Webinar 20130404

Molecular Simulation with GROMACS on CUDA GPUs

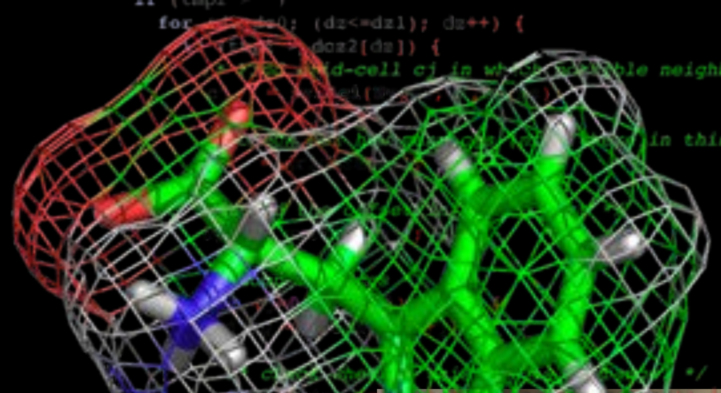
Erik Lindahl

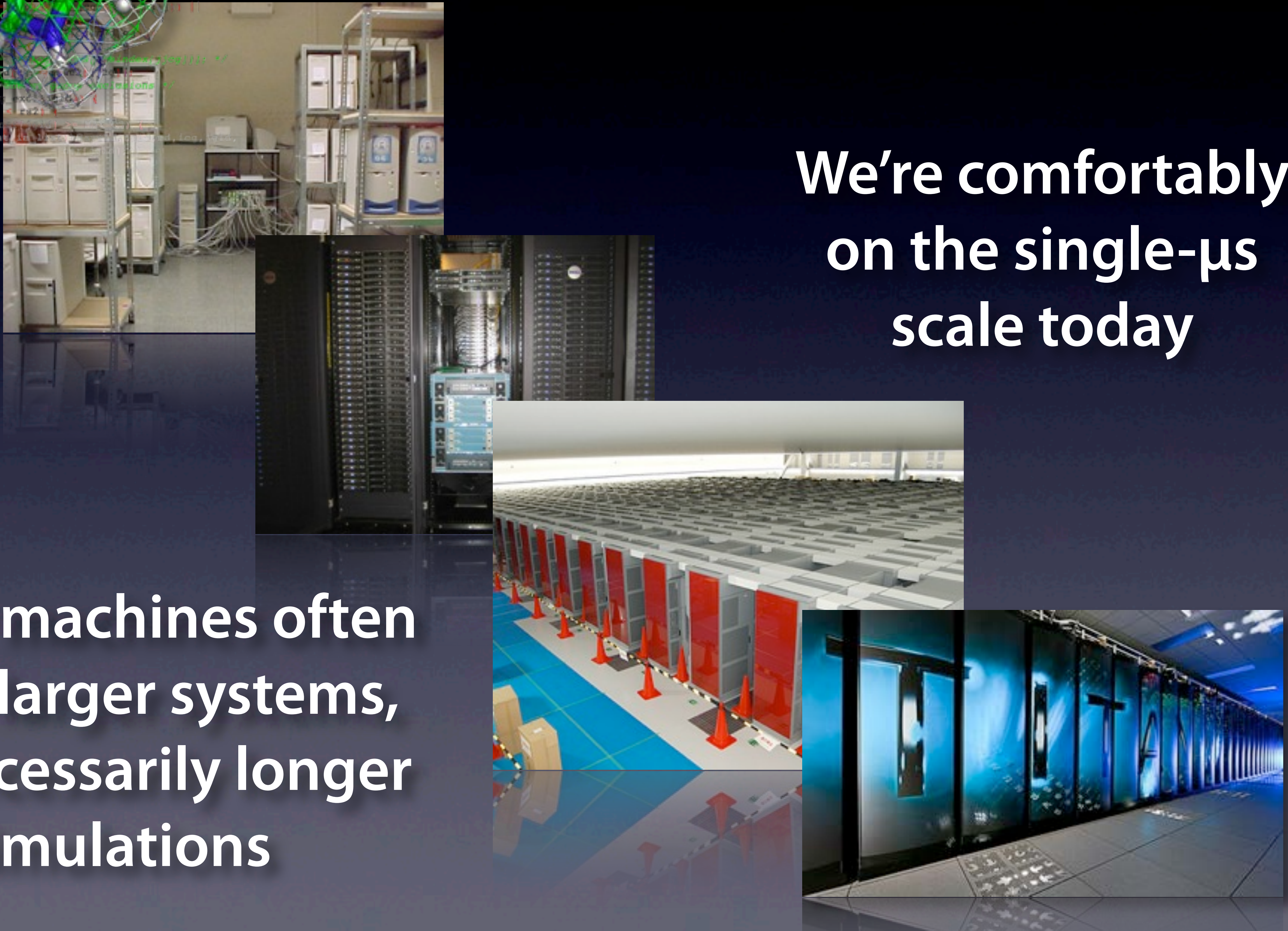
SciLifeLab
STOCKHOLM

STOCKHOLMS UNIVERSITET

KTH
VETENSKAP
OCH KONST

GROMACS is used on a wide range of resources

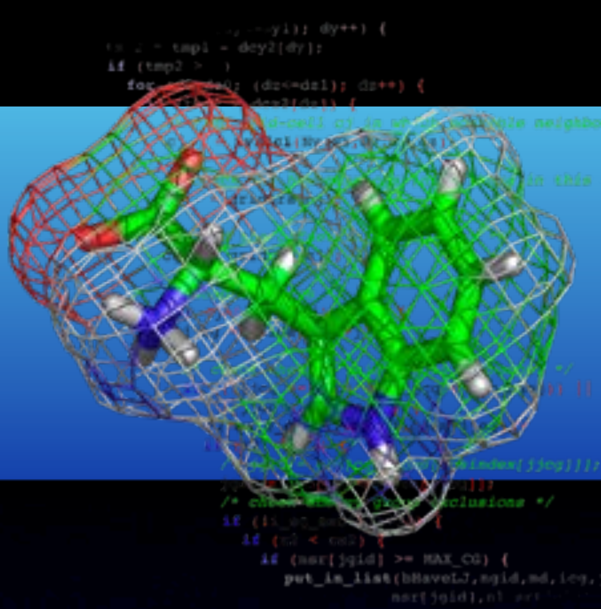




We’re comfortably on the single-μs scale today

Larger machines often mean larger systems, not necessarily longer simulations

Why use GPUs?



Throughput

- Sampling
- Free energy
- Cost efficiency
- Power efficiency
- Desktop simulation
- Upgrade old machines
- Low-end clusters

Performance

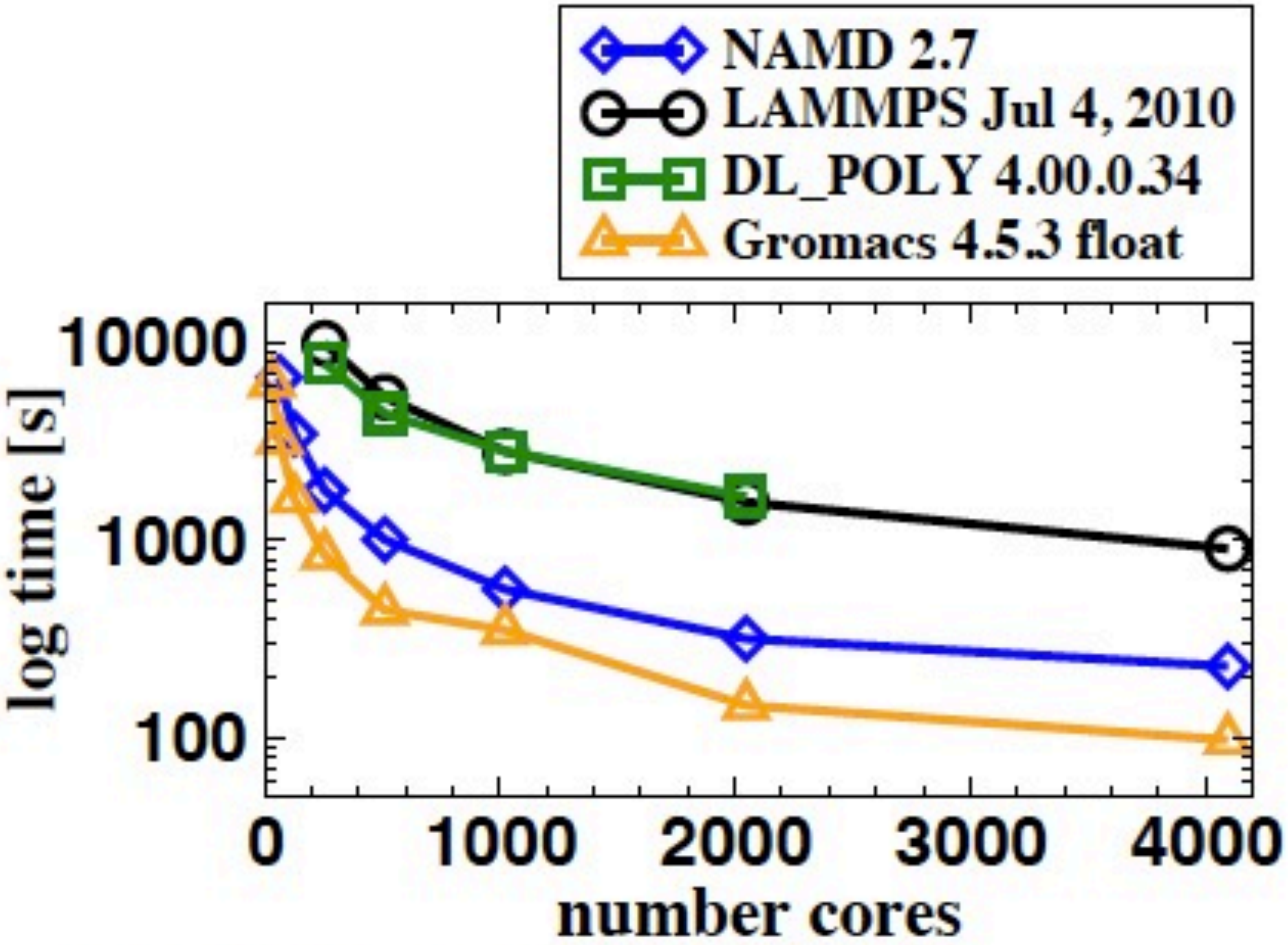
- Longer simulations
- Parallel GPU simulation using Infiniband
- High-end efficiency by using fewer nodes
- Reach timescales not possible with CPUs

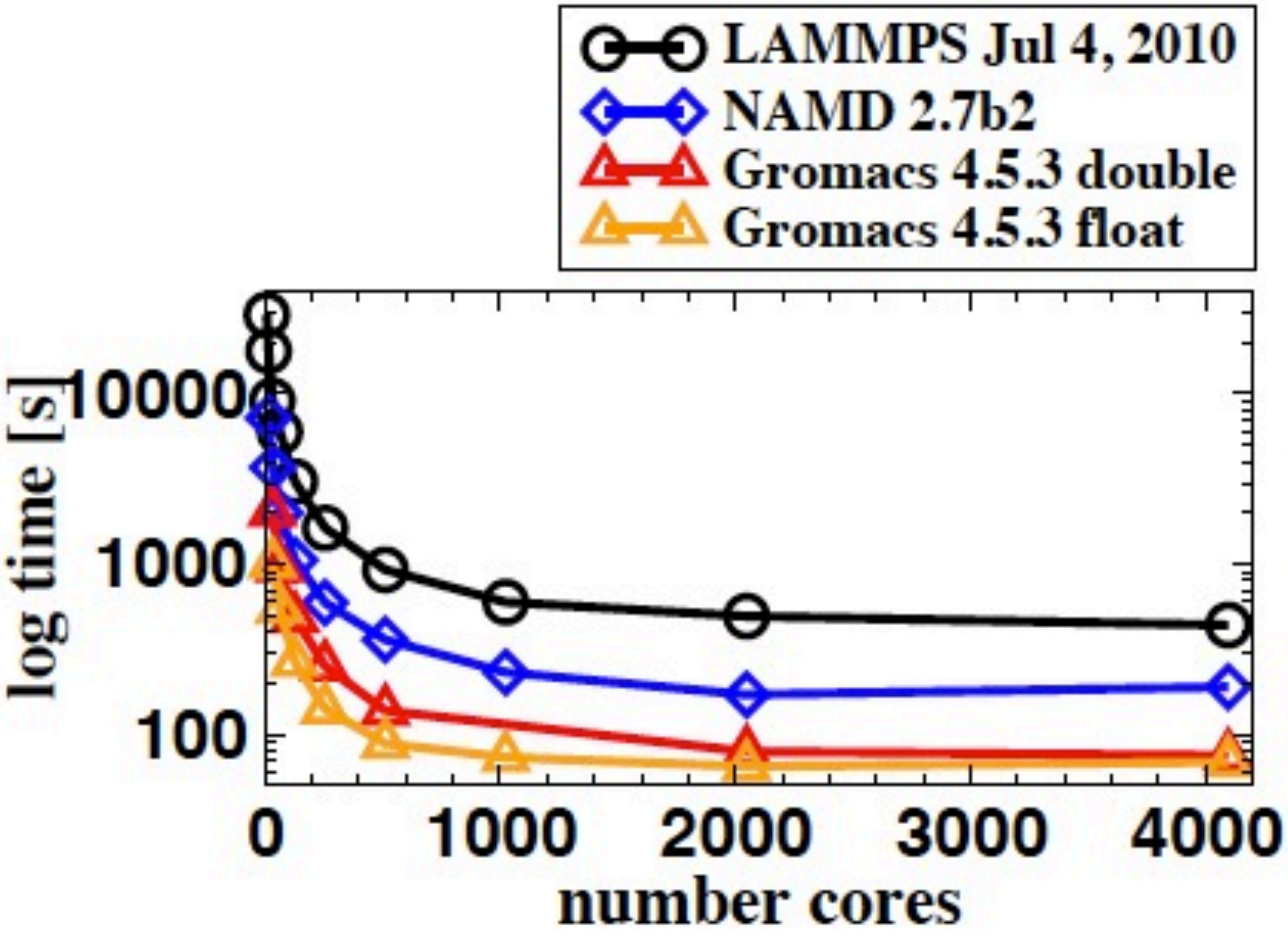
Many GPU programs today

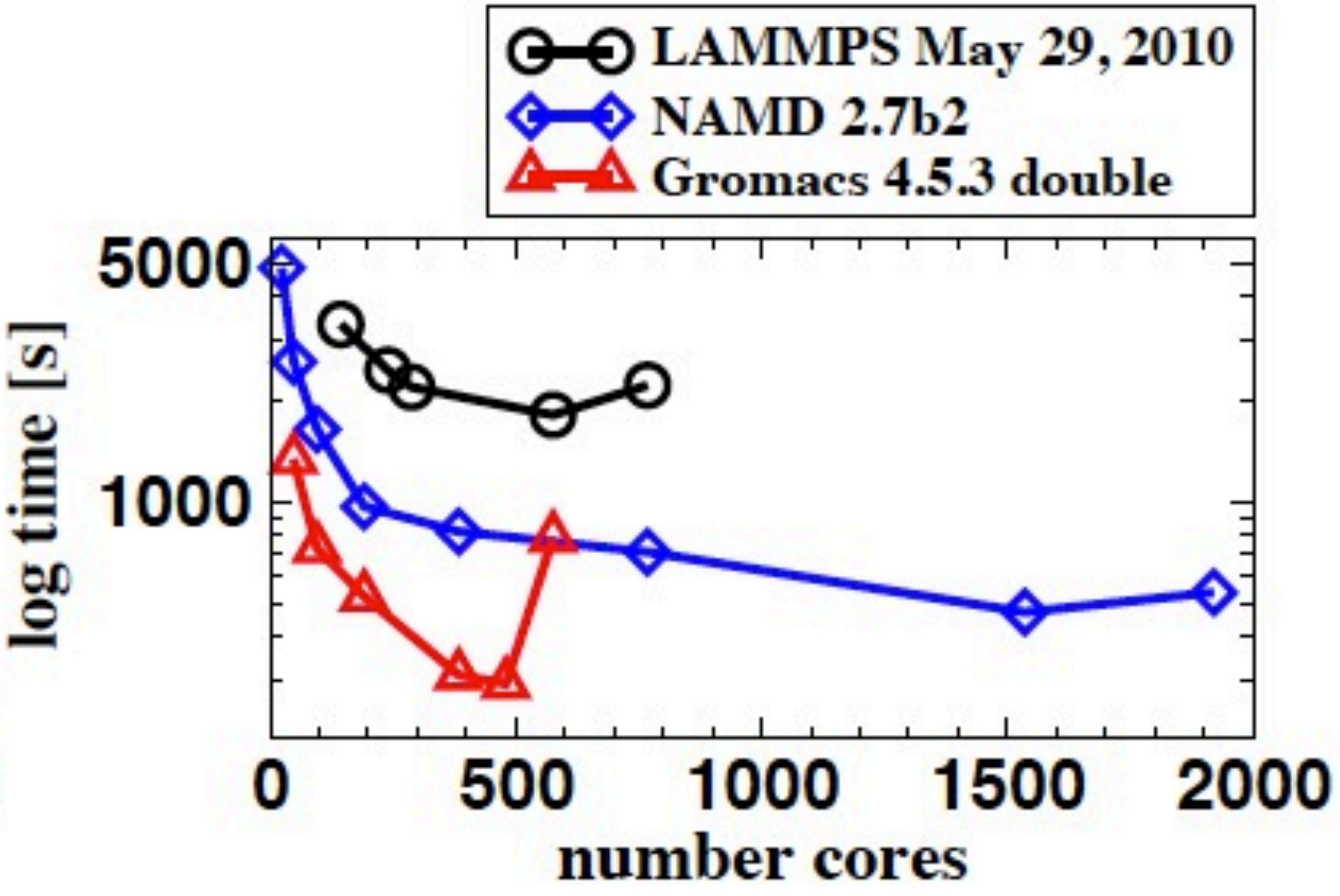
Caveat emperor:
It is much easier to get a reference problem/algorithm to scale

i.e., you see much better *relative* scaling before introducing any optimization on the CPU side

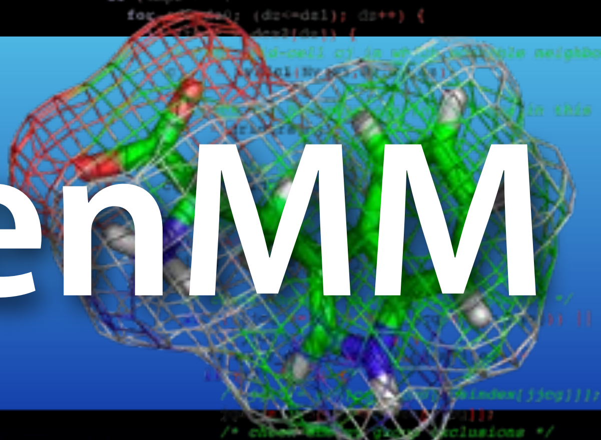
When comparing programs:
What matters is absolute performance (ns/day), not the relative speedup!


NAMD 2.7
LAMMPS Jul 4, 2010
DL_POLY 4.00.0.34
Gromacs 4.5.3 float
log time [s]
10000
1000
100
0
1000
2000
3000
4000
number cores

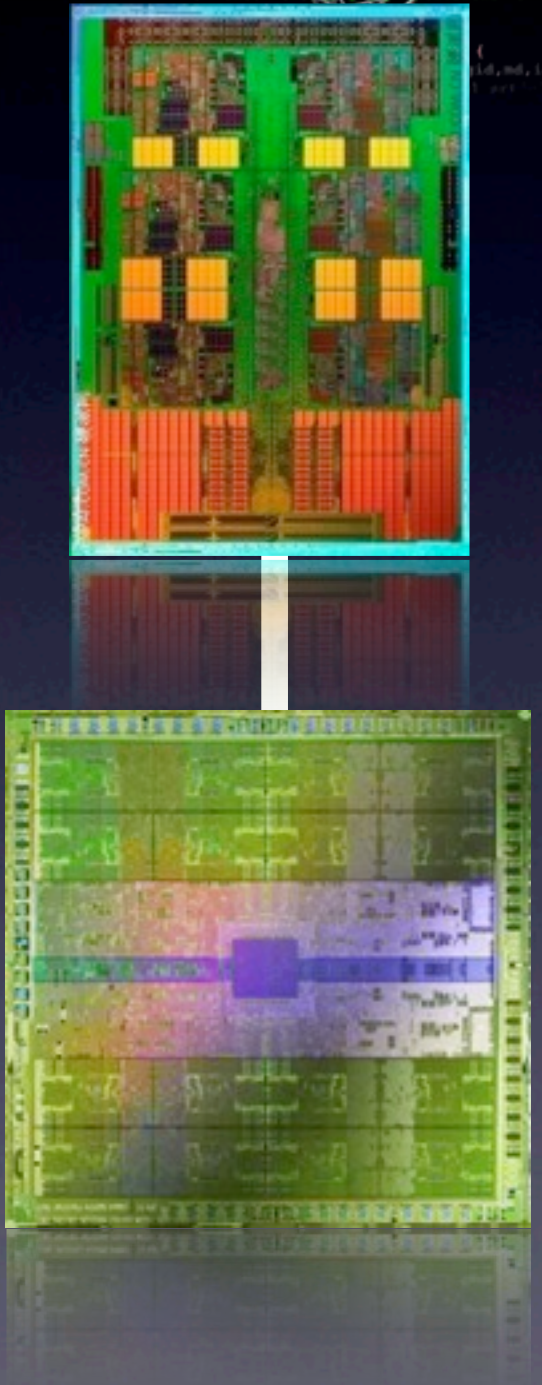

LAMMPS Jul 4, 2010
NAMD 2.7b2
Gromacs 4.5.3 double
Gromacs 4.5.3 float
log time [s]
10000
1000
100
0
1000
2000
3000
4000
number cores


LAMMPS May 29, 2010
NAMD 2.7b2
Gromacs 4.5.3 double
log time [s]
5000
1000
0
500
1000
1500
2000
number cores

Gromacs-4.5 with OpenMM



Previous version - what was the limitation?



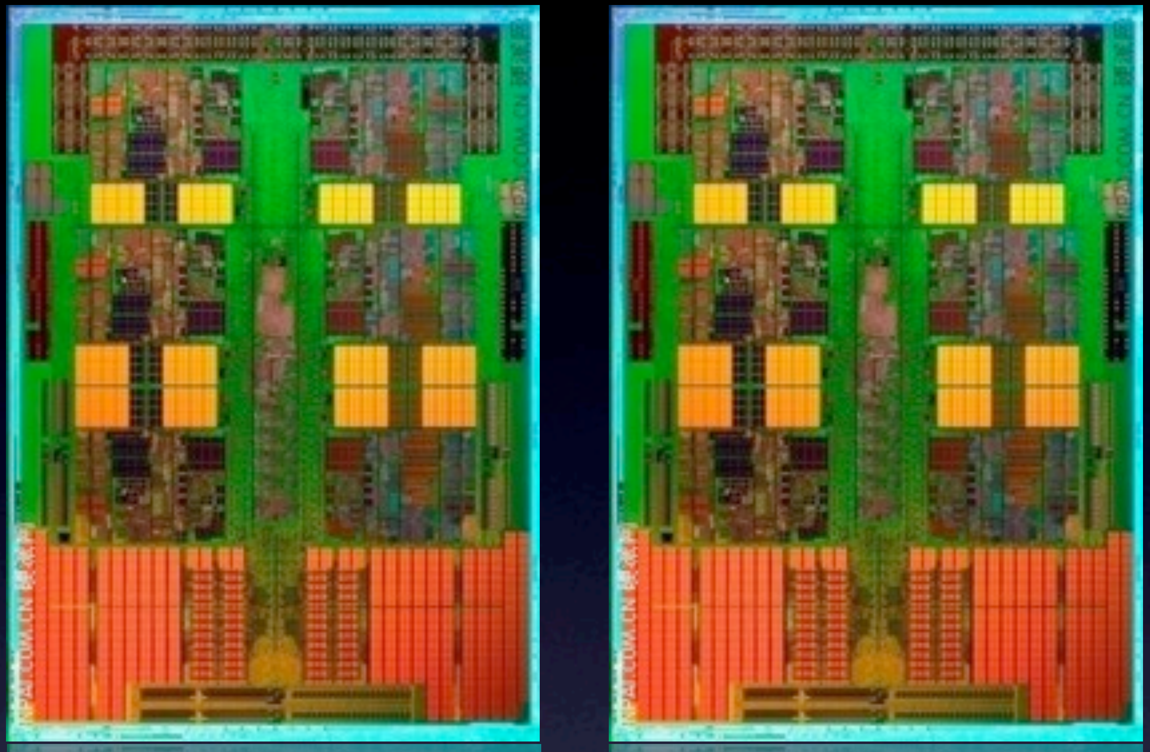
Gromacs running entirely on CPU as a fancy interface

Actual simulation running entirely on GPU using OpenMM kernels

Only a few select algorithms worked

Multi-CPU sometimes beat GPU performance...

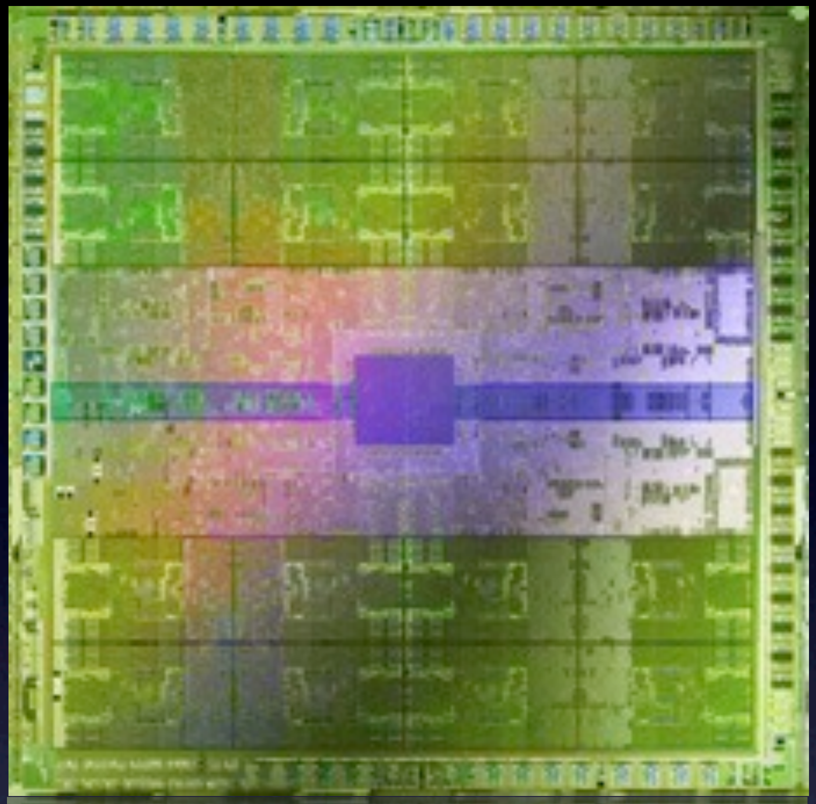
Why don't we use the CPU too?



0.5-1 TFLOP

Random memory access OK (not great)

Great for complex latency-sensitive stuff (domain decomposition, etc.)

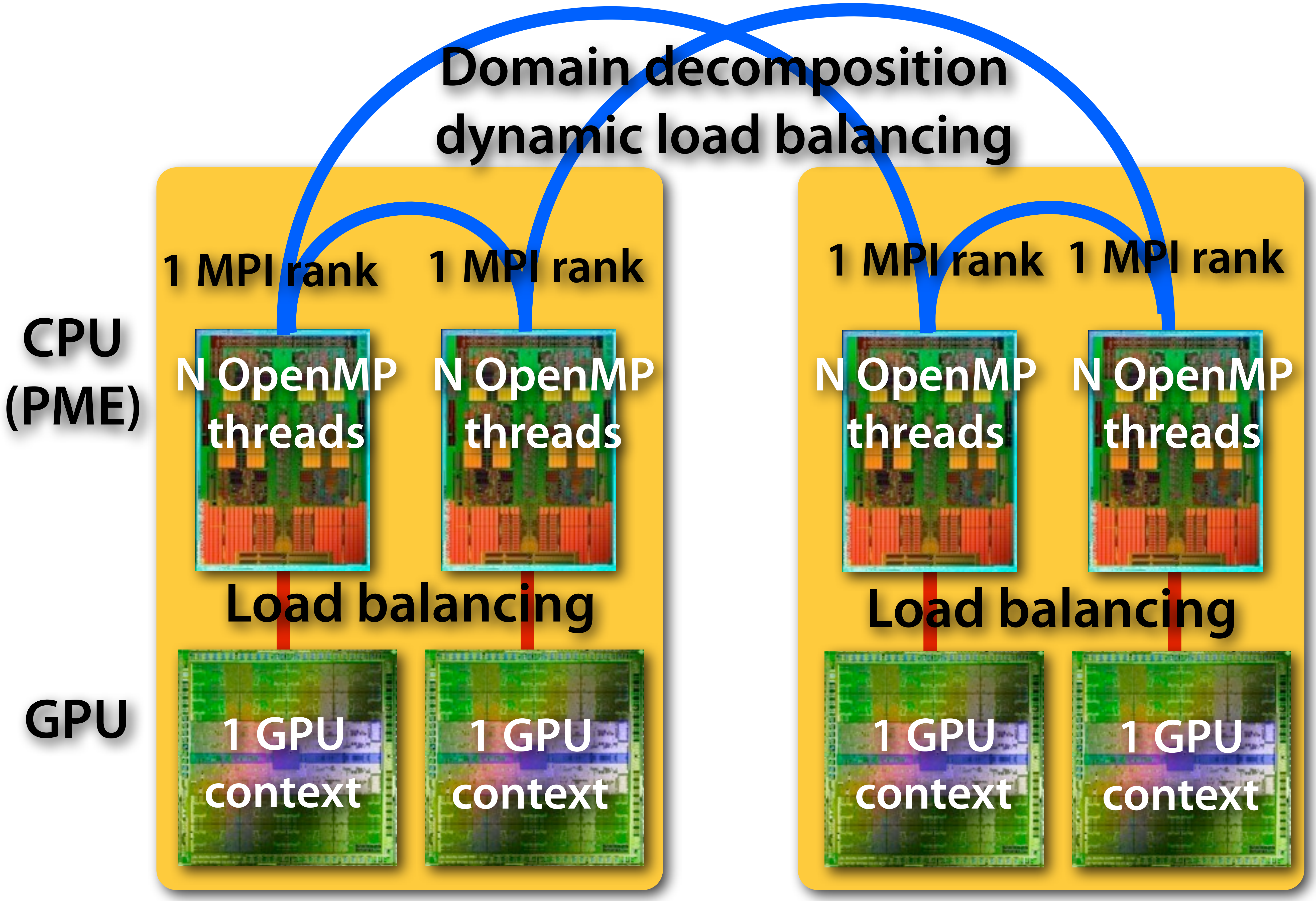


~2 TFLOP

Random memory access won't work

Great for throughput

Gromacs-4.6 next-generation GPU implementation:



Domain decomposition
dynamic load balancing

CPU
(PME)

GPU

1 MPI rank

1 MPI rank

1 MPI rank

1 MPI rank

N OpenMP threads

N OpenMP threads

N OpenMP threads

N OpenMP threads

Load balancing

Load balancing

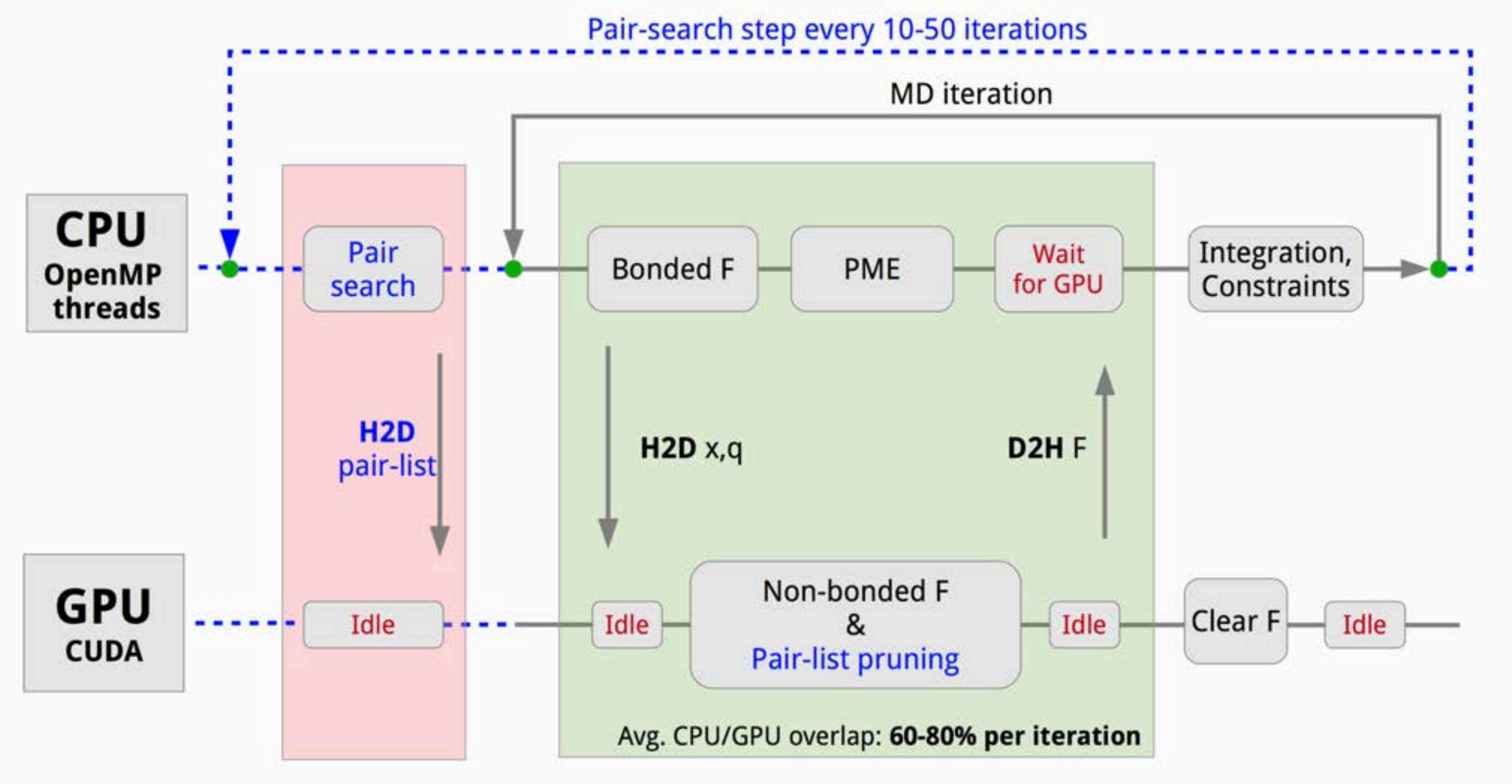
1 GPU context

1 GPU context

1 GPU context

1 GPU context

Heterogeneous CPU-GPU acceleration in GROMACS-4.6



Wallclock time for an MD step:

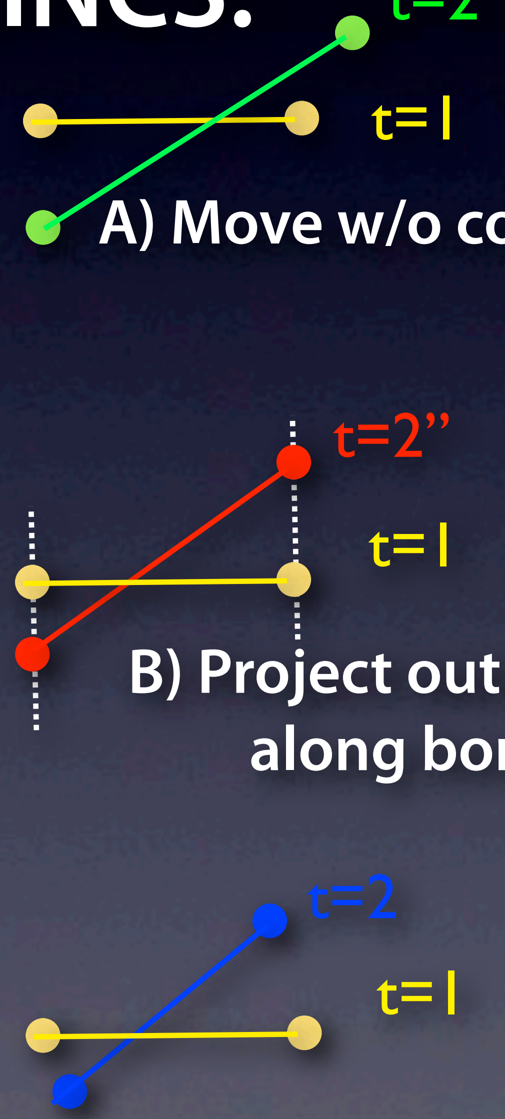
~0.5 ms if we want to simulate 1µs/day

We cannot afford to lose all previous acceleration tricks!

CPU trick 1: all-bond constraints

- Δt limited by fast motions - 1fs
  - Remove bond vibrations
- SHAKE (iterative, slow) - 2fs
  - Problematic in parallel (won’t work)
  - Compromise: constrain h-bonds only - 1.4fs
- GROMACS (LINCS):
  - LINear Constraint Solver
  - *Approximate* matrix inversion expansion
  - Fast & stable - much better than SHAKE
  - Non-iterative
  - Enables 2-3 fs timesteps
  - Parallel: P-LINCS (from Gromacs 4.0)

LINCS:

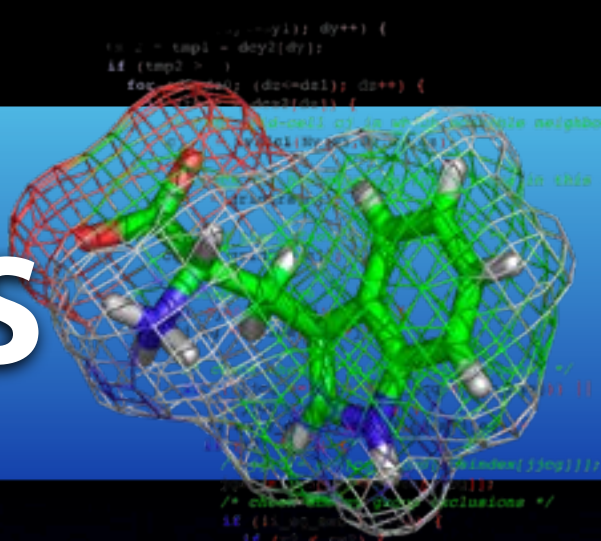


A) Move w/o constraint

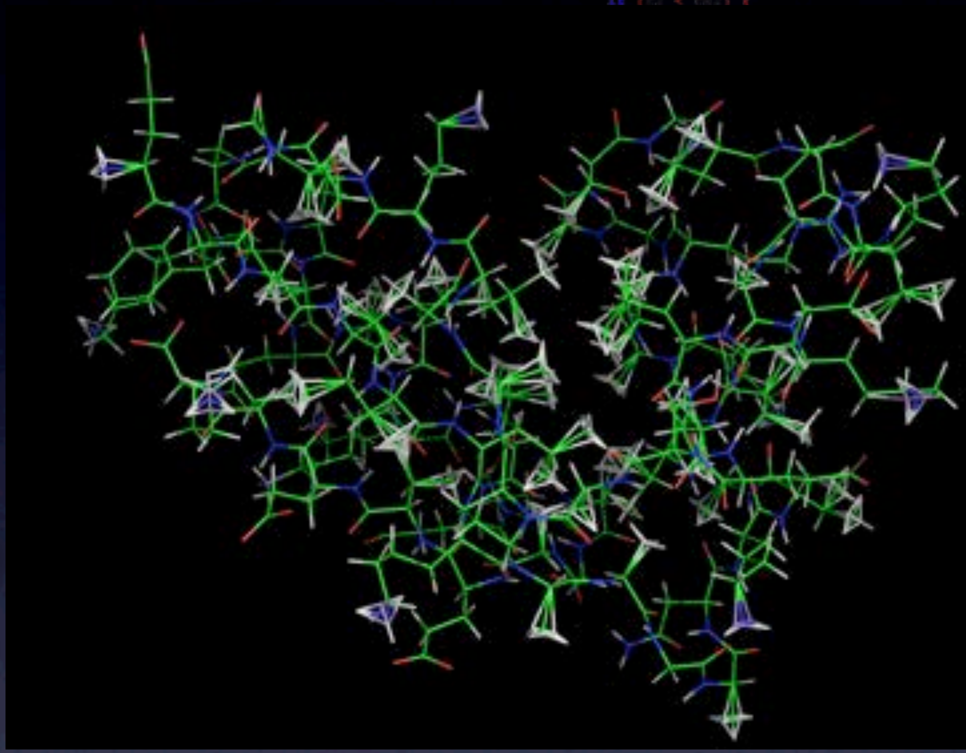
B) Project out motion along bonds

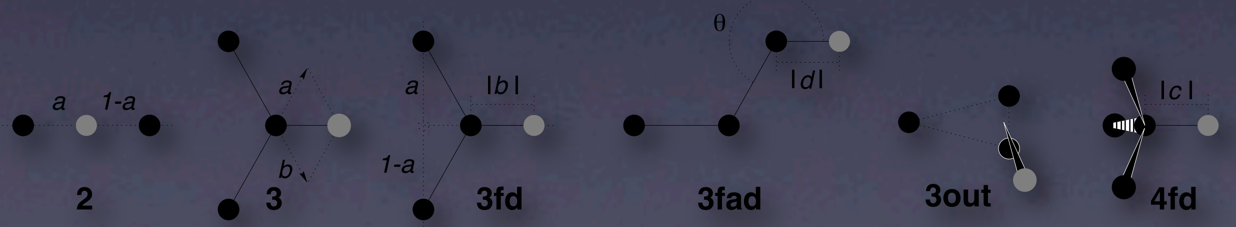
C) Correct for rotational extension of bond

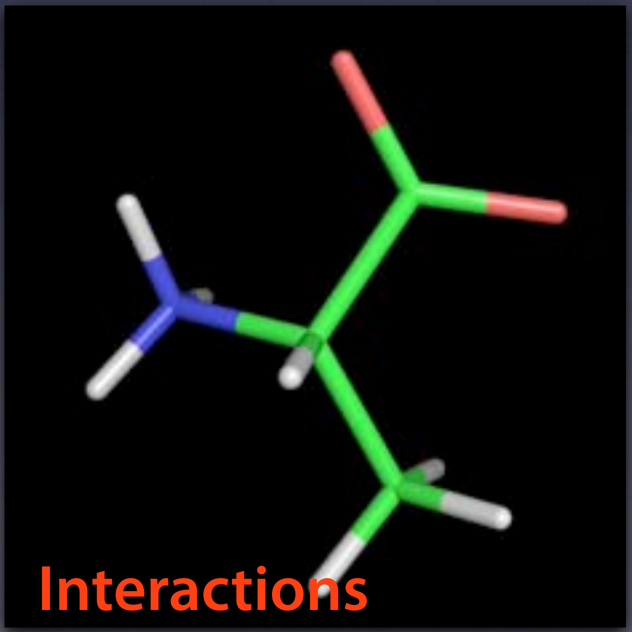
CPU trick 2: Virtual sites

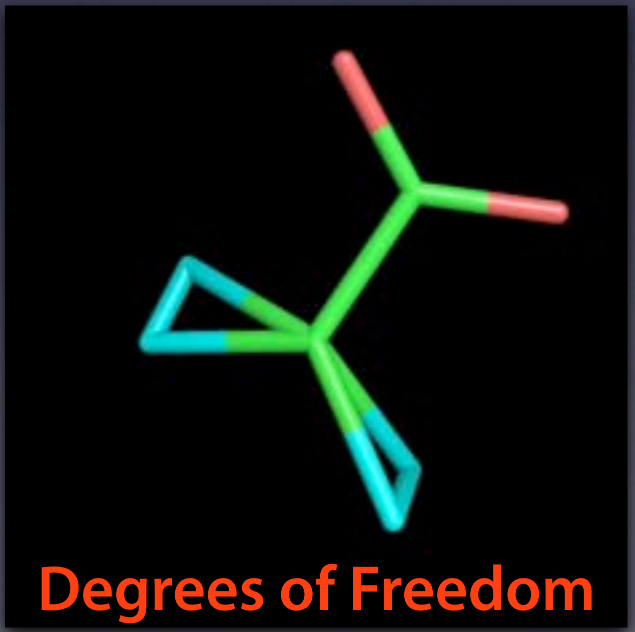


- Next fastest motions is H-angle and rotations of CH_3/NH_2 groups
- Try to remove them:
  - Ideal H position from heavy atoms.
  - CH_3/NH_2 groups are made rigid
  - Calculate forces, then project back onto heavy atoms
  - Integrate only heavy atom positions, reconstruct H's
- Enables 5fs timesteps!

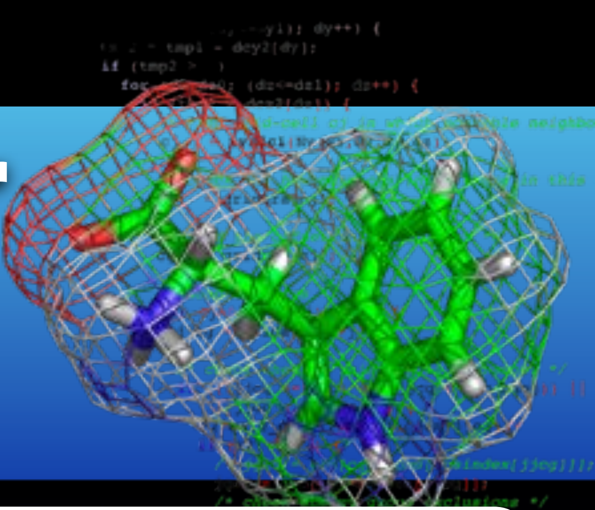


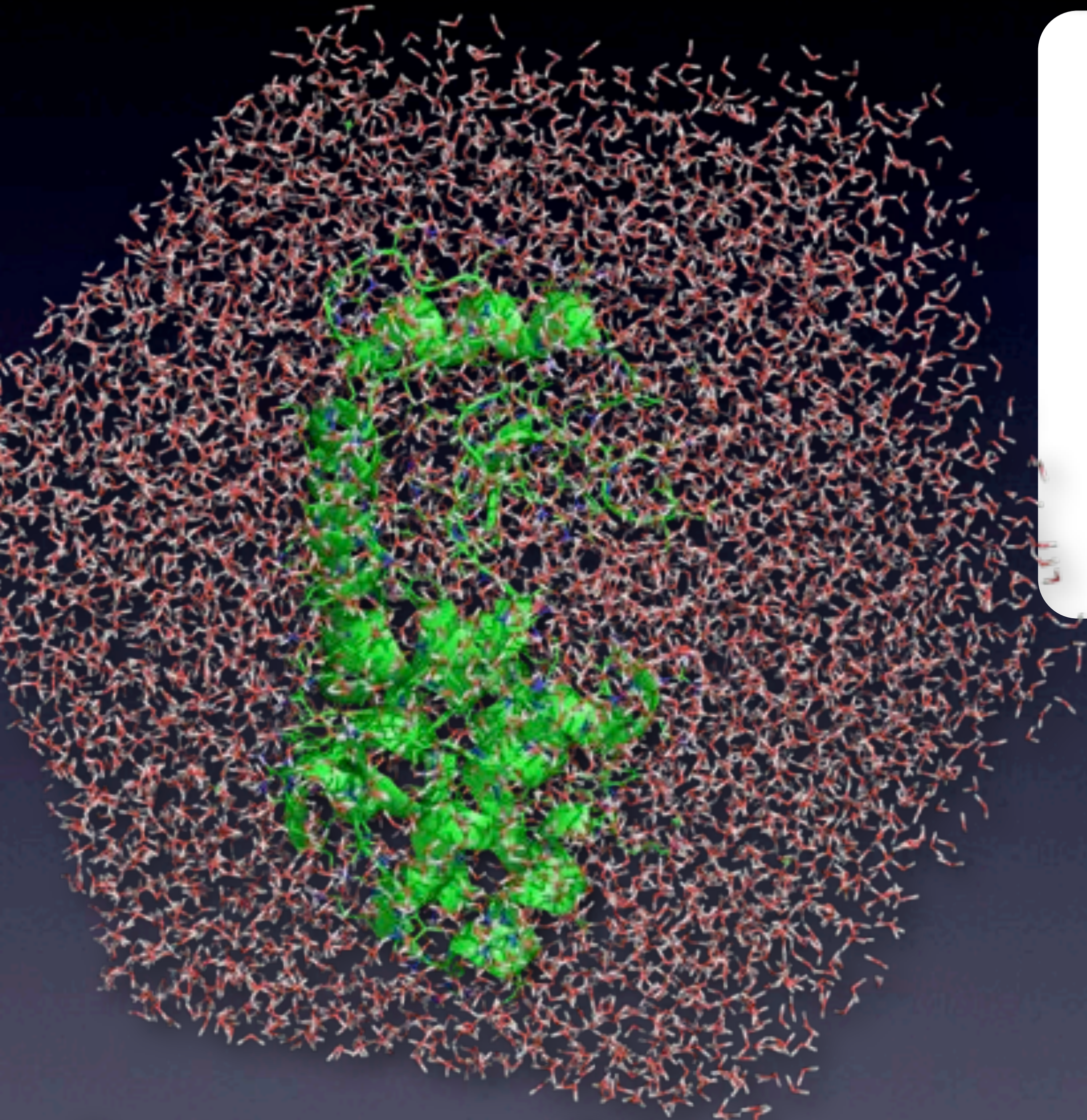


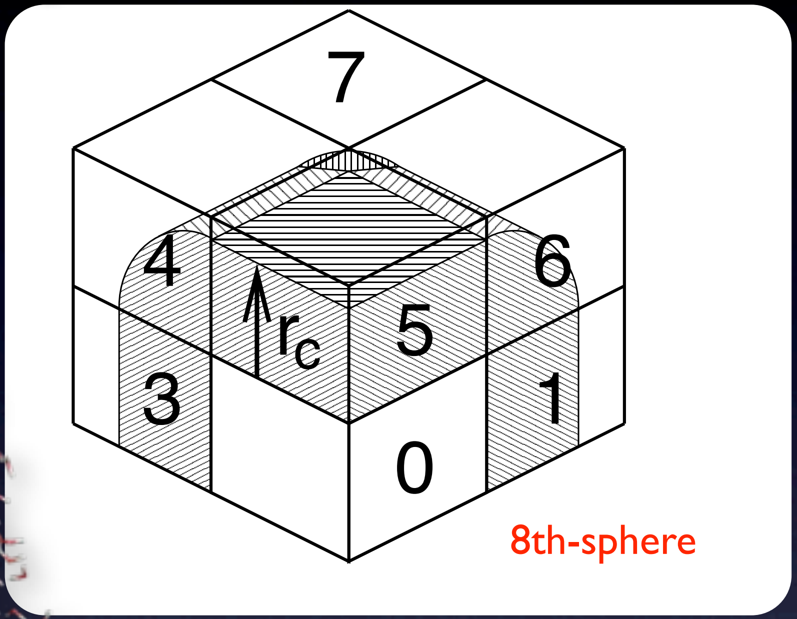


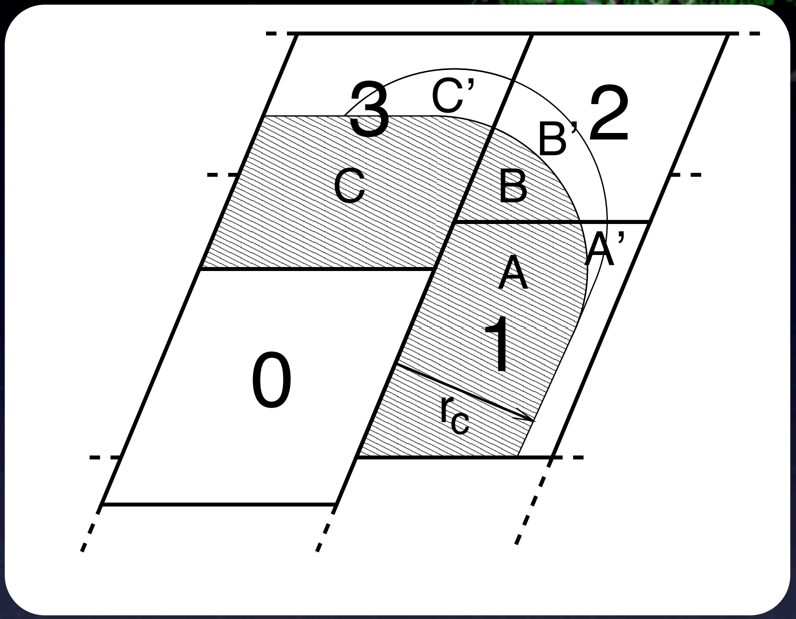


CPU trick 3: Non-rectangular cells & decomposition







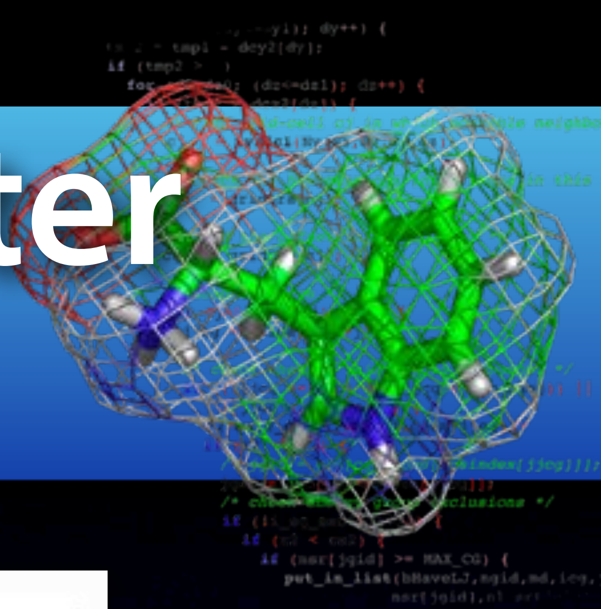


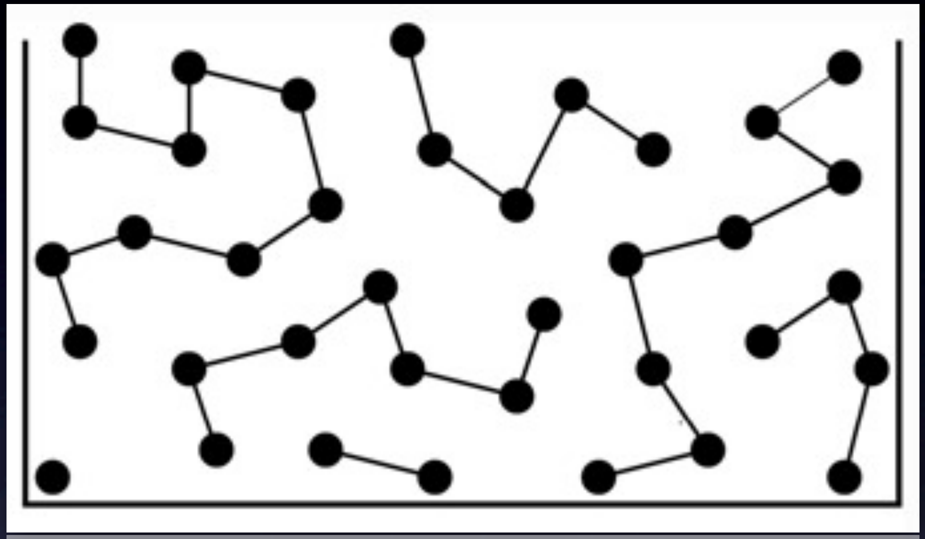
Load balancing works for arbitrary triclinic cells

Lysozyme, 25k atoms
Rhombic dodecahedron
(36k atoms in cubic cell)

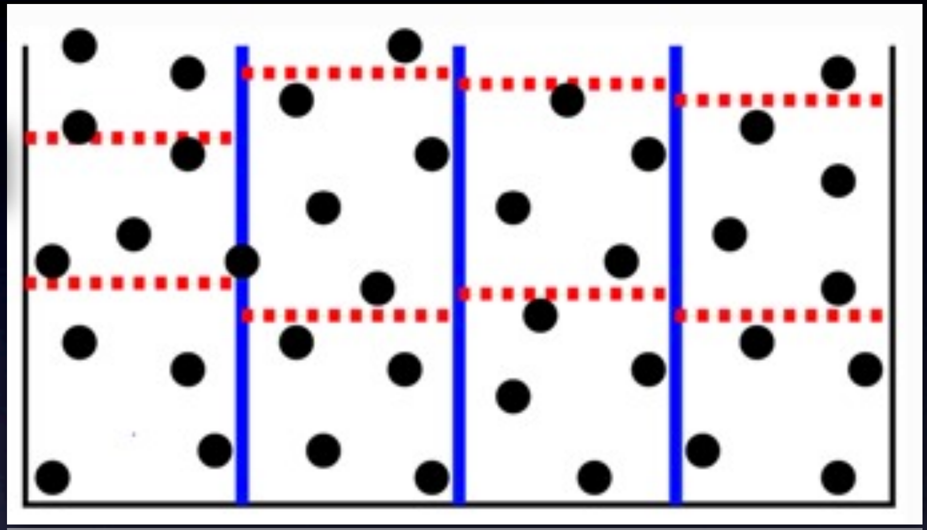
All these "tricks" now work fine with GPUs in GROMACS-4.6!

From neighborlists to cluster pair lists in GROMACS-4.6

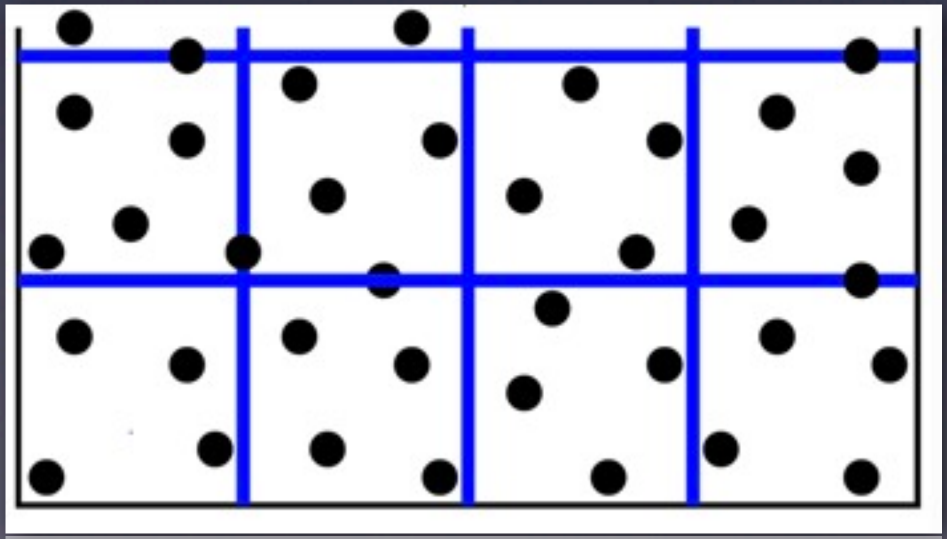




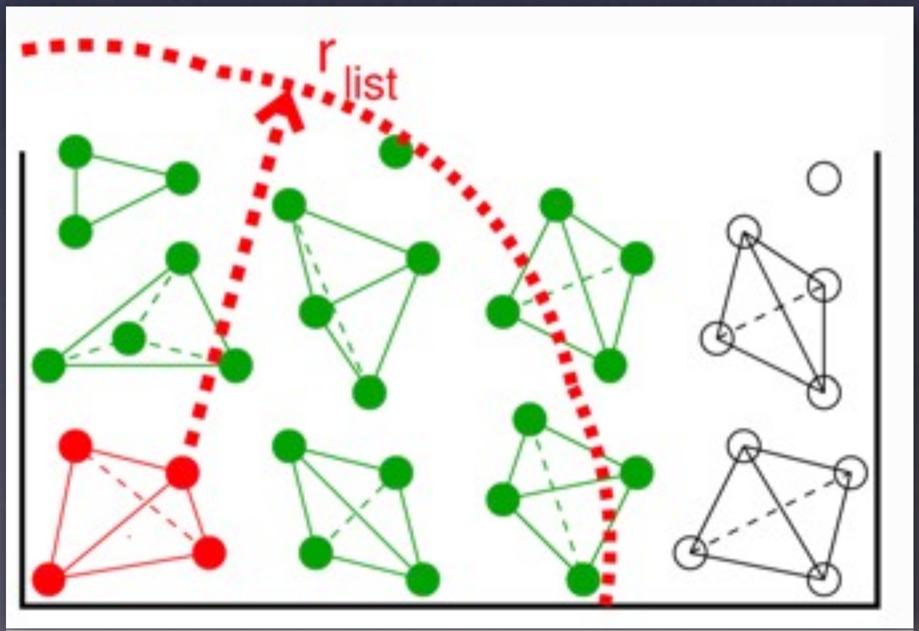
x,y grid
z sort
z bin



x,y,z
gridding



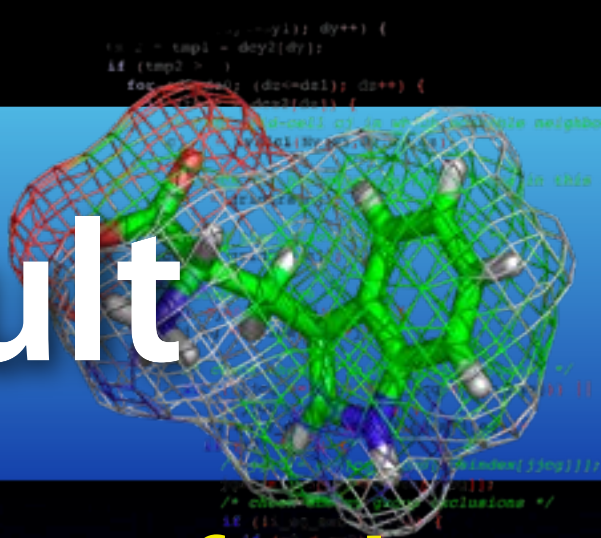
Cluster pairlist

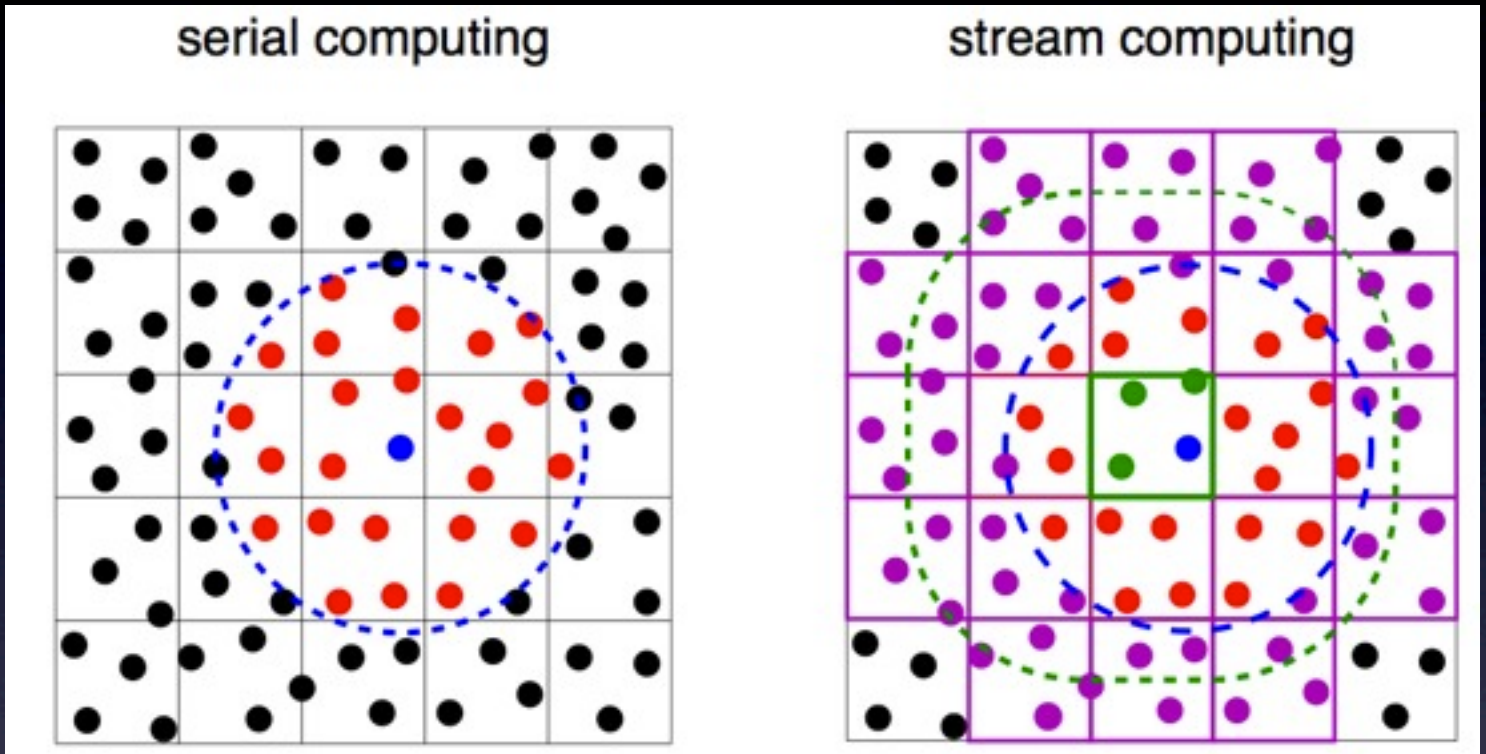


Organize
as tiles with
all-vs-all
interactions:

X	X	X	X
X	X	X	X
X	X	X	X
X	X	X	X

Tiling circles is difficult





Group cutoff Verlet cutoff

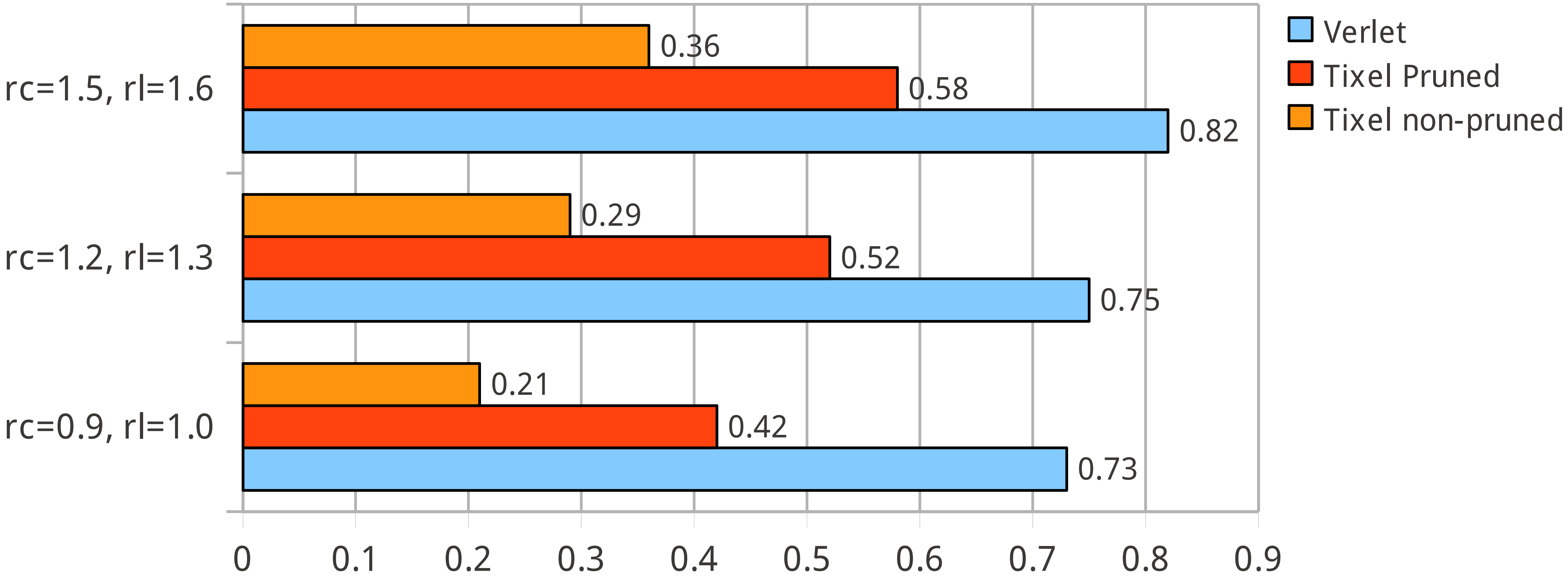
Need a lot of cubes to cover a sphere

Interactions outside cutoff should be 0.0

- GROMACS-4.6 calculates a “large enough” buffer zone so no interactions are missed
- Optimize *nstlist* for performance - no need to worry about missing any interactions with Verlet!

Tixel algorithm work-efficiency

8x8x8 tixels compared to a non performance-optimized Verlet scheme

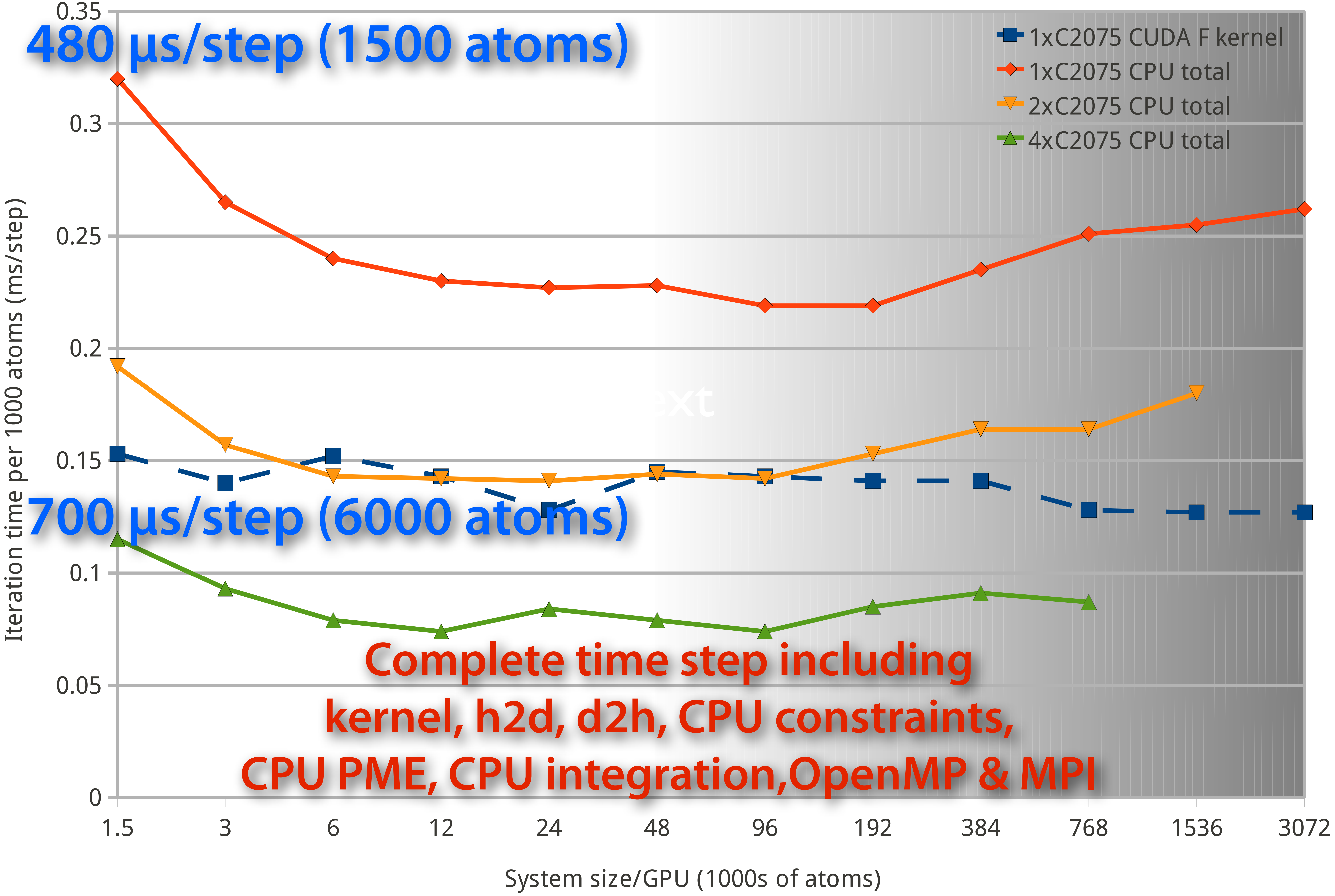


Highly memory-efficient algorithm:
Can handle 20-40 million atoms with 2-3GB memory

Even cheap consumer cards will get you a long way

PME weak scaling

Xeon X5650 3T + C2075 / process

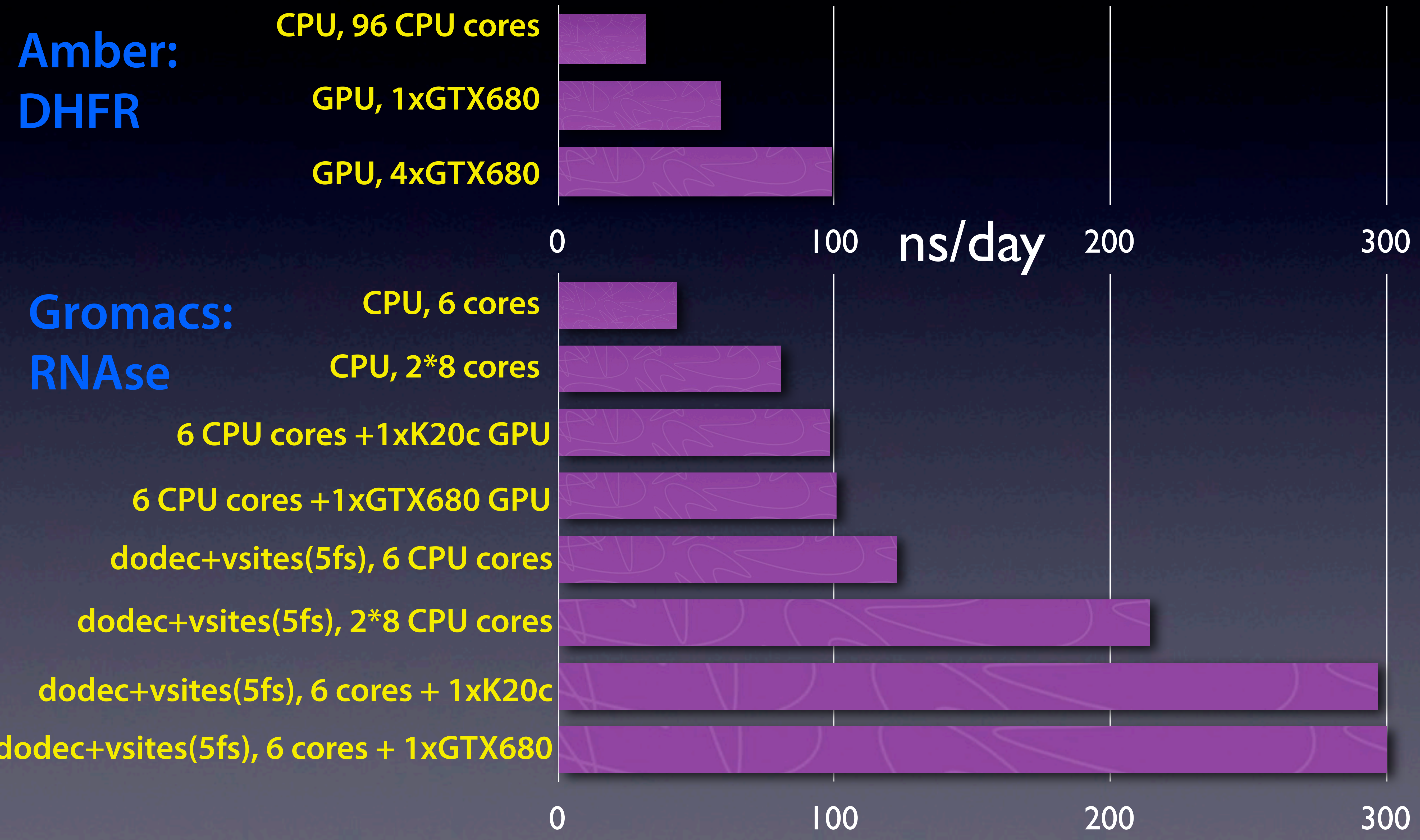


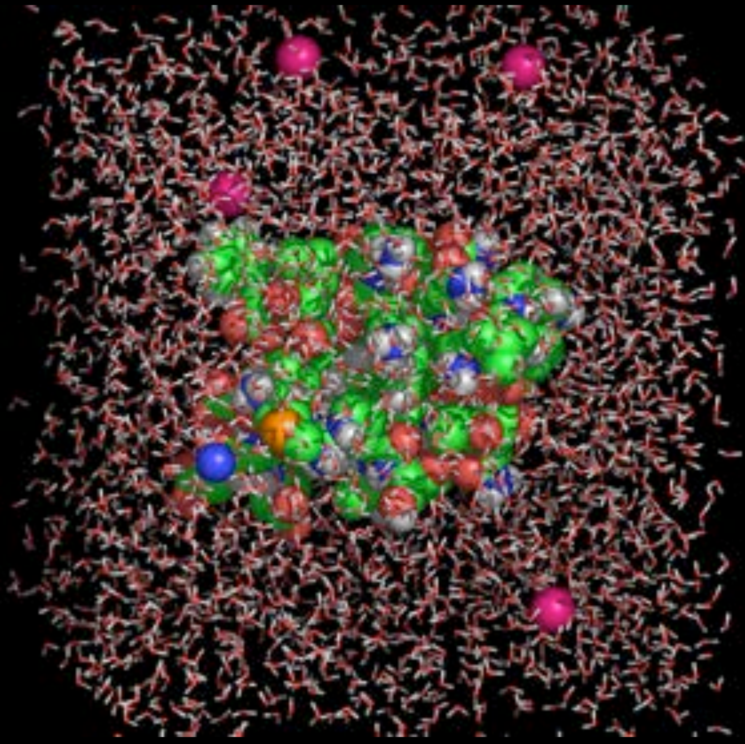
480 µs/step (1500 atoms)

700 µs/step (6000 atoms)

Complete time step including kernel, h2d, d2h, CPU constraints, CPU PME, CPU integration,OpenMP & MPI

Example performance: Systems with ~24,000 atoms, 2 fs time steps, NPT





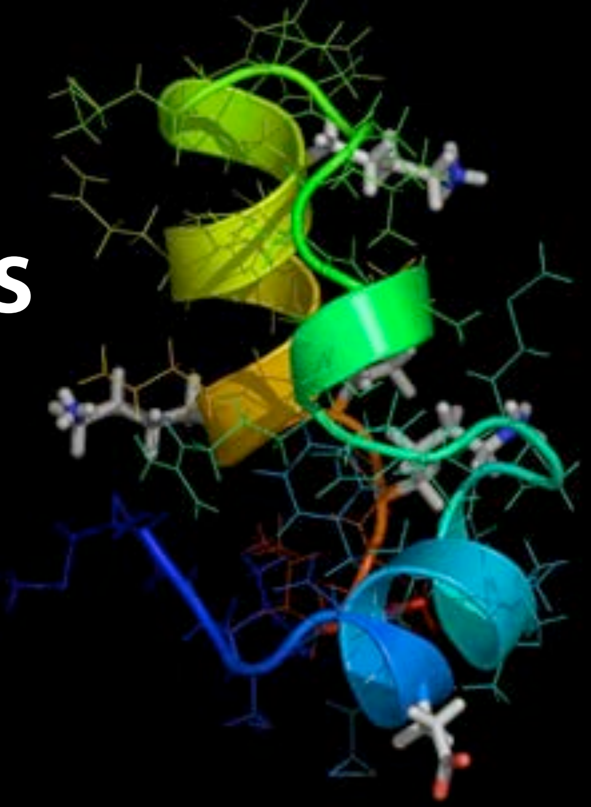
The Villin headpiece

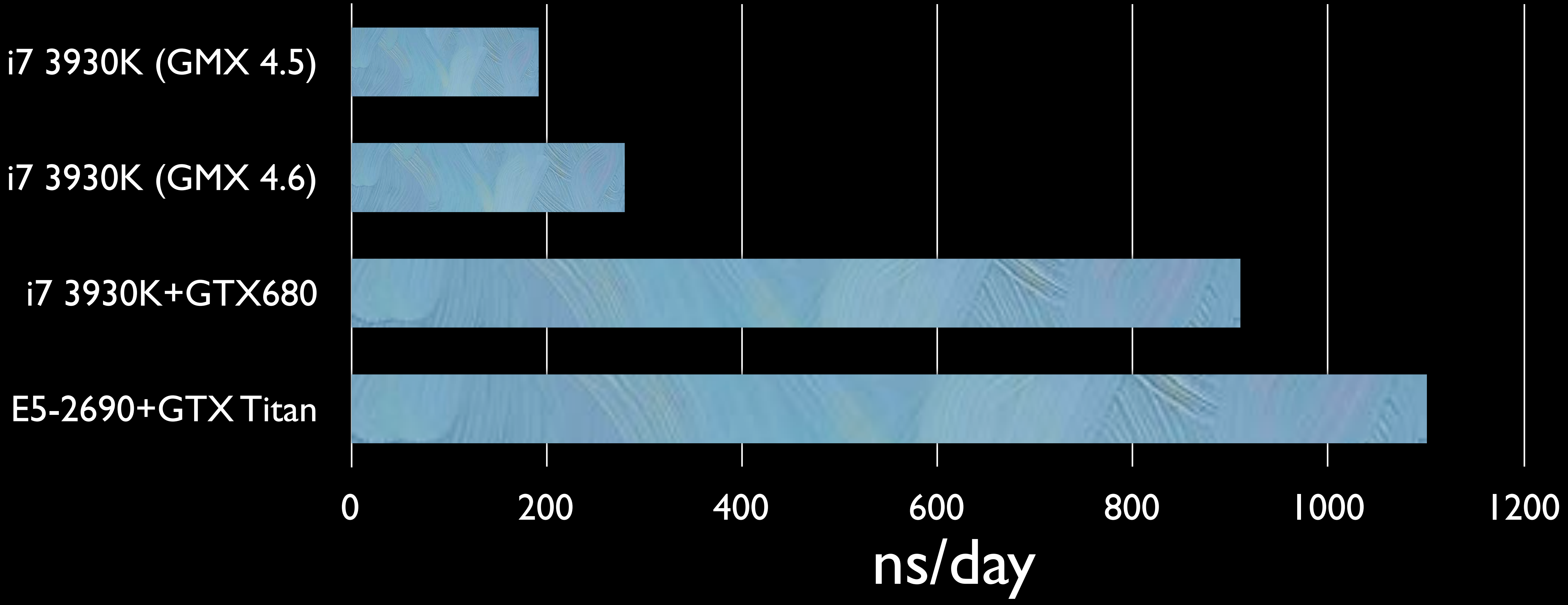
~8,000 atoms, 5 fs steps

explicit solvent

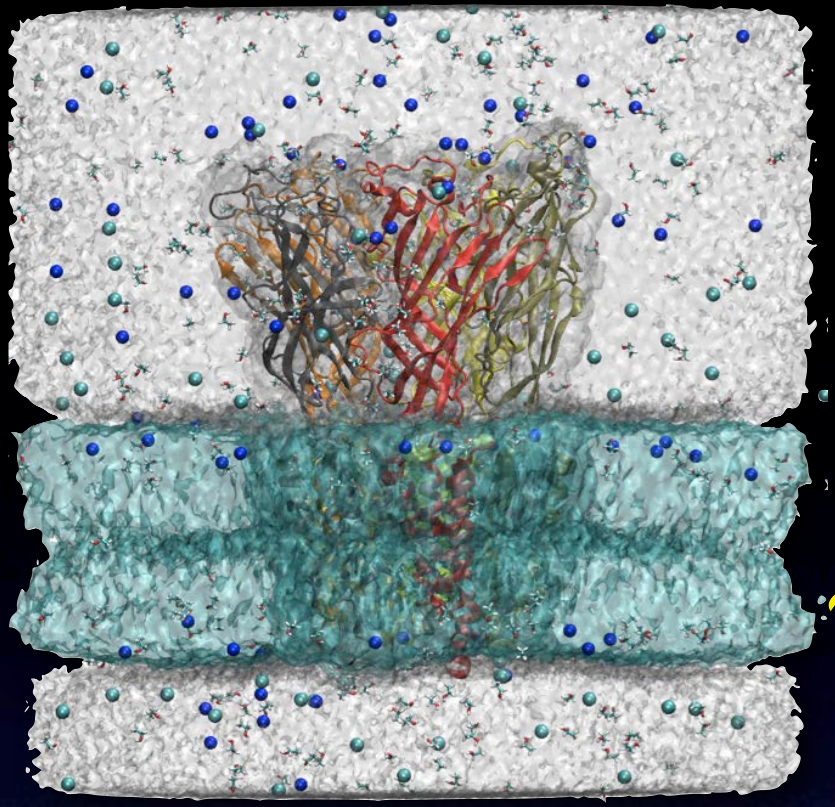
triclinic box

PME electrostatics



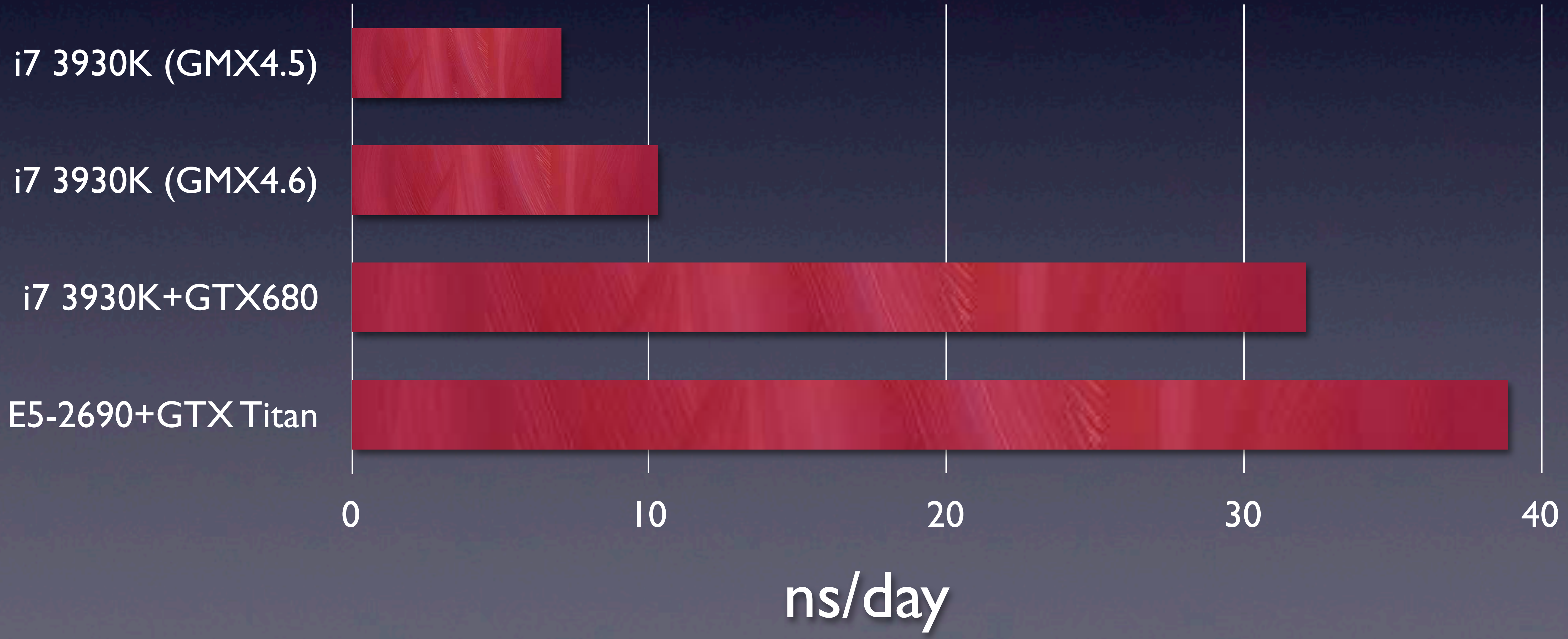


2,546 FPS (beat that, Battlefield 4)



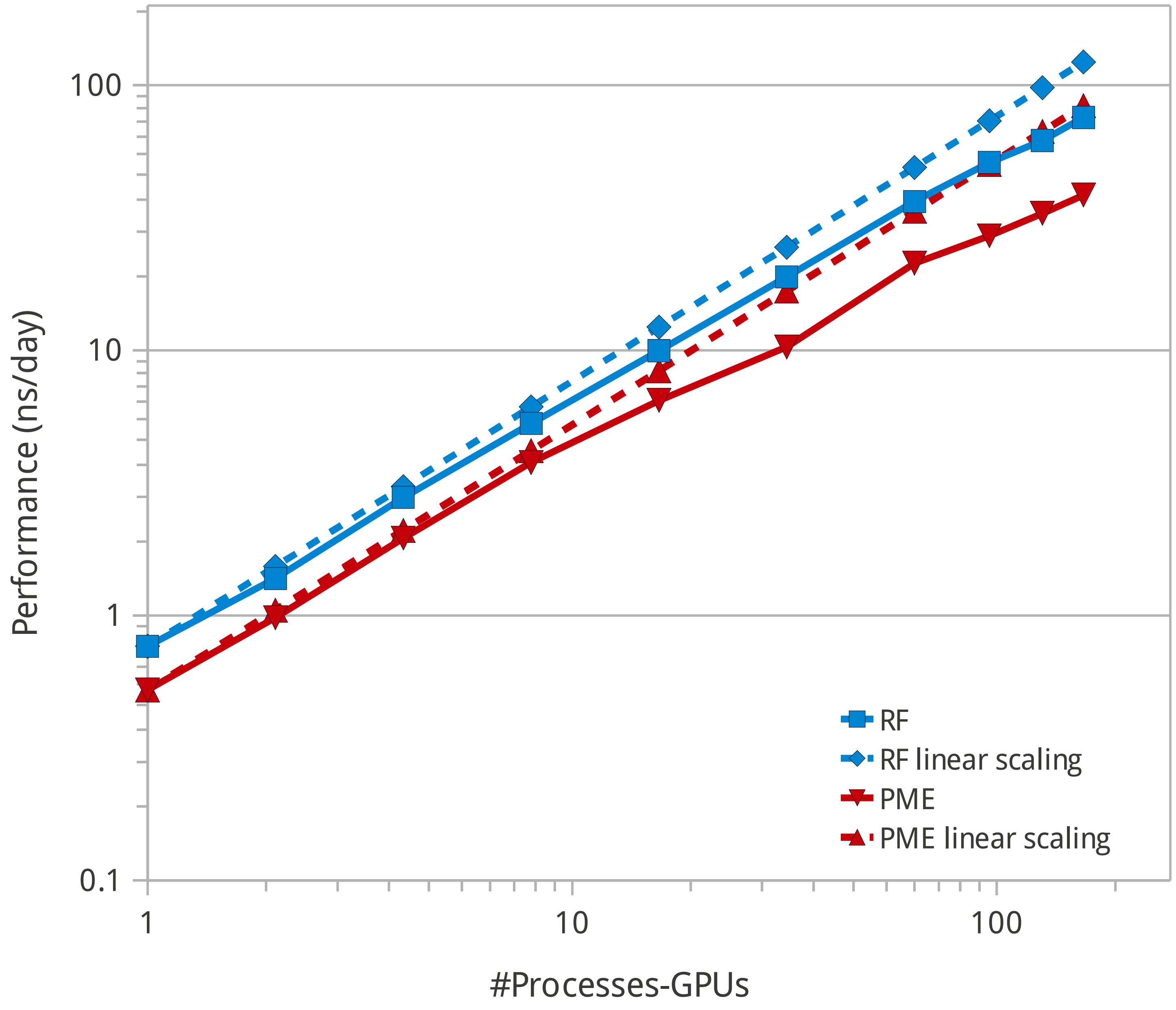
GLIC: Ion channel membrane protein 150,000 atoms

Running on a simple desktop!



Scaling of Reaction-field & PME

1.5M atoms waterbox, RF cutoff=0.9nm, PME auto-tuned cutoff

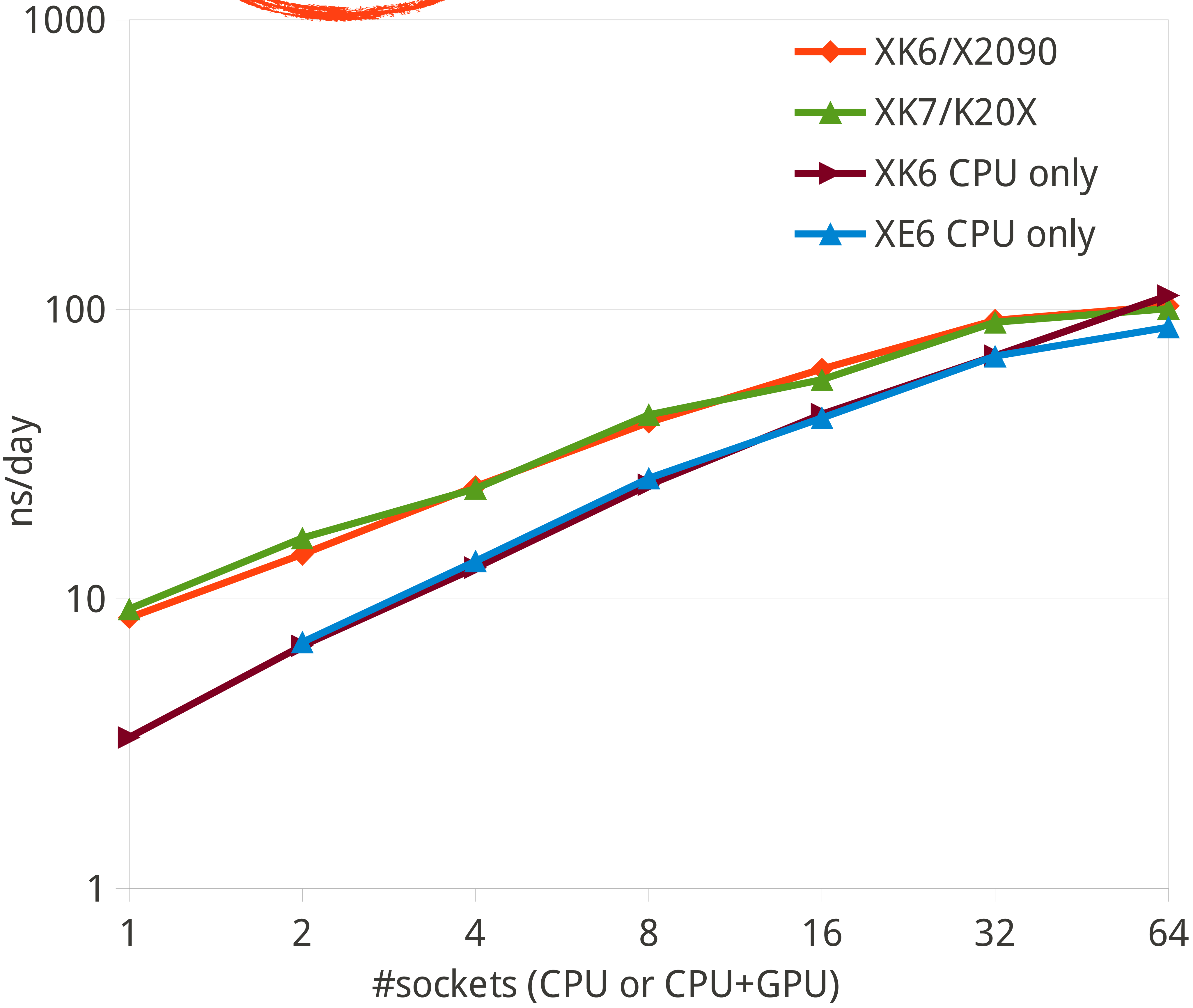


Challenge: GROMACS has very short iteration times - hard requirements on latency/bandwidth

Small systems often work best using only a single GPU!

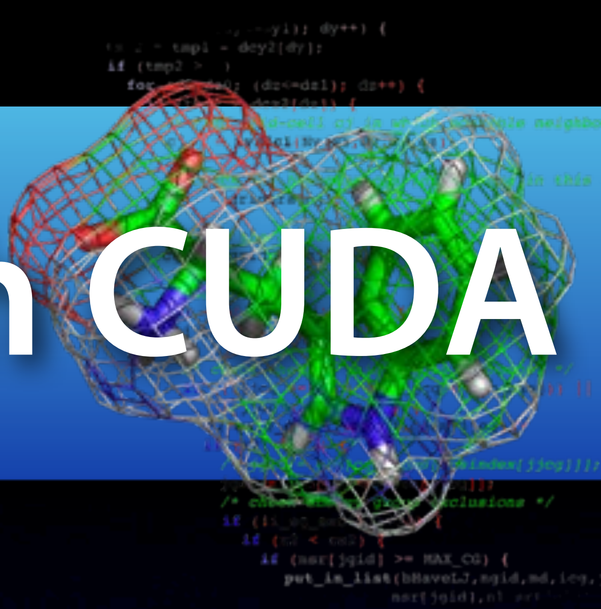
GROMACS 4.6 extreme scaling

Scaling to 130 atoms/core: ADH protein 134k atoms, PME, rc >= 0.9



Using GROMACS with GPUs in practice

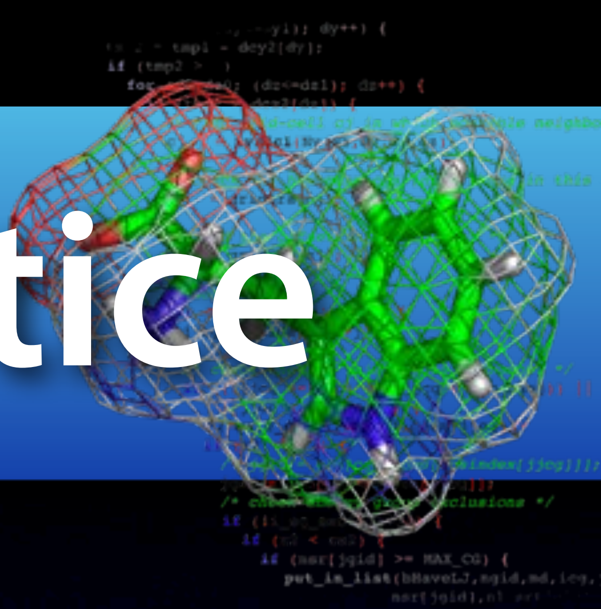
Compiling GROMACS with CUDA



- Make sure CUDA driver is installed
- Make sure CUDA SDK is in /usr/local/cuda
- Use the default GROMACS distribution
- Just run ‘cmake’ and we will detect CUDA automatically and use it
- gcc-4.7 works great as a compiler
- On Macs, you want to use icc (commercial)

Longer Mac story: Clang does not support OpenMP, which gcc does. However, the current gcc versions for Macs do not support AVX on the CPU. icc supports both!

Using GPUs in practice



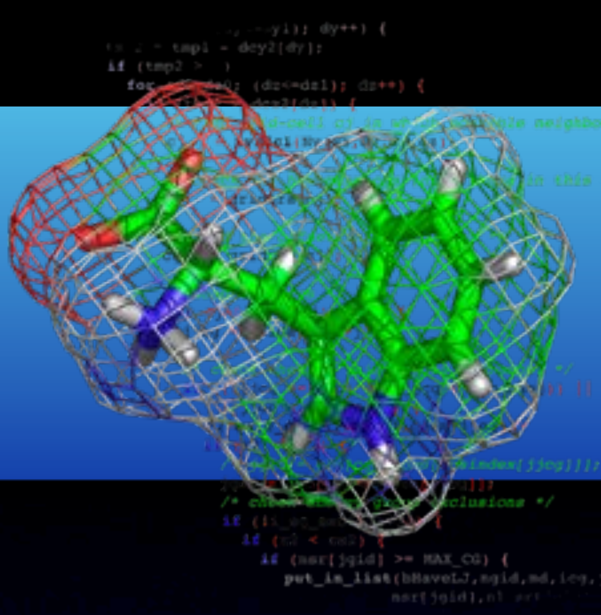
In your mdp file:

```
cutoff-scheme = Verlet
nstlist       = 10       ; likely 10-50
coulombtype   = pme      ; or reaction-field
vdw-type      = cut-off
nstcalcenergy = -1       ; only when writing edr
```

- Verlet cutoff-scheme is more accurate
- Necessary for GPUs in GROMACS
- Use -*testverlet* mdrun option to force it w. old tpr files
- Slower on a single CPU, but scales well on CPUs too!

Shift modifier is applied to both coulomb and VdW by default on GPUs - change with coulomb/vdw-modifier

Load balancing



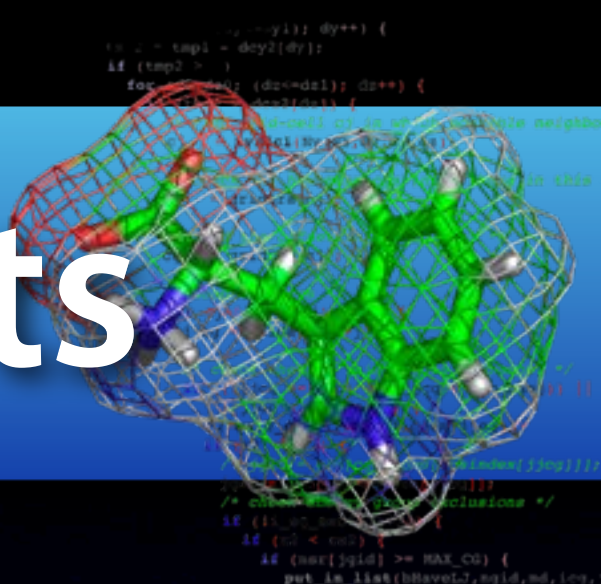
```
rcoulomb          = 1.0
fourierspacing    = 0.12
```

- If we increase/decrease the coulomb direct-space cutoff and the reciprocal space PME grid spacing by the same amount, we maintain accuracy
- ... but we move work between CPU & GPU!
- By default, GROMACS-4.6 does this automatically at the start of each run - you will see diagnostic output

GROMACS excels when you combine a fairly fast CPU and GPU. Currently, this means Intel CPUs.

Demo

Acknowledgments



- GROMACS: Berk Hess, David v. der Spoel, Per Larsson, Mark Abraham
- Gromacs-GPU: Szilard Pall, Berk Hess, Rossen Apostolov
- Multi-Threaded PME: Roland Shultz, Berk Hess
- Nvidia: Mark Berger, Scott LeGrand, Duncan Poole, and others!

erc European Research Council

SJUNDE RAMPROGRAMMET

Vetenskapsrådet

STIFTELSEN för STRATEGISK FORSKNING

NVIDIA

Test Drive K20 GPUs!

Experience The Acceleration

- Run GROMACS on Tesla K20 GPU today
- Sign up for FREE GPU Test Drive on remotely hosted clusters www.nvidia.com/GPUTestDrive

Questions?

Contact us

- Devang Sachdev - NVIDIA
  - dsachdev@nvidia.com
  - @DevangSachdev
- GROMACS questions
  - **Check** www.gromacs.org
  - gmx-users@gromacs.org **mailing list**

Stream other webinars from GTC Express: http://www.gputechconf.com/page/gtc-express-webinar.html

Register for the Next GTC Express Webinar

Molecular Shape Searching on GPUs

Paul Hawkins, Applications Science Group Leader, OpenEye

Wednesday, May 22, 2013, 9:00 AM PDT

Register at *www.gputechconf.com/gtcexpress*