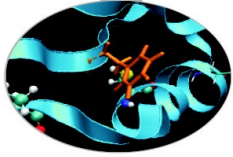


Running MD on HPC architectures I. Hybrid Clusters

Alessandro Grottesi
Cineca



Today's lecture

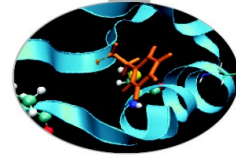


You will learn:

- Gromacs @ CINECA: set up and launch of simulations
- Launch MD code (GROMACS, NAMD)
- Optimize performance and benchmarking



Eurora



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)**

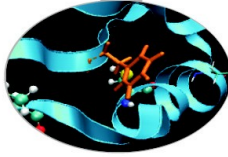
Number of accelerators: **64 nVIDIA Tesla K20 (Kepler) + 64 Intel Xeon Phi (MIC)**

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

OS: **RedHat CentOS release 6.3, 64 bit**



Pico



Model: IBM NeXtScale server

Architecture: Linux Infiniband Cluster

Processors Type: Intel Xeon (Ten-Core) E5-2670v2 2.50 GHz (Compute)

Number of nodes: 54 Compute + 4 visualization + 2 Login + 14 other

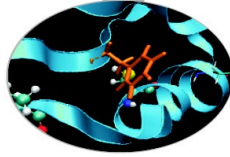
Number of cores: 1080 (compute)

Number of accelerators: 4 + 2 (only on viz nodes)

RAM: 128 GB/Compute node (2 viz nodes with 512GB)



Galileo



Model: IBM NeXtScale

Architecture: Linux Infiniband Cluster

Nodes: 516

Processors: 8-cores Intel Haswell 2.40 GHz (2 per node)

Cores: 16 cores/node, 8256 cores in total

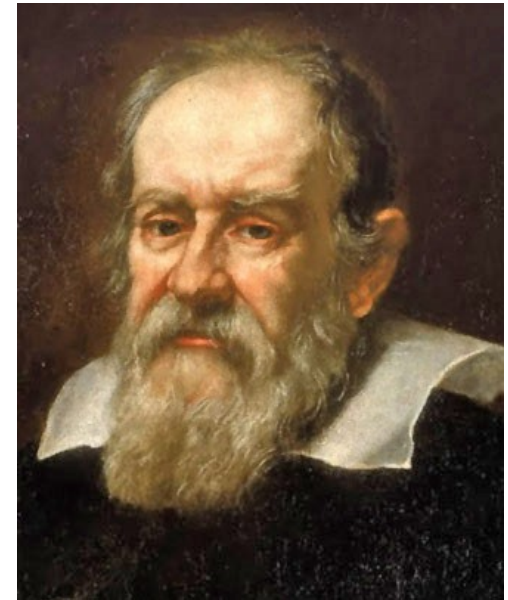
Accelerators: 2 Intel Phi 7120p per node on 384 nodes
(768 in total)

RAM: 128 GB/node, 8 GB/core

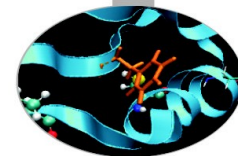
Internal Network: Infiniband with 4x QDR switches

Disk Space: 2,500 TB of local storage

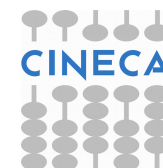
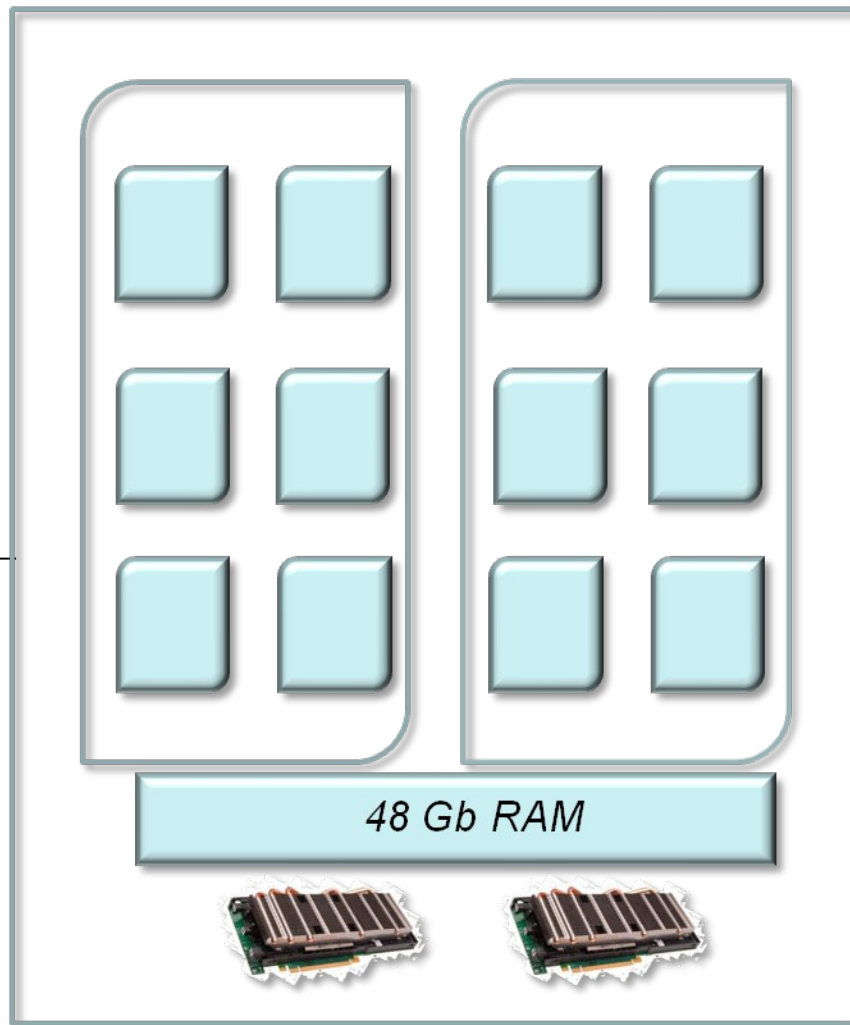
Peak Performance: xxx TFlop/s (to be defined)



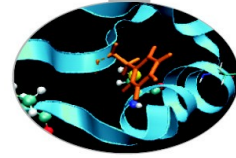
Compute nodes



Infiniband connection



Molecular Dynamics and accelerators



Intel Xeon Phi



Nvidia K80

– GROMACS

- Under development, currently (Jan 2015) only native-mode version available.

– NAMD

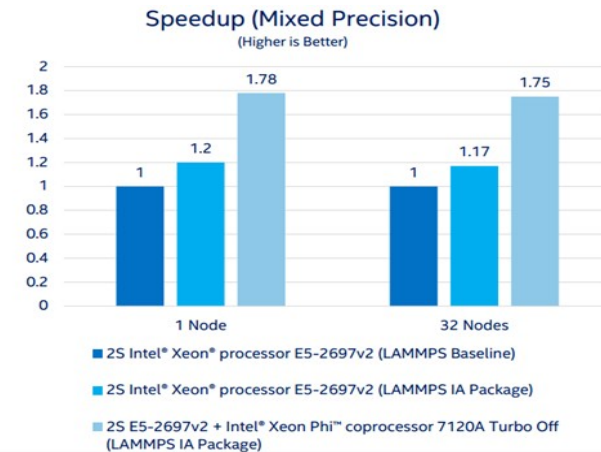
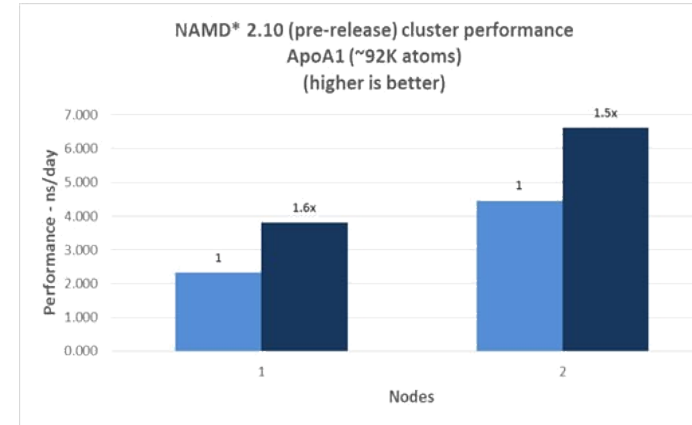
- Pre-release version of NAMD 2.10 has Xeon PHI support but still under development. Speed-ups < 2 for ApoA1 and STMV benchmarks.

– LAMMPS

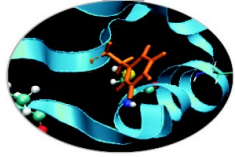
- Xeon PHI support available in current downloads for non-bonded calculations. Reported speed-ups of about 1.78x compared to non-accelerated code (one coprocessor/node) for Rhodopsin benchmark. Higher speed-ups obtained with materials simulations.

– AMBER

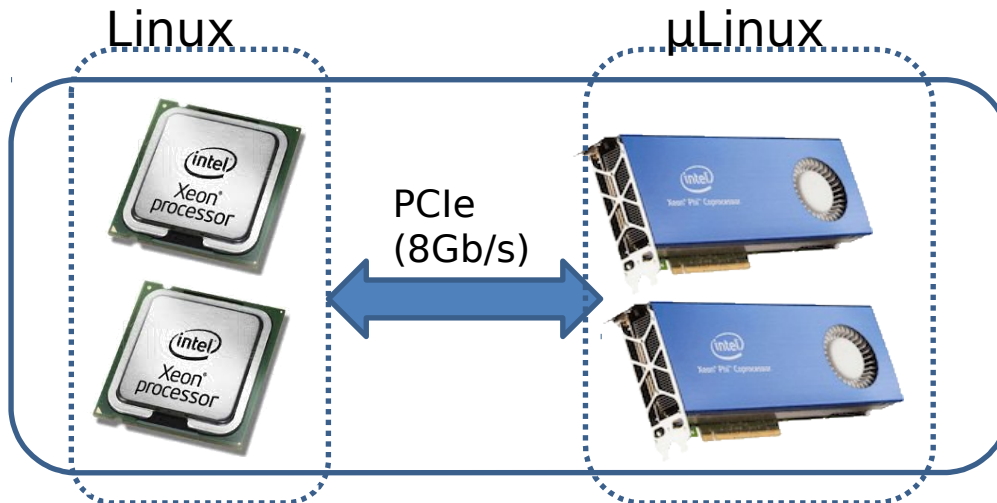
- Xeon PHI-enabled version released 6 Aug 2014. Waiting for benchmarks.



Intel Xeon PHI overview

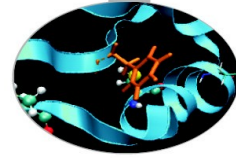


- Intel product line based on Intel's Many Integrated Core (MIC) technology where many low, power cores (>50) are packed on a single chip.
- Currently available device (Knight's Corner or KNC) can be seen as a co-processor, in direct competition to NVIDIA GPU for HPC.
 - connection to host CPU via PCI-eXpress link.
- But unlike GPU technology is not too dissimilar from host CPU → not essential to rewrite code, in principle just re-compile (with Intel compilers). Should lead to shorter development path in most cases.
- Doesn't mean though that code does need not porting – to obtain peak performance some optimisation is needed.

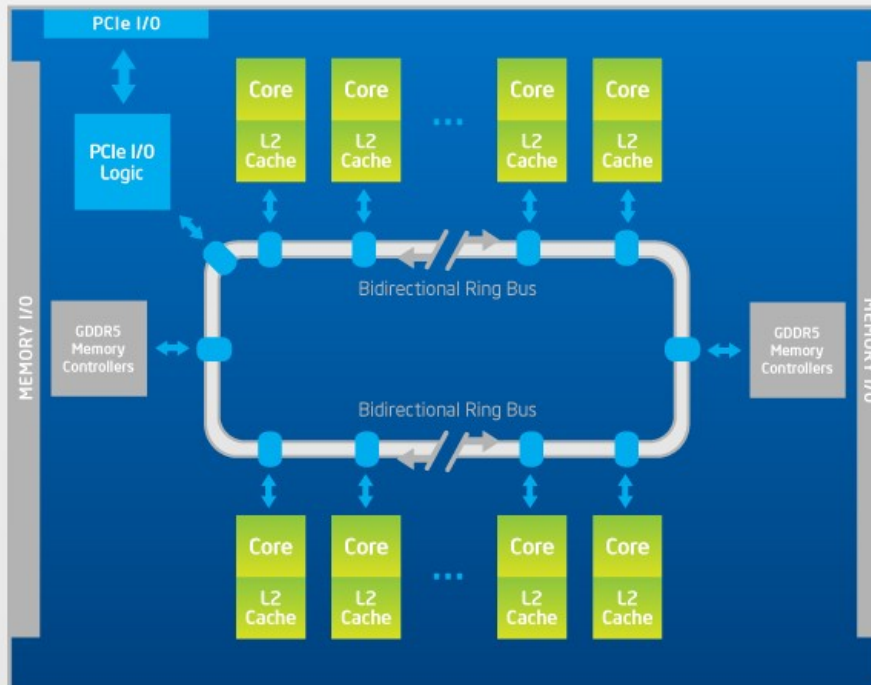


Most powerful supercomputer in TOP500 (Tianhe-2) uses 48000 Xeon PHI cards.

Xeon Phi architecture (KNC)



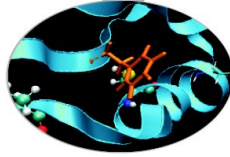
Intel® Xeon Phi™ Coprocessor Block Diagram



- 22nm technology
- Up to 61 cores (Pentium-like), ~1.1 Ghz (dep. model)
- 352 Gb/s memory bandwidth (*fast*).
- Upto 244 threads (i.e. 4 threads/core)
- 512 bit SIMD (vector) unit
- 8-16 Gb on board memory,
- ~ 1Tflop peak performance

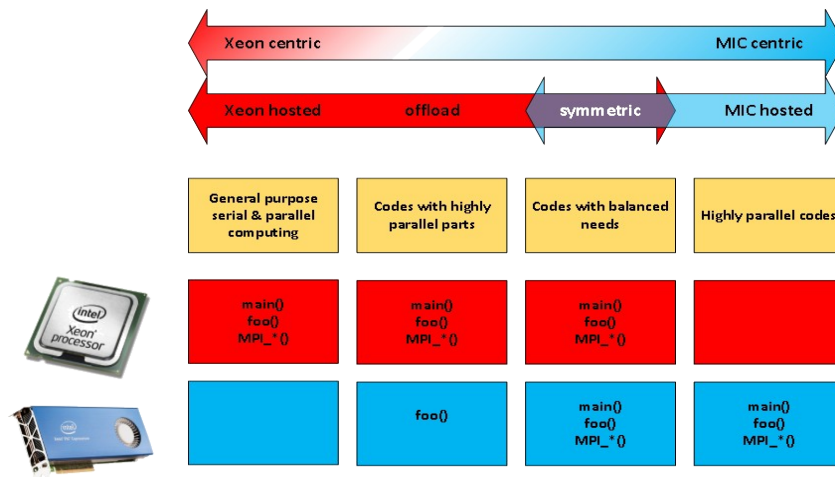
Includes also sensors for monitoring temperature and power consumption.

Xeon PHI software models

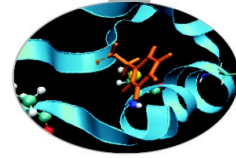


Three software models are supported:

1. Co-processor only (native mode): MPI ranks are solely on the Xeon PHI. App can be launched from host or co-processor.
2. Symmetric model: MPI ranks on both host and co-processor.
3. Off-load model: MPI ranks only on host which offload work (e.g. by OpenCL or OpenMP) onto co-processor.

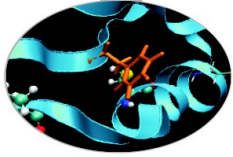


Classical Molecular Dynamics and Intel Xeon PHI



- Development some way behind GPU-CUDA versions of classical MD programs (which started about 4 years ago).
- But given that there is no need to rewrite in new languages (e.g. CUDA) development path should be shorter.
- Most current Xeon PHI versions seem to be based on the off-load model to exploit CUDA developments.
- Off-loaded calculations invariably involve non-bonded dispersion interactions may also include PME, energy calculation etc.
- Intel maintains a list of “recipes” for building Xeon PHI applications (not just MD):

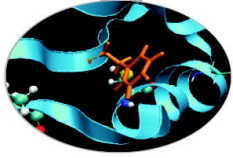
<https://software.intel.com/en-us/articles/namd-for-intel-xeon-phi-coprocessor>



Running MD code on Eurora (GROMACS)

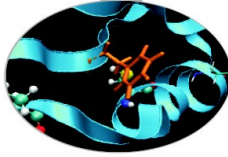


GROMACS: what for...?



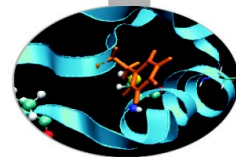
1. Minimization
2. Molecular Dynamics (classic, brownian, Langevin)
3. Normal Mode Analysis
4. Essential Dynamics and Sampling
5. Free Energy calculations (FEP, Umbrella sampling, AFM)
6. Replica Exchange Molecular Dynamics
7. Coarse-Grained MD
8. Metadynamics
9. Much more...

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

genion

Generate input file for mdrun

grompp

topol.tpr

Actually run the simulation

mdrun

Analysis of trajectory files

trjconv

g_rms

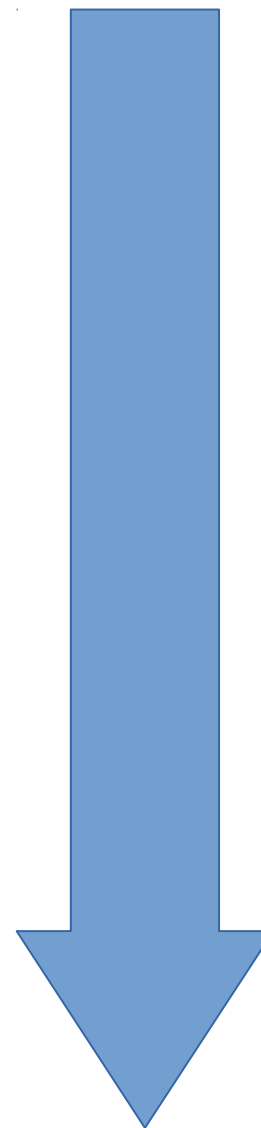
g_covar

g_anaeig

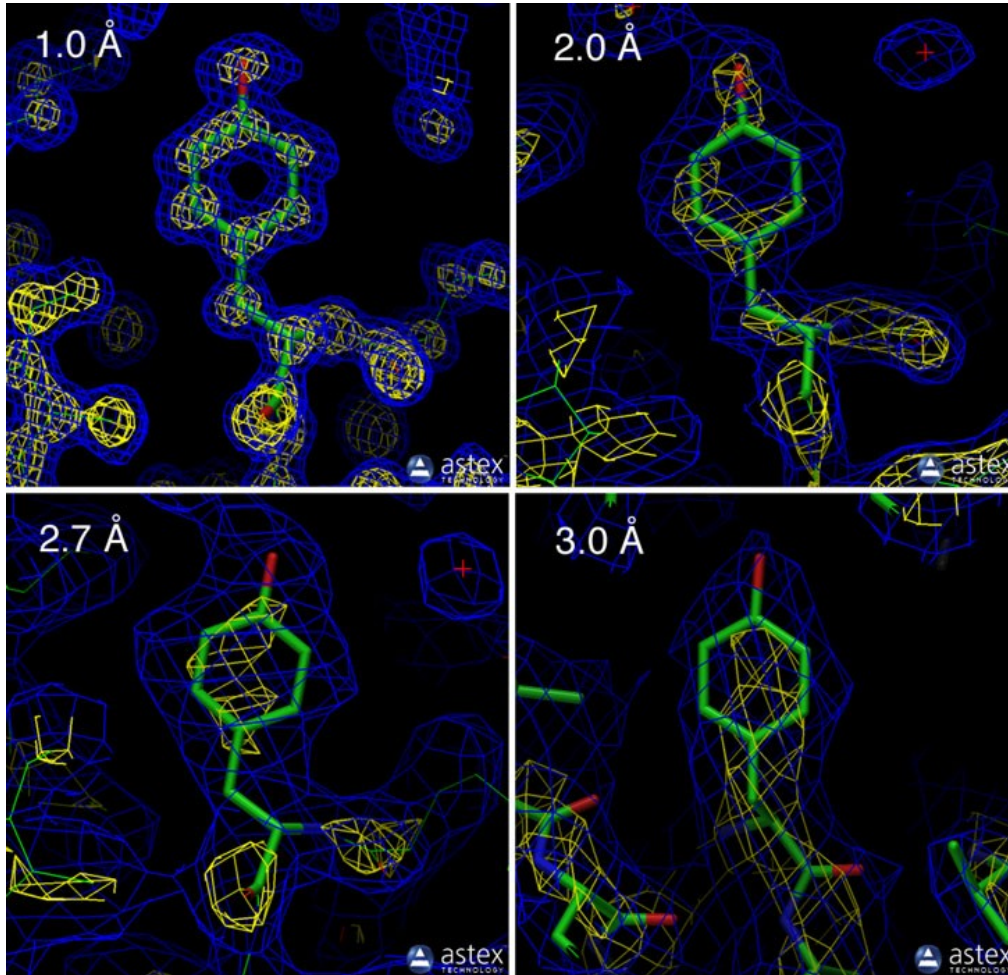
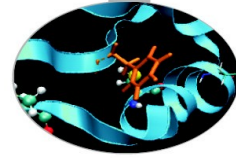
g_energy

g_rmsf

g_rdf



Initial coordinates: X-Ray vs. NMR

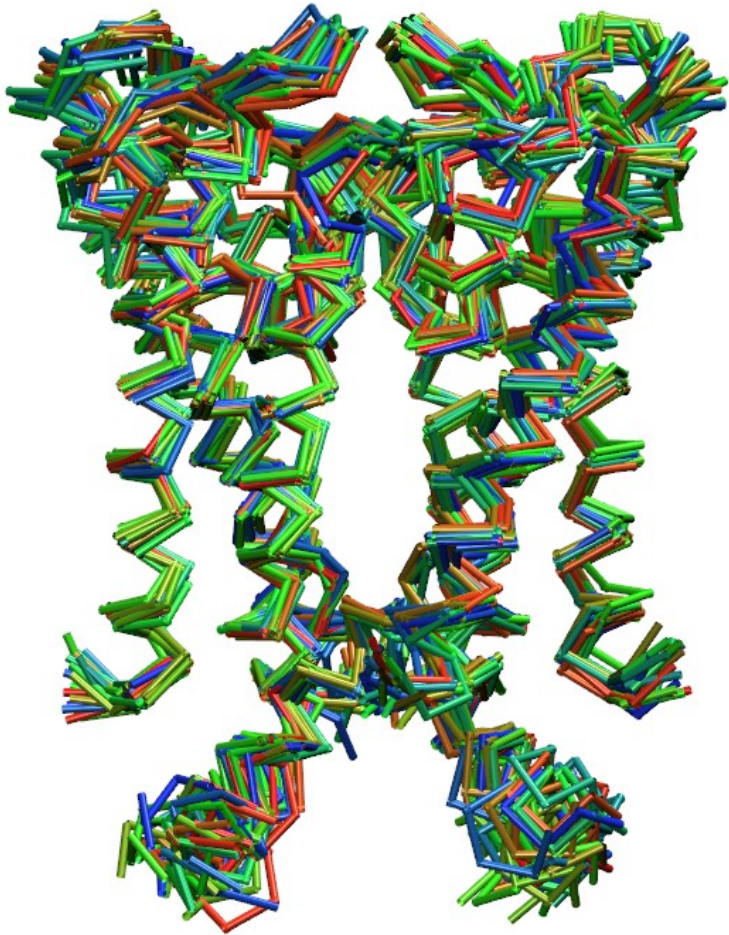
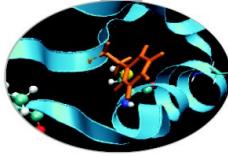


Higher X-ray resolution allows to use a more reliable starting structure in terms of amino-acids stereo-chemistry and accuracy of atomic positions

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

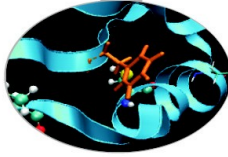
Initial coordinates: X-Ray vs. NMR



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 uncertainties in the position of atomic coordinates.

KcsA Potassium channel
(PDB code: 2K1E)

Generate topology: pdb2gmx



To convert a structure pdb file into a Gromacs topology:

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

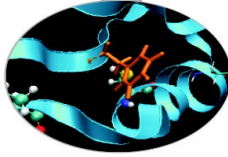
input:

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...

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 ionization state for N- and C-terminus in proteins:

```
pdb2gmx -f input.pdb -ignh -ter
```

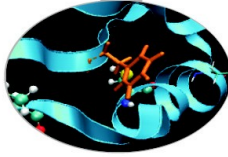
Select N-terminus type (start)

- 0: NH3+
- 1: NH2
- 2: None

Select C-terminus type (end)

- 0: COO-
- 1: COOH
- 2: None

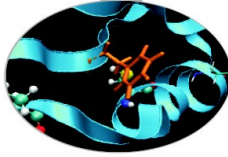
pdb2gmx: pH and net charge



By default, pdb2gmx assumes you want to simulate your protein in a neutral pH environment (pH=7). Hence the net charge carried by ionizable 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 $\text{pH} \neq 7$ consists of using the flags `-lys -arg -asp -glu` to set up interactively the net charge on lysine, arginine, apartate and glutamate residue, respectively. 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

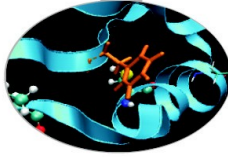
0. Not protonated (charge 0) (LYS)
1. Protonated (charge +1) (LYSH)

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



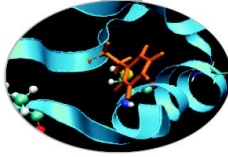
Histidine residues play an important role in protein function as they are often located within catalytic pockets and binding sites or could be involved in interactions with prosthetic group (HEME) or could bind metal ions for important enzymatic activities (cytochromes, chlorophylls, etc.)

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

```
pdb2gmx -f input.pdb -ignh -ter -his
```

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

How to generate the box: editconf



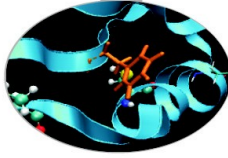
Structure generated file has to be immersed in a box of water molecules (or alternative solvent) prior to run an MD simulation. Different types of box are available in Gromacs (triclinic, cubic, dodecahedron or octahedron) and can be generated by the command:

```
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.

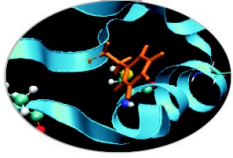
Box solvation: genbox



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

```
genbox -cp conf.gro -cs spc216.gro -o out.gro -p topol.top
```

Ionic strength: genion



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

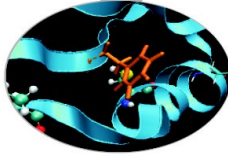
$$\text{solute charge} + \text{solvent charge} = 0$$

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 needed 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:

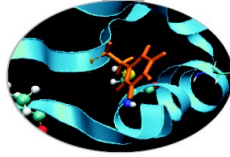
```
gmxdump -s topol.tpr
```

Output control

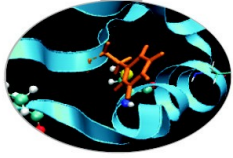
Van der Waals and electrostatics

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
```



Output control parameters

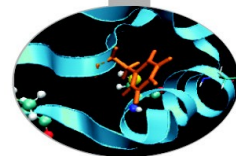


nsteps	= 500000	Total number of steps
nstxout	= 5000	coords output frequency for traj.trr
nstvout	= 5000	velocity output frequency for traj.trr
nstlog	= 5000	output frequency for log file (md.log)
nstenergy	= 250	output frequency for energy file ener.edr
nstxtcout	= 250	coords output frequency for traj.xtc
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

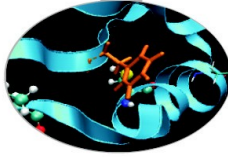
ewald_rtol = 1e-05

ewald_geometry = 3d

epsilon_surface = 0

optimize_fft = no

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

← weak temperature coupling

; 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

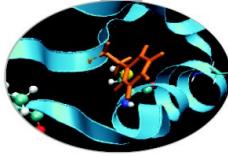
; Scaling of reference coordinates, No, All or COM

refcoord_scaling = No

; Random seed for Andersen thermostat

andersen_seed = 815131

Run the simulation: mdrun



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

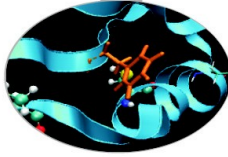
domain decomposition

particle decomposition

The command generates many output files 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 | checkpoint file for restarting runs. |
| 6. md.log | log file with output control |

What if it all crashes...?

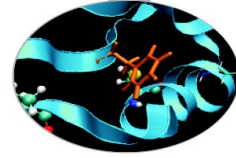


A .cpt file is produced by mdrun at specified intervals (mdrun -cpt n, where n is the interval in minutes), and contains information on all the state variables in a simulated system. In the case of a crash (hardware failure, power outage, etc), a checkpoint file can be used to resume the simulations 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



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

```
#!/bin/bash  
#PBS -N gmx  
#PBS -l select=1:ncpus=16:mpiprocs=16:mem=14GB  
#PBS -q <queue>  
#PBS -l walltime=1:00:00  
#PBS -A <account_nr>
```

```
cd $PBS_O_WORKDIR ==> change to current dir
```

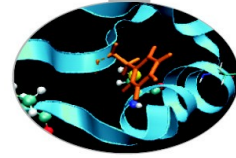
```
module load profile/advanced  
module load autoload gromacs/4.6.5
```

```
export OMP_NUM_THREADS=1 ==> set nr. Of OpenMP threads per MPI proc to 1  
# ==> set total mpi tasks = 16
```

```
mdrun=$(which mdrun_mpi)  
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -nb cpu"  
mpirun -np 16 $cmd
```



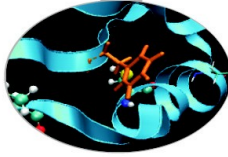
Gromacs 4.6.5, pure MPI on Eurora



```

File Edit View Search Terminal Help
top - 18:59:29 up 191 days, 8:44, 1 user, load average: 6.03, 6.58, 9.76
Tasks: 521 total, 17 running, 504 sleeping, 0 stopped, 0 zombie
Cpu(s): 99.1%us, 0.9%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 16294372k total, 1183416k used, 15110956k free, 49656k buffers
Swap: 8191992k total, 152304k used, 8039688k free, 184768k cached
  
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
25080	agrottes	20	0	192m	15m	12m	R	100.0	0.1	0:16.71	mdrun_mpi
25082	agrottes	20	0	192m	15m	12m	R	100.0	0.1	0:16.71	mdrun_mpi
25083	agrottes	20	0	192m	16m	13m	R	100.0	0.1	0:16.72	mdrun_mpi
25088	agrottes	20	0	192m	15m	12m	R	100.0	0.1	0:16.72	mdrun_mpi
25089	agrottes	20	0	192m	15m	12m	R	100.0	0.1	0:16.73	mdrun_mpi
25091	agrottes	20	0	192m	15m	12m	R	100.0	0.1	0:16.72	mdrun_mpi
25093	agrottes	20	0	192m	15m	12m	R	100.0	0.1	0:16.74	mdrun_mpi
25094	agrottes	20	0	192m	15m	12m	R	100.0	0.1	0:16.72	mdrun_mpi
25079	agrottes	20	0	52.2g	27m	23m	R	99.7	0.2	0:16.57	mdrun_mpi
25081	agrottes	20	0	192m	15m	12m	R	99.7	0.1	0:16.72	mdrun_mpi
25084	agrottes	20	0	192m	15m	12m	R	99.7	0.1	0:16.71	mdrun_mpi
25085	agrottes	20	0	192m	15m	12m	R	99.7	0.1	0:16.71	mdrun_mpi
25090	agrottes	20	0	192m	14m	11m	R	99.7	0.1	0:16.67	mdrun_mpi
25092	agrottes	20	0	192m	17m	12m	R	99.7	0.1	0:16.71	mdrun_mpi
25086	agrottes	20	0	192m	15m	12m	R	99.3	0.1	0:16.72	mdrun_mpi
25087	agrottes	20	0	192m	15m	12m	R	99.3	0.1	0:16.69	mdrun_mpi
3923	root	0	-20	8811m	320m	45m	S	0.3	2.0	1244:58	mmfsd
19541	abartoli	20	0	172m	5672	1048	S	0.3	0.0	0:00.87	deamon_mic_stat
25120	agrottes	20	0	15292	1576	940	R	0.3	0.0	0:00.04	top



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

```
>module help gromacs/4.6.5
```

```
#!/bin/bash
```

```
#PBS -N gmx
```

```
#PBS -l select=1:ncpus=16:mpiprocs=2:ngpus=2:mem=14GB
```

```
#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

```
#
```

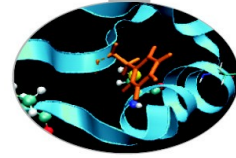
==> set total MPI tasks = 2 and bind to two GPUs

```
mdrun=$(which mdrun_mpi_cuda)
```

```
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01 "
```

```
mpirun -np 2 $cmd
```



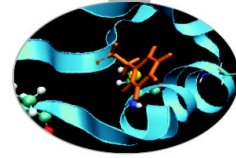


```

File Edit View Search Terminal Help
top - 18:55:58 up 191 days,  8:40,  1 user,  load average: 2.63, 8.31, 11.05
Tasks: 509 total,   5 running, 504 sleeping,   0 stopped,   0 zombie
Cpu(s): 12.5%us,  0.1%sy,  0.0%ni, 87.4%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  16294372k total, 1306364k used, 14988008k free,   49428k buffers
Swap:  8191992k total,  152304k used,  8039688k free,   185300k cached

  PID  USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM     TIME+  COMMAND
 24907 agrottes  20   0 52.3g 106m 91m  R  99.7  0.7   0:20.38 mdrun_mpi
 24908 agrottes  20   0 52.2g 103m 90m  R  99.4  0.7   0:20.51 mdrun_mpi
 24915 agrottes  20   0 15292 1572 940  R   0.3  0.0   0:00.04 top
   1  root      20   0 19352  660 432  S   0.0  0.0   0:17.84 init
   2  root      20   0   0     0   0   S   0.0  0.0   0:00.64 kthreadd
   3  root      RT   0   0     0   0   S   0.0  0.0  21:41.03 migration/0
   4  root      20   0   0     0   0   S   0.0  0.0  34:50.30 ksoftirqd/0
   5  root      RT   0   0     0   0   S   0.0  0.0   0:00.01 migration/0
   6  root      RT   0   0     0   0   S   0.0  0.0   0:42.24 watchdog/0
   7  root      RT   0   0     0   0   S   0.0  0.0  15:43.68 migration/1
   8  root      RT   0   0     0   0   S   0.0  0.0   0:00.01 migration/1
   9  root      20   0   0     0   0   S   0.0  0.0  17:55.57 ksoftirqd/1
  10  root      RT   0   0     0   0   S   0.0  0.0   0:28.06 watchdog/1
  11  root      RT   0   0     0   0   S   0.0  0.0  11:14.94 migration/2
  12  root      RT   0   0     0   0   S   0.0  0.0   0:00.01 migration/2
  13  root      20   0   0     0   0   S   0.0  0.0  12:02.44 ksoftirqd/2
  14  root      RT   0   0     0   0   S   0.0  0.0   0:18.76 watchdog/2
  15  root      RT   0   0     0   0   S   0.0  0.0  10:45.64 migration/3
  16  root      RT   0   0     0   0   S   0.0  0.0   0:00.01 migration/3
  
```


Gromacs 4.6.5 MPI/OpenMP+CUDA On Eurora



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

```
#!/bin/bash
```

```
#PBS -N gmx
```

```
#PBS -l select=1:ncpus=16:mpiprocs=2:ngpus=2:mem=14GB
```

```
#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=8
```

==> set nr. Of OpenMP threads to 8

```
#
```

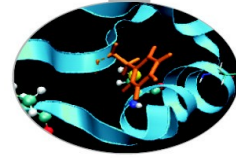
==> set total mpi tasks = 2 and bind to two GPUs

```
mdrun=$(which mdrun_mpi_cuda)
```

```
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01 "
```

```
mpirun -np 2 $cmd
```





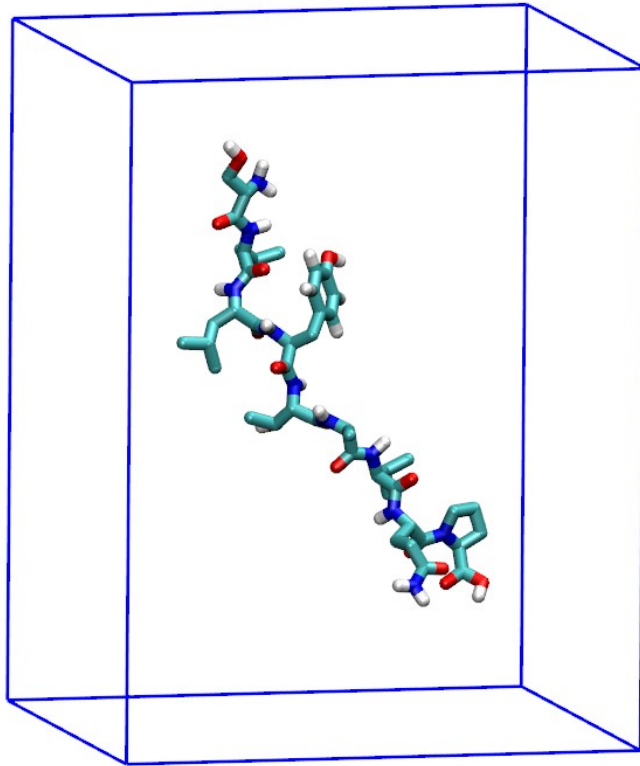
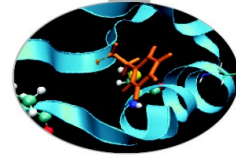
File Edit View Search Terminal Help

```

top - 18:53:13 up 191 days, 8:37, 1 user, load average: 25.69, 13.10, 12.81
Tasks: 509 total, 3 running, 506 sleeping, 0 stopped, 0 zombie
Cpu(s): 95.1%us, 4.9%sy, 0.0%ni, 0.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 16294372k total, 1387360k used, 14907012k free, 49240k buffers
Swap: 8191992k total, 152304k used, 8039688k free, 187840k cached
  
```

PID	USER	PP	NT	VT/PT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
24731	agrottes	20	0	53.4g	153m	92m	R	798.6	1.0	11:57.22	mdrun_mpi
24732	agrottes	20	0	53.3g	137m	92m	R	798.6	0.9	11:56.41	mdrun_mpi
67	root	20	0	0	0	0	S	0.3	0.0	128:39.59	events/0
75	root	20	0	0	0	0	S	0.3	0.0	75:53.93	events/8
19426	abartoli	20	0	178m	6692	1720	S	0.3	0.0	0:02.22	daemon_gpu_stat
1	root	20	0	19352	660	432	S	0.0	0.0	0:17.84	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.64	kthreadd
3	root	RT	0	0	0	0	S	0.0	0.0	21:41.03	migration/0
4	root	20	0	0	0	0	S	0.0	0.0	34:50.30	ksoftirqd/0
5	root	RT	0	0	0	0	S	0.0	0.0	0:00.01	migration/0
6	root	RT	0	0	0	0	S	0.0	0.0	0:42.24	watchdog/0
7	root	RT	0	0	0	0	S	0.0	0.0	15:43.68	migration/1
8	root	RT	0	0	0	0	S	0.0	0.0	0:00.01	migration/1
9	root	20	0	0	0	0	S	0.0	0.0	17:55.57	ksoftirqd/1
10	root	RT	0	0	0	0	S	0.0	0.0	0:28.06	watchdog/1
11	root	RT	0	0	0	0	S	0.0	0.0	11:14.94	migration/2
12	root	RT	0	0	0	0	S	0.0	0.0	0:00.01	migration/2
13	root	20	0	0	0	0	S	0.0	0.0	12:02.43	ksoftirqd/2
14	root	RT	0	0	0	0	S	0.0	0.0	0:18.76	watchdog/2

MD Performance on hybrid CPU-GPU clusters

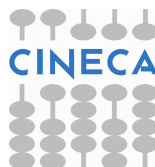


- Pure MPI (12 MPI procs) → 123.9 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) → 148.0 ns/day
- MPI/OpenMP/CUDA (2 MPI procs + 6 threads + 2 GPUs) → 253.0 ns/day

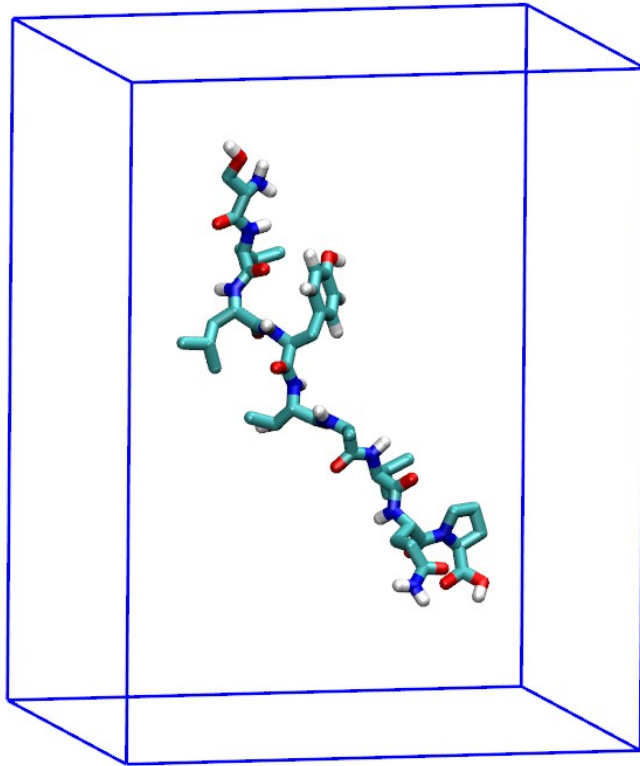
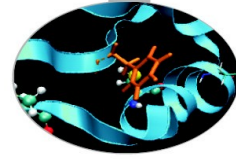
Small peptide in a box of water, ~3300 atoms

Gromacs 4.6.5 with GPU

PME for long electrostatics, 1 nm cut-off, T = 300 K



Domain vs. Particle Decomposition

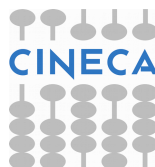


- DD – 8 MPI procs → 103.7 ns/day
- PD – 8 MPI procs → 145.8 ns/day

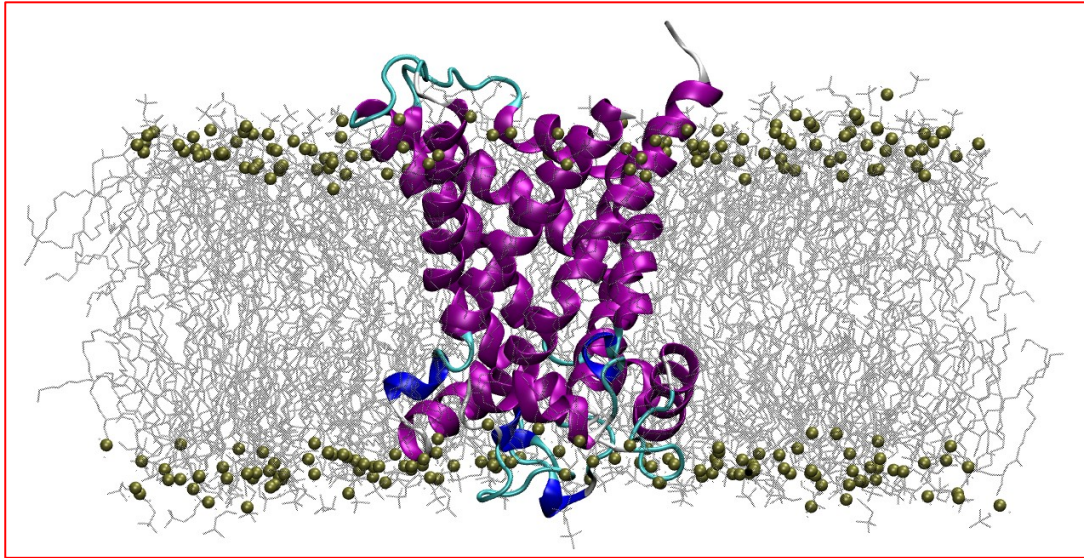
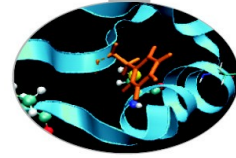
Small peptide in a box of water, ~3300 atoms

Gromacs 4.6.5 with GPU

PME for long electrostatics, 1 nm cut-off, T = 300 K



MD Performance on hybrid CPU-GPU clusters (Eurora)

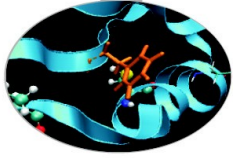


ATP/ADP Mitochondrial Carrier,
92K atoms
Gromacs 4.6.5 with GPU
PME for long electrostatics, 300 K
Domain Decomposition

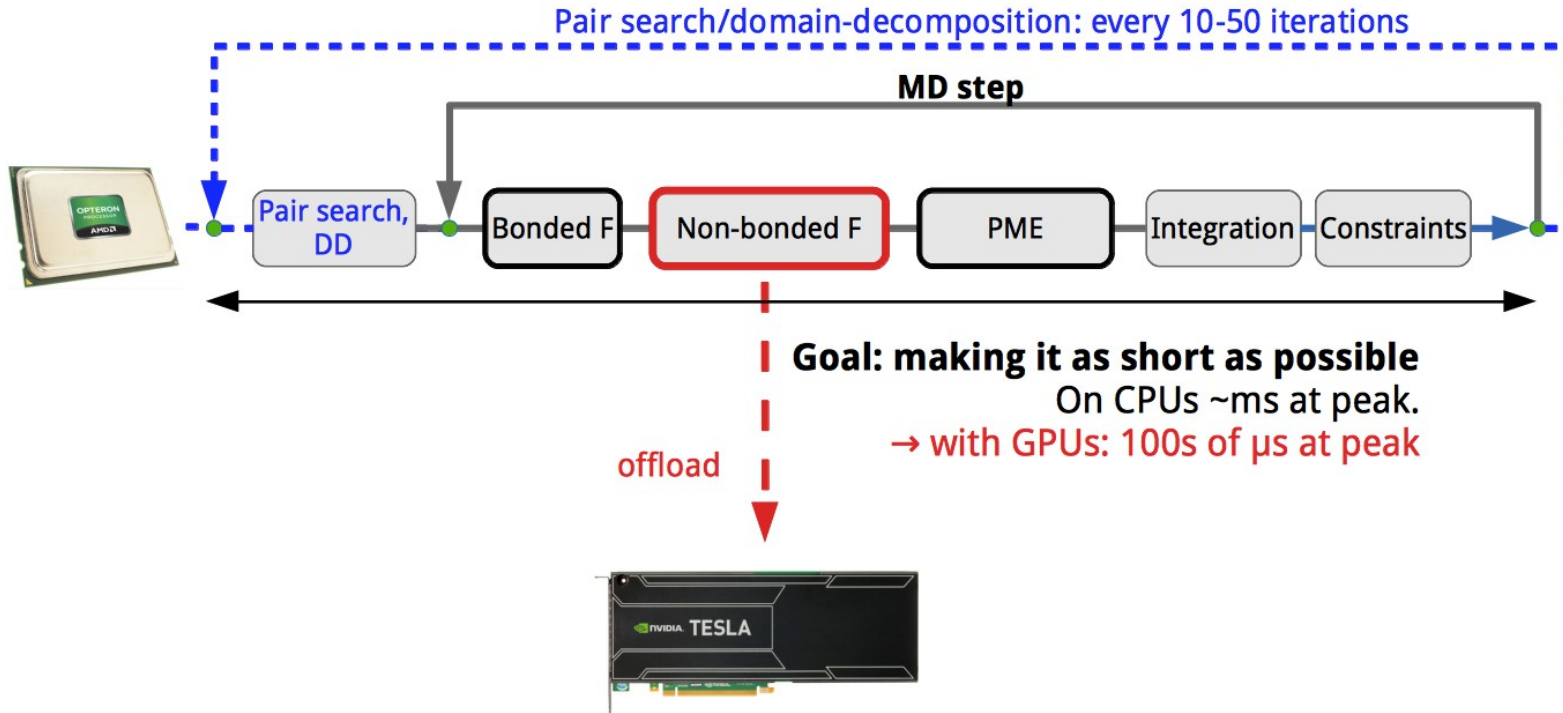
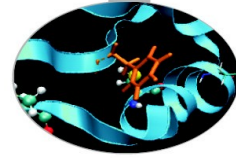
Test performed on Eurora, with cut-off radius = 1 nm

- 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) → 24.18 ns/day

MD Optimizzation on hybrid CPU-GPU Clusters



- System size and composition:
- Choice of PME vs. Cut-off based electrostatics
- Larger cut-off radius means a larger Verlet list ==> GPU is better than CPU
- Pure MPI jobs are suitable for small-sized systems



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 Xeon Phi), while the CPU does bonded forces and lattice summation (PME) in the mean time.