Compute power optimization by cluster & simulation setup

Carsten Kutzner

MPI for biophysical Chemistry, Göttingen
Theoretical and Computational Biophysics

GROMACS Workshop, Göttingen, 2016-05-19

Q. How to produce as much MD trajectory as possible for my science?

I have access to compute resources

(core-h limited)

Q1. How can optimal performance be obtained?

I want to buy a cluster

(money limited)

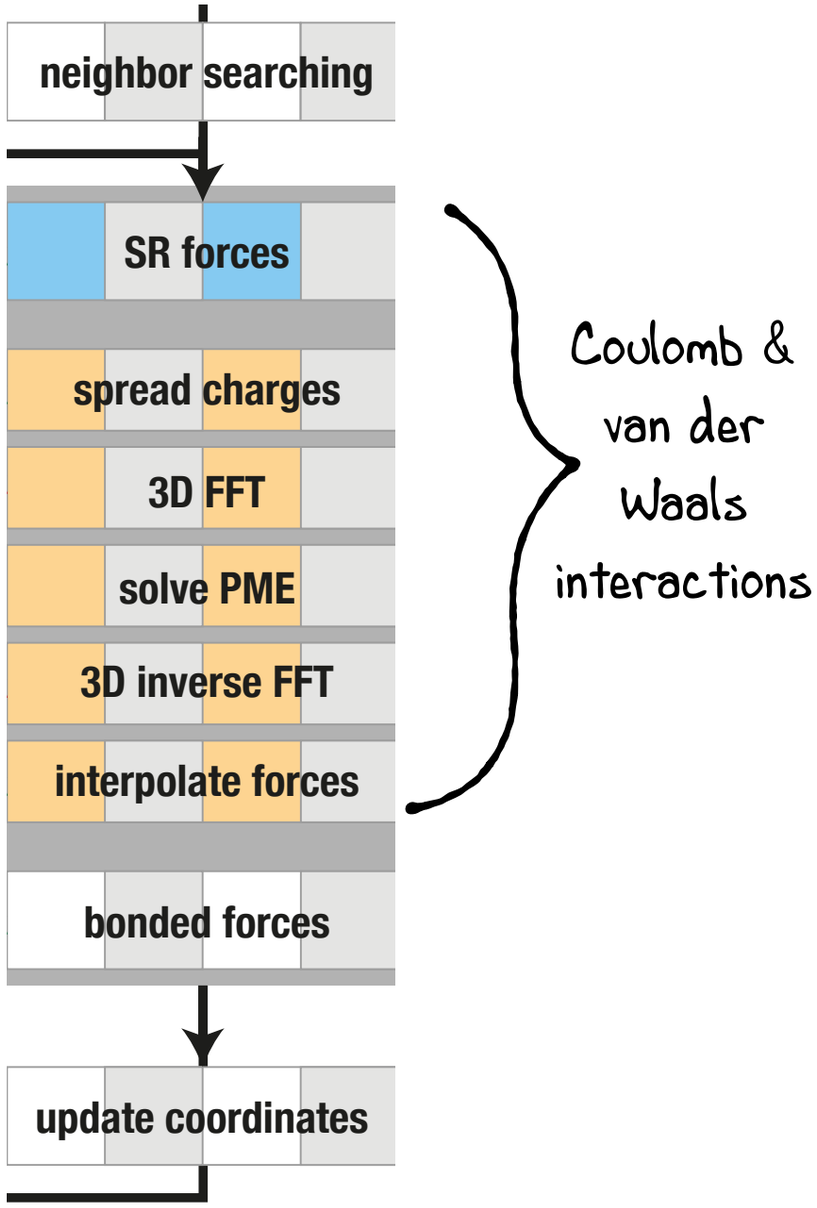
Q2. What is the optimal hardware to run GROMACS on?

Q1.

How can optimal GROMACS performance be obtained?

- Before the simulation:
  The foundation of good performance
  - compilation, e.g. compiler, SIMD instructions, MPI library
  - system setup, e.g. virtual sites
- When launching `mdrun`:
  main benefits come from optimizing the parallel run settings
  - reach a balanced computational load
  - keep communication overhead small

Recap: GROMACS serial time step

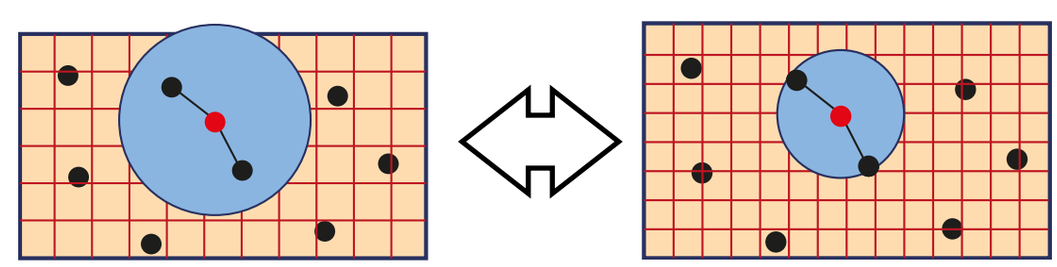


- Coul. + vdW make up for most of the time step
- PME decomposes these into SR and LR contributions
- SR can be efficiently calculated in **direct space**
- LR can be efficiently calculated in **reciprocal space**
- recip. part needs FT of charge density

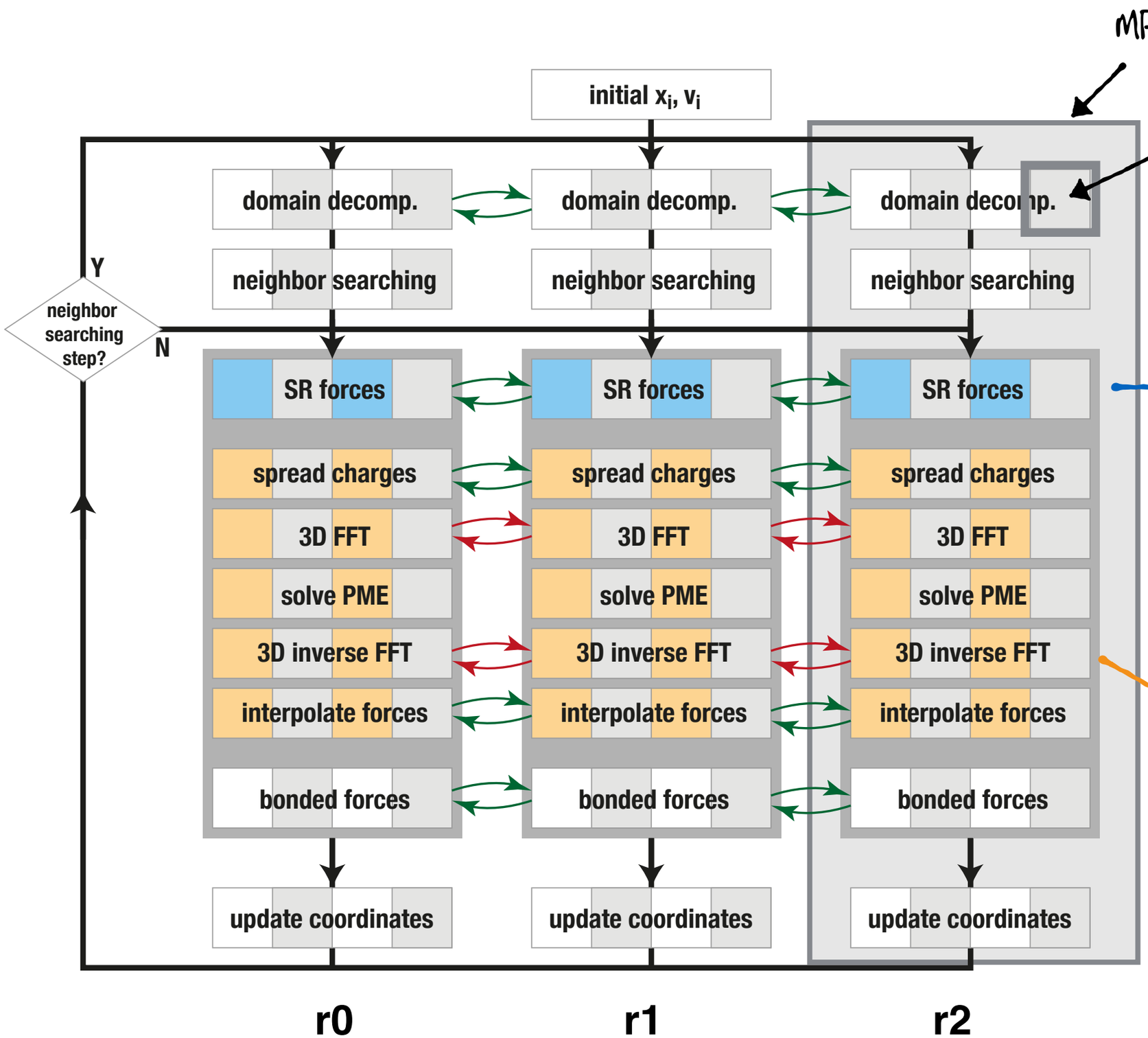
communication intense in parallel

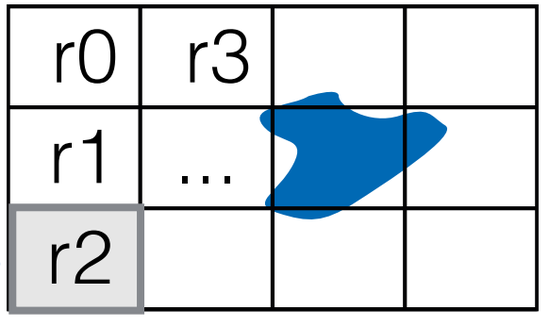
- PME allows to shift work between real, **SR** (PP), and reciprocal, **LR** (PME), space parts (balance cutoff : grid spacing)

more PP/GPU work **more PME/CPU work**



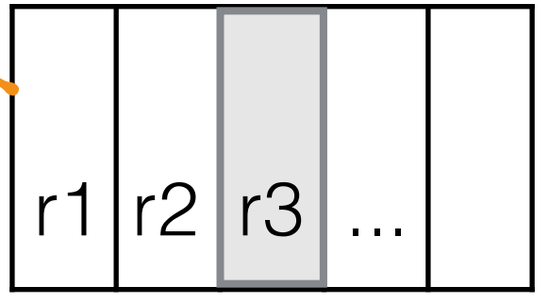
Recap: GROMACS parallel time step





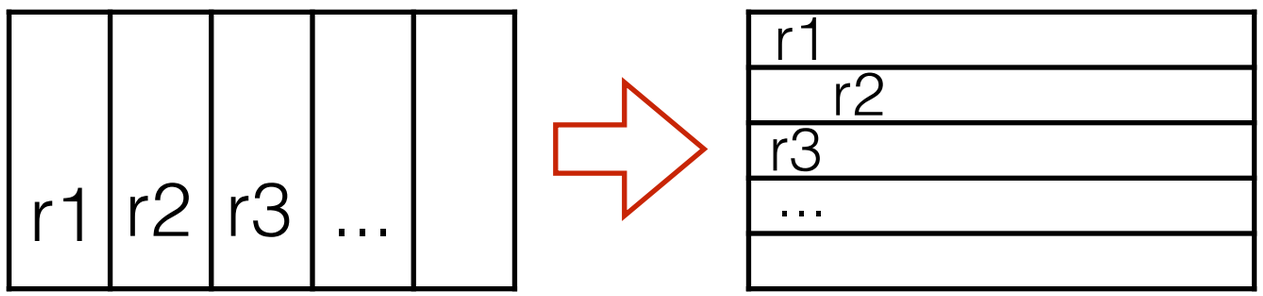
direct space interactions decomposed into domains r_{MPI} = n_x x n_y x n_z

✓ **boundary layer communication**



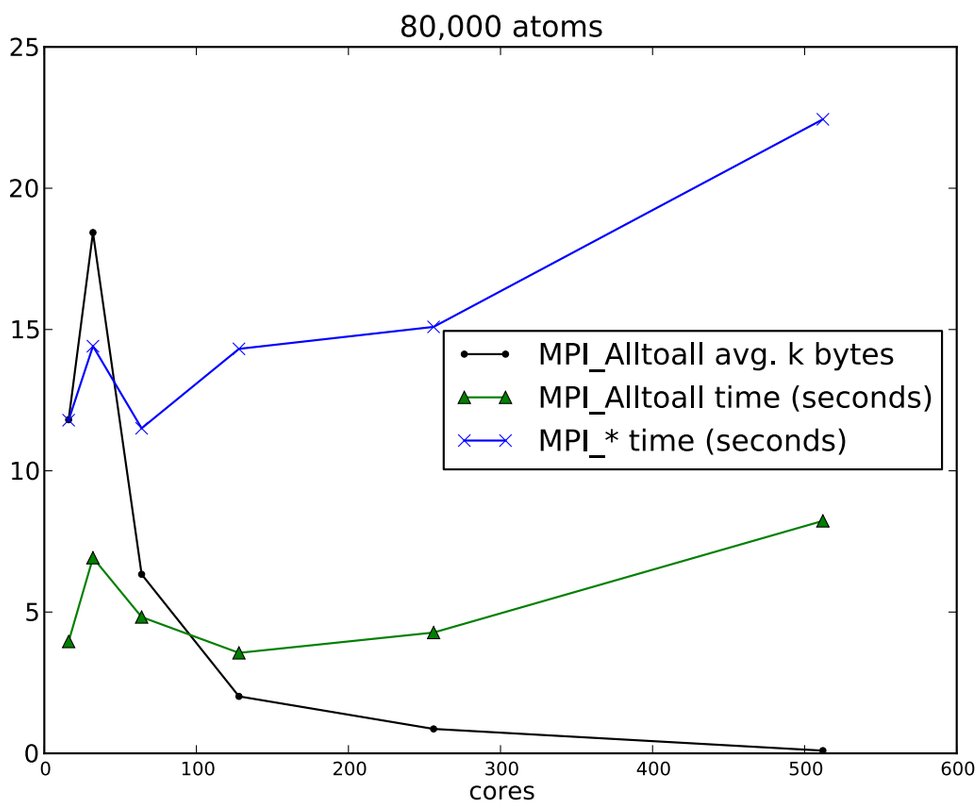
reciprocal space / LR PME
use r_{MPI} slabs

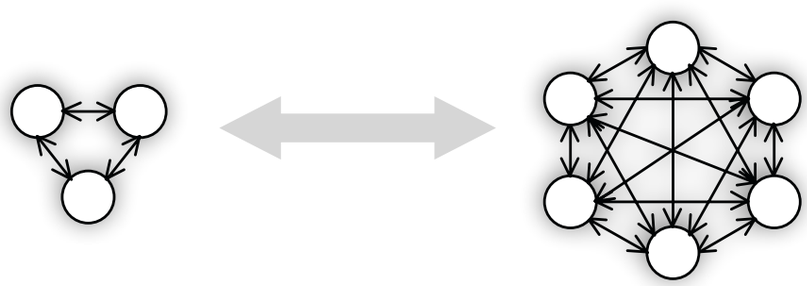
💣 **all-to-all communication r^2 during FFT grid transpose**



all-to-all
r^2 messages

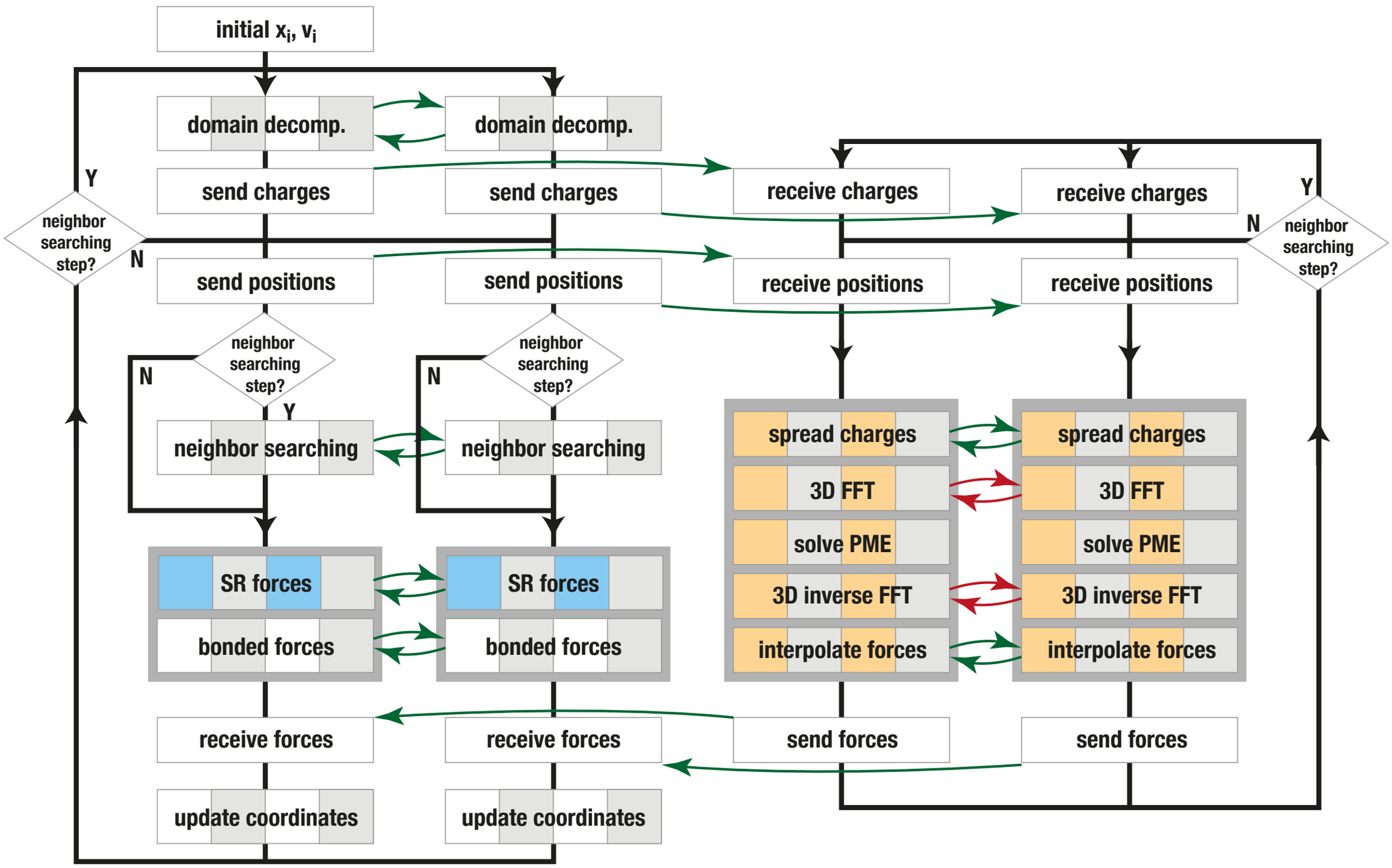
- PME calculation cost is O(N log N) with N atoms, but in parallel, **PME communication becomes the bottleneck**
- number of messages increases by r^2, therefore also total latency





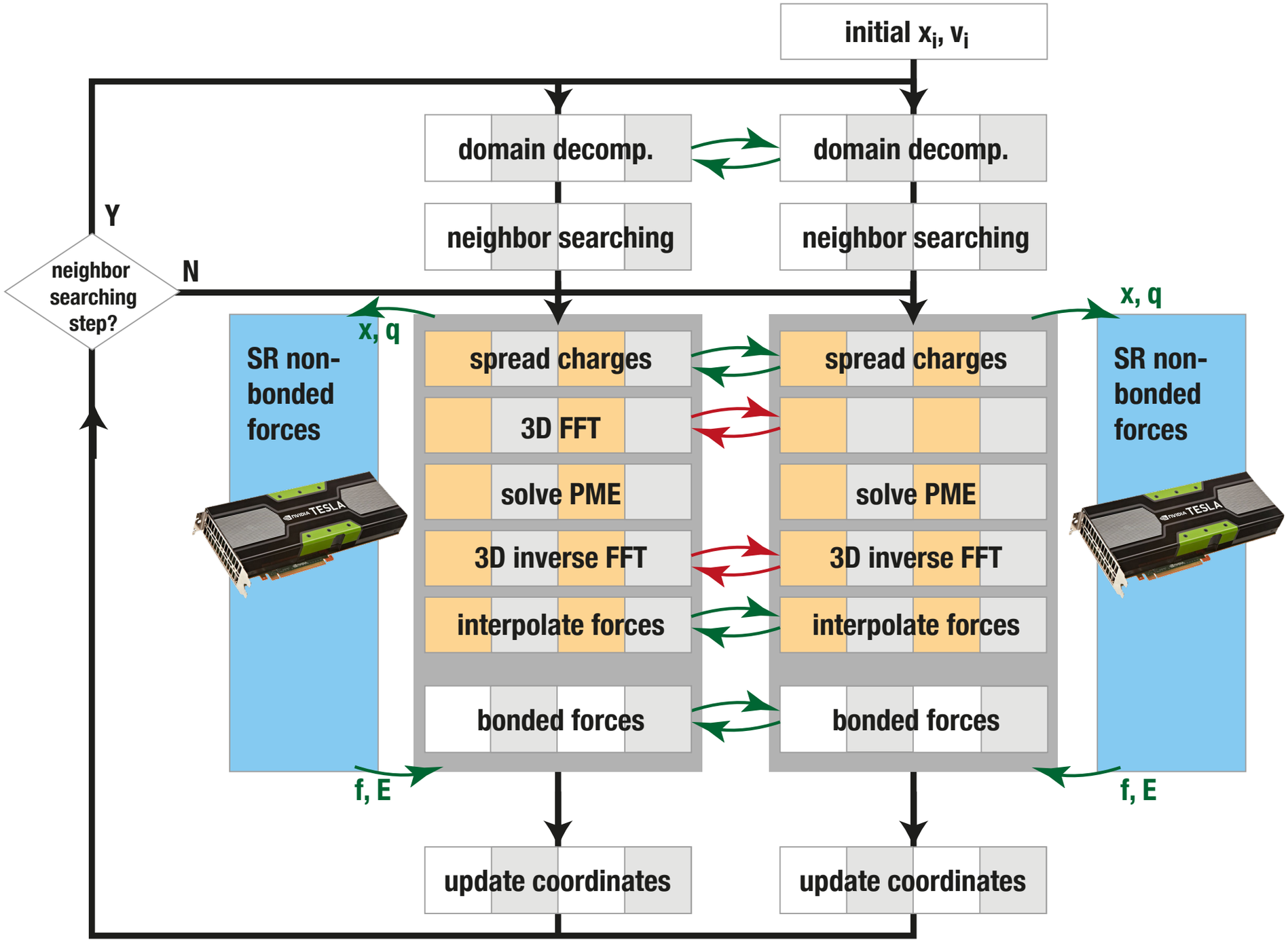
Independent calculation of SR and LR forces

- offload LR electrostatics to a subset of MPI ranks
- typically 1/4 → reduces # of messages 16-fold

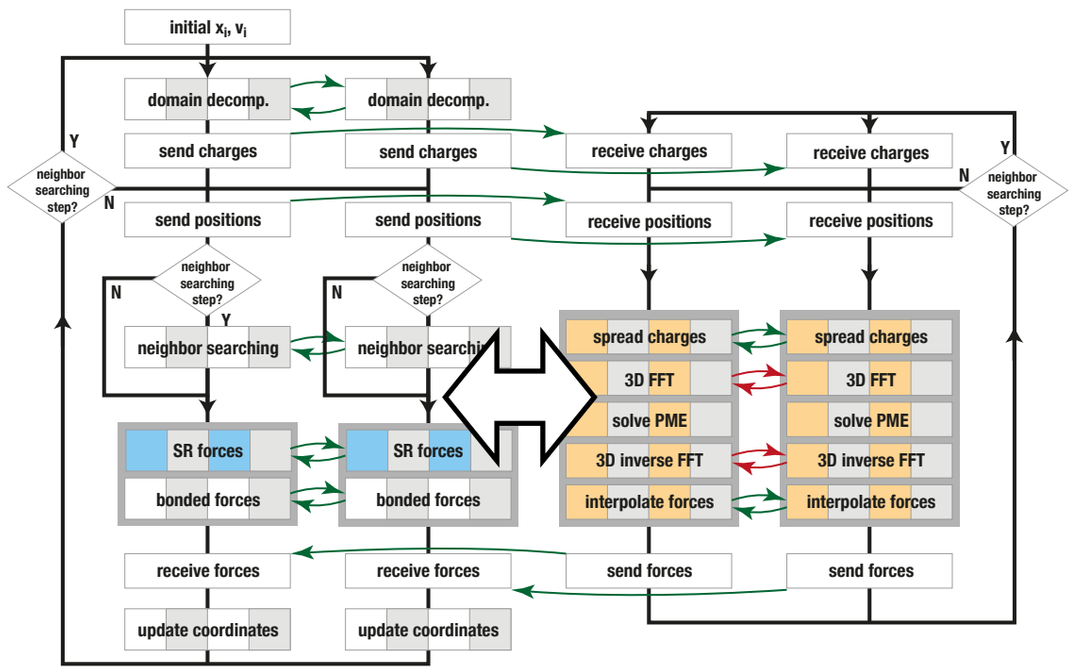


SR processes (**direct space, PP**) **LR** processes (**Fourier space, PME**)

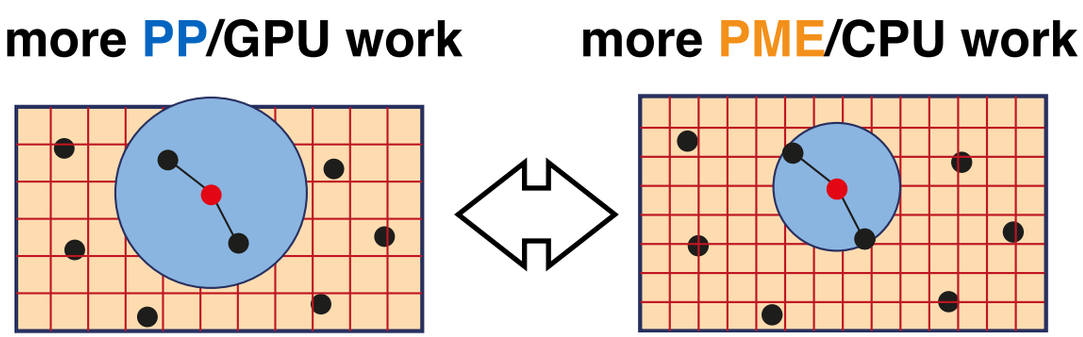
SR non-bonded forces can be offloaded to GPUs



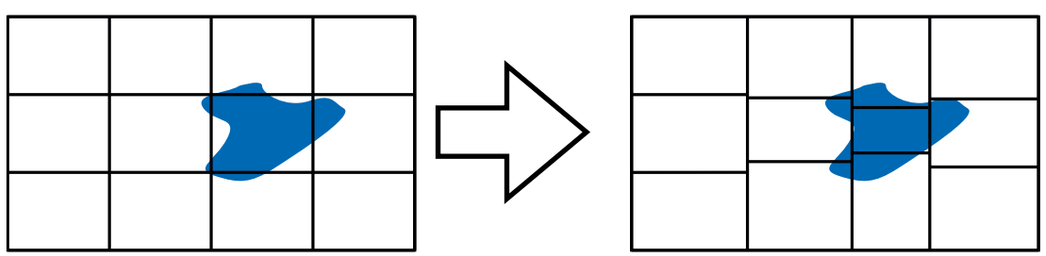
Automatic multi-level load balancing



1. Number of **SR (PP)** vs. **LR (PME)** processes is statically assigned

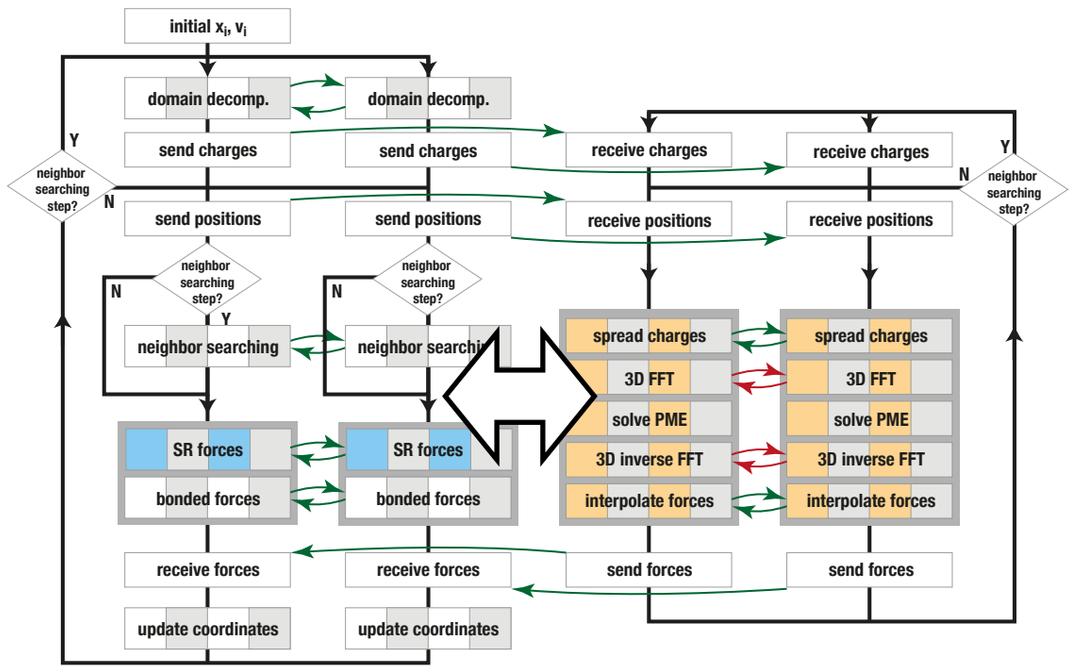


2. PME allows to shift work between real and reciprocal space parts!
   → fine-tune **SR** (PP) vs. **LR** (PME)
   (balance cutoff : grid spacing)



3. Balance **direct space** workload between **SR** domains

Automatic multi-level load balancing

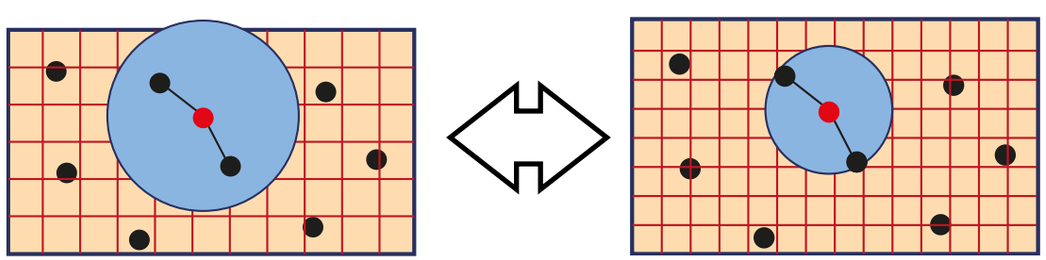


1. Number of **SR (PP)** vs. **LR (PME)** processes is statically assigned

statically assigned

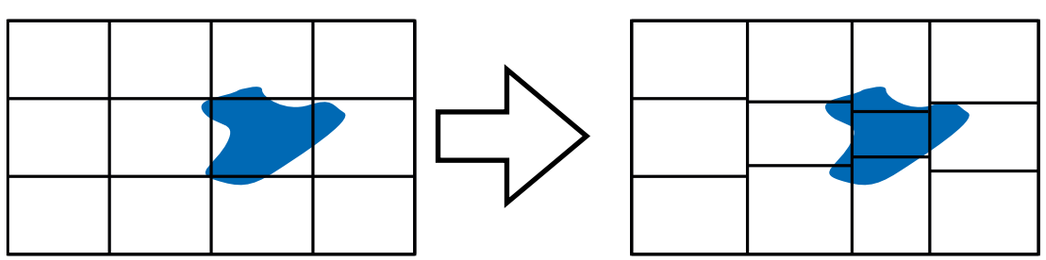
more **PP**/GPU work

more **PME**/CPU work



2. PME allows to shift work between real and reciprocal space parts!
   → fine-tune **SR** (PP) vs. **LR** (PME)
   (balance cutoff : grid spac

once at start of simulation



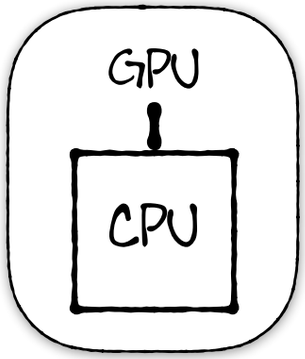
3. Balance **direct space** workload between **SR** domains

continuously

Automatic multi-level load balancing

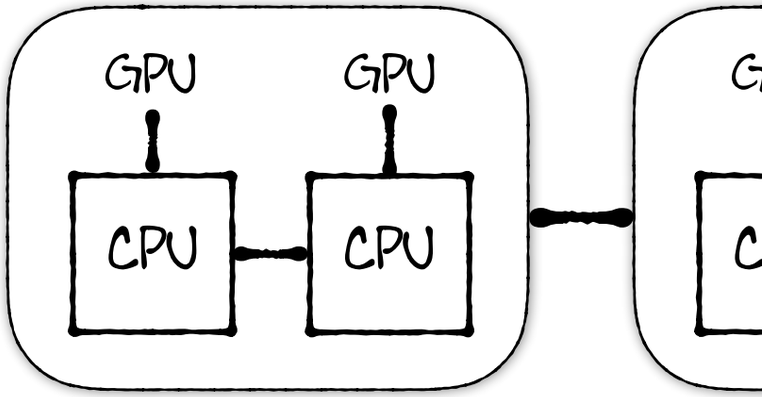
- good news:

  on single nodes with a 1 CPU and opt. 1 GPU,
  GROMACS' automatic settings often already give optimal performance (thread-MPI)



- however, …

  on multi-GPU or multi-socket CPU nodes,
  or on a cluster of nodes,
  manual tuning will in most cases enhance performance



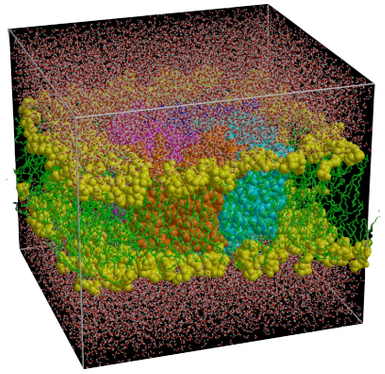
Tips & tricks for optimal GROMACS performance

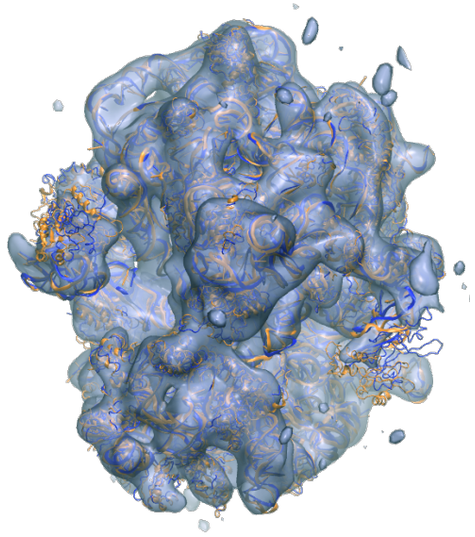
Most importantly:

If in doubt, make a benchmark

- testing different settings just takes few minutes
- will directly uncover the optimal settings for your MD system on your hardware
- the following demonstrations were done with these 2 benchmarks:

MD system	membrane protein (MEM)	ribosome (RIB)
# particles	81,743	2,136,412
system size (nm)	10.8×10.2×9.6	31.2×31.2×31.2
time step length (fs)	2	4
cutoff radii[a] (nm)	1.0	1.0
PME grid spacing[a] (nm)	0.120	0.135

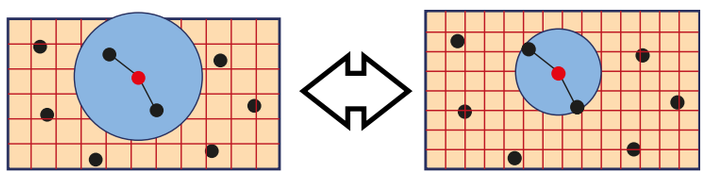


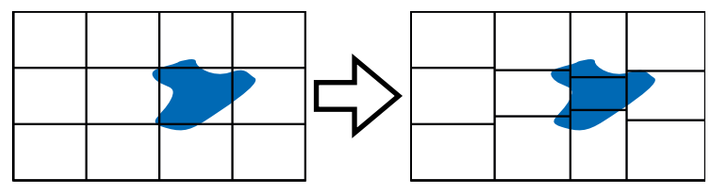


Tip 1 of 10:

Getting useful performance numbers in benchmarks

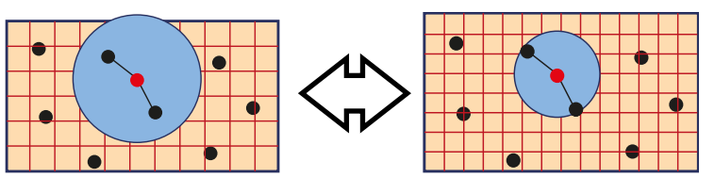
- Both automatic load balancing mechanisms need time to reach the optimum
- Reject the initial time steps from performance measurement with
- `mdrun -resetstep 2000`
  `mdrun -resethway`

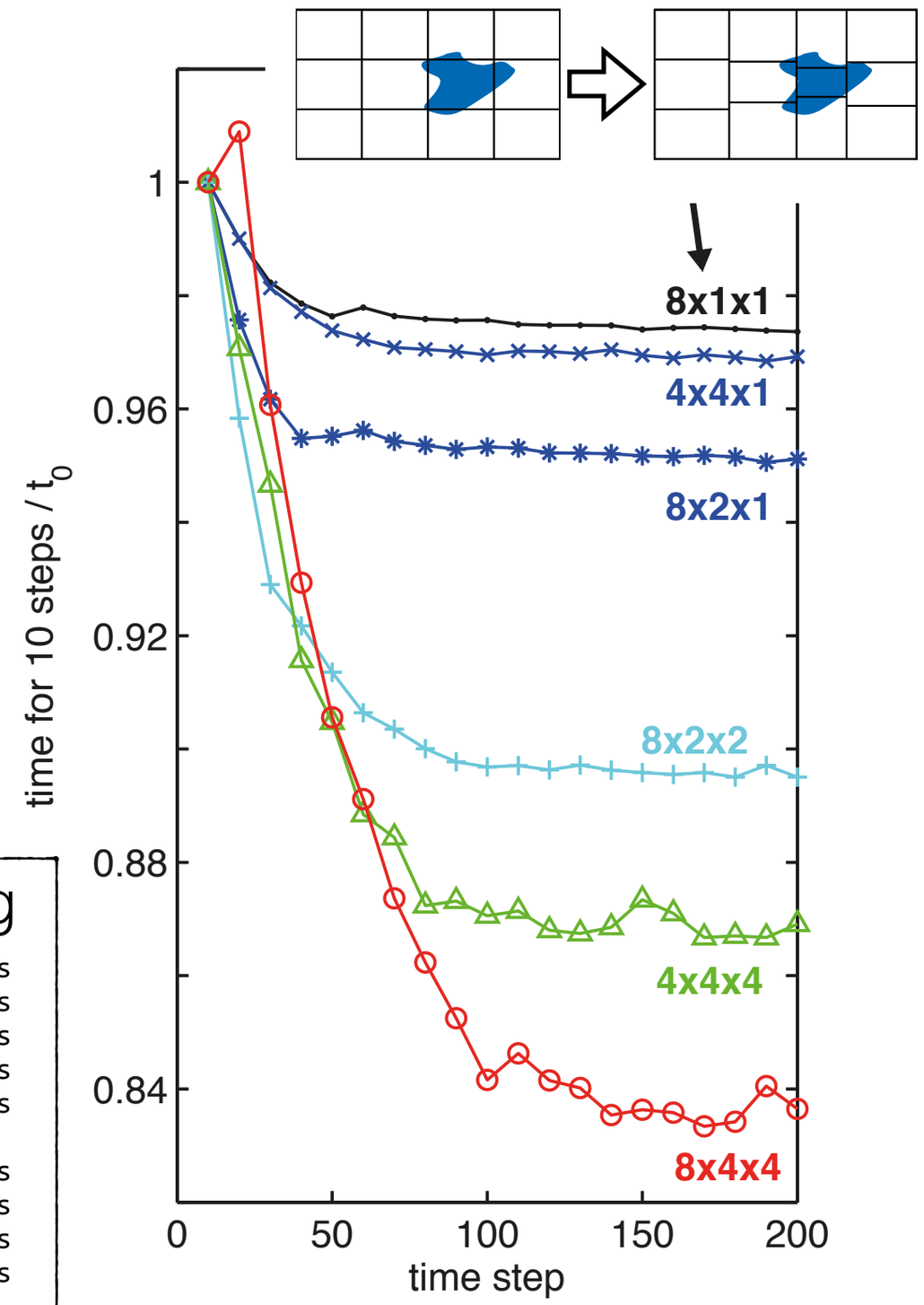




Getting useful performance numbers in benchmarks

- Both automatic load balancing mechanisms need time to reach the optimum
- Reject the initial time steps from performance measurement with
- `mdrun -resetstep 2000`
  `mdrun -resethway`



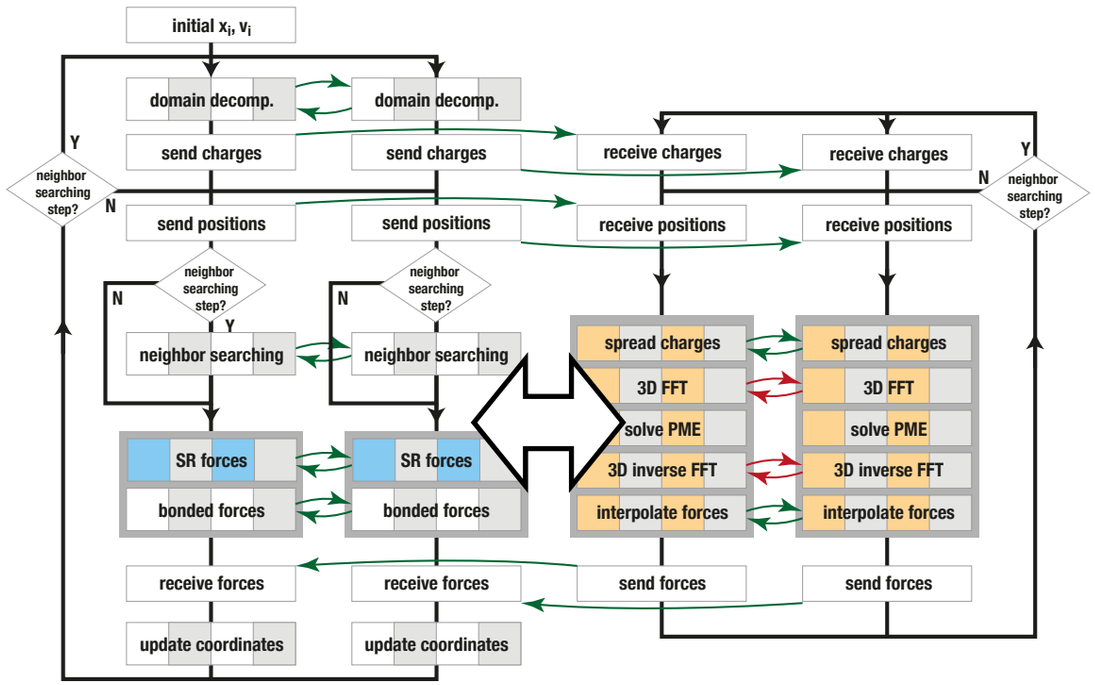


md.log

```
                    DD  step 39 load imb.: force 14.8%
step   80: timed with pme grid 96 96 240, coulomb cutoff 1.200: 2835.1 M-cycles
step  160: timed with pme grid 84 84 208, coulomb cutoff 1.311: 2580.3 M-cycles
step  240: timed with pme grid 72 72 192, coulomb cutoff 1.529: 3392.3 M-cycles
step  320: timed with pme grid 96 96 240, coulomb cutoff 1.200: 2645.2 M-cycles
step  400: timed with pme grid 96 96 224, coulomb cutoff 1.212: 2569.9 M-cycles
                                       …
step 1200: timed with pme grid 96 96 208, coulomb cutoff 1.305: 2669.8 M-cycles
step 1280: timed with pme grid 84 84 208, coulomb cutoff 1.311: 2677.5 M-cycles
step 1360: timed with pme grid 84 84 200, coulomb cutoff 1.358: 2770.5 M-cycles
step 1440: timed with pme grid 80 80 200, coulomb cutoff 1.376: 2832.6 M-cycles
                    optimal pme grid 96 96 224, coulomb cutoff 1.212
          DD  step 4999  vol min/aver 0.777  load imb.: force  0.6%
          DD  step 9999  vol min/aver 0.769  load imb.: force  1.1%
```

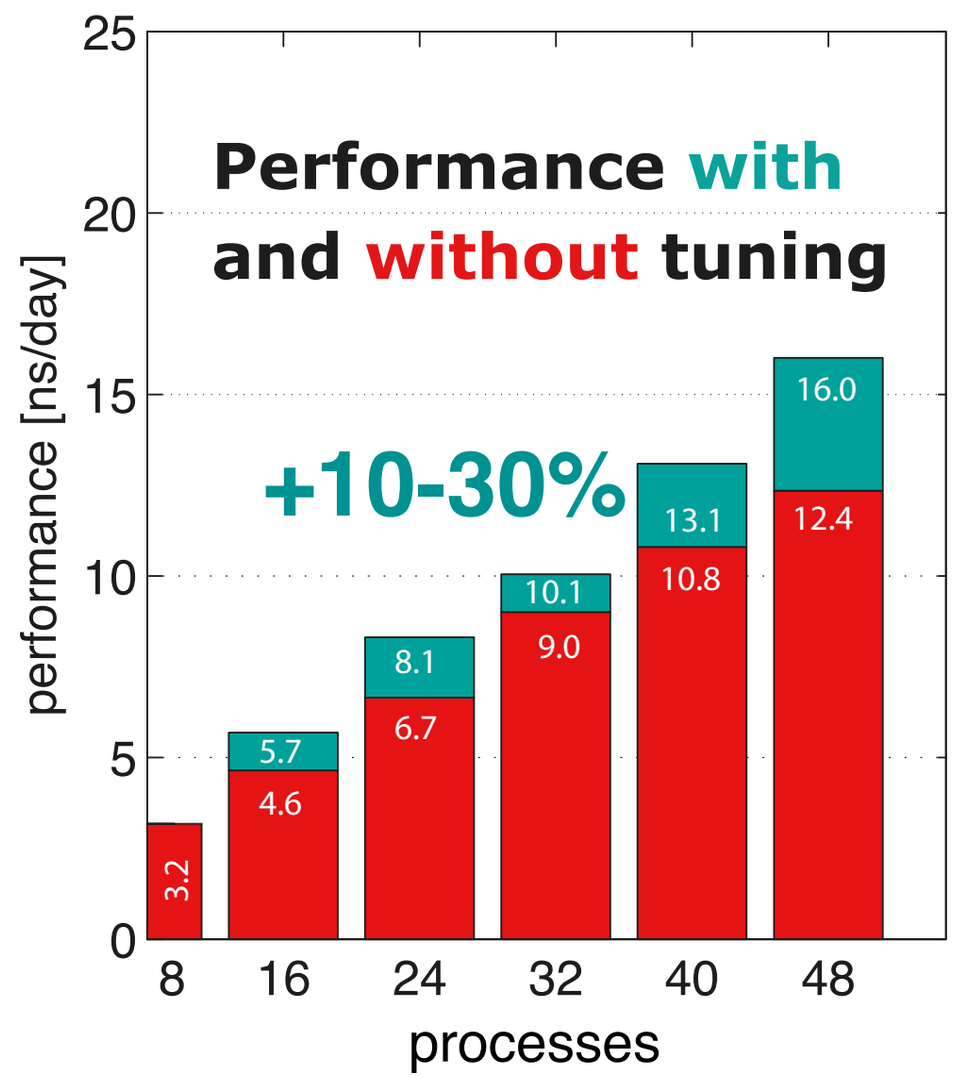
The optimal SR : LR process ratio (CPU nodes)

- GROMACS estimates **SR** : **LR** load, chooses near-optimal setting, based on cutoff + grid settings, but cannot know about network
- e.g.
  12 SR + **4 LR** for 16 MPI processes
- **`gmx tune_pme`** tries settings around this value, e.g.
  14 : **2**
  13 : **3**
  12 : **4** *
  11 : **5**
  10 : **6**
  16 : **0** (no separate LR processes)
- For > 8 MPI ranks on CPU nodes (single or multiple nodes), usually separate PME nodes perform better



The optimal SR : LR process ratio (CPU nodes)

- GROMACS estimates **SR** : **LR** load, chooses near-optimal setting, based on cutoff + grid settings, but cannot know about network
- e.g.
  12 SR + **4 LR** for 16 MPI processes
- **`gmx tune_pme`** tries settings around this value, e.g.
  14 : **2**
  13 : **3**
  12 : **4** *
  11 : **5**
  10 : **6**
  16 : **0** (no separate LR processes)
- For > 8 MPI ranks on CPU nodes (single or multiple nodes), usually separate PME nodes perform better

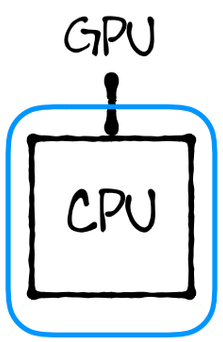


DPPC, 2fs, 120k atoms, 8 core intel Harpertown nodes, IB

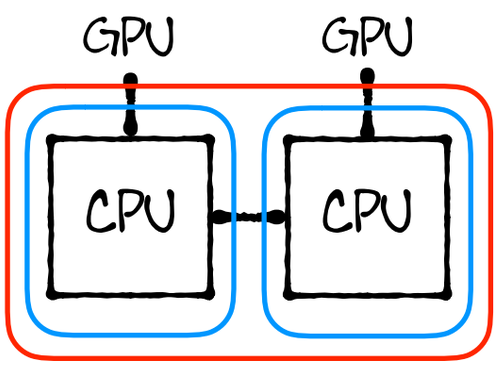
The optimal mix of threads & ranks (single node)

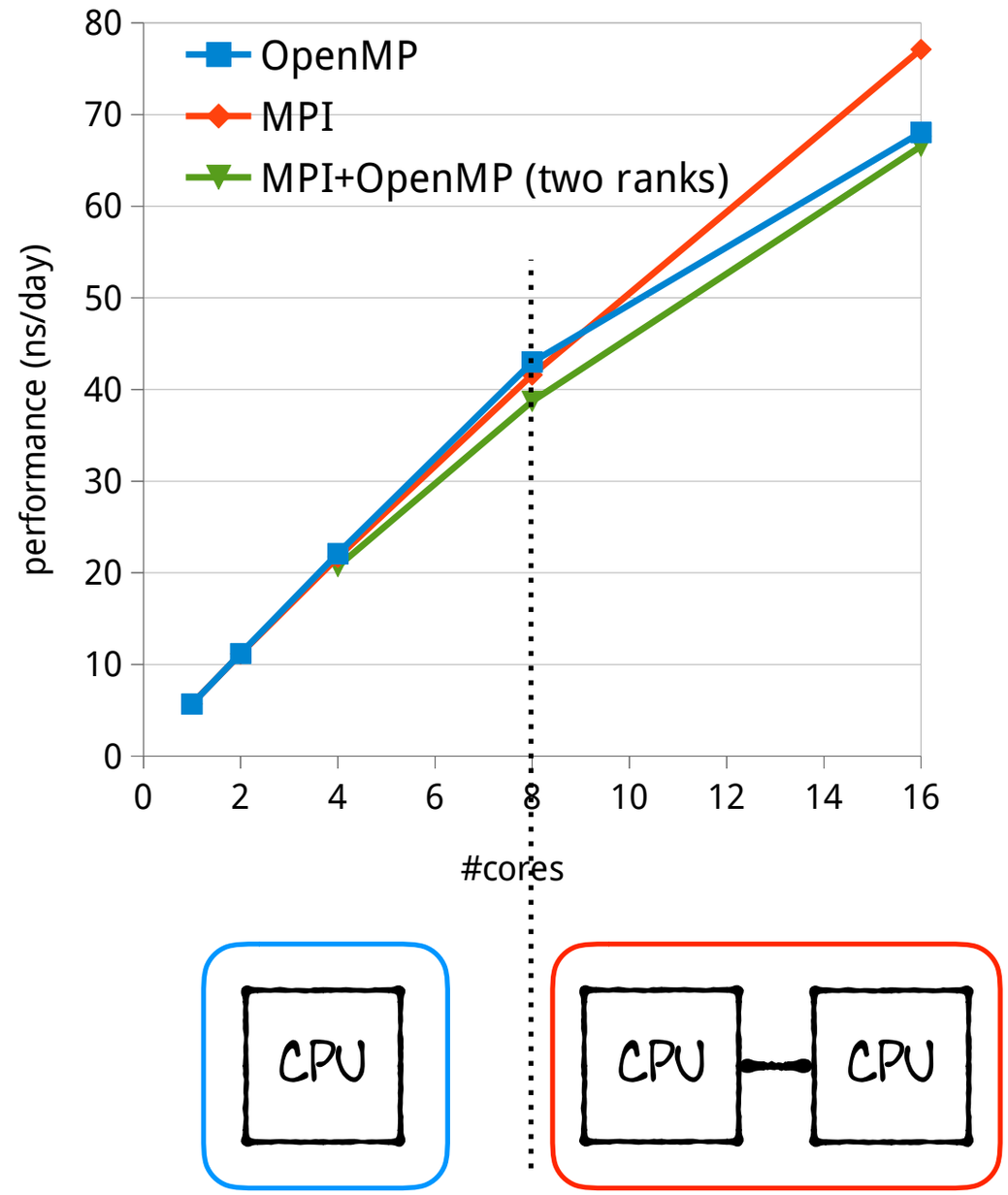
- MPI + OpenMP
  → work can be distributed in various ways
- pure OpenMP performs well on single nodes, but does not scale well across sockets
- → on multi-socket nodes pure MPI is best
- OpenMP+MPI adds overhead

- With **GPUs** it is beneficial to have few large domains offloading their data to the GPU
  → use pure OpenMP



- Multi-socket GPU nodes
  → find optimum!

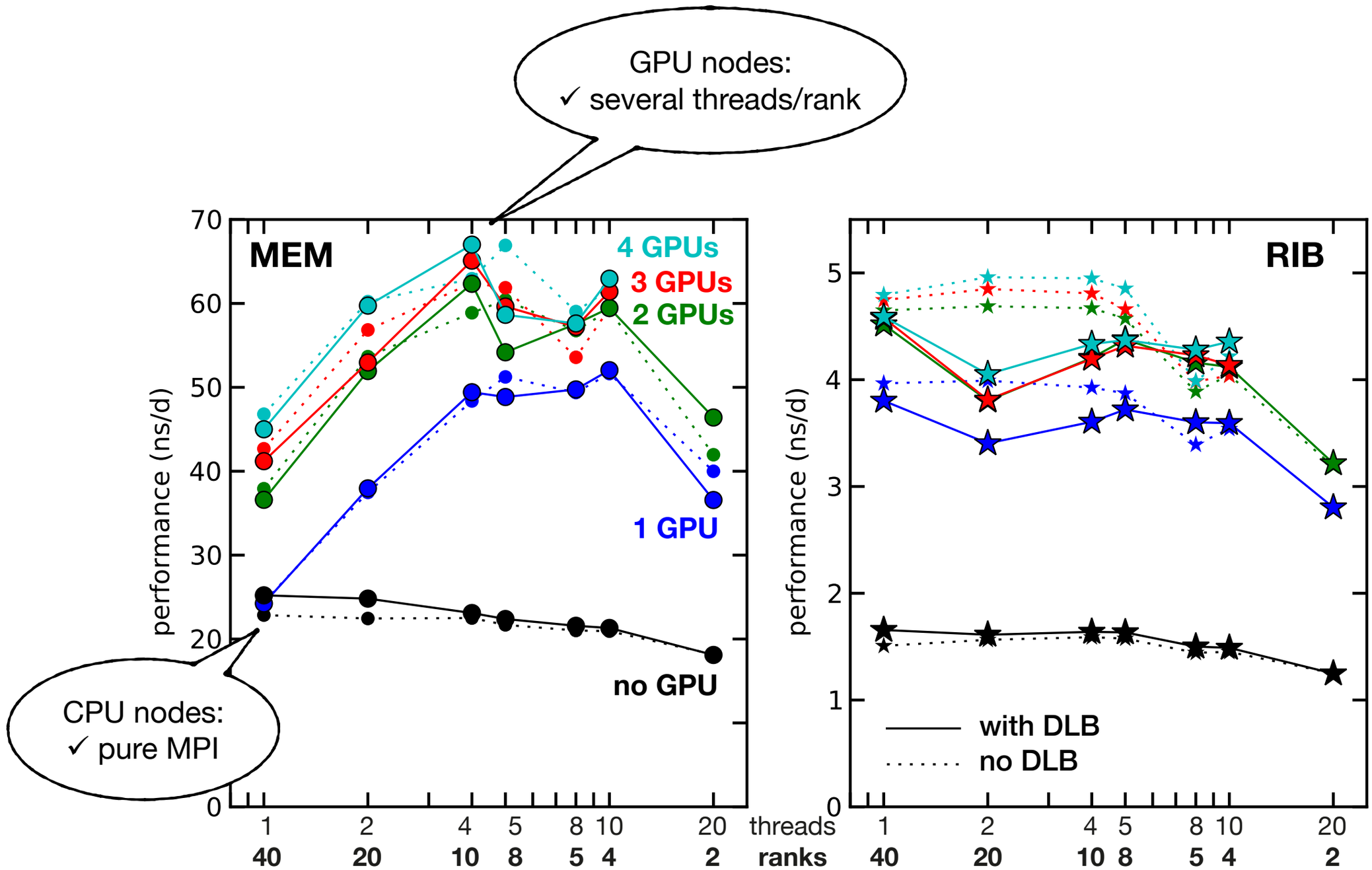




2x 8-core E5-2690 (Sandy Bridge), RNAse protein, solvated, 24k atoms, PME, 0.9 nm cutoffs (Fig. taken from S Pall, MJ Abraham, C Kutzner, B Hess, E Lindahl, EASC 2014, Springer, 2015)

Tip 3 of 10:

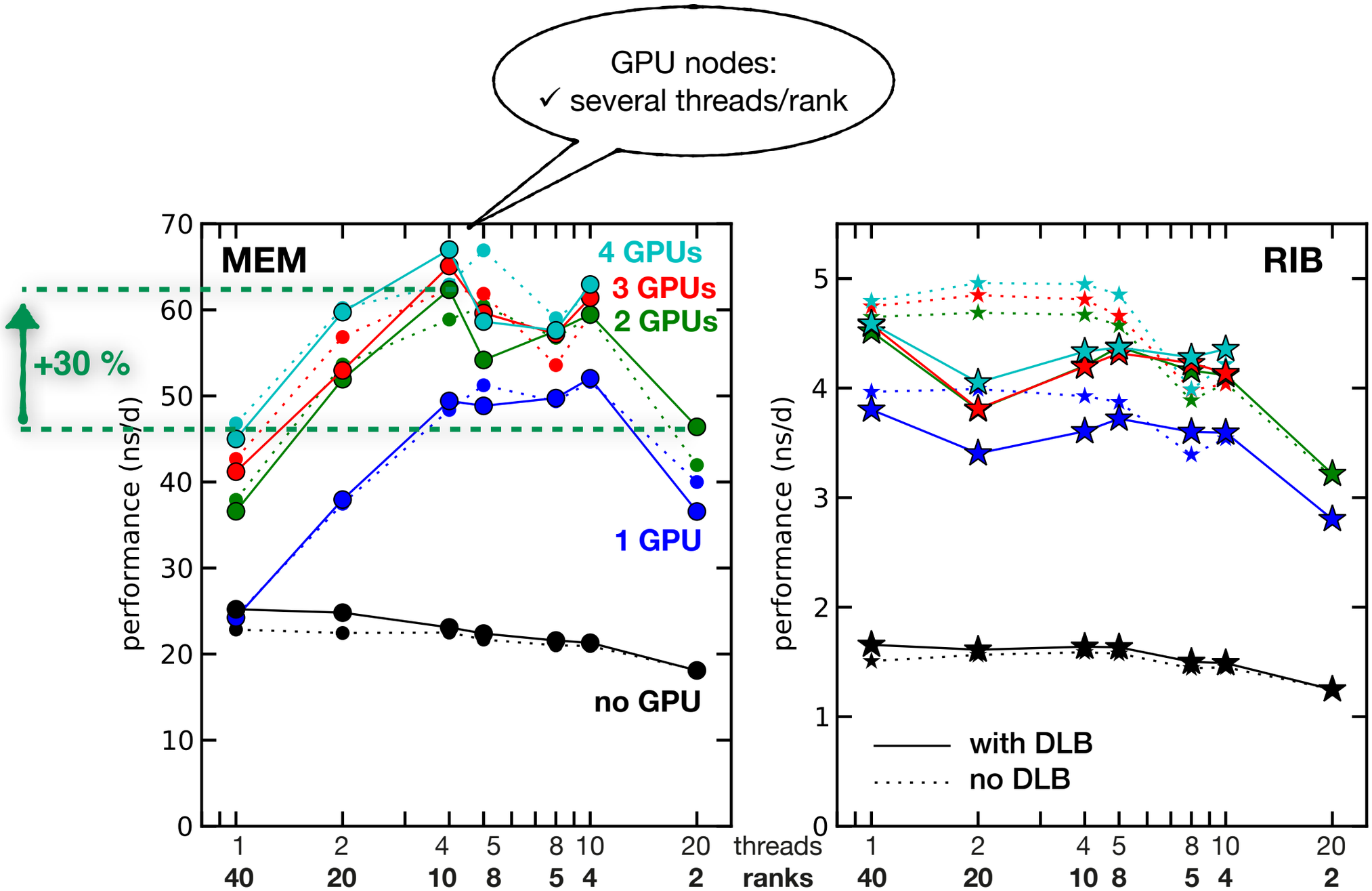
The optimal mix of threads & ranks (single node)



2x E5-2680v2 (2x 10 cores) processors with 4x GTX 980 GPUs

Tip 3 of 10:

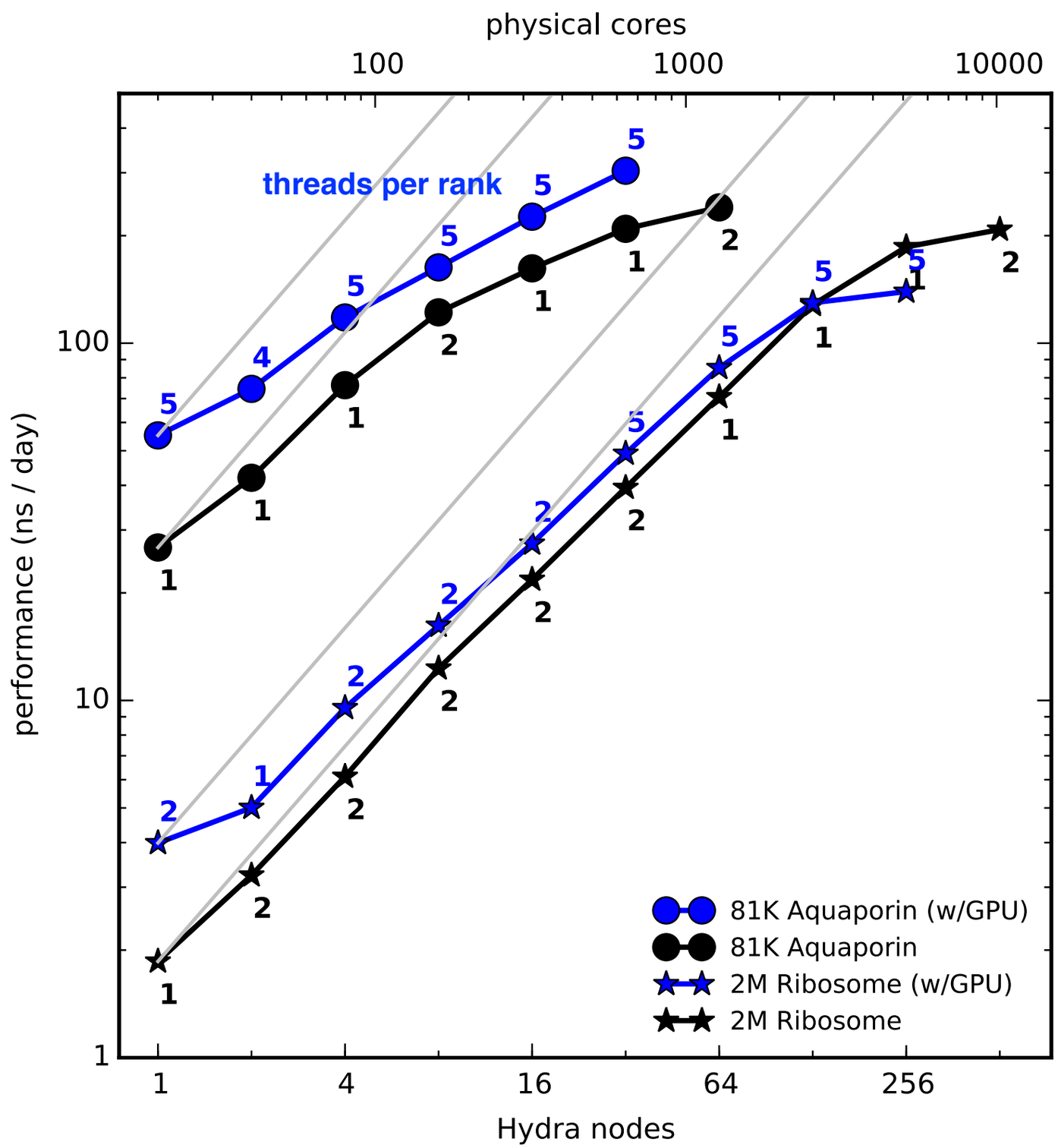
The optimal mix of threads & ranks (single node)

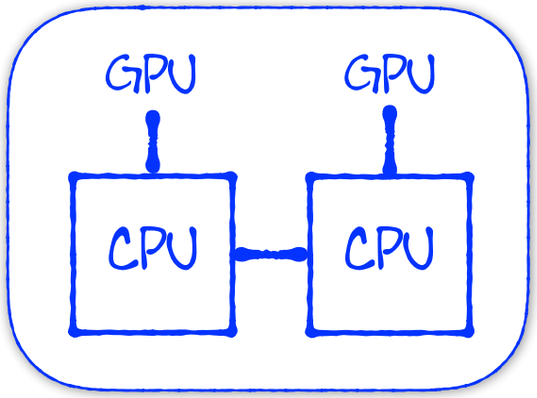


2x E5-2680v2 (2x 10 cores) processors with 4x GTX 980 GPUs

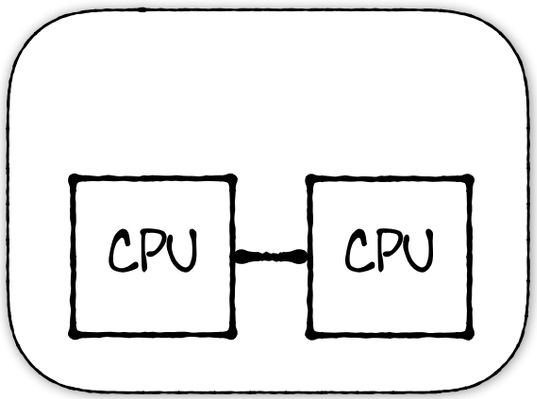
Tip 4 of 10:

The optimal mix of threads & ranks (multi node)





With GPUs:
2-5 threads per rank



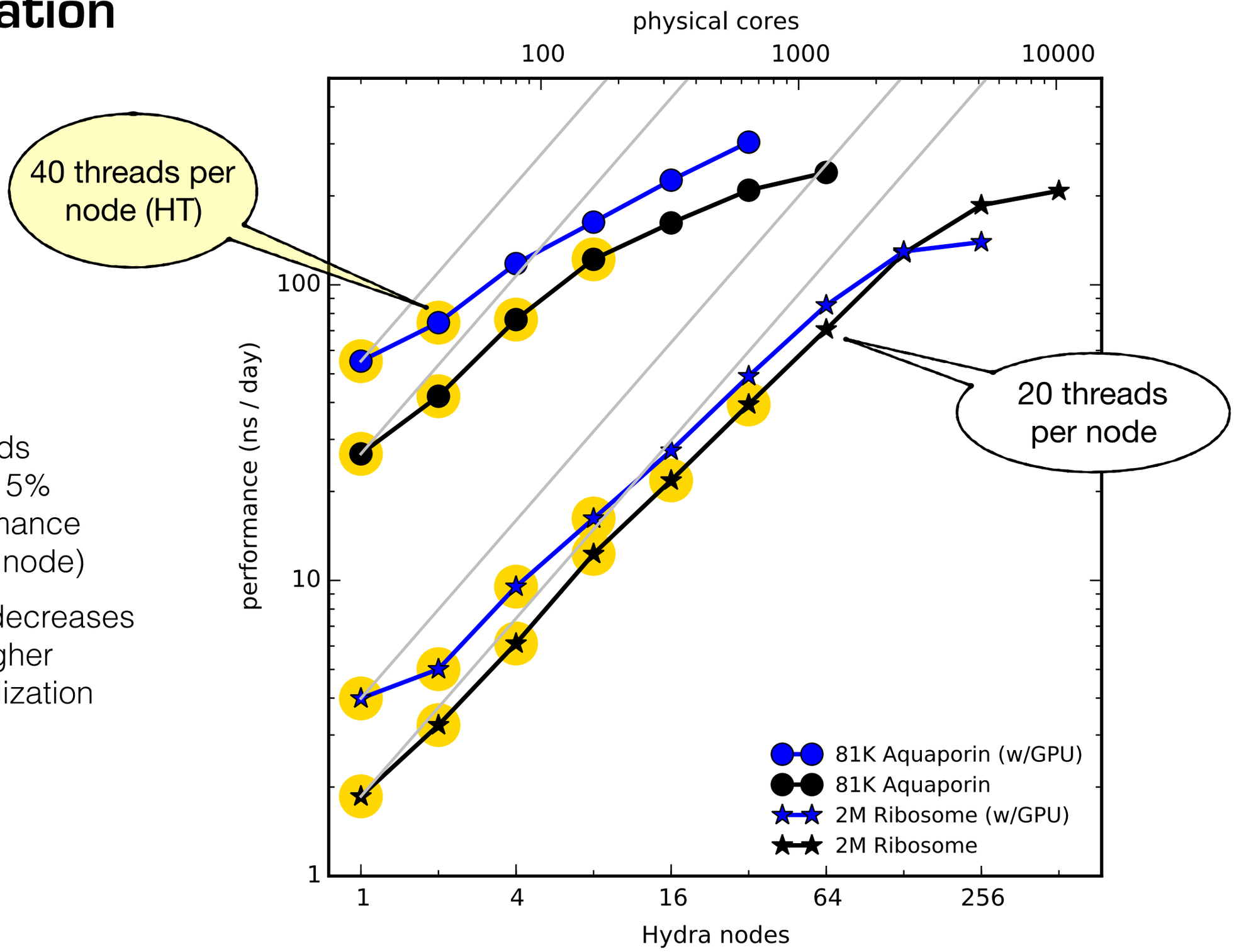
CPU nodes:
pure MPI or 2 OpenMP threads per rank

2x E5-2680v2 “Ivy Bridge” processors / node with 2x K20X GPUs, FDR-14 IB (Hydra)

Tip 5 of 10:

Hyperthreading is beneficial at moderate parallelization

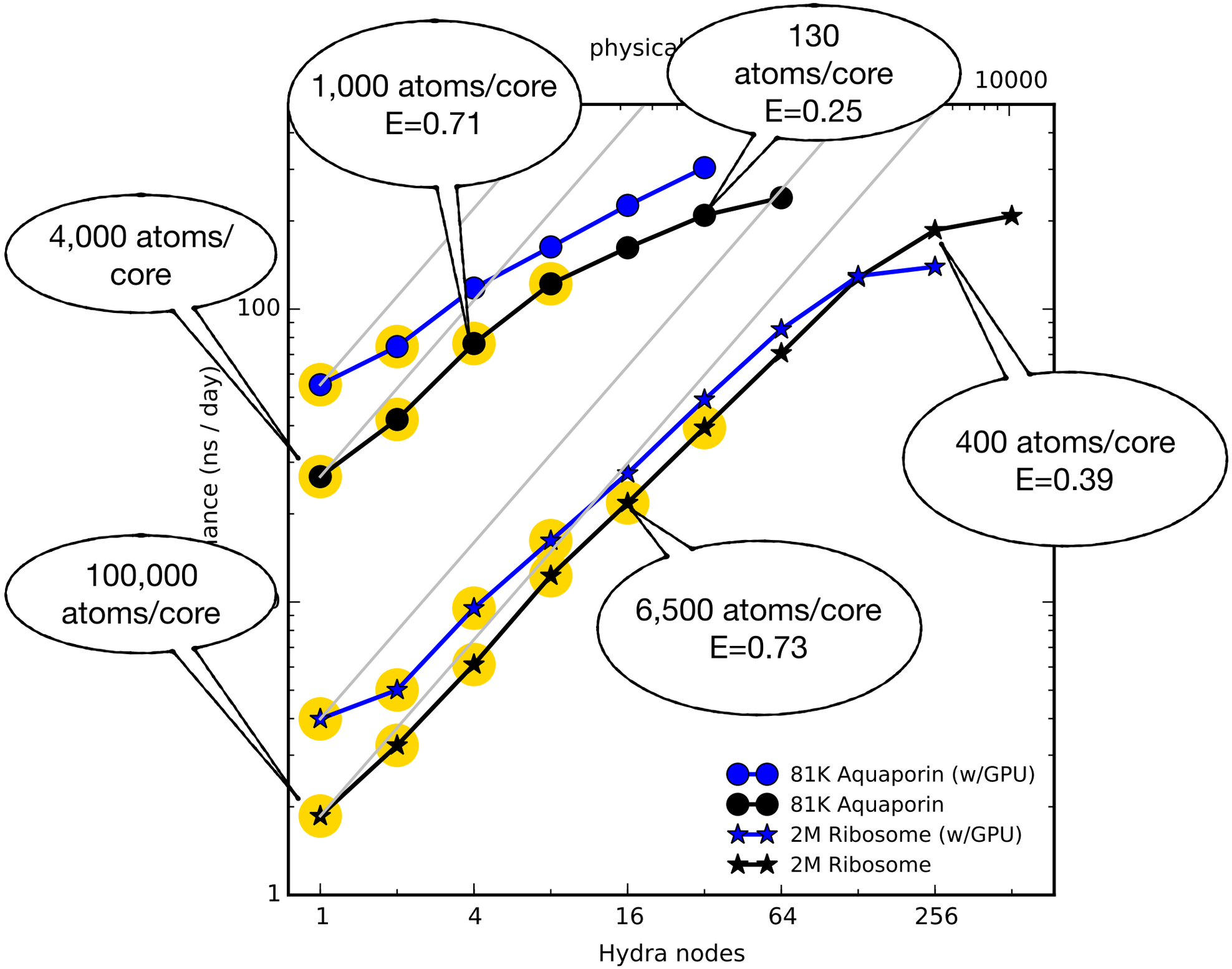
- HT yields +10 – 15% performance (single node)
- effect decreases with higher parallelization



2x E5-2680v2 “Ivy Bridge” processors / node with 2x K20X GPUs, FDR-14 IB (Hydra)

Tip 6 of 10:

>1000 atoms/core for good parallel efficiency

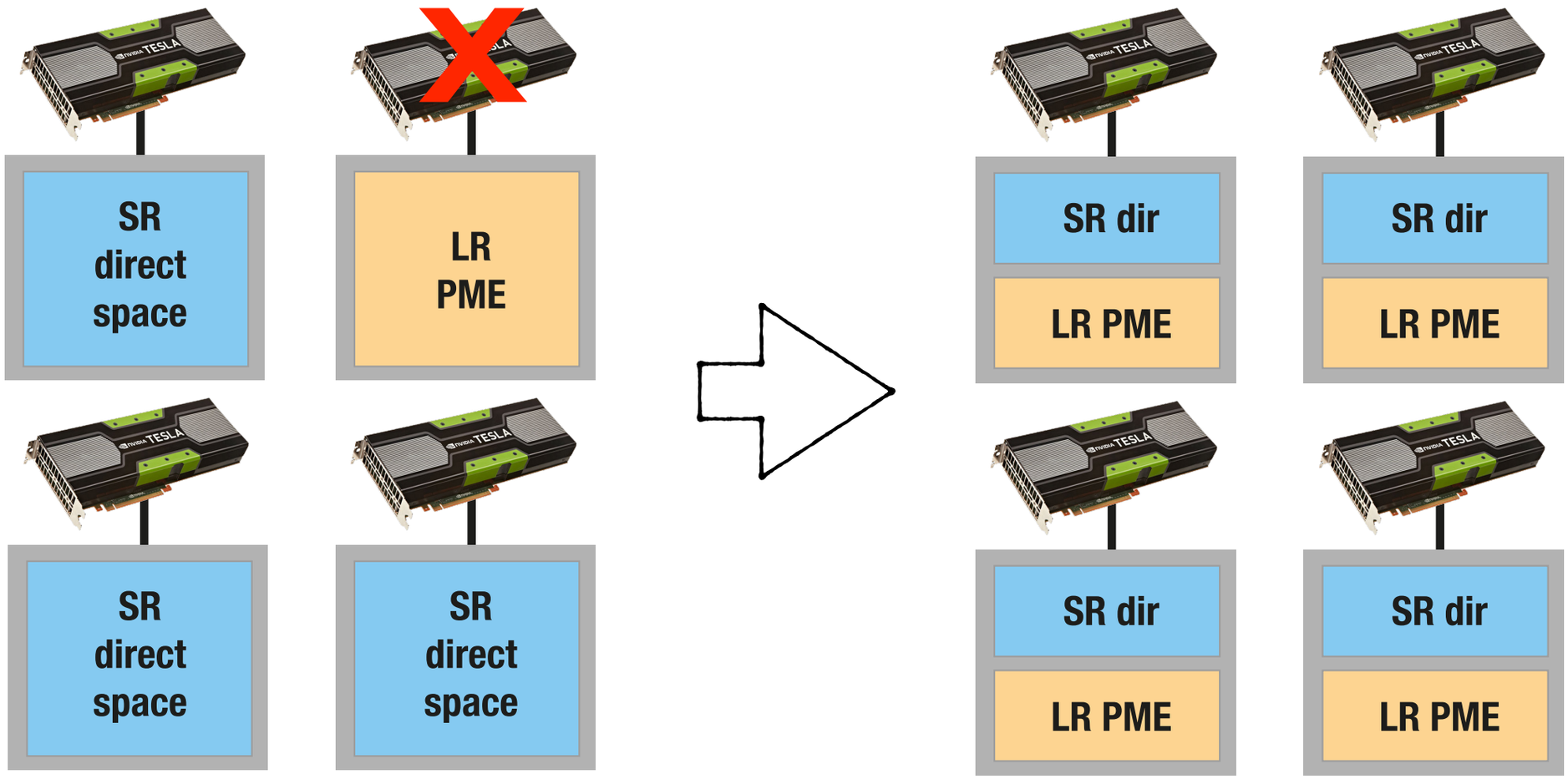


2x E5-2680v2 “Ivy Bridge” processors / node with 2x K20X GPUs, FDR-14 IB (Hydra)

Tip 7 of 10:

Separate LR PME nodes on GPU nodes

- usual approach would leave GPUs unused
- assigning half of the ranks to LR PME and interleave PME : PP nodes

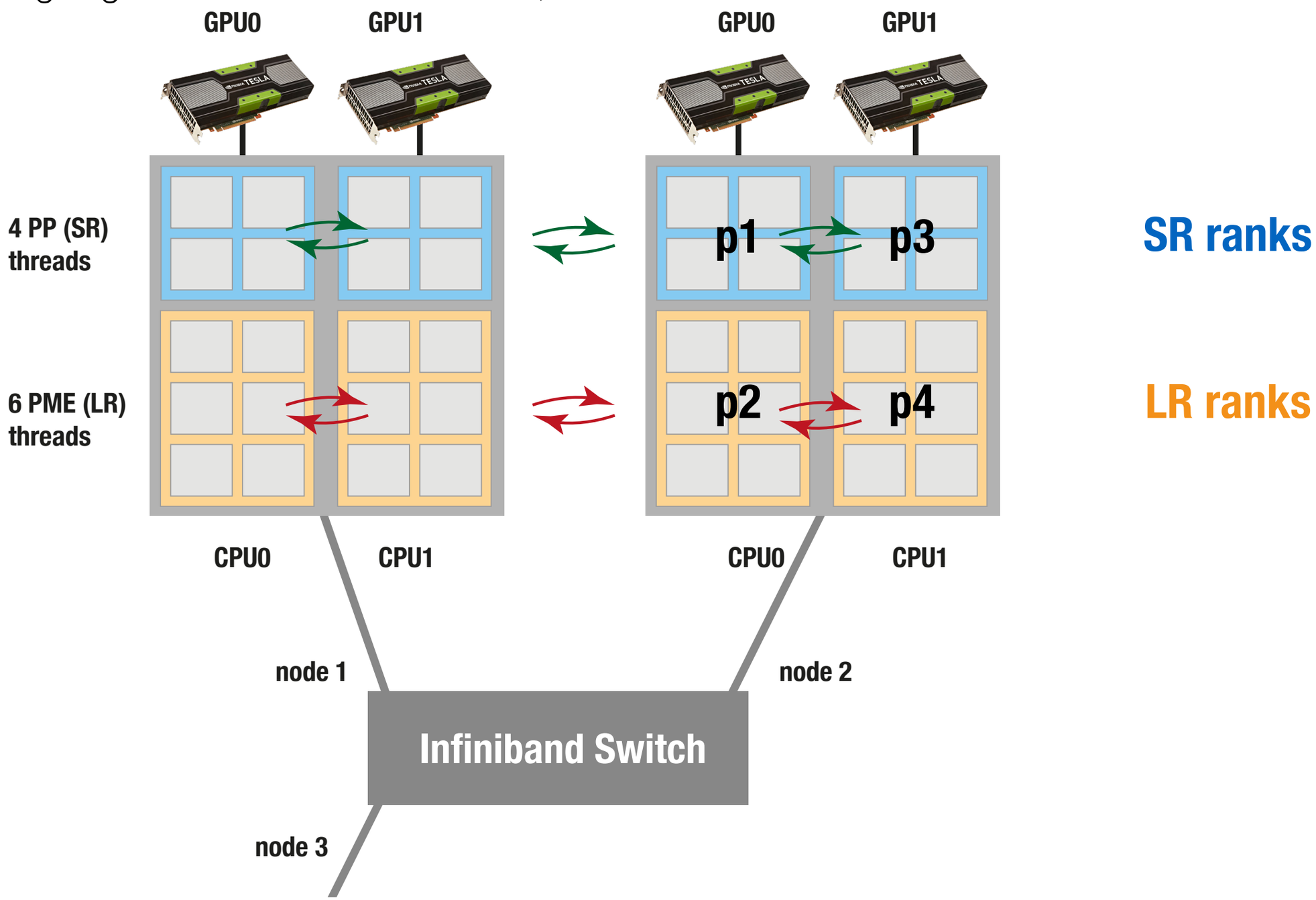



```
mpirun -np 64 mdrun_mpi -npme 32
```

Tip 7 of 10:

Separate LR PME nodes on GPU nodes

- assigning half of the ranks to LR PME, balance LR:SR load via threads




```
mpirun -np 128 mdrun_mpi -npme 64 -ntomp 4 -ntomp_pme 6 -gpu_id 01 …
```

Impact of the compiler

- recent gcc's >= 4.7 perform best
- can make a 25% difference

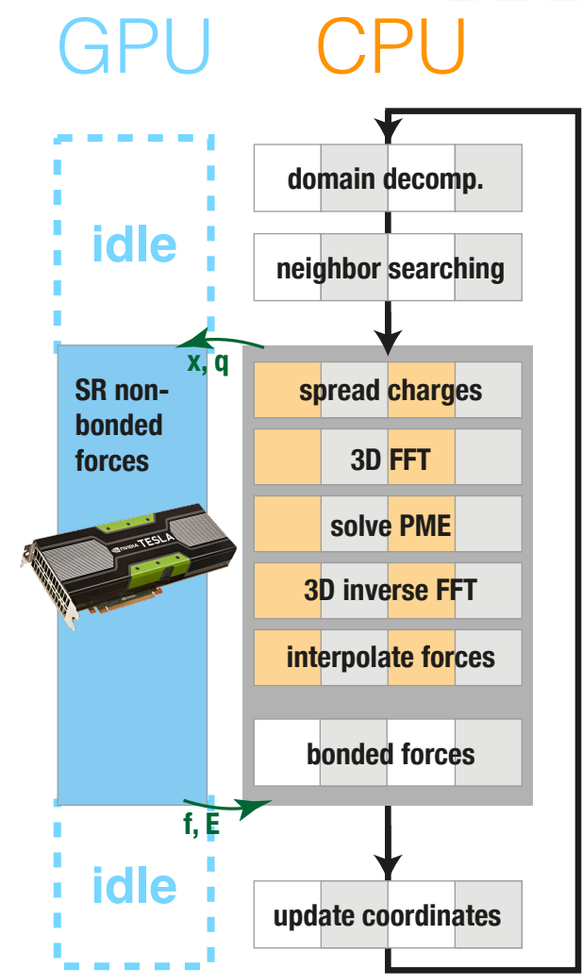
optimized performance!

Hardware	Compiler	MEM (ns/d)	RIB (ns/d)	av. speedup (%)
AMD 6380 × 2	GCC 4.4.7	14	0.99	0
	GCC 4.7.0	15.6	1.11	11.8
	GCC 4.8.3	16	1.14	14.7
	ICC 13.1	12.5	0.96	−6.9
AMD 6380 × 2	GCC 4.4.7	40.5	3.04	0
with 2× GTX 980$^+$	GCC 4.7.0	38.9	3.09	−1.2
	GCC 4.8.3	40.2	3.14	1.3
	ICC 13.1	39.7	3.09	−0.2
Intel E5-2680v2 × 2	GCC 4.4.7	21.6	1.63	0
	GCC 4.8.3	26.8	1.86	19.1
	ICC 13.1	24.6	1.88	14.6
	ICC 14.0.2	25.2	1.81	13.9
Intel E5-2680v2 × 2	GCC 4.4.7	61.2	4.41	0
with 2× GTX 980$^+$	GCC 4.8.3	62.3	4.69	4.1
	ICC 13.1	60.3	4.78	3.5

Tip 9 of 10:

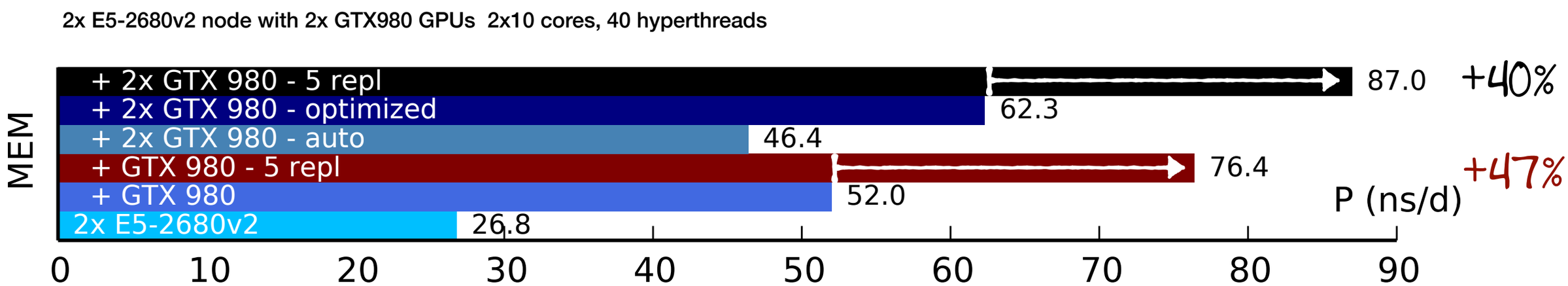
Multi-simulations enhance throughput

- The GPU is typically idle for 15 – 40 % of a time step
- Multi-simulation = running several replicas of a system
  `mpirun -np 4 mdrun -multi 4 -gpu_id 0011 -s in.tpr`
- SR non bonded forces can **interlock** on GPUs so that aggregated performance is higher
- + benefits from **higher efficiency** at lower parallelization



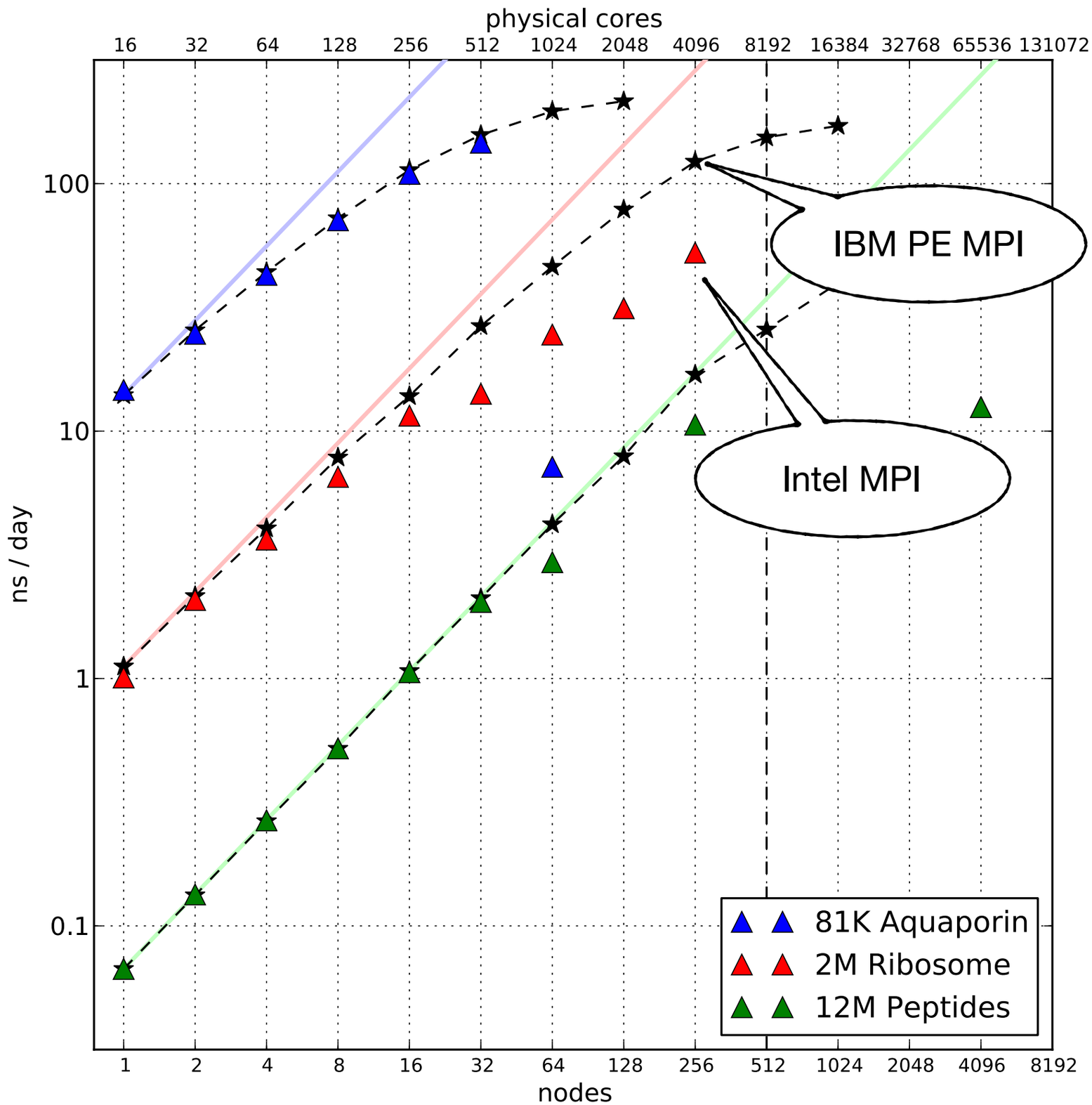
Tip 9 of 10:

Multi-simulations enhance throughput

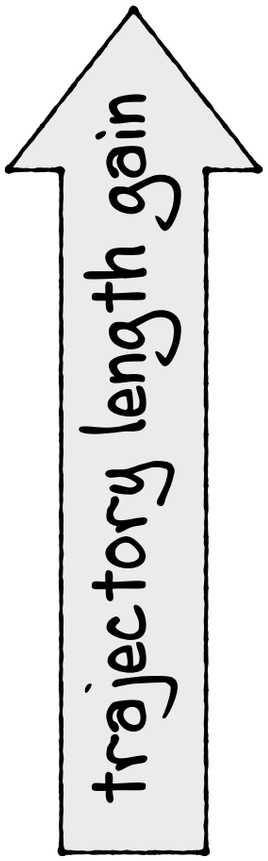


Tip 10 of 10:

Check which MPI library performs best



Q1 summary



	typical trajectory gain
virtual sites	x 2
multi simulations	50+ %
optimizing threads per rank on GPU nodes	20 – 40 %
Determine optimal # of PME nodes on CPU nodes using gmx tune_pme	10 – 30 %
Compiler	up to 20 % on CPU nodes
hyper threading	10 – 15%

Q2.

What is the ‘optimal’ hardware to run GROMACS on?

with S Páll, M Fechner, A Esztermann, BL de Groot and H Grubmüller

‘optimal’ in terms of … ?

- performance-to-price ratio
- achievable single-node performance
- parallel performance “time-to-solution”
- energy consumption “energy-to-solution”
- rack space requirements

Q2.

What is the ‘optimal’ hardware to run GROMACS on?

with S Páll, M Fechner, A Esztermann, BL de Groot and H Grubmüller

- Our goal: Cost-efficient simulations.
  Maximize MD trajectory on a fixed budget

- Method: Determine price + performance for >50 hardware configurations, 2 MD systems, 12 CPU types, 13 GPU types

- determine ‘optimal’ performance per node type
  - optimize threads x ranks
  - optimize number of LR PME nodes
  - use HT, where beneficial

GPUs used in the test nodes

2014

NVIDIA model	architec-ture	CUDA cores	clock rate (MHz)	memory (GB)	SP throughput (Gflop/s)	≈ price (€) (net)
Tesla K20X[a]	Kepler GK110	2,688	732	6	3,935	2,800
Tesla K40[a]	Kepler GK110	2,880	745	12	4,291	3,100
GTX 680	Kepler GK104	1,536	1,058	2	3,250	300
GTX 770	Kepler GK104	1,536	1,110	2	3,410	320
GTX 780	Kepler GK110	2,304	902	3	4,156	390
GTX 780Ti	Kepler GK110	2,880	928	3	5,345	520
GTX Titan	Kepler GK110	2,688	928	6	4,989	750
GTX Titan X	Maxwell GM200	3,072	1,002	12	6,156	
GTX 970	Maxwell GM204	1,664	1,050	4	3,494	250
GTX 980	Maxwell GM204	2,048	1,126	4	4,612	430
GTX 980[+]	Maxwell GM204	2,048	1,266	4	5,186	450
GTX 980[‡]	Maxwell GM204	2,048	1,304	4	5,341	450

MEM uses 50 MB of GPU RAM, RIB 225 MB

Consumer GPU error rates

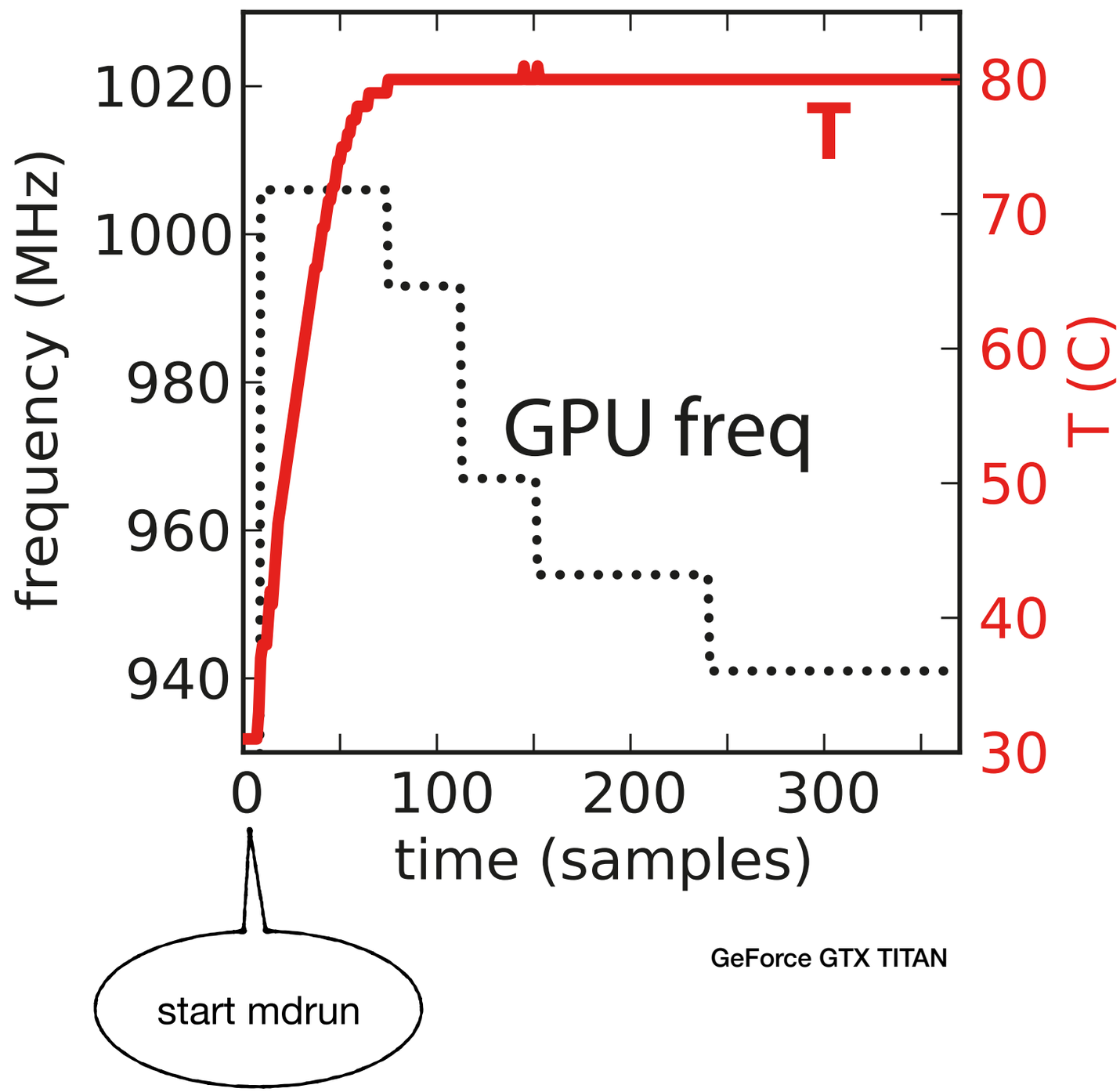
- consumer GPUs do not have ECC memory, thus cannot correct for rare bit-flips
- however, GPU stress tests can be used to sort out problematic GPUs

NVIDIA model	GPU memory checker[13]	# of cards tested	# memtest iterations	# cards with errors
GTX 580	`memtestG80`	1	$10,000$	–
GTX 680	`memtestG80`	50	$4,500$	–
GTX 770	`memtestG80`	100	$4,500$	–
GTX 780	`memtestCL`	1	$50,000$	–
GTX Titan	`memtestCL`	1	$50,000$	–
GTX 780Ti	`memtestG80`	70	$4 \times 10,000$	6
GTX 980	`memtestG80`	4	$4 \times 10,000$	–
GTX 980^{+}	`memtestG80`	70	$4 \times 10,000$	2

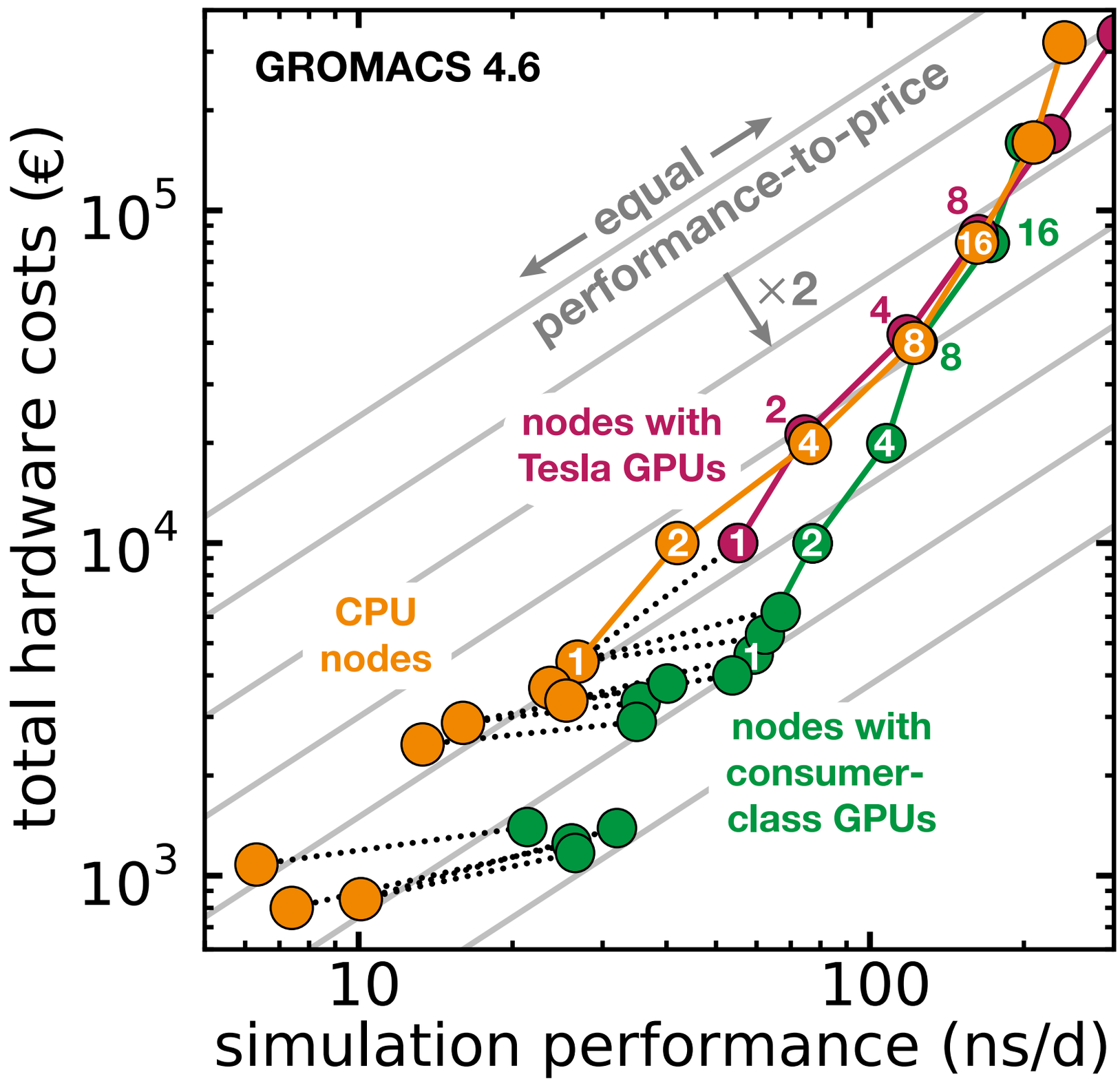
(13) I. S. Hague, V. S. Pande, In 10th IEEE/ACM International conference on cluster, cloud and grid computing: Stanford University, 2010

Consumer GPU frequency throttling due to overheating

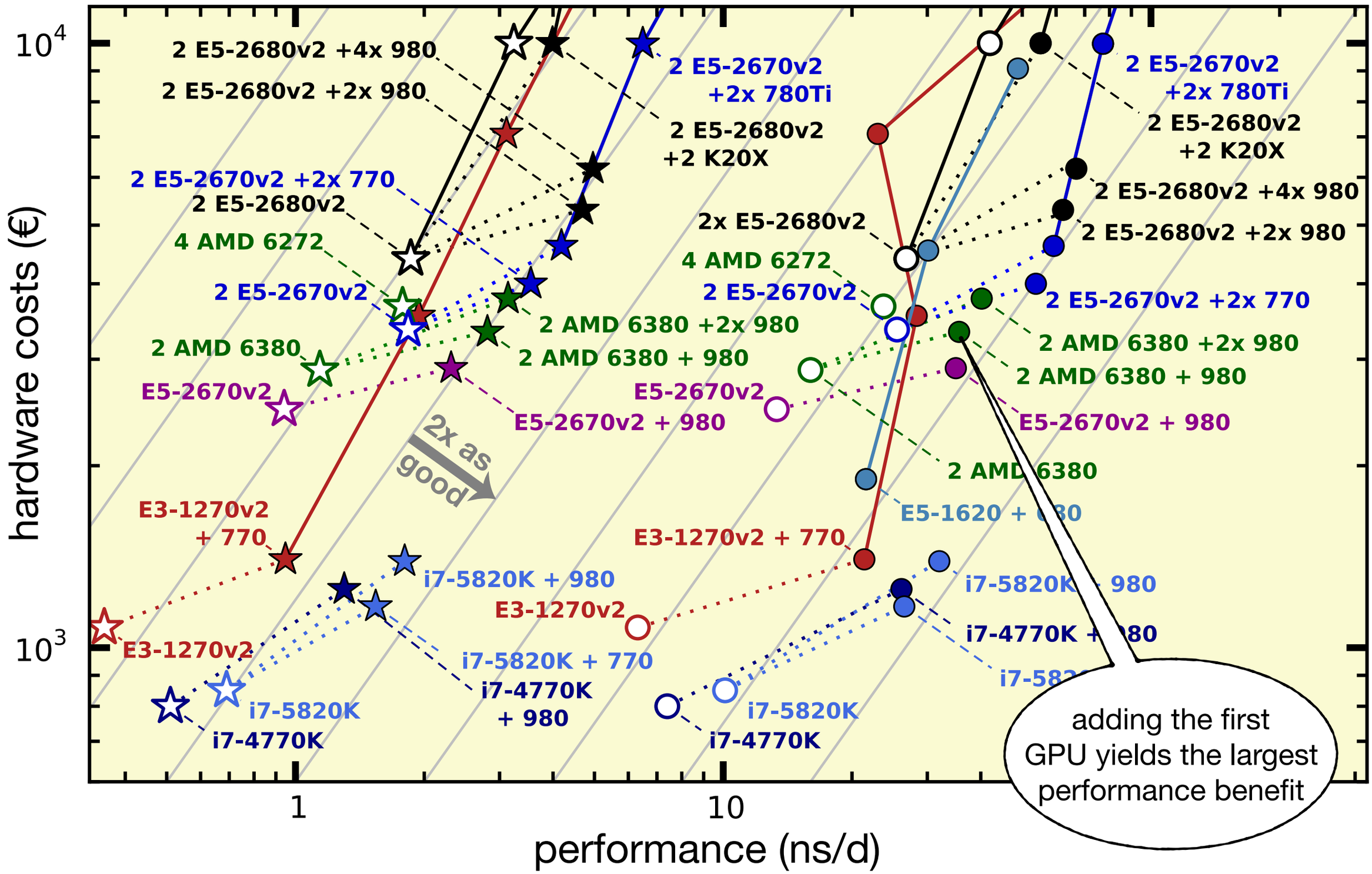
- Consumer GPUs optimized for acoustics:
  - fan speed limited to 60% of max
  - reduce GPU frequency if too hot



Performances and hardware investment

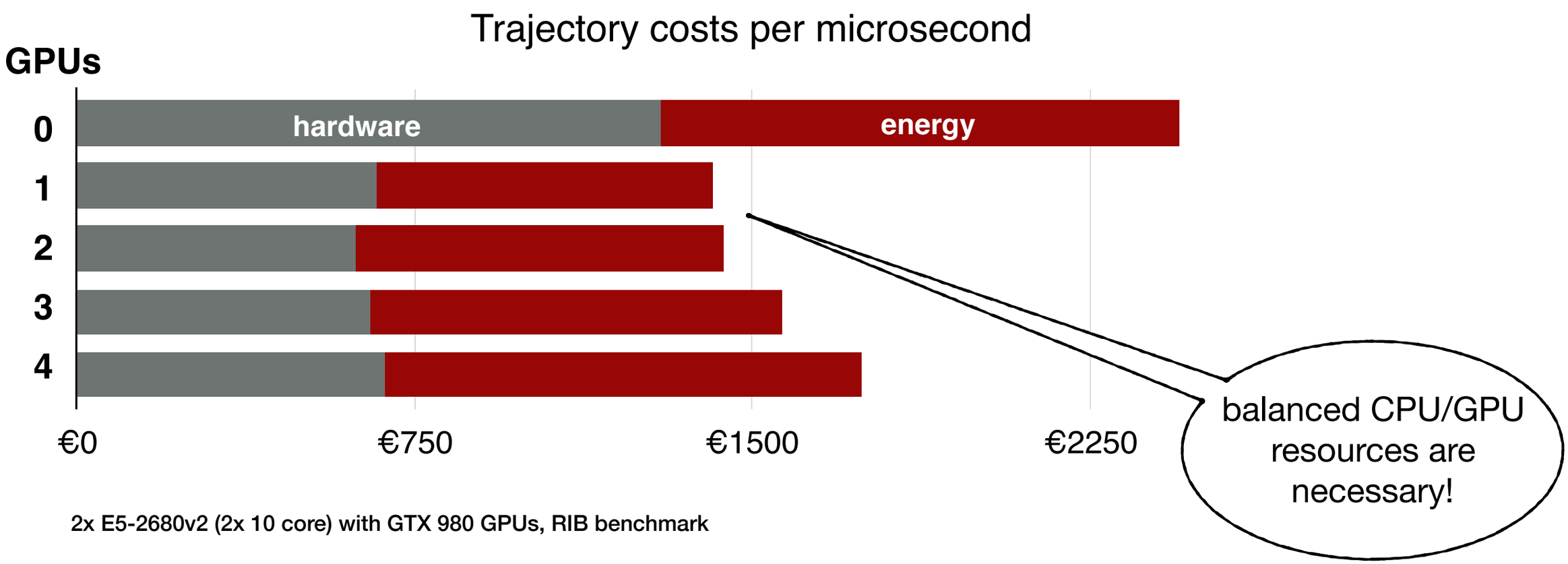


Performances and hardware investment



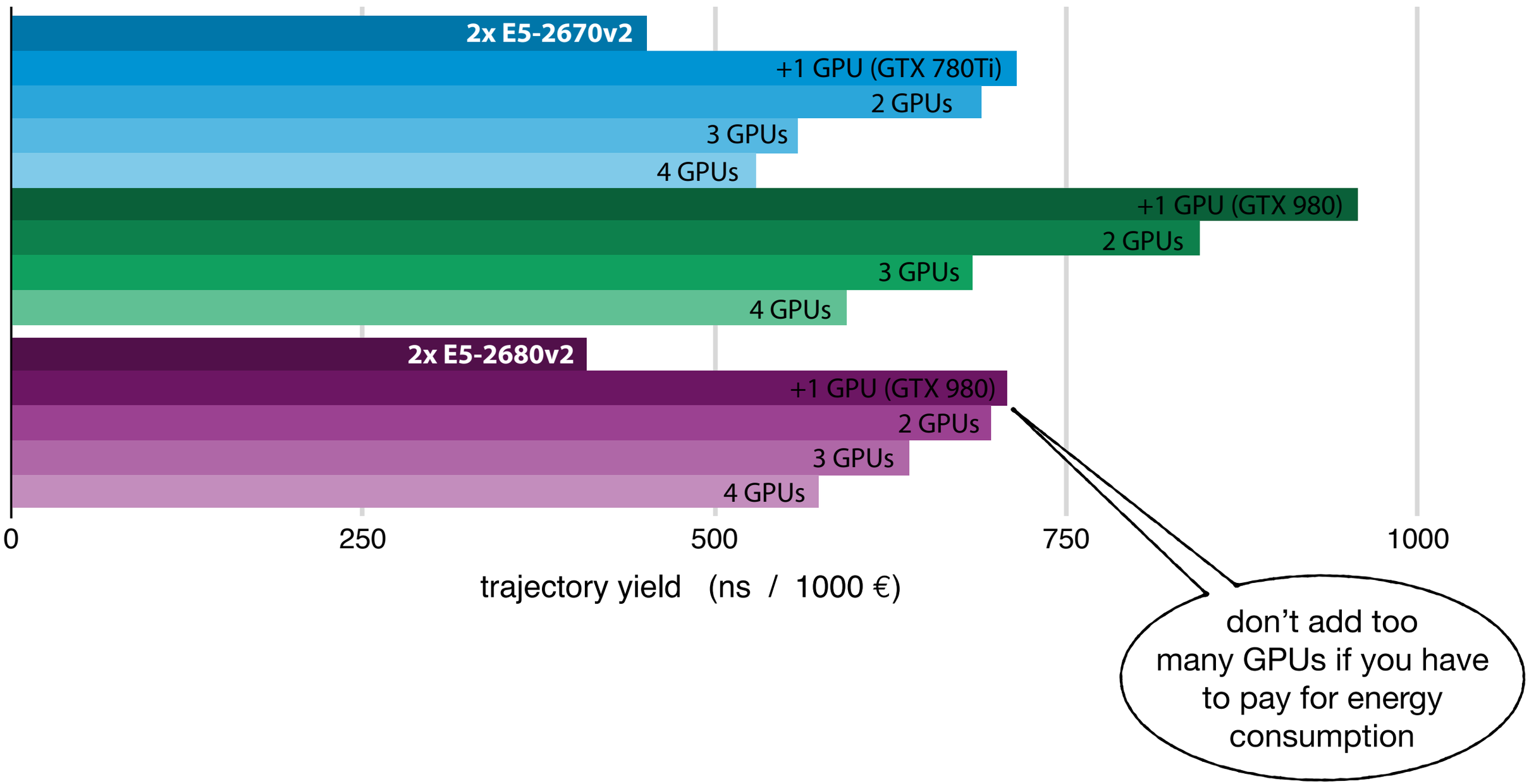
Energy efficiency

- Over cluster lifetime, energy costs become comparable to hardware costs
- assuming 5 yr of operation and 0.2 EUR / kWh (incl. cooling)



Energy efficiency

- Fixed budget trajectory yield taking into account energy + cooling (0.2 EUR / kWh) RIB



Q2 conclusions

- Nodes with 1–2 consumer-class GPUs produce >2x as much trajectory as CPU nodes or nodes with "professional" Tesla GPUs
- Highest energy efficiency for nodes with balanced CPU-GPU resources

- more details, tweaks and benchmark scripts in
  Best Bang for Your Buck:
  GPU Nodes for GROMACS Biomolecular Simulations
  C Kutzner, S Páll, M Fechner, A Esztermann, BL de Groot, H Grubmüller,
  J. Comput. Chem. 36, 1990–2008 (2015)