Cineca
TRAINING
High Performance
Computing 2017

Introduction to GALILEO

Parallel & production environment

Mirko Cestari - m.cestari@cineca.it
Alessandro Marani - a.marani@cineca.it
Domenico Guida - d.guida@cineca.it
Maurizio Cremonesi - m.cremonesi@cineca.it
SuperComputing Applications and Innovation Department

CINECA

June 6, 2017

GOALS

You will learn:

- basic concepts of the system architecture that directly affects your work during the school
- how to explore and interact with the software installed on the system
- how to launch a simulation exploiting the computing resources provided by the GALILEO system

GALILEO CHARACTERISTICS:

Model: IBM NeXtScale

Architecture: IBM NeXtScale

Processor type: Intel Xeon Haswell@ 2.4 GHz

Computing Nodes: 516

Each node: 16 cores, 128 GB of RAM

Computing Cores: 8.256

RAM: 66 TByte

Peak Performance: 1.2 PFlops

Internal Network: Infiniband 4xQDR switches (40 Gb/s)

Accelerators: 768 Intel Phi 7120p (2 per node on 384 nodes)

+ 80 Nvidia K80 (2 per node on 40 nodes)

X86 based system for production of medium scalability applications

HOW TO LOG IN

- Establish a ssh connection

```
ssh a08trbnn@login.galileo.cineca.it
```

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

- Check the **user guide**!

https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.2%3A+GALILEO+UserGuide

WORK ENVIRONMENT

$HOME:

Permanent, backed-up, and local to GALILEO.

50 Gb of quota. For source code or important input files.

$CINECA_SCRATCH:

Large, parallel filesystem (GPFS).

No quota. Run your simulations and calculations here.

A cleaning procedure automatically deletes every file untouched since 50 days

$WORK:

Similar to $CINECA_SCRATCH, but the content is shared among all the users of the same account.

1 Tb of quota. **Stick to $CINECA_SCRATCH for the school exercises!**

use the command cindata to get info on your disk occupation

https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.4%3A+Data+storage+and+FileSystem

OUTLINE

- A first step:
  - System overview
  - Login
  - Work environment
- **Production environment**
  - **Our first job!!**
  - **Creating a job script**
  - **Accounting and queue system**
  - **PBS commands**
- Programming environment
  - Module system
  - Serial and parallel compilation
  - Interactive session
- For further info...
  - Useful links and documentation

LAUNCHING JOBS

As in every HPC cluster, GALILEO allows you to run your simulations by submitting **“jobs”** to the compute nodes

Your job is then taken in consideration by a **scheduler**, that adds it to a queuing line and allows its execution when the resources required are available

The operative scheduler in GALILEO is **PBS**

PBS JOB SCRIPT SCHEME

The scheme of a PBS job script is as follows:

```
#!/bin/bash

#PBS keywords

variables environment

execution line
```

PBS JOB SCRIPT EXAMPLE

```
#!/bin/bash
#PBS -N myname
#PBS -o job.out
#PBS -e job.err
#PBS -m abe
#PBS -M user@email.com
#PBS -l walltime=00:30:00
#PBS -l select=1:ncpus=16:mpiprocs=8:mem=120GB
#PBS -A train_cmpD2017
#
echo “I’m working on GALILEO!”
date
exit
```

PBS KEYWORD ANALYSIS - 1

#PBS -N myname

Defines the name of your job

#PBS -o job.out

Specifies the file where the standard output is directed (default=jobname.o<jobID>)

#PBS -e job.err

Specifies the file where the standard error is directed (default=jobname.e<jobID>)

PBS KEYWORD ANALYSIS - 1

#PBS -m abe (optional)

Specifies e-mail notification. An e-mail will be sent to you when something happens to your job, according to the keywords you specified (a=aborted, b=begin, e=end, n=no email)

#PBS -M user@email.com (optional)

Specifies the e-mail address for the keyword above

PBS KEYWORD ANALYSIS - 2

#PBS -l walltime=00:30:00

Specifies the maximum duration of the job. The maximum time allowed depends on the queue used (more about this later)

#PBS -l select=1:ncpus=16:mpiprocs=8:mem=120GB

Specifies the resources needed for the simulation.

select – number of compute nodes (“chunks”, max. 128)

ncpus – number of cpus per node (max. 16)

mpiprocs – number of MPI tasks per node (max=ncpus)

mem – memory allocated for each node (default=3.5 GB)

ACCOUNTING SYSTEM

#PBS -A <my_account>

Specifies the account that consumes the CPU hours allocated.

Your account for course exercises is **train_cmpD2017**.

As an 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.

ACCOUNTING SYSTEM

You can check the status 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*").

```
[amarani0@r000u07102 ~]$ saldo -b

--------------------------------------------------------------------------------------------------------------------------------
account                start          end          total        localCluster    totConsumed      totConsumed      monthTotal      monthConsumed
                                                  (local h)    Consumed(local h)   (local h)             %        (local h)          (local h)
--------------------------------------------------------------------------------------------------------------------------------
cin_staff          20110323     20200323     800000008              2017268      53875045              6.7         7299270             18527
smr_prod           20130308     20171215     12000000                353600       7536365             62.8          206540             55202
cin_priorit        20131115     20191231     4000000                1197301       3622341             90.6           53643             58183
cin_external       20150319     20201231     40000                    14462         17972             44.9             567                 0
FUSIO_TEST         20161116     20200801     40                          47            47            119.1               0                 0
train_scA2017      20170213     20170305     6000                         0             0              0.0               0                 0
[amarani0@r000u07102 ~]$
```

ACCOUNT FOR THE SCHOOL

The account provided for this school is "**train_cmpD2017**"

You have to specify it on your job scripts.

It will expire 2 days after the end of the course and is shared among all the students; there are plenty of hours for everybody, but don't waste them!

```
#PBS -A train_cmpD2017
```

RESERVATION FOR THE SCHOOL

The reservation will let you bypass the regular queue and let your jobs run immediately.

The “#PBS -q” keyword regulates the queue selection. If you omit it (regular user behaviour), your job will be processed among the queue suited for the resources you asked (debug, prod, bigprod). To use the reservation, add this to your jobscript:

```
#PBS -q R2475696
#PBS -W group_list=train_cmpD2017
```

PBS COMMANDS

After the job script is ready, all is left to do is to submit it:

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 -u

qstat -u <username>

Shows the list of all your scheduled jobs, along with their status (idle, running, closing, …) Also, shows you the job id required for other PBS commands.
Hint: add the flag “-w” for an extended vision and the full name of your jobid

PBS COMMANDS

qstat -f

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

OUTLINE

- A first step:
  - System overview
  - Login
  - Work environment
- Production environment
  - Our first job!!
  - Creating a job script
  - Accounting and queue system
  - PBS commands
- **Programming environment**
  - **Module system**
  - **Serial and parallel compilation**
  - **Interactive session**
- For further info...
  - Useful links and documentation

AN EXAMPLE OF A PARALLEL JOB

```
#!/bin/bash
#PBS -l walltime=1:00:00
#PBS -l select=2:ncpus=16:mpiprocs=16
#PBS -o job.out
#PBS -e job.err
#PBS -A train_cmpD2017

cd $PBS_O_WORKDIR     # points to the folder you are
                      # actually working into
module load autoload intelmpi
mpirun –np 16 ./myprogram
```

MODULE SYSTEM

- All the optional software on the system is made available through the **"module" system**. It provides a way to rationalize software and its environment variables.

MODULE SYSTEM

- Modules are divided in several *profiles*:
  - **profile/base** default - stable and tested compilers, libraries, tools
  - **profile/advanced** libraries and tools compiled with different setups that the default
  - **profile/eng** "domain" profile with the application software specific for the field of research

MODULE SYSTEM

- The profile is divided in 6 categories:
  - **tools** (octave, paraview)
  - **compilers** (gnu, intel, openmpi)
  - **environment** (autoload)
  - **libraries** (boost, metis)
  - **applications** (abaqus,pointwise)

MODULE SYSTEM

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

- "loading" a module means that a series of (useful) shell environment variables will be set

MODULE SYSTEM

- E.g. after a module is loaded, an environment variable of the form “<MODULENAME>_HOME” is set

```
[mcremone@node165 ~]$ module load autoload hdf5
[mcremone@node165 ~]$ ls $HDF5_HOME
bin  include  lib  share
```

For certain modules, a specific profile must be loaded before (“module load profile/...”).

MODULE COMMANDS

COMMAND	DESCRIPTION
module av	list all the available modules
module load <module_name(s)>	load module <module_name>
module list	list currently loaded modules
module purge	unload all the loaded modules
module unload <module_name>	unload module <module_name>
module help <module_name>	print out the help (hints)
module show <module_name>	print the env. variables set when loading the module

MODULE PREREQS AND CONFLICTS

Some modules need to be loaded after other modules they depend from (e.g.: parallel compiler depends from basic compiler). You can load both compilers at the same time with “autoload”

```
[mcremone@node165 ~]$ module load hdf5
WARNING: hdf5/1.8.17_ser--intel--pe-xe-2017--binary
   cannot be loaded due to missing prereq.
HINT: the following modules must be loaded first:
   intel/pe-xe-2017--binary
```

MODULE PREREQS AND CONFLICTS

You may also get a “conflict error” if you load a module not suited for working together with other modules you already loaded (e.g. different compilers). Unload the previous module with “module unload”

JOB SCRIPT FOR PARALLEL EXECUTION

Let's take a step back…

```
#PBS -l select=2:ncpus=16:mpiprocs=4
```

This example line means "allocate 2 nodes with 16 CPUs each, and 4 of them should be considered as MPI tasks"

So a total of 32 CPUs will be available. 8 of them will be MPI tasks, the others will be OpenMP threads (4 threads for each task).

In order to run a pure MPI job, ncpus must be equal to mpiprocs.

EXECUTION LINE IN JOB SCRIPT

```
mpirun -np 8 ./myprogram
```

Your parallel executable is launched on the compute nodes via the command *“mpirun”.*
With the “-np” flag you can set the number of MPI tasks used for the execution. The default is the maximum number allowed by the resources requested.

EXECUTION LINE IN JOB SCRIPT

WARNING

In order to use mpirun, **openmpi-intelmpi has to be loaded** inside the job script:

```
module load autoload intelmpi
```

Be sure to load the same version of the compiler that you used to compile your code!!

DEVELOPING IN COMPUTE NODES: INTERACTIVE SESSION

It might be easier to compile and develop directly in the compute nodes, without recurring to a batch job.

For this purpose, you can launch an interactive job to enter inside a compute node by using PBS.

The node will be reserved to you as if it had been requested by a regular batch job

DEVELOPING IN COMPUTE NODES: INTERACTIVE SESSION

Basic interactive submission line:

```
qsub -I -l select=1 -A <account_name> -q <queue_name>
```

Other PBS keyword can be added to the line as well (walltime, resources, …)

Remember that you are using computing nodes, and by consequence you are consuming computing hours!

To exit from an interactive session, just type “exit”

OUTLINE

- A first step:
  - System overview
  - Login
  - Work environment
- Production environment
  - Our first job!!
  - Creating a job script
  - Accounting and queue system
  - PBS commands
- Programming environment
  - Module system
  - Serial and parallel compilation
  - Interactive session
- For further info...
  - Useful links and documentation

Useful links and documentation

- **Reference guide:**
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.2%3A+GALILEO+UserGuide
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.5.2%3A+Batch+Scheduler+PBS
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.4%3A+Data+storage+and+FileSystem

- **About vector optimization:**
  https://wiki.u-gov.it/confluence/display/SCAIUS/How+to+Improve+Code+Vectorization

- **Stay tuned with the HPC news: http://www.hpc.cineca.it/content/stay-tuned**

- **HPC CINECA User Support:** mail to **superc@cineca.it**
- **HPC Courses: corsi@cineca.it**