The Fast Multipole Method in molecular dynamics

Berk Hess
KTH Royal Institute of Technology, Stockholm, Sweden

ADAC6 workshop Zurich, 20-06-2018

Partners

Funding

Horizon 2020
European Union Funding
for Research & Innovation

BioExcel

Key Software	GROMACS FAST. FLEXIBLE. FREE. · CPMD QM/MM · HADDOCK High-Ambiguity Driven Docking
Key Workflows and Platforms	Galaxy PROJECT · Open PHACTS · COMPSs · Open for Innovation KNIME · Apache Taverna
Partners	Barcelona Supercomputing Center Centro Nacional de Supercomputación · The University of Edinburgh · IRB Barcelona Institute for Research in Biomedicine · Forward Technologies · Jülich Forschungszentrum · Max Planck Gesellschaft · KTH · EMBL-EBI · epcc · Manchester 1824 The University of Manchester · Pistoia Alliance · Utrecht University

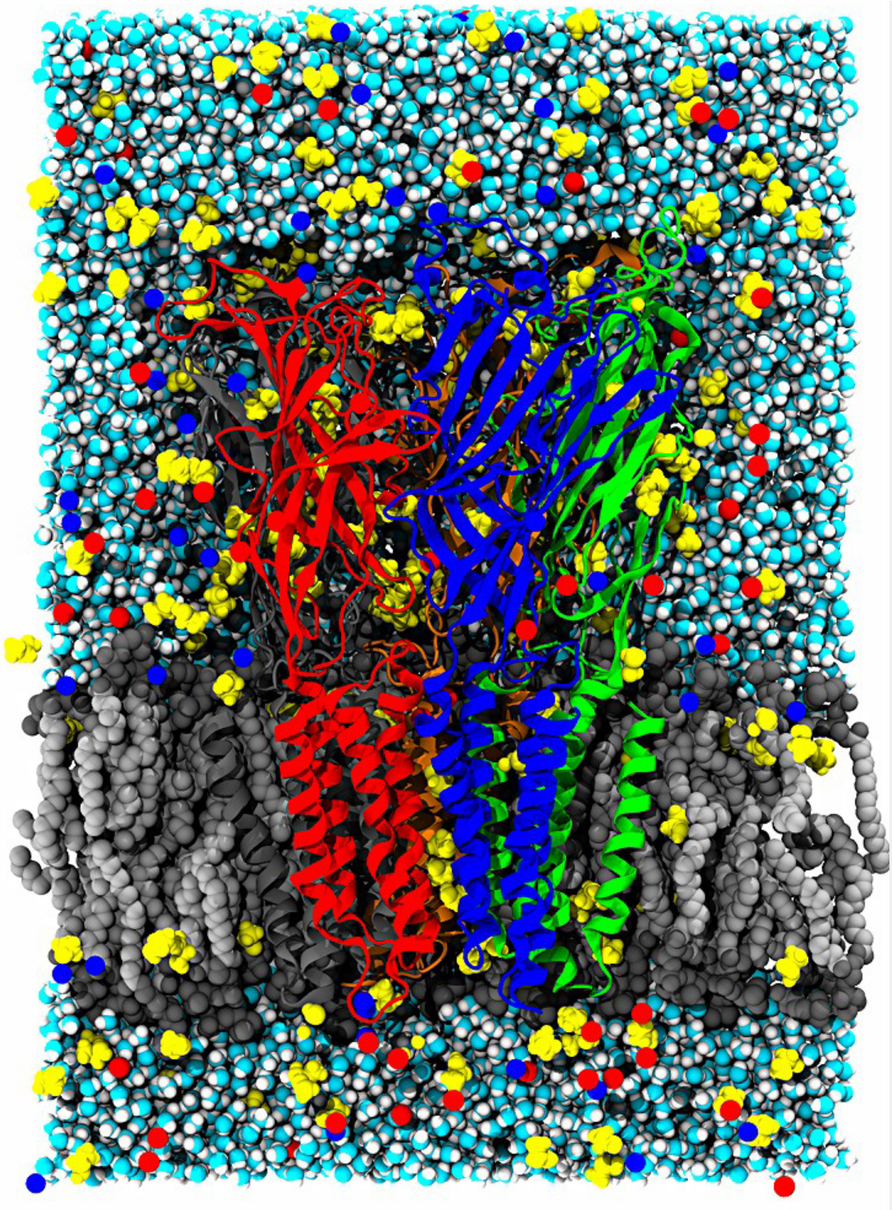
Molecular Dynamics of biomolecules

Powerful because:

- structure and dynamics in atomistic detail

Challenging because:

- fixed system size: **strong scaling needed**
- always more simulation needed than possible
- force-field (model) accuracy can be limiting
- results can be difficult to analyse



Molecular Dynamics

Newton's equation of motion

$$m_i \frac{d^2 \mathbf{x}_i}{dt^2} = \mathbf{F}_i = -\nabla_i V(\mathbf{x}) \ , \quad i = 1, \ldots, N$$

Simple (symplectic) integrator

$$v_i(t + \frac{\Delta t}{2}) = v_i(t - \frac{\Delta t}{2}) + a_i(t)\Delta t$$

$$x_i(t + \Delta t) = x(t) + v_i(t + \frac{\Delta t}{2})$$

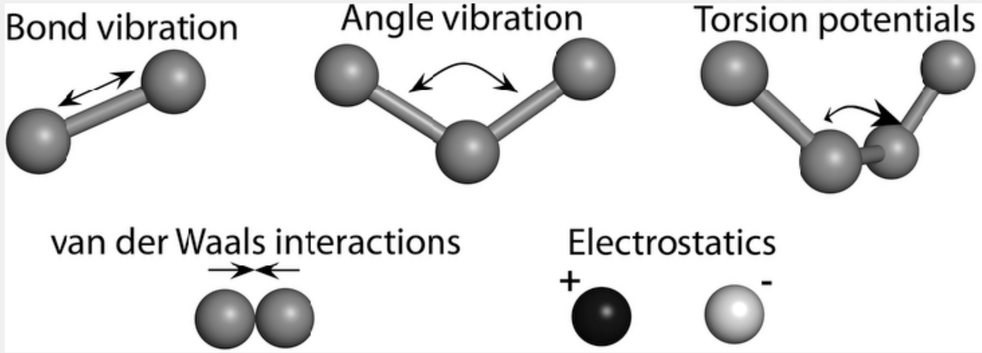
Fundamental issue: **sequential** problem

Δt ~ 1 fs, timescales of interest μs - s

⇨ need **10^9 - 10^{15}** steps!

Computational cost mostly in computing the forces

$$U(\mathbf{r}) = \sum_{bonds} U_{bond}(\mathbf{r}) + \sum_{angles} U_{angle}(\mathbf{r}) + \sum_{dihs} U_{dih}(\mathbf{r}) + \sum_{i} \sum_{j>i} \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} + \frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}$$



We need the force on each particle i

$$\mathbf{F}_i = -\frac{dU}{d\mathbf{r}_i}$$

pair-sum: largest cost, use cut-off (and long-range electrostatics algorithm)

The MD strong scaling challenge

To reach 10^9 - 10^{15} steps we need to minimise the **walltime per step**

The best MD codes in the world are at **100 microseconds** per step (on a single node):

- at ~ 200 atoms per CPU core ⇨ **we run in L1/L2 cache**
- at ~ 3000 atoms per GPU
- but systems of interest are usually > 100000 atoms

Extreme demands on hardware and parallelisation libraries (MPI, OpenMP):

- ideally overheads should be **< 1 microsecond**

GROMACS

- Started as hardware+software project in Groningen (NL)
- Now core team in Stockholm and many contributors world-wide
- Focus on high performance: efficient algorithms, SIMD & GPUs
- Good parallel scaling
- Compiles and runs efficiently on nearly any hardware
- Open source, LGPL
- Incorporate important algorithms for flexibly manipulating systems
- C++, MPI, OpenMP, CUDA, OpenCL
- C++ and Python API in the works

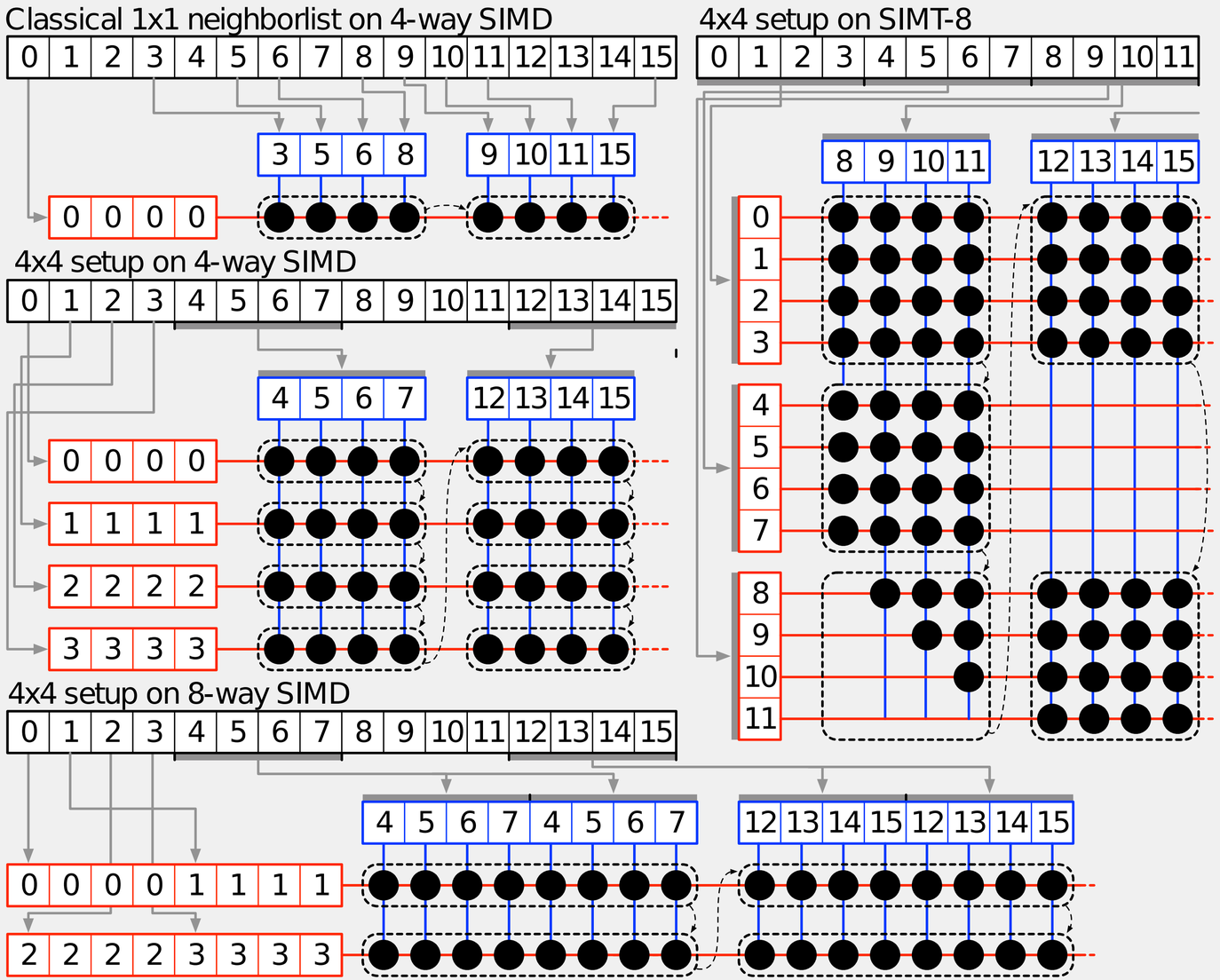
Highly optimised vectorization:

Small C++ SIMD layer with highly optimised math functions for:

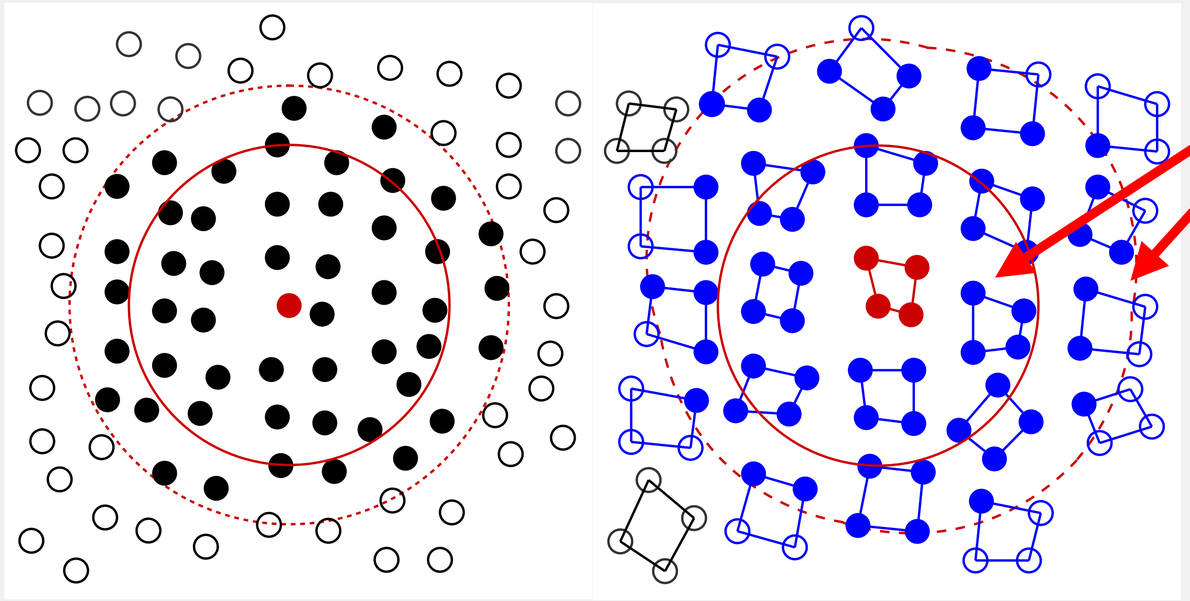
SSE2/4, AVX2-128, AVX(2)-256, AVX-512, ARM, VSX, HPC-ACE

CPU kernels reach 50% of peak

CUDA and OpenCL need separate kernels



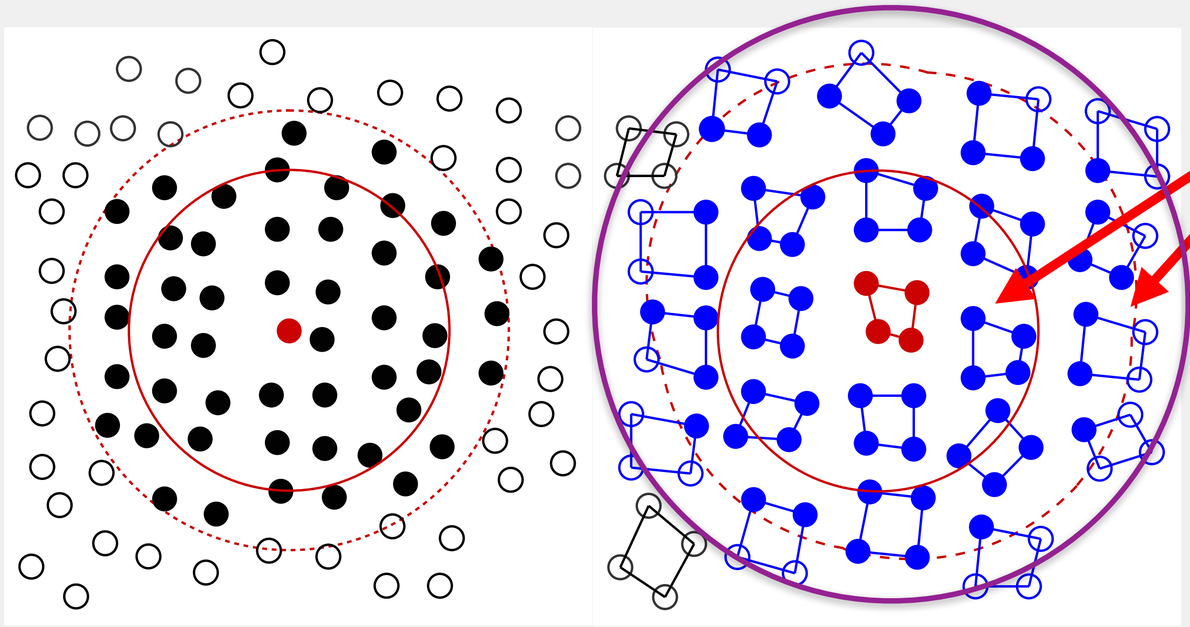
Atom clustering (regularization) and pair list buffering



We can optimize the size of this buffer

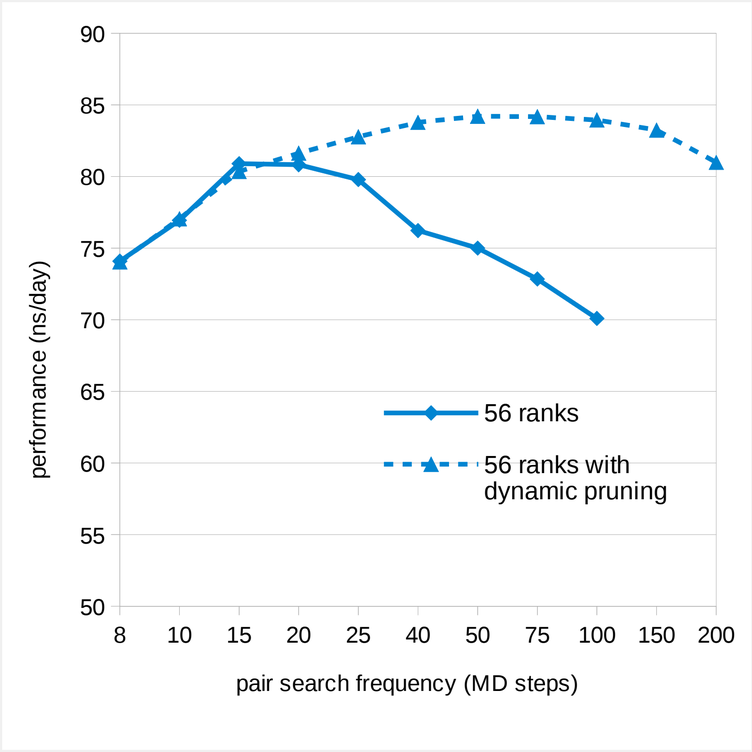
Larger buffer means more calculations, but we can update the neighbor list less frequently

Atom clustering and pair list buffering



We can optimize the size of this buffer

Larger buffer means more calculations, but we can update the neighbor list less frequently



New in GROMACS-2018: dual-pair list:

- reduces overhead
- less sensitive to parameters

Intra-rank parallelisation: OpenMP

GROMACS has efficient parallelization of all algorithms using MPI + OpenMP

OpenMP is (performance) portable, but limited:

- No way to run parallel tasks next to each other
- No binding of threads to cores (cache locality)

Need for a better threading model, requirements:

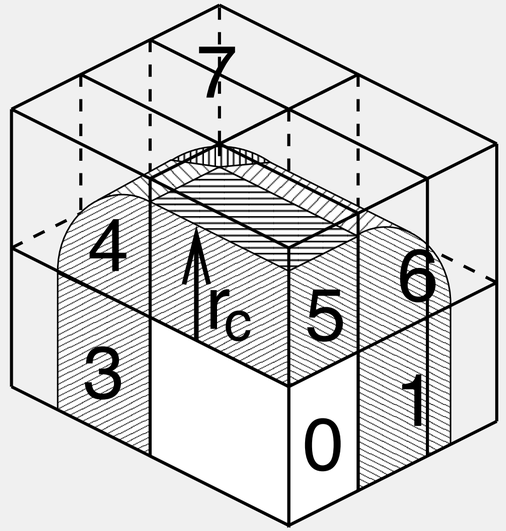
- Extremely low overhead barriers (all-all, all-1, 1-all)
- Binding of threads to cores
- Portable

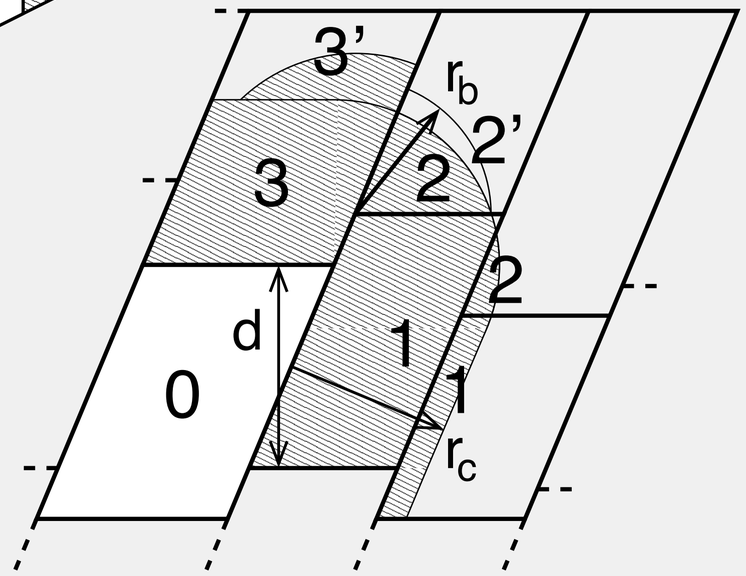
Spatial decomposition

8th-shell domain decomposition:

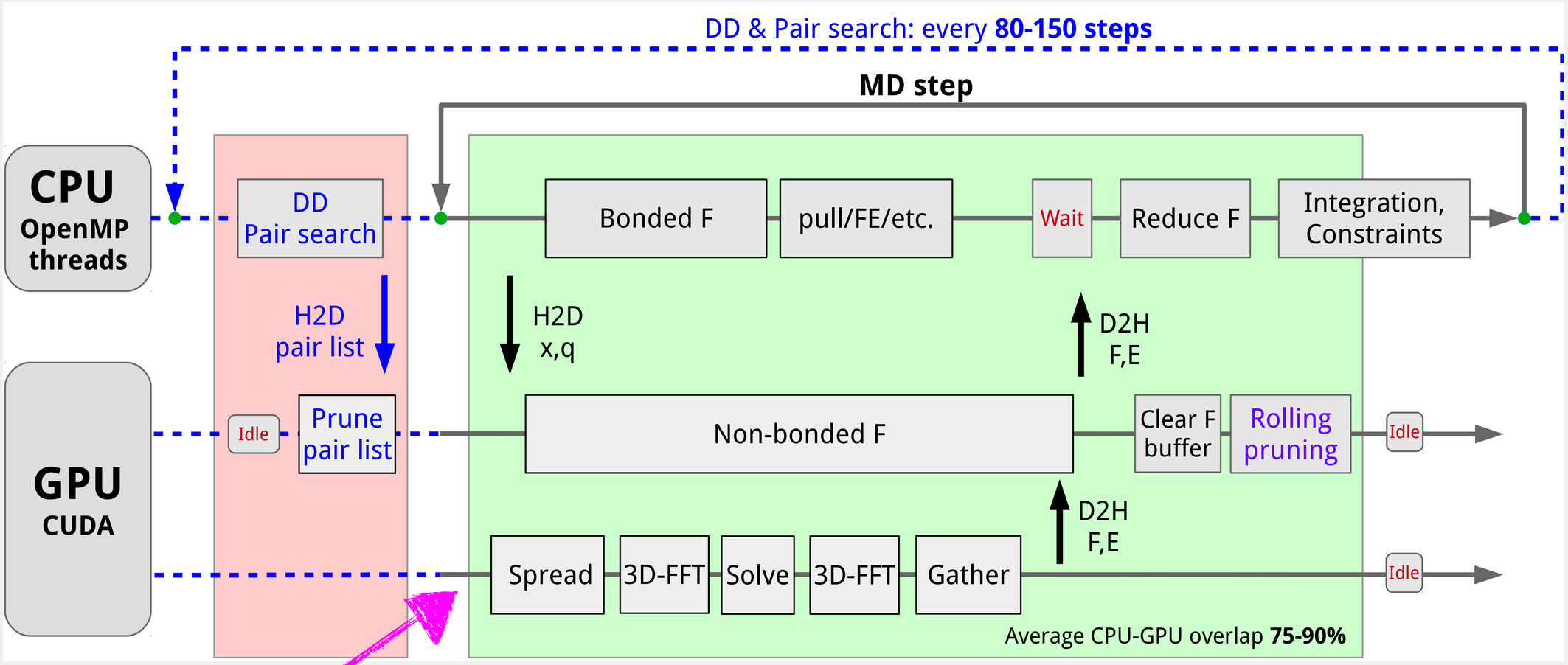
- minimizes communicated volume
- minimizes communication calls
- implemented using aggregation of data along dimensions
- 1 MPI rank per domain
- dynamic load balancing

Question: still the best approach for networks that support many messages?





Hess, Kutzner, Van der Spoel, Lindahl; J. Chem. Theory Comput. 4, 435 (2008)



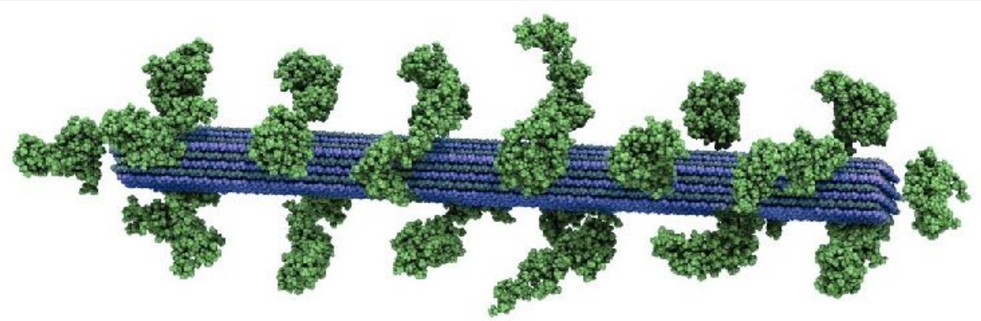
100-500 µs

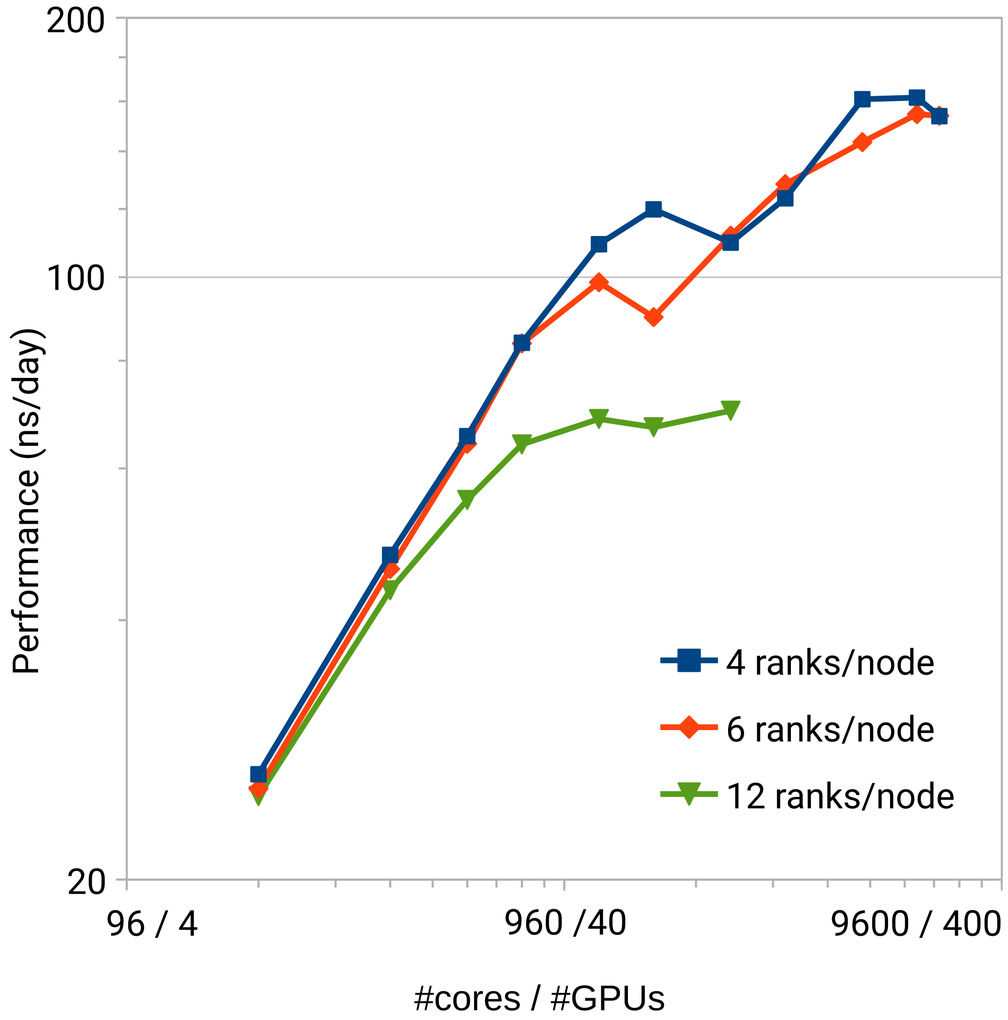
New in GROMACS-2018: PME on GPU!

GROMACS strong scaling

Lignocellulose benchmark

- 3.3 million atoms
- No long-range electrostatics!
- machine: Piz Daint at CSCS
  - Cray XC50
  - NVIDIA P100 GPUs



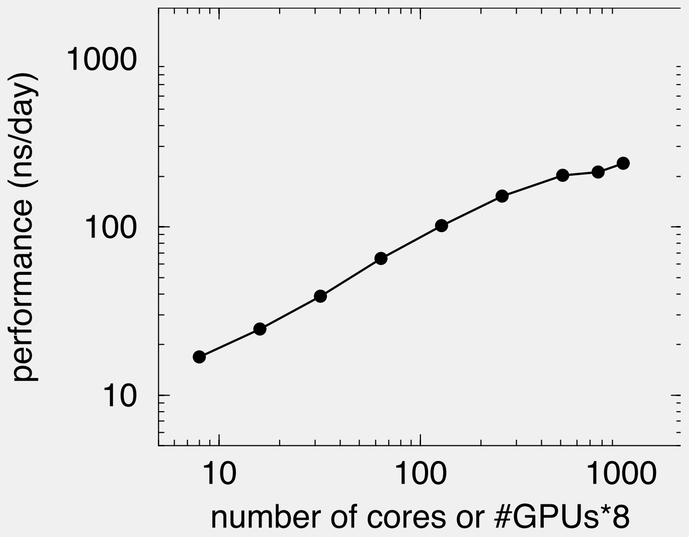


Strong scaling issues

At 100s of microseconds per step:

- **The 3D-FFT in PME electrostatics**
- MPI overhead
  - we need MPI_Put_notify()
- OpenMP barriers take significant time
- Load imbalance
- CUDA API overhead can be 50% of the CPU time
- Too many GROMACS options to tweak manually

140000 atom ion channel
on Piz Daint xc



Long-range electrostatics

All examples up till now without long-range electrostatics, but needed in most cases

We need all-vs-all Coulomb interactions

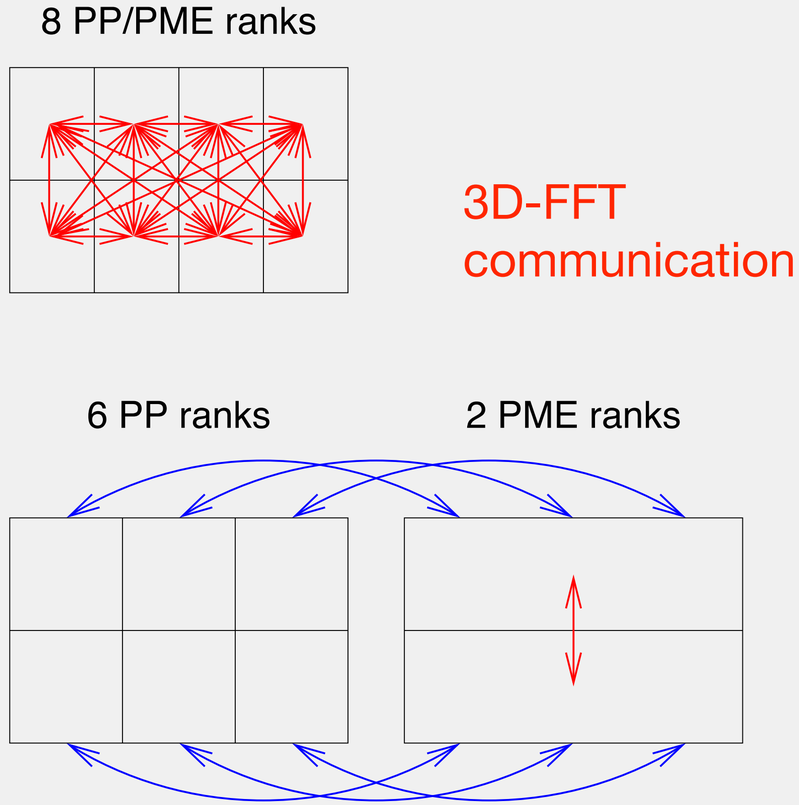
All MD codes use Particle-mesh Ewald (PME) for electrostatics

PME uses a 3D-FFT, 64^3 - 128^3 grid points

Issues:

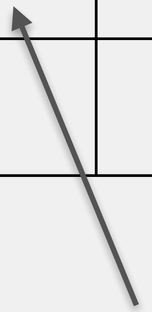
- The 3D-FFT requires two grid transposes that require All-to-All communication in slabs
- Pencil decomposition clashes with 3D decomposition
- **3D-FFT scaling is the bottleneck for MD**

GROMACS Multiple-Program Multiple-Data parallelisation improves scaling by a factor 4



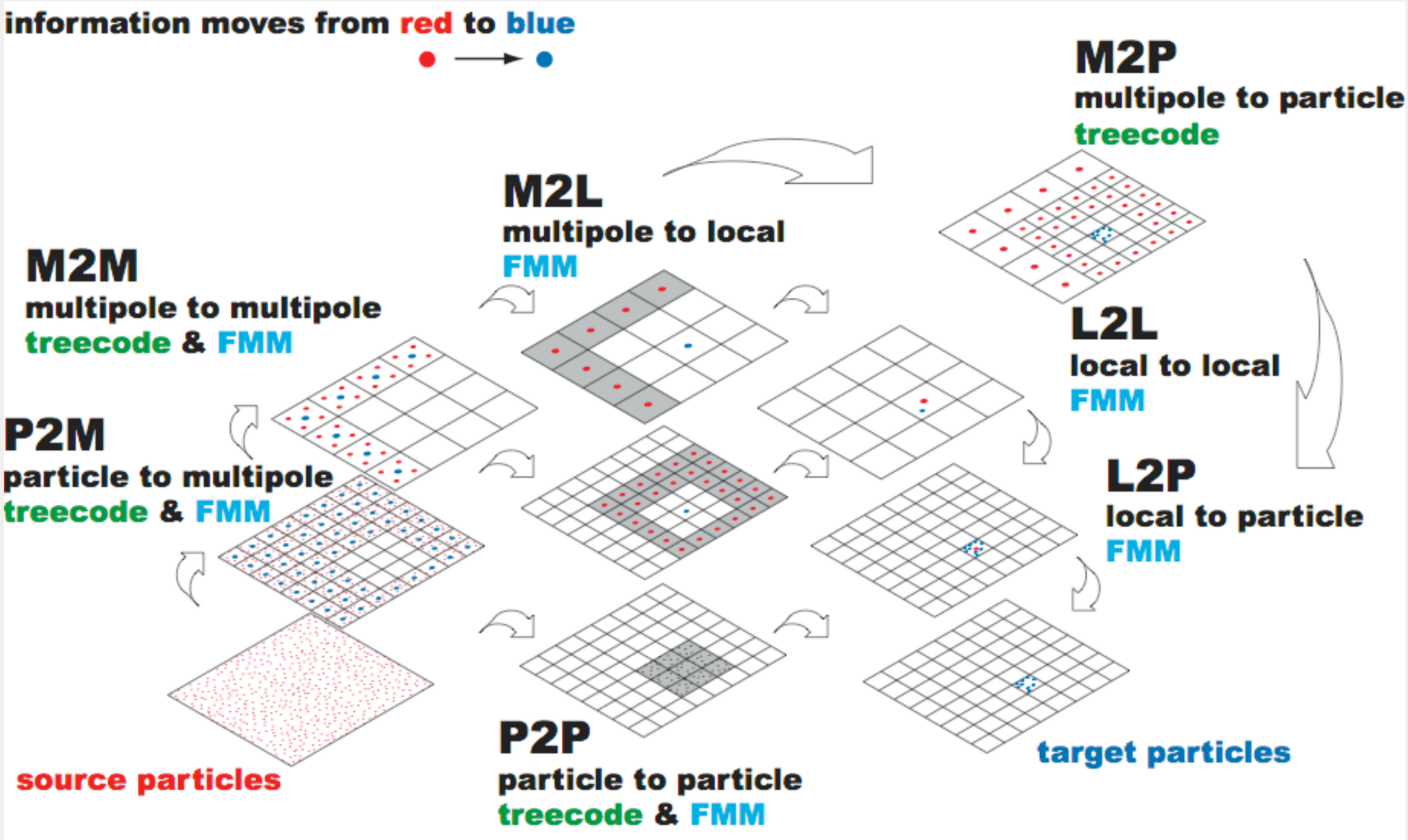
Long-range electrostatics methods for molecular dynamics

	in use in MD	computational cost	pre-factor	communication cost
Ewald summation	since 70s	$O(N^{3/2})$	small (FFT)	high
PPPM / Particle Mesh Ewald	since 90s	O(N log N)	small (FFT)	high
Fast Multipole Method	not yet	O(N)	larger	low
Multigrid Electrostatics	2010	O(N)	larger	low



we recently solved the energy conservation issue

FMM scheme



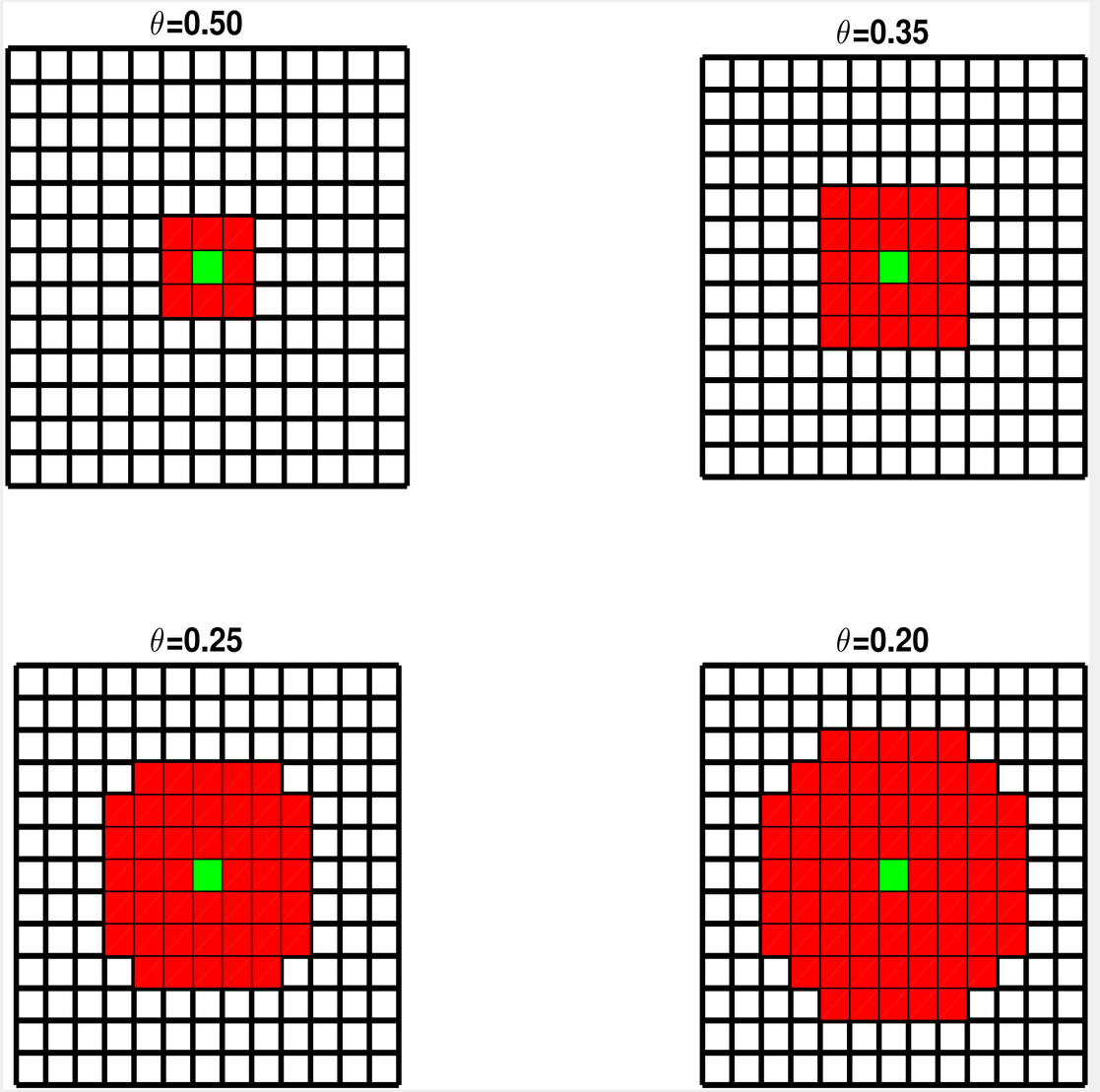
Rio Yokota
Tokyo Tech

The accuracy of FMM is controlled by

- the order of the expansion(s)
- the theta parameter

Main issue for FMM in MD:

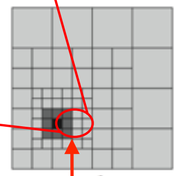
- We want low accuracy (or better: high speed)
- We want energy conservation

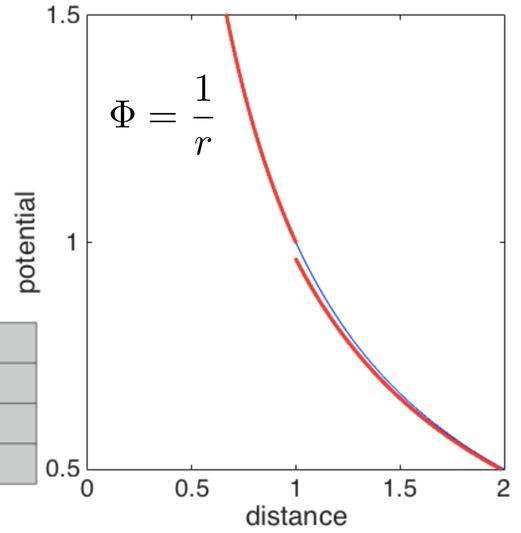


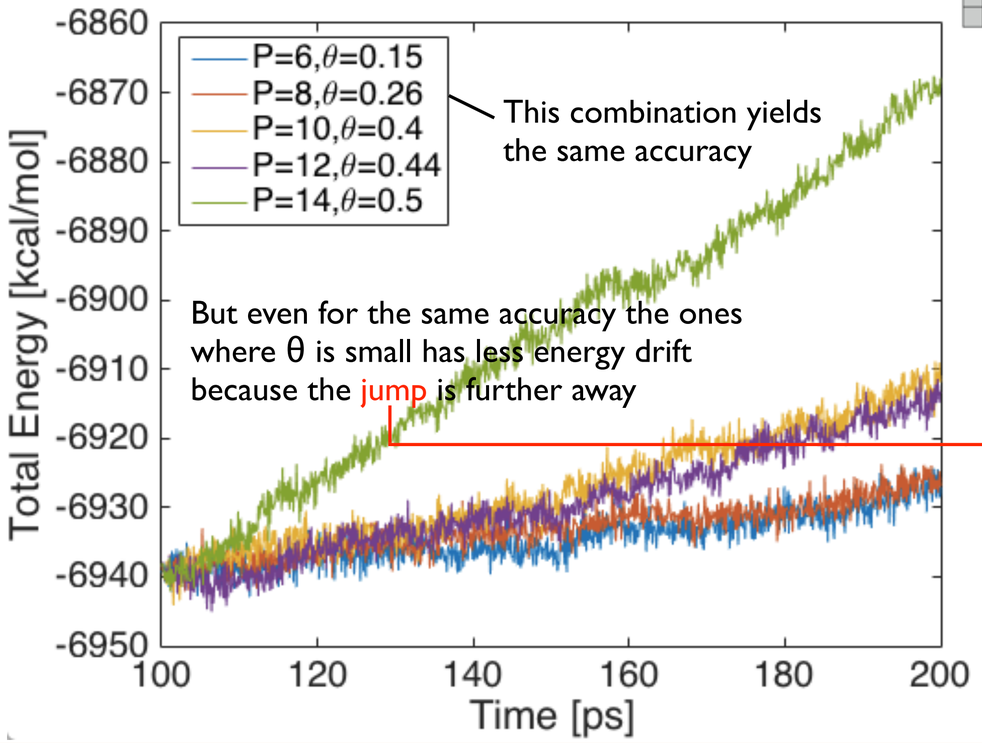
FMM for molecular dynamics

p : Order of expansion
θ : Opening angle

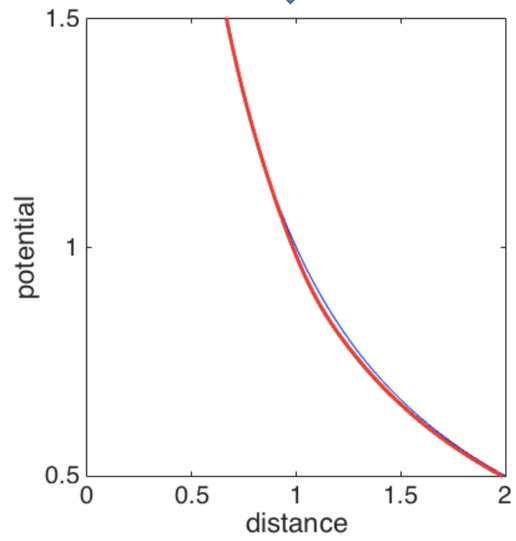
Complexity	Error
$\mathcal{O}(p^3\theta^{-3})$	$\mathcal{O}(\theta^p)$







What happens if we smooth the jump?

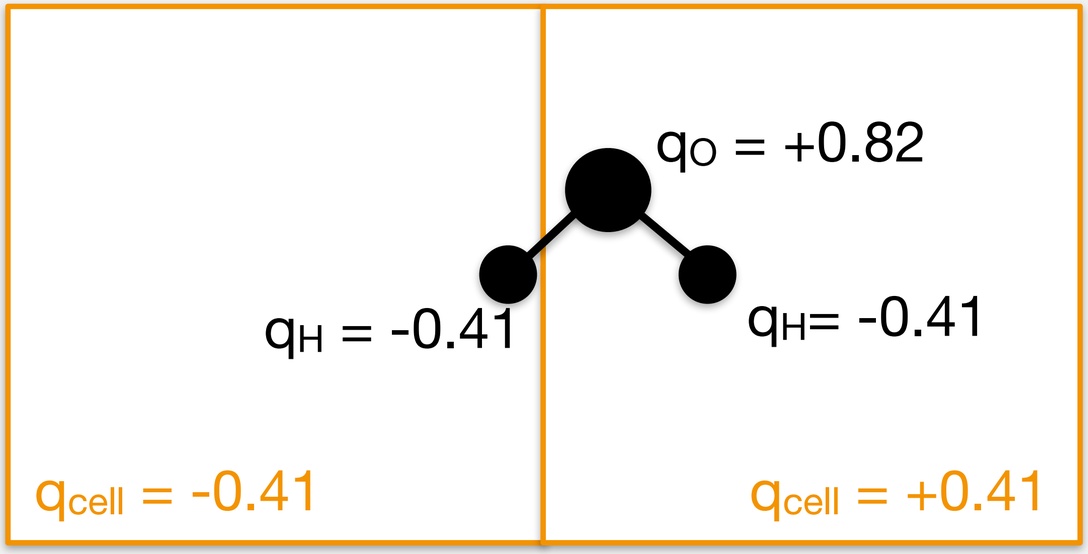


work by
Rio Yokota

Main physics issue with FMM in MD:

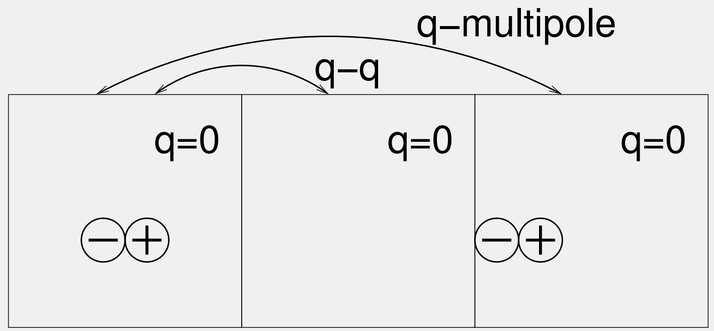
- All atoms have partial charges, but molecules are mostly locally neutral
- The sharp boundaries of FMM cells introduce charges at the boundaries

a neutral water molecule crossing 2 FMM cells

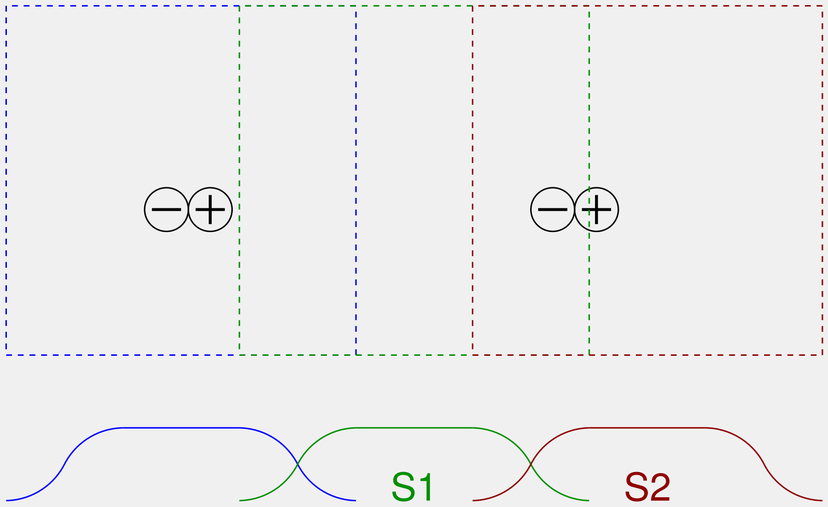


at long distance
water is a dipole

FMM regularization: add smooth overlap between cells



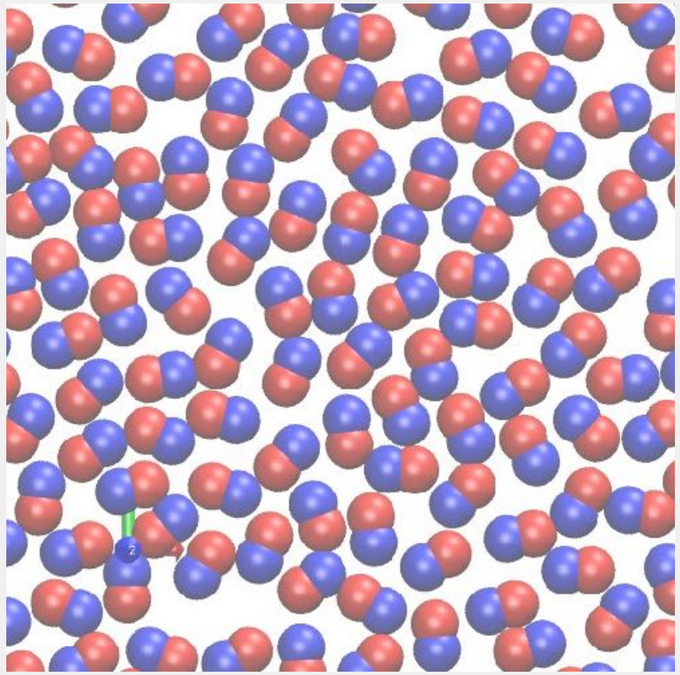
orig. idea: Chartier, Darrigand Faou,
BIT Num. Math. 50, 23 (2010)

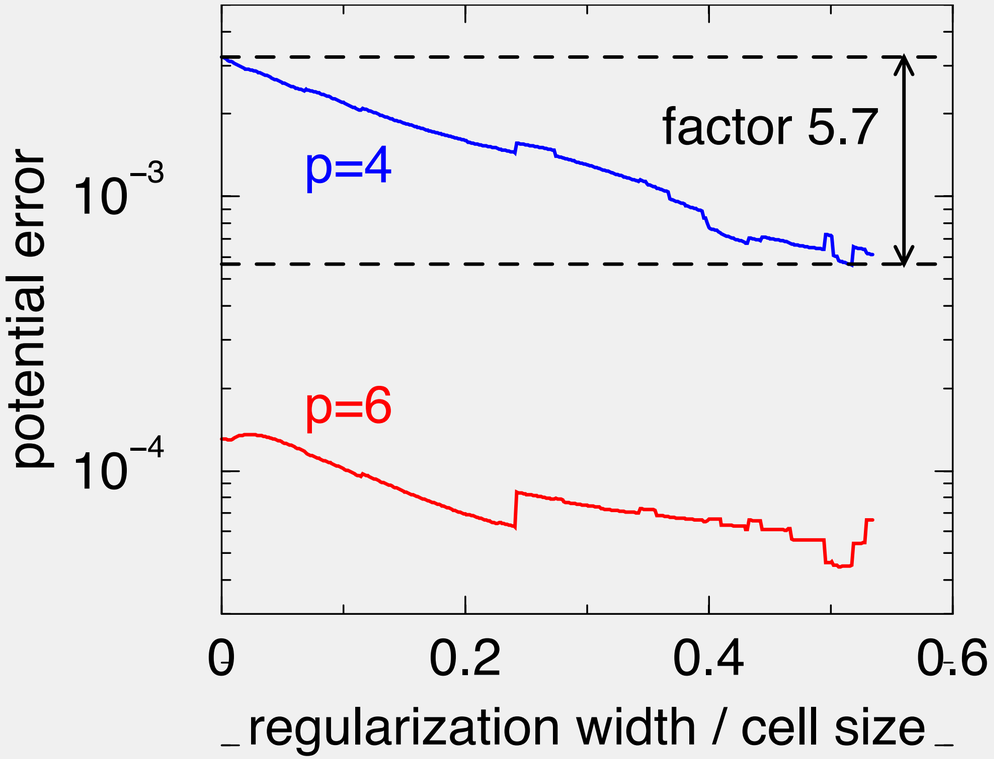


$$V = s_1(x)V_1(x) + s_2(x)V_2(x)$$

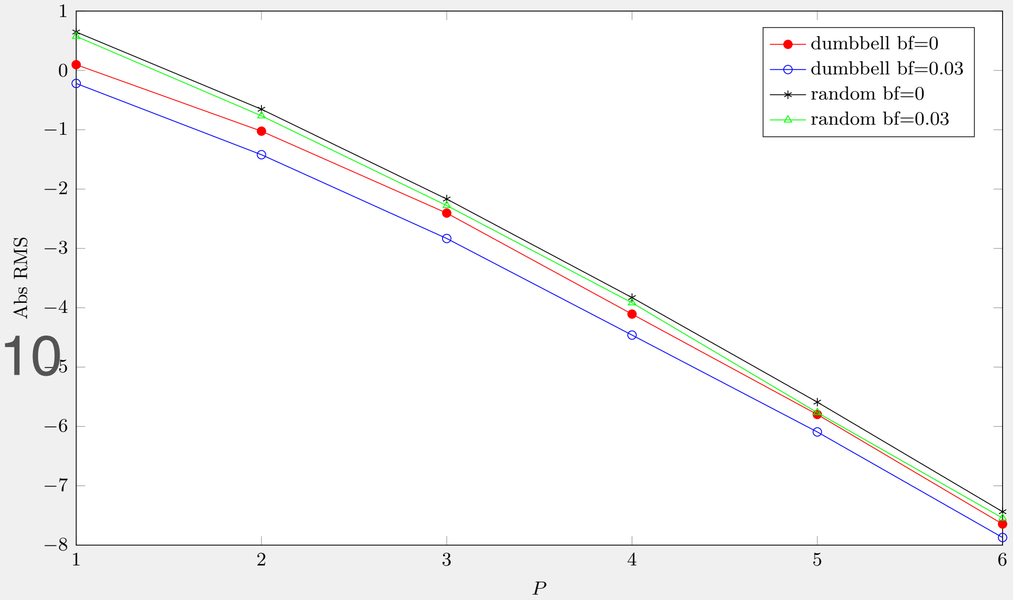
$$\begin{aligned} F &= -\nabla V \\ &= s_1(x)F_1(x) + s_2(x)F_2(x) \\ &\quad -s_1'(x)V_1(x) - s_2'(x)V_2(x) \end{aligned}$$

- Davoud Saffar Shamshir Gar has analyzed the accuracy of regularization
- Currently: 2D FFM with full regularization
- Results for charge pairs with θ=0.35 = second nearest neighbors, 8 atoms per cell

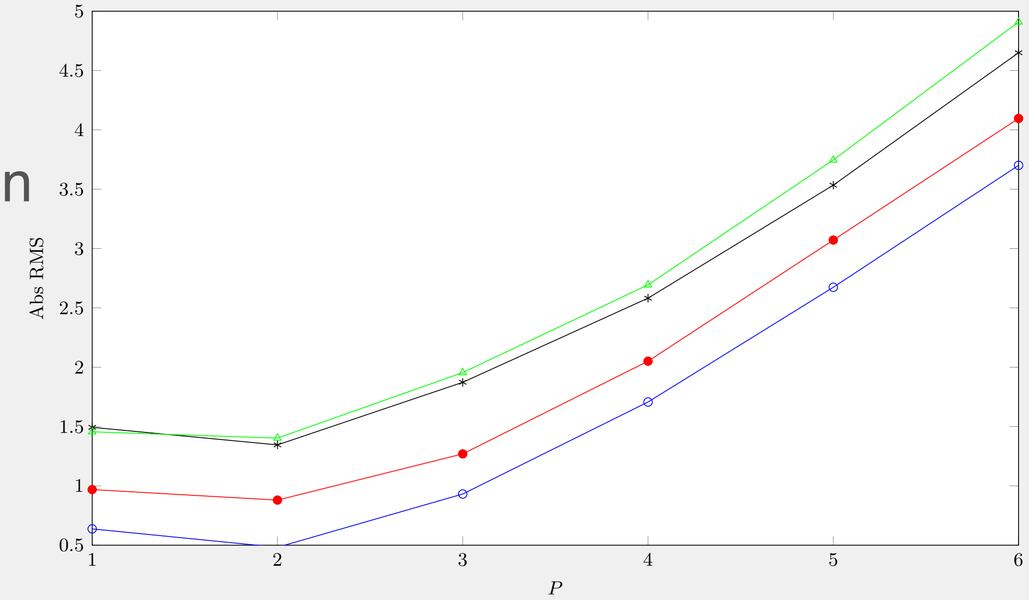




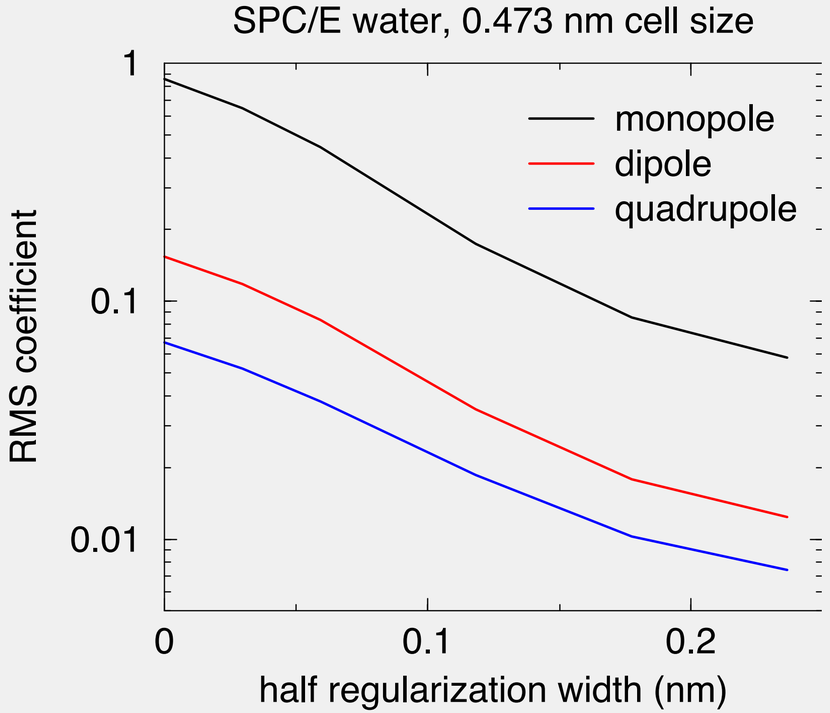
multipole coefficients,log10



local expansion coefficients, log10



Real water shows the same decrease of coefficients



Computational cost of regularization

- 1/2 regularization width = 1/4 cell
- Second nearest neighbor direct
- Direct pair cost factor $(2.25/2)^3 = 1.42$
  - At 0.473 nm (10.6 atoms) cell **free** with LJ
- Multipole lowest level: factor $1.5^3 = 3.4$

- But we get factor 5 more accuracy with P=4
- And we get energy conservation!

Conclusions

- PME 3D-FFT is limiting scaling
- FMM has better communication characteristics
- FMM is a good fit for SIMD/GPU acceleration
- FMM regularization gives energy conservation
- FMM regularization improves accuracy
- FMM fits wells with GROMACS pair computation

- We are implementing optimized, regularised FMM

Is there a better threading/tasking model than OpenMP on the horizon?

Acknowledgements

Erik Lindahl
Mark Abraham
Szilard Pall
Aleksei Iupinov

Rio Yokota, Tokyo Tech
John Eblen, ORNL
Roland Schulz, Intel
Mark Berger, NVIDIA
rest of the GROMACS team world-wide

Partners

Funding

Horizon 2020 European Union Funding for Research & Innovation