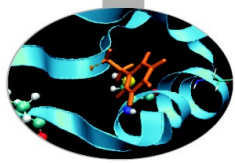
CINECA SCAI
SuperComputing Applications and Innovation

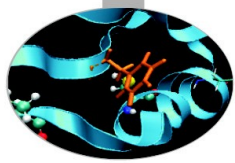


Introduzione ambiente unix: moduli, job scripts, PBS

A. Grottesi (CINECA)

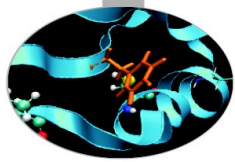
PRACE CINECA

This afternoon you will learn...



- Basic commands for UNIX environment @ CINECA
- How to submitt your job to the PBS queueing system on Eurora
- Tutorial #1:
  - Compare scripts for pure MPI, OpenMP-MPI and MPI-GPU gromacs

How to become a CINECA user



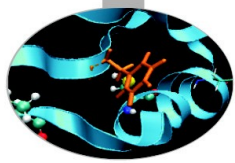
- Please fill out the form on:

  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 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 HPC systems

How to log in



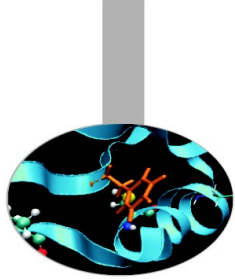
- Establish a ssh connection

  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/documentation

Storage and Filesystem



$HOME:

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

$CINECA_SCRATCH:

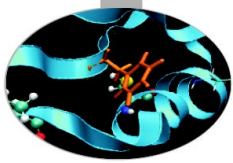
- Large, parallel filesystem (GPFS).
- Temporary (files older than 30 days automatically deleted), no backup.
- No quota. Run your simulations and calculations here.

$WORK:

- Permanent, backed-up, project specific

More info:
http://www.hpc.cineca.it/content/data-storage-and-filesystems-0

Accounting: saldo



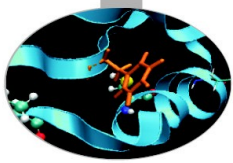
```
[mcestari@node342] (~)
$ saldo -b

--------------------------------------------------------------------------------------------------
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

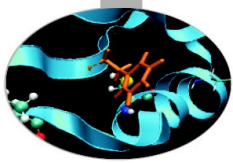
In the CINECA system it is possible to have more than 1 budget (“account”) from which you can use time. The accounts available to your UNIX username can be found from the `saldo` command.

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, ...)

Modules

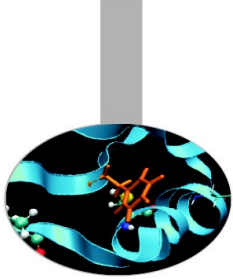


- CINECA' s work environment is organized in modules, a set of installed libs, tools and applications available for all users.

- "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 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

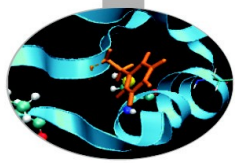
> module (un)load <module_name>
(Un)loads a specific module

> 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

Launching jobs



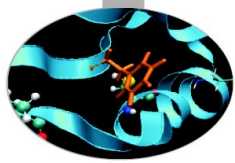
- Now that we have our executable, it's time to learn how to prepare a job for its execution

- Eurora, Pico and Galileo have the **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

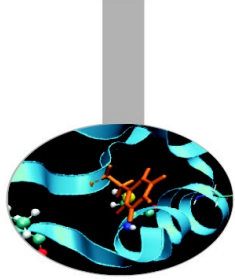
arg_1 arg_2 are the normal arguments of myexe

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                                          # name of the job
#PBS -o job.out                                          # output file
#PBS -e job.err                                          # error file
#PBS -l select=1:ncpus=16:mpiprocs=16:mem=ngpus=2        # resources
#PBS -l walltime=1:00:00                                 # hh:mm:ss
#PBS -q <queue>                                          # chosen queue
#PBS -A <my_account>                                     # name of the account
```

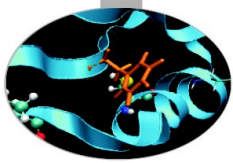
select = number of chunk requested

ncpus = number of cpus per chunk requested

mpiprocs = number of mpi tasks per node/chunk

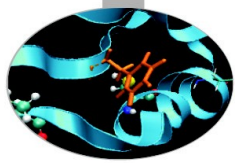
mem = RAM memory per chunk

Eurora job script template



```
#!/bin/bash
#PBS -l walltime=1:00:00
#PBS -l select=1:ncpus=16:mpiprocs=16:ngpus=2:mem=14GB
#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>

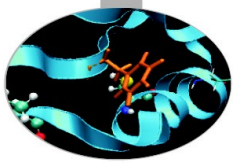
Your job will be submitted to the PBS scheduler and executed when there will be nodes available (according to your priority and the queue you requested)

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>

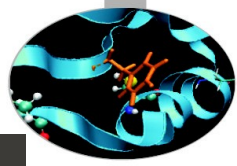
Provides a long list of informations for the job requested.
In particular, if your job isn't running yet, you'll be notified about its estimated start time or, if you made an error on the job script, you will learn that the job won't ever start

qdel

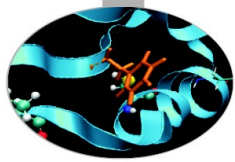
qdel <job_id>

Removes the job from the scheduled jobs by killing it

PBS commands: qstat

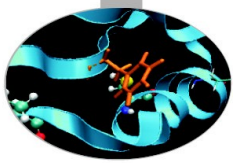


```
File  Edit  View  Search  Terminal  Help

node129:
                                                                  Est
                                                        Req'd  Req'd   Start
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
451912.node129  amessina parallel abq_parall     --   4  32    4gb 00:10 H    --
1587463.node129 aemerson parallel qsub.job       --  12 192  120gb 00:30 H    --
1587543.node129 aemerson parallel qsub.job       --   9 144   90gb 00:30 H    --
1596264.node129 ggrazios parallel a7_nor_wat 113134   1  16   14gb 04:00 R    --
1596269.node129 dborello parallel sub.sh      26304  16  64   16gb 04:00 R    --
1596279.node129 adimasci parallel Xnavis95_c  83686   1   1    2gb 04:00 R    --
1596277.node129 adimasci parallel Xnavis95_c  87830   1   1    2gb 04:00 R    --
1596278.node129 adimasci parallel Xnavis95_c 101923   1   1    2gb 04:00 R    --
1596287.node129 gagate00 parallel esegui_8SP  19442   1  16   14gb 04:00 R    --
1596304.node129 mrizzini parallel ipcOptimal  90425   1  12    9gb 04:00 R    --
1596305.node129 mrizzini parallel ipcOptimal  72089   1  12    9gb 04:00 R    --
1596306.node129 mrizzini parallel ipcOptimal  71042   1  12    9gb 04:00 R    --
1596307.node129 mrizzini parallel ipcOptimal   8235   1  12    9gb 04:00 R    --
1596308.node129 mrizzini parallel ipcOptimal 102900   1  12    9gb 04:00 R    --
1596309.node129 mrizzini parallel ipcOptimal  42979   1  12    9gb 04:00 R    --
1596310.node129 mrizzini parallel ipcOptimal  92927   1  12    9gb 04:00 R    --
1596311.node129 mrizzini parallel ipcOptimal  90698   1  12    9gb 04:00 R    --
1596290.node129 gagate00 parallel esegui_4SP  78531   1  16   14gb 04:00 R    --
1596291.node129 gagate00 parallel esegui_4SP  37027   1  16   14gb 04:00 R    --
1596292.node129 gagate00 parallel esegui_2SP  78795   1  16   14gb 04:00 R    --
1596312.node129 mrizzini parallel ipcOptimal 105767   1  12    9gb 04:00 R    --
1596313.node129 mrizzini parallel ipcOptimal  87469   1  12    9gb 04:00 R    --
[agrottes@node129 ~]$
```



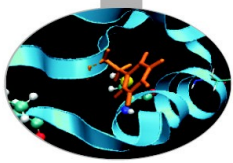
Scripts for running MD codes on PLX/Eurora

Gromacs 4.6.5, pure MPI on Eurora



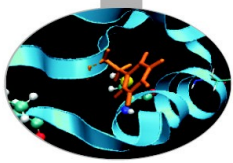
```
>module load autoload gromacs/4.6.5
>module help gromacs/4.6.5

#!/bin/bash
#PBS -N gmx
#PBS -l select=1:ncpus=16:mpiprocs=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 to1

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

Gromacs 4.6.5 MPI+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=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
```

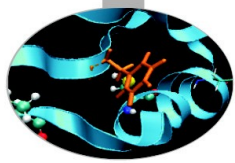
Tutorial 1: running MD code @ CINECA



- Copy gzipped file from here:
  - https://hpc-forge.cineca.it/files/CoursesDev/public/2015/High_Performance_Molecular_Dynamics/Rome/February/Tutorial1.tar.gz
- Or, alternativelly, copy from here:

  /gpfs/scratch/userinternal/agrottes/Corsi/Tutorial1.tar.gz
- Run a small run (10000 steps) using file pure MPI, MPI+CUDA and MPI_OpenMP+CUDA

Tutorial 1



Run MPI-CUDA and MPI-OpenMP_CUDA jobs on Eurora:

- queue = R1598562
- add PBS keyword: #PBS -A train_cmd12015