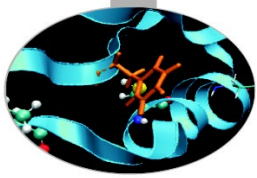
CINECA SCAI
SuperComputing Applications and Innovation

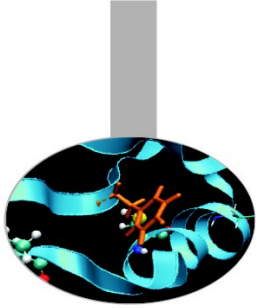
Tutorial 3:
Scalability tests for biolgical systems



Today you will learn:

- How to run benchmarks on your system
- How to build a scalability curve
- Compare and select best MPI/OpenMP ranks ratio for best performance

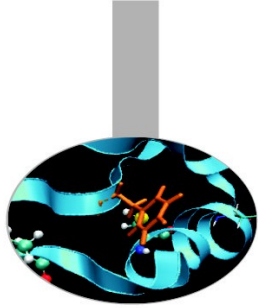
PRACE CINECA



- 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 with: tar zxvf Tutorial2.tar.gz
- Modify the number of CPUs and MPI procs used in the GMX-Pure-MPI.sh script:

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

How to compute speed-up plots



Parallel Efficiency: $E_n = 100 \frac{P_n}{n P_1}$

P_n = performance at n cores (ns/day)

P_1 = performance for 1 core (ns/day)

Performance in ns/day:

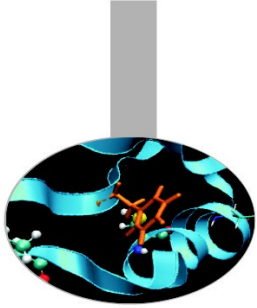
For walltime W (seconds) this is given by:

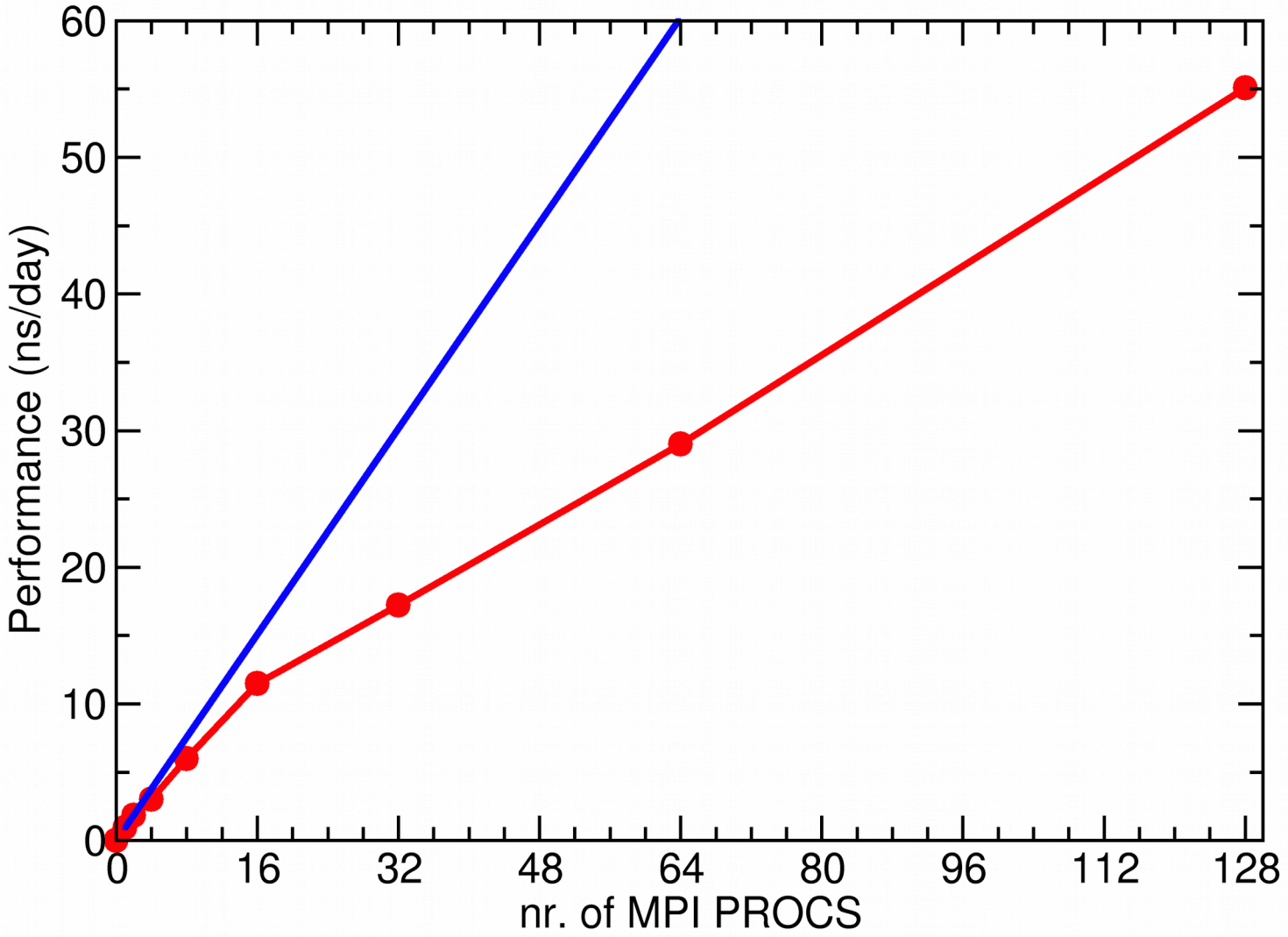
P = no. of. time steps * time step (ns) * 86400 / W

(86400 = seconds in 24h)

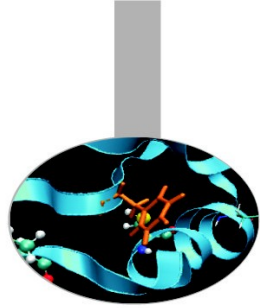
Time step is given in picoseconds (1ns = 10^3 ps).

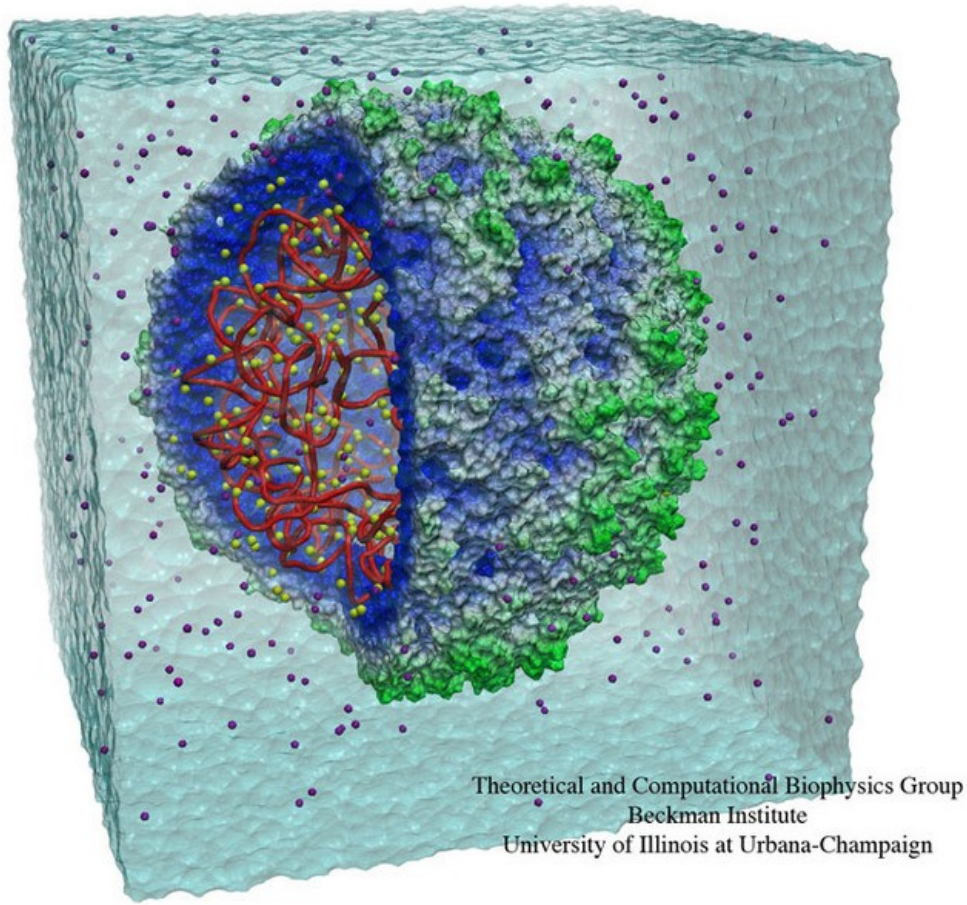
Results




Performance (ns/day)
nr. of MPI PROCS
0
10
20
30
40
50
60
0
16
32
48
64
80
96
112
128

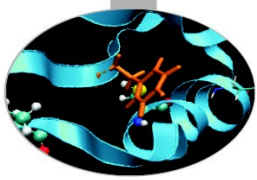
Satellite Tobacco Mosaic virus

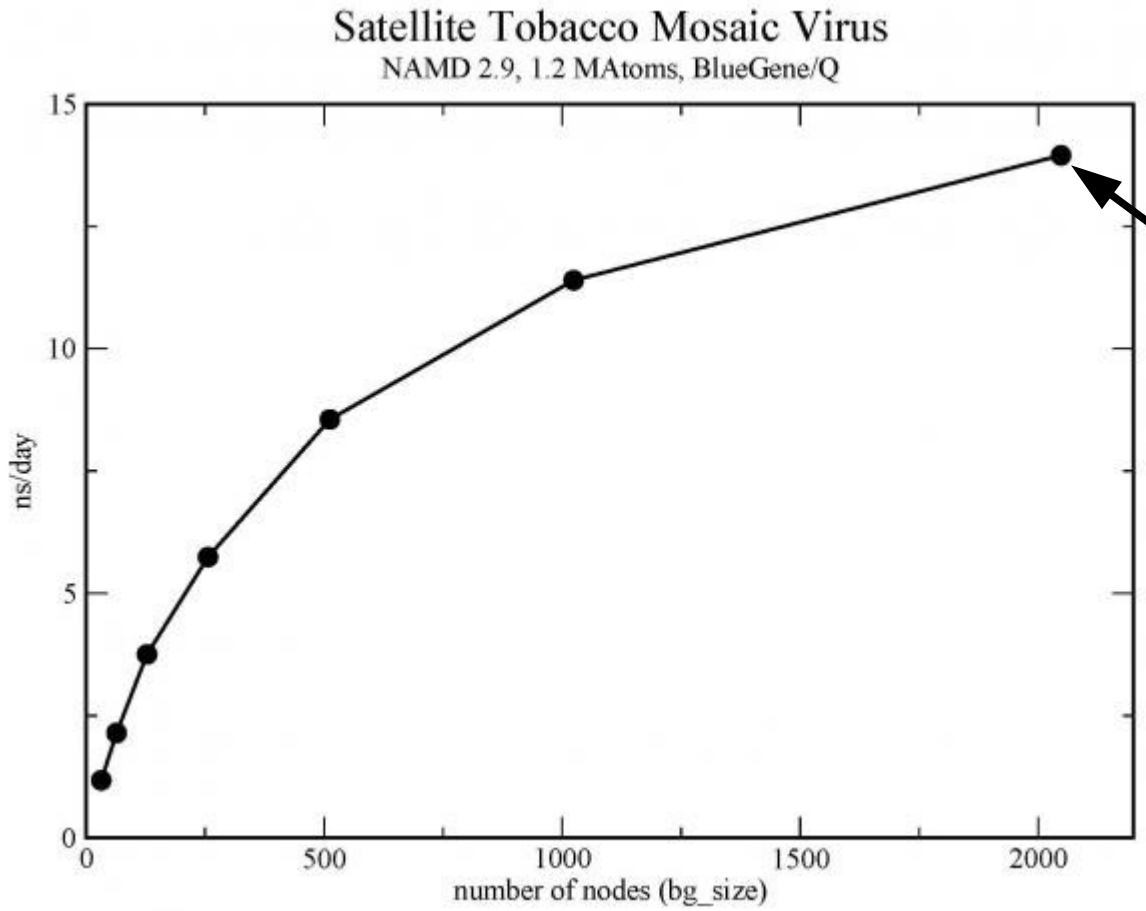




STMV (virus) benchmark (1,066,628 atoms, periodic, PME)

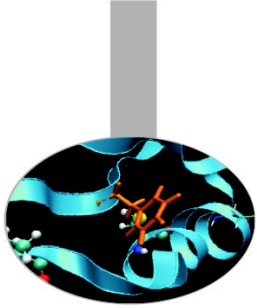
Fermi: Benchmarks and scalability

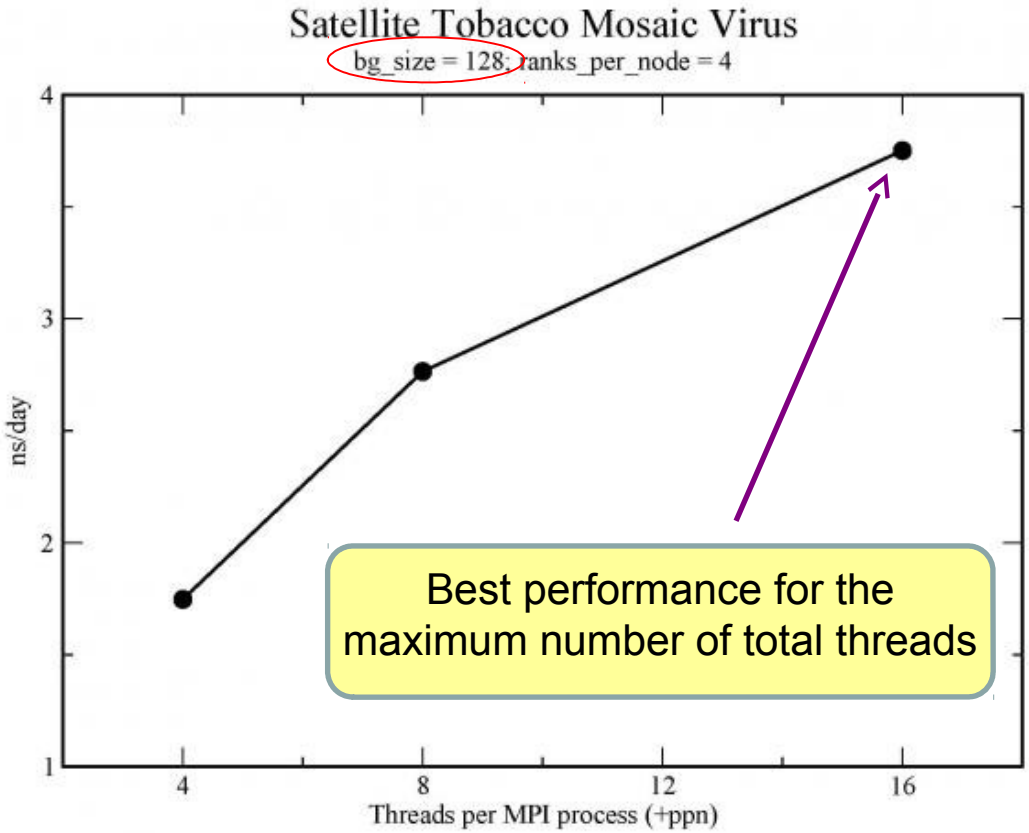




Increase of performance up to 32,768 cores! (bg_size=2048

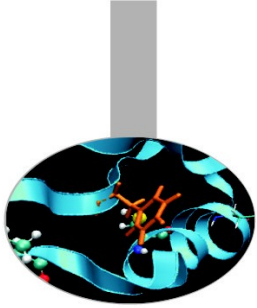
Fermi: Benchmarks and scalability



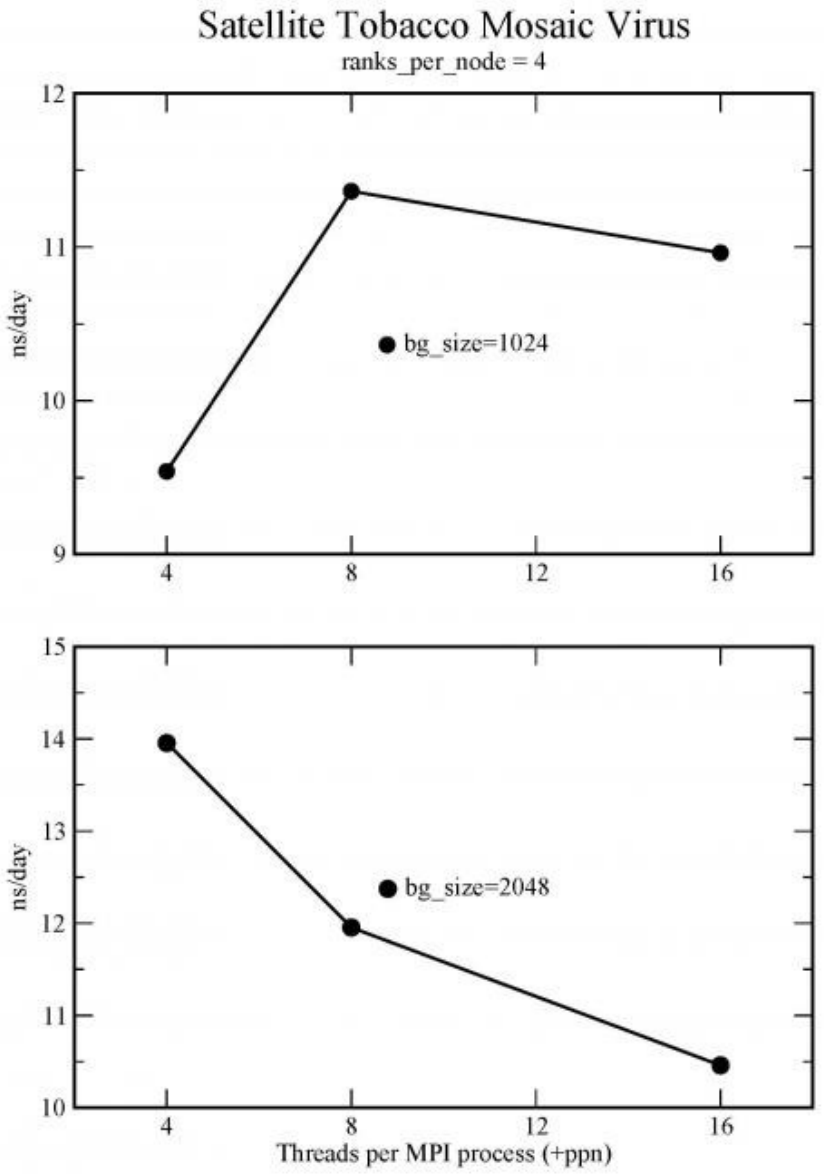



```
runjob --np 512 --ranks-per-node 4 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn4 > stmv.out
runjob --np 512 --ranks-per-node 4 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn8 > stmv.out
runjob --np 512 --ranks-per-node 4 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn16 > stmv.out
```

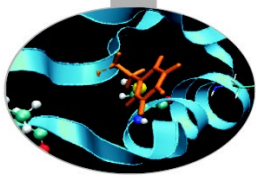
Fermi: Benchmarks and scalability

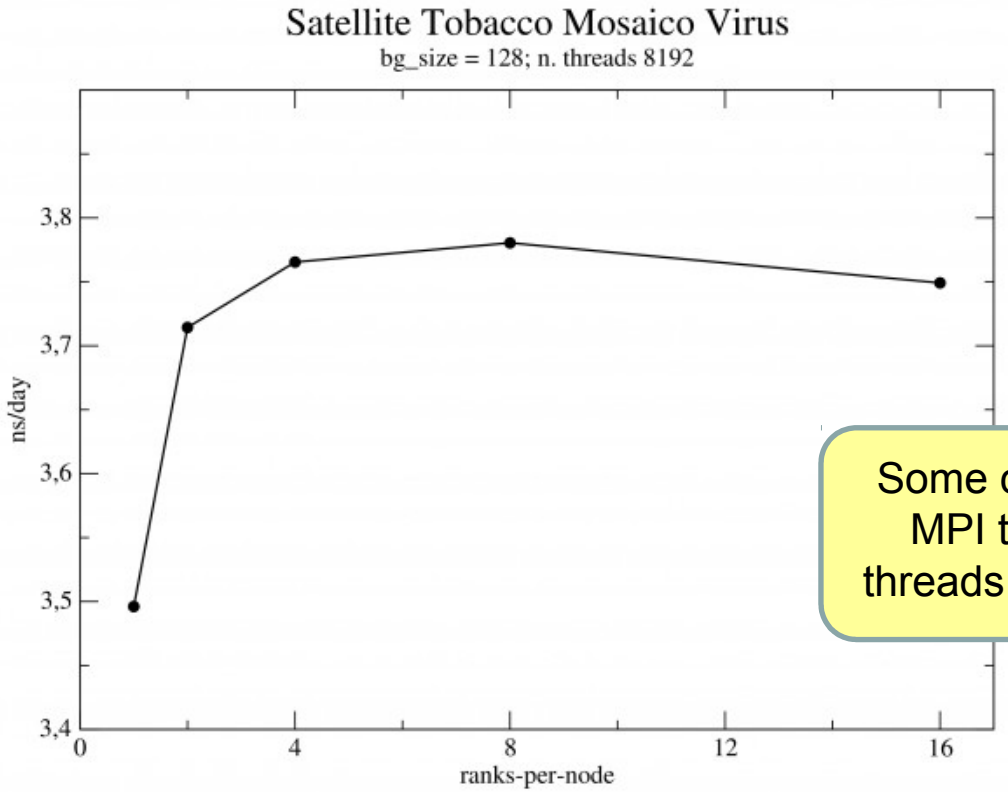


That is not true anymore when increasing bg_size: Check the performance on your system!



Fermi: Benchmarks and scalability

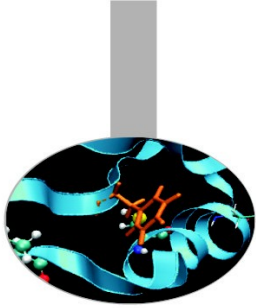




Some combinations of MPI tasks/OpenMP threads are less efficient

```
runjob --np 2048 --ranks-per-node 16 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn4 > stmv.out
runjob --np 1024 --ranks-per-node  8 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn8 > stmv.out
runjob --np  512 --ranks-per-node  4 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn16 > stmv.out
runjob --np  256 --ranks-per-node  2 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn32 > stmv.out
runjob --np  128 --ranks-per-node  1 :  $NAMD_HOME/namd2 stmv_ori.namd +ppn64 > stmv.out
```

Advanced MD Script for Gromacs (I)



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.

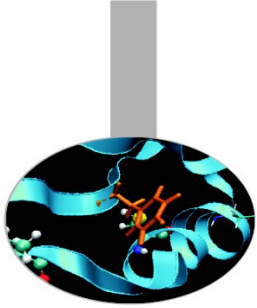
Example #1:

```
#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
```

Advanced MD Script for Gromacs (II)



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
```