

Tutorial 2

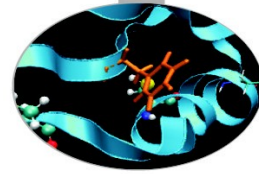
MD @ CINECA: scripts and benchmarks

Today you will learn:

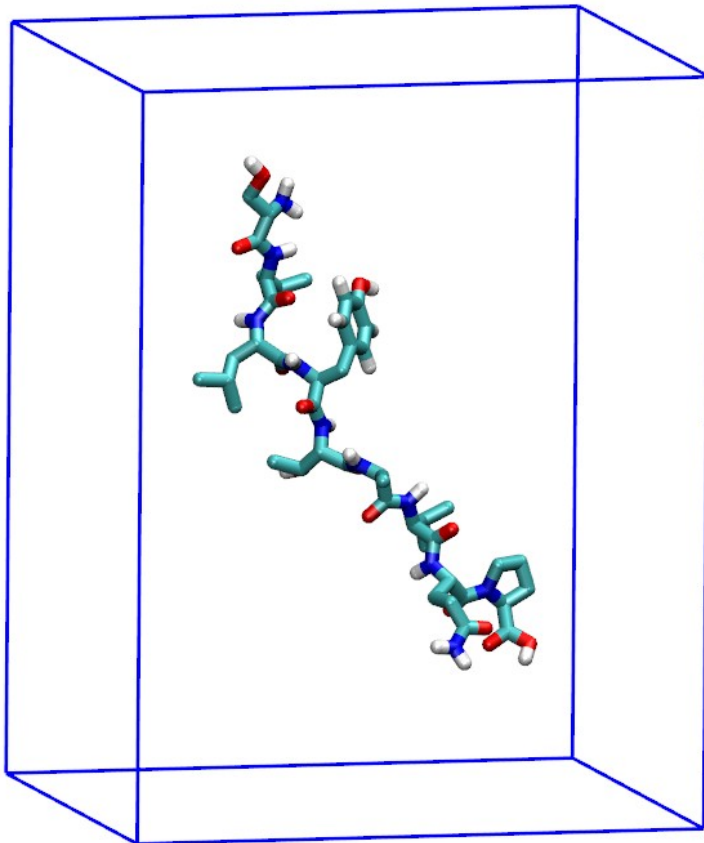
- How to run benchmarks on your system
- Compare and select best performance
- Getting familiar with acceleration in HPC MD



MD Performance on hybrid CPU-GPU clusters



Case study: small peptide in water



- 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
- MPI + Intel-Phi (12 MPI procs + 34 threads) → 134.1 ns/day

Small peptide in a box of water, ~3300 atoms

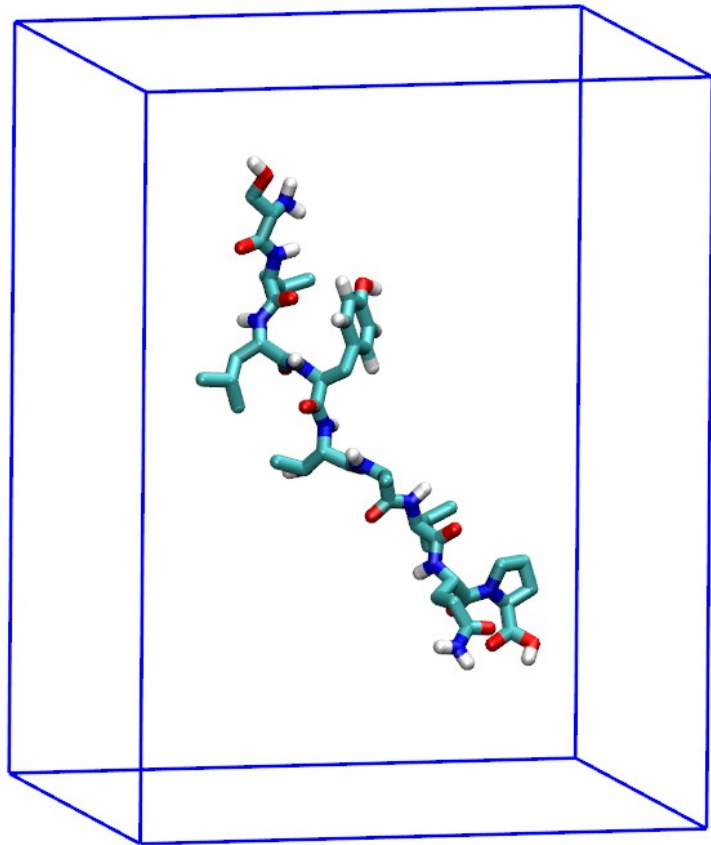
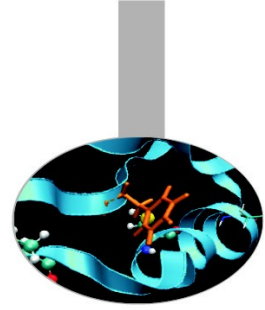
Gromacs 4.6.5 with GPU

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



MD Performance on hybrid CPU-GPU clusters

multiple MPI ranks with 2 GPUs



- MPI-CUDA (2 MPI procs + 2 GPUs) → 148.0 ns/day
- MPI-CUDA (4 MPI procs + 2 GPUs) → 304.1 ns/day
- MPI-CUDA (8 MPI procs + 2 GPUs) → 221.5ns/day
- MPI-CUDA (8MPI procs + 2 thrds + 2GPUs) → not applicable
- MPI-CUDA (16MPI + 2 GPUs) → not applicable

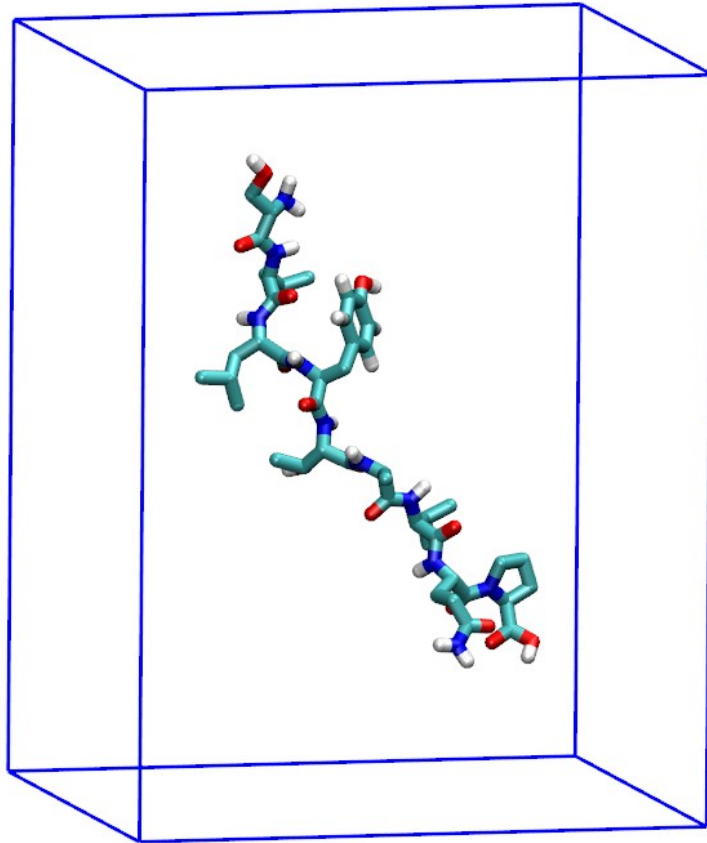
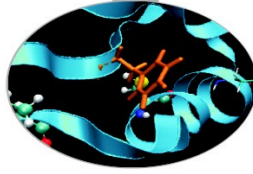
Small peptide in a box of water, ~3300 atoms

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

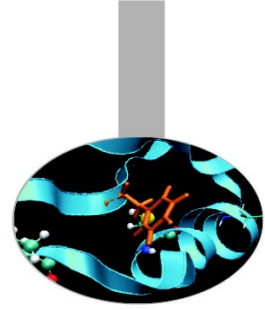
Small peptide in a box of water, ~3300 atoms

Gromacs 4.6.5 with GPU

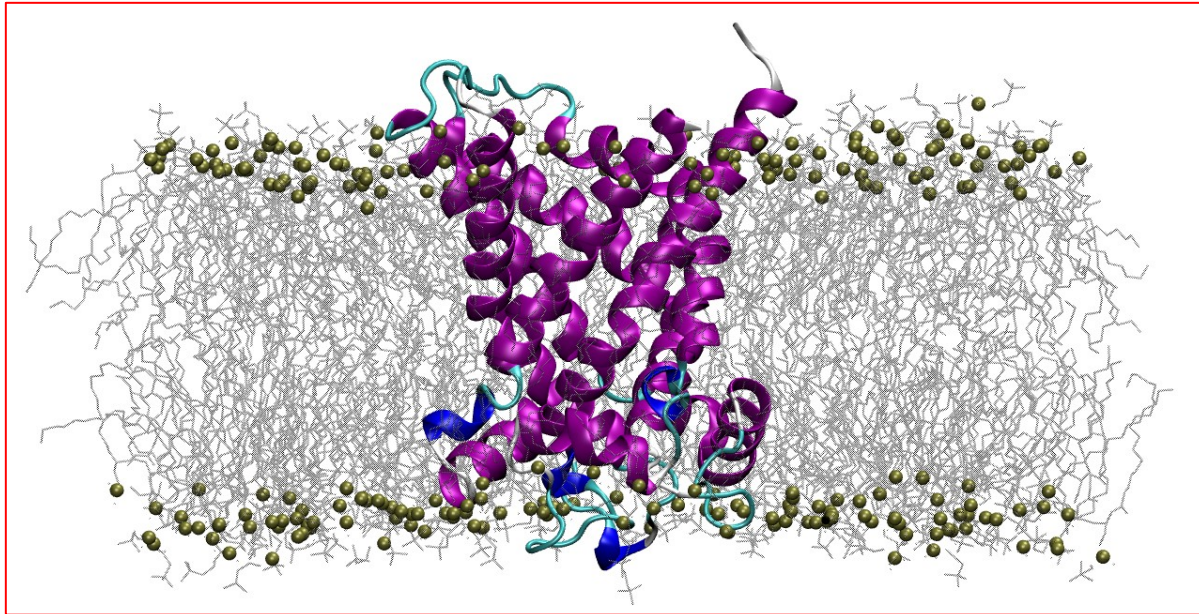
PME for long electrostatics, 1 nm cut-off, T = 300 K



MD Performance on hybrid CPU-GPU clusters (Eurora)



Case study: membrane protein

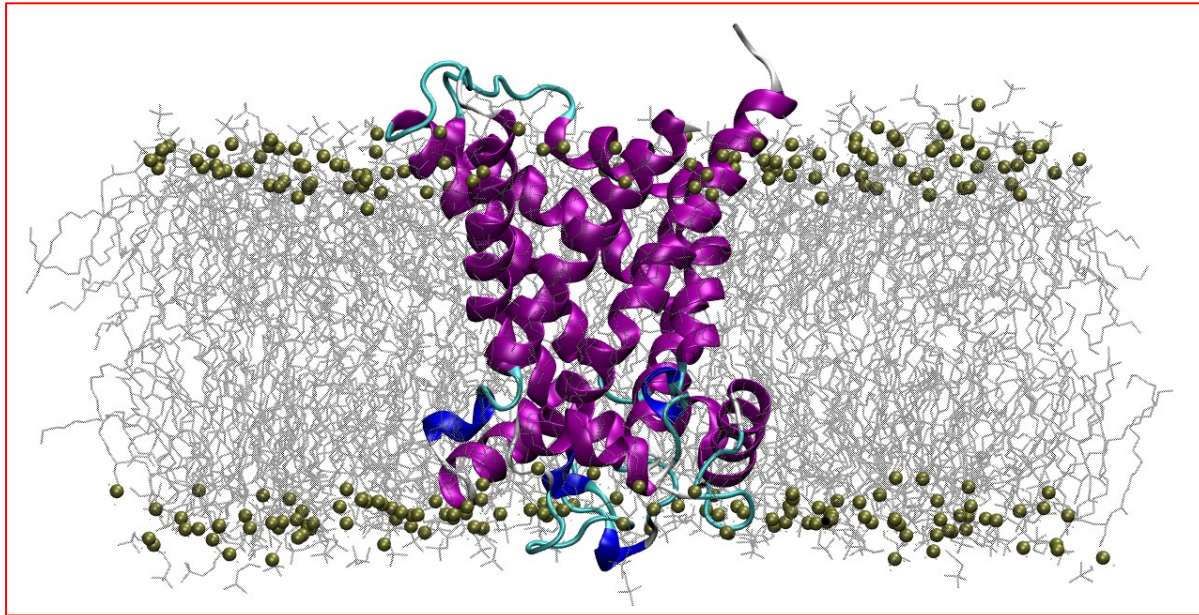
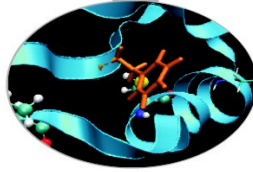


ATP/ADP Mitochondrial Carrier,
92K atoms
Gromacs 5.0.4 with GPU
PME for long electrostatics, 300 K,
Cut-off = 1 nm
Domain Decomposition

- Pure MPI (16 MPI procs) → 11.6 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) → 9.5 ns/day
- MPI/OpenMP/CUDA (2 MPI procs + 8 threads + 2 GPUs) → 24.6 ns/day
- MPI + Intel Phi (8 MPI procs + 34 threads) → 14.6 ns/day



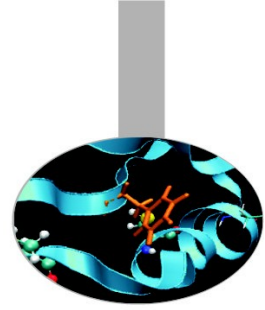
MD Performance on hybrid CPU-GPU clusters (Eurora)



ATP/ADP Mitochondrial Carrier,
92K atoms
Gromacs 5.0.4 with GPU
PME for long electrostatics, 300 K,
Cut-off = 1 nm
Domain Decomposition

- Pure MPI (16 MPI) → 11.6 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) → 9.5 ns/day
- MPI-CUDA (4 MPI procs + 2 GPUs) → 14.7 ns/day
- MPI-CUDA (8 MPI procs + 2 GPUs) → 22.2 ns/day
- MPI-CUDA (16 MPI + 2 GPUs) → 27.9 ns/day
- **MPI-CUDA (8 MPI procs + 2 OpenMP + 2 GPUs) → 29.2 ns/day**





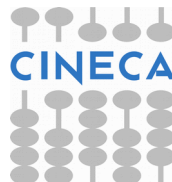
To address the bottleneck caused by multi-threading inefficiencies, it can be advantageous to reduce the number of OpenMP threads per rank. However, to not leave cores empty, this requires using more MPI ranks, hence more PP ranks, and therefore ranks will have to **share GPUs**. GPU sharing is possible by passing a GPU ID to mdrun multiple times, e.g -gpu_id 0011 will allow the first two PP ranks in a compute node to use GPU0 and the third and fourth GPU1.

Example #1:

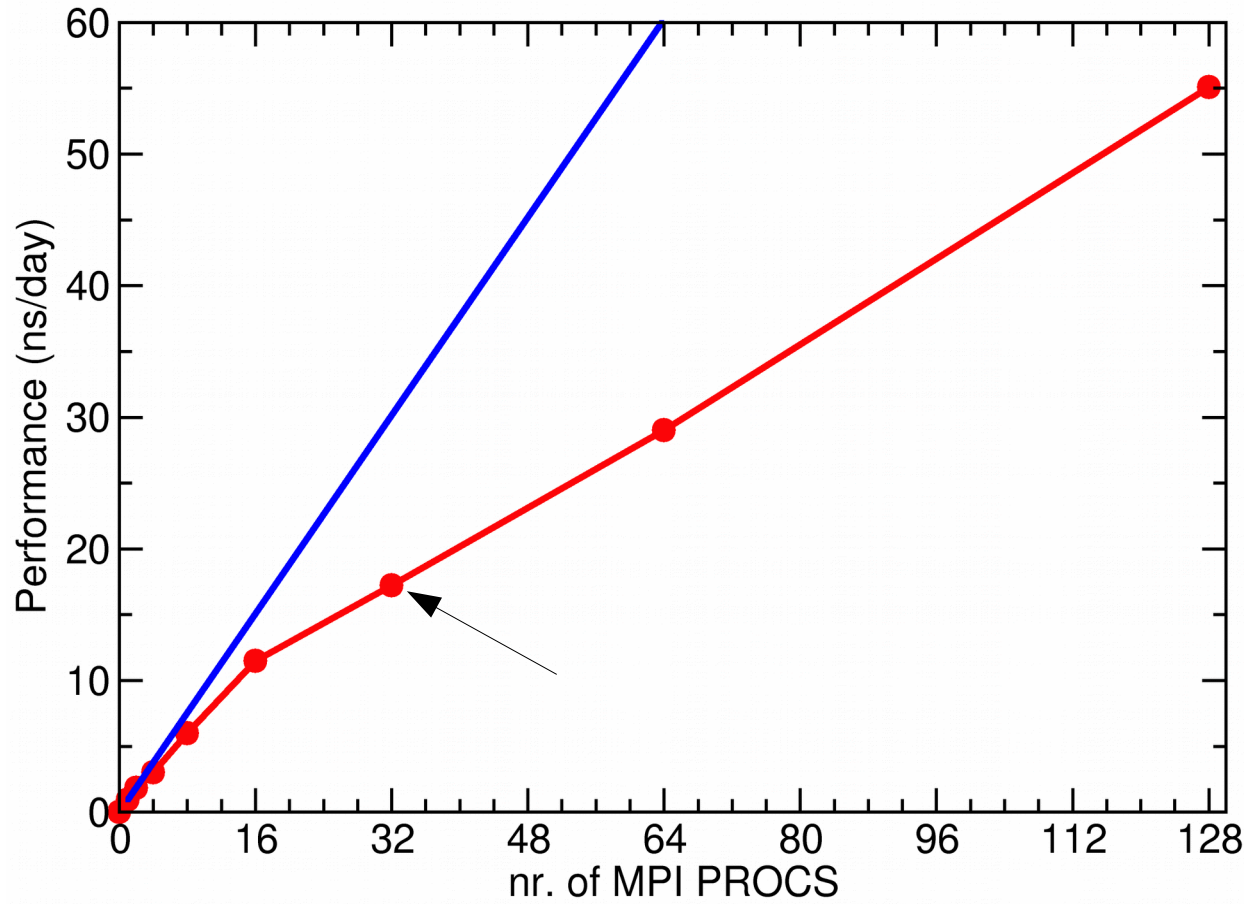
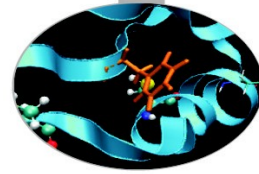
```
#PBS -l select=1:ncpus=8:mpiprocs=8:ngpus=2:mem=14GB
...
OMP_NUM_THREADS=1
...
mpirun -np 8 mdrun_mpi_cuda -s topol.tpr -maxh 1.0 -deffnm test -gpu_id 00001111
```

Example #2:

```
#PBS -l select=1:ncpus=16:mpiprocs=16:ngpus=2:mem=14GB
...
OMP_NUM_THREADS=1
...
mpirun -np 16 mdrun_mpi_cuda -s topol.tpr -maxh 1.0 -deffnm test -gpu_id 0000000011111111
```

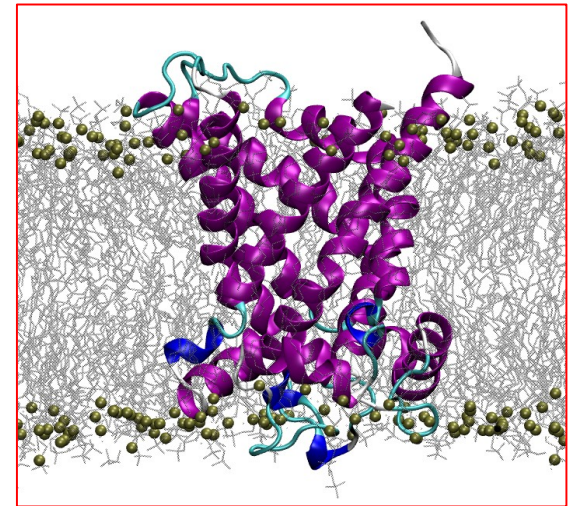


Speed up analysis pure MPI job

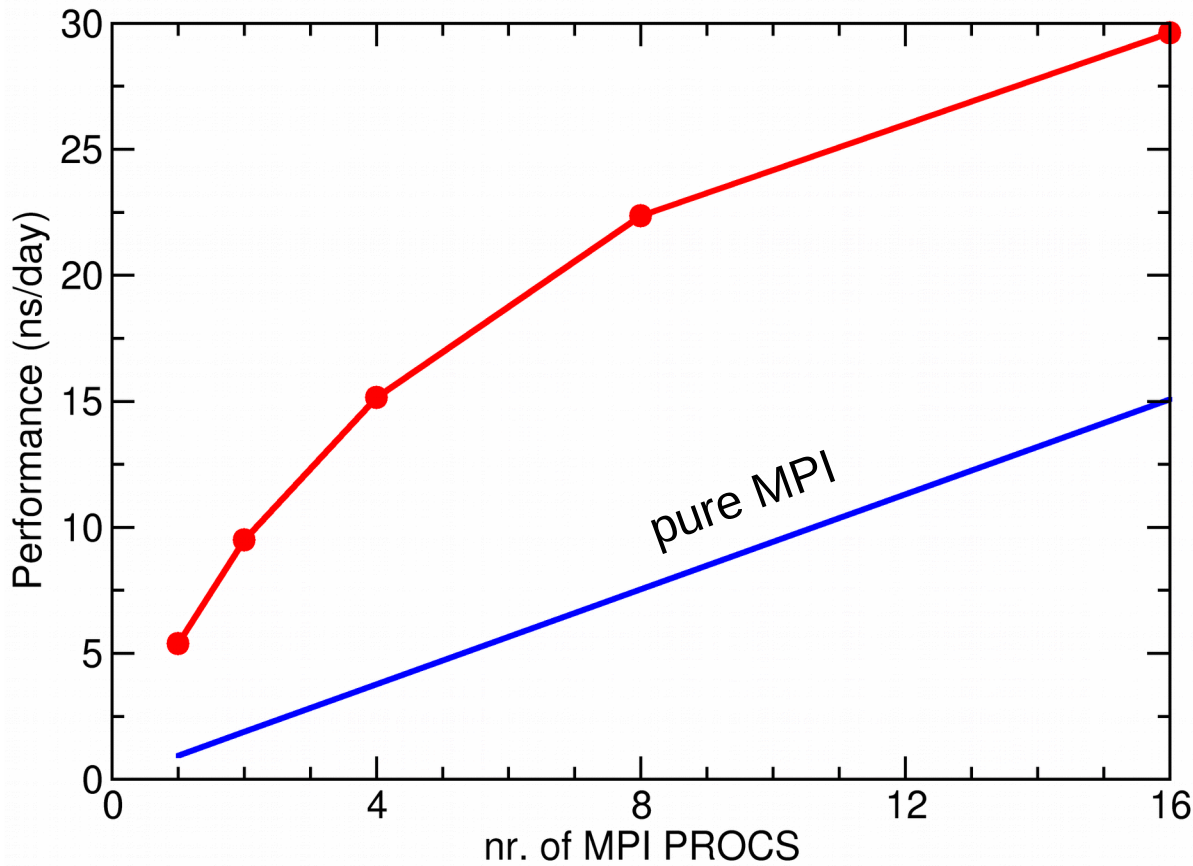
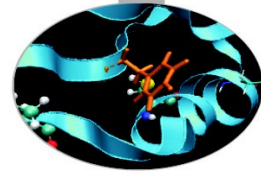


Parallel Efficiency with 32 MPI procs = 57.2 %

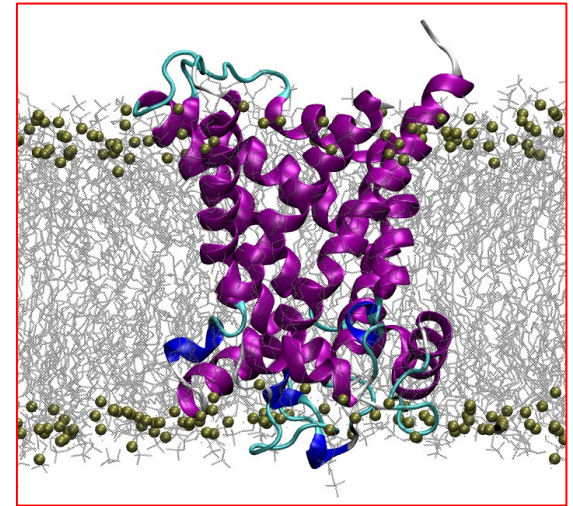
ATP-Carrier



MD Performance on hybrid CPU-GPU clusters (Eurora)

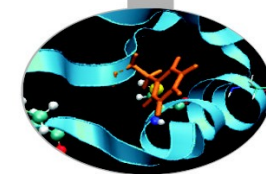


ATP-Carrier

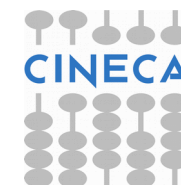


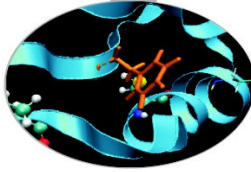
GPU acceleration in GROMACS

(Galileo)



	Small peptide (3K atoms)	Membrane protein (92K atoms)
Pure MPI	1	1
MPI-CUDA	1.2x	0.8x
multiple MPI ranks/CUDA	2.5x	2.5x
MPI-OpenMP/CUDA	2.0x	2.1x
Intel Phi	1.1x	1.3x

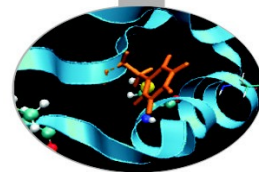


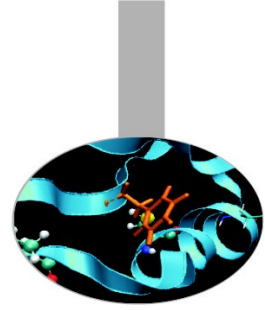


- Connect to Eurora: `ssh username@login.eurora.cineca.it`
- Password: **prog15ram**
- Copy gzipped tar file from here: `/gpfs/scratch/userinternal/agrottes/Corsi/November-2015/Tutorial2.tar.gz`
- Extract archive: `tar zxvf Tutorial2.tar.gz`
- Run jobs using the scripts provided and try optimizing the performance using multiple MPI ranks and GPUS.
-



Perform the calculations





Parallel Efficiency: $E_n = 100 \frac{P_n}{nP_1}$

P_n = performance at n cores (ns/day)

P_1 = performance for 1 core (ns/day)

Performance in ns/day:

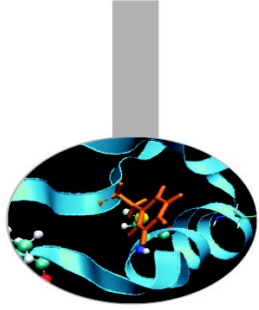
For walltime W (seconds) this is given by:

$$P = \text{no. of. time steps} * \text{time step (ns)} * 86400 / W$$

(86400 = seconds in 24h)

Time step is given in picoseconds (1ns = 10^3 ps).





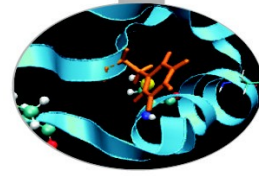
Example #3:

```
#PBS -l select=1:ncpus=8:mpiprocs=8:ngpus=2:mem=14GB
...
OMP_NUM_THREADS=1
...
mpirun -np 8 mdrun_mpi_cuda -s topol.tpr -maxh 1.0 -deffnm test -gpu_id 01010101
```

Example #4 (Galileo):

```
#PBS -l select=1:ncpus=16:mpiprocs=8:ngpus=2:mem=14GB
...
OMP_NUM_THREADS=2
...
mpirun -np 8 mdrun_mpi_cuda -s topol.tpr -maxh 1.0 -deffnm test -gpu_id 00001111
```





```
#!/bin/bash
#PBS -N gmx
#PBS -l select=1:ncpus=16:mpiprocs=16:ngpus=2:mem=14GB
#PBS -l walltime=1:00:00
#PBS -A train_cmd32015
#PBS -q R1660526
#PBS -W group_list=train_cmd32015
```

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

```
module load profile/advanced
module load cuda/6.5.14
module load autoload gromacs/5.0.4
```

```
export OMP_NUM_THREADS=1 ==> set nr. of OpenMP threads to 1 per node
# ==> 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 0000000011111111 "
mpirun -np 16 $cmd
```

