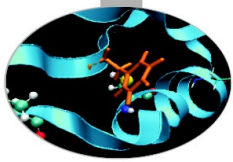
CINECA SCAI
SuperComputing Applications and Innovation

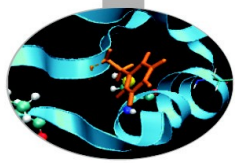


Running MD on HPC architectures I. Clusters and Hybrid Clusters

Alessandro Grottesi
Cineca

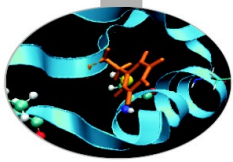
PRACE CINECA

Outlook



- System architecture of hybrid clusters @CINECA
- How to explore and interact with the software installed on the system
- Running MD simulation exploiting the computing resources available at CINECA
- Performance optimizzation and benchmarks

PLX characteristics



Model: IBM iDataPlex DX360M3

Architecture: Linux Infiniband Cluster

Processor Type:

- Intel Xeon (Esa-Core Westmere) E5645 2.4 GHz (Compute)
- Intel Xeon (Quad-Core Nehalem) E5530 2.66 GHz (Service and Login)

Number of nodes: 274 Compute + 1 Login + 1 Service + 8 Fat + 6 RVN + 8 Storage + 2 Management

Number of cores: 3288 (Compute)

Number of GPUs: 548 nVIDIA Tesla M2070 + 20 nVIDIA Tesla M2070Q

RAM: 14 TB (48 GB/Compute node + 128GB/Fat node)

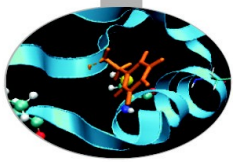
PLX system performance

Peak performance: **32 Tflops (3288 cores at 2.40GHz clock frequency)**

Peak performance: **565 TFlops SP or 283 TFlops DP (548 Nvidia M2070)**

CINECA SCAI
SuperComputing Applications and Innovation

Eurora characteristics



Model: Eurora prototype

Architecture: Linux Infiniband Cluster

Processors Type:

- Intel Xeon (Eight-Core SandyBridge) E5-2658 2.10 GHz (Compute)
- Intel Xeon (Eight-Core SandyBridge) E5-2687W 3.10 GHz (Compute)
- Intel Xeon (Esa-Core Westmere) E5645 2.4 GHz (Login)

Number of nodes: 64 Compute + 1 Login

Number of cores: 1024 (compute) + 12 (login)

Number of accelerators: 64 nVIDIA Tesla K20 (Kepler) + 64 Intel Xeon Phi (MIC)

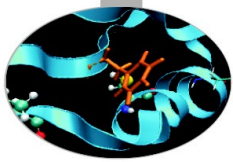
RAM: 1.1 TB (16 GB/Compute node + 32GB/Fat node)

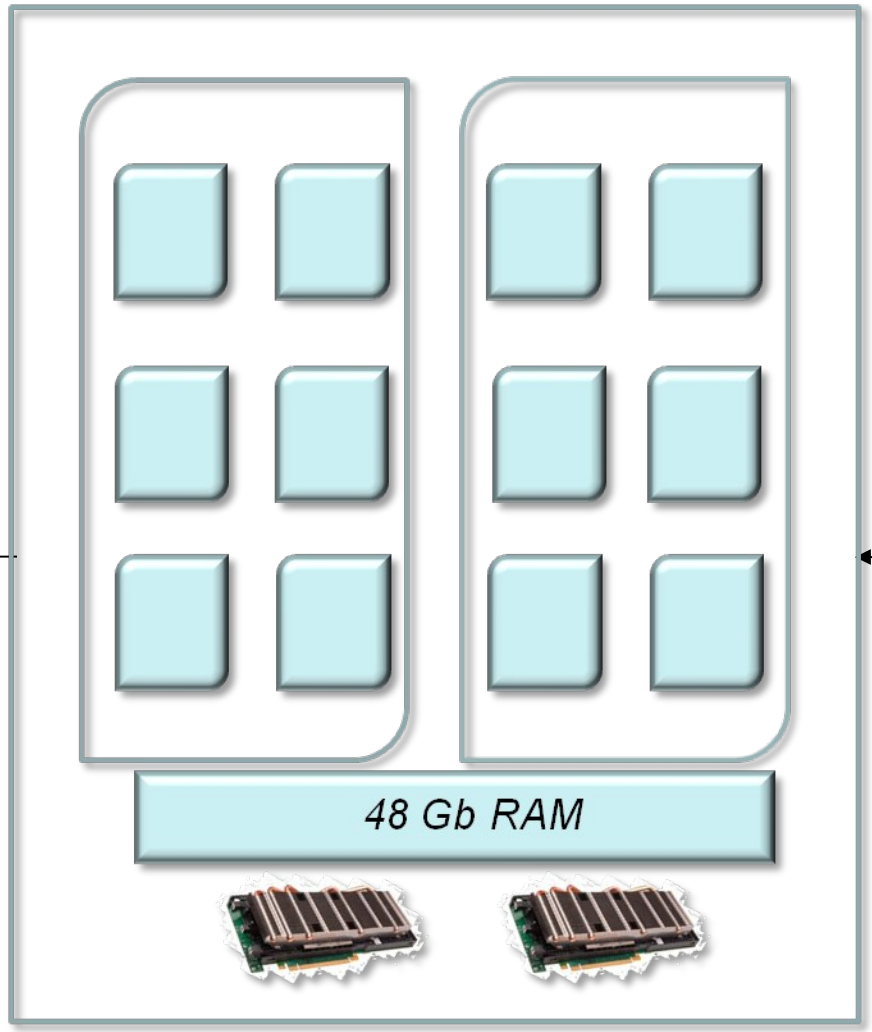
OS: RedHat CentOS release 6.3, 64 bit

PRACE

CINECA

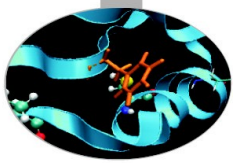
Compute nodes



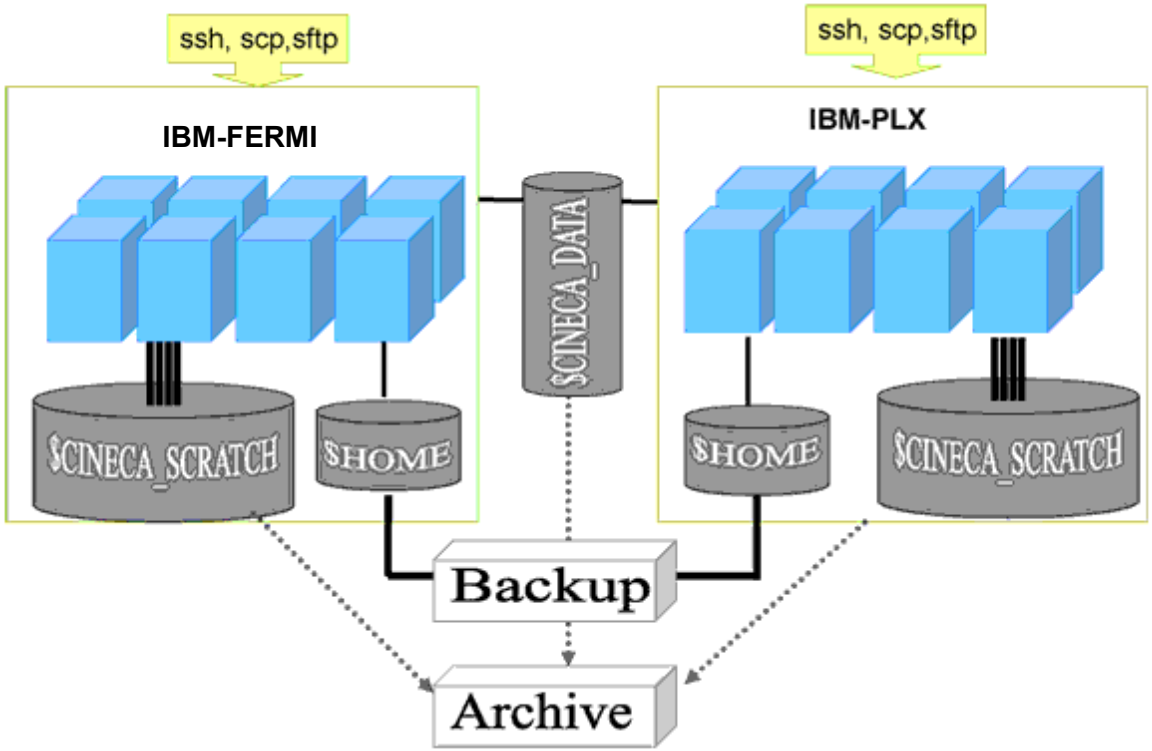


Infiniband connection

Disks and filesystems

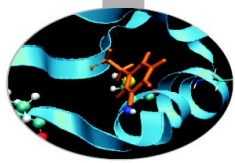


Standard CINECA environment



please use “cindata” command to get info on your disk occupation

Work Environment



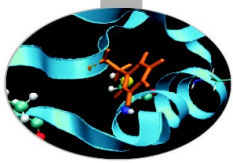
$HOME:

- Permanent, backed-up,and local to PLX.
- Quota = 5GB.
- For source code or important input files.

$CINECA_SCRATCH:

- Large, parallel filesystem (GPFS).
- Temporary (files older than 30 days automatically deleted), no backup.
- No quota. Run your simulations and calculations here.

Accounting: saldo



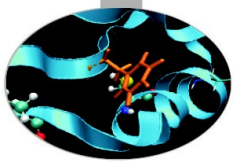
```
[mcestari@node342] (~)
$ saldo -b

--------------------------------------------------------------------------------------------------
account              start         end          total        localCluster   totConsumed     totConsumed
                                                (local h)    Consumed(local h)   (local h)          %
--------------------------------------------------------------------------------------------------
try11_test        20110301    20111201         10000                   0            2           0.0
cin_staff         20110323    20200323     200000000               64581      6689593           3.3
ArpaP_prod        20130130    20131101       1500000                   0            0           0.0
```

Accounting philosophy is based on the resources requested for the time of the batch job:

cost = no. of cores **requested** x job duration

In the CINECA system it is possible to have more than 1 budget (“account”) from which you can use time. The accounts available to your UNIX username can be found from the `saldo` command.

Modules

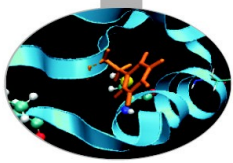


- CINECA' s work environment is organized in modules, a set of installed libs, tools and applications available for all users.

- "loading" a module means that a series of (useful) shell environment variables wil be set

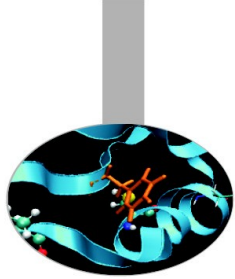
- E.g. after a module is loaded, an environment variable of the form "<MODULENAME>_HOME" is set

module, my best friend



- all the optional software on the system is made available through the **"module" system**
  - provides a way to rationalize software and its env variables
- modules are divided in 3 *profiles*
  - **profile/base** (stable and tested modules)
  - **profile/engineering** (contains specific software for engineering simulations)
  - **profile/advanced** (software not yet tested or not well optimized)
- each profile is divided in 4 categories
  - **compilers** (Intel, GNU, Portland)
  - **libraries** (e.g. LAPACK, BLAS, FFTW, ...)
  - **tools** (e.g. Scalasca, GNU make, VNC, ...)
  - **applications** (software for chemistry, physics, ...)

Module commands



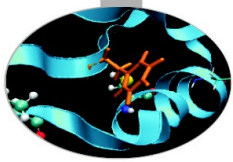
> module available (or just “> module av”)
Shows the full list of the modules available in the profile you’re into, divided by: environment, libraries, compilers, tools, applications

> module (un)load <module_name>
(Un)loads a specific module

> module show <module_name>
Shows the environment variables set by a specific module

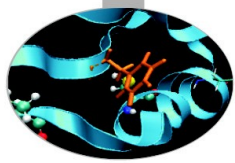
> module help <module_name>
Gets all informations about how to use a specific module

> module purge
Gets rid of all the loaded modules



Running MD code on PLX/Eurora

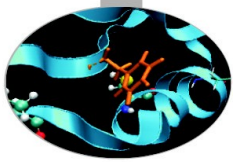
GROMACS: what for...?



1. Minimizzation
2. Molecular Dynamics (classic, brownian, Langevin)
3. Normal Mode Analysis
4. Essential Dynamics and Sampling
5. Free Energy calculations (FEP, Umbrella sampling, AFM)
6. Replica Exchange Molecular Dynamics
7. Coarse-Grained MD
8. Metadynamics
9. Much more...

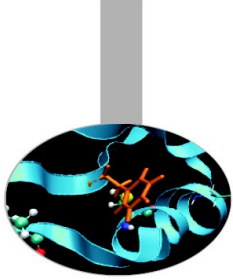
CINECA

Available forcefield in Gromacs (4.6.5)



1. AMBER03 protein, nucleic AMBER94 (Duan et al., J. Comp. Chem. 24, 1999-2012, 2003)
2. AMBER94 force field (Cornell et al., JACS 117, 5179-5197, 1995)
3. AMBER96 protein, nucleic AMBER94 (Kollman et al., Acc. Chem. Res. 29, 461-469, 1996)
4. AMBER99 protein, nucleic AMBER94 (Wang et al., J. Comp. Chem. 21, 1049-1074, 2000)
5. AMBER99SB protein, nucleic AMBER94 (Hornak et al., Proteins 65, 712-725, 2006)
6. AMBER99SB-ILDN protein, nucleic AMBER94 (Lindorff-Larsen et al., Proteins 78, 1950-58, 2010)
7. AMBERGS force field (Garcia & Sanbonmatsu, PNAS 99, 2782-2787, 2002)
8. CHARMM27 all-atom force field (with CMAP) - version 2.0
9. GROMOS96 43a1 force field
10. GROMOS96 43a2 force field (improved alkane dihedrals)
11. GROMOS96 45a3 force field (Schuler JCC 2001 22 1205)
12. GROMOS96 53a5 force field (JCC 2004 vol 25 pag 1656)
13. GROMOS96 53a6 force field (JCC 2004 vol 25 pag 1656)
14. GROMOS96 54a7 force field (Eur. Biophys. J. (2011), 40,, 843-856)
15. OPLS-AA/L all-atom force field (2001 aminoacid dihedrals)
16. [DEPRECATED] Encad all-atom force field, using full solvent charges
17. [DEPRECATED] Encad all-atom force field, using scaled-down vacuum charges
18. [DEPRECATED] Gromacs force field (see manual)
19. [DEPRECATED] Gromacs force field with hydrogens for NMR

Workflow for running MD simulations in GROMACS



Generate a topology — pdb2gmx

Generate simulation box — editconf

Solvate the system — genbox

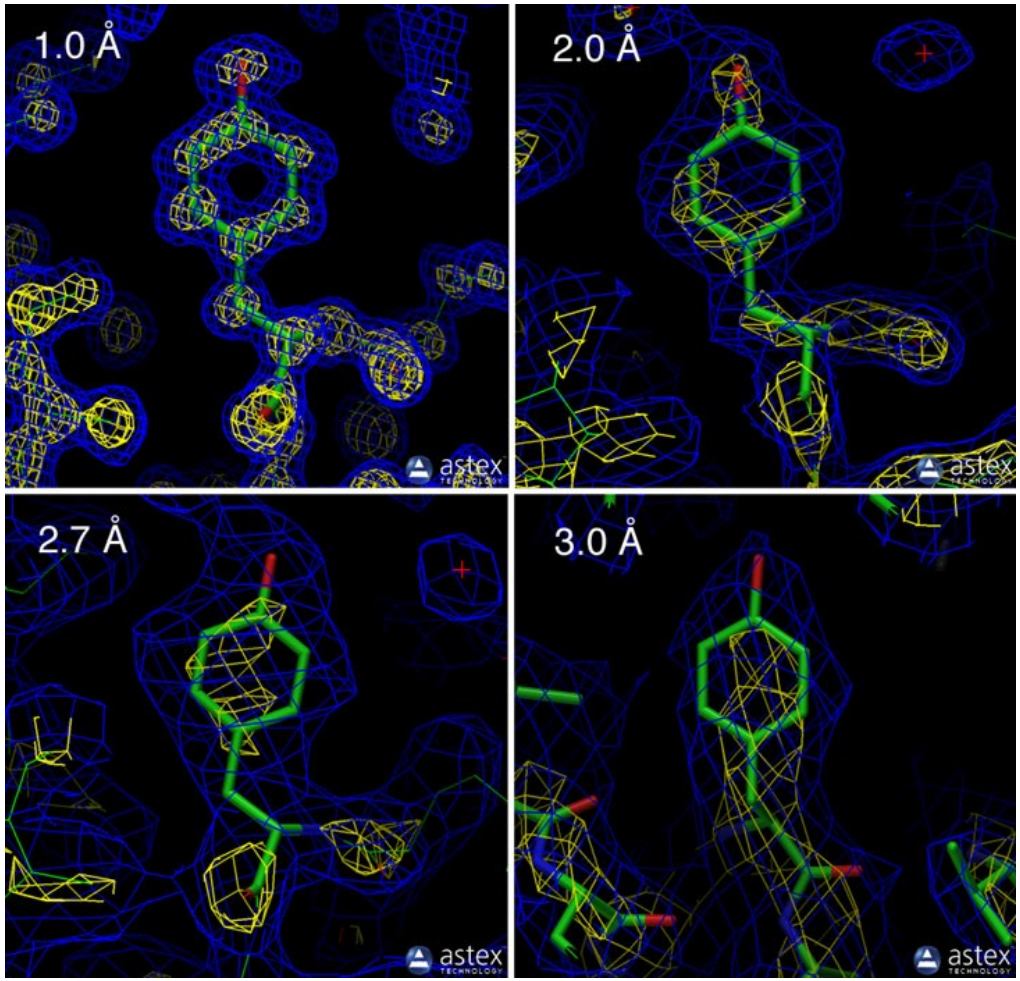
Generate input file for mdrun — grompp

topol.tpr

Actually run the simulation — mdrun

Analysis of trajectory files — trjconv, g_rms, g_covar, g_anaeig, g_energy, g_rmsf, g_rdf

Initial coordinates: X-Ray vs. NMR

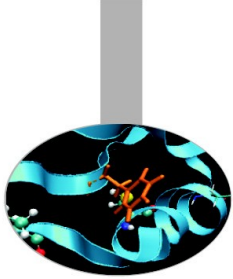


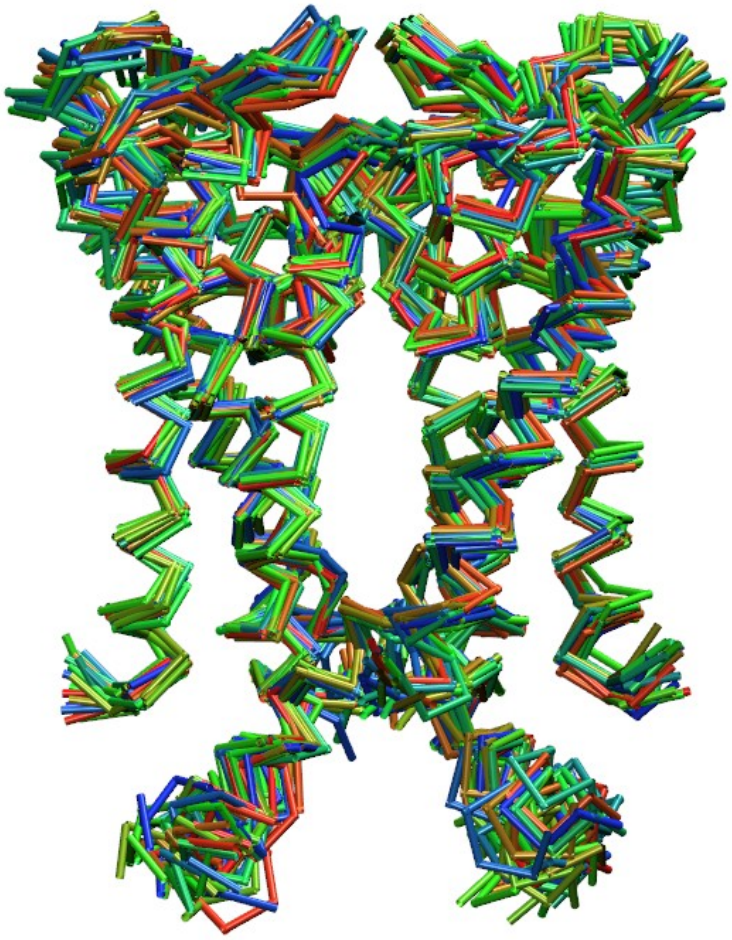
Higher X-ray resolution allows to use a more reliable starting structure in terms of amino-acids stereo-chemistry and accuracy of atomic positions

Error on initial position of protein atoms determines local structural alterations of the protein structure

X-ray resolutions smaller than 2 Å are much more reliable, although difficult to achieve. Generally, a resolution in the range 2 < R < 3 Å are acceptable. Beyond 3 Å the uncertainty of the initial position may cause artefacts in the MD simulation

Initial coordinates: X-Ray vs. NMR



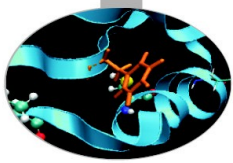


KcsA Potassium channel

(PDB code: 2K1E)

NMR determined structure provide information in a more realistic physiological environment as compared to X-ray determined structures although this could result in lower quality of initial coordinates and incertainties in the position of atomic coordinates.

Generate topology: pdb2gmx



To convert a structure pdb file into a Gromacs topology:

```
pdb2gmx –f input_file.pdb –ignh –ter
```

input:

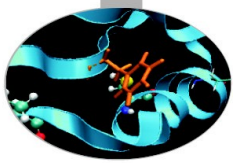
1. file_in.pdb — initial set of coordinates (either pdb or gro format)

output:

1. topol.top — system topology
2. posre.itp — position restraints file
3. conf.gro — coordinate file (gro format by default)
4. topolA.itp, topolB.itp, etc — topology of chain A, B, etc...
5. posreA.itp, posreB.itp, etc — position restraints file for chain A, B, etc...

CINECA

pdb2gmx: interactive options



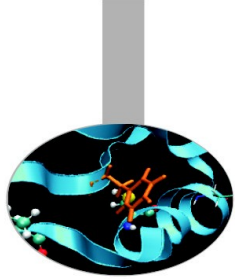
Proteins extremities (N- and C-terminus) have to be treated with particular care as they are usually charged at neutral pH (7.2/7.3). However, in most cases, protein sequences in the PDB databank are composed of a sub-set of the actual primary structure and therefore extremities are likely to be neutral.
To set up ionizzation state for N- and C-terminus in proteins:

pdb2gmx –f input.pdb –ignh –ter

Select N-terminus type (start)
0: NH3+
1: NH2
2: None

Select C-terminus type (end)
0: COO-
1: COOH
2: None

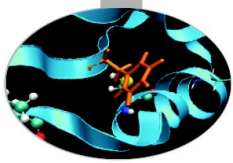
pdb2gmx: pH and net charge



By default, pdb2gmx assumes you want to simulate your protein in a neutral pH environment (ph=7). Hence the net charge carried by ionizzable residues is the default at that pH. Otherwise you need to set up net charge according to the following table:

pH	Lysine	Arginine	Glutamate	Aspartate
7.0	+1	+1	-1	-1
< 5.0	+1	+1	0	0
> 9.0	0	0	-1	-1

How to set-up non default pH



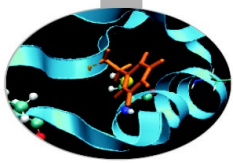
In Gromacs, a usefull way to simulate a biological macromolecule at pH ≠ 7 consists of using the flags -lys -arg -asp -glu to set up interactivelly the net charge on lysine, arginine, apartate and glutamate residue, respectivelly. By these flag we can indeed change the default ionizzation state on single residue in the protein sequence.

pdb2gmx –f input_file.pdb –ignh –ter –lys –arg –asp –glu

Processing chain 1 'A' (803 atoms, 100 residues)
Which LYSINE type do you want for residue 33
0. Not protonated (charge 0) (LYS)
1. Protonated (charge +1) (LYSH)

Processing chain 1 'A' (803 atoms, 100 residues)
Which GLUTAMIC ACID type do you want for
residue 13
0. Not protonated (charge -1) (GLU)
1. Protonated (charge 0) (GLUH)

Histidines



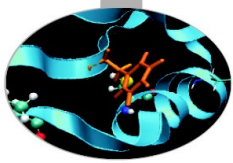
Histidine residues play an important rule in protein function as they are often located within catalytic pockets and binding sites or could be involved in interactions with prosthetic group (HEME) or could binf metal ions for important enzymatic activities (cytochromes, clorophylls, etc.)

In Gromacs, four types of histidine residue are available for use with special case, differing for it ionizzation state:

pdb2gmx –f input.pdb –ignh –ter –his

1. HISA (neutral): hydrogen atom on Nδ1
2. HISB (neutral): hydrogen atom on Nε2
3. HISH (net charge = +1): hydrogen atom on Nδ1 and Nε2
4. HIS1: histidine bound to HEME

How to generate the box: editconf



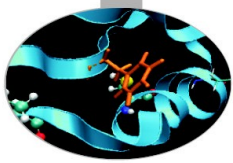
Structure generated file has to be immersed in a box of water molecules (or alternative solvent) prior to run an MD simulation. Different types of box are available in Gromacs (triclinic, cubic, dodecahedron or octahedron) and can be generated by the command:

editconf –f conf.gro -bt triclinic –d 0.8 –o output.gro

Box type: triclinic in this case

Minimal solute-solvent distance along box axes.

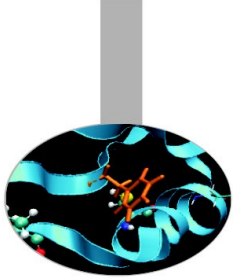
Box solvation: genbox



Once defined, box has to be physically soaked with water (or alternative solvent). This can be easly performed by running the command:

```
genbox –cp conf.gro –cs spc216.gro –o out.gro
```

Ionic strength: genion



Grid based electrostatic treatment (Ewald sums, PME, etc.) are better performed with system net charge = 0. Namely, make sure that:

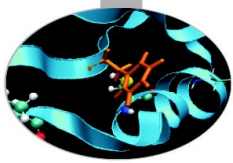
solute charge + solvent charge = 0

To set up box neutrality we can replace as many water molecule with corresponding positive or negative ions to generate a total charge = 0. To do so, we can run the genion command as follows:

```
genion –s topol.tpr –random –seed XXX –o oution.gro
–nn 20 –np 10 -p topol.top
```

This command replace randomly a total of 30 water molecules with 20 negative ions (chloride) and 10 positive ions (sodium) and updates the topol.top file with the new list of atoms.

grompp:
the GROMACS preprocessor



Command grompp generates a binary input file with all structural info and forcefield parameters neeed to run an MD simulation.

grompp –f param.mdp –c coord.gro –n index.ndx –p topol.top –o topol.tpr

Grompp output is a binary file called topol.tpr that can be used as input for running the calculation. To visualize and check all info stored in the topol.tpr file we can use the following command:

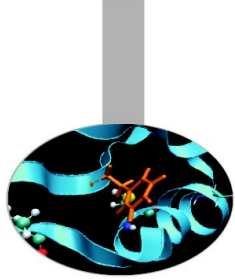
gmxdump –s topol.tpr

Output control

Van der Waals and electrostatics

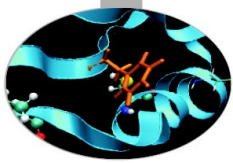
Temperature and pressure coupling

```
title                    = Yo
cpp                      = cpp
Include                  = -I../top
cefine                         = -DPOSRES
integrator               = md
dt                       = 0.002
nsteps                   = 500000
nstxout                  = 5000
nstvout                  = 5000
nstlog                         = 5000
nstenergy                = 250
nstxtcou                 = 250
xtc_grps                 = Protein
energygrps               = Protein  SOL
nstlist                  = 10
ns_type                  = grid
rlist                    = 0.8
coulombtype              = PME
rcoulomb                 = 1.4
rvdw                     = 0.8
tcoupl                         = V-Rescale
tc-grps                  = Protein      SOL
tau_t                    = 0.1  0.1
ref_t                    = 300  300
pcoupl                   =
tau_p                    = 1.0
compressibility          = 4.5e-5
ref_p                    = 1.0
gen_vel                  = yes
gen_temp                 = 300
gen_seed                 = 173529
constraints              = all-bonds
```



CINECA

Output control parameters

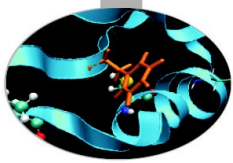


nsteps	= 500000	Total number of steps
nstxout	= 5000	coords output frequency for traj.trr
nstvout	= 5000	velocity output frequency for traj.trr
nstlog	= 5000	output frequency for log file (md.log)
nstenergy	= 250	output frequency for energy file ener.edr
nstxtcout	= 250	coords output frequency for traj.xtc
xtc_grps	= Protein	content of file traj.xtc
energygrps	= Protein SOL	energy groups to store in file ener.edr

File traj.xtc contains coordinates of our simulated system. Atomic coordinates are saved in a compressed format so that to reduce file size. This file is the main trajectory file used for simulation analysis.

File traj.trr contains atomic coordinates, velocities and forces of our simulated system. These data are saved as 4 digits floating point numbers and are usefull to recover coordinates and velocities after a job crash or if we need velocities and forces for special analyses.

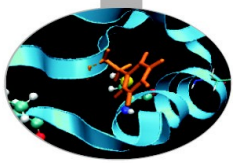
Electrostatics control



```
; Method for doing electrostatics
coulombtype             = PME
rcoulomb-switch         = 0
rcoulomb                = 1.2
; Relative dielectric constant for the medium and the reaction field
epsilon_r               = 1
epsilon_rf              = 1
; Method for doing Van der Waals
vdw-type                = Cut-off
; cut-off lengths
rvdw-switch             = 0
rvdw                    = 1.2
; Spacing for the PME/PPPM FFT grid
fourierspacing          = 0.150
; FFT grid size, when a value is 0 fourierspacing will be used
fourier_nx              = 0
fourier_ny              = 0
fourier_nz              = 0
; EWALD/PME/PPPM parameters
pme_order               = 4
ewald_rtol              = 1e-05
ewald_geometry          = 3d
epsilon_surface         = 0
optimize_fft            = no
```

CINECA

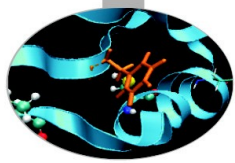
Parameter for temperature and pressure coupling



```
; Temperature coupling
Tcoupl                  = V-rescale
; Groups to couple separately
tc-grps                 = System
; Time constant (ps) and reference temperature (K)
tau_t                   = 0.1
ref_t                   = 250.0
; Pressure coupling
Pcoupl                  = Parrinello-Rahman
Pcoupltype              = isotropic
; Time constant (ps), compressibility (1/bar) and reference P (bar)
tau_p                   = 0.5
compressibility         = 4.5e-5
ref_p                   = 1.0
; Scaling of reference coordinates, No, All or COM
refcoord_scaling        = No
; Random seed for Andersen thermostat
andersen_seed           = 815131
```

weak temperature coupling

Run the simulation: mdrun



mdrun –s topol.tpr –dd dx dy dz -pd -npme N

domain decomposition

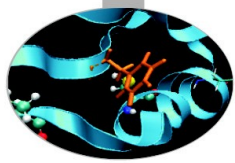
particle decomposition

The command generates many outpuf filed at the end of the job. Among them:

1. confout.gro — final coordinates file (gro format)
2. traj.xtc — simulation trajectory file (compressed)
3. traj.trr — simulation trajectory file (coord+velocity, high prec.)
4. ener.edr — energy file
5. state.cpt — chekpoint file for restarting runs.
6. md.log — log file with output control

CINECA

What if it all crashes...?



A .cpt file is produced by mdrun at specified intervals (mdrun -cpt n, where n is the interval in minutes), and contains information on all the state variables in a simulated system. In the case of a crash (hardware failure, power outage, etc), a checkpoint file can be used to resume the simulations exactly as it was before the failure. Simulations can also be extended using a checkpoint file (www.gromacs.org).

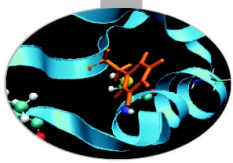
mdrun -s topol.tpr -cpi state.cpt

mdrun -s topol.tpr -cpi state.cpt -append

Write down coordinates on previous generated files

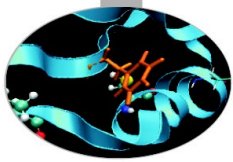
CINECA

Gromacs 4.6.5, pure MPI on PLX



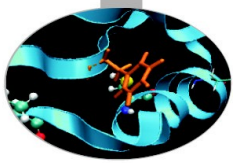
```
>module load autoload gromacs/4.6.5
>module help gromacs/4.6.5

#!/bin/bash
#PBS -N gmx
#PBS -l select=1:ncpus=12:mpiprocs=12:mem=47GB
#PBS -q <queue>
#PBS -l walltime=1:00:00
#PBS -A <account_nr>

cd $PBS_O_WORKDIR                    ==> change to current dir

module load profile/advanced
module load autoload gromacs/4.6.5

export OMP_NUM_THREADS=1             ==> set nr. Of OpenMP threads per MPI proc to 1
#                                    ==> set total mpi tasks = 12 and bind to two GPUs
mdrun=$(which mdrun_mpi)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -nb cpu"
mpirun -np 12 $cmd
```

Gromacs 4.6.5, pure MPI on PLX



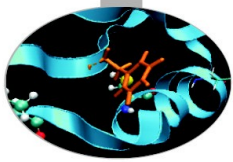
```
File  Edit  View  Search  Terminal  Help
top - 10:05:33 up 64 days, 17:01,  0 users,  load average: 9.62, 3.47, 2.01
Tasks: 263 total,  13 running, 250 sleeping,   0 stopped,   0 zombie
Cpu(s): 99.5%us,  0.4%sy,  0.0%ni,  0.2%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  49431720k total,  2186064k used, 47245656k free,   422140k buffers
Swap:  8193140k total,        0k used,  8193140k free,   942864k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5025 agrottes  25   0  148m  14m 8596 R 100.2  0.0   1:33.41 mdrun_mpi
 5026 agrottes  25   0  148m  13m 7260 R 100.2  0.0   1:33.12 mdrun_mpi
 5027 agrottes  25   0  148m  14m 8360 R 100.2  0.0   1:33.61 mdrun_mpi
 5030 agrottes  25   0  148m  12m 6956 R 100.2  0.0   1:33.63 mdrun_mpi
 5031 agrottes  25   0  148m  14m 8336 R 100.2  0.0   1:33.36 mdrun_mpi
 5024 agrottes  25   0  148m  14m 8996 R 99.8  0.0   1:33.49 mdrun_mpi
 5028 agrottes  25   0  148m  13m 7284 R 99.8  0.0   1:33.50 mdrun_mpi
 5032 agrottes  25   0  148m  13m 7144 R 99.8  0.0   1:33.55 mdrun_mpi
 5033 agrottes  25   0  148m  13m 7132 R 99.8  0.0   1:33.53 mdrun_mpi
 5034 agrottes  25   0  148m  12m 6948 R 99.8  0.0   1:33.61 mdrun_mpi
 5029 agrottes  25   0  148m  13m 7052 R 99.5  0.0   1:33.10 mdrun_mpi
 5023 agrottes  25   0  165m  18m  11m R 97.8  0.0   1:32.62 mdrun_mpi
 5039 root      15   0 62052 1176  612 S  0.3  0.0   0:23.14 sshd
 5759 agrottes  15   0 12908 1252  828 R  0.3  0.0   0:00.02 top
    1 root      15   0 10372  696  584 S  0.0  0.0   0:03.16 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.91 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.04 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.91 migration/1
    6 root      34  19     0    0    0 S  0.0  0.0   0:00.03 ksoftirqd/1
    7 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/1
```

Gromacs 4.6.5 MPI+CUDA on PLX



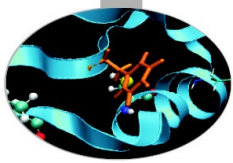
```
>module load autoload gromacs/4.6.5
>module help gromacs/4.6.5

#!/bin/bash
#PBS -N gmx
#PBS -l select=1:ncpus=12:mpiprocs=2:ngpus=2:mem=47GB
#PBS -q <queue>
#PBS -l walltime=1:00:00
#PBS -A <account_nr>

cd $PBS_O_WORKDIR                    ==> change to current dir

module load profile/advanced
module load autoload gromacs/4.6.5

export OMP_NUM_THREADS=1             ==> set nr. Of OpenMP threads to 1
#                                    ==> set total MPI tasks = 2 and bind to two GPUs

mdrun=$(which mdrun_mpi_cuda)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01 "
mpirun -np 2 $cmd
```

Gromacs 4.6.5 MPI+CUDA on PLX



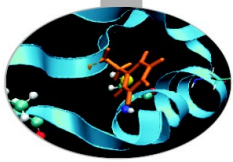
```
File  Edit  View  Search  Terminal  Help
top - 10:17:06 up 64 days, 17:13,  0 users,  load average: 0.64, 1.82, 2.09
Tasks: 253 total,   3 running, 250 sleeping,   0 stopped,   0 zombie
Cpu(s): 16.6%us,  0.1%sy,  0.0%ni, 83.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  49431720k total,  2195440k used, 47236280k free,   422160k buffers
Swap:  8193140k total,        0k used,  8193140k free,   932580k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 6740 agrottes  25   0 68.2g  51m  24m R 100.2  0.1   0:13.42 mdrun_mpi_cuda
 6739 agrottes  25   0 68.2g  52m  25m R 99.9  0.1   0:13.41 mdrun_mpi_cuda
    1 root      15   0 10372  696  584 S  0.0  0.0   0:03.16 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.91 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.04 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.91 migration/1
    6 root      34  19     0    0    0 S  0.0  0.0   0:00.03 ksoftirqd/1
    7 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/1
    8 root      RT  -5     0    0    0 S  0.0  0.0   0:00.76 migration/2
    9 root      34  19     0    0    0 S  0.0  0.0   0:00.07 ksoftirqd/2
   10 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/2
   11 root      RT  -5     0    0    0 S  0.0  0.0   0:00.40 migration/3
   12 root      34  19     0    0    0 S  0.0  0.0   0:00.02 ksoftirqd/3
```

Gromacs 4.6.5 MPI/OpenMP+CUDA on PLX



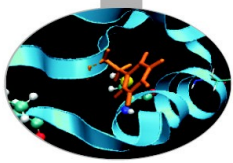
```
>module load autoload gromacs/4.6.5
>module help gromacs/4.6.5

#!/bin/bash
#PBS -N gmx
#PBS -l select=1:ncpus=12:mpiprocs=2:ngpus=2:mem=47GB
#PBS -q <queue>
#PBS -l walltime=1:00:00
#PBS -A <account_nr>

cd $PBS_O_WORKDIR                    ==> change to current dir

module load profile/advanced
module load autoload gromacs/4.6.5

export OMP_NUM_THREADS=6             ==> set nr. Of OpenMP threads to 6
#                                    ==> set total mpi tasks = 2 and bind to two GPUs

mdrun=$(which mdrun_mpi_cuda)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01 "
mpirun -np 2 $cmd
```

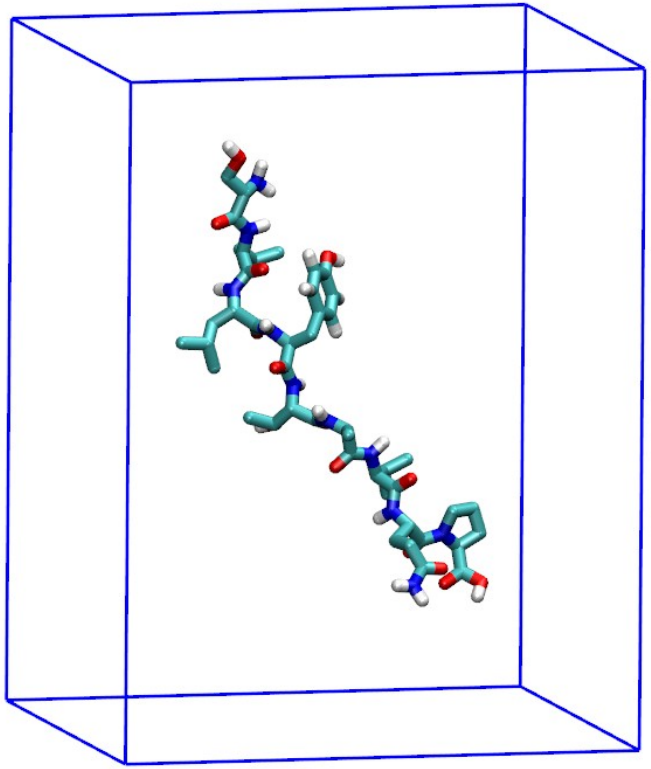
Gromacs 4.6.5 MPI/OpenMP+CUDA on PLX



```
File  Edit  View  Search  Terminal  Help
top - 10:09:28 up 64 days, 17:05,  0 users,  load average: 3.86, 3.24, 2.27
Tasks: 253 total,   3 running, 250 sleeping,   0 stopped,   0 zombie
Cpu(s): 99.7%us,  0.3%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  49431720k total,  2202496k used, 47229224k free,   422156k buffers
Swap:  8193140k total,        0k used,  8193140k free,   932516k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5808 agrottes  18   0 68.3g  56m  25m R 600.0  0.1   1:39.62 mdrun_mpi_cuda
 5809 agrottes  25   0 68.3g  55m  24m R 599.7  0.1   1:39.15 mdrun_mpi_cuda
    1 root      15   0 10372  696  584 S  0.0  0.0   0:03.16 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.91 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.04 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.91 migration/1
    6 root      34  19     0    0    0 S  0.0  0.0   0:00.03 ksoftirqd/1
    7 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/1
    8 root      RT  -5     0    0    0 S  0.0  0.0   0:00.76 migration/2
    9 root      34  19     0    0    0 S  0.0  0.0   0:00.07 ksoftirqd/2
   10 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/2
   11 root      RT  -5     0    0    0 S  0.0  0.0   0:00.40 migration/3
   12 root      34  19     0    0    0 S  0.0  0.0   0:00.02 ksoftirqd/3
```

MD Performance on hybrid CPU-GPU clusters (PLX)





- Pure MPI (8 MPI procs) →88.3 ns/day

  (scheme = verlet, domain decomposition)
- MPI-CUDA (2 MPI procs + 2 GPUs) →155.6 ns/day
- MPI/OpenMP/CUDA →229.9 ns/day

  (2 MPI procs + 4 threads + 2 GPUs)
- Pure MPI (8 MPI procs) →134.3 ns/day

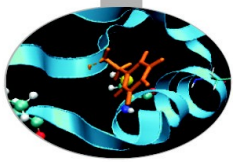
  (scheme = groups, domain decomposition)

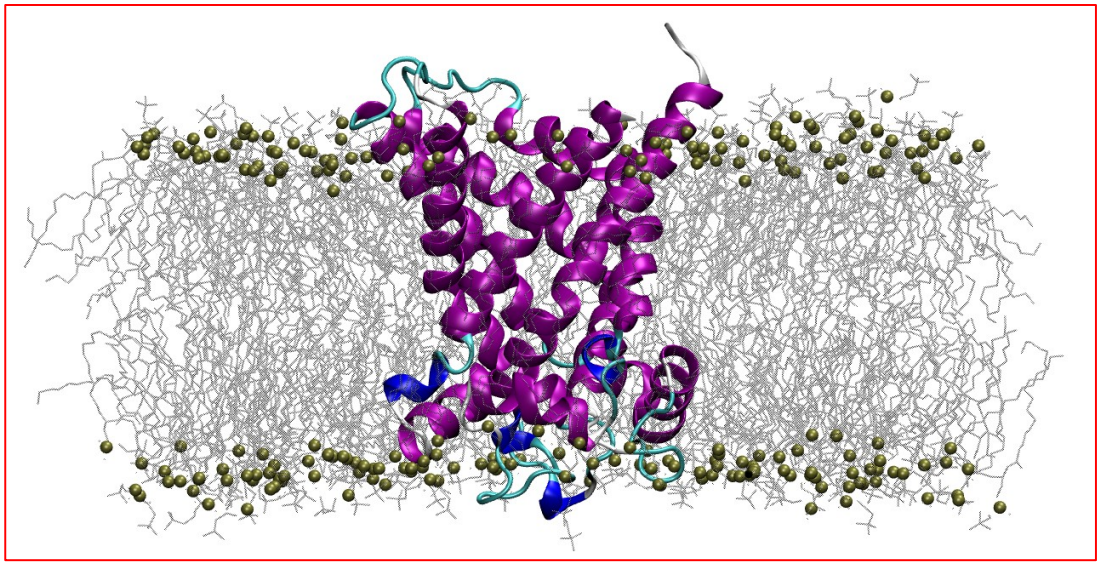
Small peptide in a box of water, ~3300 atoms

Gromacs 4.6.5 with GPU PME, 1 nm cut-off, T = 300 K

PRACE CINECA

MD Performance on hybrid CPU-GPU clusters (Eurora)

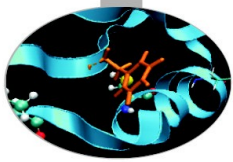


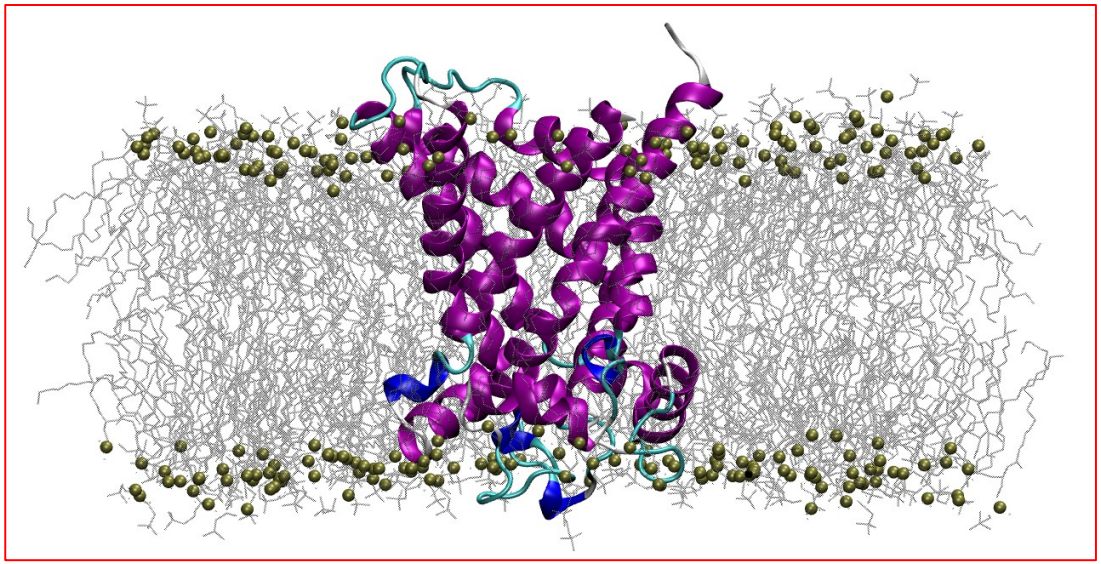


ATP/ADP Mitochondrial Carrier, 92K atoms
Gromacs 4.6.5 with GPU
PME for long electrostatics, 300 K
Domain Decomposition
Cut-off = 1 nm
Neigh. Search scheme = Verlet

- Pure MPI (16 MPI procs) →11.63 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) →9.53 ns/day
- MPI/OpenMP/CUDA (2 MPI procs + 8 threads + 2 GPUs) →24.6 ns/day

MD Performance on hybrid CPU-GPU clusters (Eurora)



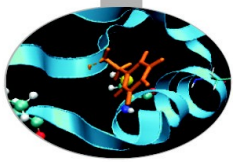


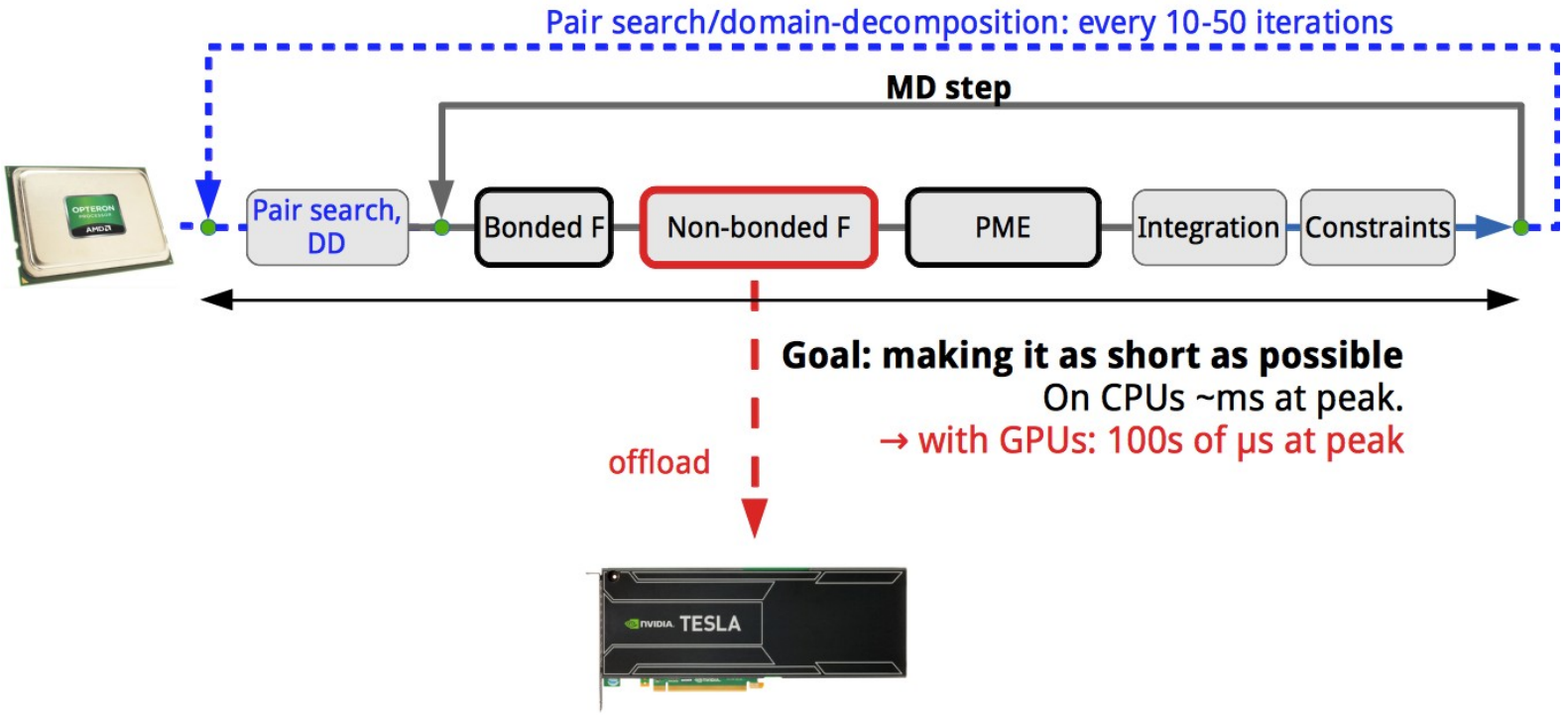
ATP/ADP Mitochondrial Carrier, 92K atoms
Gromacs 4.6.5 with GPU
PME for long electrostatics, 300 K
Domain Decomposition
Cut-off = 1 nm
Neigh. Search scheme = Verlet

- Pure MPI (16 MPI procs) →11.63 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) →9.53 ns/day
- MPI/OpenMP/CUDA (2 MPI procs + 8 threads + 2 GPUs) →24.6 ns/day
- Pure MPI (16 MPI procs) Scheme = Group →16.48 ns/day

GPU acceleration in GROMACS

www.gromacs.org





The idea behind the native GPU acceleration in GROMACS is that we offload the heavy nonbonded force calculation to an accelerator (either a GPU or something else), while the CPU does bonded forces and lattice summation (PME) in the mean time.