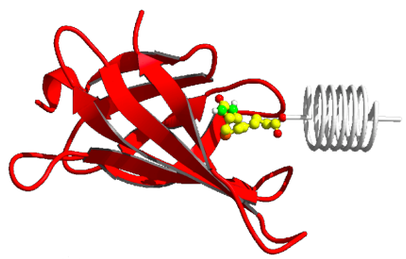
Improving PME on distributed computer systems

Carsten Kutzner

Max-Planck-Institut für Biophysikalische Chemie, Göttingen
Theoretical and Computational Biophysics Department



Outline

- how to get optimum performance from MD simulations with PME
  - Recap: Particle-Mesh-Ewald (PME)
  - parallel PME in GROMACS
  - understanding the PME parameters
  - tuning the performance of an example benchmark

Recap: PME

- N atoms at $\mathbf{r}_i$ with charges q_i in neutral box (PBC)
- electrostatic potential (conditionally convergent, s-l-o-w)

$$U = \frac{1}{2} \sum_{i,j=1}^{N} {\sum_{\vec{n} \in \mathbb{Z}^3}}' \frac{q_i q_j}{|\vec{r}_{ij} + \vec{n}L|}$$

- Coulomb: strong variation at small r, smooth at large r
- Ewald summation: split up U with

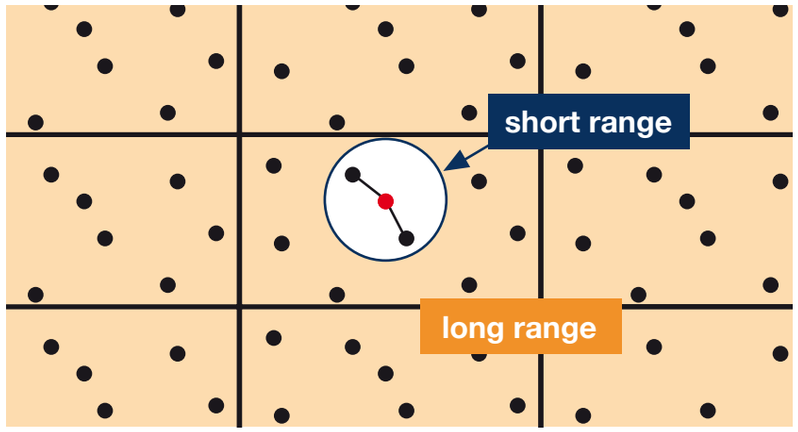
$$\frac{1}{r} = \frac{f(r)}{r} + \frac{1 \quad f(r)}{r}$$

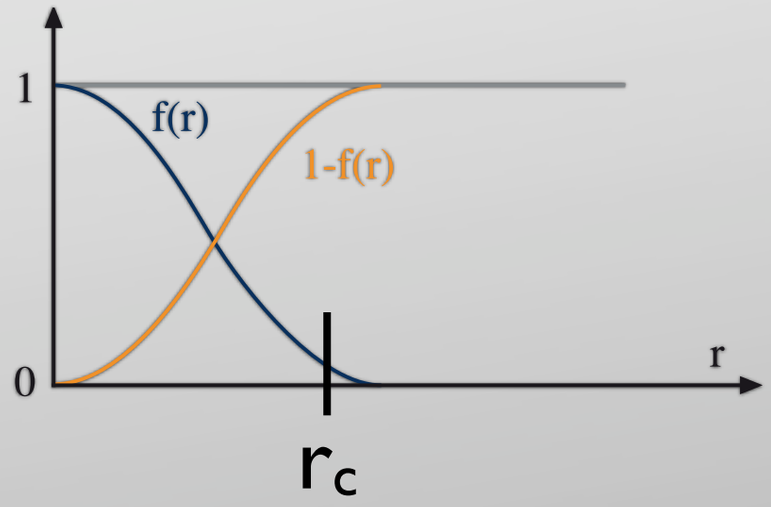
choose f = erfc(r)

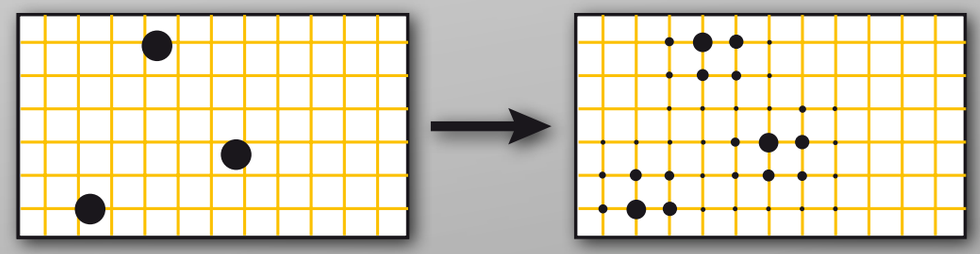
≈ 0 beyond cutoff (SR)

slowly varying func, needs only few wave vectors in Fourier space (LR)

- $\rho \rightarrow$ discrete mesh (DFT / FFT)







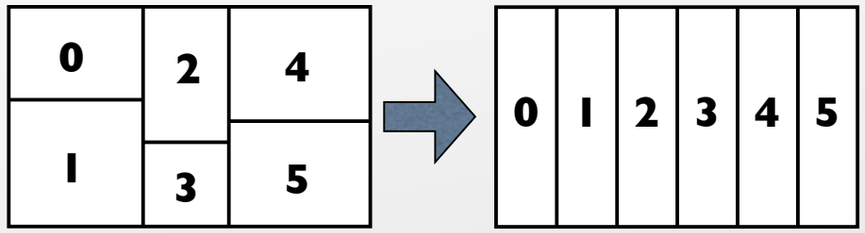
PME time step

1. SR part can be directly calculated

2. LR part: need FT charge density
    1. spread charges on grid: each charge is spread on pme_order grid points in x,y,z
    2. FFT to reciprocal space
    3. solve PME
    4. FFT back yields potential on grid points
    5. extrapolate forces on the particles from grid with splines of order pme_order

Parallel PME time step

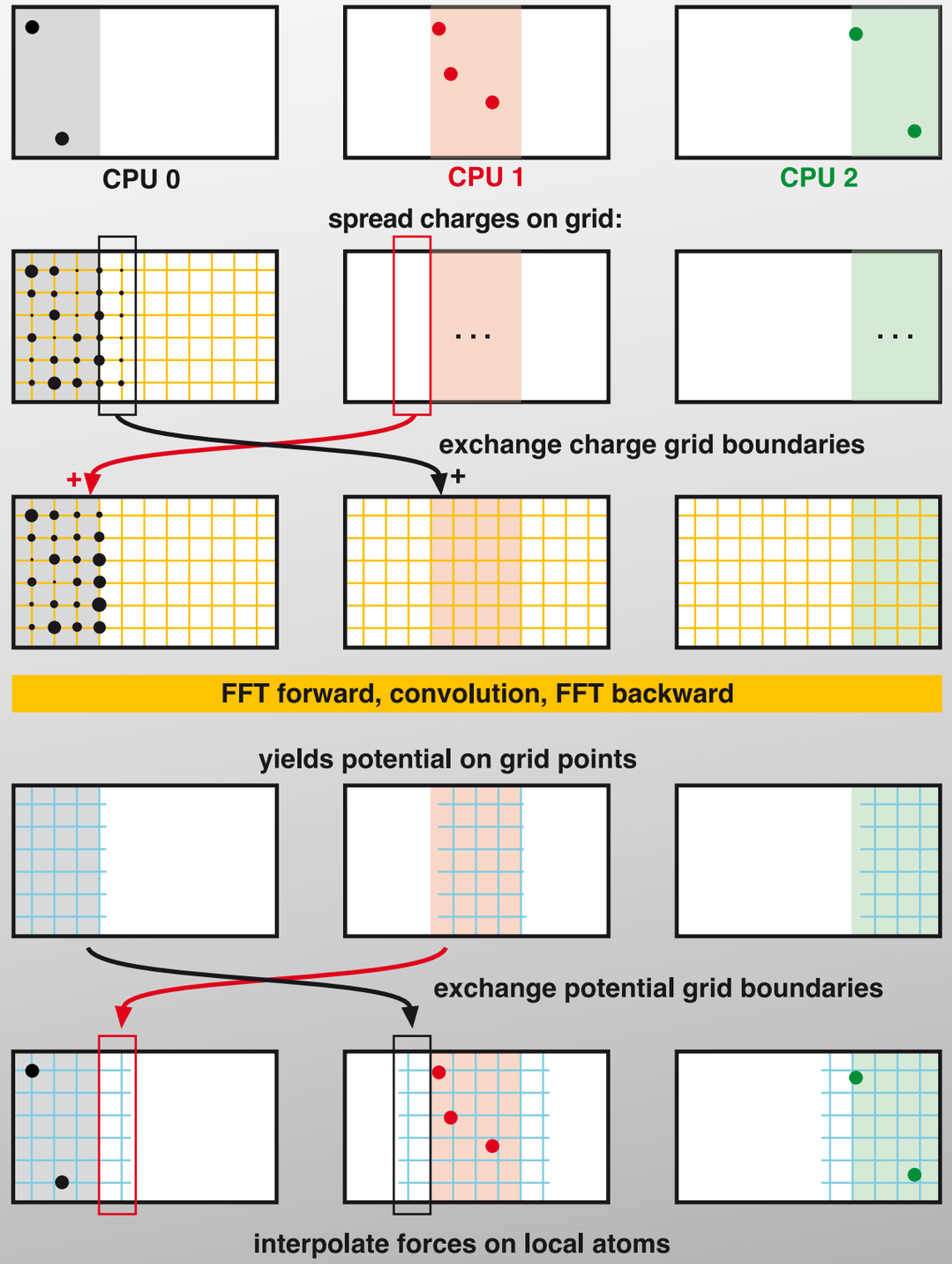
- particle redistribution
  xyz-domains → x-slabs



- communicate grid overlaps

- parallel data transpose within FFT
  needs all-to-all communication
  (2x to and from reciprocal space)

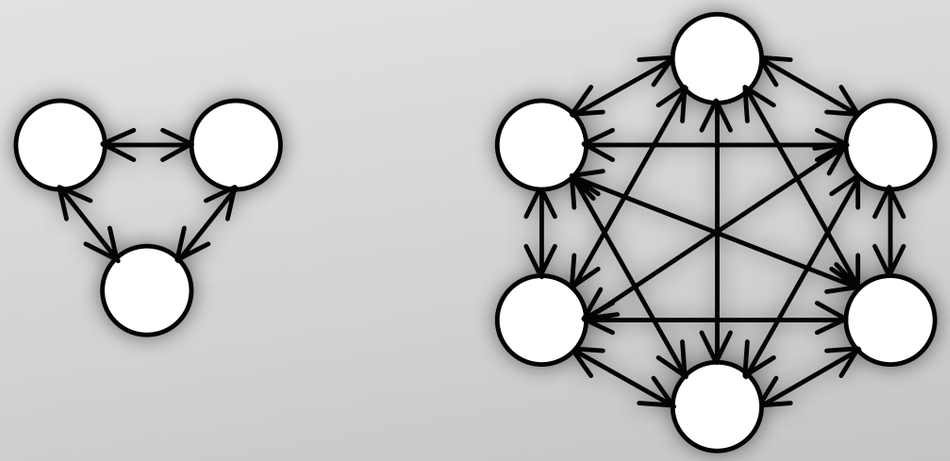
- force redistribution
  slabs → domains



Parallel PME: scaling bottlenecks

- all-to-all communication becomes increasingly expensive with N (latency increases by N^2)
- all-to-all might overload network,
  MPI_Alltoall performance = f(N, MPI lib, network type, ...)
- PME grid cannot always be distributed evenly among all processors:
  90 points → 64 CPUs?

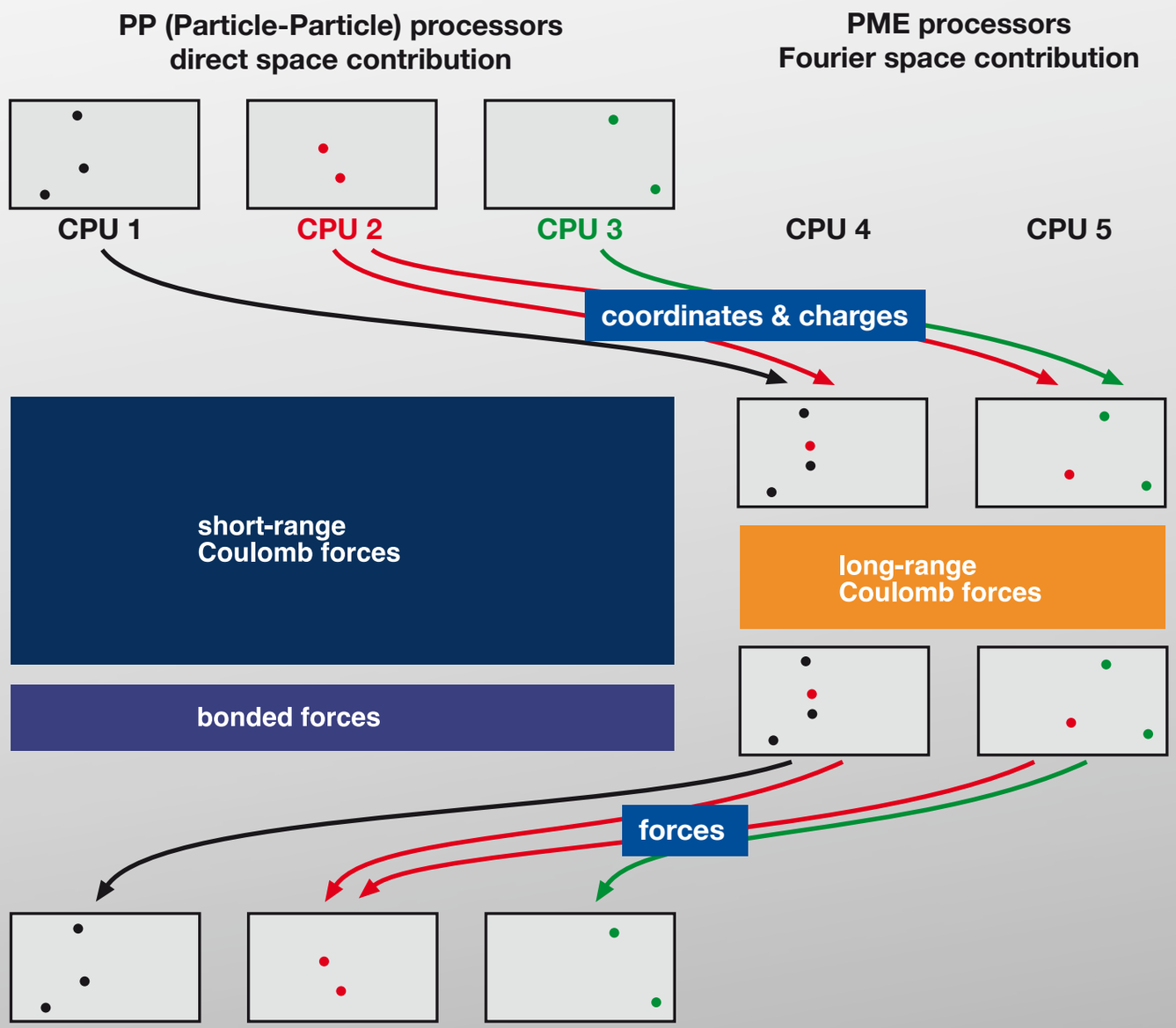
N•(N–1) messages



N=3 N=6

Multiple-Program Multiple-Data PME

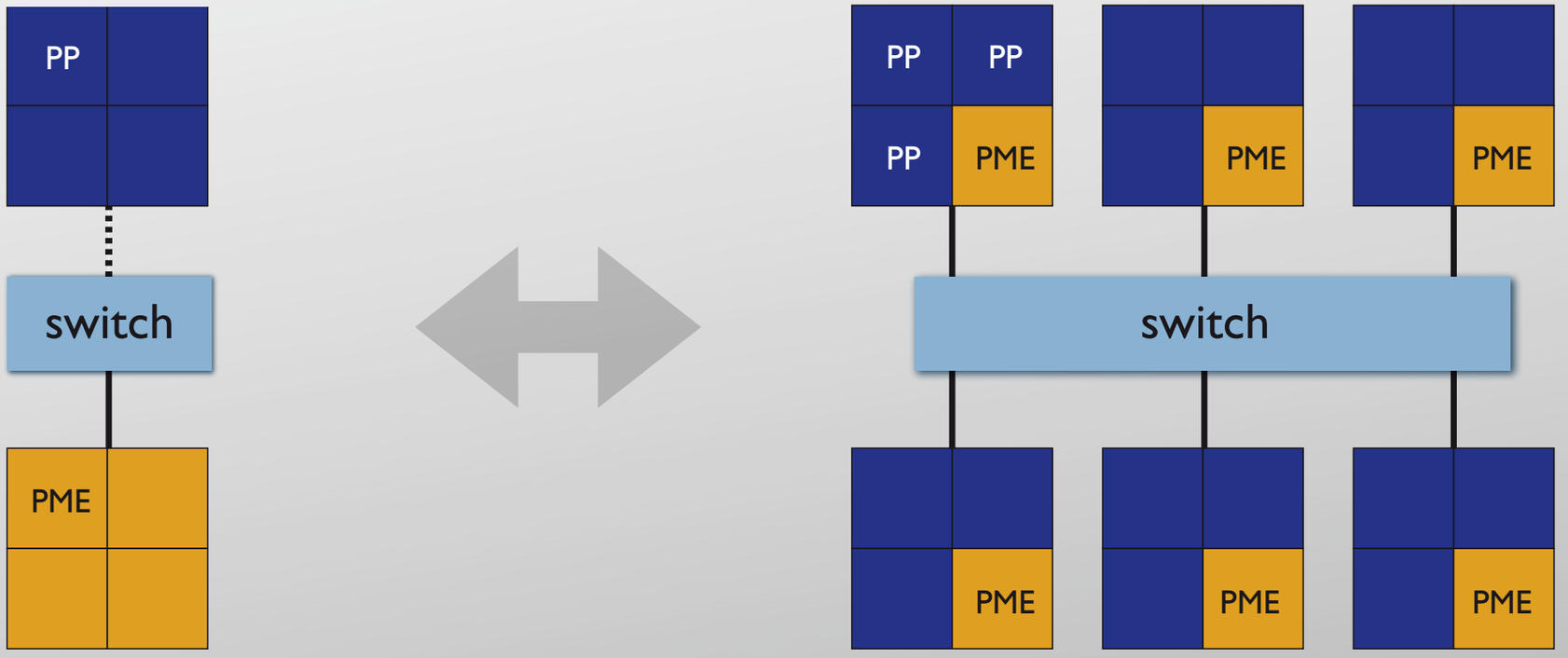
- direct and reciprocal space contributions can be computed independently
- assign a subset of the processors for recip. space with -npme
  `mpirun -np 5 mdrun -npme 2`



+ MPI_Alltoall on only 1/4 to 1/2 of the processors, reducing the communication to 1/16 to 1/4
+ better distribution of PME grid onto processors

Multiple-Program Multiple-Data PME

- today typically 2, 4, or 8 CPUs share a network interconnect
- PP/PME node interleaving
- higher bandwidth on multi-CPU nodes
- let the LR processors use all available network bandwidth




```
mpirun -np 8 mdrun -npme 4 -ddorder pp_pme
```

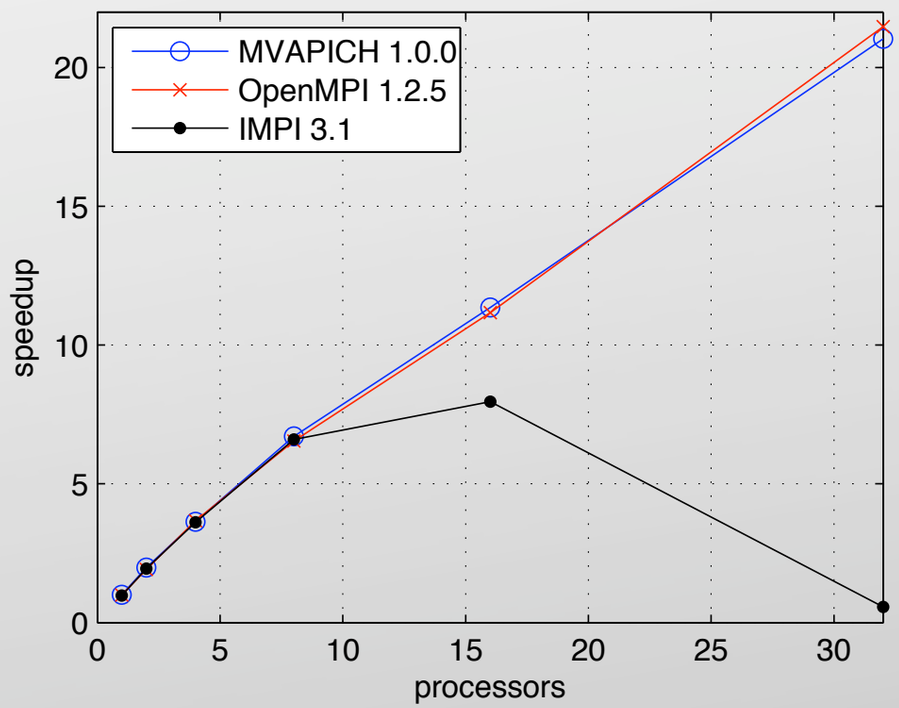
```
... -ddorder interleave
```

Knobs to turn

same accuracy:

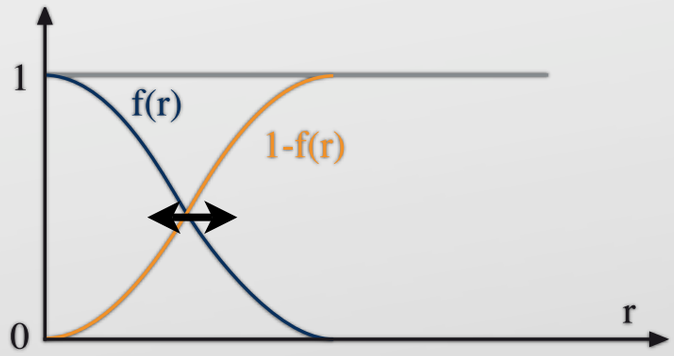
- node separation yes/no,
  # of PME nodes
- node order: PP_PME, interleaved,...
- DD grid, dlb yes/no

- compiler & MPI library!



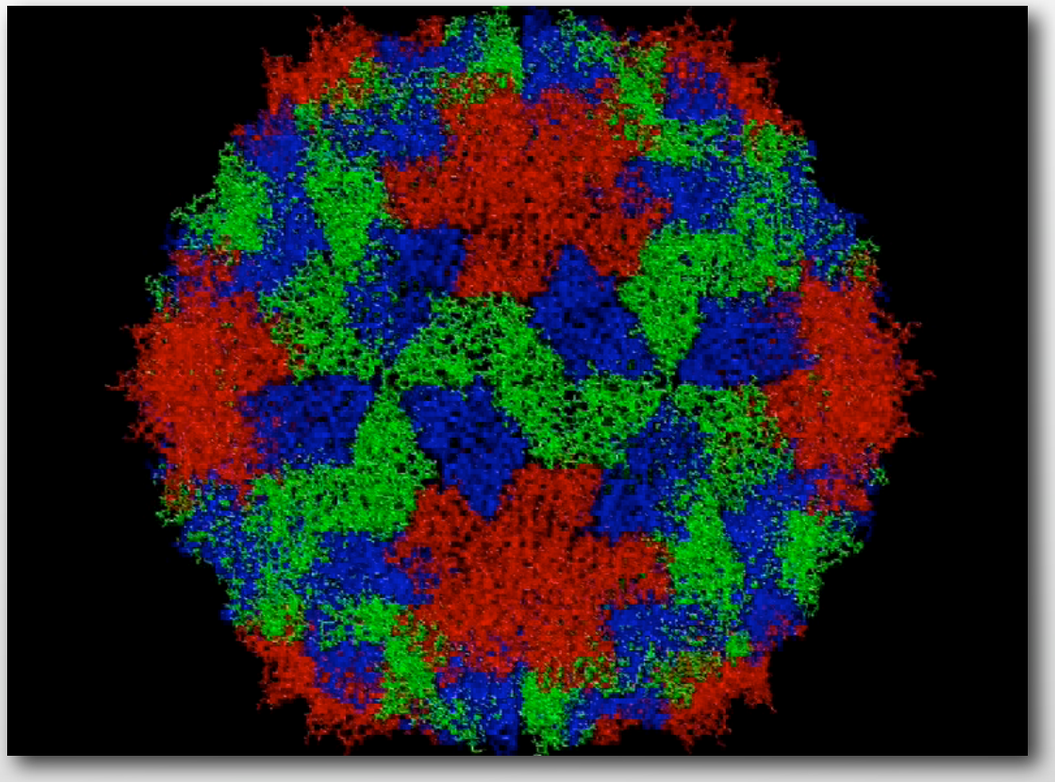
affects accuracy:

- PME grid points
- cutoff radius
  - to shift real/recip load @ same accuracy: multiply both r_c and fourierspacing by factor



- PME order 4, 6, 8, ...
  but we don't want to enlarge the grid overlap in the parallel case!

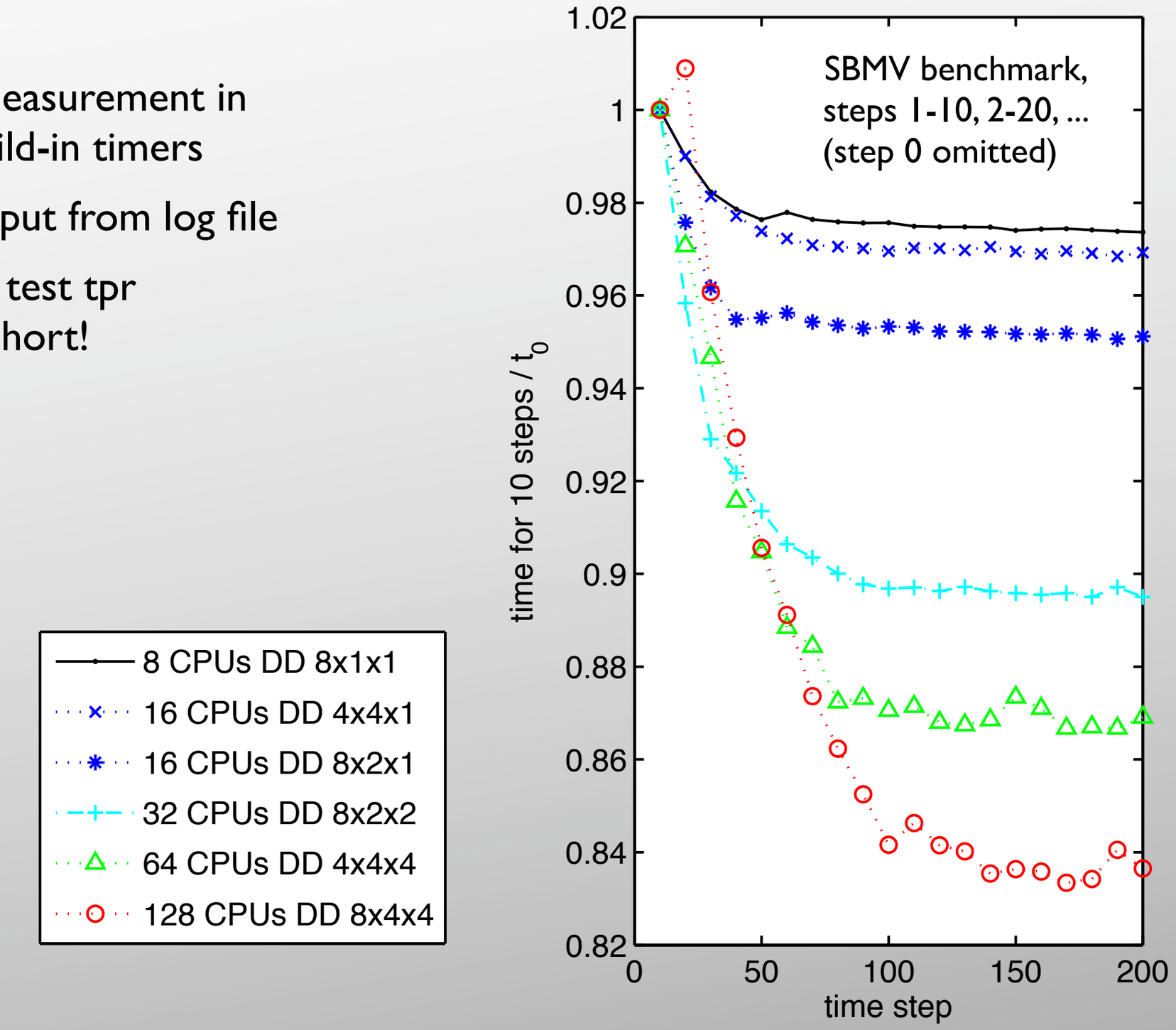
Example benchmark system



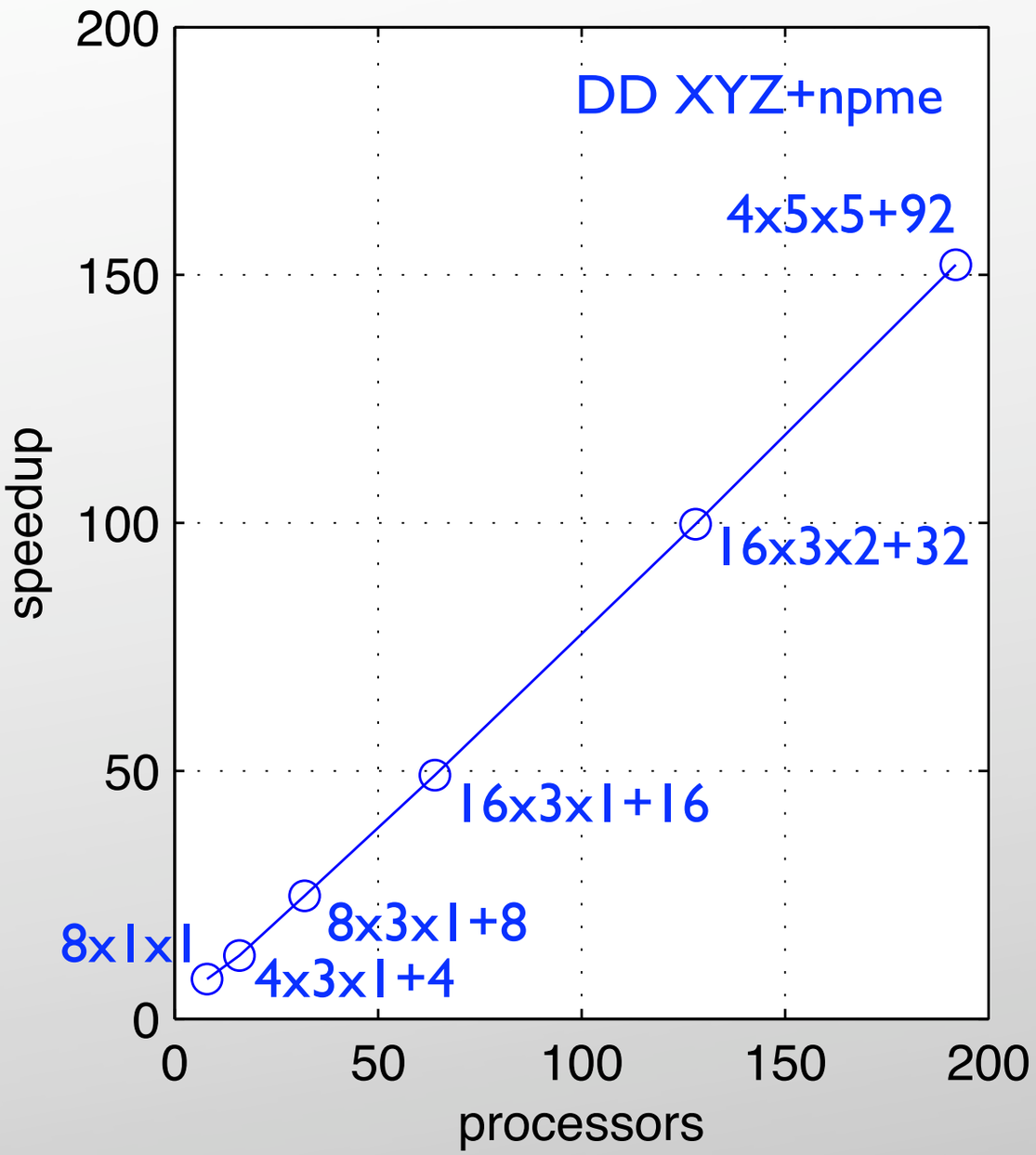
- SBMV in water, 4.5 M particles
- cutoffs 1 nm
- PME order 4, grid spacing 0.135 nm (275^3 grid)
- Intel 2.66 GHz, 8 cores / node
- Infiniband Interconnect

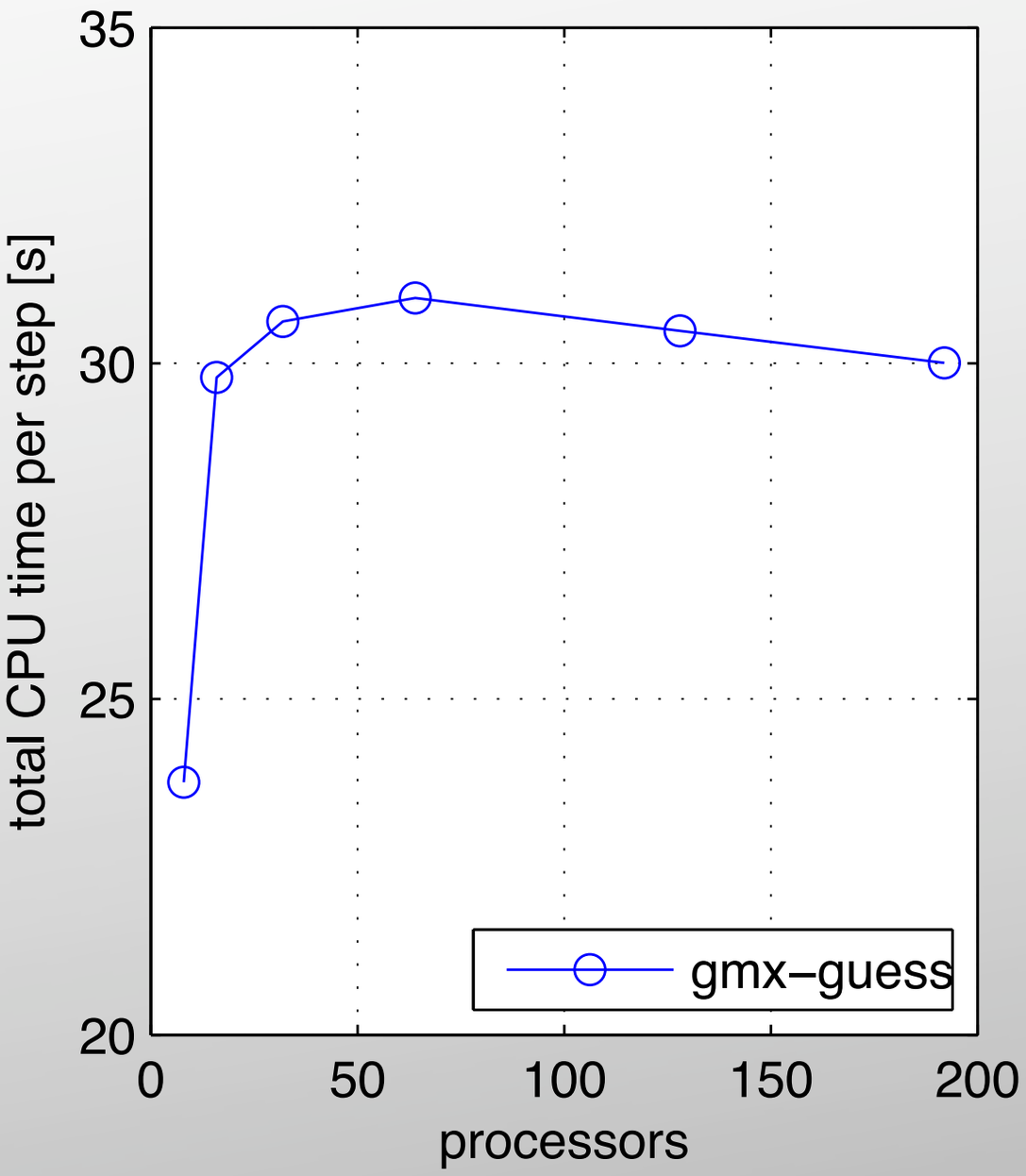
Easy GROMACS benchmarking

- performance measurement in GROMACS: build-in timers
- e. g. ns/day output from log file
- make a “short” test tpr
  ... but not too short!



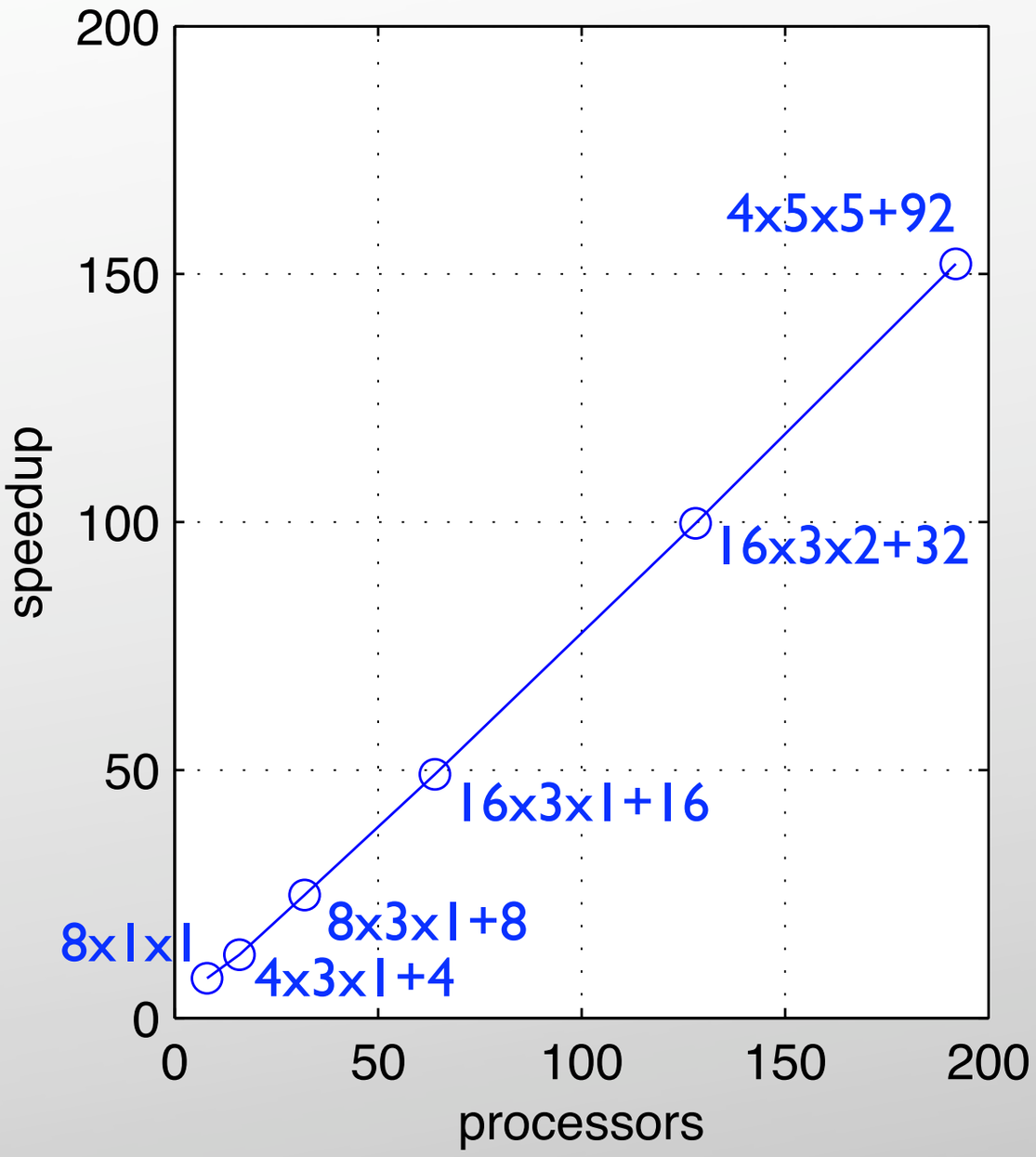
Performance tuning example

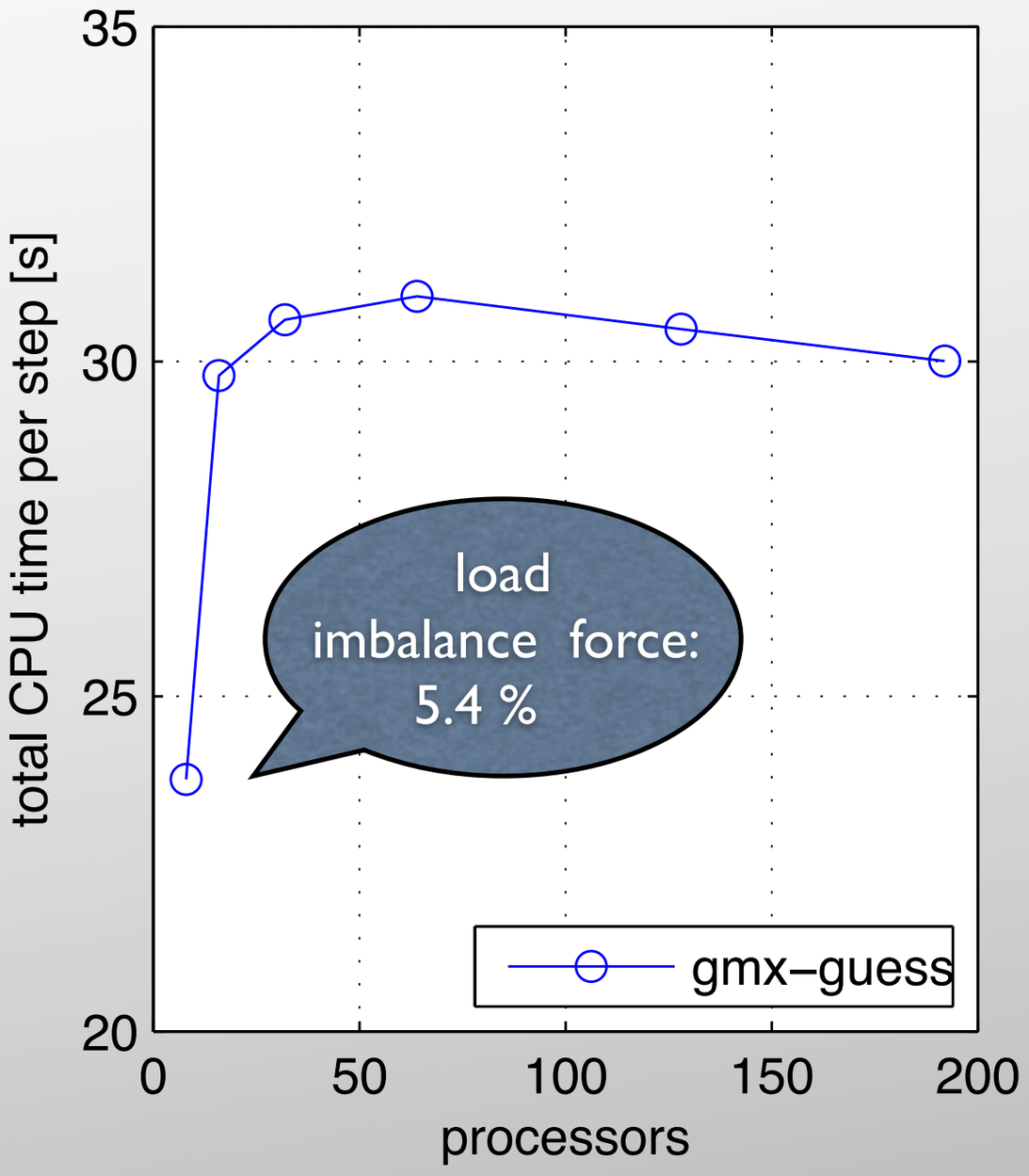




mpirun -np N mdrun

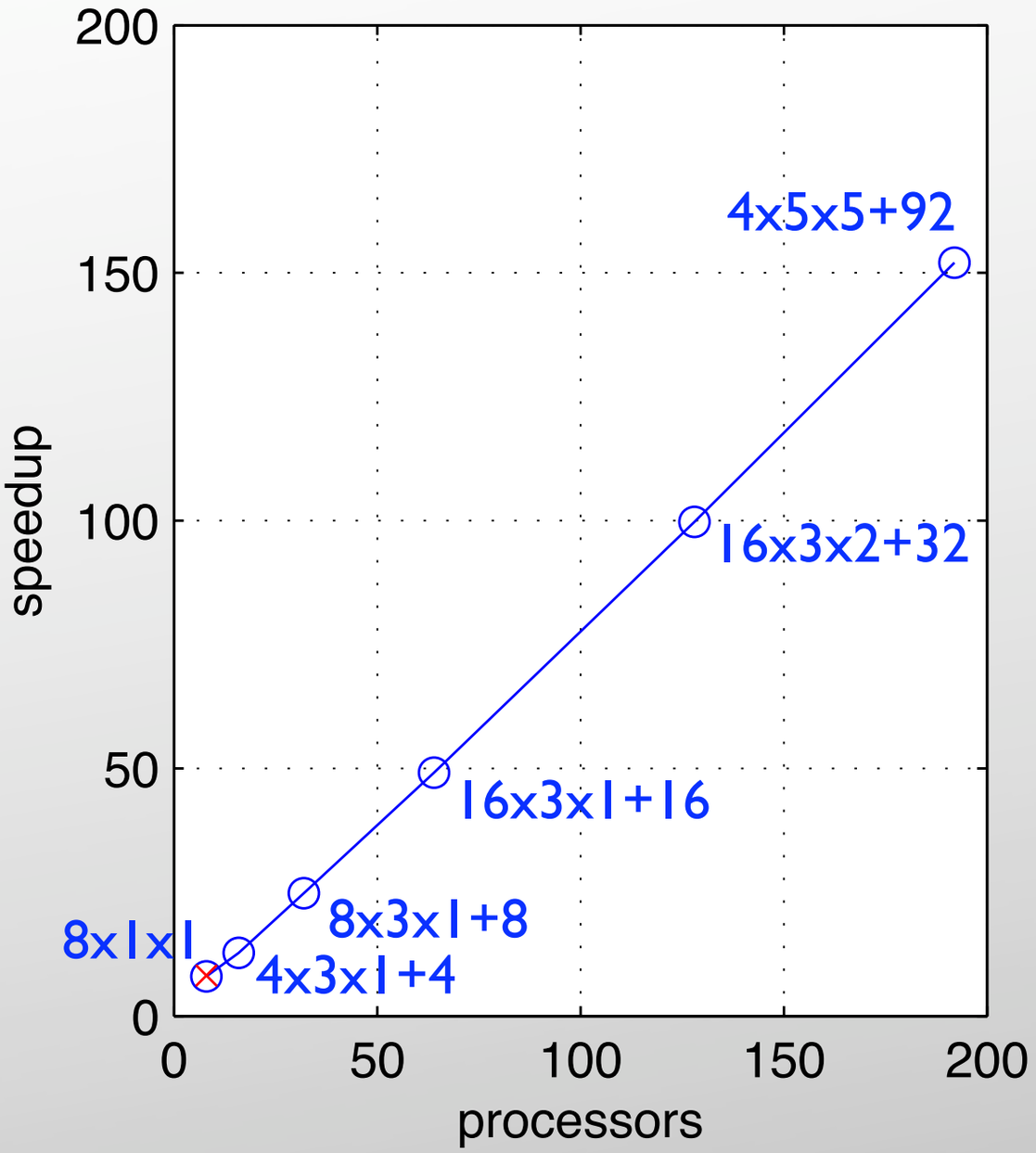
Performance tuning example

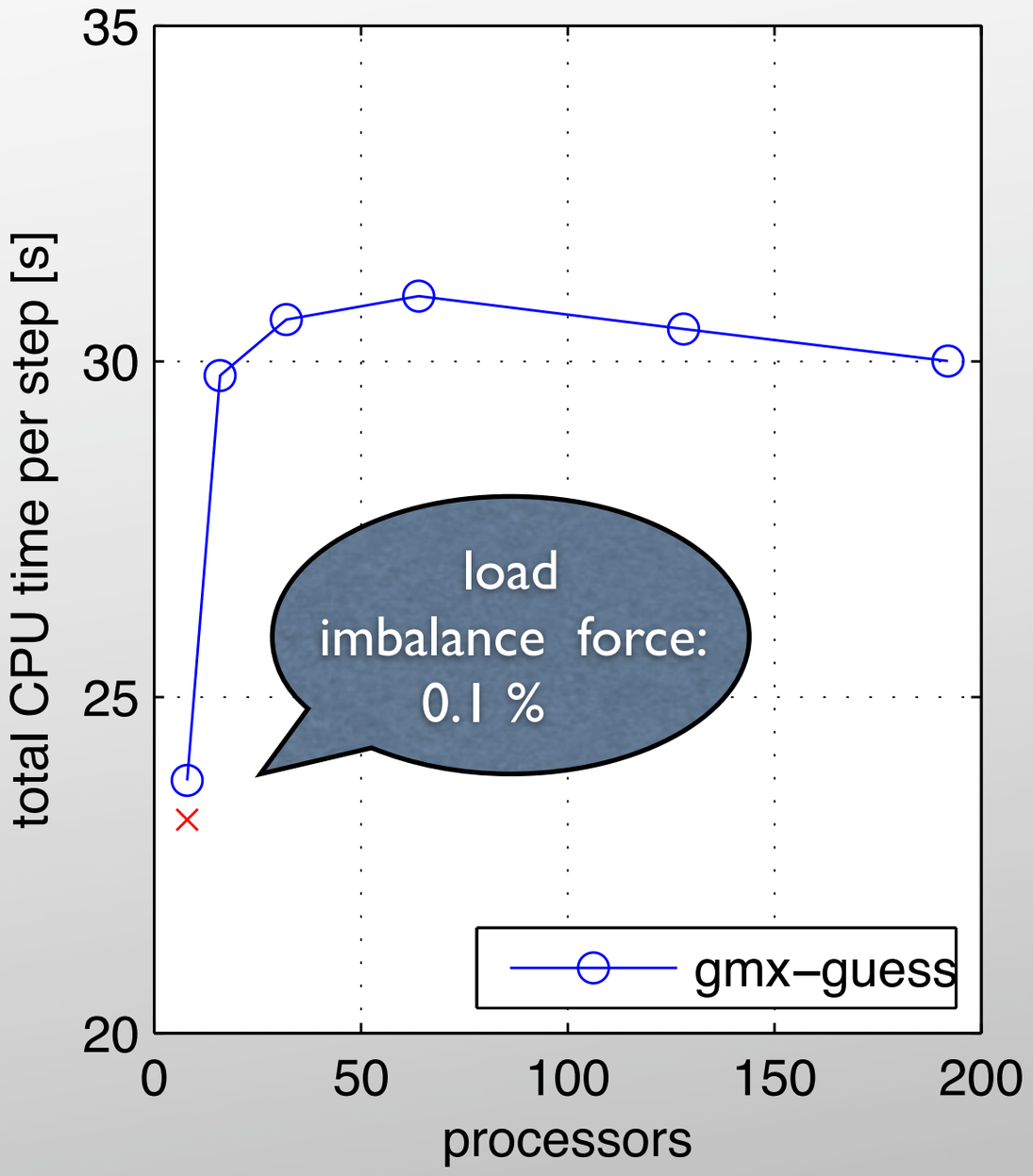




mpirun -np N mdrun **-v**

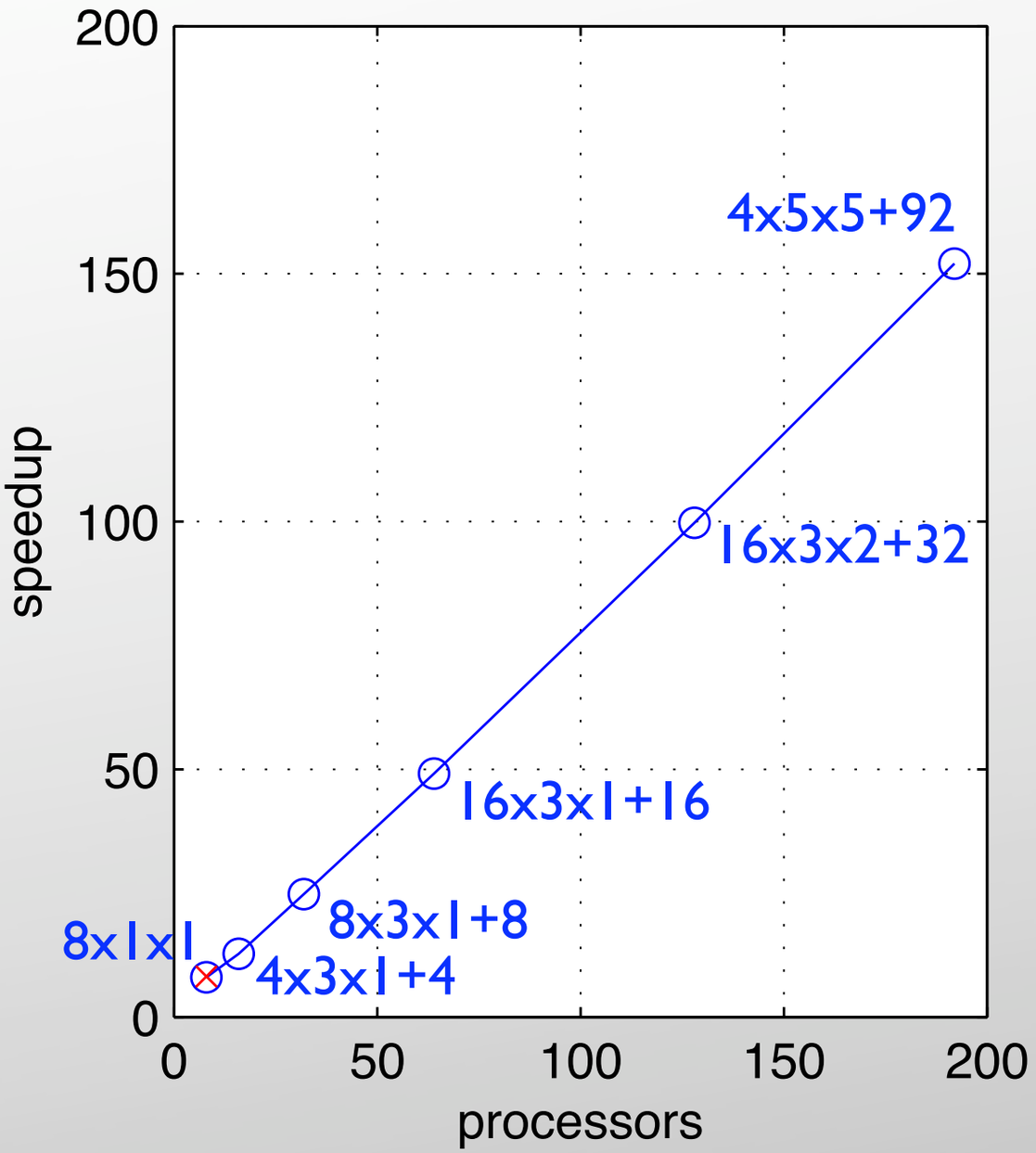
Performance tuning example

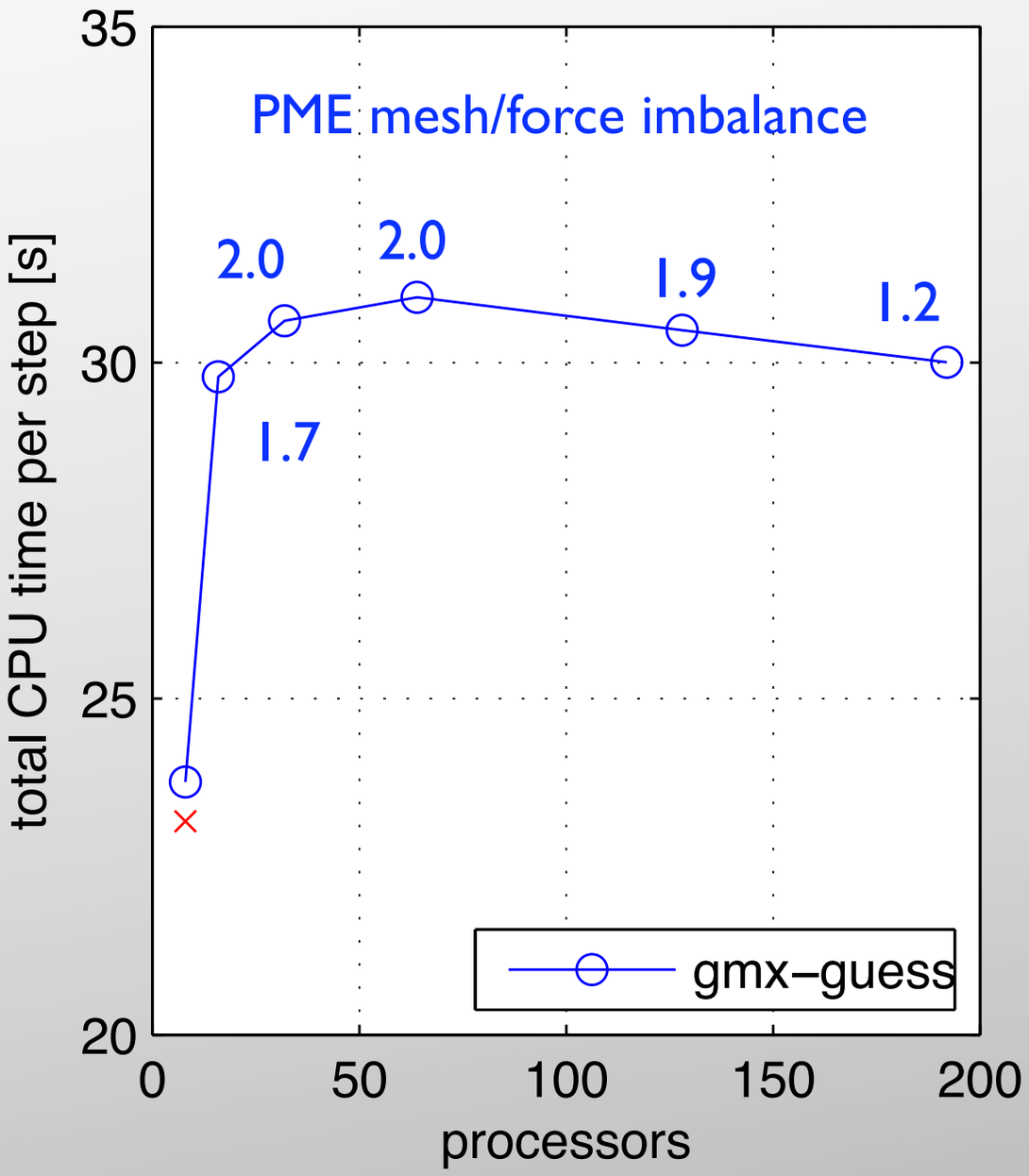




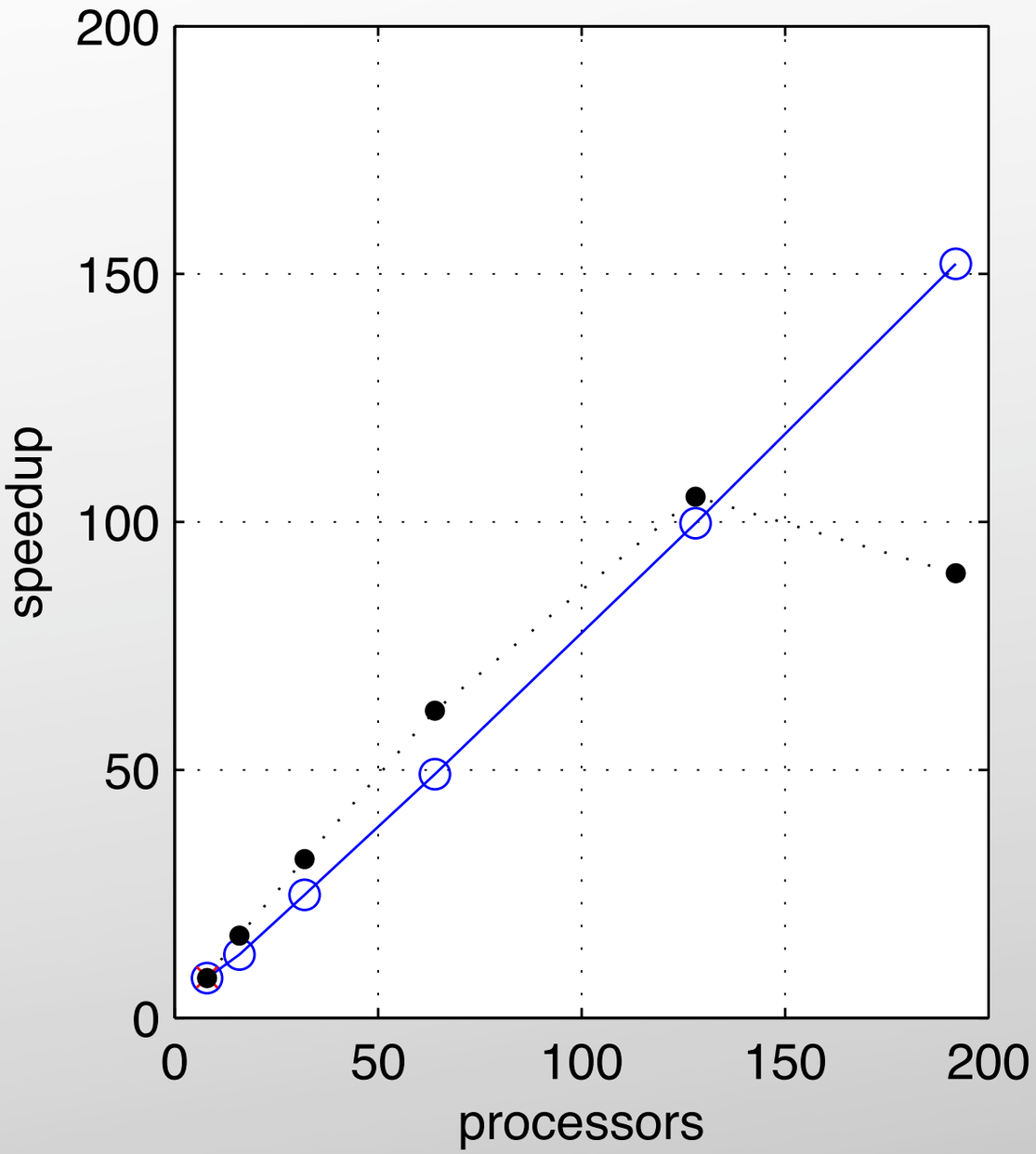
mpirun -np N mdrun -dlb yes

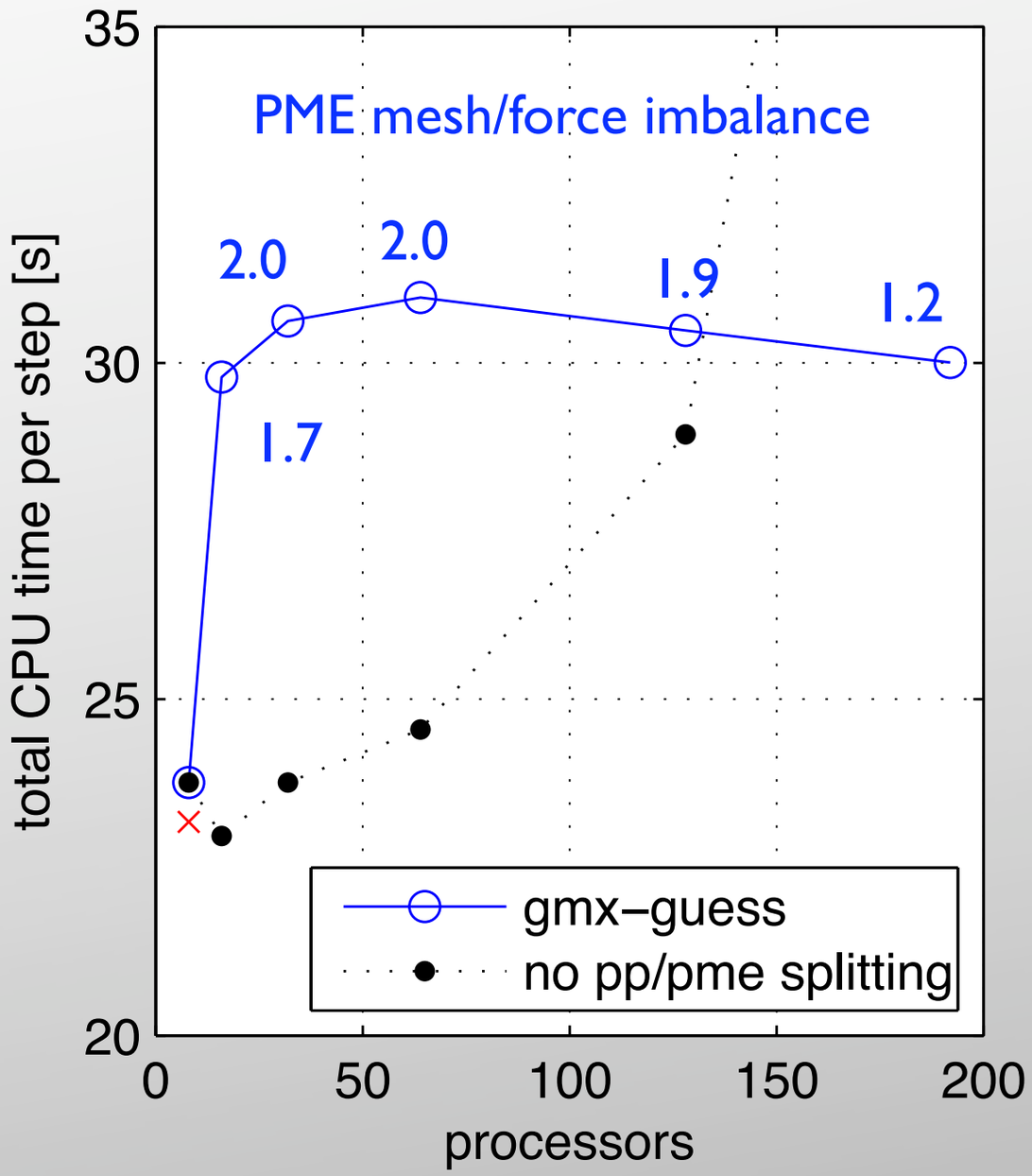
Performance tuning example





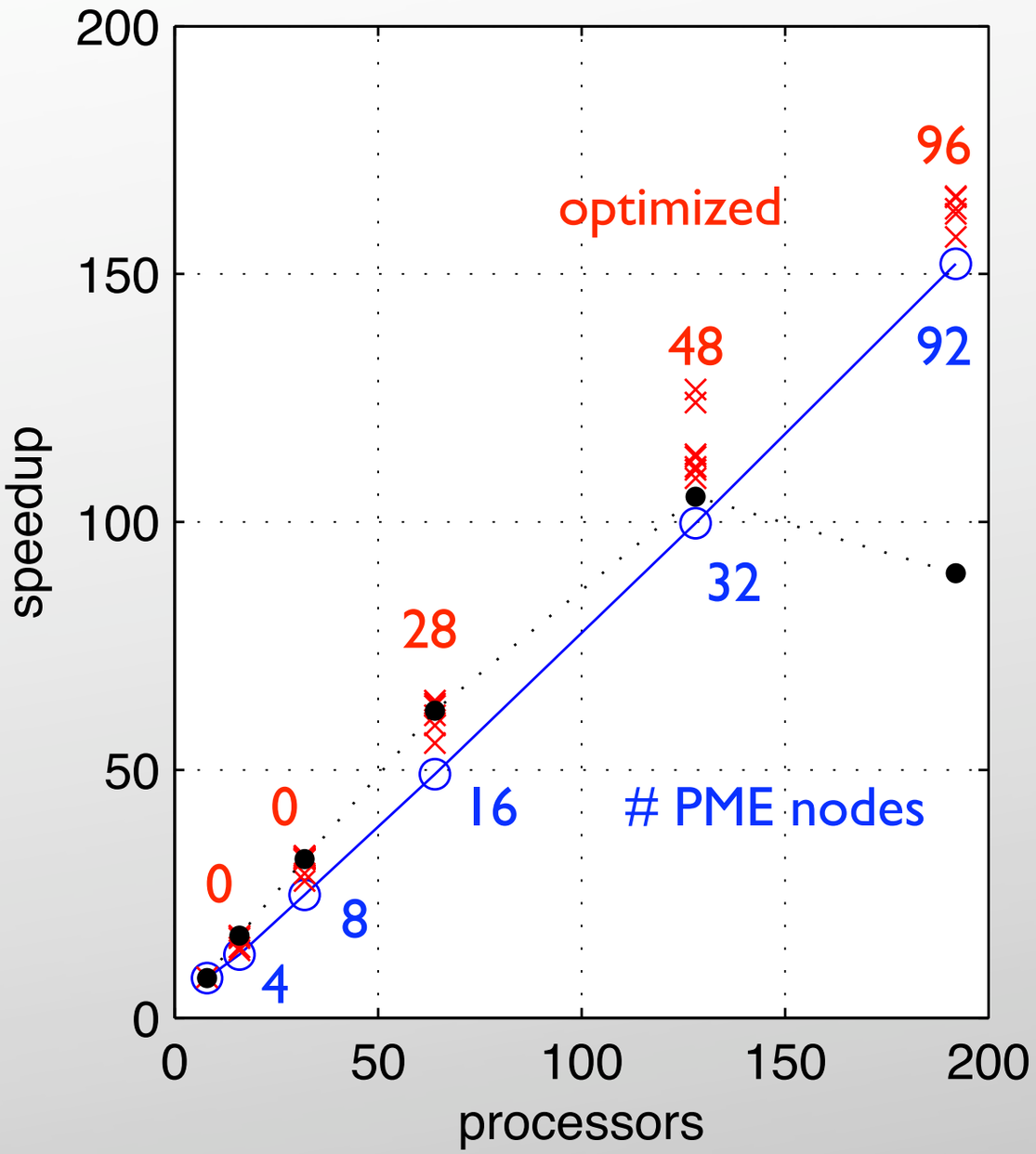
Performance tuning example

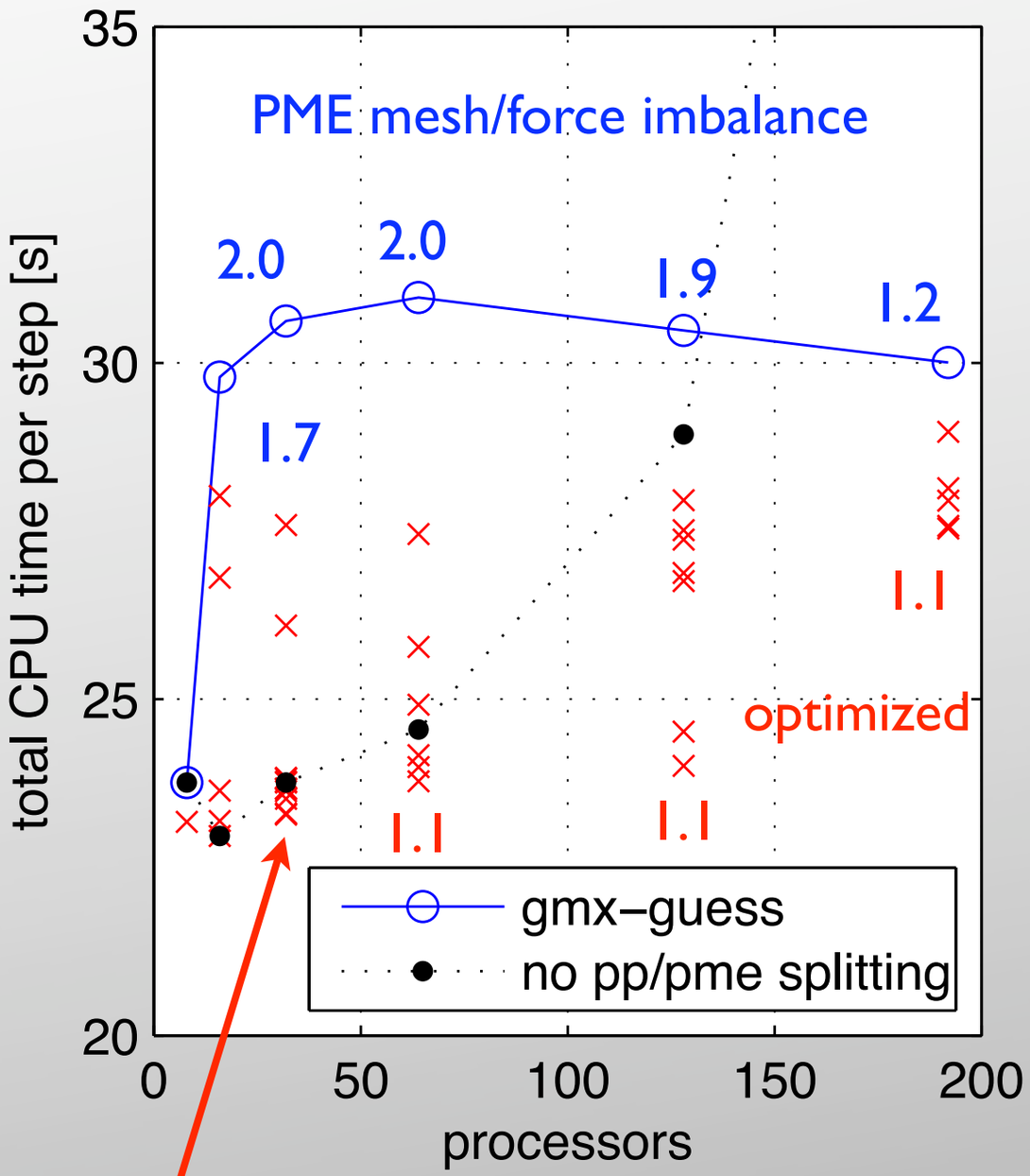




mpirun -np N mdrun -npme 0

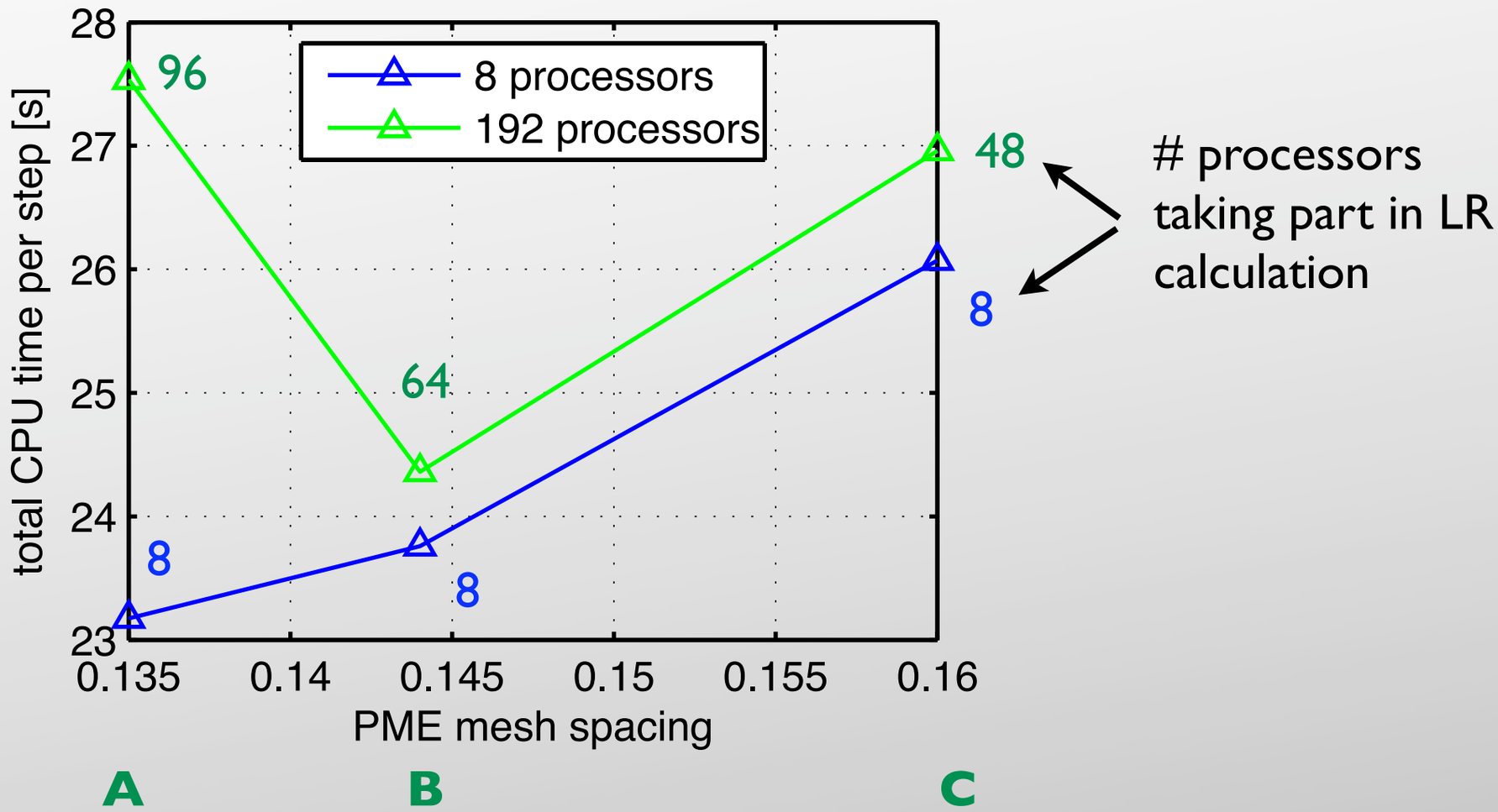
Performance tuning example





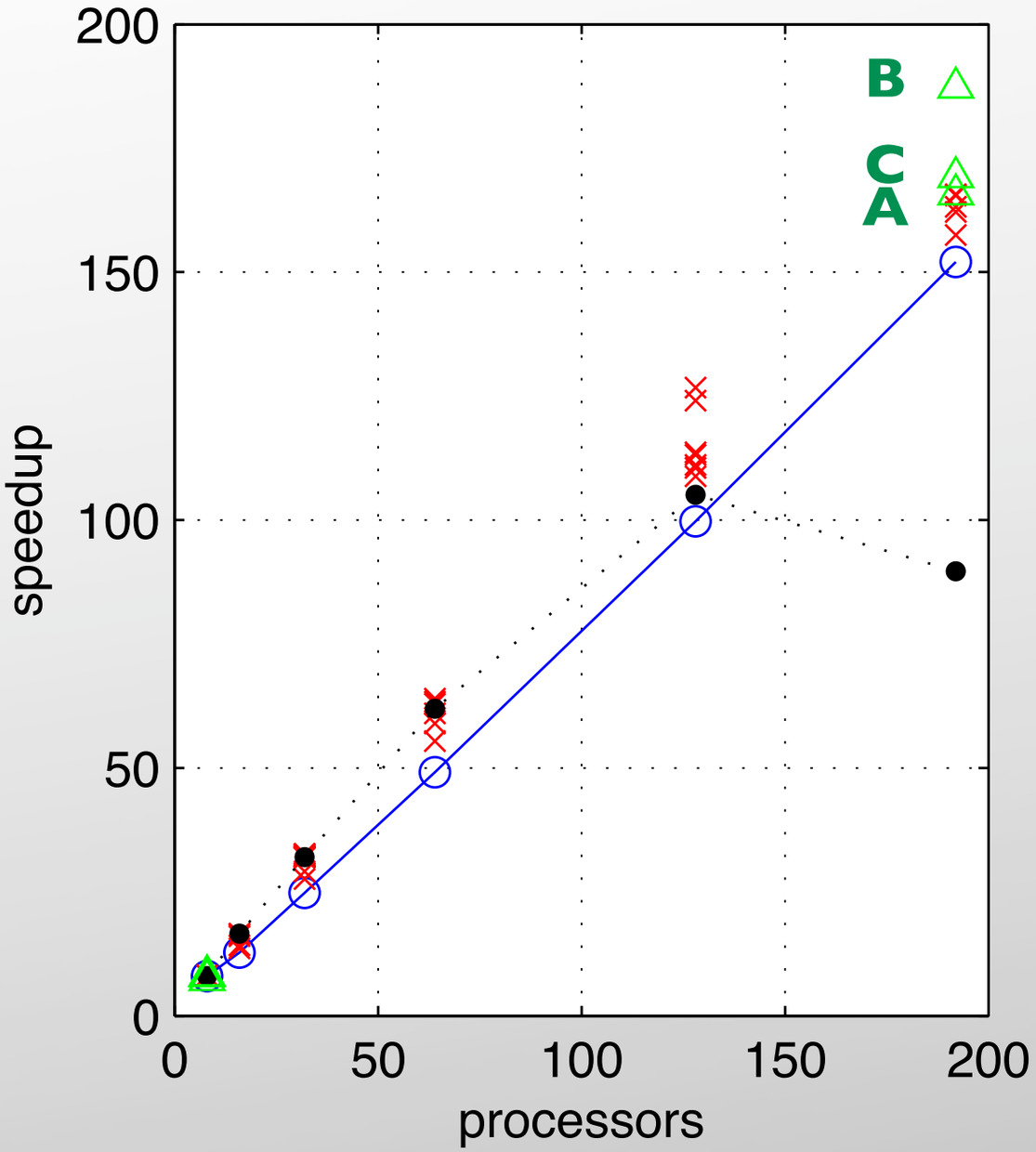
grid
8x2x2 (auto)
→ 8x4x1 (opt)

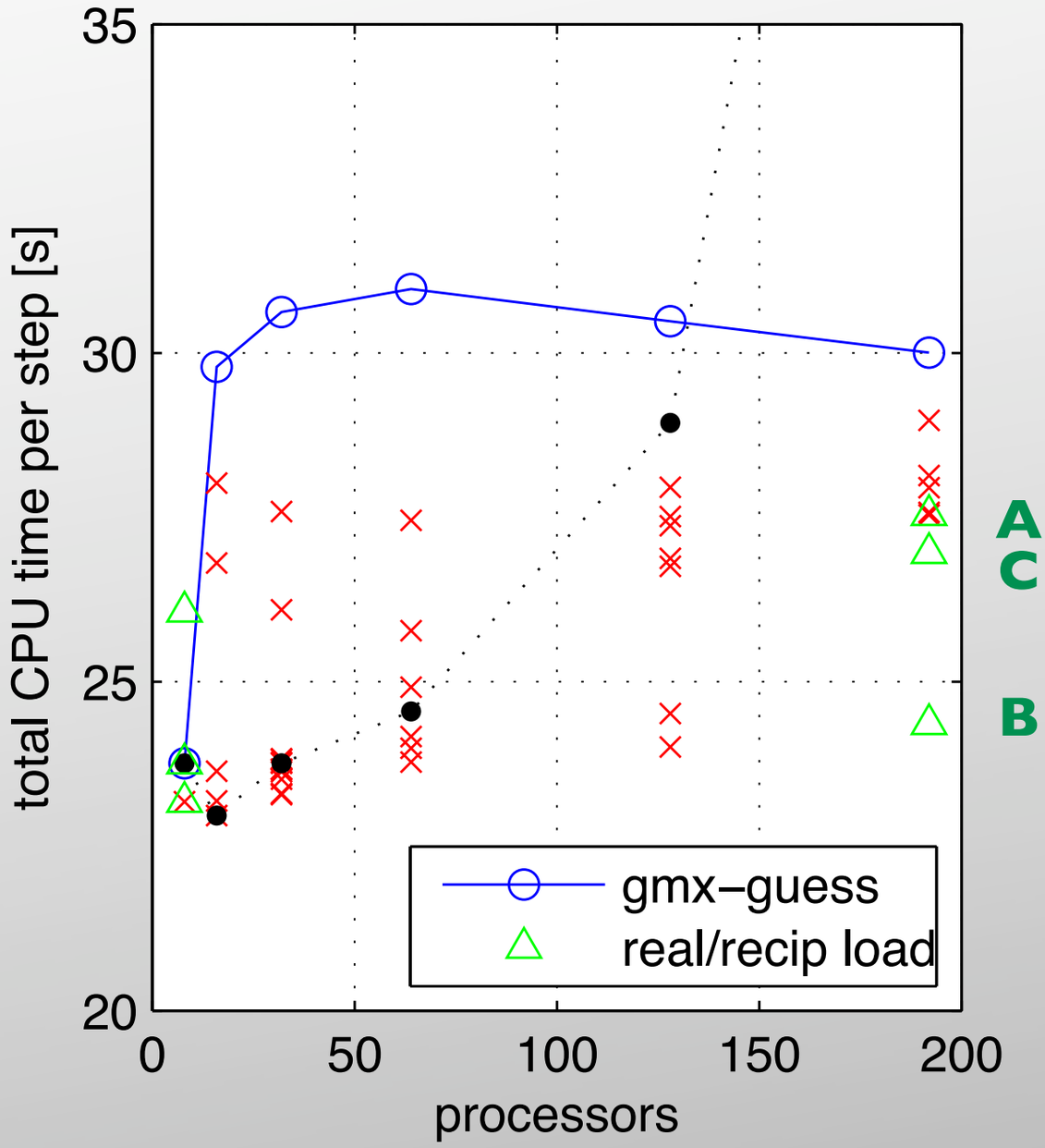
Shifting load to real space



	r_c / nm	grid spacing / nm	grid dimensions
A	1,00	0,135	275^3
B	1,07	0,144	256^3
C	1,20	0,160	231^3

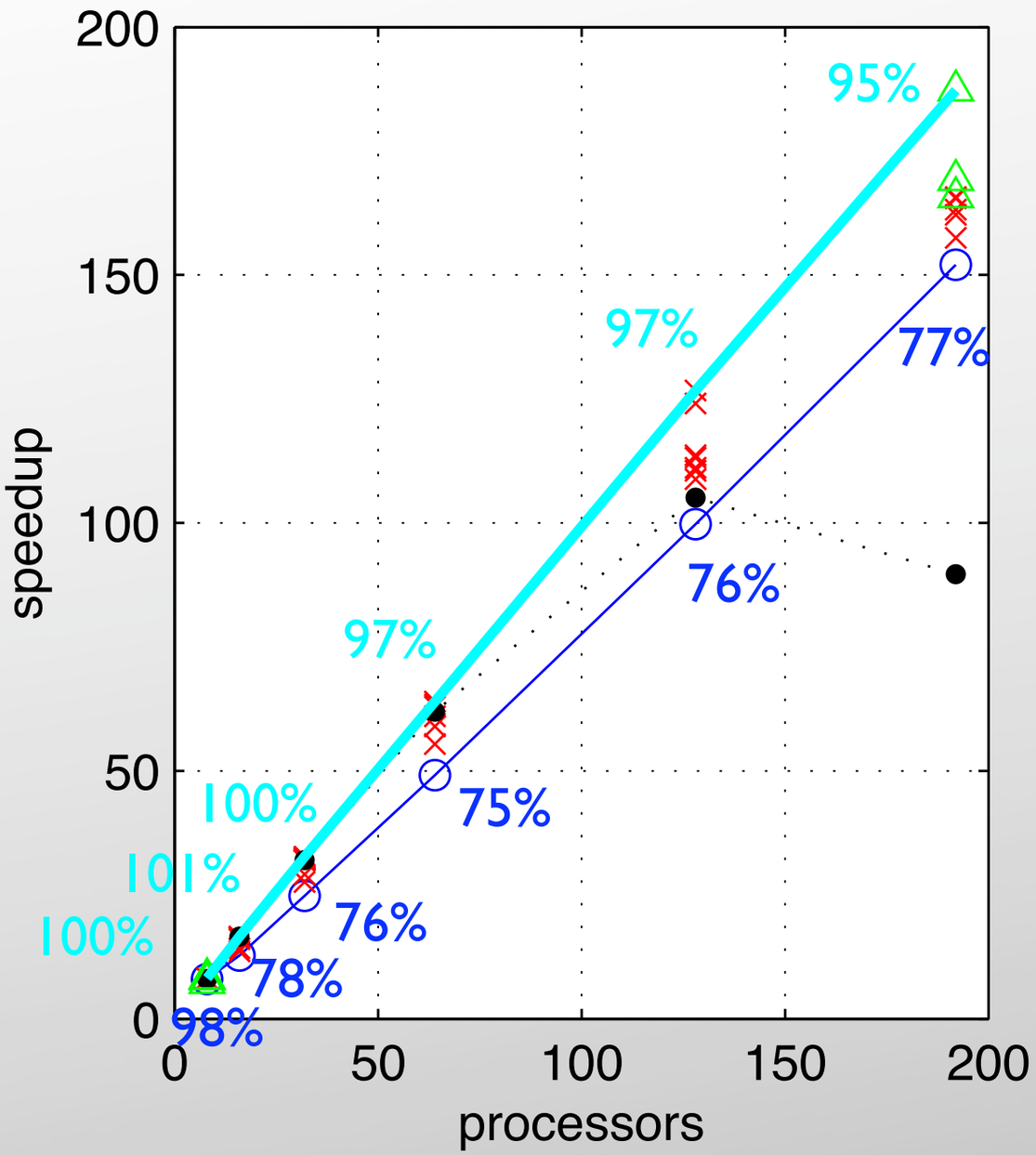
Performance tuning example

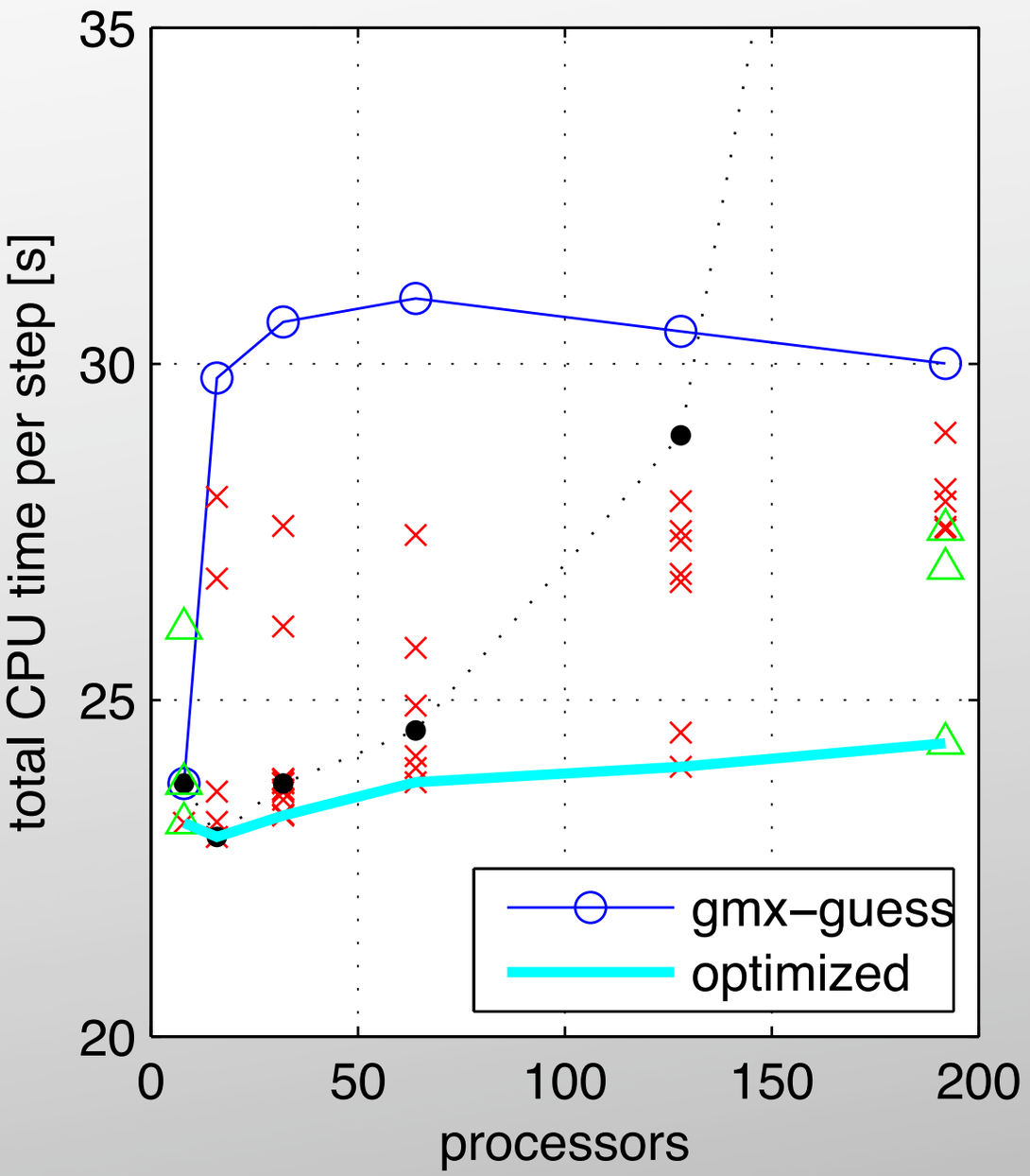




	r_c / nm	grid spacing / nm	grid dimensions
A	1,00	0,135	275^3
B	1,07	0,144	256^3
C	1,20	0,160	231^3

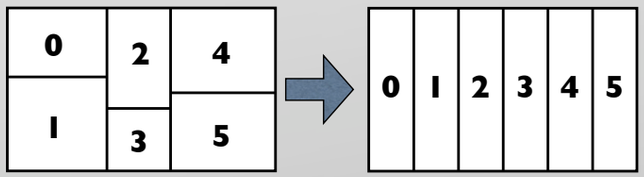
Performance tuning example





Tuning Howto

- make a short (>>100 steps) benchmark tpr, run with **-v**
- start with automatic settings, **apply performance hints**
- few CPUs:
  - turn on dlb if there is a load imbalance in the force
  - turn on/off PME nodes
- many CPUs:
  - **adjust # of PME nodes** if an imbalance pme mesh/force is reported
  - **adjust real/reciprocal space workload** by multiplying both r_c and fourierspacing by (same) factor near 1
- might want to try other DD grids:
  - hight # of x-slabs mimicks the data organization for PME



  - can be favourable NOT to decompose in one direction (no x/f communication pulse needed in that direction)

Acknowledgments

- The GROMACS developers
  - David van der Spoel
  - Erik Lindahl
  - Berk Hess
- The Theoretical and Computational Biophysics Department
  - Mareike Zink
  - Gerrit Groenhof
  - Bert de Groot
  - Helmut Grubmüller