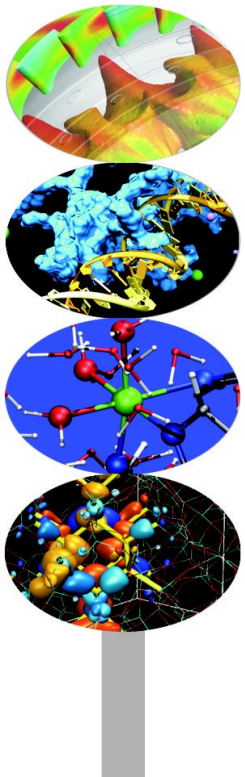
CINECA SCAI
SuperComputing Applications and Innovation

CINECA COURSES
High Performance Computing 2014

Production enviroment on PLX



Giorgio Amati – SCAI/CINECA

HPC enabling OpenFOAM for CFD applications

26-27-28 March 2014

g.amati@cineca.it

PRACE

CINECA

About this lecture

- Simply to introduce the very basic of PLX environment in order to lunch jobs and complete the exercises succesfully.

- If you are a CINECA user you already knows everything but it is important for all non-CINECA users...

It will lasts only half an hour...so be patient

About this lecture /2

- PLX
- Environment & modules
- How to submit jobs

PLX

- **Architecture**: Linux Infiniband Cluster
- **Processor**: Intel Xeon (6-Core Westmere)@2.4 GHz
- **Number of processors (cores)**: 3288
- **Number of nodes**: 274 (12 cores per node)
- **RAM**: 14 TB (4 GB/core)
- **Interconnection network**: Infiniband
- **Number of GPUs**: 548 (2 per node)
- **Operative system**: Linux
- **Peak performance**: 32 TFlop/s (CPU)
  565 TFlop/s (GPU)
- **Compilers**: Fortran, C, C++
- **Parallel libraries**: MPI,OpenMP

Username & Work Eenvironment

user → a08traXX

password → cSlgffigc

ssh user@login2.plx.cineca.it

- Once you're logged on a cluster, you are on your **home** space. It is best suited for programming environment (compilation, small debugging sessions…). You can acces via environment variable: **$HOME**
- Another space you can access to is your **scratch** space. It is best suited for **production** environment. You can access via environment variable: **$CINECA_SCRATCH**
- WARNING: is active a **cleaning procedure**, that deletes your files older than 30 days!
- Use command "**cindata**" for a quick briefing about your space occupancy

Accounting

As any user, you have access to a limited number of CPU hours to spend. They are not assigned to users, but to **projects** and are shared between the users who are working on the same project (i.e. your research partners). Such projects are called **accounts** and are a different concept from your username.

You can check the state of your account with the command "**saldo –b**", which tells you how many CPU hours you have already consumed for each account you're assigned at

(a more detailed report is provided by "**saldo –r**").

```
-bash-3.2$ saldo -b
--------------------------------------------------------------------------------------------
account               start        end         total       localCluster   totConsumed     totConsumed
                                               (local h)   Consumed(local h)    (local h)              %
--------------------------------------------------------------------------------------------
train_cnl2014       20140310    20140314          2000                  0            0             0.0
train_copf2014      20140319    20140414         20000                  0            0             0.0
```

Accounting

The account provided for this course is “**train_copf2014**” (you have to specify it on your job scripts). It expires in two weeks and is shared between all the students.

Module

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

"loading" a module means defining all the environment variables that point to the path of what you have loaded.

After a module is loaded, an environment variable is set of the form "MODULENAME_HOME"

```
-bash-3.2$ module load autoload openfoam/2.3.0-gnu-4.7.2
### auto-loading modules gnu/4.7.2 openmpi/1.6.3--gnu--4.7.2
### auto-loading modules gnu/4.7.2 openmpi/1.6.3--gnu--4.7.2
-bash-3.2$ echo $FOAM_SRC
/cineca/prod/applications/openfoam/2.3.0-gnu-4.7.2/openmpi--1.6.3--gnu--4.7.2/OpenFOAM-2.3.0/src
-bash-3.2$
```

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 load <module_name>

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 list (or just “module li”)
List of all loaded modules

module purge
Remove all loaded modules

Here you have simply to type:
module load autoload openfoam/2.3.0-gnu-4.7.2

Module

(partial) output for module show

```
giorgio@giorgio-Latitude-E6330: ~/SCRATCH/RUN_2D
module-whatis   The OpenFOAM® (Open Field Operation and Manipulation) CFD Toolbox can simulate anything from complex fluid flows involving
 chemical reactions, turbulence and heat transfer, to solid dynamics, electromagnetics and the pricing of financial options.
conflict        openfoam
setenv  OPENFOAM_HOME   /cineca/prod/applications/openfoam/2.3.0-gnu-4.7.2/gnu--4.7.2
setenv  FOAM_CINECA_SCRIPT      /cineca/prod/applications/openfoam/2.3.0-gnu-4.7.2/openmpi--1.6.3--gnu--4.7.2/cineca_script
...
setenv  MPI_BUFFER_SIZE 20000000
setenv  WM_ARCH linux64
setenv  WM_ARCH_OPTION  64
setenv  WM_CC   gcc
setenv  WM_CFLAGS       -m64 -fPIC
setenv  WM_COMPILE_OPTION       Opt
setenv  WM_COMPILER     Gcc
setenv  WM_COMPILER_LIB_ARCH    64
setenv  WM_CXXFLAGS     -m64 -fPIC
setenv  WM_CXX  g++
setenv  WM_DIR  /cineca/prod/applications/openfoam/2.3.0-gnu-4.7.2/openmpi--1.6.3--gnu--4.7.2/OpenFOAM-2.3.0/wmake
setenv  WM_LDFLAGS      -m64
setenv  WM_LINK_LANGUAGE        c++
setenv  WM_MPLIB        SYSTEMOPENMPI
setenv  WM_OPTIONS      linux64GccDPOpt
setenv  WM_OSTYPE       POSIX
setenv  WM_PRECISION_OPTION     DP
setenv  WM_PROJECT_DIR  /cineca/prod/applications/openfoam/2.3.0-gnu-4.7.2/openmpi--1.6.3--gnu--4.7.2/OpenFOAM-2.3.0
setenv  WM_PROJECT_INST_DIR     /cineca/prod/applications/openfoam/2.3.0-gnu-4.7.2/openmpi--1.6.3--gnu--4.7.2
setenv  WM_PROJECT      OpenFOAM
setenv  WM_PROJECT_USER_DIR     /plx/usertrain/a08tra69/OpenFOAM/a08tra69-2.3.0
setenv  WM_PROJECT_VERSION      2.3.0
..
"qqqqq" 56L, 4169C written                                                                    31,3          10%
```

Launching jobs

PLX uses a scheduler called **PBS**.

The job script scheme is:

- **#!/bin/bash**
- **PBS keywords**
- **variables environment**
- **execution line**

“standard” 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=8:mpiprocs=2      #resources requested *
#PBS -l walltime=1:00:00                 #max 24h, depending on the queue
#PBS -q parallel                         #queue desired
#PBS -A <my_account>                     #name of the account
```

* select = number of chunks (not exactly the nodes) requested

ncpus = number of cpus per chunk requested

mpiprocs = number of mpi tasks per chunk

for pure MPI jobs, ncpus = mpiprocs. For OpenMP jobs, mpiprocs < ncpus

Specific keyworkds for the course

```
#PBS -A train_copf2014                  # your account name
#PBS -q private                         # special queue reserved for you
#PBS -W group_list=train_copf2014       # needed for entering in private queue
```

"**private**" queue is a particular queue composed by 8 nodes reserved for internal staff and course students.

"**private**" nodes have only 8 cores per node.

In order to grant fast runs to all the students, we ask you to not launch too big jobs (you won't need them, anyways). Please don't request more than 1 node at a time!

Enviroment setup & execution line

```
module purge
module load <module to load>
cd $PBS_O_WORKDIR
...
your command
….
```

$PBS_O_WORKDIR points at the folder you're submitting the job from.

That's because with PBS you start at your home space ($HOME) and not the point you are submitting from.

PLX Job Script Example

```
#!/bin/bash
#PBS -l walltime=0:10:00
#PBS -l select=1:ncpus=4:mpiprocs=4
#PBS -o job.out
#PBS -e job.err
#PBS -q private
#PBS -A train_copf2014
#PBS -W group_list=train_copf2014

....
```

PBS commands/1

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

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

PBS commands/2

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 <job_id>

Removes the job from the scheduler, killing it

JOB Classes

After the end of the course, you won't be able to use the private queue anymore: how can you launch jobs then?

You have to modify your jobscript by changing the "**#PBS –q private**" keyword with something else: you will be able to submit your jobs, but as a regular user (so expect long waiting times)

The queue you're going into is the one you ask (it has to be specified!):

debug:	max nodes= 2,	wall_clock_time <= 00:30:00
parallel:	max nodes= 44,	wall_clock_time <= 06:00:00
longpar:	max nodes= 22,	wall_clock_time <= 24:00:00

PS: You don't need the "#PBS –W ..." keyword anymore

Useful documentation

Check out the User Guides on our website www.hpc.cineca.it

PLX:

http://www.hpc.cineca.it/content/ibm-plx-gpu-user-guide-0

http://www.hpc.cineca.it/content/batch-scheduler-pbs-0