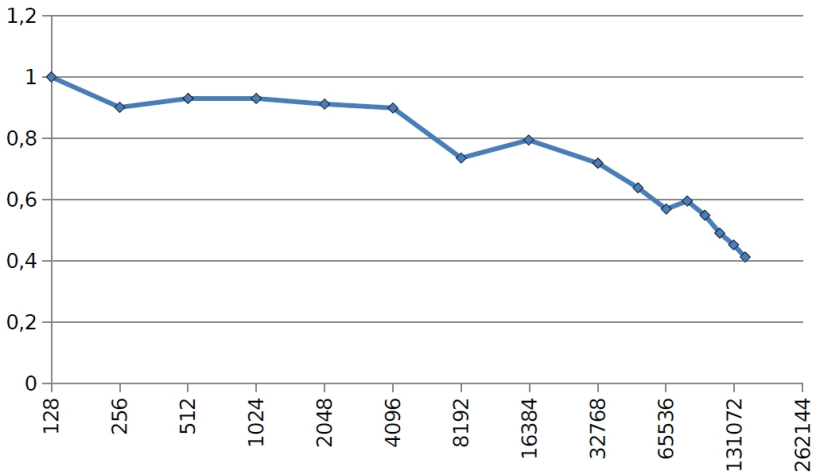
Influence of the number of particles

Investigate performance in dependence of particle number and cut-off radius:

- Larger cut-off radius favorable from application point of view
- Smaller cut-off radius favorable from computational point of view



Performance on 8 nodes / 128 cores



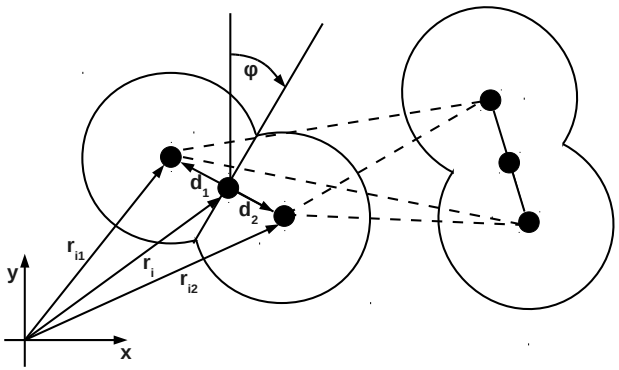
Parallel efficiency compared to 8 nodes / 128 cores

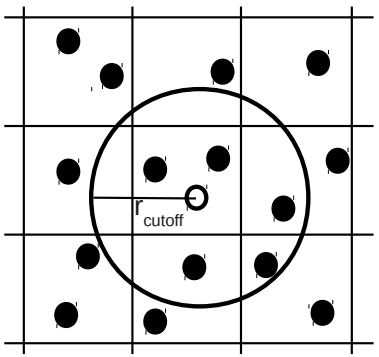
- For $N = 3 \cdot 10^8$ particles, flop rate of roughly 550 GFLOPs
- Strong scaling: decrease of efficiency largely due to decreasing single-node performance

Outline

Extension to Multi-Site Molecules

Rigid-body multi-site particles are described by position, velocity, orientation, angular velocity and type





Two molecules interact, if the distance between their centers of mass is smaller than the cutoff radius r_c.

Total effective force on a molecule i:

$$F_i = \sum_{j \in particles, j \neq i} \sum_{n \in sites_i} \sum_{m \in sites_j} F_{nm}(r_{nm}).$$

Torque on the molecule

$$\tau_i = \sum_{n \in sites_i} d_n \times F_n.$$

Potential energy:

$$U_{pot} = \frac{1}{2} \sum_{i} \sum_{j \neq i} \sum_{n \in sites_i} \sum_{m \in sites_j} U_{nm}(r_{nm}).$$

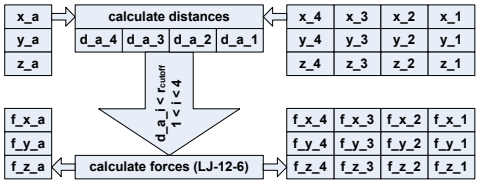
Virial pressure on the molecule

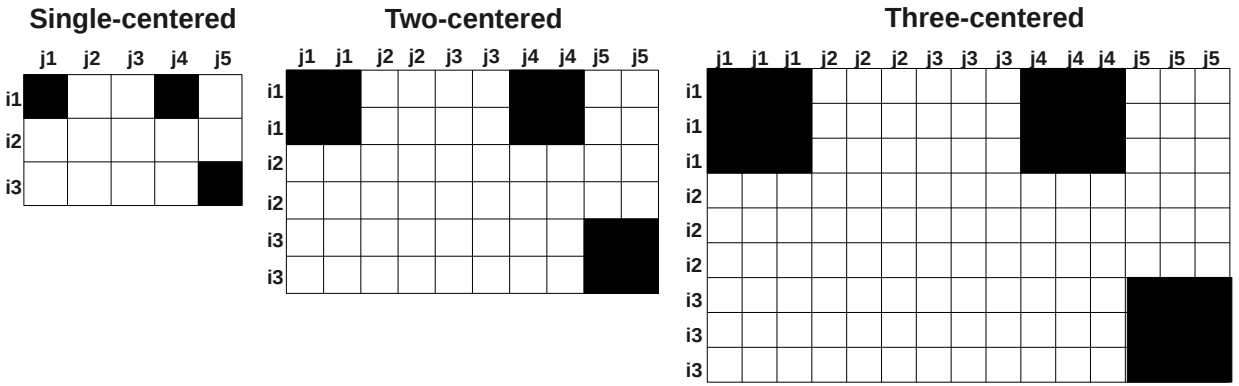
$$p = \frac{1}{2} \sum_{i} \sum_{j \neq i} \sum_{n \in sites_i} \sum_{m \in sites_j} F_{nm}(r_{nm}) \cdot r_{ij}.$$

Vectorized Implementation for Multi-Site Molecules

Straight-forward extension of scheme: store position of molecule with each site

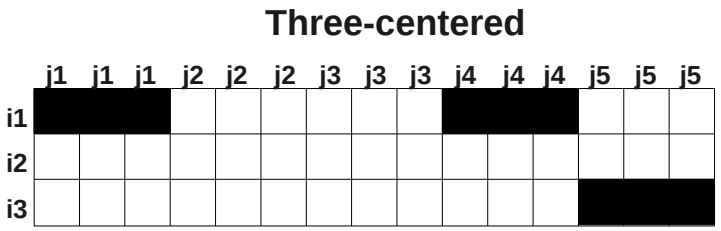
- **Pro:** Simple calculation of mask for blending forces
- **Con:** Quadratic increase of evaluations of the cutoff condition with number of sites



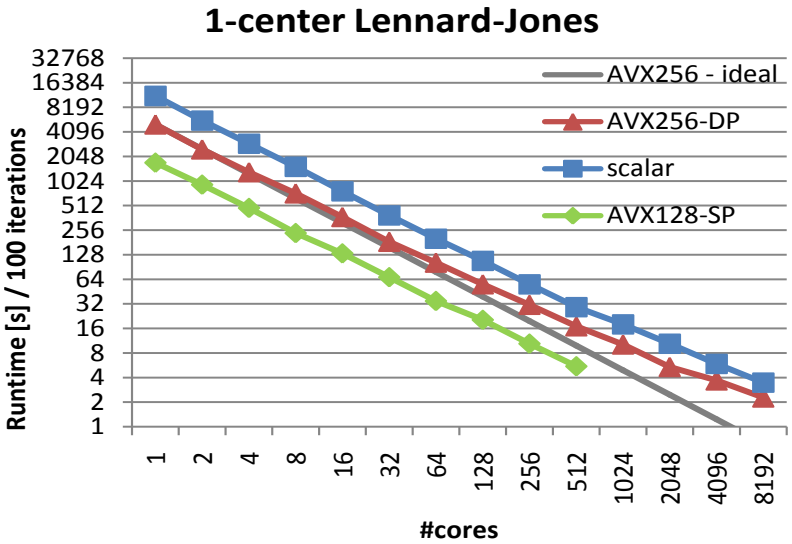


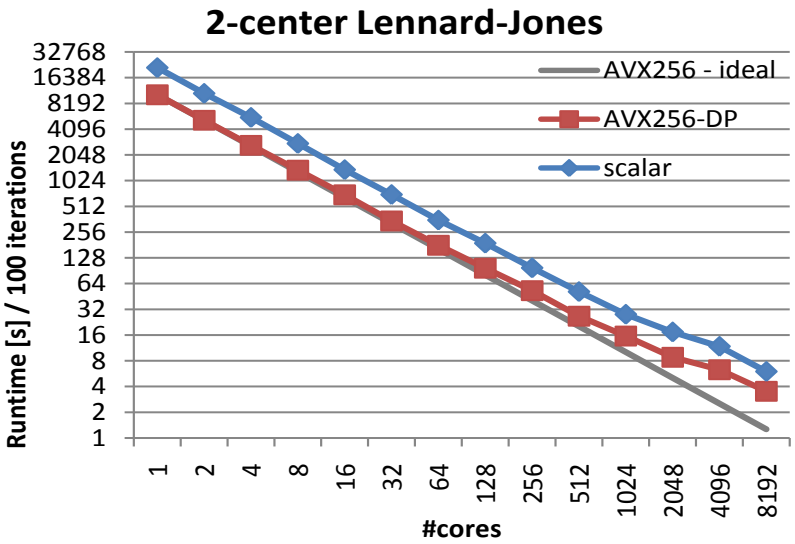
Solution:

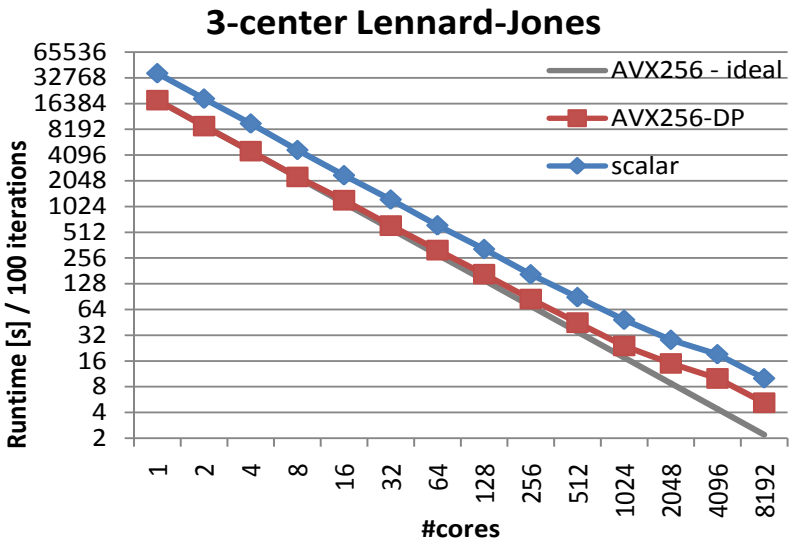
- Compute distances on molecule – center basis, store blend mask
- Skip force calculation completely for non-interacting molecules

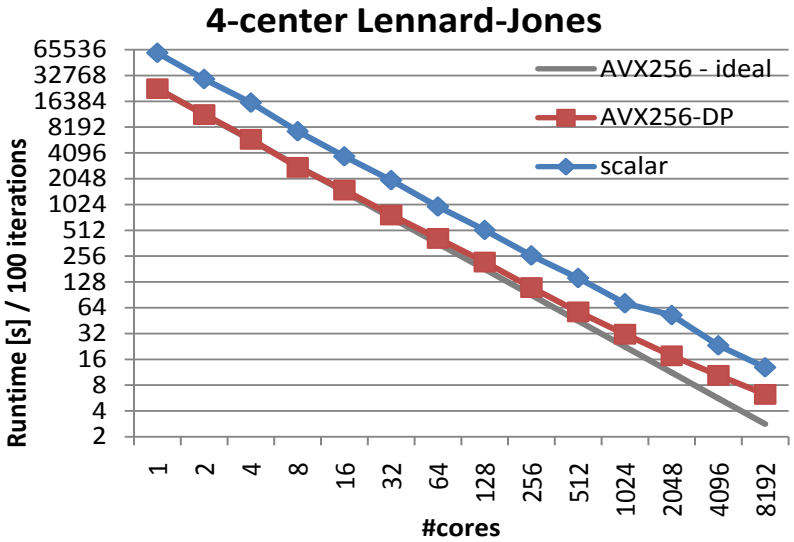


- First results of implementation for multi-site molecules with AVX256 (DP)
- Runtime for $10.7 \cdot 10^6$ particles at liquid density



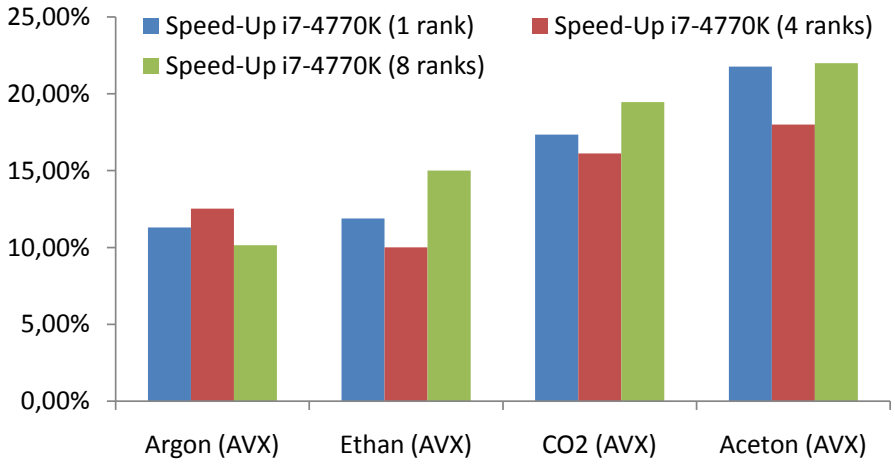


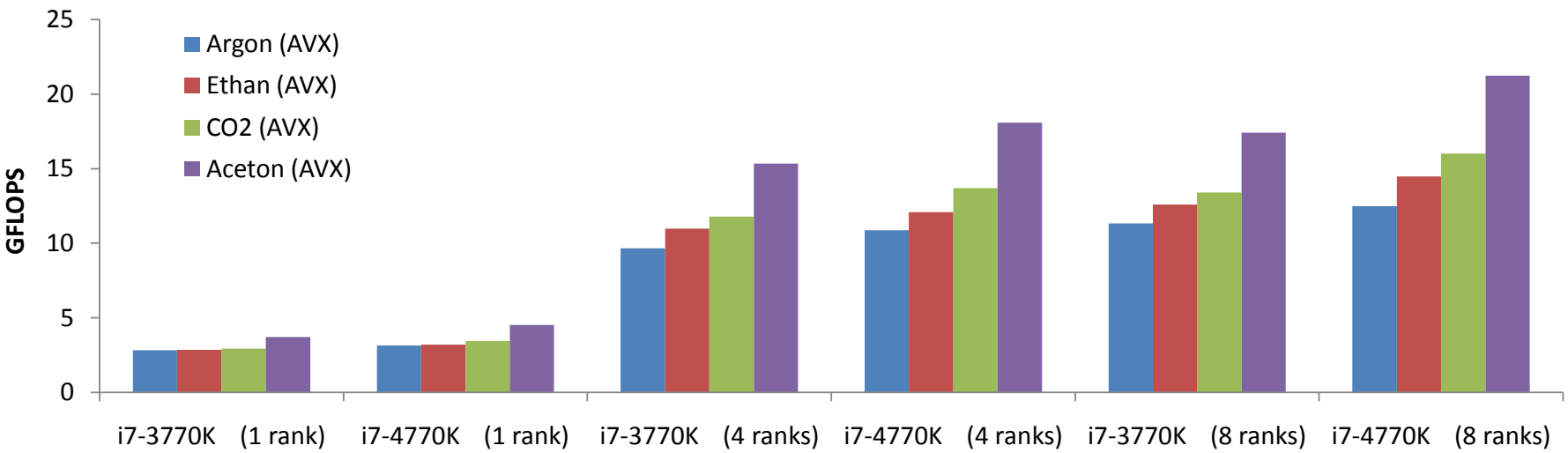




Outlook on Intel Haswell

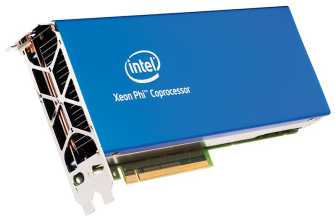
- Duplication of MUL/ADD-ports for FMA
- Increased L1-bandwidth (64-byte write / 32-byte read)

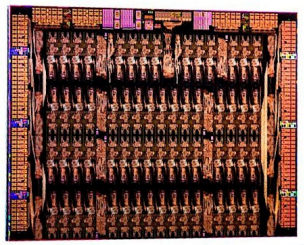




- i7-3770K: Ivy Bridge Quad-Core 3.5 GHz
- i7-3770K: Haswell Quad-Core 3.5 GHz

The Intel Xeon Phi SE10P

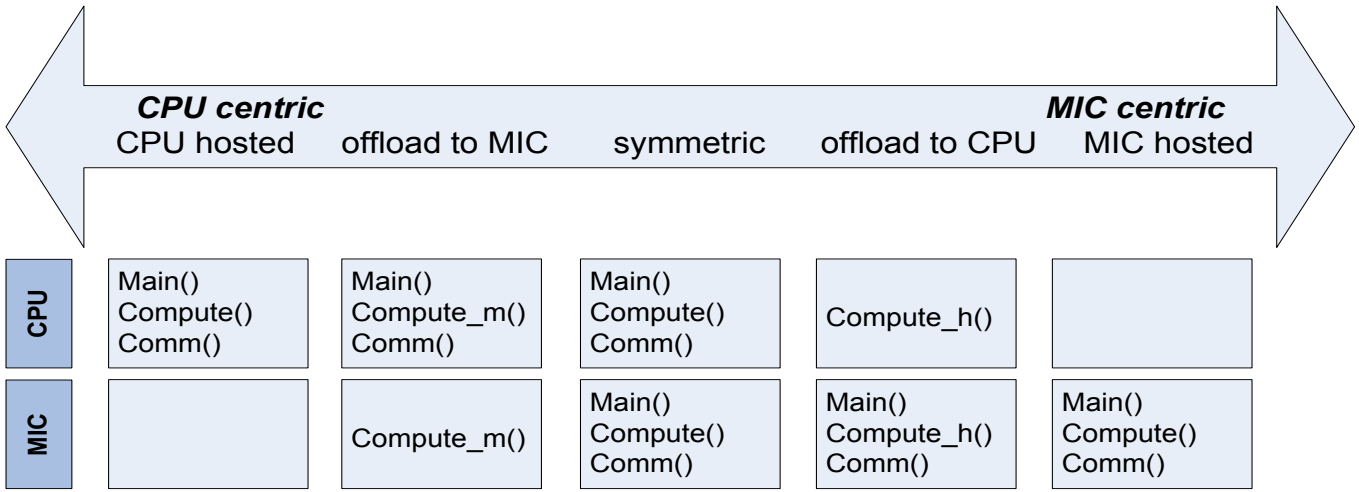




- 61 cores with 4 threads each at 1.1 GHz
- 32 KB L1$, 512 KB L2$, per core
- L2$ are kept coherent through a fast on-chip ring bus network
- 512 bit wide vector instructions, 2 cycle throughput
- 1074 GFLOPS Peak DP / 2148 GFLOPS Peak SP
- 8 GB GDDR5, 512 bit interface, 352 GB/s

http://ark.intel.com/de/products/71991/Intel-Xeon-Phi-Coprocessor-SE10P-8GB-1_100-GHz-61-core

Usage Models:



ls1 mardyn @MIC

So far, AVX-Code directly ported to MIC
AVX-256 (DP):

- Compute distances

d_a_m3	d_a_m2	d_a_m1	d_a_m0

- Execute force computation
- Compute distances

d_a_m7	d_a_m6	d_a_m5	d_a_m4

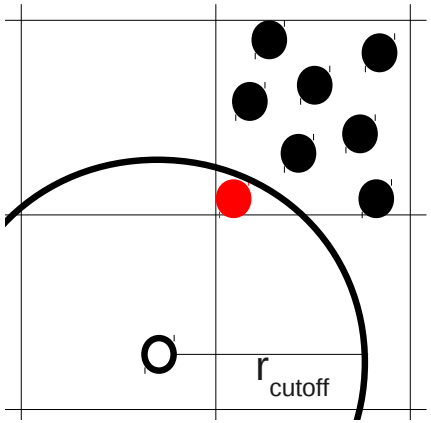
- Skip force computation

MIC-512 (DP):

- Compute distances

d_a_m7	d_a_m6	d_a_m5	d_a_m4	d_a_m3	d_a_m2	d_a_m1	d_a_m0

- Execute force computation

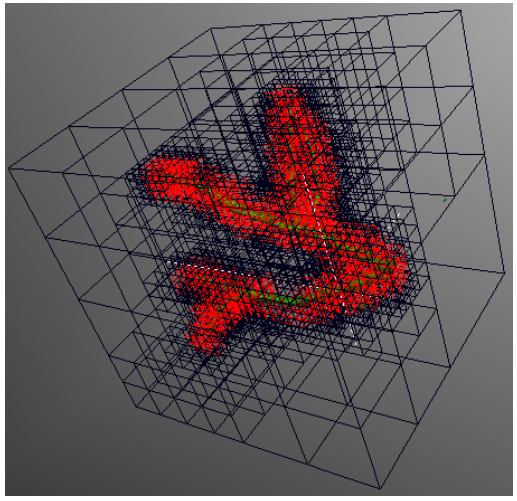


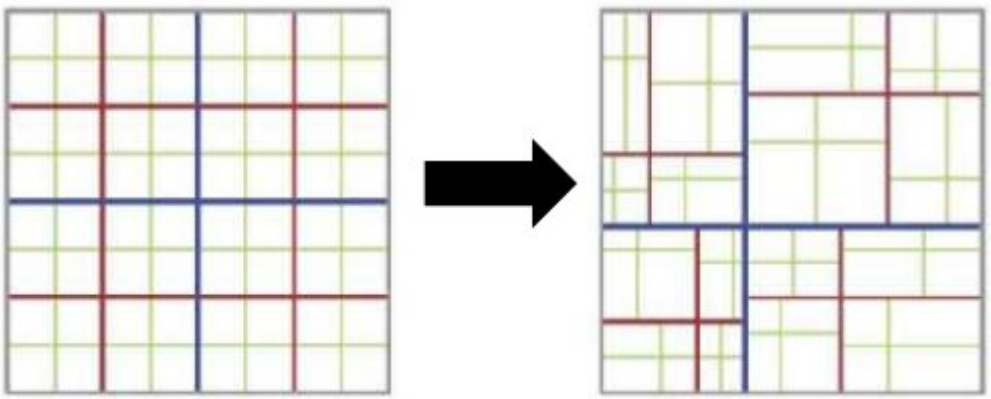
Ongoing work:

- Incorporation of gather / scatter instructions
- Shared-Memory parallelisation
- Load balancing considering heterogeneous system
- Use both SNB and MIC cores

KD-Tree-based Loadbalancing

- Hierarchical subdivision of the simulation volume through recursive bisection
- Choice of planes for bisection that lead to an equal load in the sub-domains
- Designed for strongly inhomogeneous molecular systems
- Capable to deal with rapidly changing inhomogeneity





Conclusion

- Applications in Chemical Engineering require large-scale molecular dynamics simulation
- We have presented our highly-tuned, highly-scalable implementation for single-site molecules
- Bridging the gap between microscopic and macroscopic scales will be soon possible
- Based on the experience gained we are able to tune our main code

Both efficient, hardware-aware programming and carefully tweaked algorithms are neccessary to tackle “large“ problems

Thank you for your attention!

Backup: Time Integration Methods

- Explicit Euler (1st order):

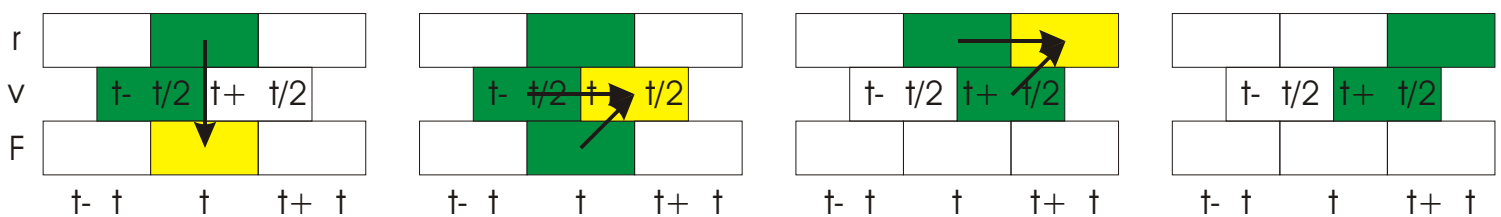
$$\vec{v}(t+\Delta t) \doteq \vec{v}(t) + \Delta t \frac{\vec{F}(t)}{m} \quad (1)$$

$$\vec{r}(t+\Delta t) \doteq \vec{r}(t) + \Delta t \frac{\vec{F}(t)}{m} \quad (2)$$

- Leap-Frog Integrator (2nd order):
  the velocity calculations are shifted for a half time step with respect to the position calculations:

$$\vec{v}(t+\frac{\Delta t}{2}) = \vec{v}(t-\tfrac{\Delta t}{2}) + \Delta t \frac{\vec{F}(t)}{m} \quad (3)$$

$$\vec{r}(t+\Delta t) = \vec{r}(t) + \Delta t\, \vec{v}(t+\tfrac{\Delta t}{2}) \quad (4)$$



Backup: Intel Haswell