bioexcel
Centre of Excellence for Computational Biomolecular Research

Horizon 2020
European Union Funding
for Research & Innovation

Maximizing Gromacs performance for HTC and HPC

Christian Blau

blau@kth.se

Typical Simulation Time Scales

- Interesting biology at microsecond time-scale

- Simulation step at 2-4fs
- → need 10^10 time steps

- Biomolecular processes are stochastic
- → average over simulation ensembles
- → for solid answers, need up to 10^15 simulation steps

- Best performance is at 100 microseconds / step
- → typical simulations run days to months

- → increase performance

Why work on increasing performance

Pro

- Decrease time to result, less time needed for research
- More precision with more sampling
- New discoveries
  - sample protein conformations that were inaccessible
  - apply methods that were inaccessible before
- Less resource cost (energy, computers, compute center maintenance)

Why not work on increasing performance

Contra

- Increased complexity
  - risk of bugs, more work and time needed to understand
  - algorithms
  - implementation code
  - simulation setup
- Increased complexity in algorithms
- Vendor dependency
  - Newest hardware is often proprietary
- Constant “catch up” game

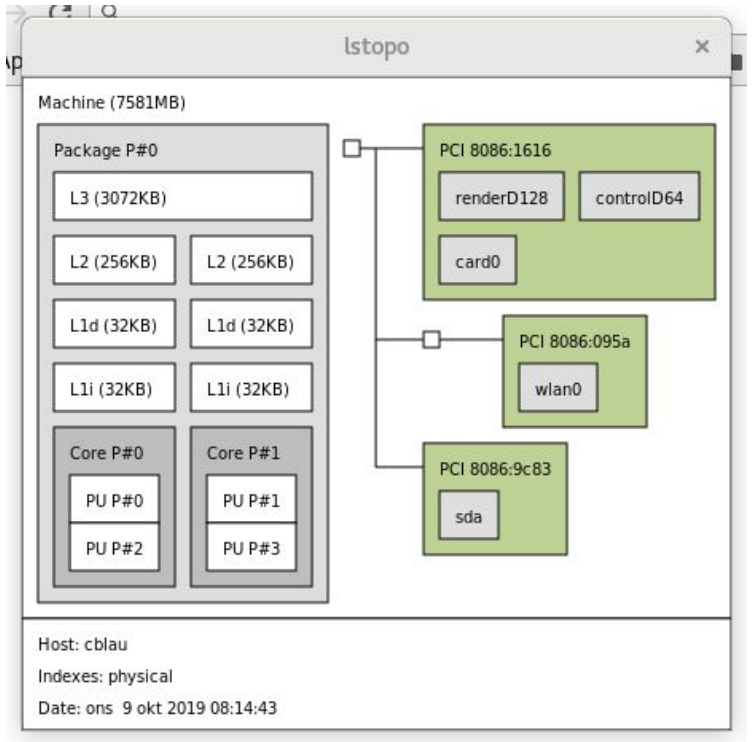
Concepts to increase performance

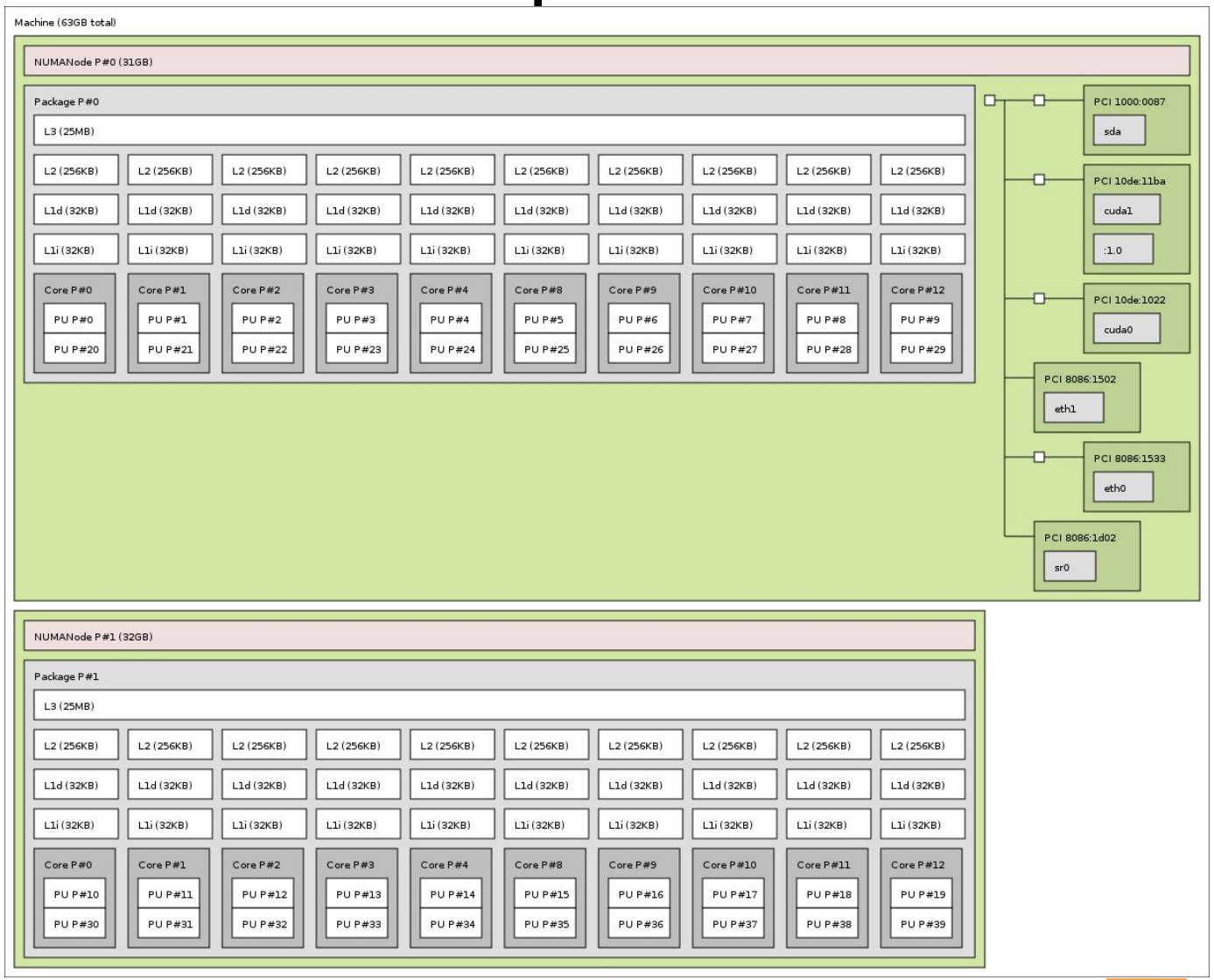
- Avoid simulations
  - carefully design your computer experiments
  - simulate only systems that have a chance of giving results
- Use good sampling algorithms
  - Avoid sampling uninteresting conformations
  - Get the physics right with the least effort
- Use good implementations
  - Use the newest software version - GROMACS CurrentYear.highestNumber
  - Compare software packages
  - Use the best other libraries

Concepts to increase performance

- Use expert knowledge where available
  - Read and follow advice from super-computing centers
  - Read the manual and documentation
- Use a good “build”
  - Use the newest compilers available
  - Check compilation guidelines and options
- Use good hardware
  - GPUs give large performance gains

Modern hardware is complex





Laptop above,

Node at compute center right

Strategies to Achieve Answers in Appropriate time

Use all power of a single core

- Modern processors use SIMD instructions
- (single instruction multiple data)
- Small C++ SIMD layer with highly optimised math functions for
  - SSE2/4, AVX2-128, AVX(2)-256, AVX-512, ARM, VSX, HPC-ACE
- CPU kernels reach 50% of peak

Strategies to Achieve Answers in Appropriate time

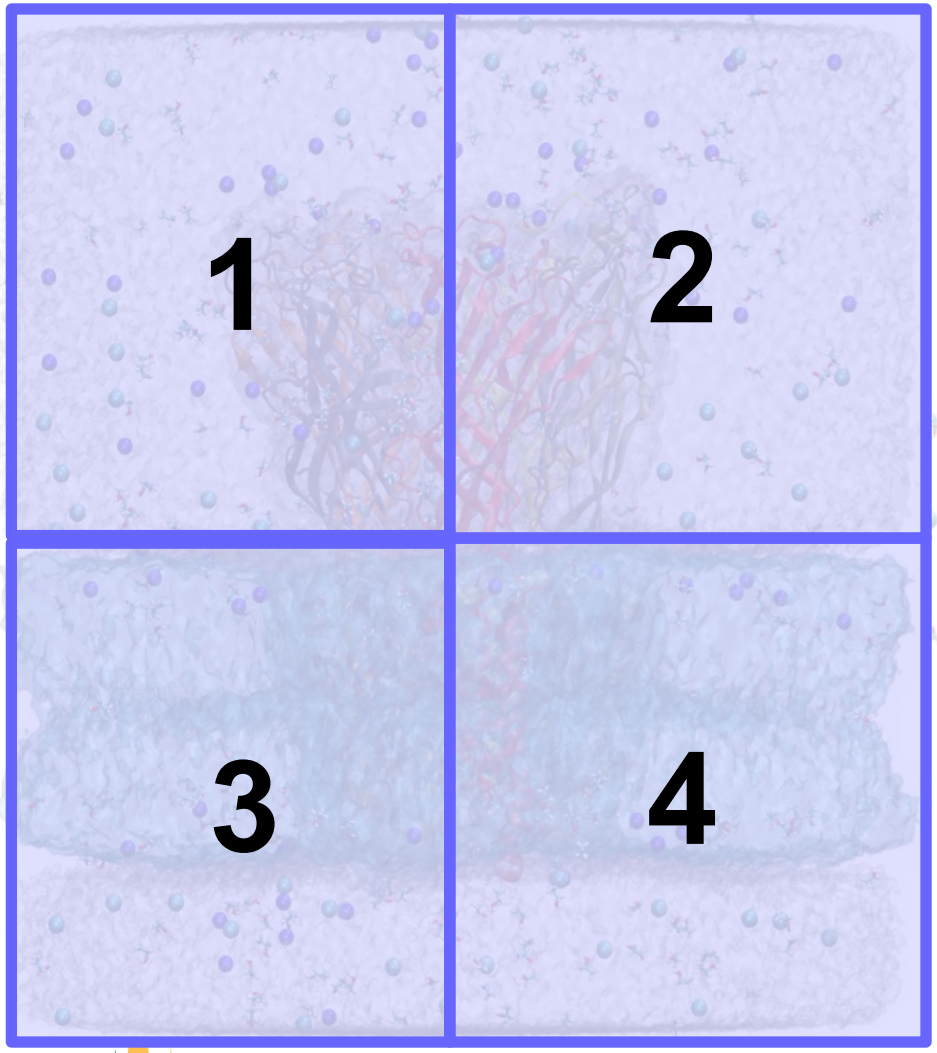
Use all power of a single node

- Use OpenMP to use all CPUs on a single node

- OpenMP is (performance) portable, but limited:
- No way to run parallel tasks next to each other
- No binding of threads to cores (cache locality)

- Need for a better threading model, requirements:
- Extremely low overhead barriers (all-all, all-1, 1-all)
- Binding of threads to cores
- Portable

Strategies to Achieve Answers in Appropriate time

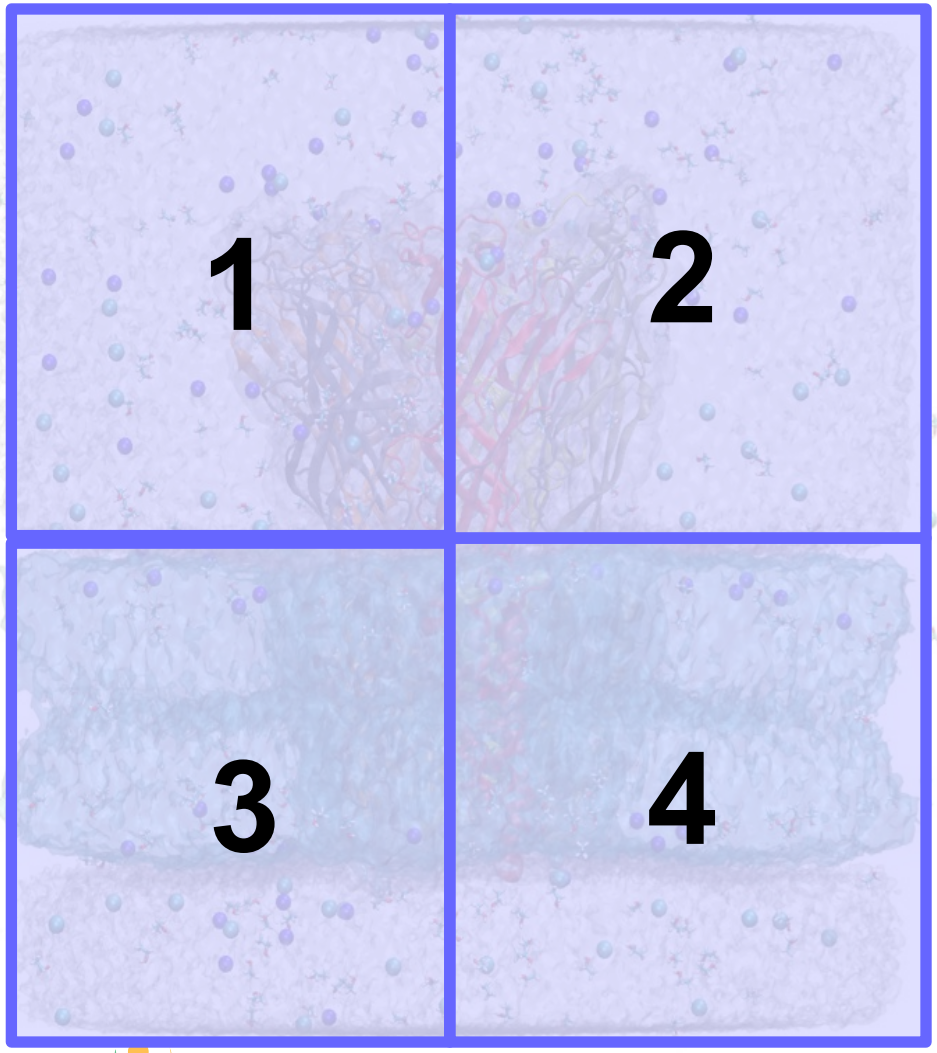


Use multiple nodes

- Use super computing infrastructure with multiple nodes and accelerators (GPUs)

- Split up the simulation system into parts
- Simulate each part on its own node

Strategies to Achieve Answers in Appropriate time



Specialise node usage for the tasks

- Use super computing infrastructure with multiple nodes and accelerators (GPUs)

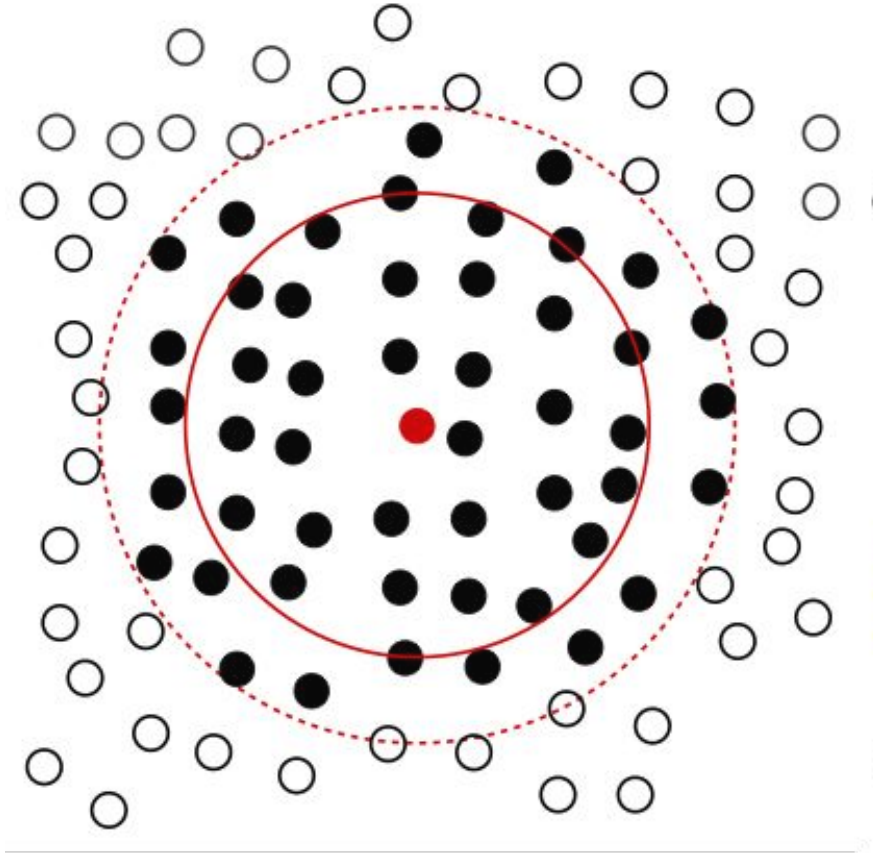
- Split up the simulation system into parts
- Simulate each part on its own node

- Set apart nodes for lange-range interactions

5

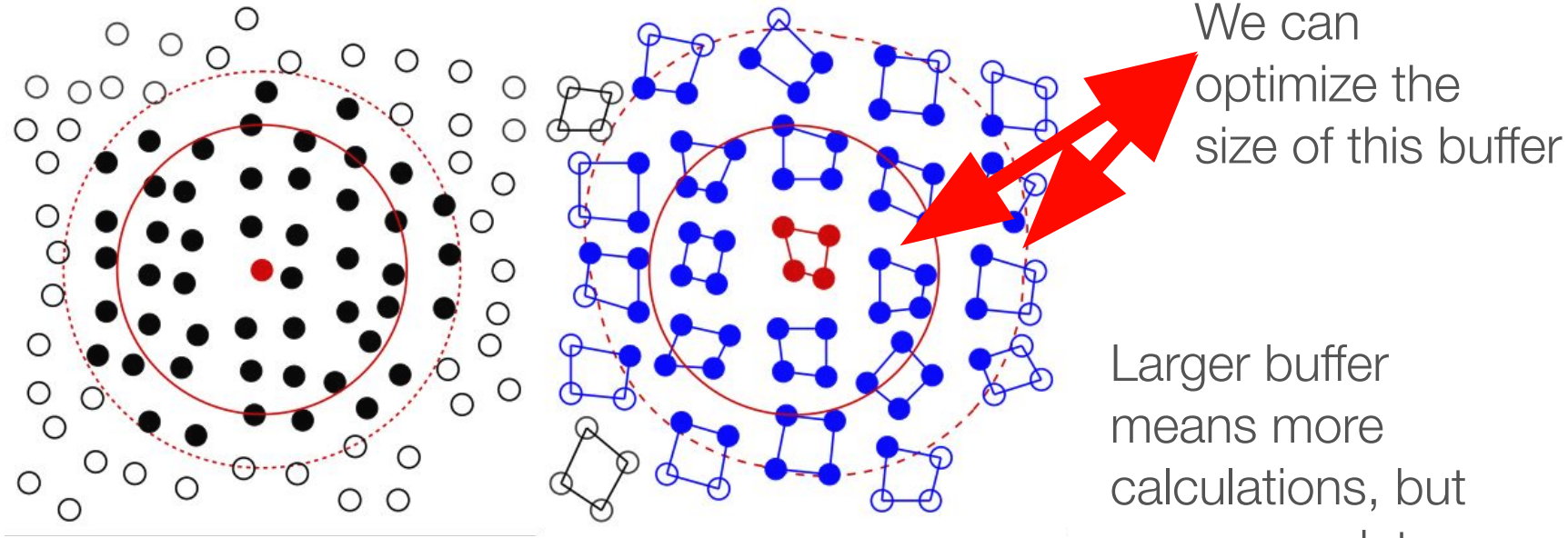
6

Challenges when dividing long-range and short-range interactions between multiple nodes



- Atoms move
- Keep track of the neighbours via a “neighbour-list”

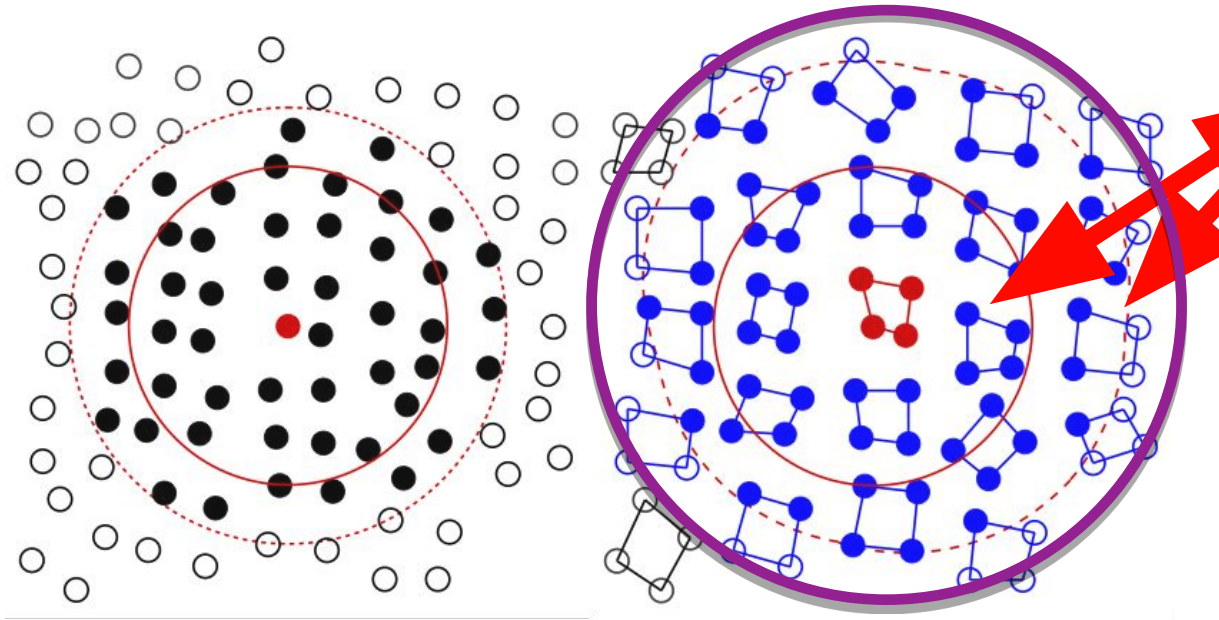
Challenges when using multiple nodes
The neighbour list



We can optimize the size of this buffer

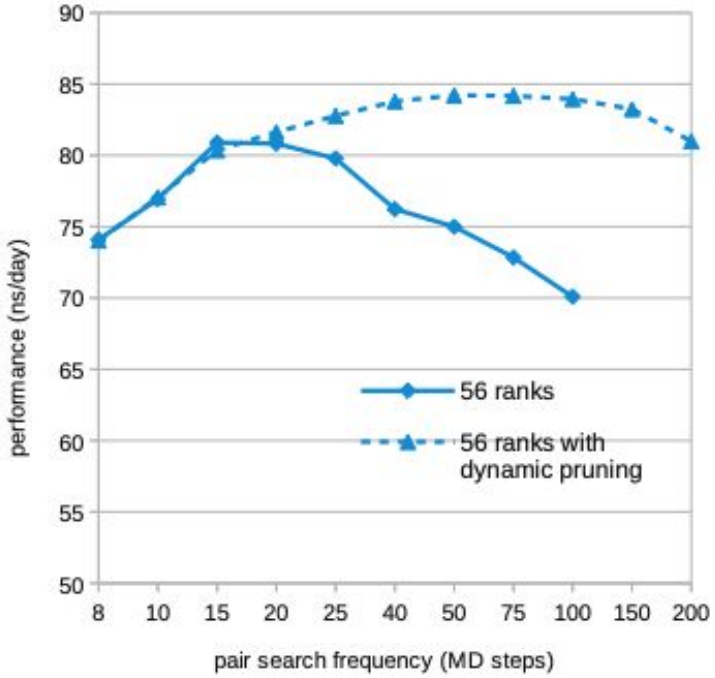
Larger buffer means more calculations, but we can update the neighbor list less frequently

Atom clustering and pair list buffering



We can optimize the size of this buffer

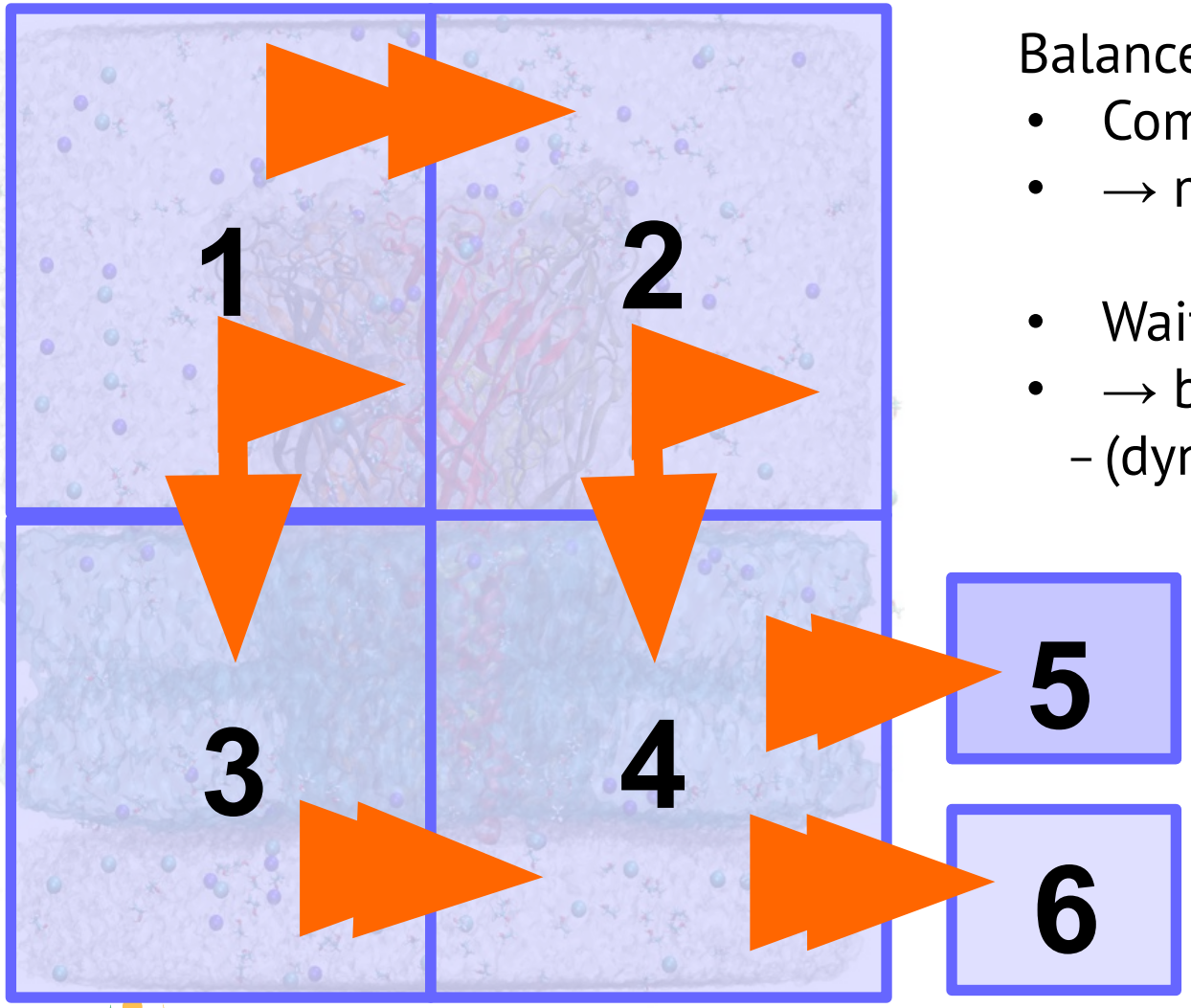
Larger buffer means more calculations, but we can update the neighbor list less frequently



New in GROMACS-2018: dual-pair list:

- reduces overhead
- less sensitive to parameters

Strategies to Achieve Answers in Appropriate time



Balance node usage

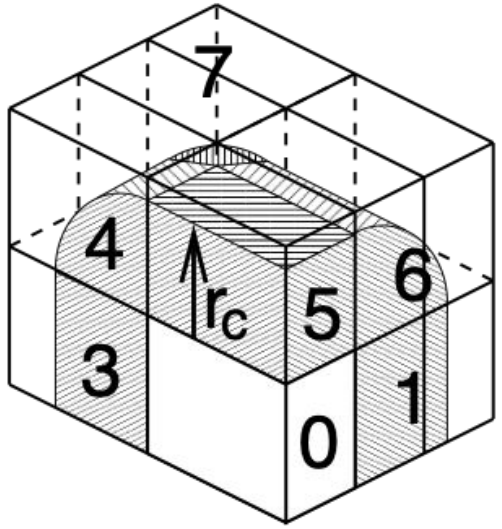
- Communication costs time
- → minimize communication

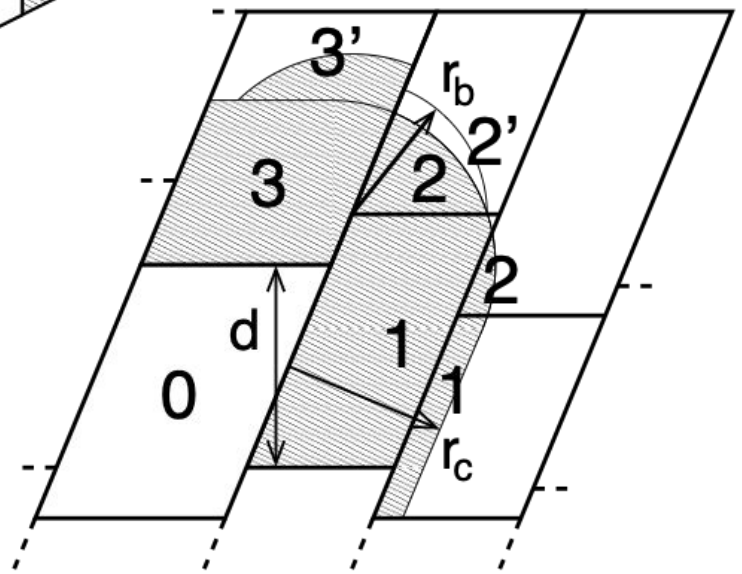
- Waiting costs time
- → balance compute time
  - (dynamic load balancing)

Spatial decomposition

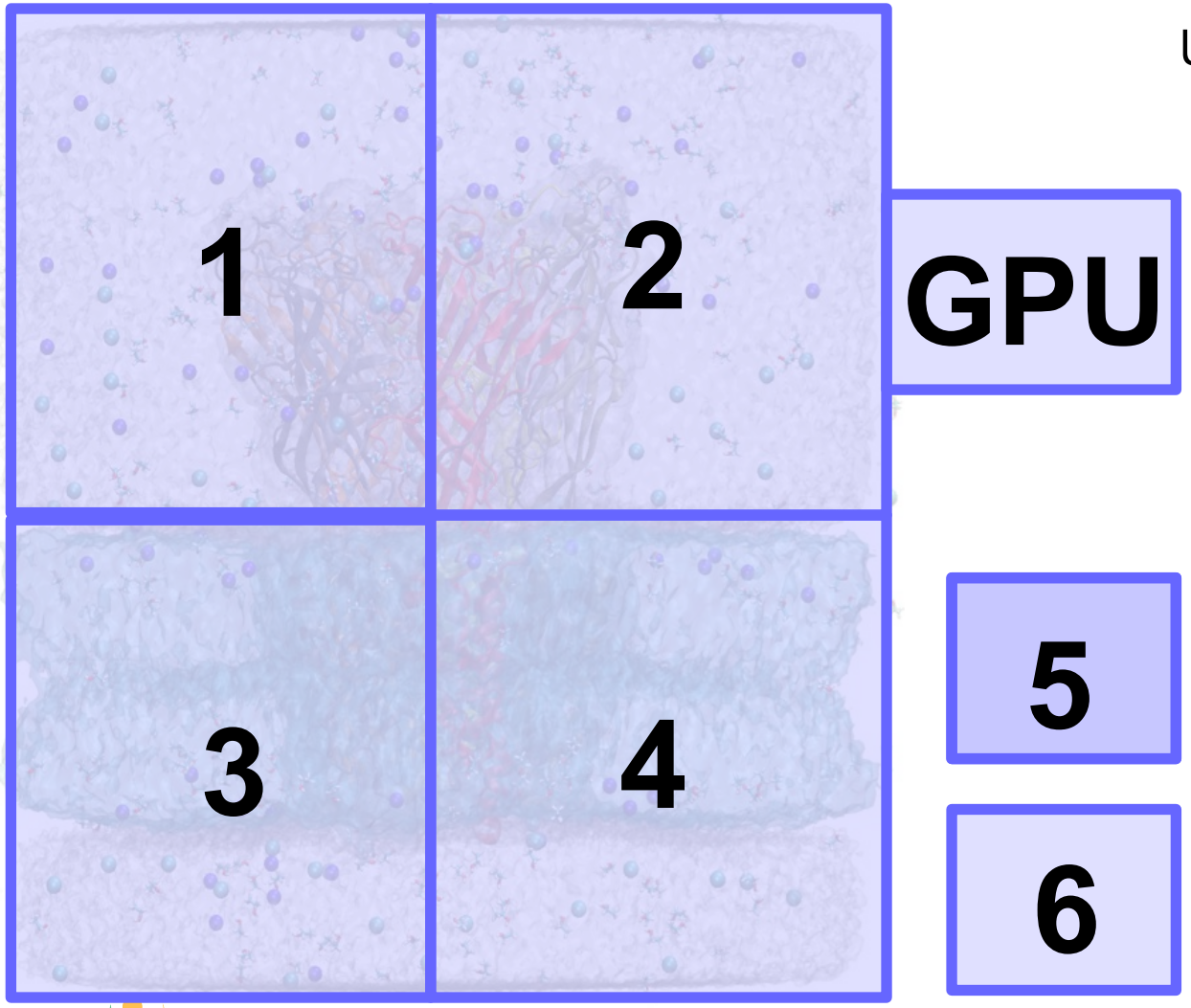
8th-shell domain decomposition:

- minimizes communicated volume
- minimizes communication calls
- implemented using aggregation of data along dimensions
- 1 MPI rank per domain
- dynamic load balancing



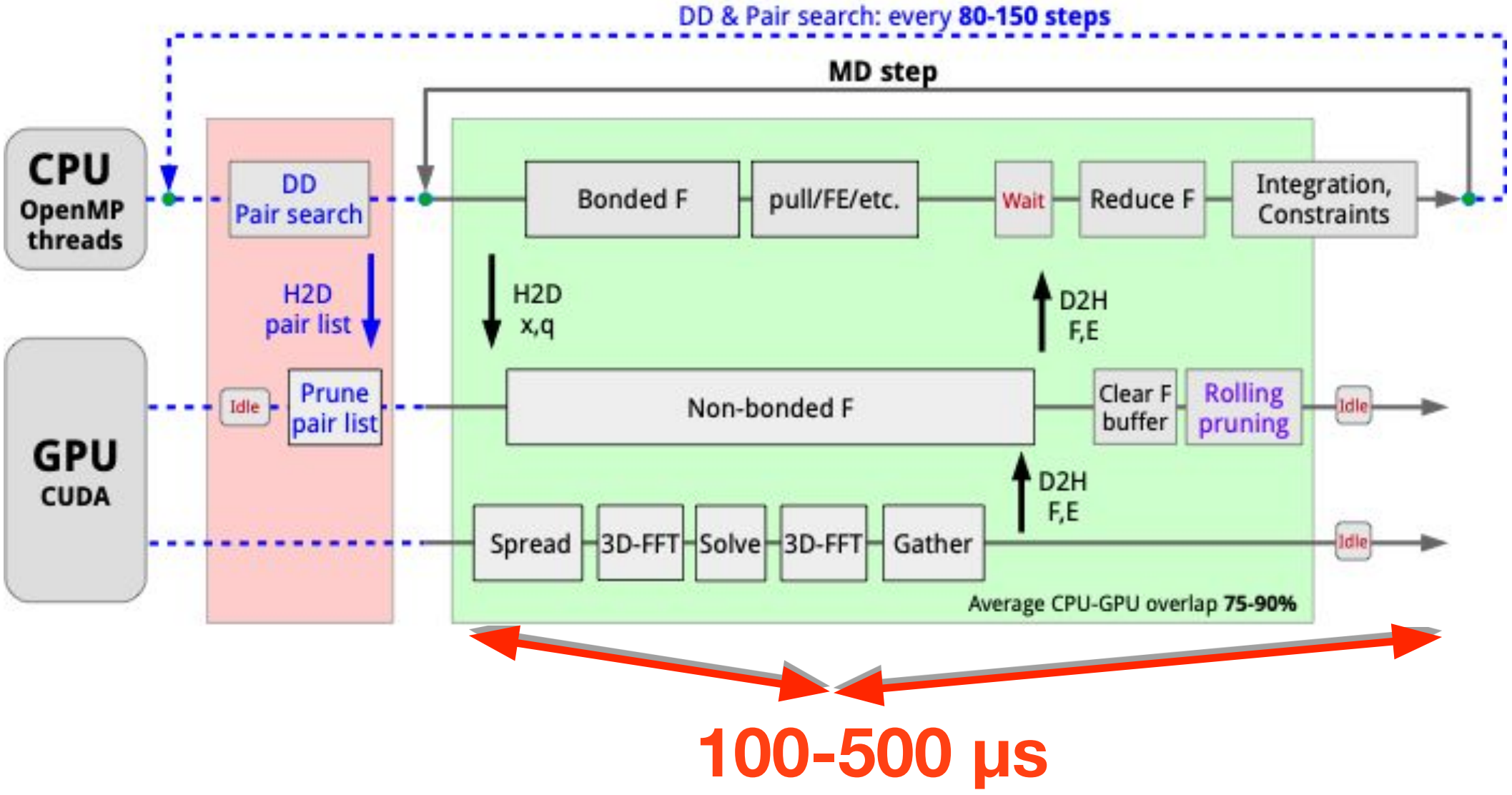


Strategies to Achieve Answers in AppropriateTime



Using accelerators (GPUs)

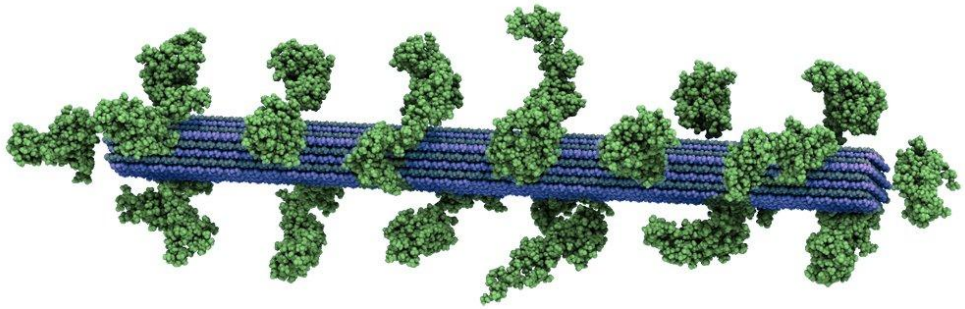
- GPUs perform a set of operations on lots of data very efficiently
- Challenges:
  - Additional communication to and from GPU
  - Not all nodes need have the same GPU setup
  - Some nodes might have more than one GPU
  - Code needs to be written in specific language

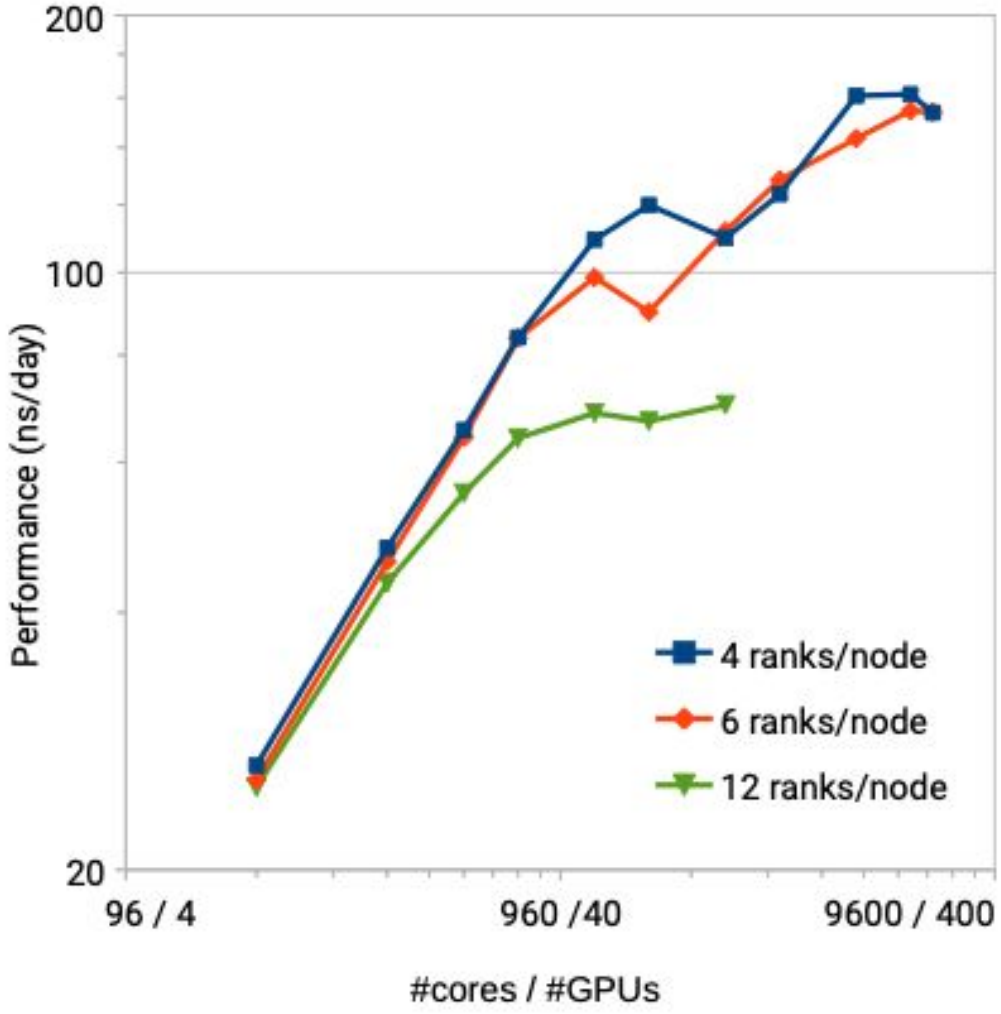

DD & Pair search: every 80-150 steps
MD step
CPU
OpenMP threads
GPU
CUDA
DD
Pair search
H2D
pair list
Idle
Prune
pair list
H2D
x,q
Bonded F
pull/FE/etc.
Wait
Reduce F
Integration,
Constraints
D2H
F,E
Non-bonded F
Clear F
buffer
Rolling
pruning
Idle
D2H
F,E
Spread
3D-FFT
Solve
3D-FFT
Gather
Idle
Average CPU-GPU overlap 75-90%
100-500 µs

GROMACS performs on many nodes

Lignocellulose beznchmark

- 3.3 million atoms
- No long-range electrostatics!
- machine: Piz Daint at CSCS
  - Cray XC50
  - NVIDIA P100 GPUs



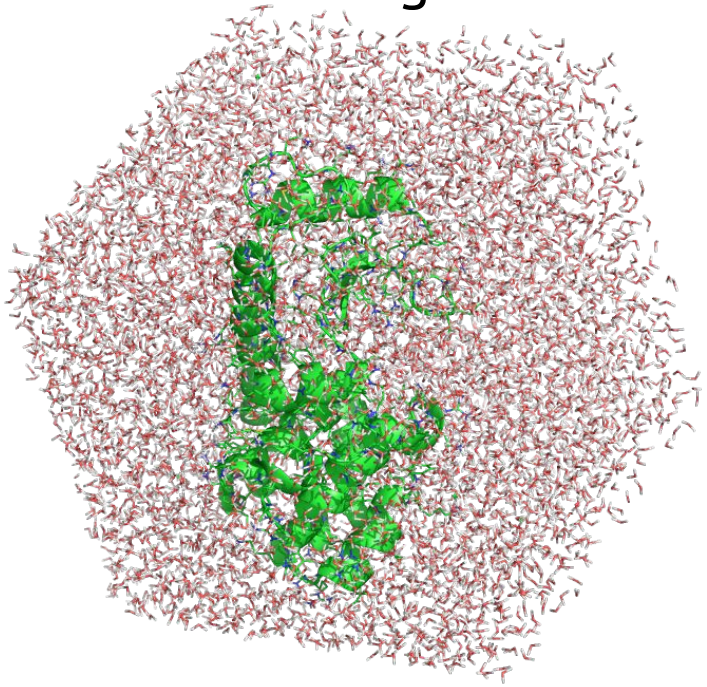


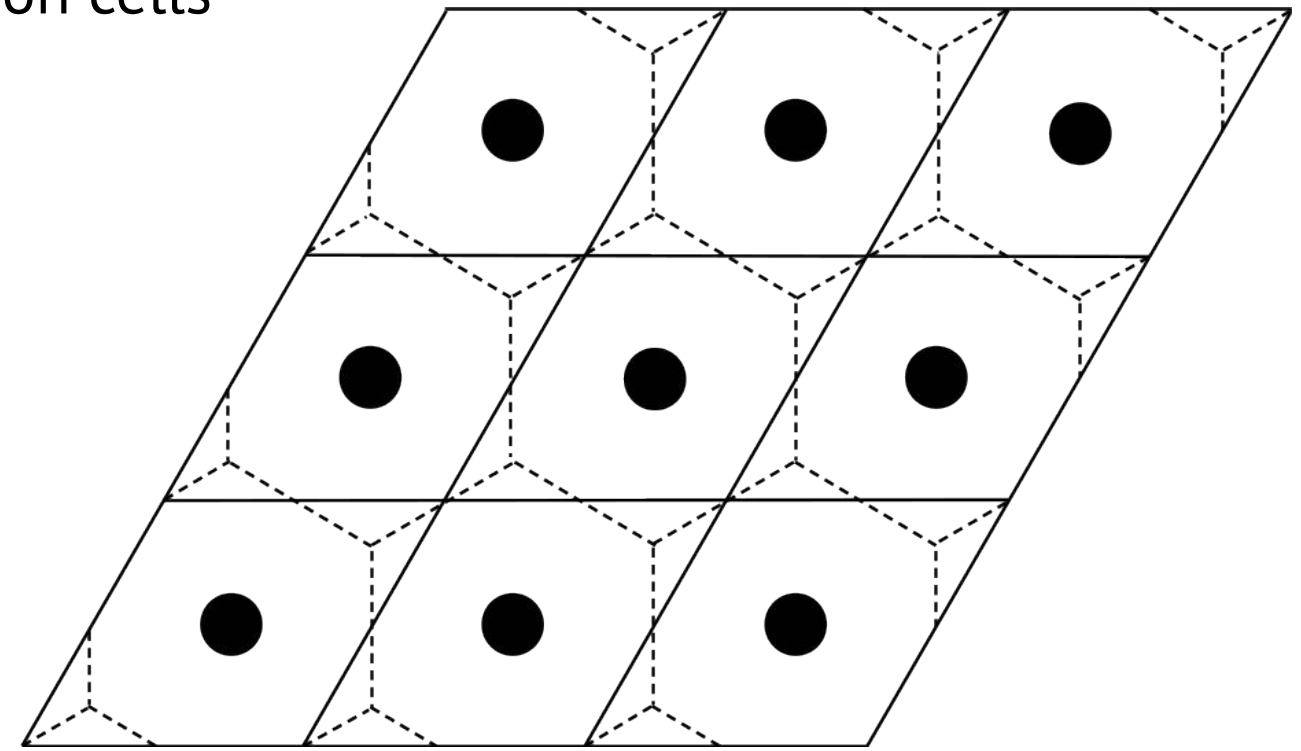
Building GROMACS well

- Build the most recent version of GROMACS
- Consult the install guide http://manual.gromacs.org/documentation/
- Use a recent (preferably latest) version of everything:
  - compiler e.g. gcc 9, clang 0
  - CUDA/OpenCL SDK, GDK and drivers
  - MPI libraries
- Configure FFTW appropriately (both $\tt --enable-sse2$ and $\tt --enable-avx$), or use ``cmake -DGMX_BUILD_OWN_FFTW=ON``
- Build with MPI only to run on multi-node clusters
- Build a non-MPI version for single-node usage, including running PME tuning, as well as pre- and post-processing
- Build in double precision only if you know why you need it

Prepare your simulation well

- Choose a simulation cell that is the right shape, just large enough for your science and your physical model
- Use non-rectangular simulation cells





Prepare your simulation well

- Prepare topologies with ``gmx pdb2gmx -vsite hydrogen`` for 4 fs time steps (and use LINCS with all-bonds constraints)
- Otherwise, use LINCS and h-bonds constraints and 2 fs time steps

Choose your simulation setup well

- Write only the output you can use
  - Less overhead during simulation
  - Faster post-processing and analysis
  - Plan to use ``gmx mdrun -rerun``
- Don't use coupling algorithms every MD step
- Choose ``nst*`` parameters to have a large common factor, like 10 or 100
- VDW cutoffs are part of your force field, follow standard practice

Acknowledgements

Thanks for input to this presentation from

- Mark Abraham, providing slides
- Szilard Pall
- Lea Rems
- Cathrine Berg

bioexcel

Centre of Excellence for Computational Biomolecular Research

BioExcel Partners 2019

Horizon 2020
European Union Funding
for Research & Innovation

BioExcel is funded by the European Union Horizon 2020 program under grant agreements 675728 and 823830.