Performance Tuning and Optimization of GROMACS

BioExcel Educational Webinar Series

Presenter: Mark Abraham
Host: Rossen Apostolov

11 May, 2016

Partners

Funding

Horizon 2020
European Union Funding
for Research & Innovation

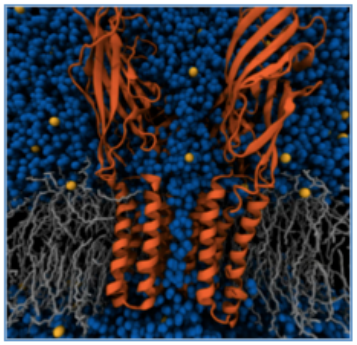
This webinar is being recorded

Objectives of BioExcel

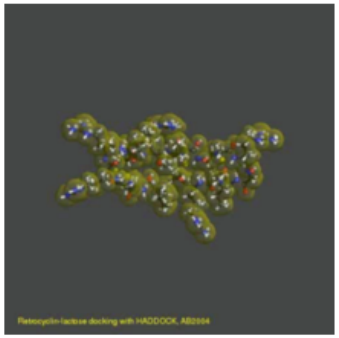
Excellence in Biomolecular Software

Improve the performance, efficiency and scalability of key codes

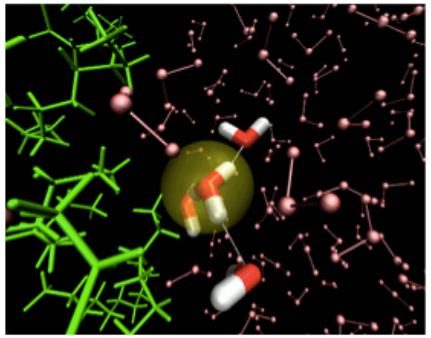
- GROMACS (Molecular Dynamics Simulations)
- HADDOCK (Integrative modeling of macro-assemblies)
- CPMD (hybrid QM/MM code for enzymatic reactions, photochemistry and electron transfer processes)



MD simulations
/GROMACS/



Docking
/HADDOCK/

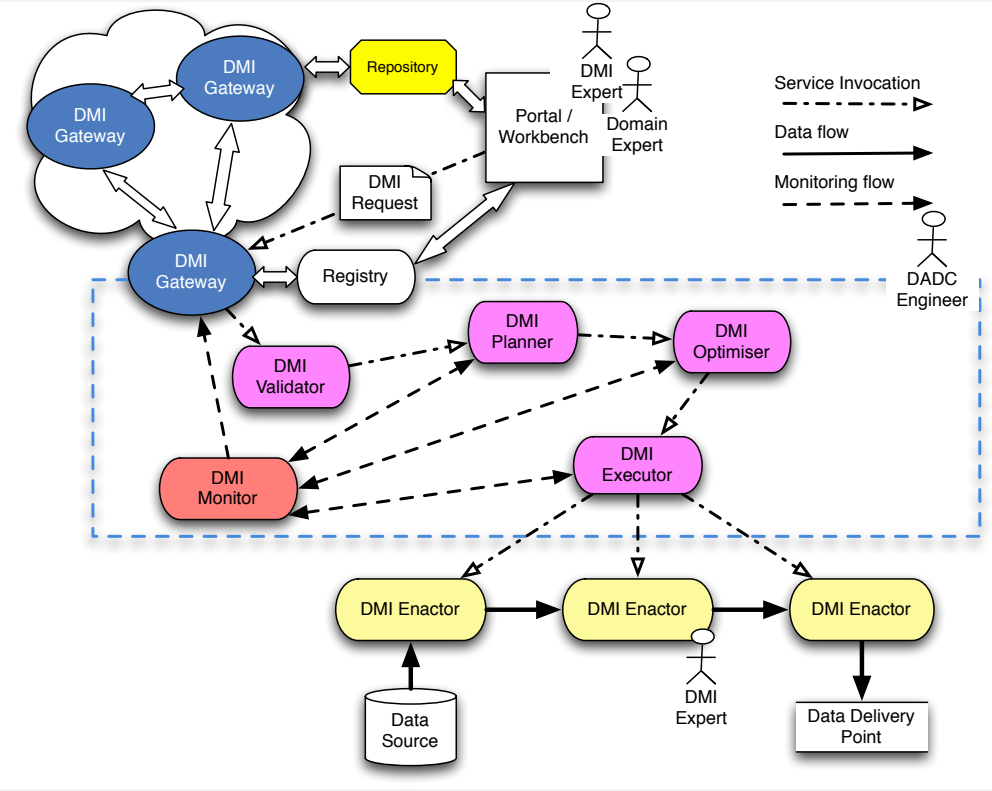


QM/MM
/CPMD/

Objectives of BioExcel

Excellence in Usability

- Make ICT technologies easier to use by biomolecular researchers, both in academia and industry
- Devise efficient workflow environments with associated data integration



Key Workflows and Platforms

Galaxy PROJECT · Open PHACTS · COMPSs · Open for Innovation KNIME · Apache Taverna

Objectives of BioExcel

Competence-building among academia and industry

Promote best practices and train end users to make best use of both software and computational infrastructure

- academic and non-profit users
- industrial users
- independent software vendors (ISVs) and academic code providers of related software
- academic and commercial resource providers

Interest Groups

- Integrative Modeling
- Free Energy Calculations
- Best practices for performance tuning
- Hybrid methods for biomolecular systems
- Biomolecular simulations entry level users
- Practical applications for industry

Support platforms

http://bioexcel.eu/contact

 Forums  Code Repositories  Chat channel  Video Channel

The presenter

Mark is the project manager for GROMACS and one of the lead developers for the package. He received his PhD from Australian National University and is an expert on disordered proteins, simulation, clustering, replica exchange sampling, parallelization, and large-scale software development. He oversees and coordinates the worldwide development efforts of GROMACS, and has been responsible for the molecular simulation application work in the CRESTA FP7 project. His current interests are focused on making efficient use of accelerators with multi-level heterogeneous and task-based parallelization.

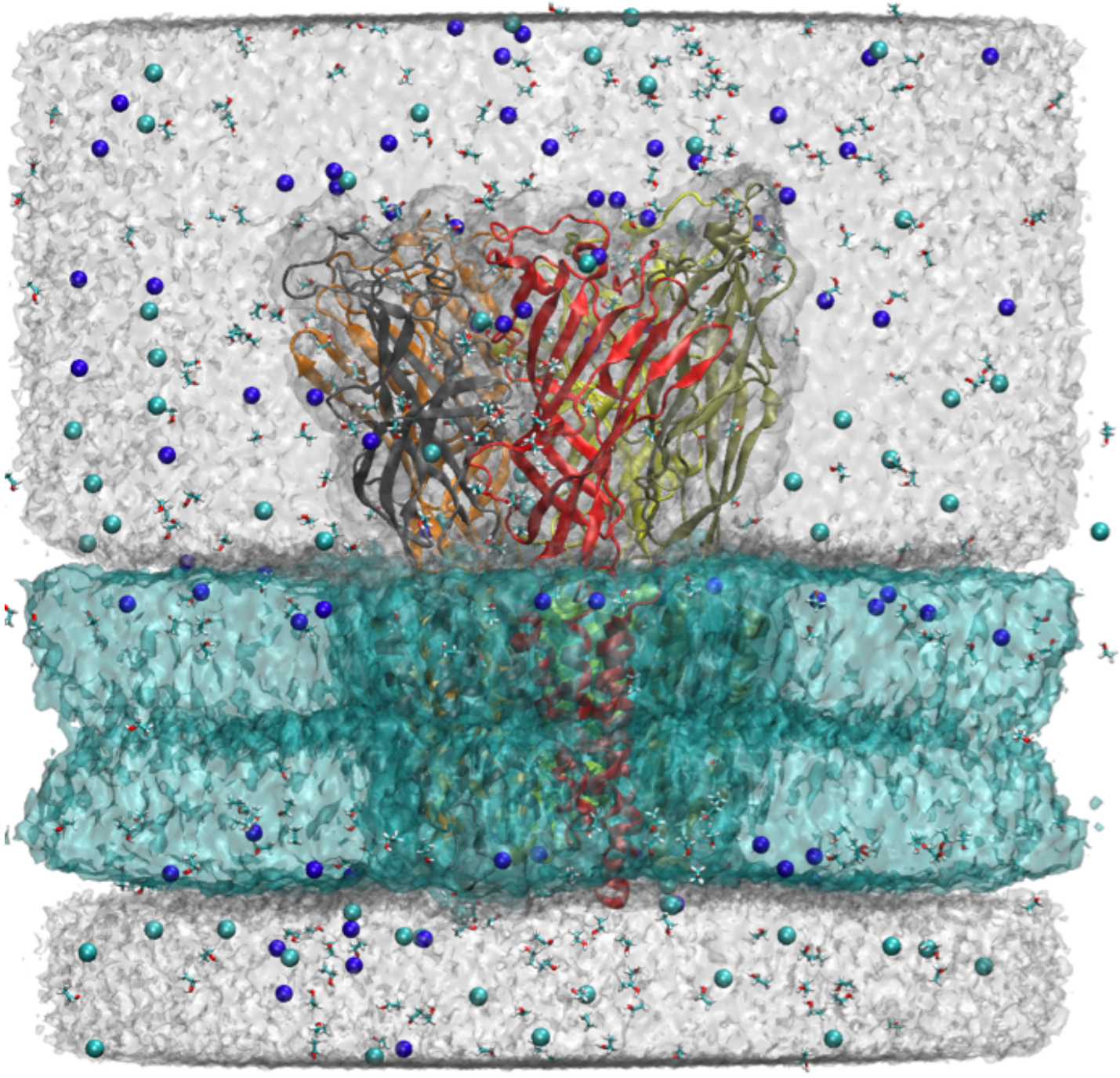
Performance tuning and optimization of GROMACS

May 11, 2016

Mark Abraham, GROMACS development manager

GROMACS

- Classical molecular dynamics
- Mostly targets problems from biochemistry
- Free and open-source C++11 community project
- Developed by multiple institutions
- Used by hundreds of research groups



Does GROMACS performance optimization matter?

- Quality of science often relates to number of independent configurations sampled
- Default performance is pretty good
- Don't bother when you are a beginner
- Don't bother if you can go and do something else while it runs
- Do bother if you are running lots of the same kind of simulation, particularly on the same kind of hardware
- Do bother if your resources cost more than your time

What do we need to consider?

- How was GROMACS built?
- What will we simulate?
- How will we simulate it?
- How does GROMACS work inside?
- What hardware will we use?
- How do we map the simulation to the hardware?
- How do we find out what might be improved?

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 4.8+ or Intel 14+
  - CUDA/OpenCL SDK, GDK and drivers
  - MPI libraries
- Configure FFTW appropriately (both $--$`enable-sse2` and $--$`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

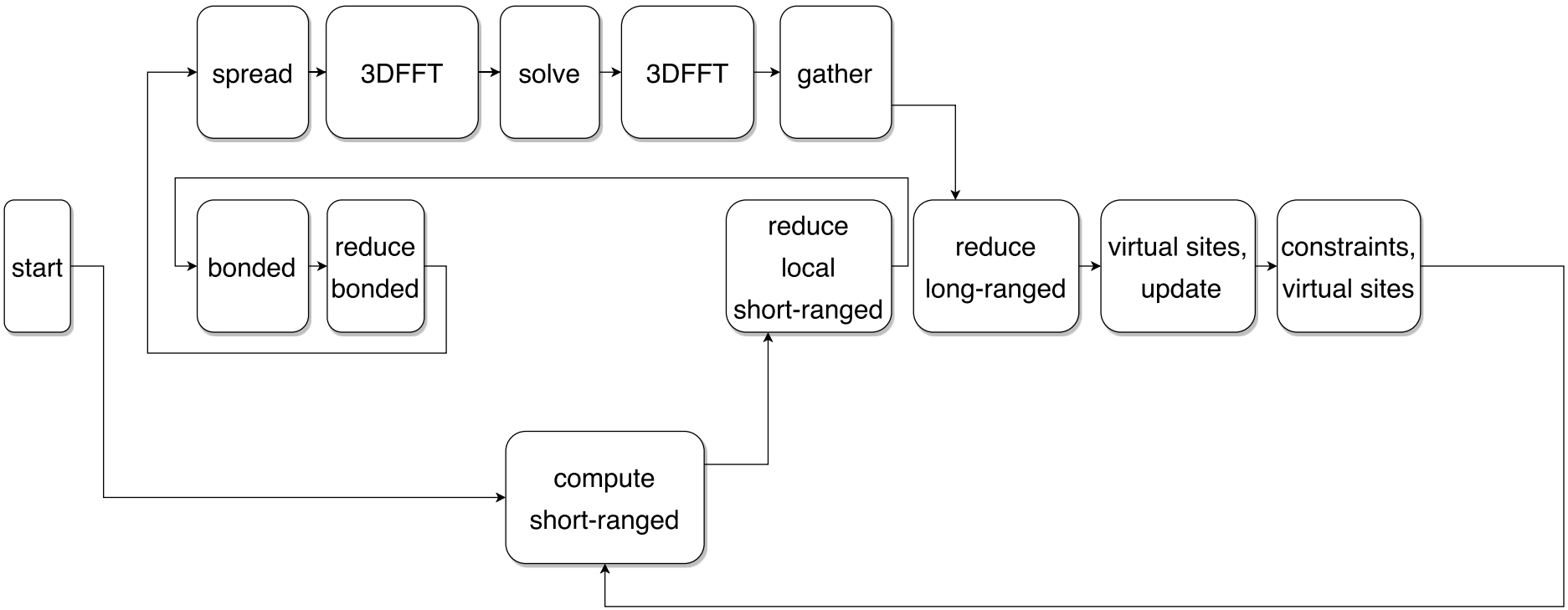
Prepare your simulation well

- Choose a simulation cell that is the right shape, just large enough for your science and your physical model
- 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
- Be aware that typical water models are rigid
- Use 3-site water models unless there's a clear scientific reason
- Orient your simulation box with load-balancing in mind

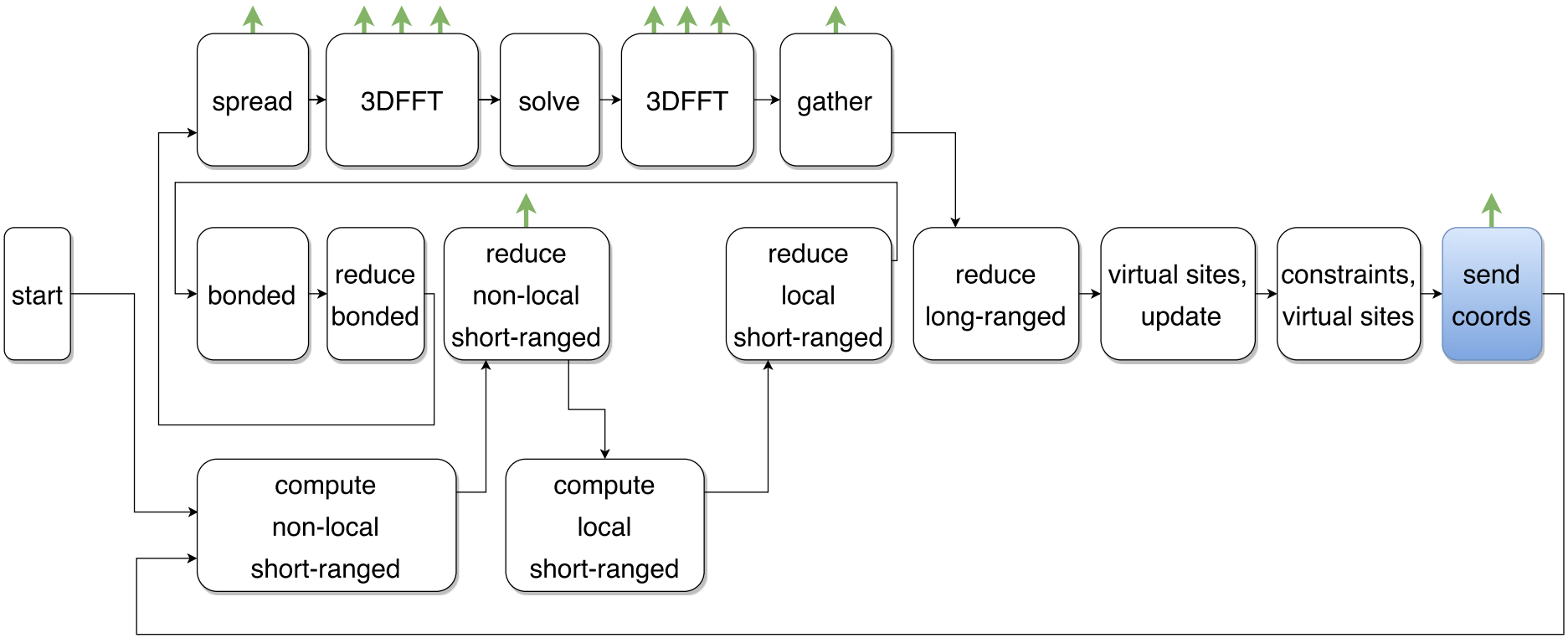
Choose your simulation setup well

- Write only the output you can use http://www.gromacs.org/Documentation/How-tos/Reducing_Trajectory_Storage_Volume
  - 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
- Use default settings for long-range PME unless you can show why you need it (but consider PME order 5 and a larger grid spacing)
- Choose appropriate LINCS settings (see http://manual.gromacs.org/documentation/5.1.2/user-guide/mdp-options.html#bonds)

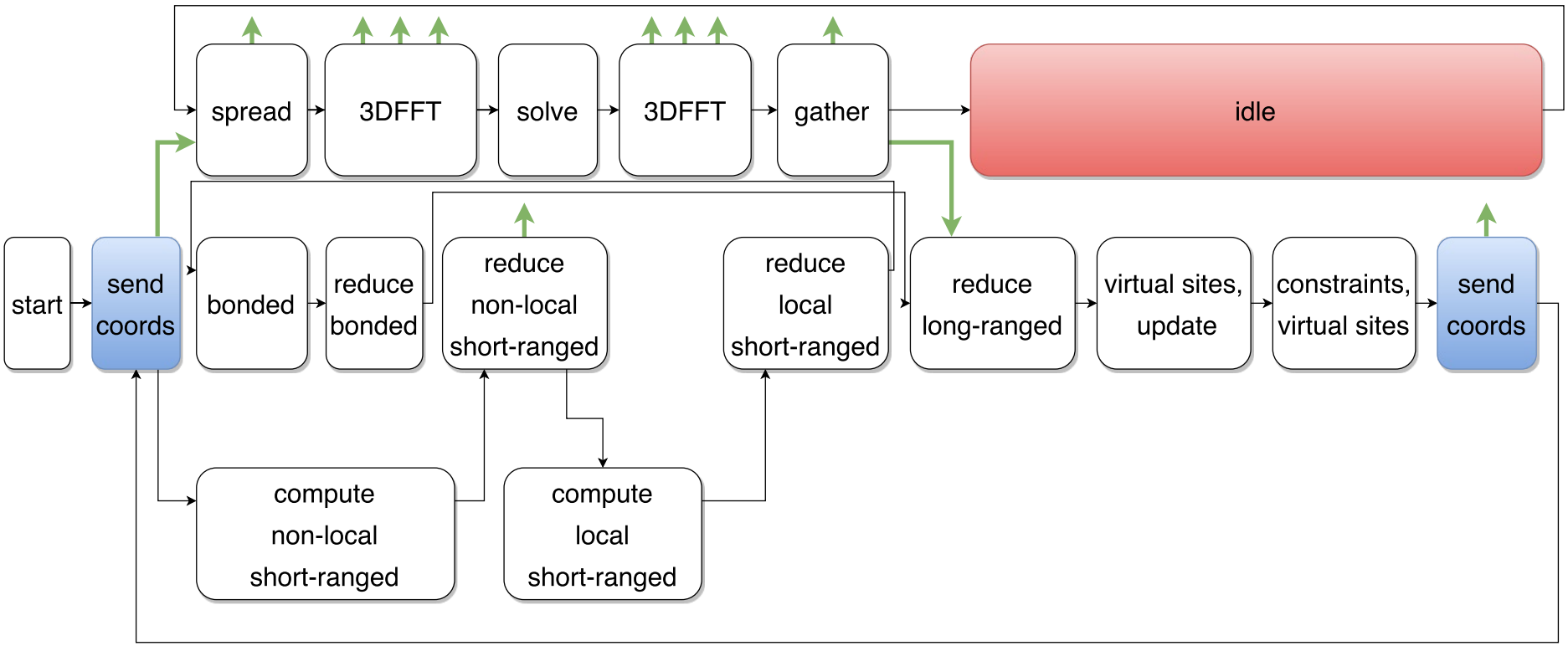
GROMACS single rank with no GPU



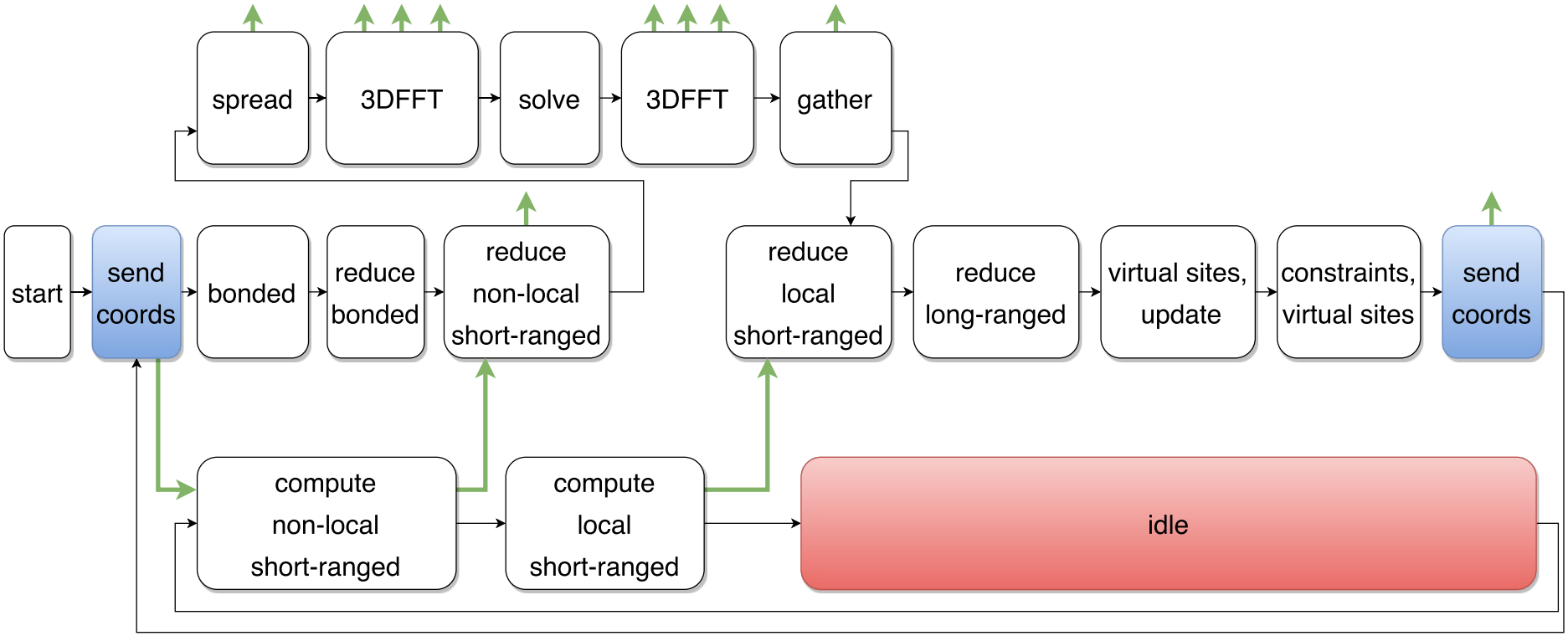
GROMACS multiple ranks, with no GPU



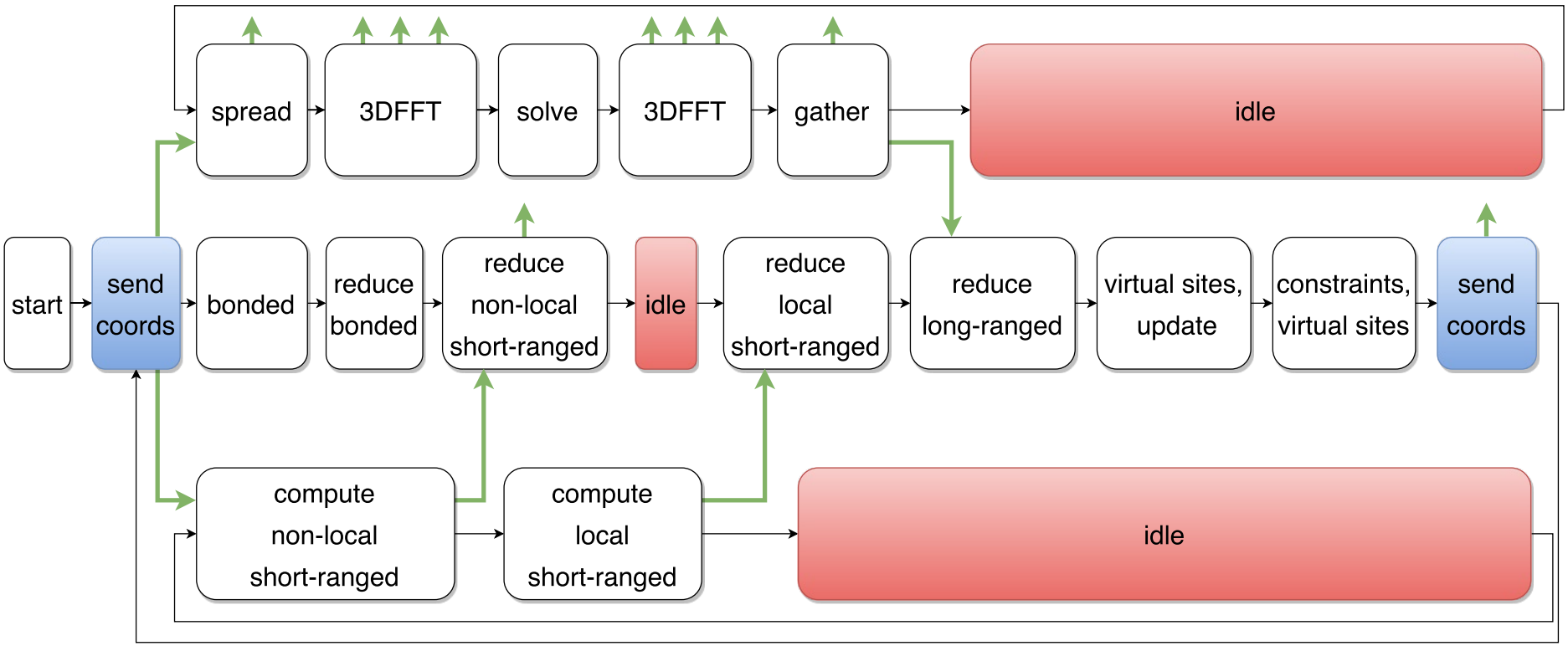
GROMACS multiple ranks, separate PME ranks, and no GPU



GROMACS multiple ranks with GPUs



GROMACS multiple ranks, separate PME ranks, and with GPUs



Get appropriate hardware

- GROMACS needs a well-balanced set of CPU and GPU resources - read https://doi.org/10.1002/jcc.24030
- To scale across multiple GPUs, you want several tens of thousands of particles per GPU
- Multi-node runs need at least gigabit ethernet, or preferably Infiniband
- Memory and disk don't matter
- Cloud resources can be fine, but avoid running inside virtual machines
- Homogeneity is much better

Running mdrun on a single CPU-only node

- Use the default build, which compiles with thread-MPI support, not MPI
- mdrun defaults do a good job
- Consider varying −ntmpi M and −ntomp N so that $M \times N$ equals the total number of threads
- Hyperthreading on Intel CPUs useful only with thousands of particles per core

On a node with 16 cores, you might try

```
gmx mdrun -ntmpi 16 -ntomp 1
gmx mdrun -ntmpi  8 -ntomp 2
gmx mdrun -ntmpi  4 -ntomp 4
```

More examples in GROMACS user guide

Running mdrun on a single GPU+CPU node

- Use the default build
- mdrun defaults do a good job of maximizing total resource usage
- You need a number of domains that's a multiple of the number of GPUs
- Consider varying −ntmpi M and −ntomp N so that $M \times N$ equals the total number of threads, and M is a multiple of the number of GPUs. You may need to set −gpu_id appropriately.
- Consider varying − nstlist L over e.g. 40/50/60/70

On a nodes with 16 cores and two GPUs, you might try

```
gmx mdrun -ntmpi 8 -ntomp 2 -gpu_id 00001111
gmx mdrun -ntmpi 4 -ntomp 4 -gpu_id 0011
gmx mdrun -ntmpi 2 -ntomp 8 -gpu_id 01
```

More examples in GROMACS user guide

Running mdrun on multi-node clusters

- Build MPI-enabled GROMACS
- GROMACS uses the network heavily - latency and variability normally limits performance and scaling
- Requesting nodes close in network space can help
- Consider tweaking MPI library settings to favour small messages requiring minimal rendezvous and buffer copy overhead
- Using separate PME MPI ranks with −npme P is essential once you run on more than a handful of nodes
  - gmx tune_pme is very useful for this
  - Now two domain decompositions and multiple communication phases are involved
  - Best when the two groups of {PP, PME} MPI ranks have sizes that are composite numbers with lots of common factors, e.g. $\{48,16\} > \{40,24\} >> \{42,22\}$

Running mdrun on CPU-only multi-node clusters

- Similar to single-node case, but you have to use mpirun gmx_mpi mdrun, and −npme now matters
- Often need to ask the job scheduler for resources and settings that match how you run mdrun
- Read the documentation for your cluster

On 4 nodes, each with 16 cores, you might try

```
mpirun -np 64 gmx_mpi mdrun -ntomp 1 -npme 16
mpirun -np 32 gmx_mpi mdrun -ntomp 2 -npme  8
mpirun -np 16 gmx_mpi mdrun -ntomp 4 -npme  4
```

More examples in GROMACS user guide

Running mdrun on GPU multi-node clusters

Similar to all the foregoing, but more complex :-(

On 4 nodes, each with 16 cores and 2 GPUs, you might try

```
mpirun -np 64 gmx_mpi mdrun -ntomp 1
                -npme 16 -gpu_id 0000000111111
mpirun -np 32 gmx_mpi mdrun -ntomp 2
                -npme  8 -gpu_id 000111
mpirun -np 16 gmx_mpi mdrun -ntomp 4
                -npme  4 -gpu_id 001
mpirun -np 24 gmx_mpi mdrun -ntomp 3
                -npme  8 -ntomp_pme 2 -gpu_id 0011
```

In the last case, each node has 4 PP ranks with 3 threads each, and 2 PME ranks with 2 threads each, total 16.

More examples in GROMACS user guide

PME tuning

- Given a number of domains (= MPI ranks), mdrun will choose whether to turn on PME tuning
  - shifts workload between PP and PME ranks at constant accuracy
  - picks whatever runs fastest and uses it for the rest of that run
  - can be turned off with −notunepme
  - can interact poorly with dynamic PP load balancing, try −dlb no
- The number and type of MPI ranks was chosen before mdrun started, which limits its options
- gmx tune_pme can run mpirun −np N gmx_mpi mdrun −npme P for fixed N and a range of P, adjusting your .tpr over a range of PME settings
- Need both MPI and non-MPI builds available
- Can work with GPUs, but there's only a small number of per-node layouts that could be reasonable

Strong scaling vs throughput

If you anyway

- need many copies of similar simulations, and
- can wait longer to get the full set of results, and
- have finite resources

... then for four simulations, each with four ranks, consider using

```
mpirun -np 16 gmx_mpi mdrun -multidir A/ B/ C/ D/
```

Multi-simulations and GPUs

- Multi-simulation is particularly efficient with GPUs
- GPUs are used only during short-ranged forces, and lie idle during calculations of constraints, virtual sites, coordinate update, and communication
- However PP ranks from two simulations per GPU can run mutually out of phase and keep the GPUs busy!

Then for four simulations, each with four ranks, on a single node with two GPUs, consider using

```
mpirun -np 16 gmx_mpi mdrun -multidir A/ B/ C/ D/
                  -gpu_id 0101010101010101
mpirun -np 16 gmx_mpi mdrun -multidir A/ B/ C/ D/
                  -gpu_id 0000000011111111
```

Measuring performance

- Use the actual production .tpr you intend to use
- Run a few thousand MD steps to permit tuning and load balancing to stabilise
- Then reset the counters to observe performance

For example

```
gmx mdrun -nsteps 6000 -resetstep 5000
```

Reading the log file

- Summary of hardware and software configuration at the start
- Reports on how the simulation has been set up just before it starts
- Analysis of walltime at the end
- Use side-by-side diff to compare different runs to understand where they were diffferent and what effect that had

Audience Q&A session

Please use the **Questions** function in GoToWebinar

NEXT WEBINAR

"Atomistic Molecular Dynamics Setup with MDWeb"

with Adam Hospital

25 May 2016
16:00-17:00 CEST

www.bioexcel.eu/contact