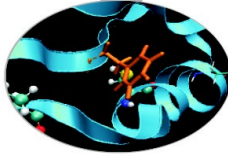
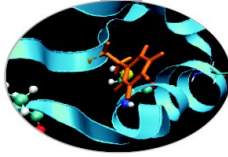


# This afternoon you will learn...



- How to set up a protein topology using GROMACS
- How to prepare your submission script
- How to submit your job to the PBS queueing system on PLX/eurora
- Tutorial:
  - Example: small peptide in solution
  - Domain vs. Particle Decomposition

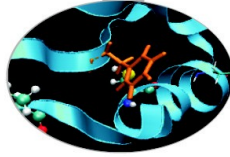
# How to become a CINECA user



- Please fill out the form on:  
<https://userdb.hpc.cineca.it/user/register>
- You'll receive userdb credentials: Then
  - Click on “HPC Access” and follow the on-screen instructions
  - You'll be asked to upload an image of a valid ID document
  - Ask your PI or send an email to [superc@cineca.it](mailto:superc@cineca.it) to be included on an active project.
- When everything is done an automatic procedure sends you (via 2 separate emails) the username/password to access PLX



# How to log in



- Establish a ssh connection

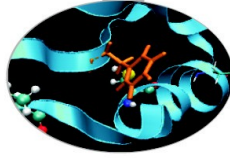
```
ssh <username>@login.plx.cineca.it  
ssh <username>@login.eurora.cineca.it
```

- Remarks:
  - **ssh** available on all linux distros
  - **Putty** (free) or **Tectia** ssh on Windows
  - *secure shell plugin* for **Google Chrome!**
  - important messages can be found in the *message of the day*

Check the **user guide!** <http://www.hpc.cineca.it/content/ibm-plx-gpu-user-guide-0>



# Launching jobs

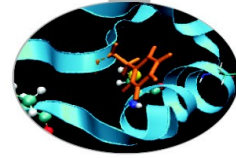


- Now that we have our executable, it's time to learn how to prepare a job for its execution
- PLX uses **PBS** scheduler.
- The job script scheme is:

```
#!/bin/bash  
#PBS keywords  
variables environment  
execution line
```



# Environment setup and execution line



The execution line starts with mpirun: Given: `./myexe arg_1 arg_2`

```
mpirun -n 24 ./myexe arg_1 arg_2
```

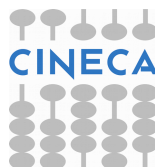
**-n** is the number of **cores** you want to use

**arg\_1 arg\_2** are the normal arguments of myexe

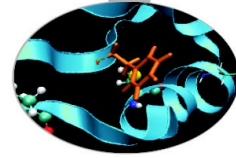
In order to use mpirun, **openmpi** (or **intelmpi**) has to be loaded. Also, if you linked dynamically, you have to remember to load every library module you need (automatically sets the LD\_LIBRARY\_PATH variable).

The environment setting usually starts with “**cd \$PBS\_O\_WORKDIR**”. That’s because by default you are launching on your home space the executable may not be found.

**\$PBS\_O\_WORKDIR** points to the directory from where you’re submitting the job .



# PBS keywords



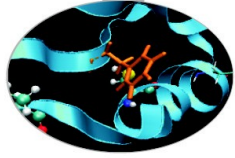
```
#PBS -N jobname
#PBS -o job.out
#PBS -e job.err
#PBS -l select=1:ncpus=12:mpiprocs=12:mem=ngpus=2
#PBS -l walltime=1:00:00
#PBS -q <queue>
#PBS -A <my_account>
```

```
# name of the job
# output file
# error file
# resources
# hh:mm:ss
# chosen queue
# name of the account
```

**select** = number of chunk requested  
**ncpus** = number of cpus per chunk requested  
**mpiprocs** = number of mpi tasks per chunk  
**mem** = RAM memory per chunk



# PLX job script template



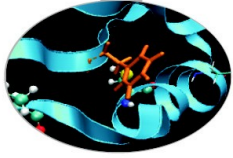
```
#!/bin/bash
#PBS -l walltime=1:00:00
#PBS -l select=1:ncpus=12:mpiprocs=12:ngpus=2:mem=47GB
#PBS -o job.out
#PBS -e job.err
#PBS -q parallel
#PBS -A <account_no>
#PBS -m mail_events ==> specify email notification
                        (a=aborted,b=begin,e=end,n=no_mail)
#PBS -M user@email.com

cd $PBS_O_WORKDIR
module load autoload openmpi
module load somelibrary

mpirun ./myprogram < myinput
```



# PBS commands



## qsub

```
qsub <job_script>
```

Your job will be submitted to the PBS scheduler and executed when there will be nodes available (according to your priority and the queue you requested)

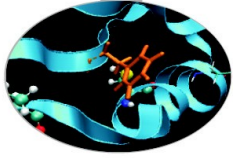
## qstat

```
qstat
```

Shows the list of all your scheduled jobs, along with their status (idle, running, closing, ...) Also, shows you the job id required for other qstat options



# PBS commands



## qstat

```
qstat -f <job_id>
```

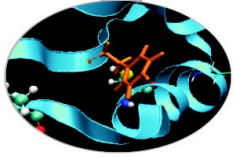
Provides a long list of informations for the job requested.

In particular, if your job isn't running yet, you'll be notified about its estimated start time or, if you made an error on the job script, you will learn that the job won't ever start

## qdel

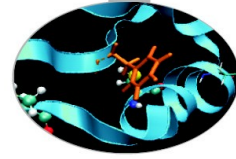
```
qdel <job_id>
```

Removes the job from the scheduled jobs by killing it



# Scripts for running MD codes on PLX/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=16:mem=14GB  
#PBS -q <queue>  
#PBS -l walltime=1:00:00  
#PBS -A <account_nr>
```

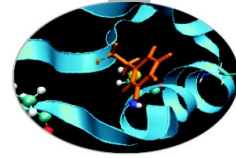
```
cd $PBS_O_WORKDIR ==> change to current dir
```

```
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
```

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





```
>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=2
```

==> set nr. Of OpenMP threads to 2

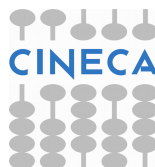
```
#
```

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

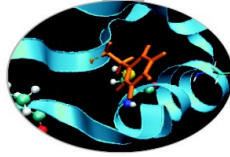
```
mdrun=$(which mdrun_mpi_cuda)
```

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

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



# NAMD 2.10 MPI+CUDA on PLX



```
>module load autoload namd/2.10  
>module help namd/2.10
```

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

```
#PBS -l select=1:ncpus=12:mpiprocs=12:ngpus=2
```

```
#PBS -l walltime=0:30:00
```

```
#PBS -o namd.out
```

```
#PBS -e namd.err
```

```
#PBS -l place=free:excl
```

==> (the PBS directive place=free:excl to reserve the whole node)

```
#PBS -A <account_no>
```

```
#PBS -q <queue>
```

```
cd $PBS_O_WORKDIR
```

==> change to current dir

```
module load profile/advanced
```

```
module load autoload namd/2.10
```

```
namd=$(which namd2_cuda)
```

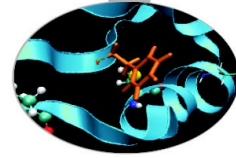
==> set path to namd executable

```
mpirun $namd +idlepoll md.namd
```

==> run CUDA version of NAMD



# Amber-12 on PLX (pure MPI version)



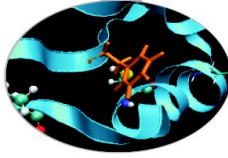
```
>module load autoload amber/12  
>module help amber/12
```

```
#!/bin/bash  
#PBS -A <account_no>  
#PBS -N amber  
#PBS -l walltime=1:00:00  
#PBS -l select=1:ncpus=12:mpiprocs=12:ncpus=12  
#PBS -o job.out  
#PBS -q <queue>
```

```
cd $PBS_O_WORKDIR ==> change to current dir  
module load autoload amber/12
```

```
cmd="pmemd.MPI -O -i mdin -o mdout -p prmtop -c inpcrd -r restrt -x mdcrd"  
mpirun -np 12 $cmd
```

# Amber 12 – MPI+CUDA version



```
#!/bin/bash
#PBS -A <account_no>
#PBS -l select=1:ncpus=12:mpiprocs=2:ngpus=2
#PBS -l walltime=1:00:00
#PBS -o job.out
#PBS -q longpar
```

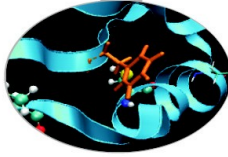
```
cd $PBS_O_WORKDIR
module load autoload amber/12
```

# for best performance use 1 mpi task/1 gpu. In this example we have 1\*2 gpus = 2 MPI tasks.

```
cmd="pmemd.cuda.MPI -O -i mdin -o mdout -p prmtop -c inpcrd -r restrt -x mdcrd"
```

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

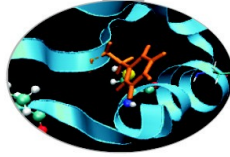




- Copy gzipped file from here:  
[https://hpc-forge.cineca.it/files/CoursesDev/public/2014/Molecular\\_dynamics\\_in\\_an\\_HPC\\_environment/Bologna/Tutorial2.tar.gz](https://hpc-forge.cineca.it/files/CoursesDev/public/2014/Molecular_dynamics_in_an_HPC_environment/Bologna/Tutorial2.tar.gz)  
Alternatively here on Eurora:  
</gpfs/scratch/userinternal/agrottes/Corso-PATC/Tutorial2.tar.gz>
- Generate topol.top using starting file: Peptide.pdb
- Minimize peptide structure using file steep.mdp
- Run a small run (10000 steps) using file pure MPI, MPI+CUDA and MPI\_OpenMP+CUDA
- Analyze runs using Gromacs log files and determine speed up curves
- Optional: domain vs. particle decomposition: compare performances



# Tutorial 2: small peptide in water



Run pure MPI jobs on PLX:

- queue = private
- add PBS keyword: #PBS -W group\_list=train\_cnov2014

Run MPI-CUDA and MPI-OpenMP\_CUDA jobs on Eurora:

- queue = R1422507
- add PBS keyword: #PBS -W group\_list=train\_cnov2014

