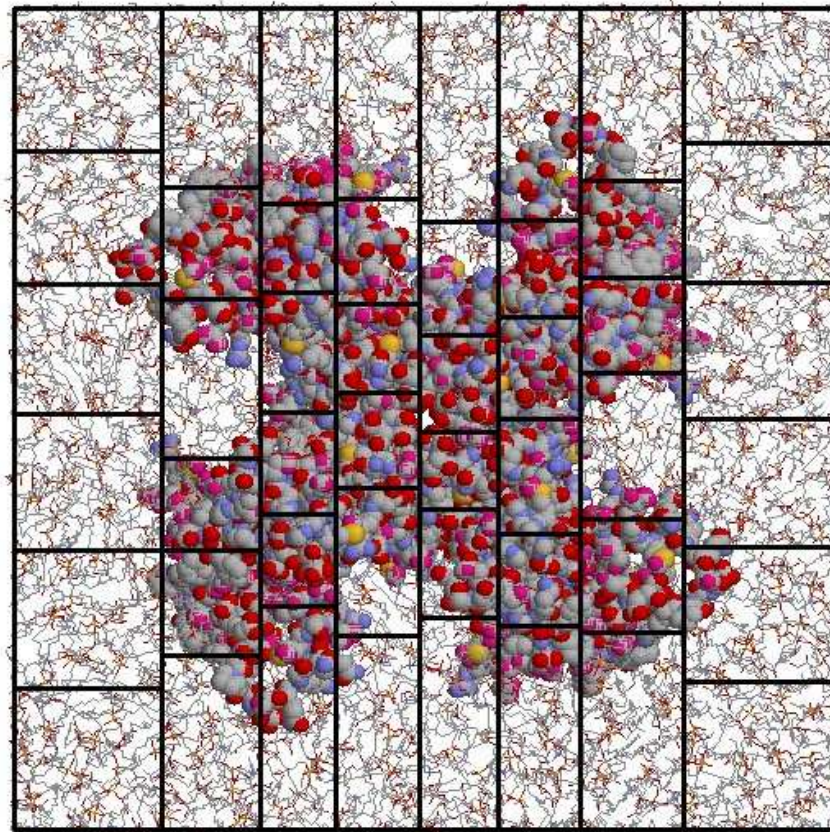


GROMACS: Fast, Free and Flexible MD



Berk Hess

Max Planck Institute for Polymer Research, Mainz



Domain decomposition

GROMACS 4: Algorithms for highly efficient, load-balanced,
and scalable molecular simulation

B. Hess, C. Kutzner, D. van der Spoel and E. Lindahl

J. Chem. Theory Comput. 4, 435 (2008)



New features in GROMACS 4.0

Huge improvement in parallel performance:

GROMACS 3.3: particle decomposition

GROMACS 4.0: also domain decomposition

Several other new features:

- heuristics neighbor list updates with a buffer region
- replica exchange with each replica in parallel
- automated A/B topology modification for free energy
- complete rewrite of the pull code (PMF, Umbrella sampling), now works in parallel
- full checkpointing support for binary exact continuation
- tabulated bonded interactions (coarse-graining)
- ...



Domain decomposition in practice

```
grompp
```

```
mpirun -np <np> mdrun
```



Domain decomposition in practice

```
grompp
```

```
mpirun -np <np> mdrun
```

At high parallelization one should optimize some mdp and mdrun options



Virtual interaction sites

Virtual interaction sites are sites without mass that are constructed from the coordinates of some of the masses in the system.

Forces working on them need to be redistributed to the masses.

Example: the oxygen charge site in the TIP4P water model

In GROMACS (nearly) all hydrogens can be replaced automatically by virtual interaction sites

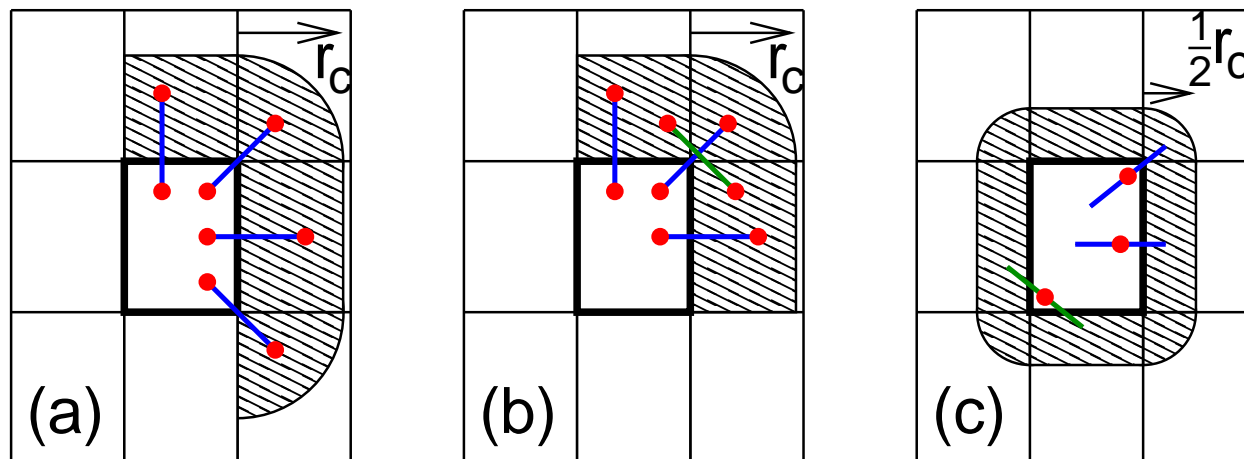
This allows the time step to be increased from 2 to 4 or 5 fs

A doubling of the simulation speed!

Feenstra et al., JCC 20, 327 (1999)



Domain decomposition methods



comm.	cut-off	Half Shell	Eighth Shell	Midpoint
#cells	$r_c < L_d$	13	7	26
	$r_c < 2L_d$	62	26	26
volume	$r_c = \frac{1}{2}L_d$	$2.94 L_d^3$	$2.15 L_d^3$	
	$r_c = L_d$	$9.81 L_d^3$	$5.88 L_d^3$	
	$r_c \rightarrow \infty$	$\frac{1}{2}$ sphere	$\frac{1}{8}$ sphere	

Eighth shell: Liem, S. Y.; Brown, D.; Clarke, J. H. R. *Comput. Phys. Commun.* 67(2), 261 (1991)

Midpoint: Bowers, Dror, Shaw, *JCP* 124, 184109 (2006)

The eighth shell method



Coordinate communication:

1D: pulse -x

2D: pulse -y, pulse -x

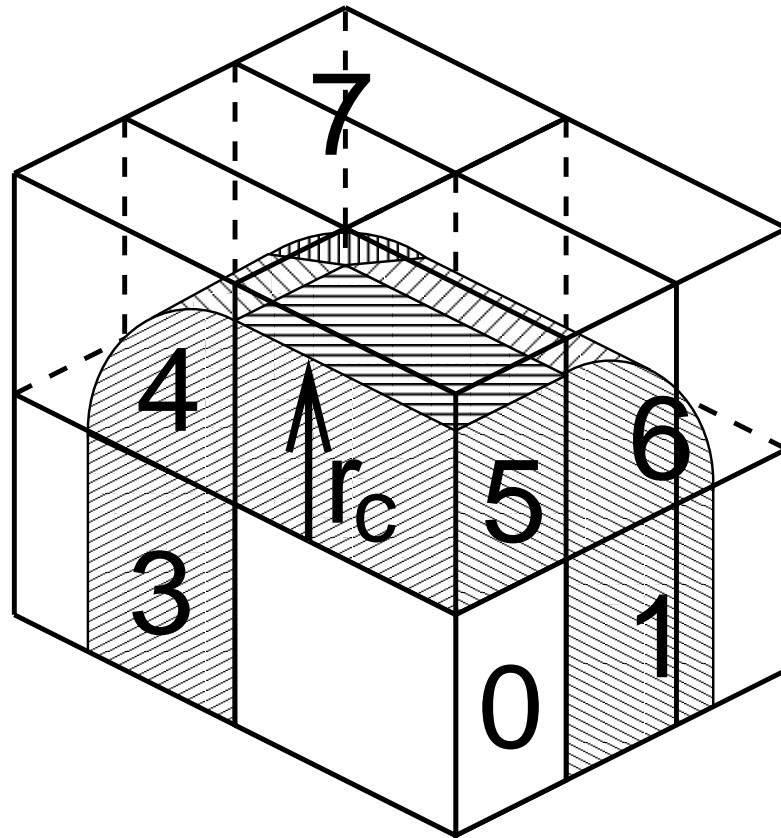
3D: pulse -z, pulse -y, pulse -x

Force communication:

1D: pulse x

2D: pulse x, pulse y

3D: pulse x, pulse y, pulse z





Full, dynamic load balancing

Load imbalance can occur due to three reasons:

- inhomogeneous particle distribution
- inhomogeneous interaction cost distribution (charged/uncharged, water/non-water due to GROMACS water innerloops)
- statistical fluctuation (only with small particle numbers)



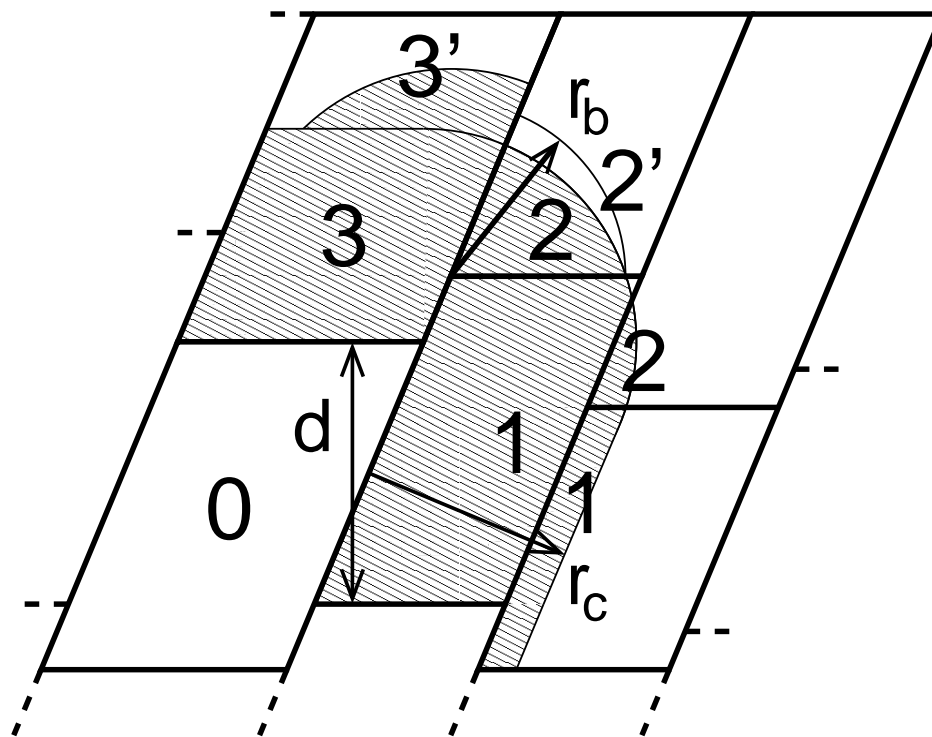
Full, dynamic load balancing

Load imbalance can occur due to three reasons:

- inhomogeneous particle distribution
- inhomogeneous interaction cost distribution (charged/uncharged, water/non-water due to GROMACS water innerloops)
- statistical fluctuation (only with small particle numbers)

So we need a dynamic load balancing algorithm where the volume of each domain decomposition cell can be adjusted **independently**

Triclinic unit cells with load balancing





Automatic dynamic load balancing

The dynamic load balancing costs a bit of performance, mainly due the extra load and cell boundary communication

mdrun option -dlb:

auto	turn DLB on automatically at $> 5\%$ performance loss
no	no DLB
yes	DLB always on

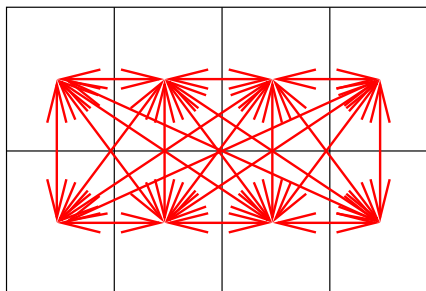


Multiple-program, multiple data PME

PME requires a lot of communication:
4 times per step
communication from all to all nodes

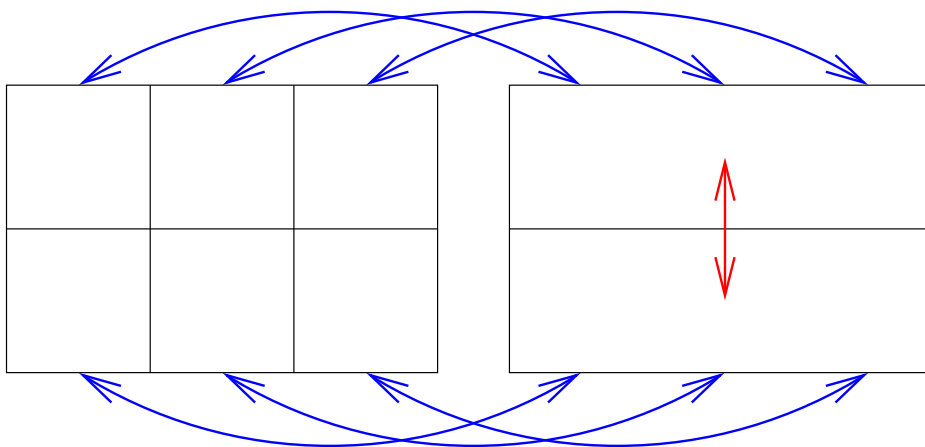
In GROMACS 4.0
the number of PME nodes is reduced by typically a factor of 4:
 $4^2 = 16$ times less communication calls

8 PP/PME nodes



6 PP nodes

2 PME nodes





Domain decomposition setup

The domain decomposition grid is optimized automatically by mdrun

The PP / PME node ratio is guessed from the particle density and the PME grid dimensions.

For low parallelization this is fine

For high parallelization the user should optimize #PME nodes

Clock cycle counts are printed at the end of the log file



Constraints in parallel

The time step in MD is limit by the fastest vibration, these are usually the bond vibrations

Removing bond vibrations allow for a larger time step:

H-? only constrained $\Delta t \approx 1$ fs

all bonds constrained $\Delta t \approx 2$ fs

all bonds constrained + v-sites $\Delta t \approx 5$ fs

Currently no MD code supports constraints over CPU boundaries:

- Use H-? constraints only: $\Delta t = 1$ fs (most codes)
- Molecule decompostion: can not split proteins

Most MD codes use SHAKE which does not parallelize easily
LINCS (**L**inear **C**onstraint **S**olver) does parallelize easily!

LINCS: Hess et al., JCC 18, 1463 (1997)

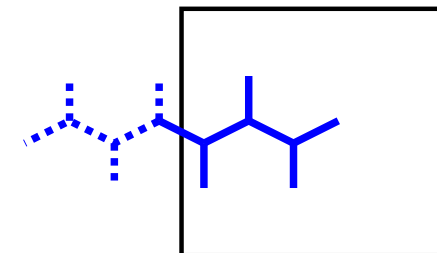
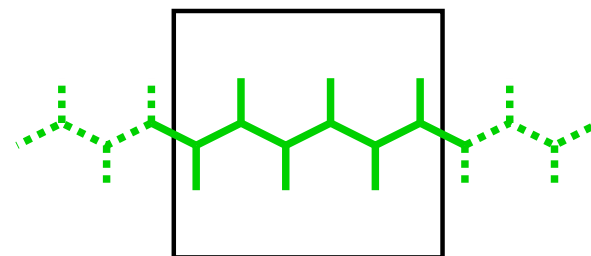
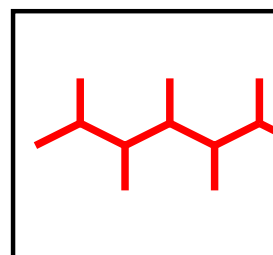
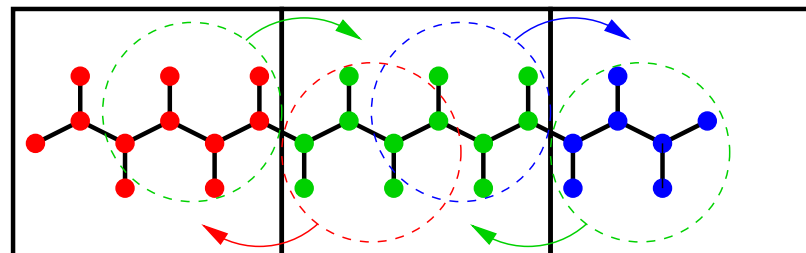


LINCS in parallel: P-LINCS

LINCS transforms the non-linear constraint problem into an iteration of linear, sparse-matrix, equations

LINCS usually requires:
1 initial step
1 iteration

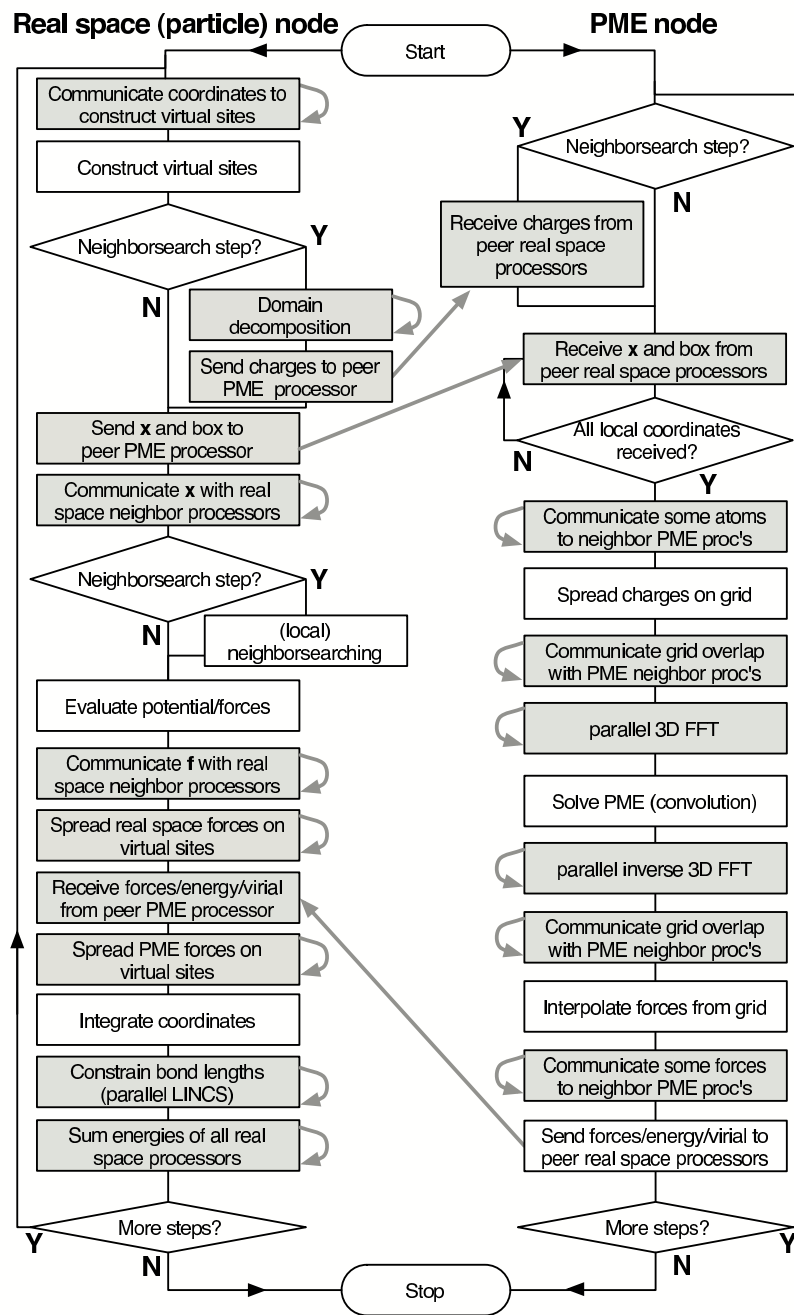
For P-LINCS this means:
2 communication steps



P-LINCS: Hess, JCTC 4, 116 (2008)



Flow chart



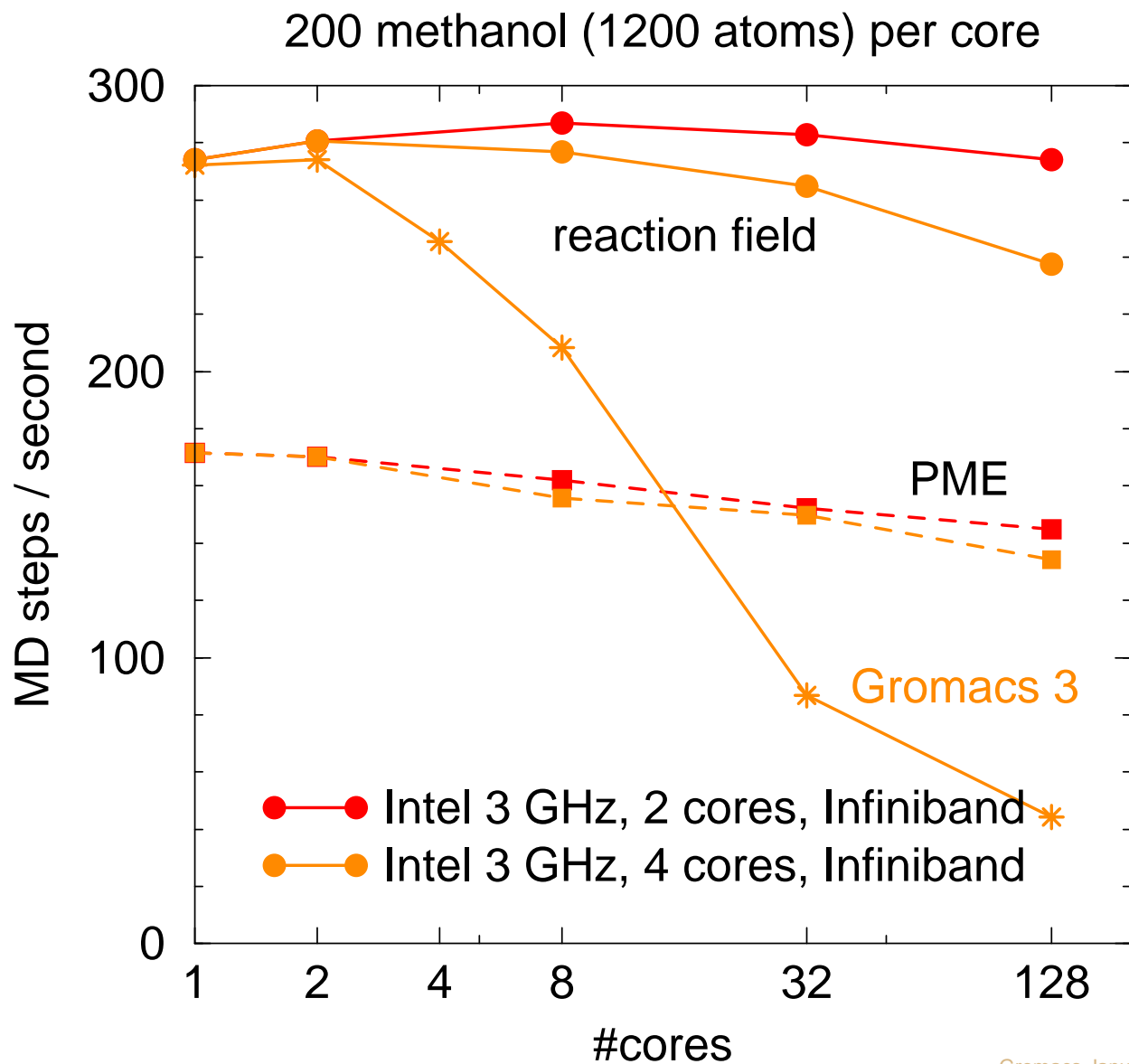


'Weak' scaling

methanol
OPLS/AA
all-atom model

cut-off 1.0 nm

PME grid
spacing:
0.12 nm





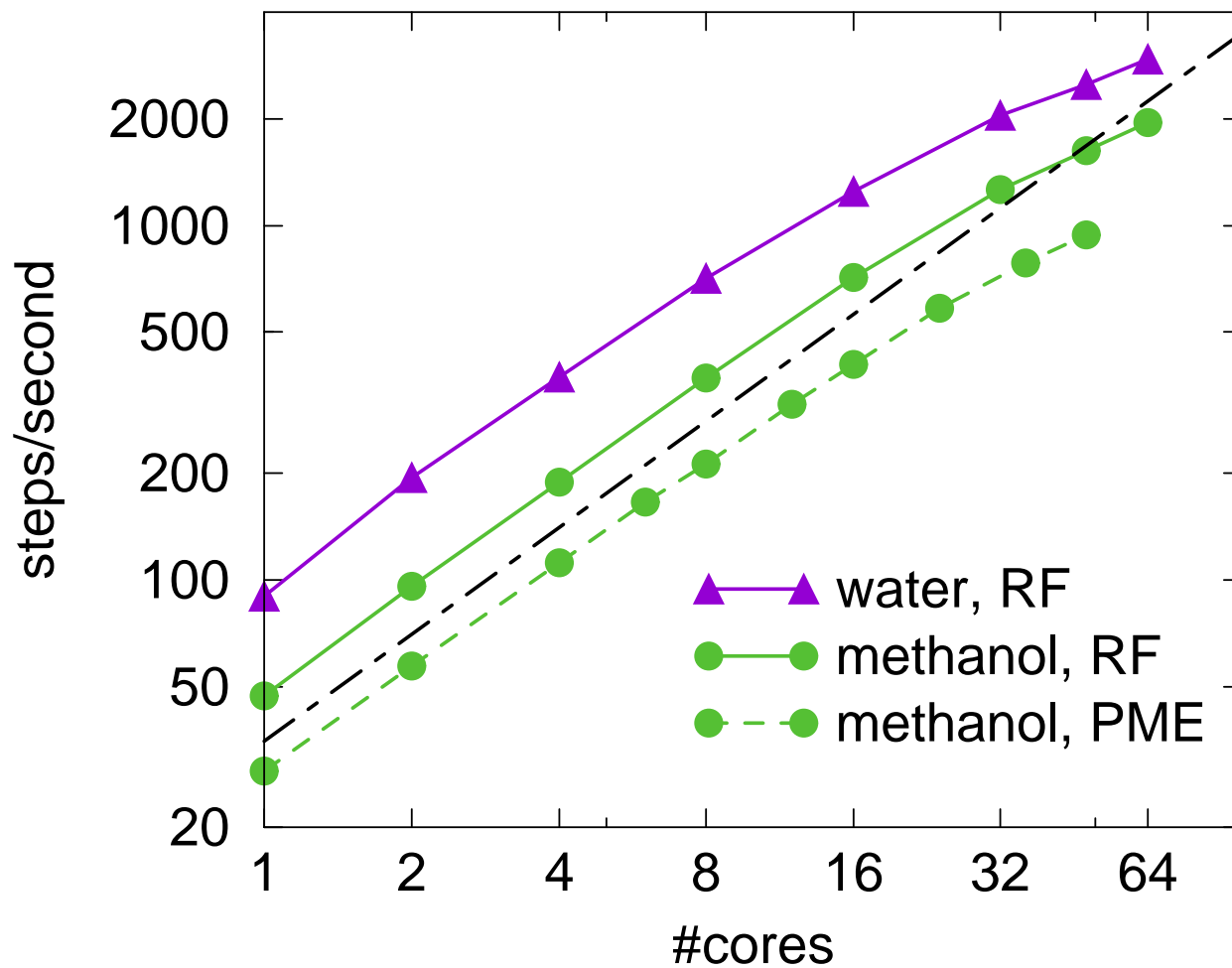
'Strong' scaling

SPC/E water

methanol
OPLS/AA

cut-off 1.0 nm

PME
grid spacing:
0.12 nm





Coarse-grained simulations

GROMACS 3.3 already had **tabulated non-bonded** potentials

New for GROMACS 4.0:

tabulated bonds, angles and dihedrals

All tabulated interactions use cubic spline interpolation:
consistent and continuous potentials and forces

Coarse-grained simulations put a much higher demand on the parallelization, because of the small number of interactions per particle



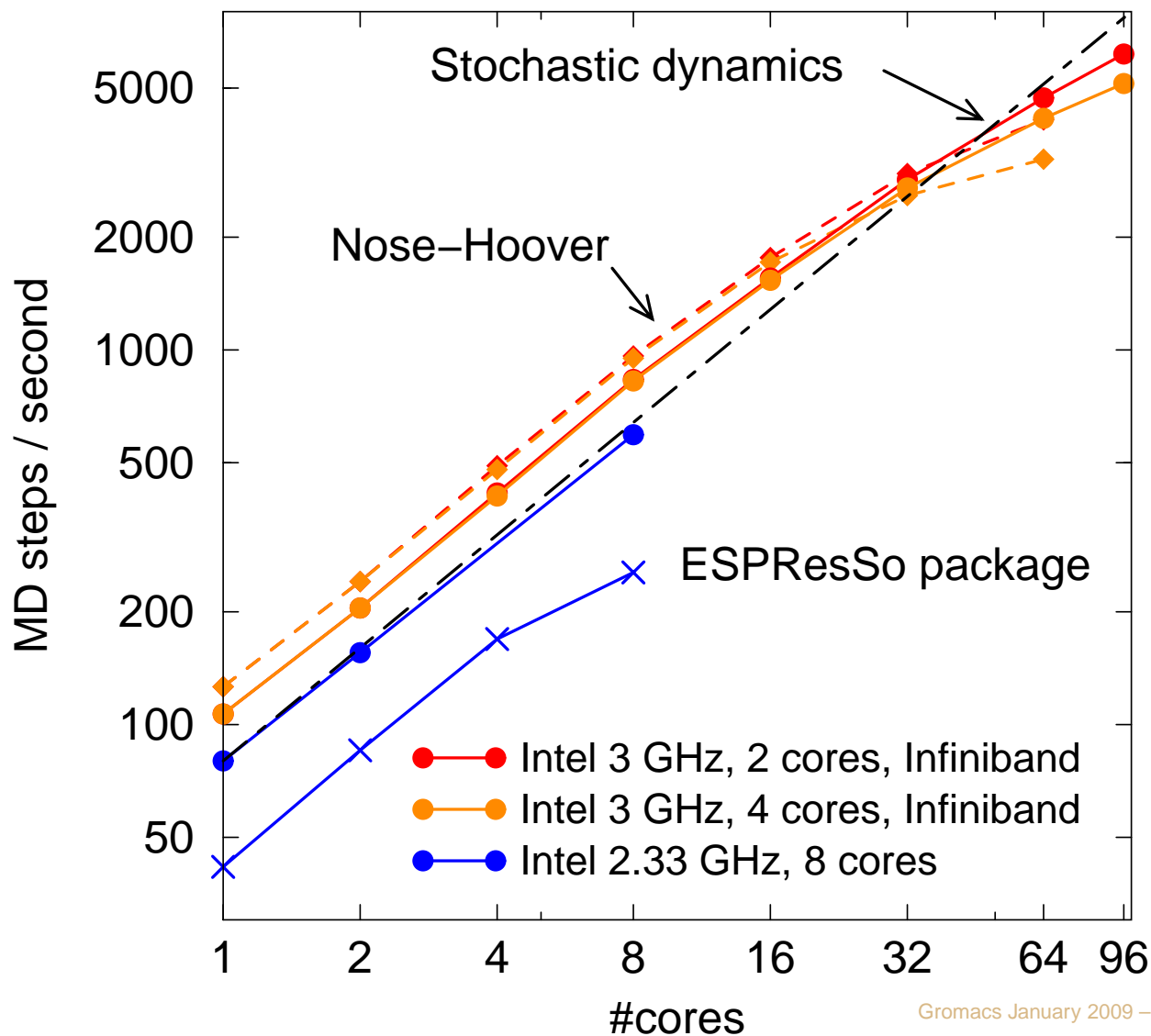
Coarse-grained polystyrene

one sidechain bead
one backbone bead
tabulated bonds, angles and dihedrals

8:1 mapping
0.85 nm cut-off

⇒
11 non-bonded interactions per bead

50 chains of 192 beads: 9600 beads

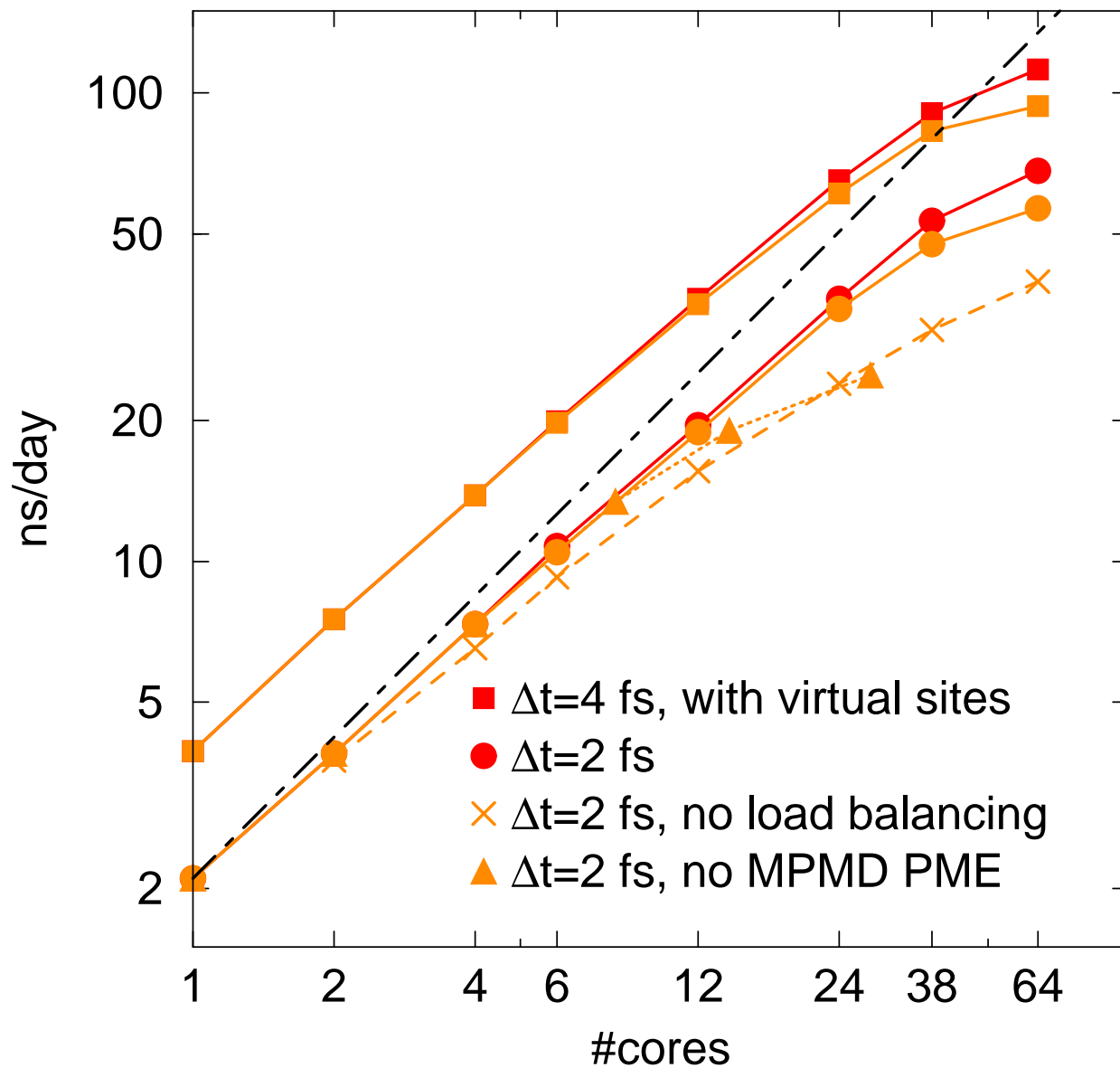




Lysozyme benchmark

T4-lysozyme
7156 SPC/E
8 Cl⁻
24119 atoms
dodecahedron 7 nm

cut-off 1 nm
PME
grid 56 × 56 × 56



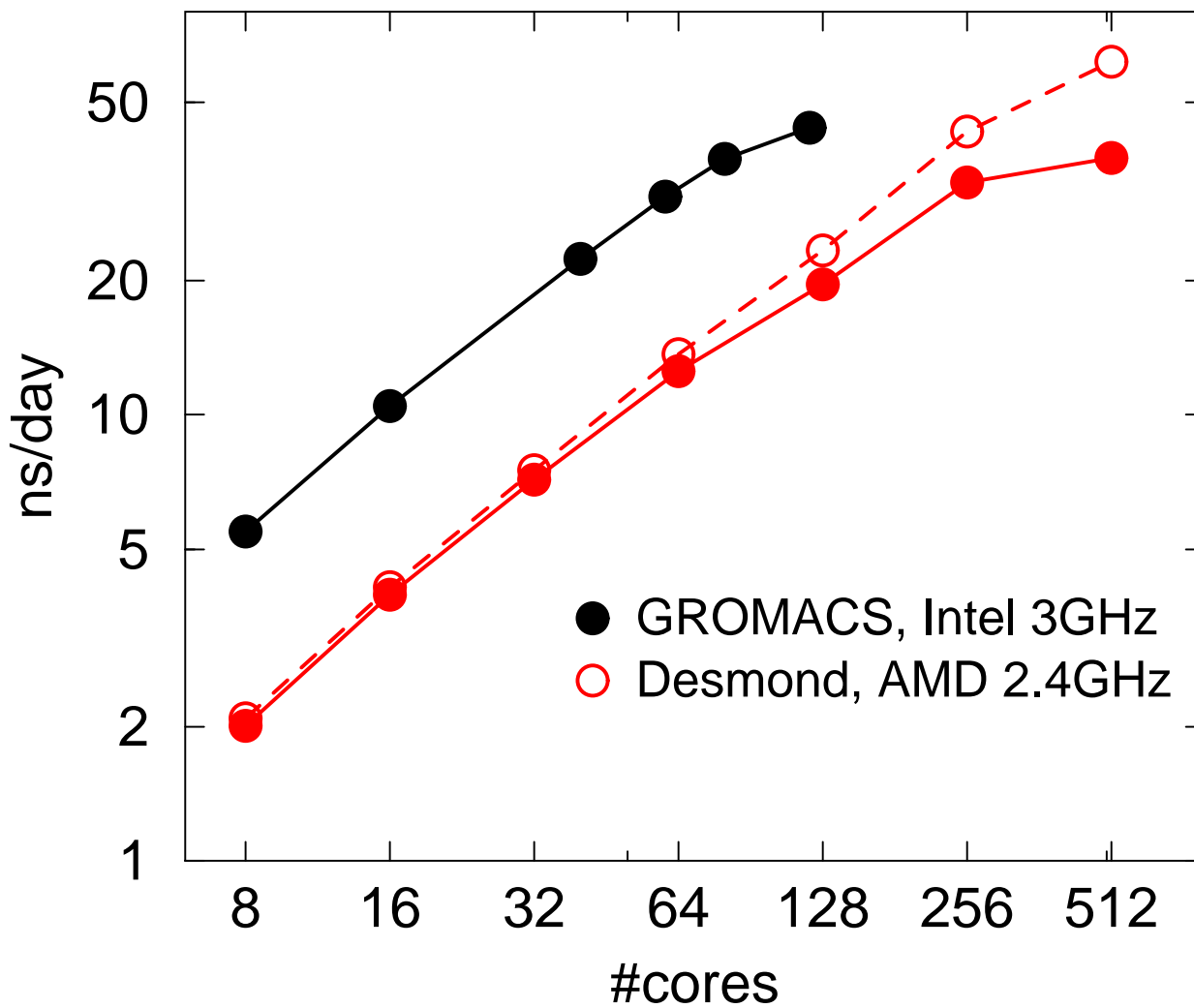


DHFR benchmark

DHFR
9011 TIP3P
23558 atoms
cubic box 6.2 nm

GROMACS:
cut-off 0.96 nm
PME
grid $60 \times 60 \times 60$
drift: $0.011 k_B T/\text{ns}$

Desmond, NAMD:
cut-off 0.9 nm
PME
grid $64 \times 64 \times 64$
drift: $0.004 k_B T/\text{ns}$



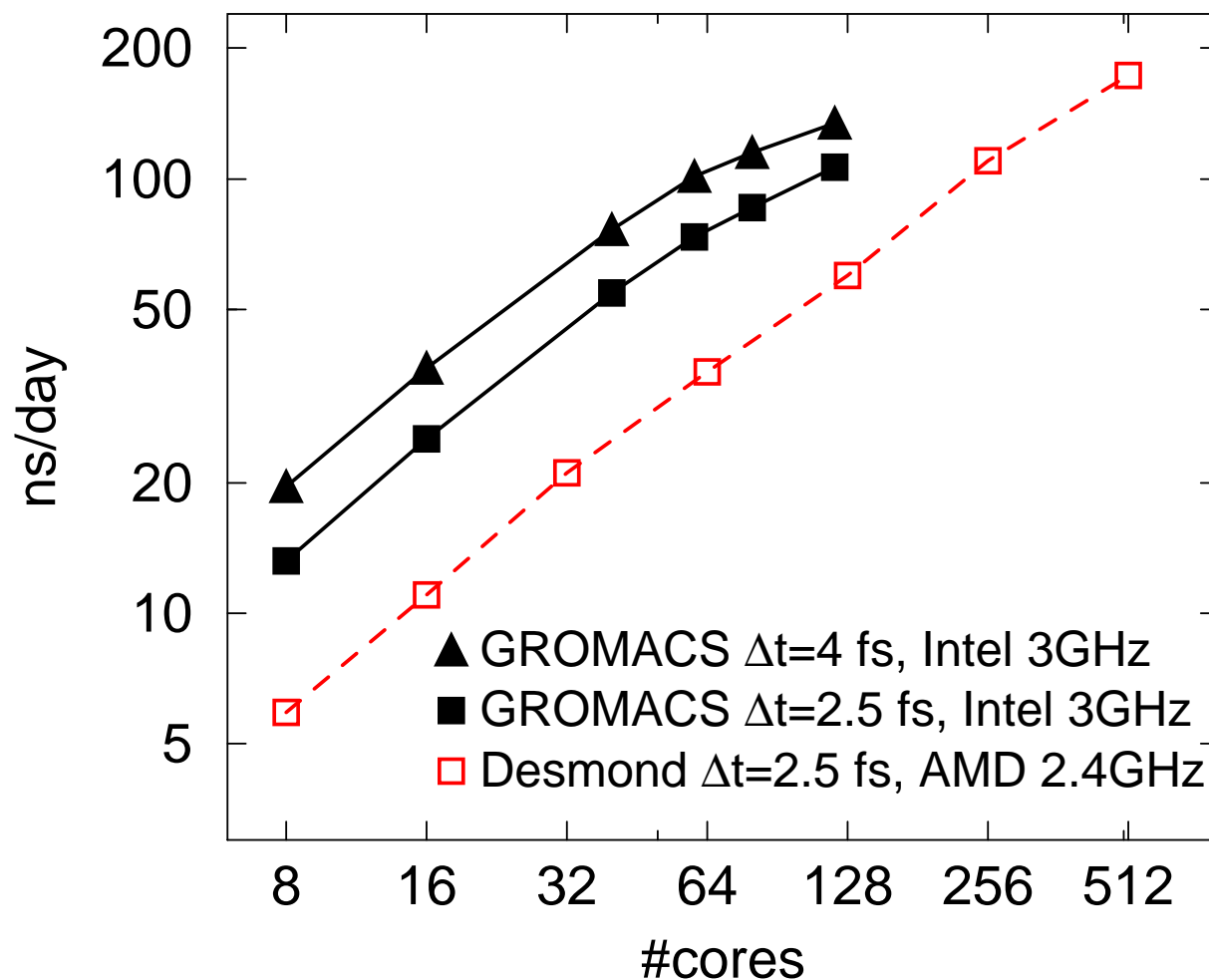


DHFR benchmark

DHFR
9011 TIP3P
23558 atoms
cubic box 6.2 nm

GROMACS:
cut-off 0.96 nm
PME
grid $60 \times 60 \times 60$
drift: $0.01 k_B T/\text{ns}$

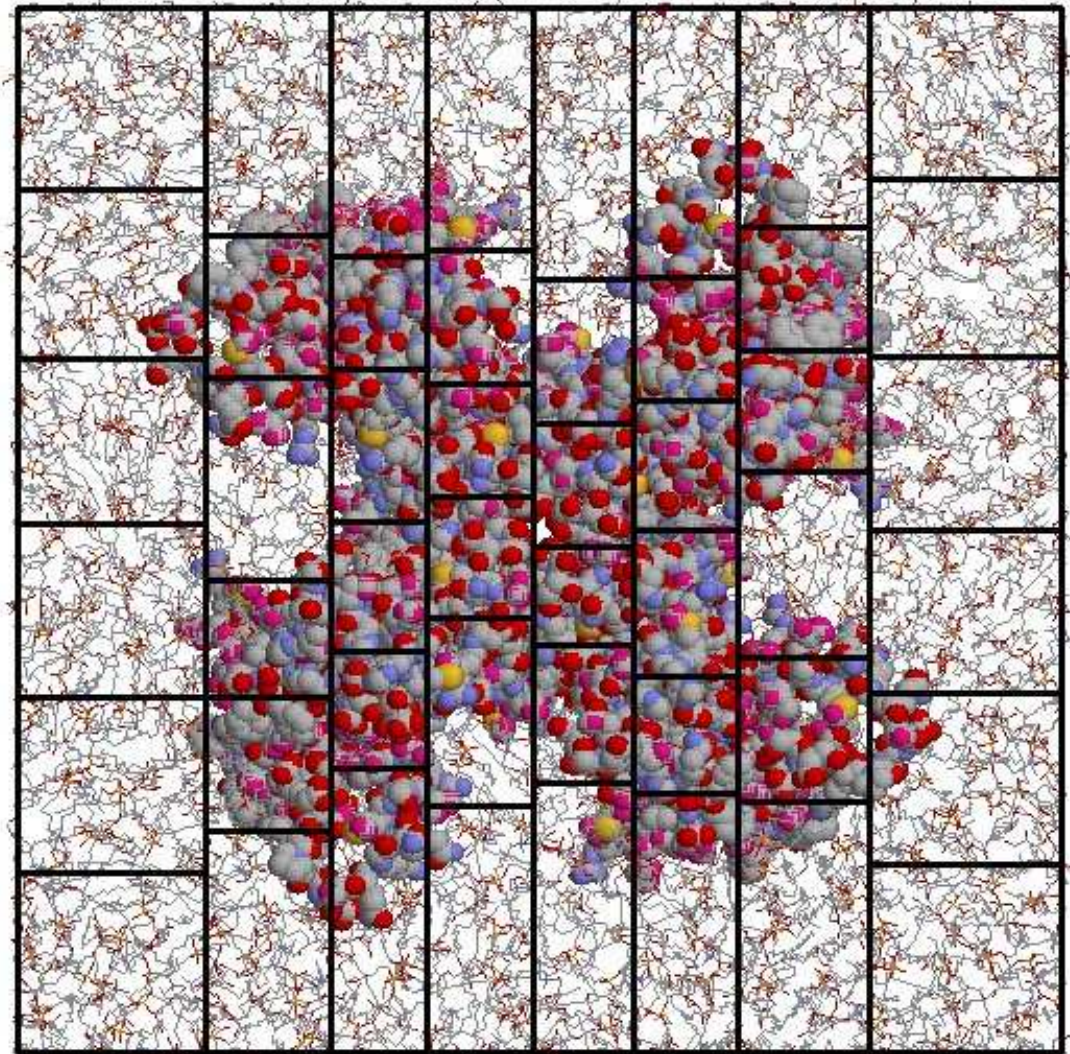
Desmond:
cut-off 0.9 nm
PME
grid $64 \times 64 \times 64$
drift: $0.004 k_B T/\text{ns}$



Membrane protein benchmark



Kv12 ion channel
(1040 res)
424 lipids
27017 waters
119857 atoms
13 × 13 × 9 nm

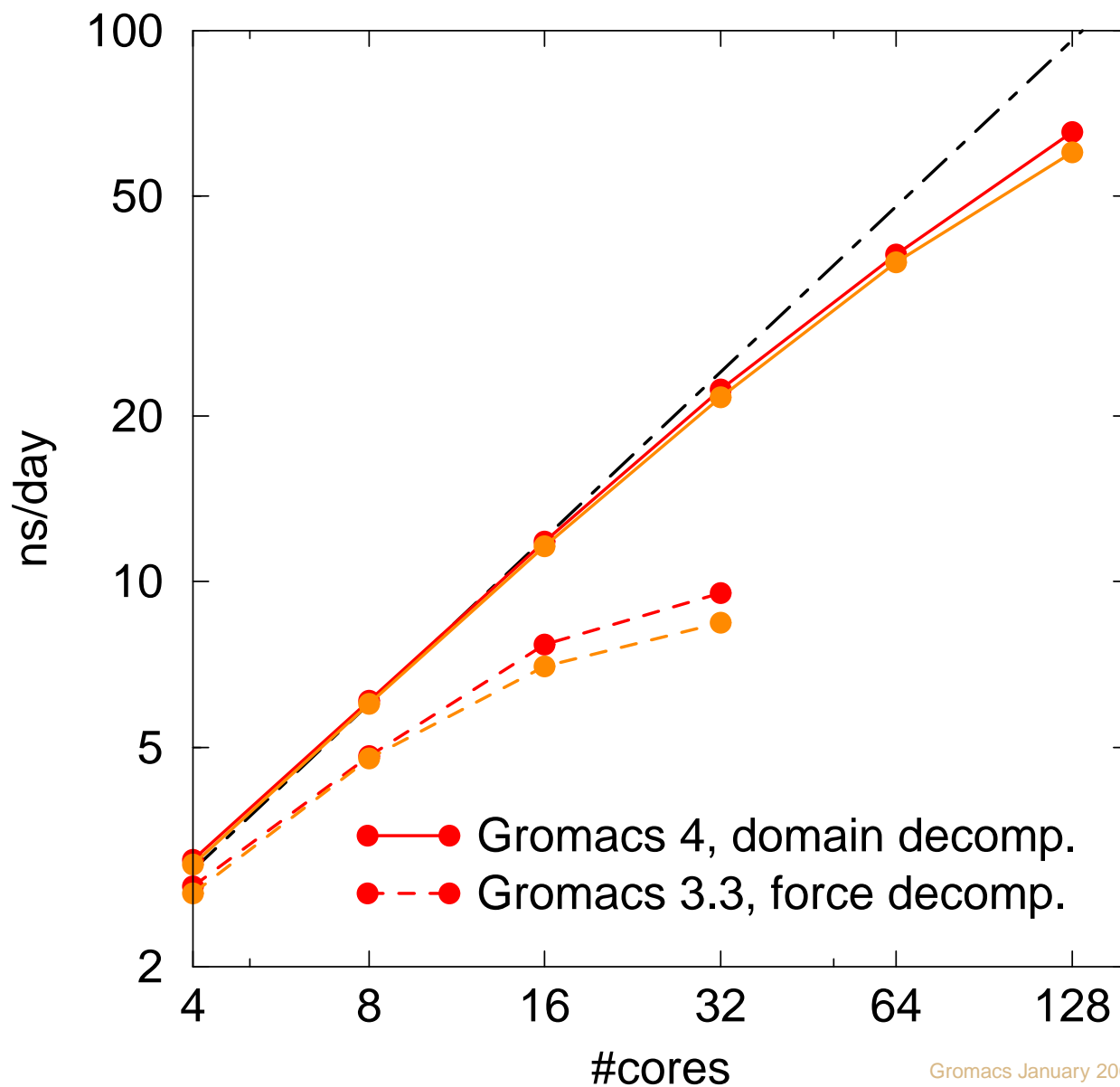




Membrane protein benchmark

Kv1.2 ion channel
(1040 res)
424 lipids 1024
lipids
27017 waters
11913 atoms
13 × 13 × 9 nm

cut-off 1.1 nm
PME
grid 96 × 96 × 64





mdrun memory usage

Resident memory usage per process proportional to:
 $\#atoms / \#processes$

Except for the master process:

$\approx \#atoms \times 50 \text{ bytes}$ (10 times less than GROMACS 3.3)

All processes:

$\approx (100 \text{ bytes} + N_{\text{atoms in cut-off}} \times 4 \text{ bytes}) * \#atoms / \#processes$



mdrun memory usage

Resident memory usage per process proportional to:
 $\#atoms / \#processes$

Except for the master process:

$\approx \#atoms \times 50 \text{ bytes}$ (10 times less than GROMACS 3.3)

All processes:

$\approx (100 \text{ bytes} + N_{\text{atoms in cut-off}} \times 4 \text{ bytes}) * \#atoms / \#processes$

Example system: 10 million atoms, 100 charge groups in cut-off:

master process: 500 MB

all processes together: 5 GB

On 8 8-core nodes (64 cores):

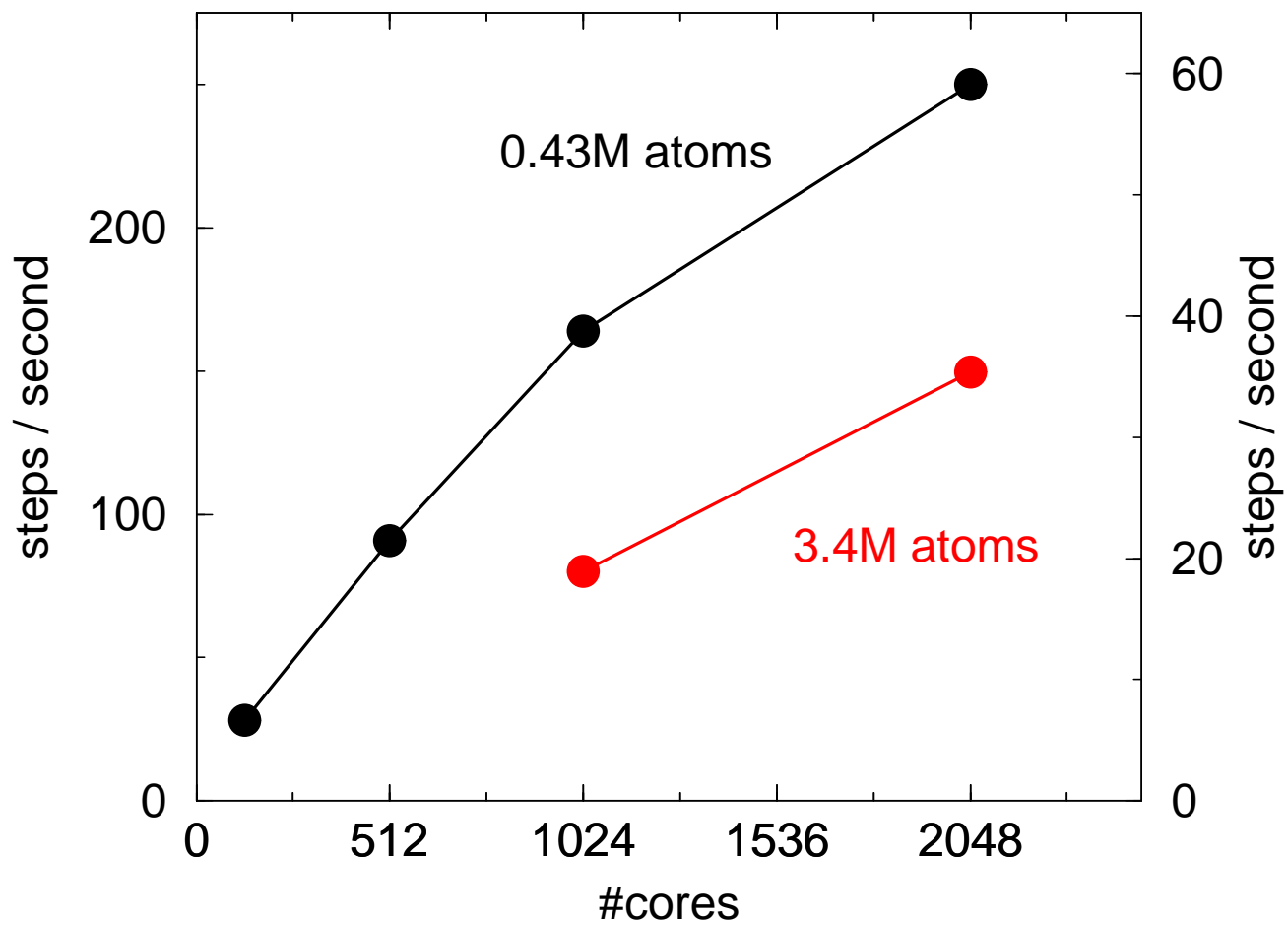
master node: 1.1 GB

other nodes: 0.6 GB



LJ system on IBM Blue Gene

`rlist = 2.6 σ_{LJ}`





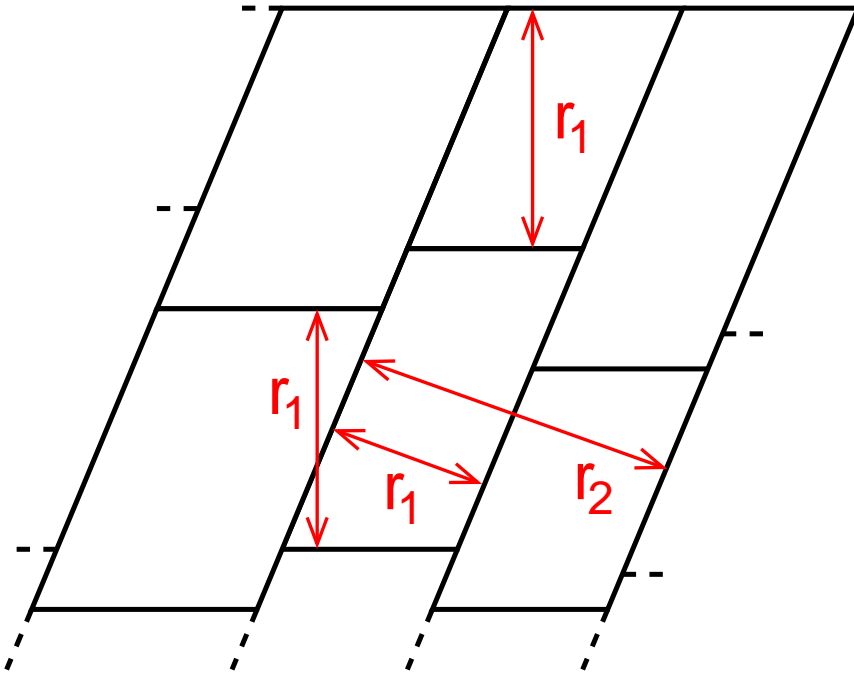
GROMACS 4.0 conclusions

- Slightly better single processor performance
- Two orders of magnitude better scaling
- MPMD PME: good PME scaling
- Better performance than (all) other simulation packages
- (Nearly) all algorithms work with domain decomposition

Scaling limits:

- With PME: PME
solution: 2D PME decomposition
- Without PME: energy summation (global communication)
solution: `mdrun -nosum`

Domain Decomposition limitations



Limitations:

non-bonded

$$r_{ij} \leq r_n$$

bonded 2-atom

$$r_{ij} \leq r_n$$

bonded n -atom

$$r_{ij} \leq r_1$$

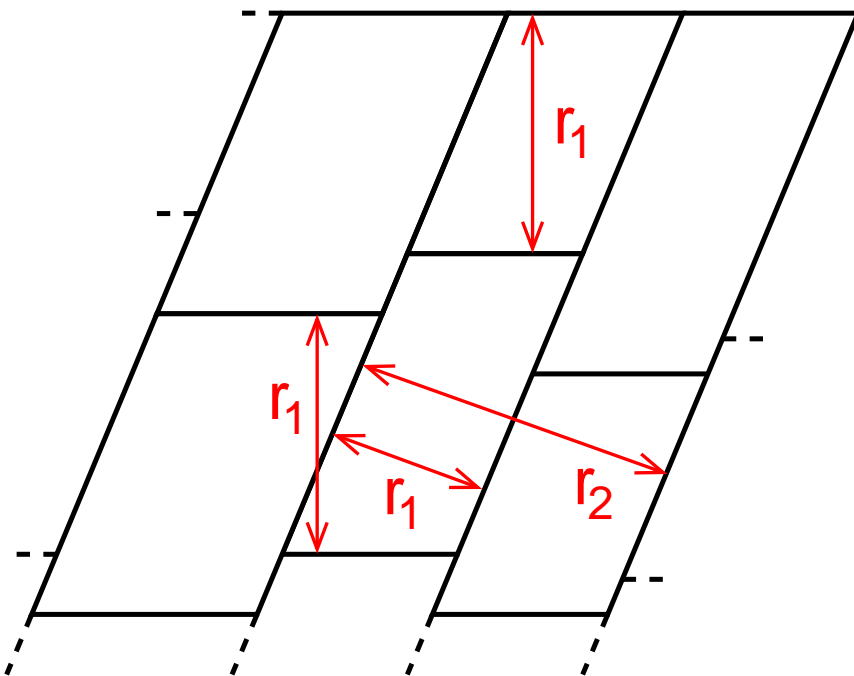
constraints

$$r_{ij} \leq r_1$$

virtual sites

$$r_{ij} \leq r_1$$

Domain Decomposition limitations



Limitations:

non-bonded

$$r_{ij} \leq r_n$$

bonded 2-atom

$$r_{ij} \leq r_n$$

bonded n -atom

$$r_{ij} \leq r_1$$

constraints

$$r_{ij} \leq r_1$$

virtual sites

$$r_{ij} \leq r_1$$

- limit for r_1 for bondeds determined by `mdrun x0` or `-rdd`
- limit for r_1 for constraints determined by `mdrun` or `-rcon`
- communication range n , no DLB: dynamic, DLB: set indirectly by `-dds`



Domain Decomposition limitations

At the beginning of the `md.log` file:

The maximum allowed distance for charge groups involved in interactions

non-bonded interactions		1.000 nm
two-body bonded interactions	(-rdd)	1.000 nm
multi-body bonded interactions	(-rdd)	1.000 nm
virtual site constructions	(-rcon)	2.477 nm
atoms separated by up to 5 constraints	(-rcon)	2.477 nm



Domain Decomposition setup

The optimal domain decomposition grid is determined automatically (but can also be set with option `-dd`)

The optimal number of PME-only nodes is guessed by `mdrun`

Low parallelization:

```
mpirun -np <nnodes> mdrun
```

High parallelization with PME:

```
mpirun -np <nnodes> mdrun -npme
```



Domain Decomposition setup

At the end of the `md.log` file:

DOMAIN DECOMPOSITION STATISTICS

av. #atoms communicated per step for force: 2 x 281957.7

av. #atoms communicated per step for vsites: 3 x 1940.9

av. #atoms communicated per step for LINCS: 2 x 33852.6

Average load imbalance: 6.4 %

Part of the total run time spent waiting due to load imbalance: 2.0 %

Steps where the load balancing was limited by `-rdd`, `-rcon` and/or `-dds`:
0 % 0 %

Average PME mesh/force load: 1.141

Part of the total run time spent waiting due to PP/PME imbalance: 6.5 %

NOTE: 6.5 % performance was lost because the PME nodes had more work to do than the PP nodes.

You might want to increase the number of PME nodes or increase the cut-off and the grid spacing.



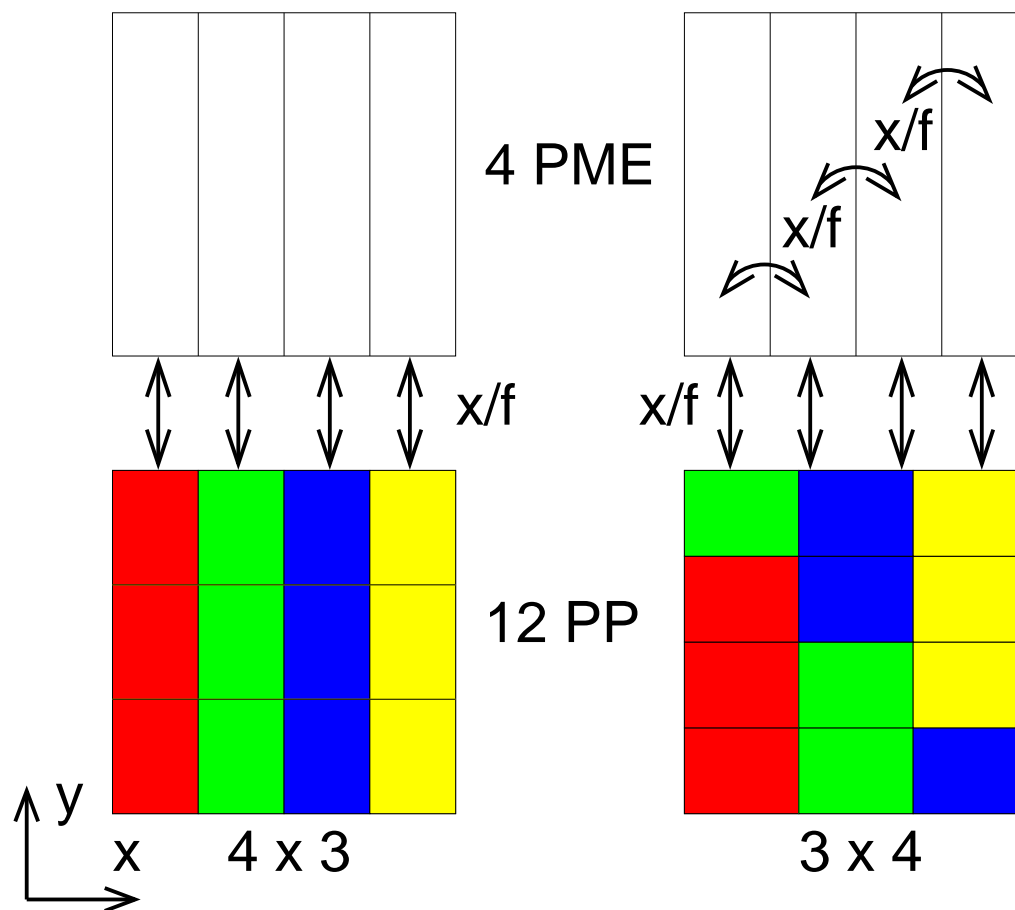
Timing output in md.log

R E A L C Y C L E A N D T I M E A C C O U N T I N G

Computing:	Nodes	Number	G-Cycles	Seconds	%
Domain decomp.	96	5334	4240.160	1421.1	4.7
Vsite constr.	96	32000	788.544	264.3	0.9
Send X to PME	96	32000	142.179	47.7	0.2
Comm. coord.	96	32000	2058.475	689.9	2.3
Neighbor search	96	5334	4377.657	1467.2	4.8
Force	96	32000	29216.823	9792.1	32.1
Wait + Comm. F	96	32000	8961.081	3003.3	9.9
PME mesh	32	32000	15331.173	5138.3	16.9
Wait + Comm. X/F	32	32000	7392.531	2477.6	8.1
Wait + Recv. PME F	96	32000	8144.996	2729.8	9.0
Vsite spread	96	64000	2245.412	752.6	2.5
Write traj.	96	1	3.457	1.2	0.0
Update	96	32000	595.352	199.5	0.7
Constraints	96	32000	5307.394	1778.8	5.8
Comm. energies	96	32000	1546.288	518.2	1.7
Rest	96		544.303	182.4	0.6
Total	128		90895.826	30464.0	100.0



PME communication



The PP grid setup is optimized by mdrun



PME parameters

The PME cost is order $N \log(N)$

But at high parallelization the communication becomes most costly

setup	r_c	grid spacing
“old standard”	0.85	0.120
single processor	1.00	0.135
100 PME nodes	1.20	0.160

For high parallelization, optimal PP:PME node ratio

rectangular box 3:1

rhombic dodecahedron 2:1

The PME grid dimensions no longer need to be divisible by #PME nodes, but it is advisable for performance at high parallelization



Communication setup `-ddorder`

There are different communication setups (for PP and PP-PME) implemented in GROMACS 4 for different types of computers/clusters

For “normal” multi-core clusters: `-ddorder interleave` (default)

For torus communication (Blue Gene): `-ddorder Cartesian`

For certain other super computers: `-ddorder PP-PME`



Summary practicalities

Without PME:

- in most cases no user choices are required
- do NOT use primes for the number of nodes

With PME:

- choose appropriate cut-off and PME grid spacing
- small systems: no further choices required
- large systems: optimize `-npme`

At high parallelization:

- use `-nosum`
- for very inhomogeneous system change `-dds`