Washington 20120330

Current & Future Exascale MD Challenges from the GROMACS perspective

Erik Lindahl

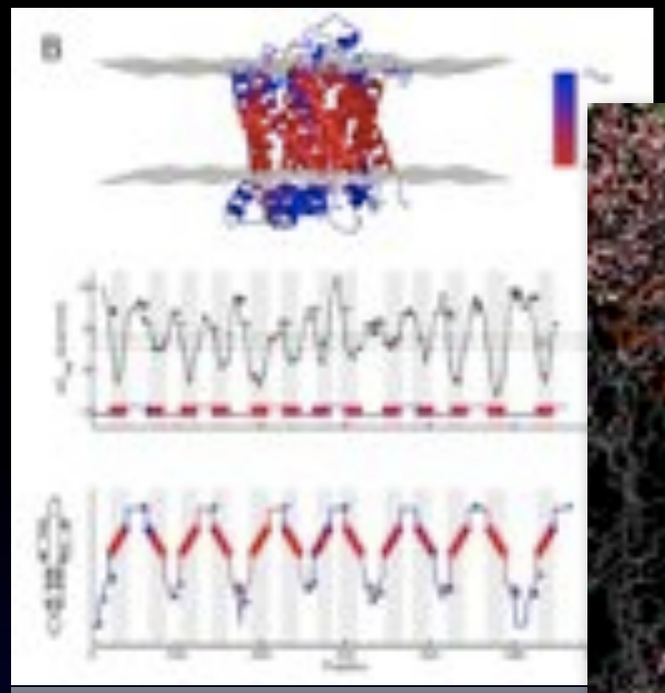
erik@kth.se

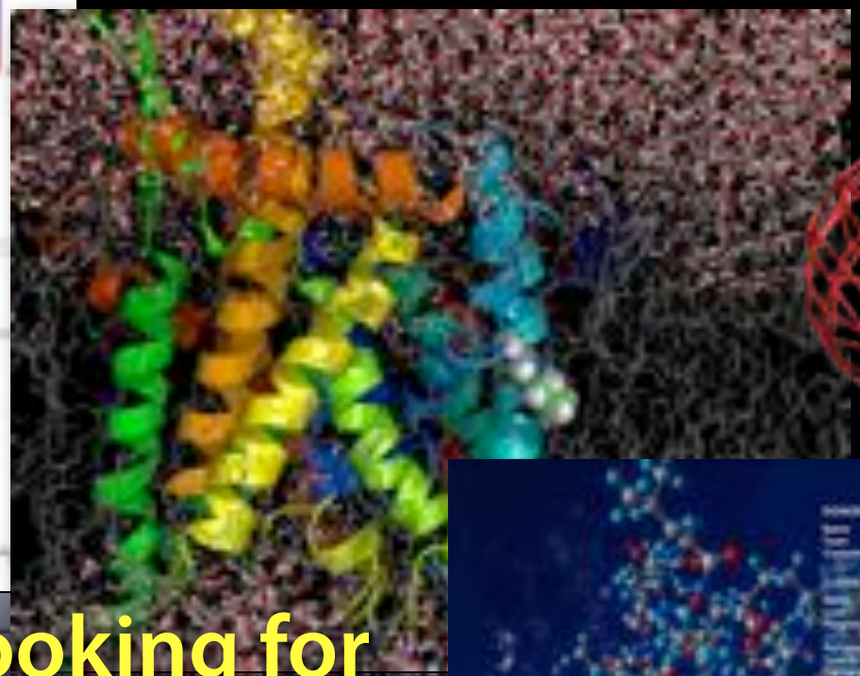
Royal Institute of Technology

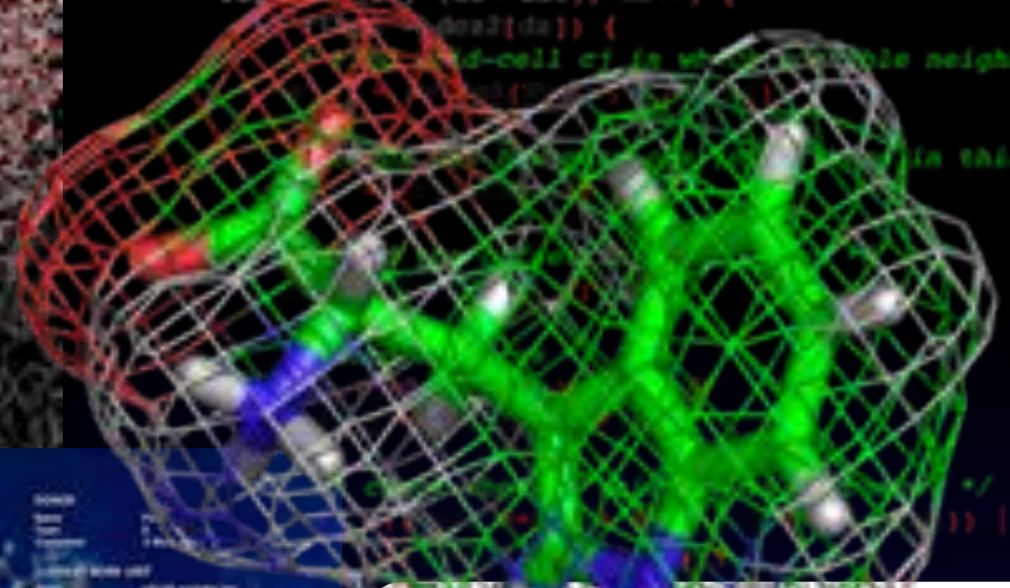
Center for Biomembrane Research

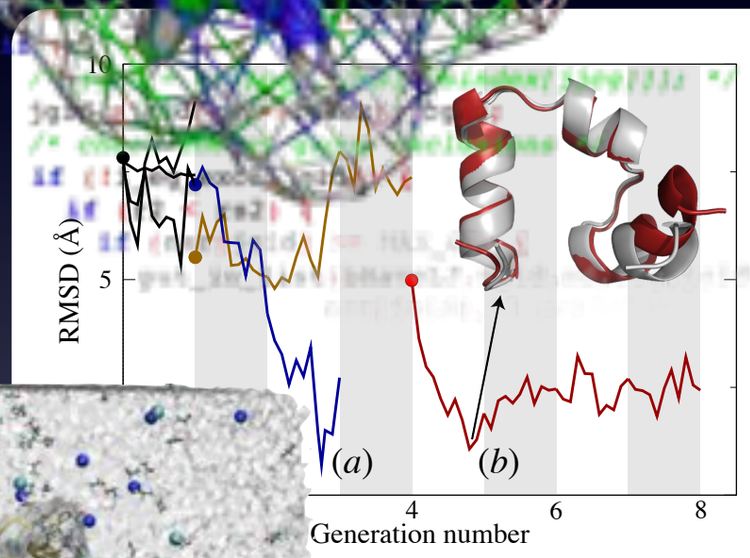
We're always looking for talented postdocs interested in theory, simulation, and applications!

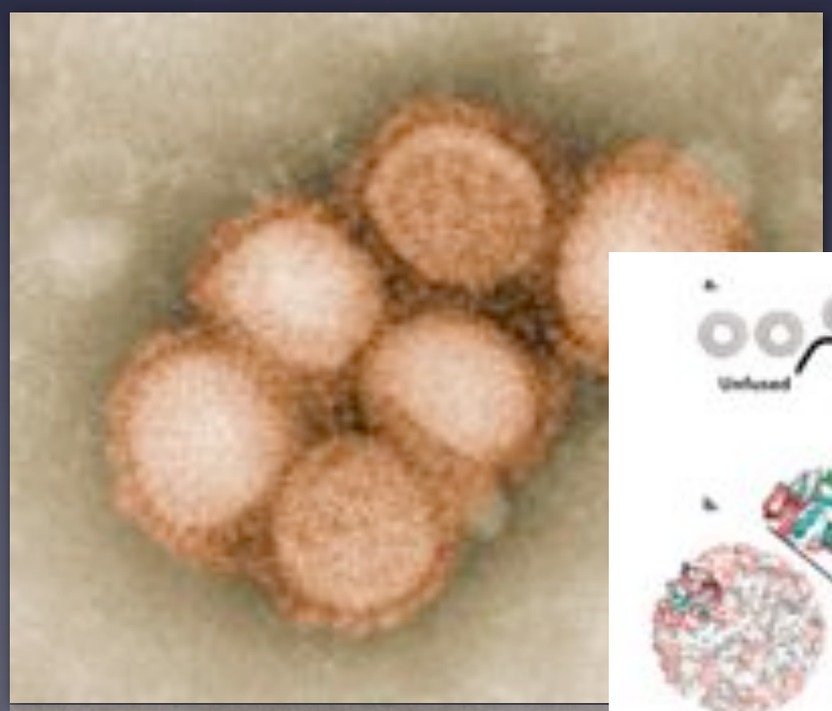
erik@kth.se

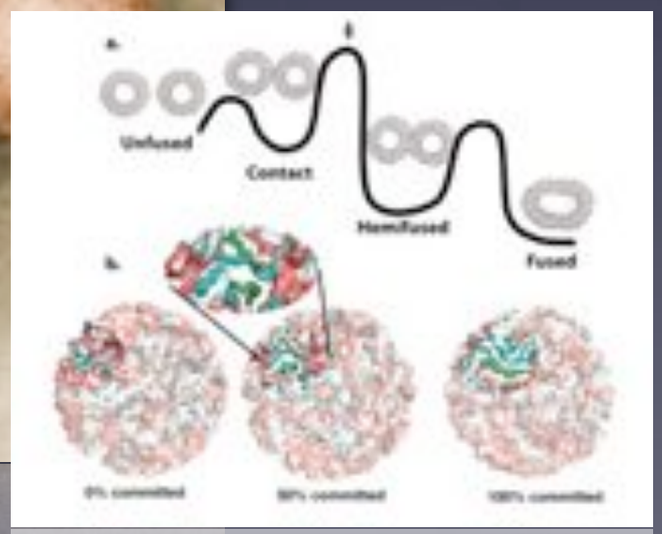


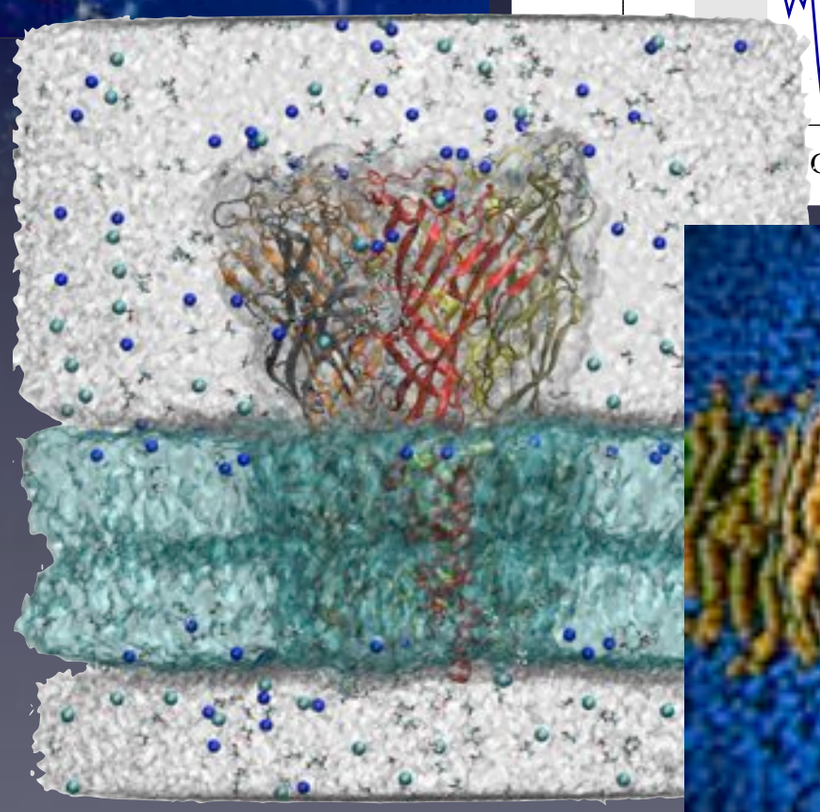


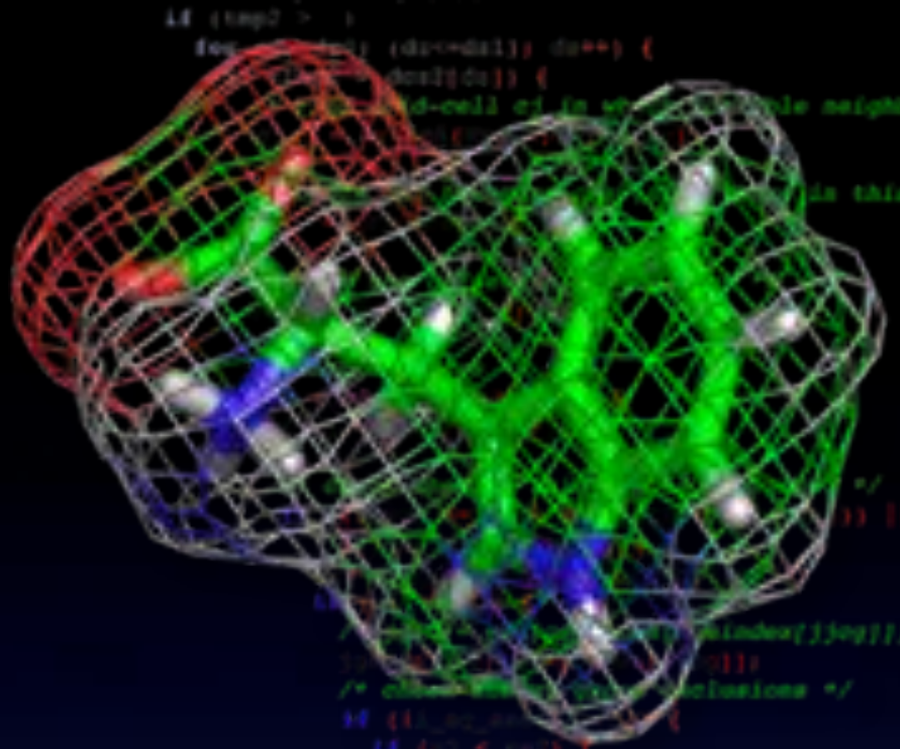












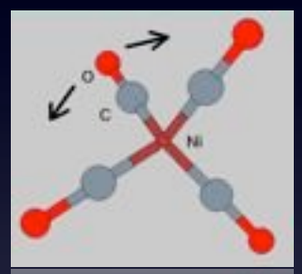
Our Interests, Trials, Errors & Occasional Successes in Computation

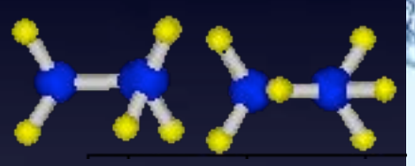
MD Simulation Challenges

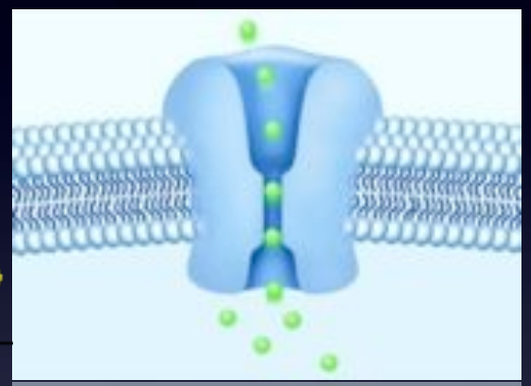
Experiments

Efficient averaging

Less detail







Physics

Chemistry

Biology

10^{-15}s 10^{-12}s 10^{-9}s 10^{-6}s 10^{-3}s 10^{0}s 10^{3}s

Where we need to be

Simulations

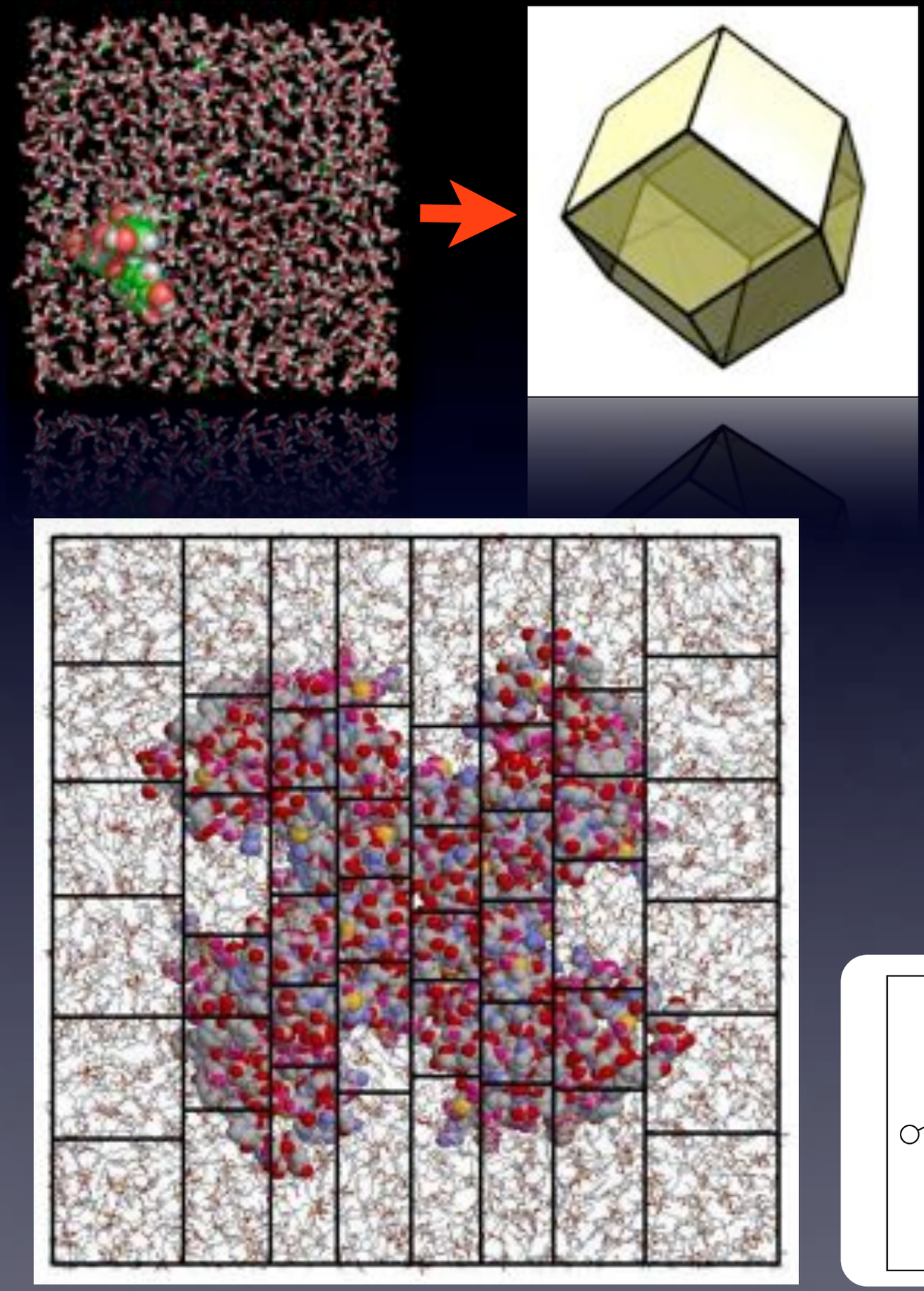
Extreme detail

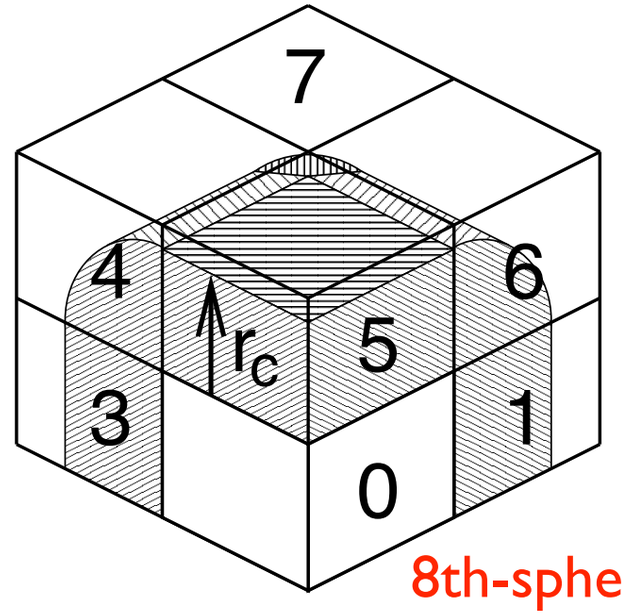
Sampling issues?

Parameter quality?

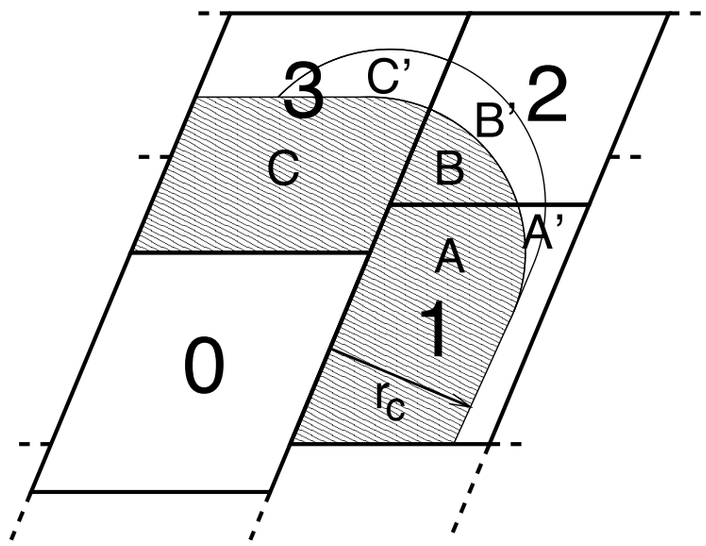
Where we are

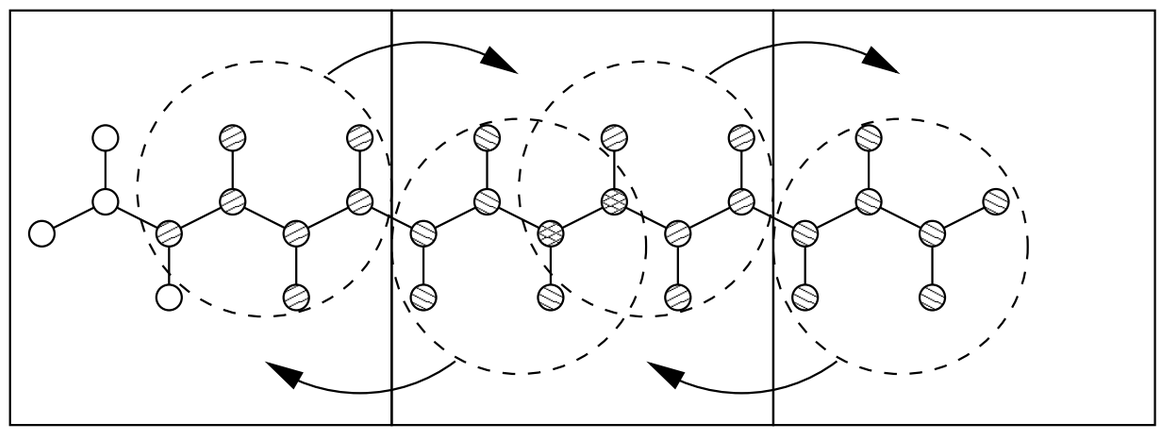
Where we want to be





8th-sphere

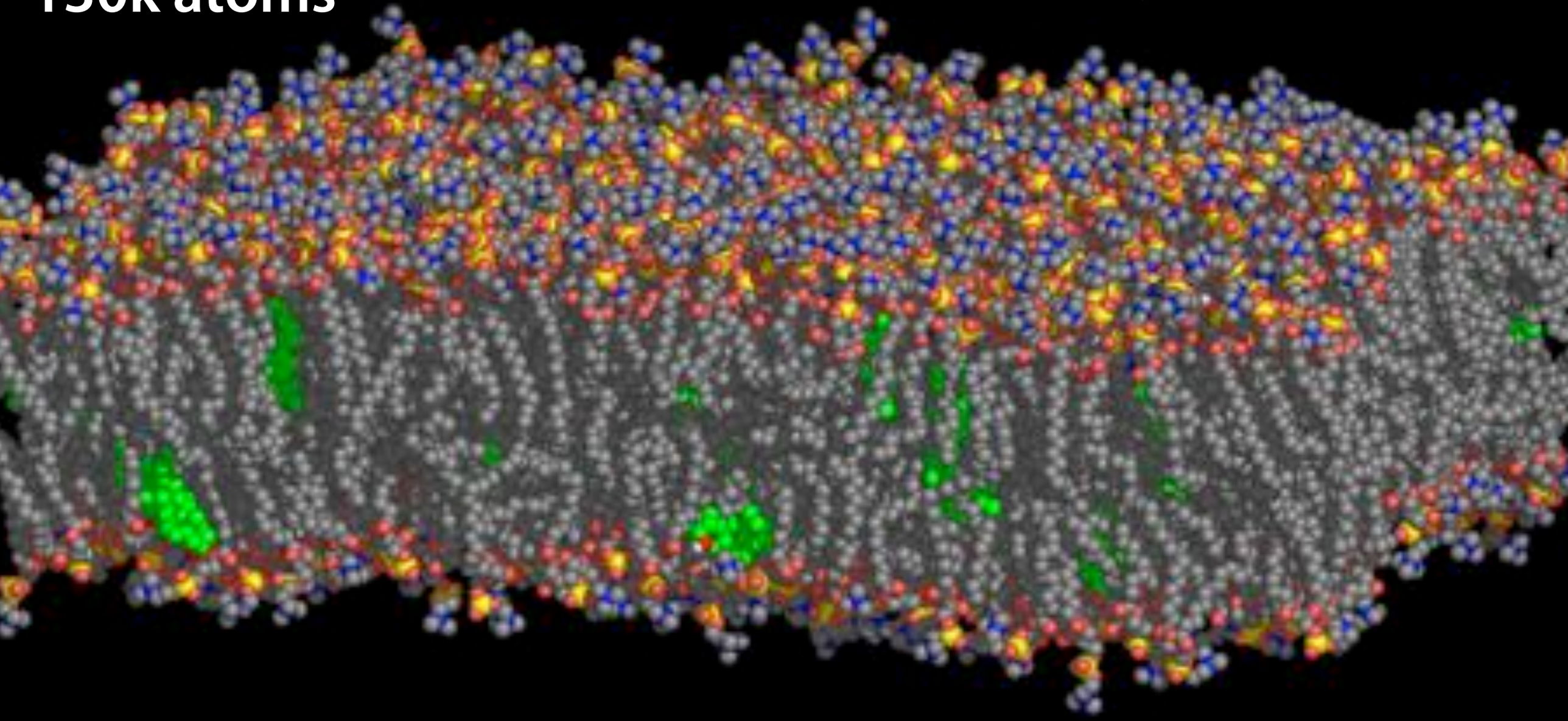




(Scaling data from 2008)

DPPC & Cholesterol
130k atoms

Blue Gene/L & Blue Matter:
scaled to 3 atoms/CPU
~10ns/day on 8192 CPUs



GROMACS 3: 2ns/day

...on a single dual
dual-core Opteron!

It is easier to get a simple problem/algorithm to scale!

i.e., you see much better *relative* scaling before introducing any optimization

Even with amazing network we hit a limit at ~200 atoms/core

100 atoms/core is certainly within reach, maybe 10, not 1

We need faster nodes, not just more nodes at lower clock

Programming model

Domain decomposition
dynamic load balancing

CPU

1 MPI rank

N OpenMP threads

1 MPI rank

N OpenMP threads

1 MPI rank

N OpenMP threads

1 MPI rank

N OpenMP threads

Load balancing

Load balancing

GPU

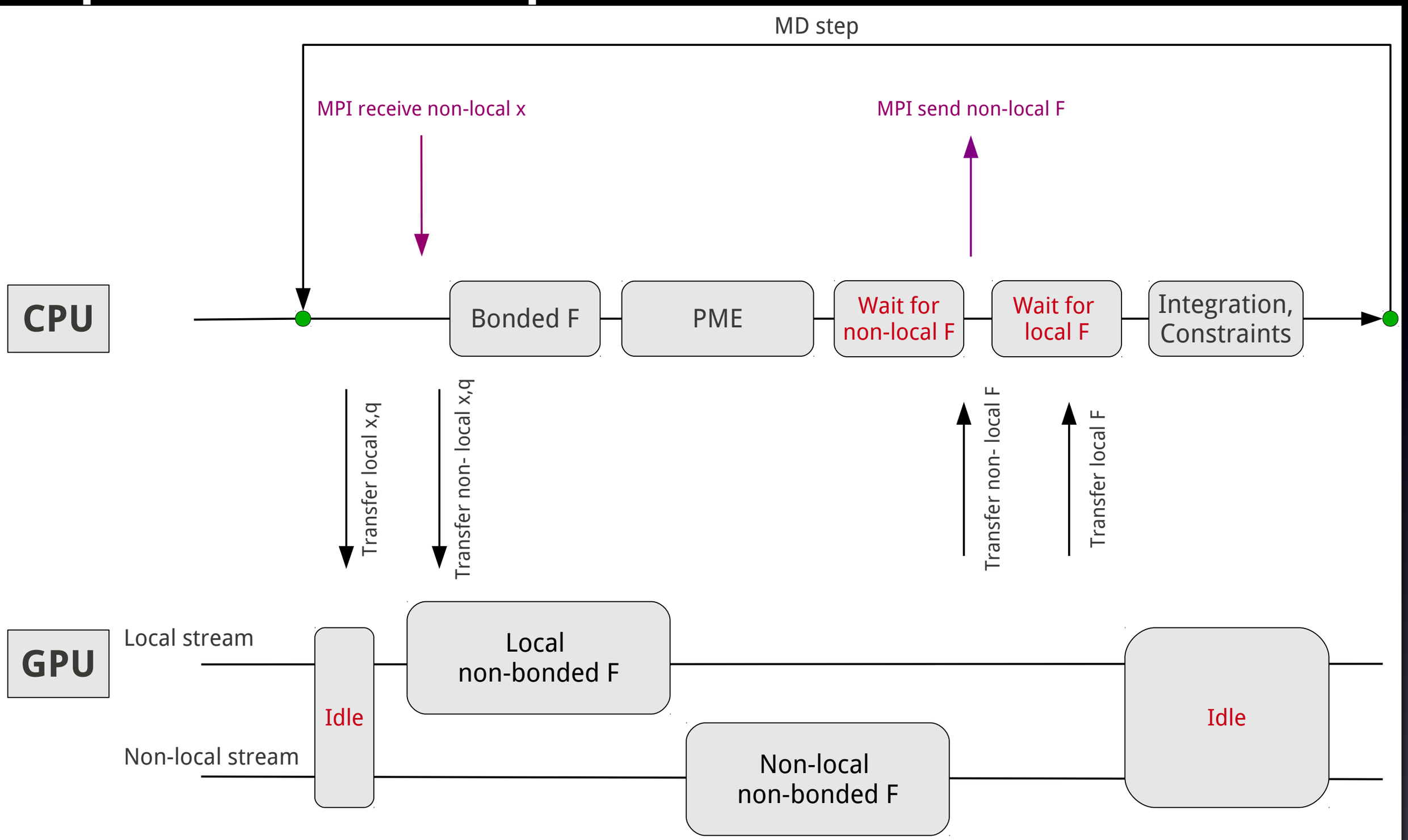
1 GPU context

1 GPU context

1 GPU context

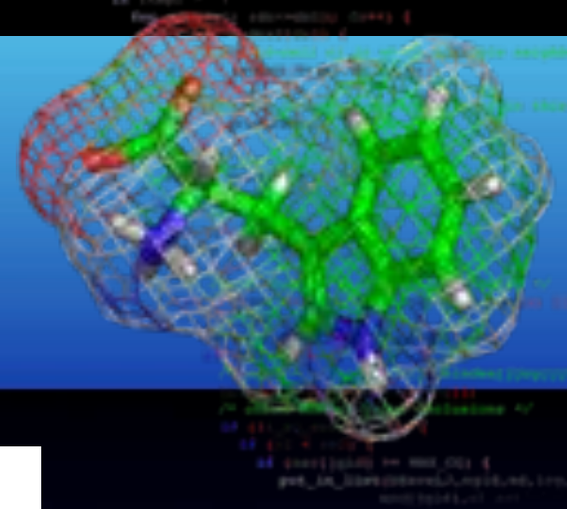
1 GPU context

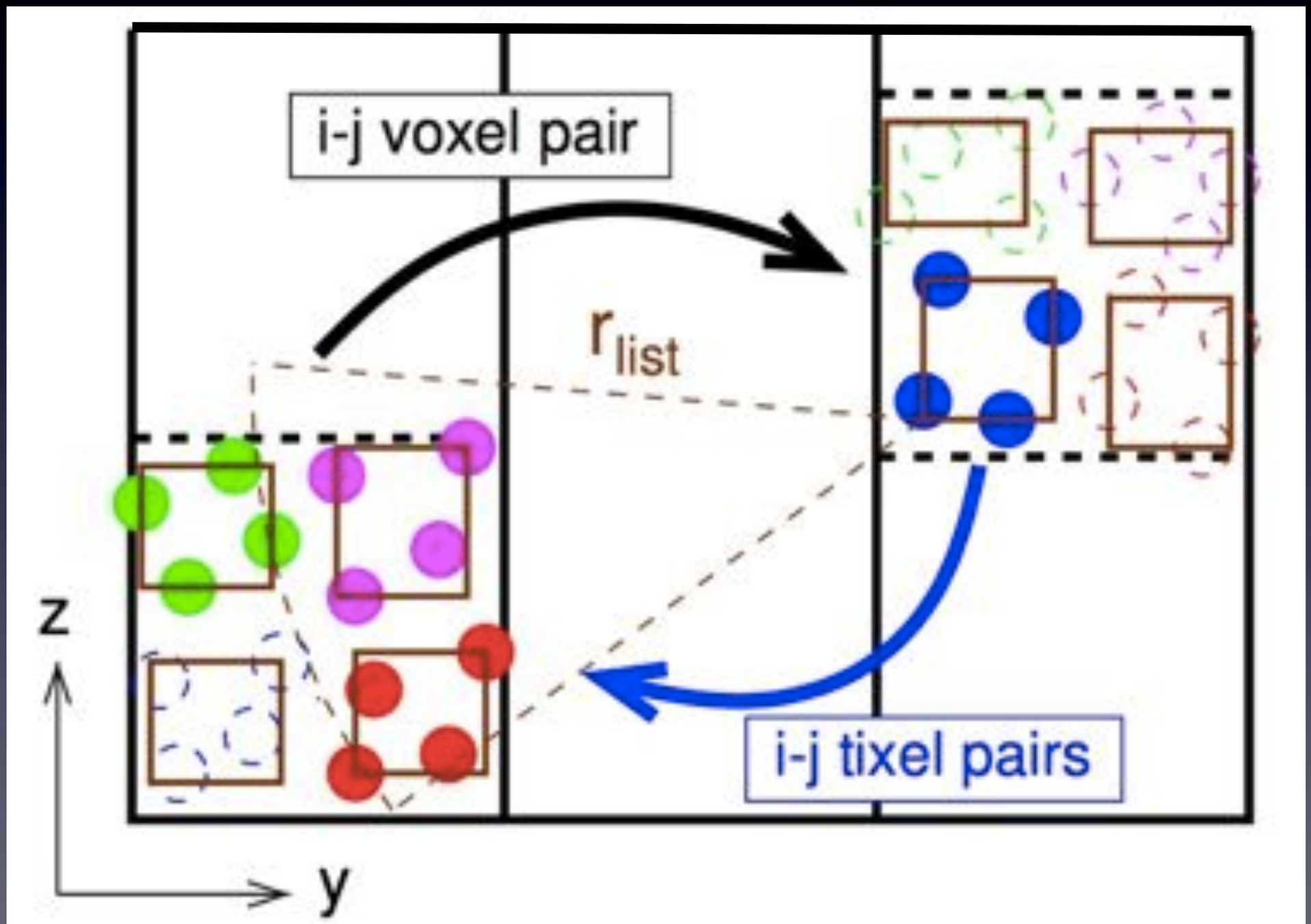
Simplified execution path



Wallclock time for an MD step:
~0.5 ms if we want to simulate 1µs/day

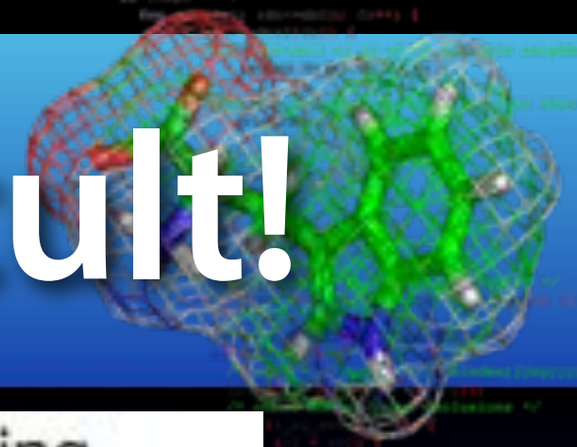
From neighborlists to pairs of proximity cells

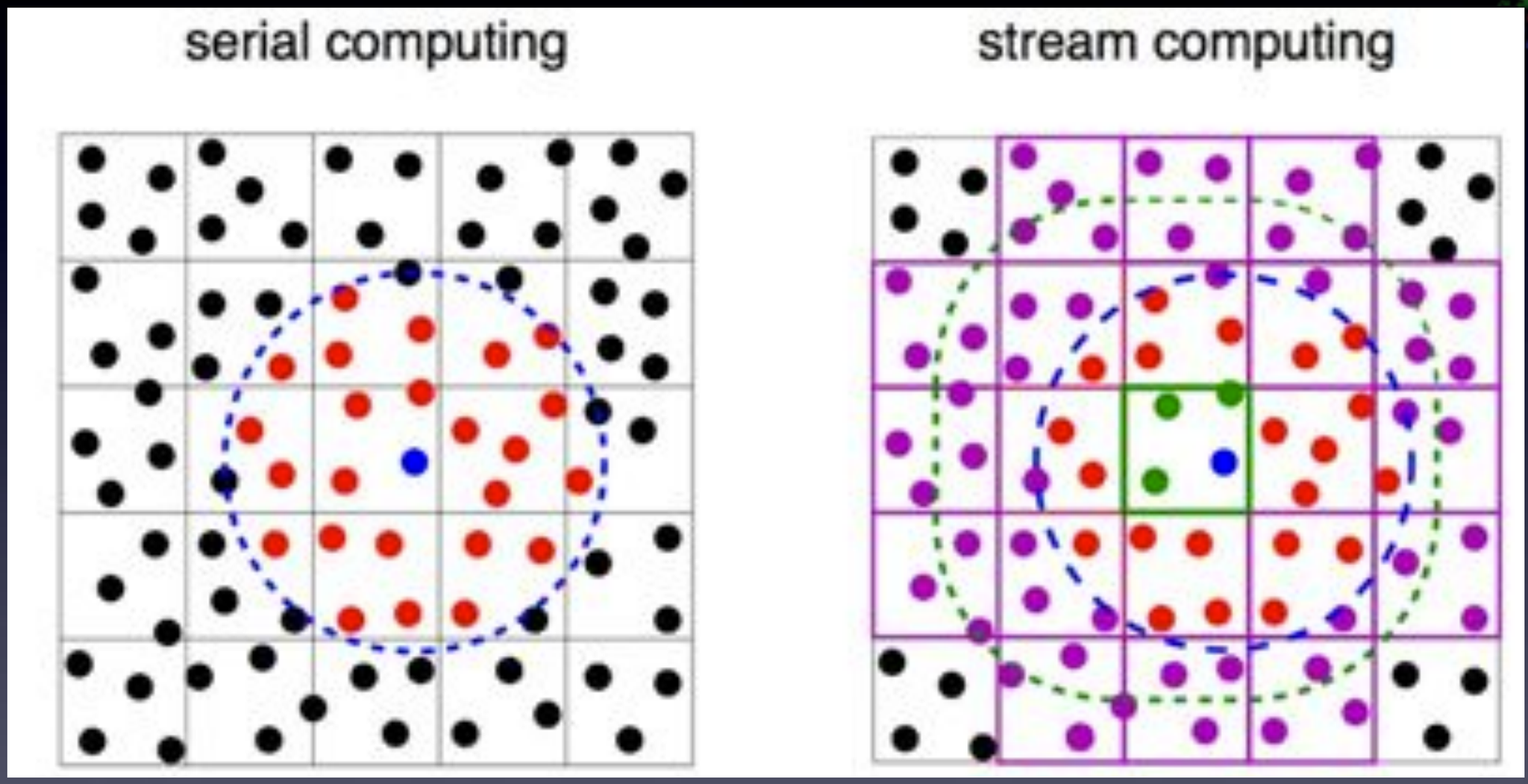




Organize as tiles with all-vs-all interactions:

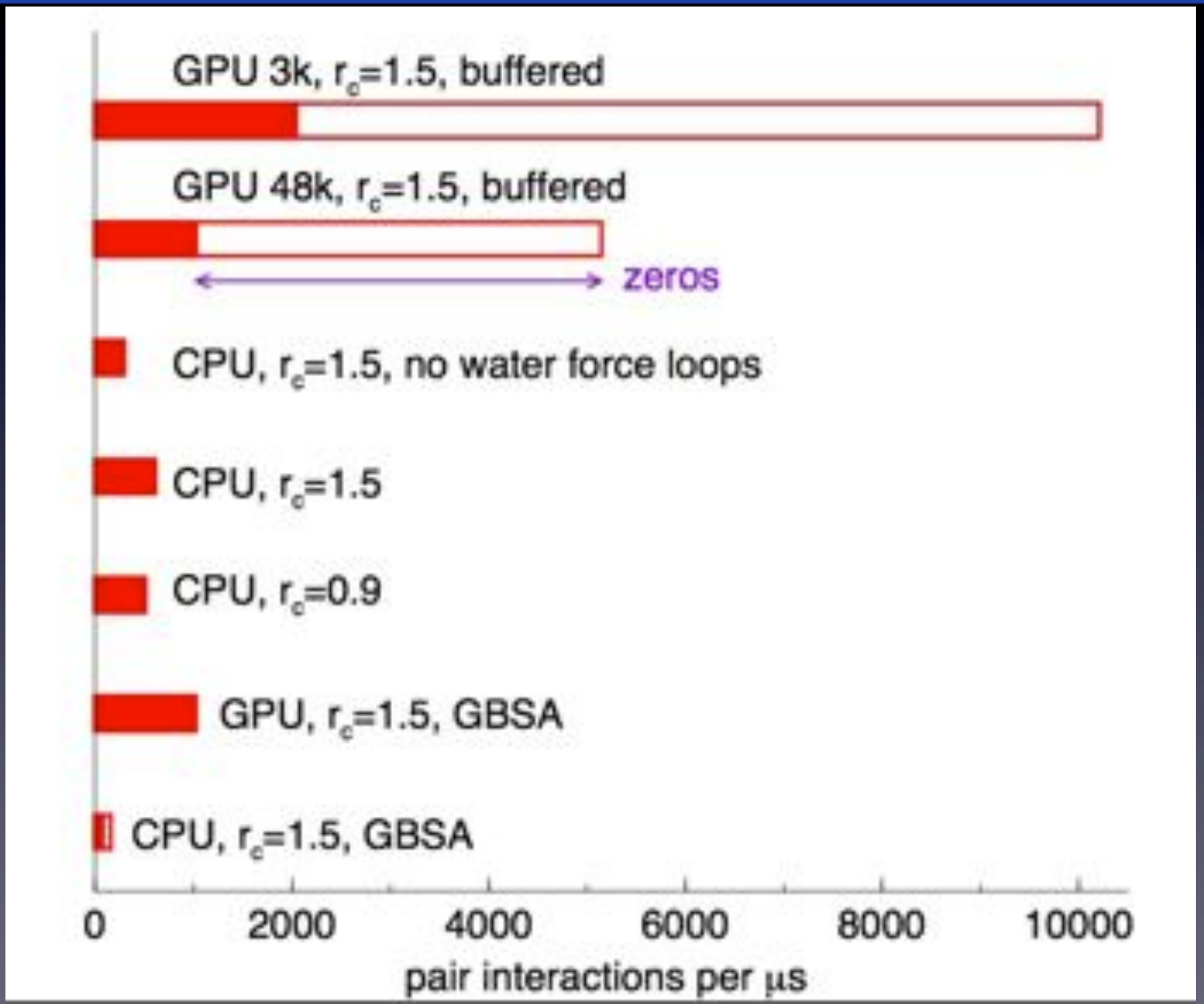
Tiling circles is difficult!





- You need a lot of cubes to cover a sphere
- All interactions beyond cutoff need to be zero

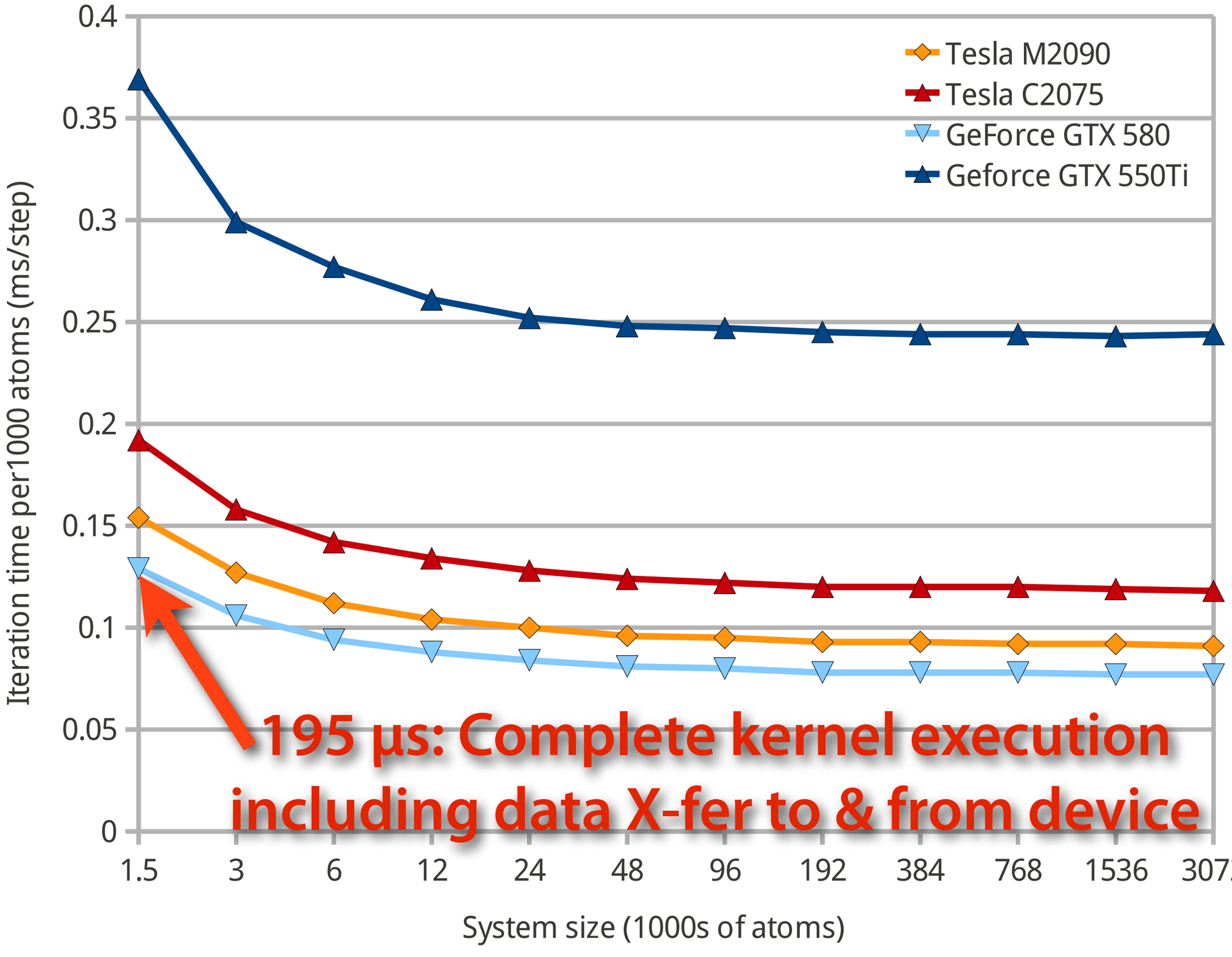
The art of calculating zeros



Kernel only

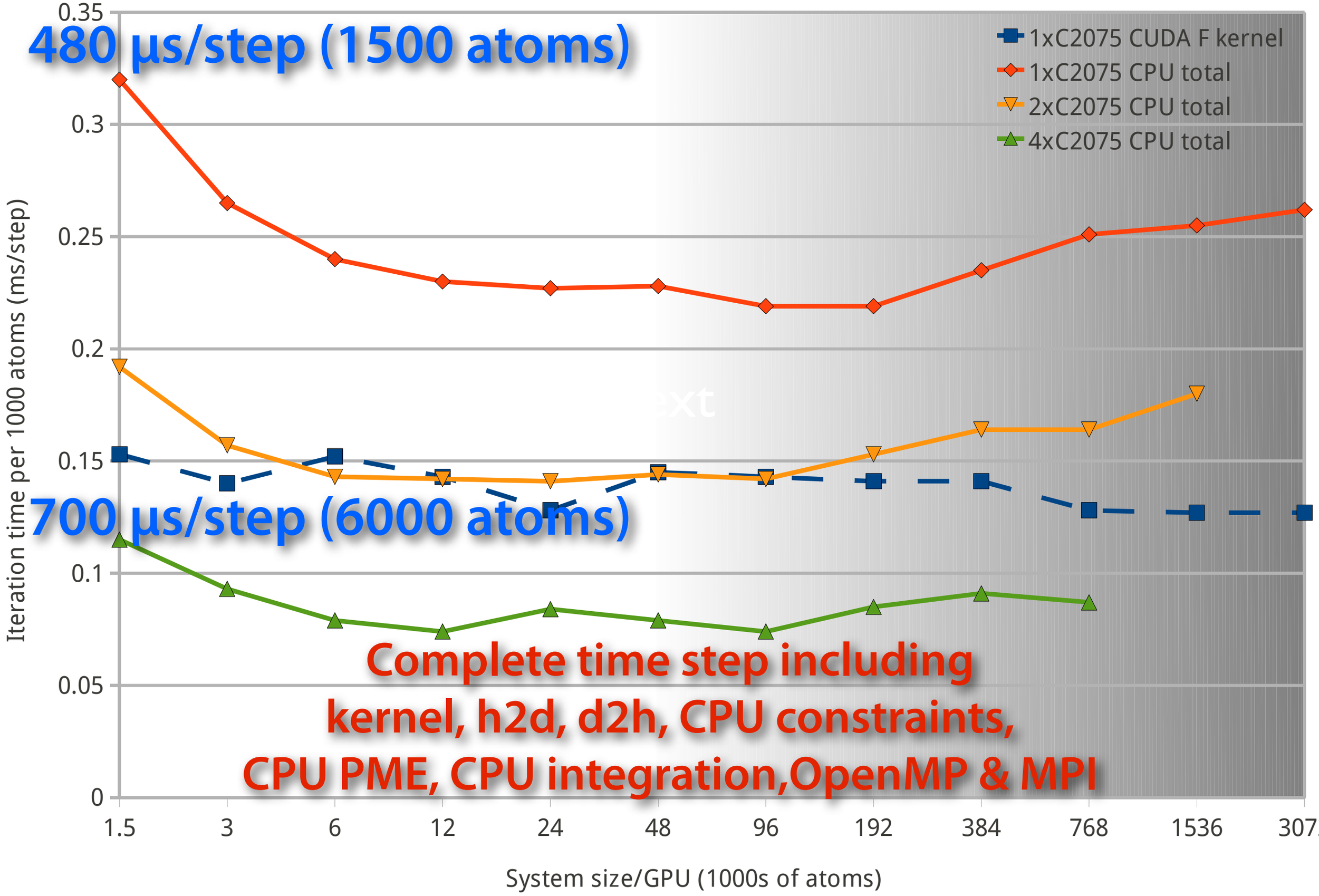
CUDA non-bonded force kernel weak scaling

PME, cutoff=1.0 nm

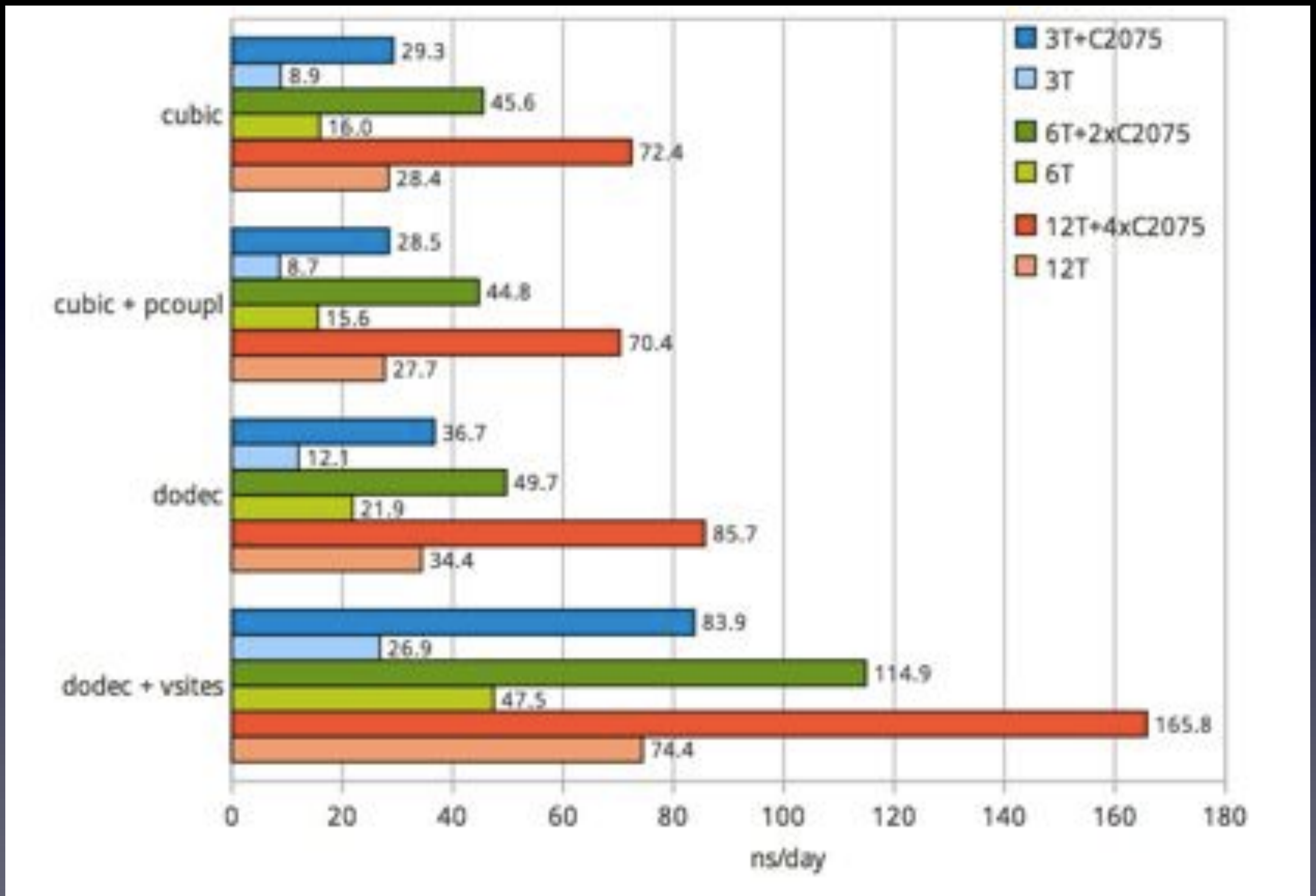


PME weak scaling

Xeon X5650 3T + C2075 / process

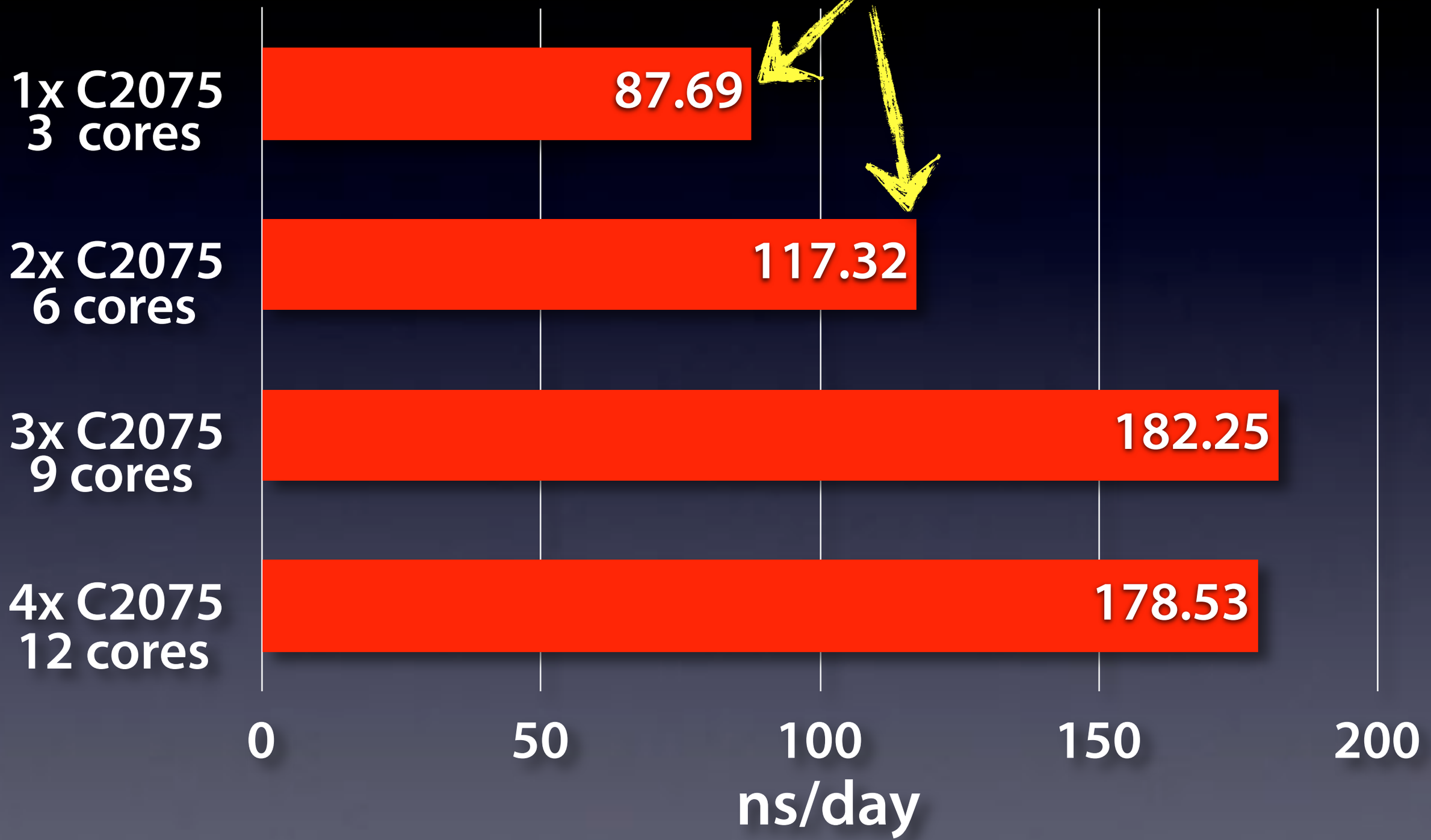


Example performance: 24,000 atom protein (ns/day)

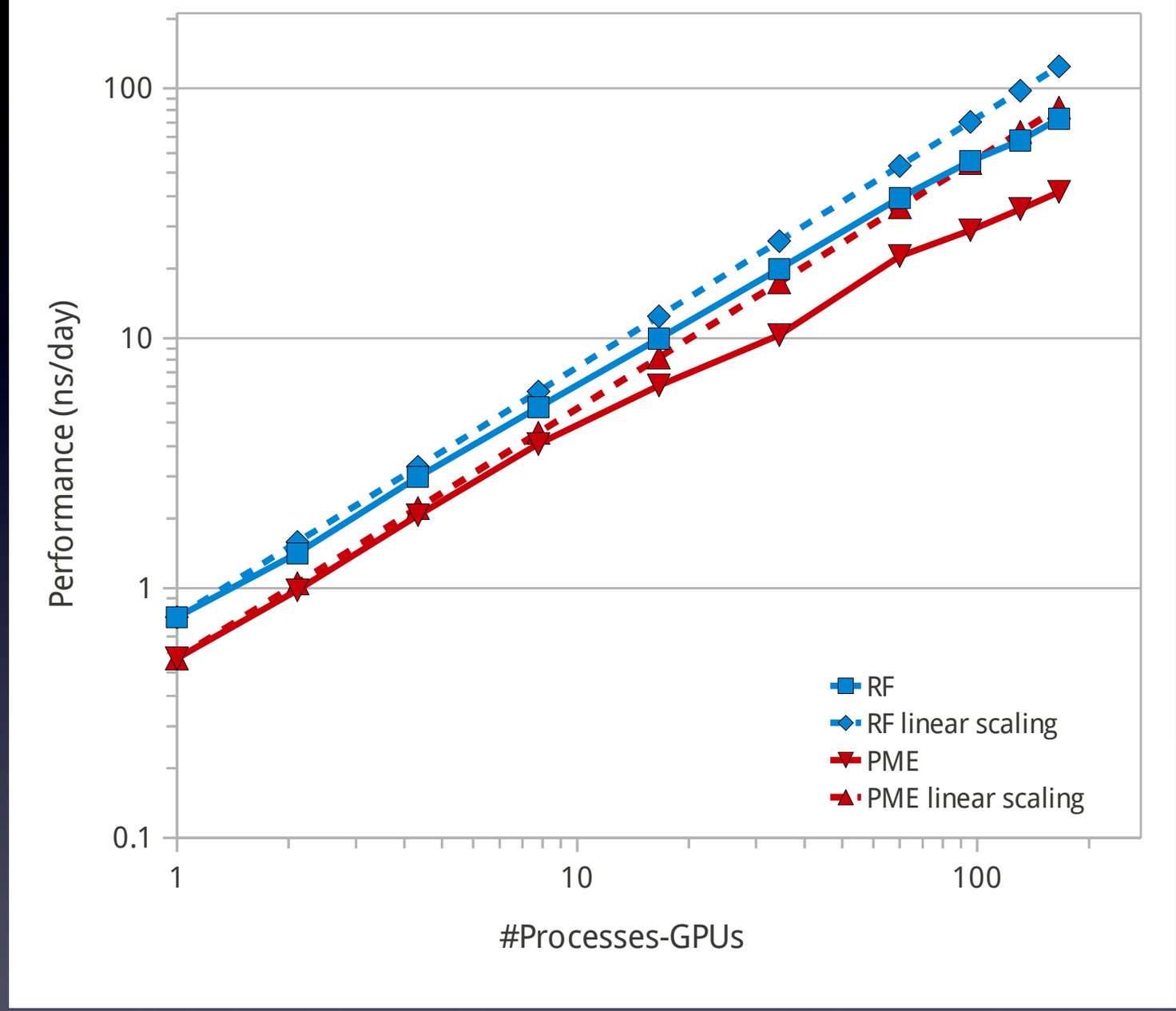


Current performance (Still 24k atoms)

Domain decomposition overhead enters from 1 to 2 GPUs



Scaling of Reaction-field & PME

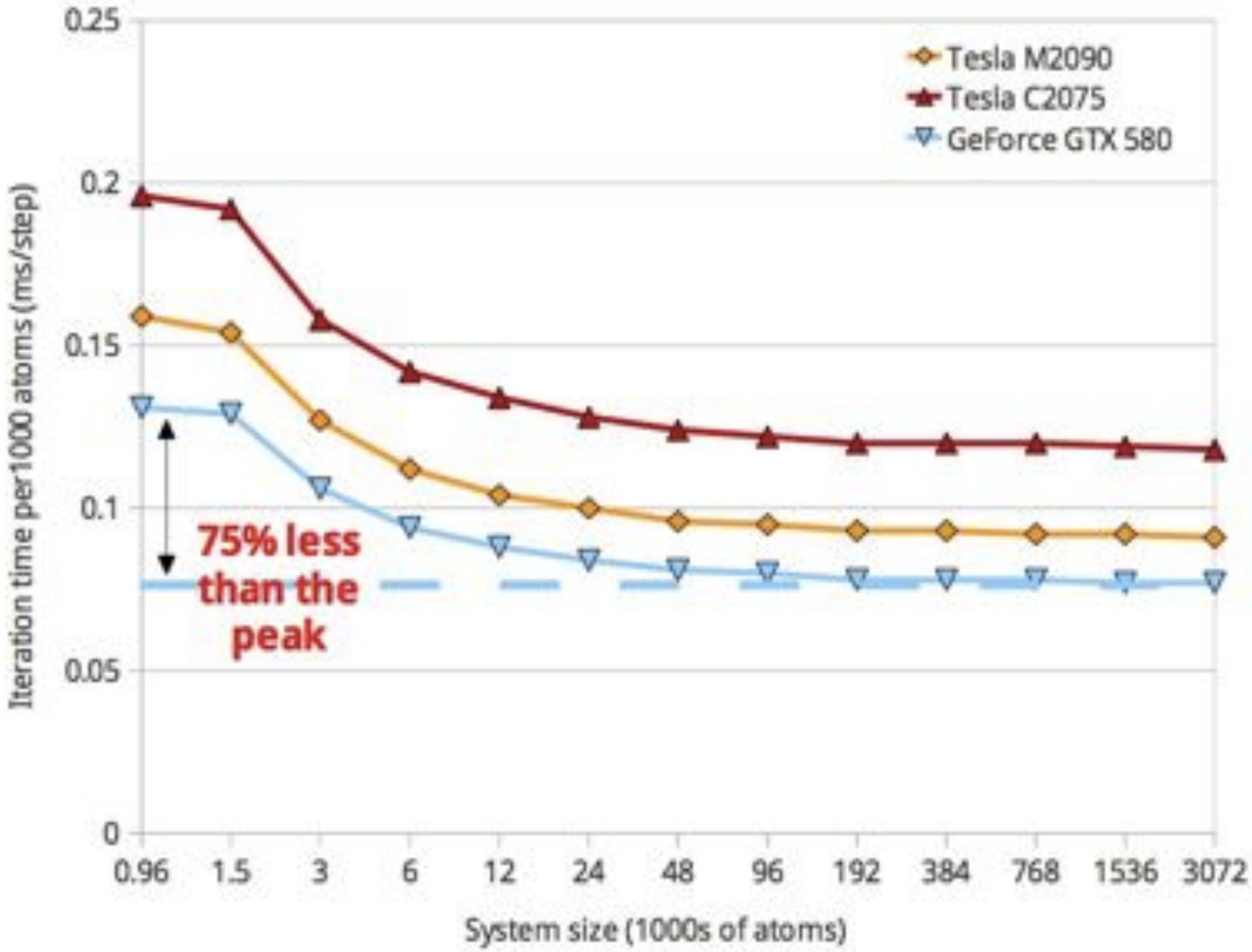


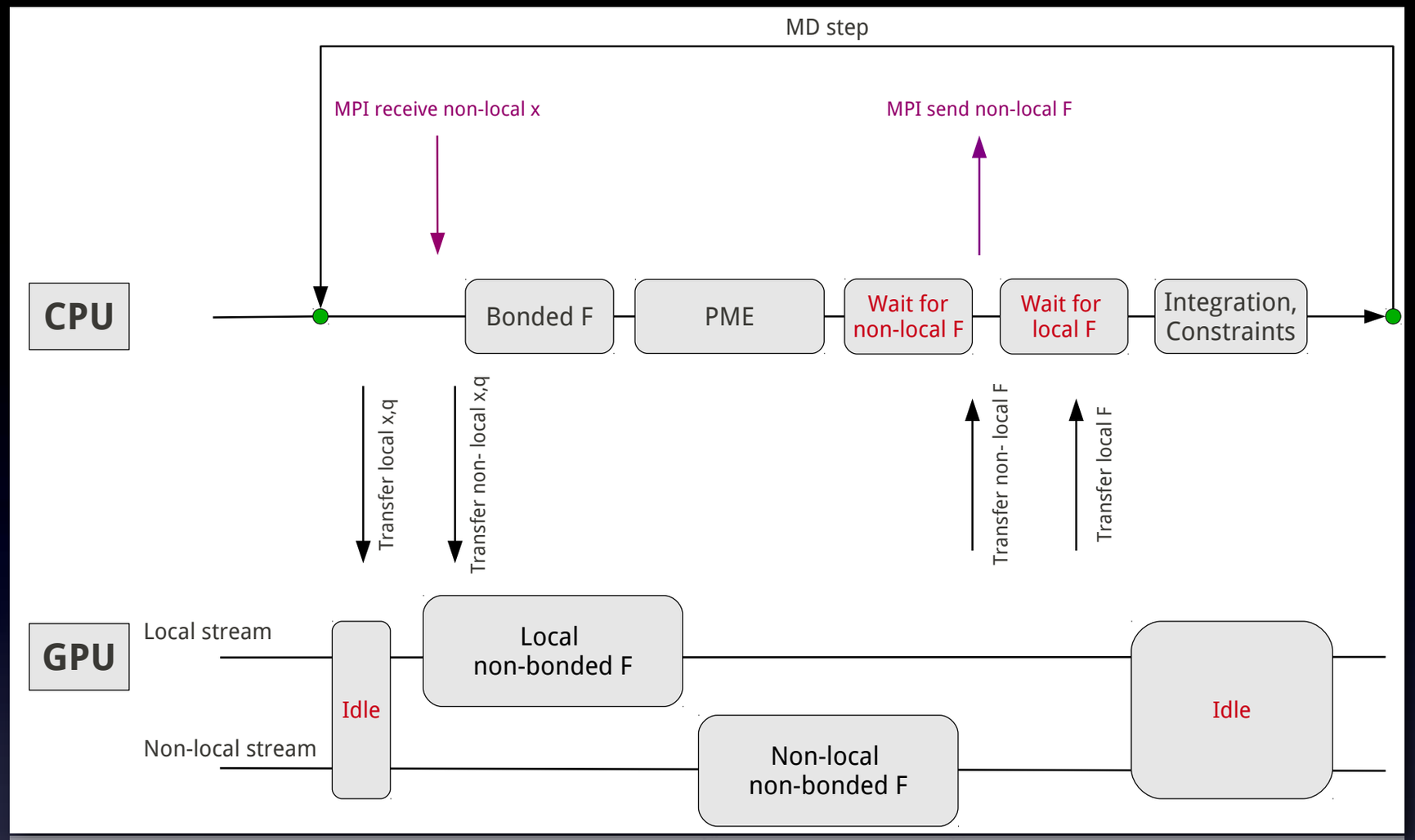
Challenge: GROMACS has very short iteration times - hard requirements on latency/bandwidth

http://www.cse.scitech.ac.uk/cbg/benchmarks/Report_II.pdf

CUDA non-bonded force kernel weak scaling

cutoff=1.0 nm



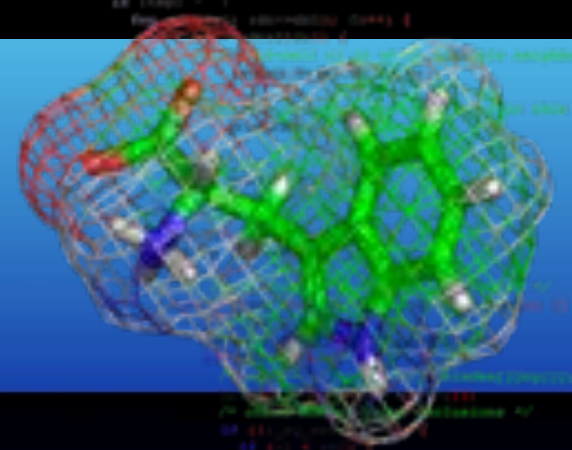


We’re essentially hitting the hard scaling limit.
Communication/calculation must overlap to proceed

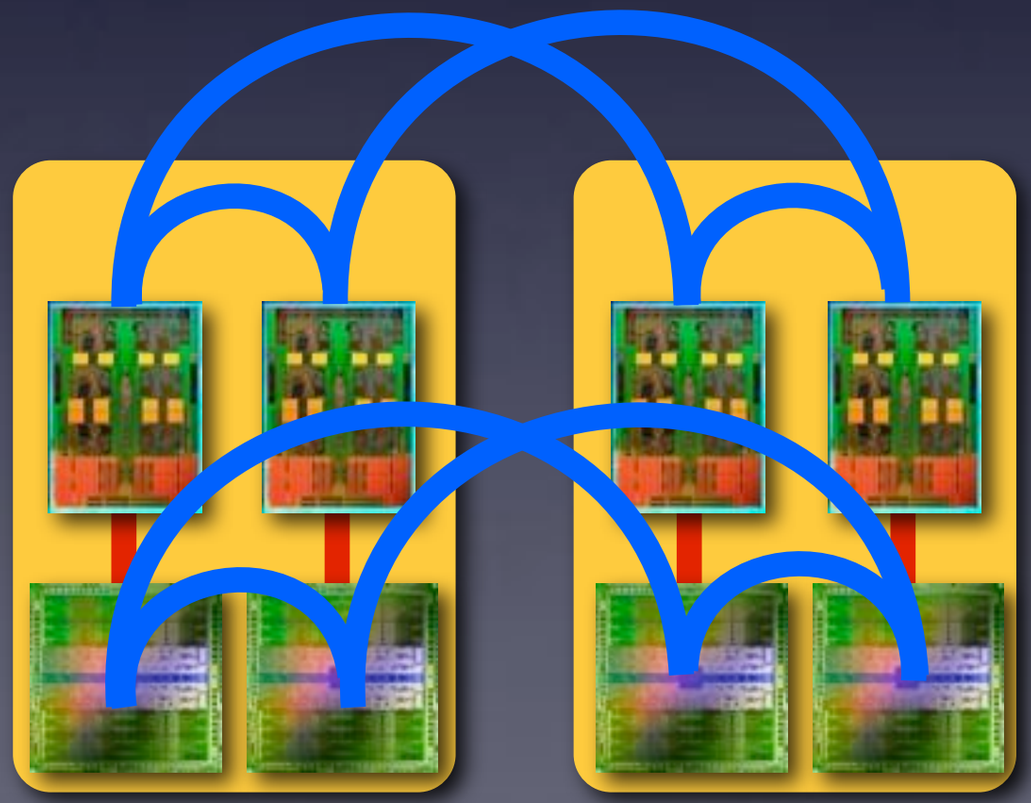
Prioritize calculations on remote data when available

At least low/high priority; more levels could be used for background tasks like compression of output data

Future directions



- Move PME to GPU if necessary
- Communicate asynchronously from GPU/CPU with other nodes if necessary
- But we want to keep the CPU for complex stuff!
- We don’t need a Xeon, but if you go ARM we need
  - Tight coupling (on die)
  - High-end future 64bit ARM
  - Flexible memory handling
  - Fine-grained thread control



Exascale?

These will soon be small computers

~2024: 1B 'cores'

2022: ~300M cores

2020: ~100M cores

2018: ~30M cores

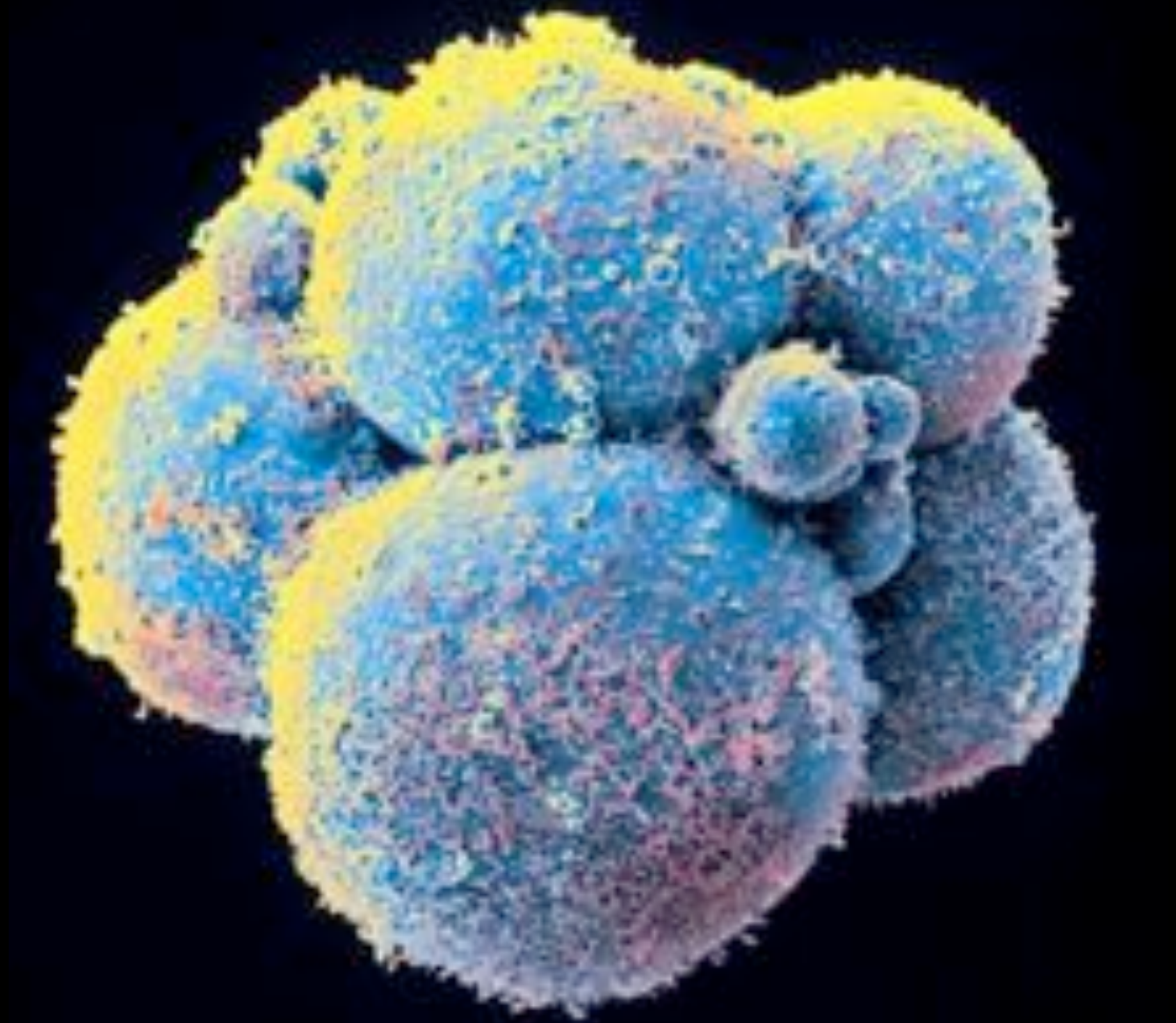
2016: ~10M cores

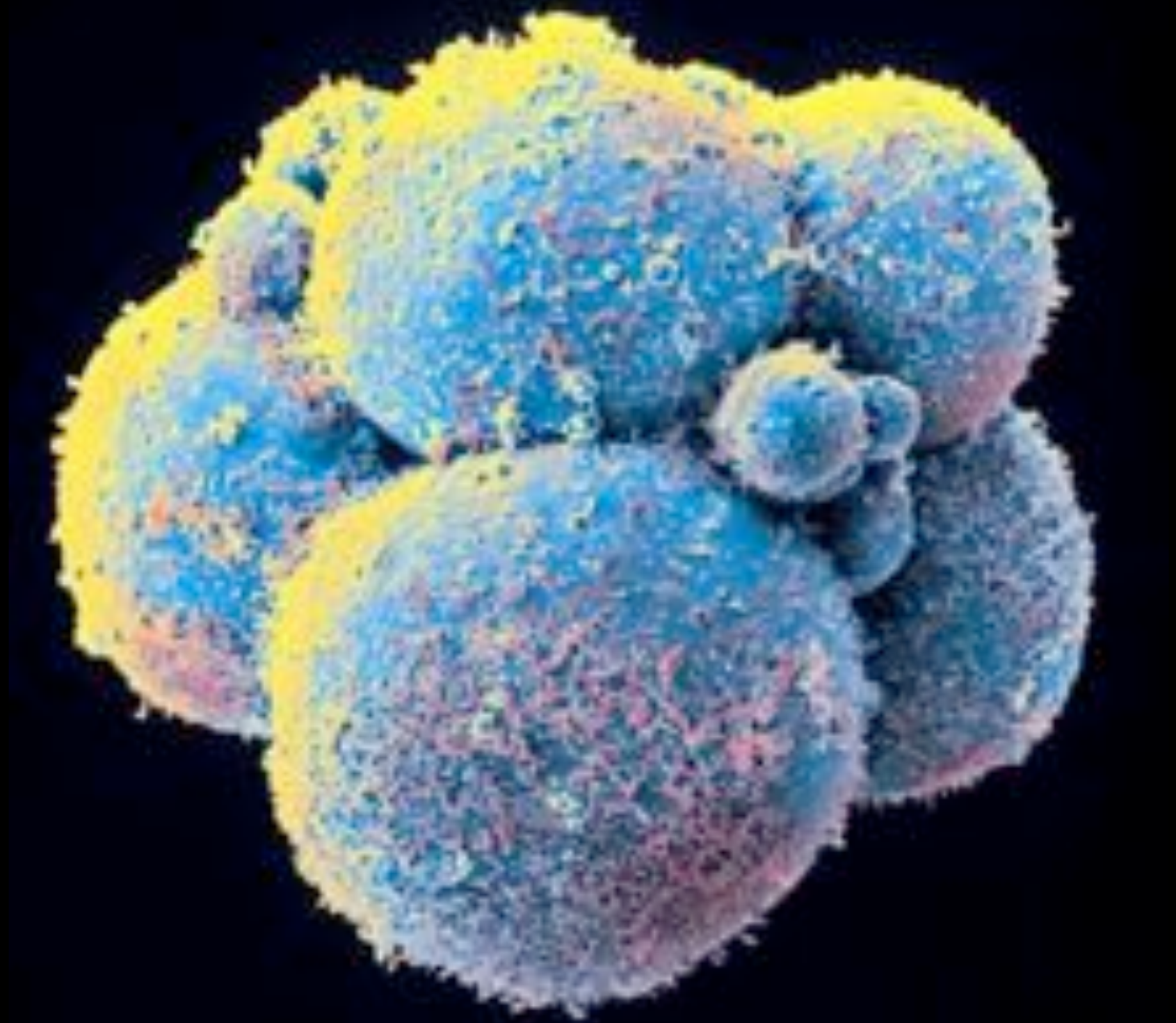
2014: ~3M cores

2012: ~1M cores

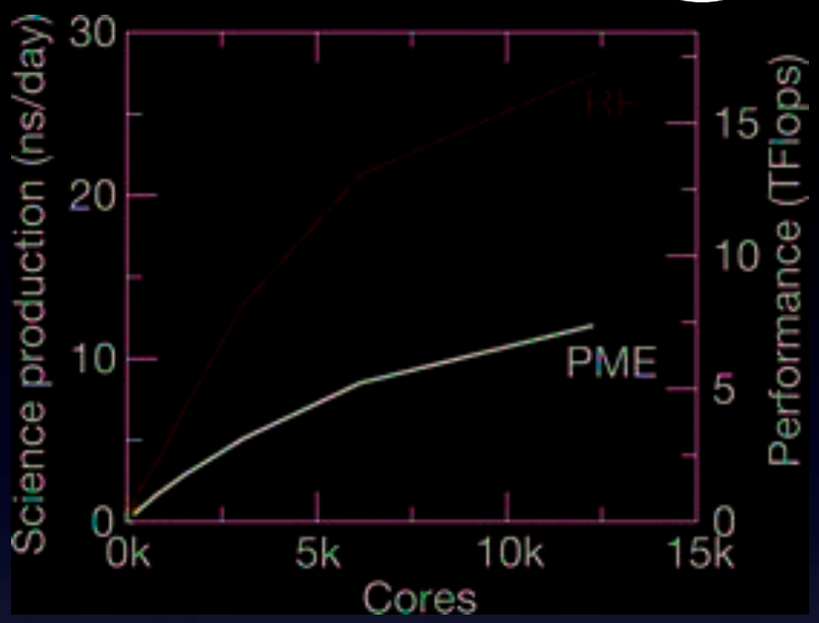
2010: ~300,000 cores

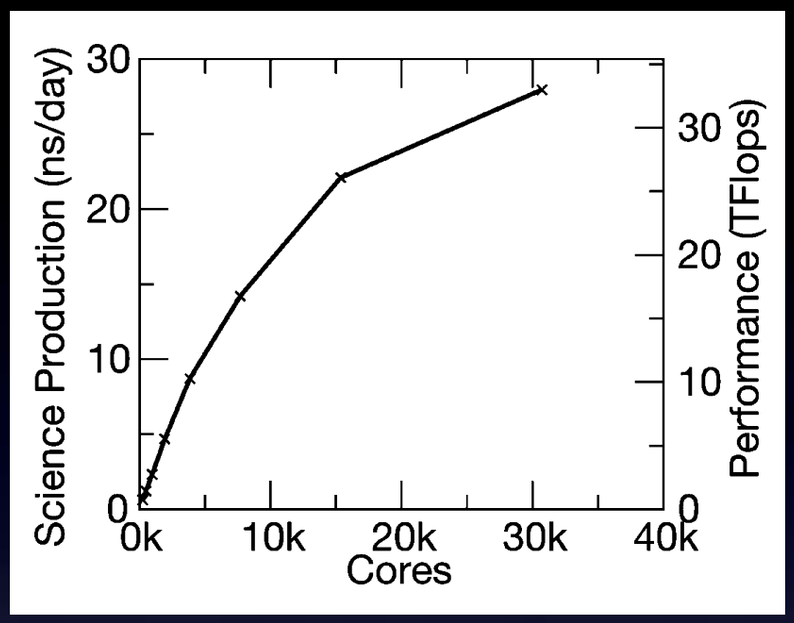
How will YOU use a billion cores?





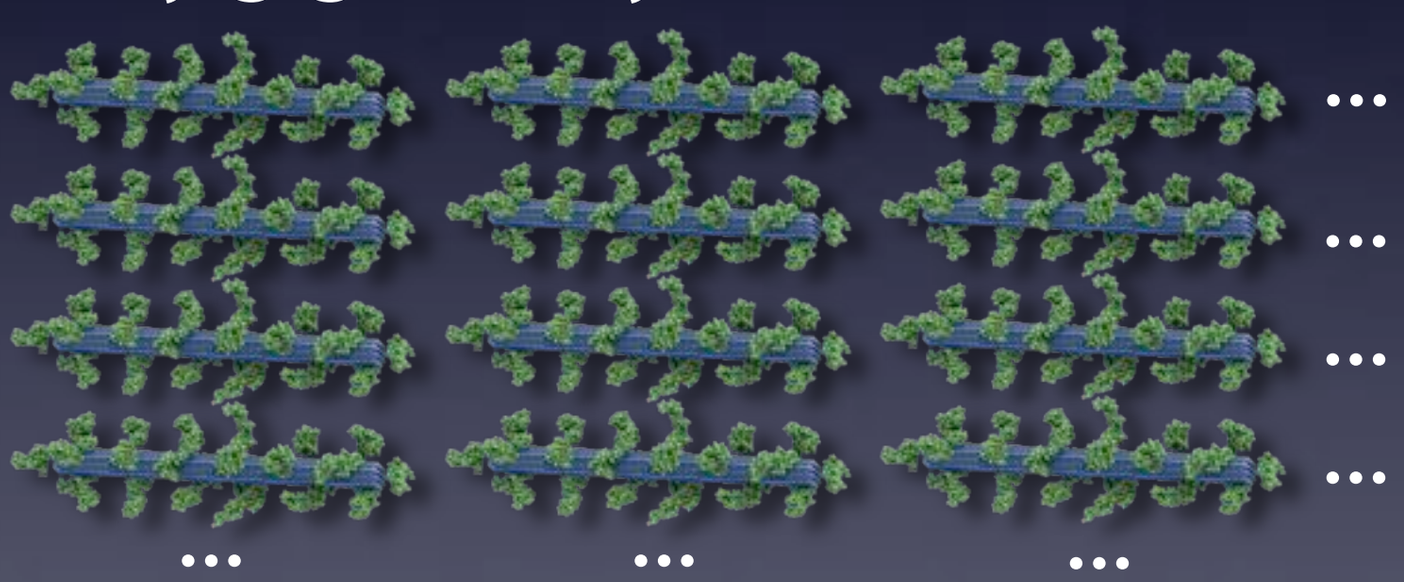
Scaling as an Obsession?





Gromacs has scaled to 150k cores on Jaguar @ ORNL

Only gigantic systems scale - limited number of applications

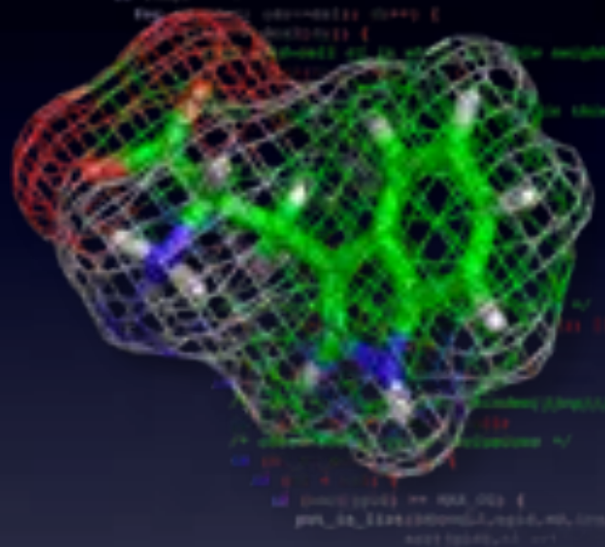


1M-100M atoms

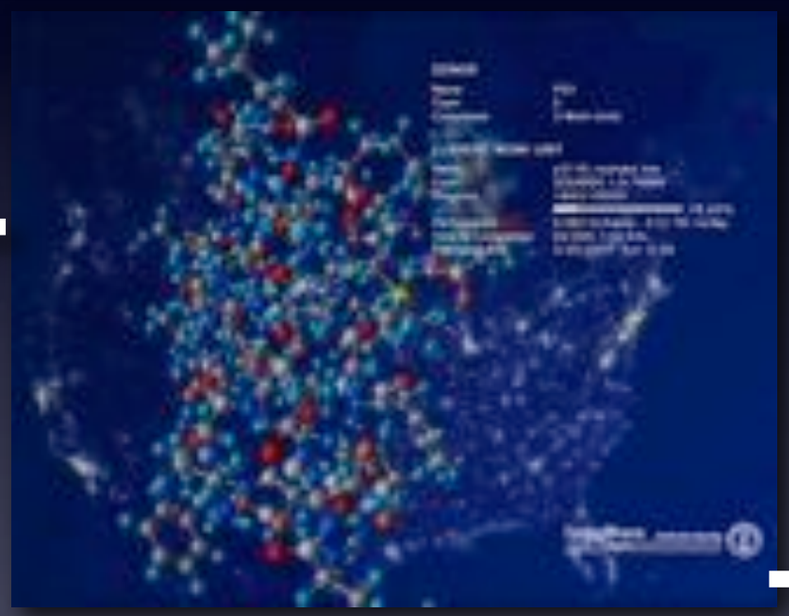
But: <u>Small</u> systems won't scale to large numbers of cores!

How shall we break this impasse?

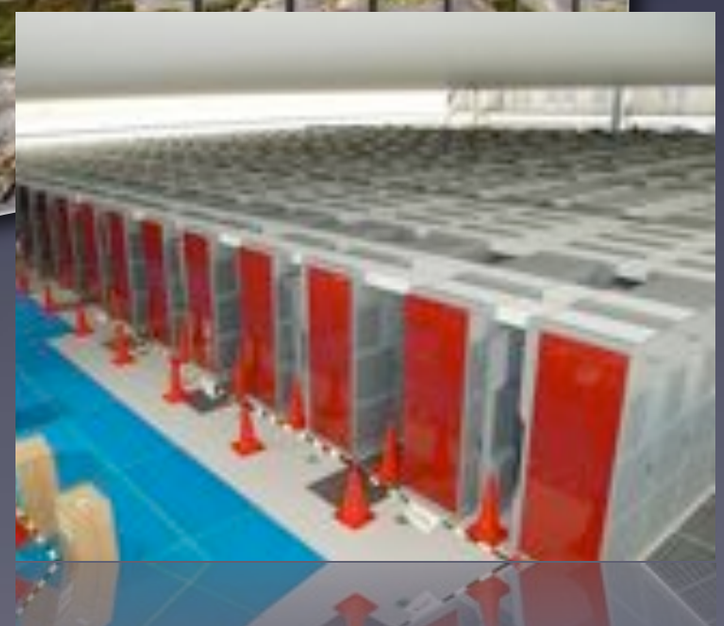
A New Open Source Marriage: The Copernicus Project



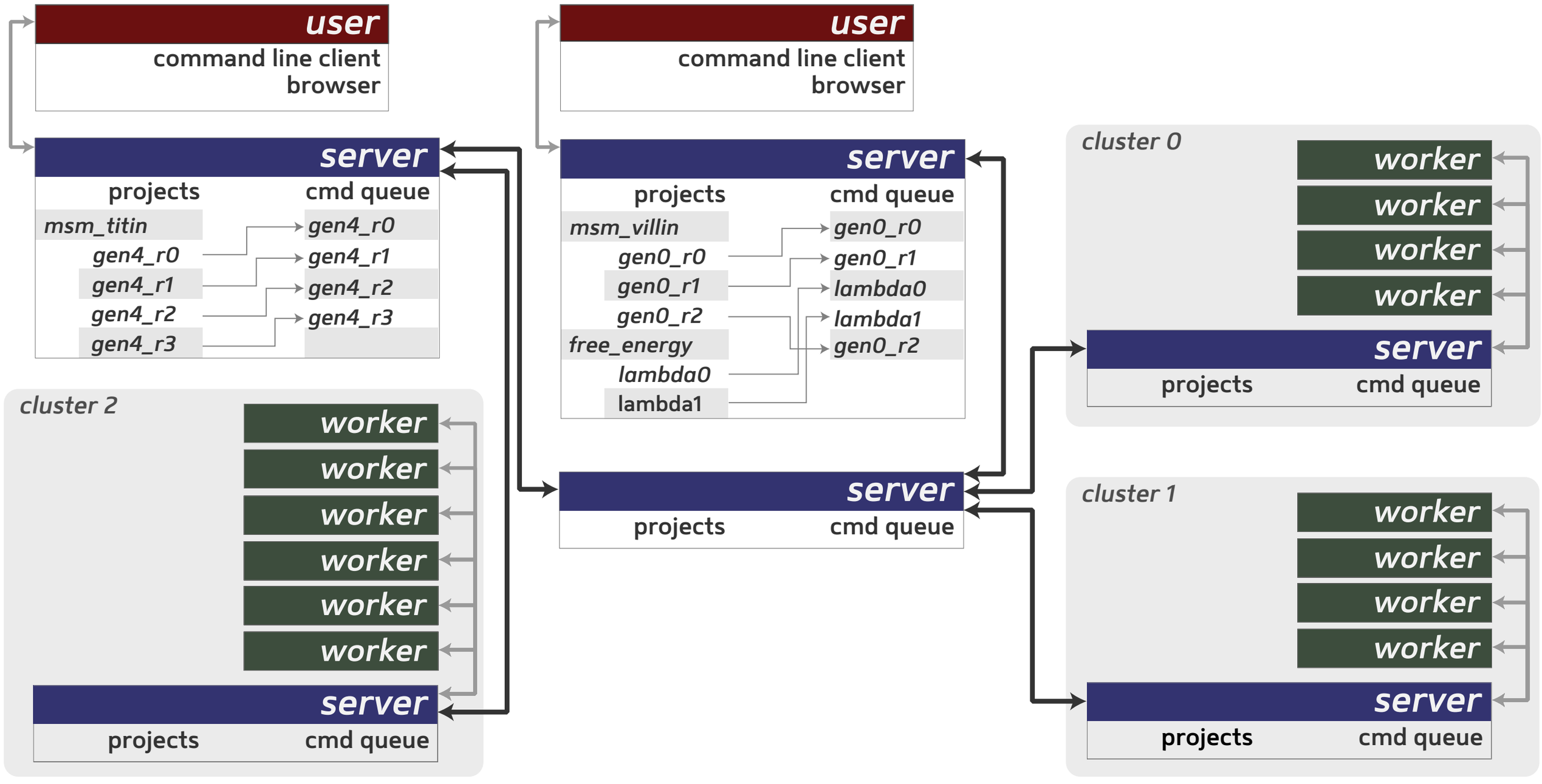
+

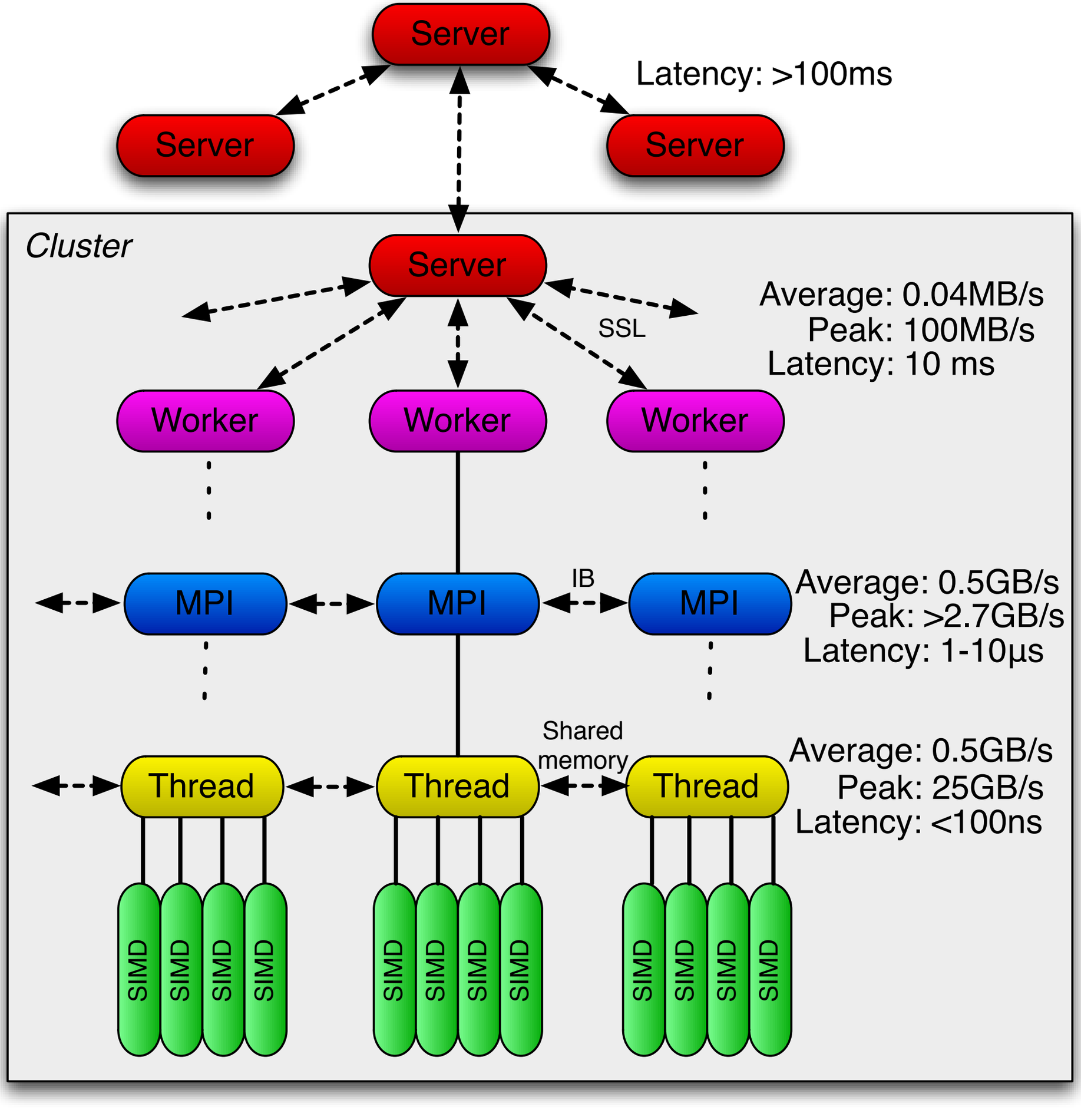


+

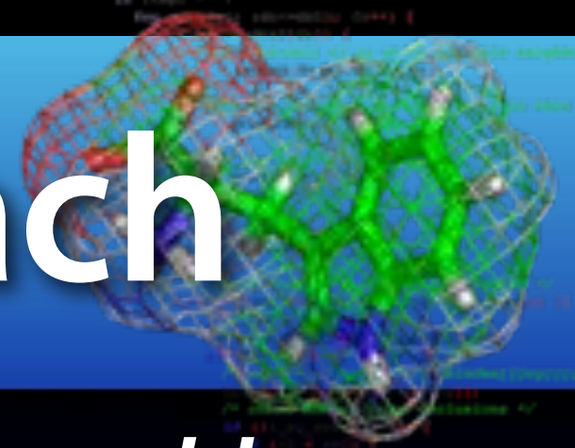


Copernicus

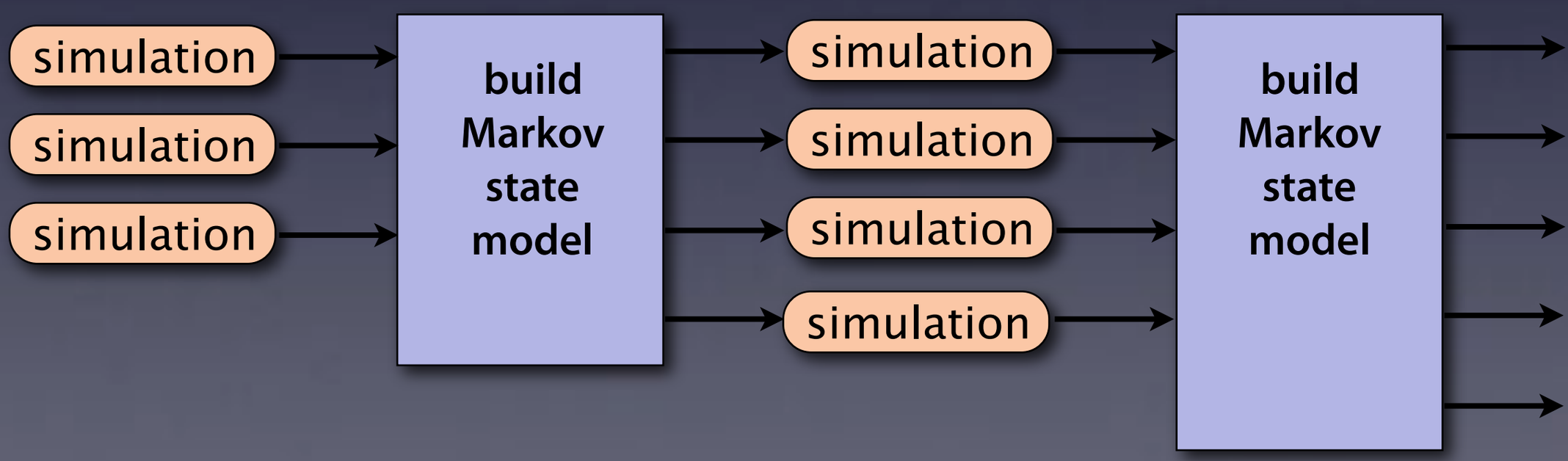

user
command line client
browser
user
command line client
browser
server
projects
cmd queue
msm_titin
gen4_r0
gen4_r1
gen4_r2
gen4_r3
gen4_r0
gen4_r1
gen4_r2
gen4_r3
server
projects
cmd queue
msm_villin
gen0_r0
gen0_r1
gen0_r2
free_energy
lambda0
lambda1
gen0_r0
gen0_r1
lambda0
lambda1
gen0_r2
cluster 0
worker
worker
worker
worker
worker
server
projects
cmd queue
cluster 2
worker
worker
worker
worker
worker
worker
server
projects
cmd queue
server
projects
cmd queue
cluster 1
worker
worker
worker
worker
worker
server
projects
cmd queue


Server
Server
Server
Latency: >100ms
Cluster
Server
SSL
Average: 0.04MB/s
Peak: 100MB/s
Latency: 10 ms
Worker
Worker
Worker
MPI
MPI
IB
MPI
Average: 0.5GB/s
Peak: >2.7GB/s
Latency: 1-10µs
Shared
memory
Thread
Thread
Thread
Average: 0.5GB/s
Peak: 25GB/s
Latency: <100ns
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD
SIMD

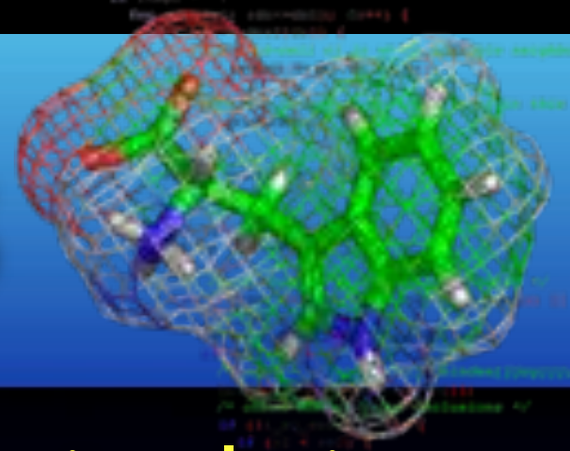
A different approach

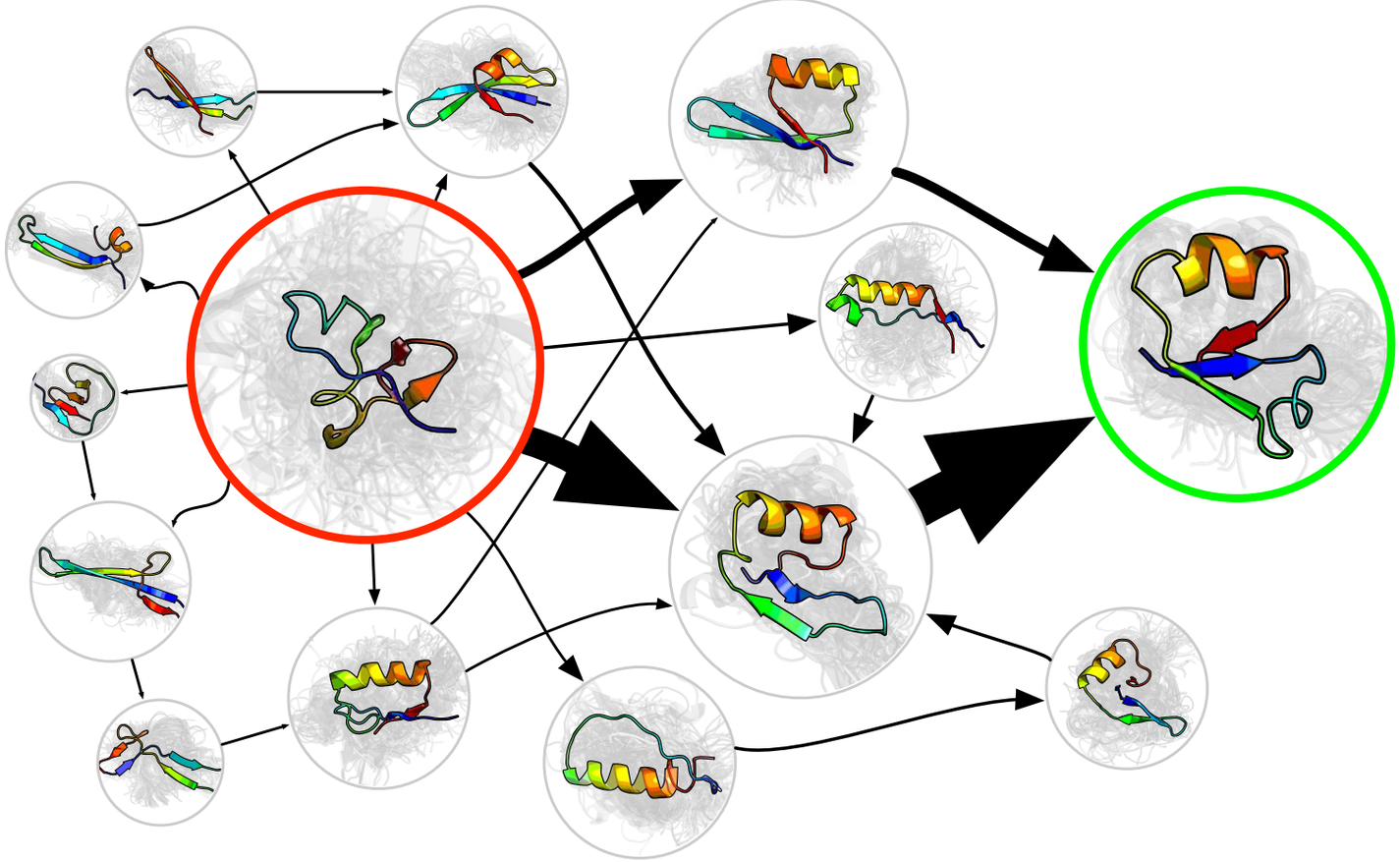


- Each project is really a simulation *ensemble*
- As a user, I don't run a single trajectory, but e.g.
  - A markov state model of dynamics
- A free energy calculation for 1,000 compounds



Markov State Models





Vince Vaughn

Start many simulations

Cluster conformations

Identify macrostates

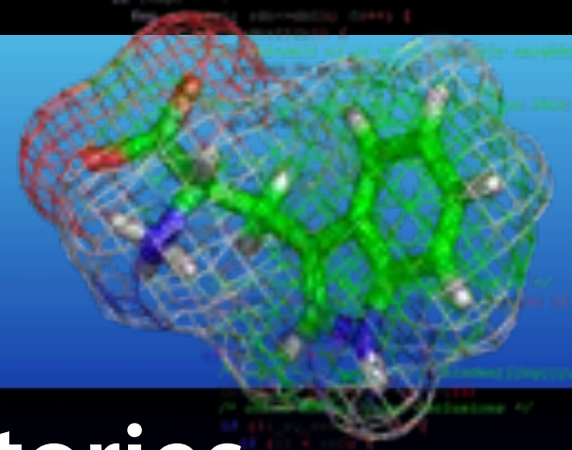
Calculate transition rates between them

Restart from states with least sampling

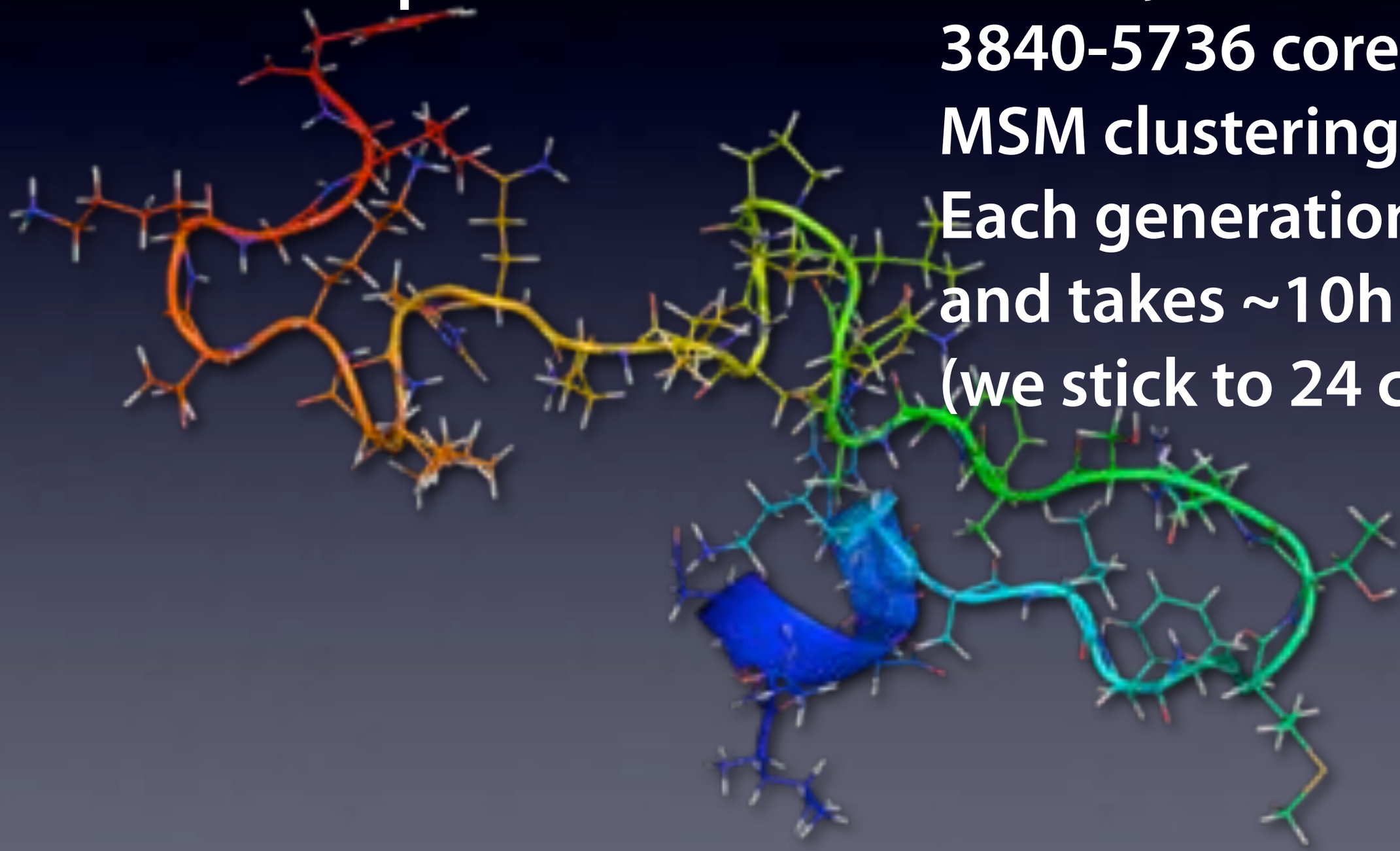
Monitor convergence of eigenvalues of transition matrix

Ensemble simulation is not an approximation - chemistry is ALL about ensembles!

Copernicus in action

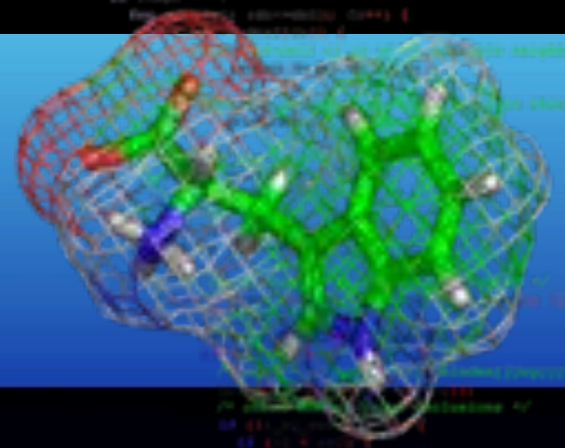


The villin headpiece

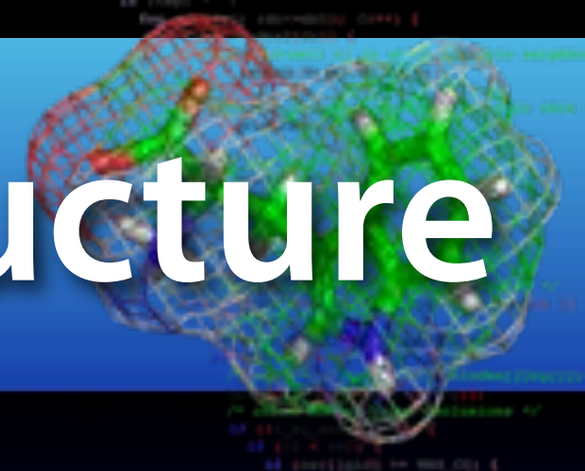


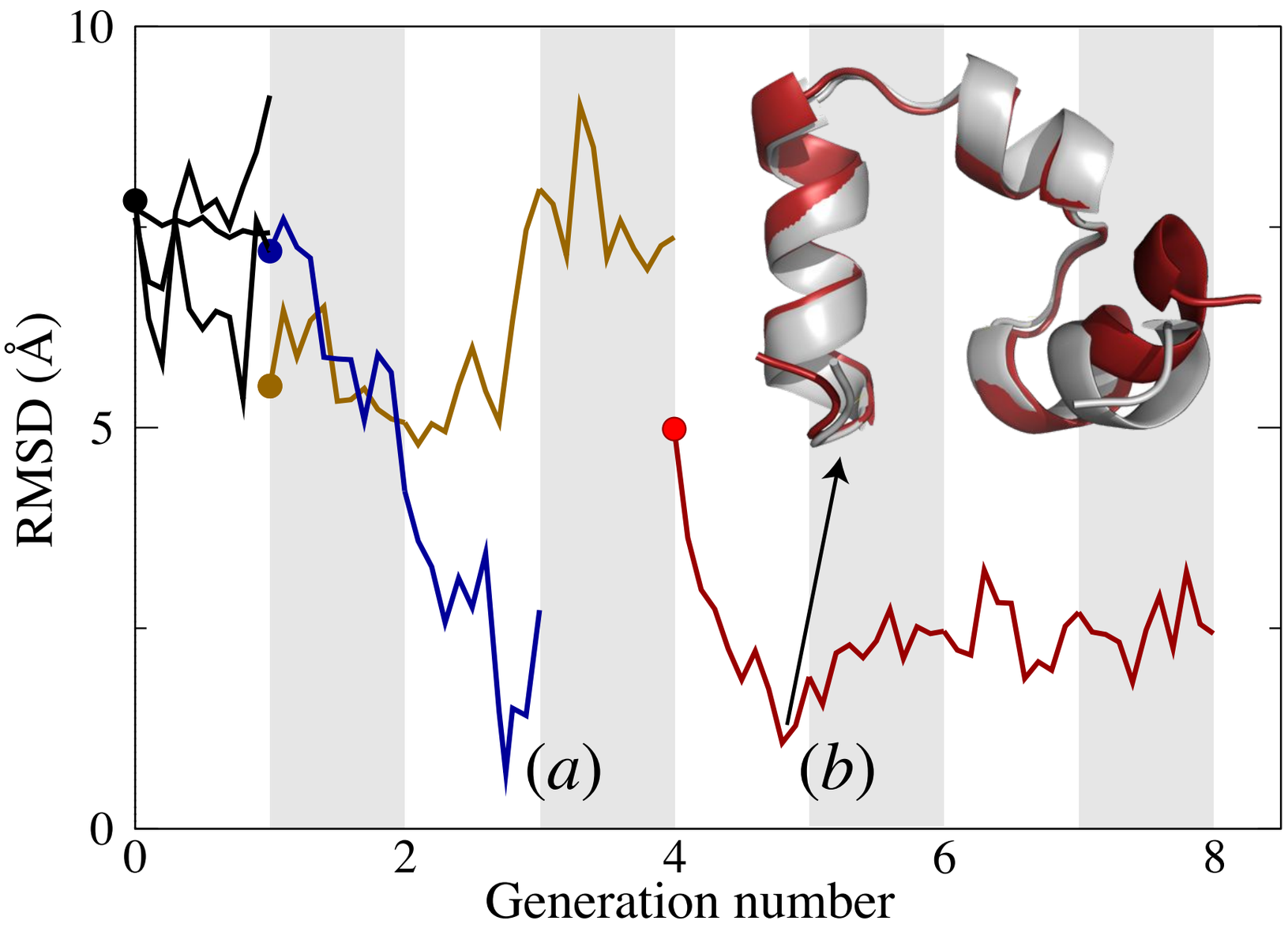
244 trajectories
3840-5736 cores used
MSM clustering
Each generation 50ns,
and takes ~10h to run
(we stick to 24 cores)

30 hours later



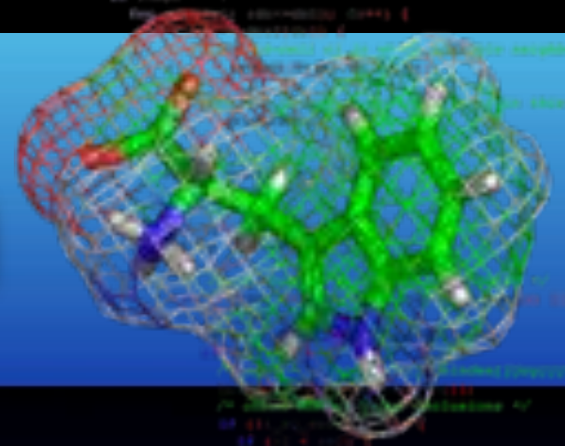
We can *predict* the structure

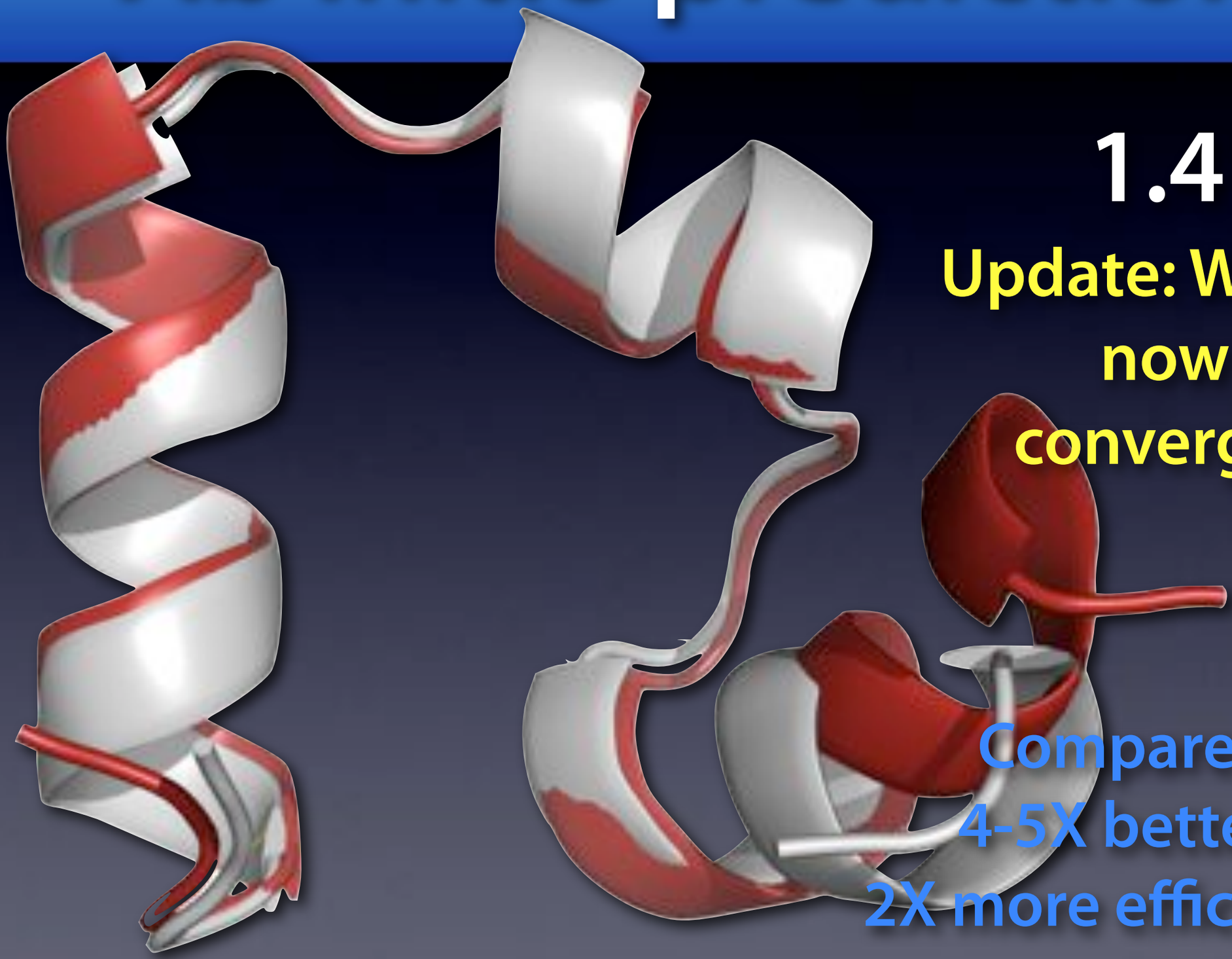




Convergence of transition state matrix after ~72h

Ab initio prediction



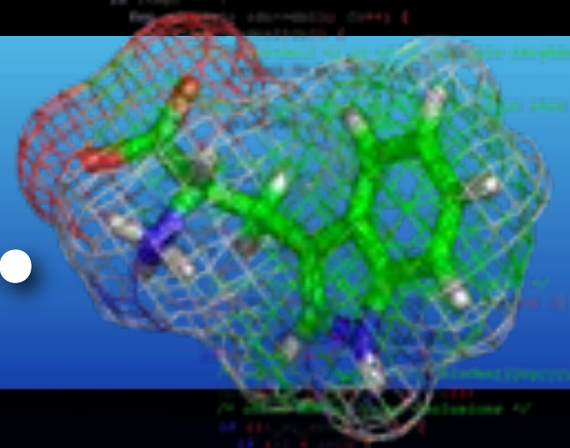


1.4 Å RMSD

Update: With vsites we now achieve TSM convergence in 46h!

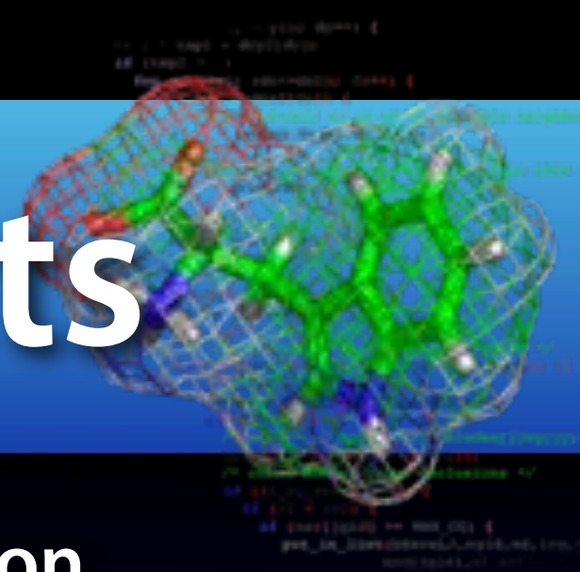
Compared to ASIC HW:
4-5X better throughput
2X more efficient sampling
10X total

Sort of a vision...



- Multithreading & multigrid to get lattice-based algorithms to scale efficiently
- Individual simulation parts scaling to >10,000 cores even for small systems
  - (Large systems will scale to anything)
- Ensembles of ~1000 active simulations that exchange MSM data as a single job
- While we hope for more general stream processors than GPUs, this is not the bottleneck

Acknowledgments



- **GROMACS:** Berk Hess, David van der Spoel, Per Larsson
- **Gromacs-GPU:** Szilard Pall, Berk Hess, Rossen Apostolov
- **Multi-Threaded PME:** Roland Shultz, Berk Hess
- **Copernicus:** Sander Pronk, Iman Pouya, Peter Kasson, Vijay Pande
- **Nvidia:** Mark Berger, Scott LeGrand, Duncan Poole,Andrew Walsh

erc European Research Council

SJUNDE RAMPROGRAMMET

Vetenskapsrådet

Stiftelsen för Strategisk Forskning

NVIDIA

AMD