Cineca
TRAINING
High Performance
Computing 2016

Introduction to GALILEO

Parallel & production environment

Mirko Cestari - m.cestari@cineca.it
Alessandro Marani - a.marani@cineca.it
Alessandro Grottesi - a.grottesi@cineca.it
SuperComputing Applications and Innovation Department

CINECA

June 7, 2016

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

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

GALILEO CHARACTERISTICS

Model: IBM NeXtScale

Architecture: Linux Infiniband Cluster

Processors Type: 8-cores Intel Haswell 2.40 GHz (2 per node)

Number of nodes: 516 Compute

Number of cores: 8256

Accelerators: 2 Intel Phi 7120p per node on 384 nodes (768 in total)
4 nVIDIA Tesla K40 on 40 nodes (160 in total)

RAM: 128 GB/node, 8 GB/core

OS: RedHat CentOS release 7.0, 64 bit

GALILEO CHARACTERISTICS

- **Compute Nodes:** 516 16-core compute cards (nodes).
  - 384 nodes contain 2 Intel Phi 7120p processors
  - 40 nodes contain 2 nVIDIA Tesla K80 "Kepler" per node (being 4 the total number of K40 visible devices)
  - The nodes have 16GB of memory, but the allocatable memory on the node is 120 GB.
  - Not all nodes are available for all the users. A partition of the cluster (including 30 out of the 40 nVIDIA nodes) is reserved to industrial users, and the rest is available for academical users.

- **Login node:** 8 Login & Viz node NX360M5 are available, equipped with 2 nVidia K40 GPU each.
- **Network**: all the nodes are interconnected through a custom Infiniband network with 4x QDR switches, allowing for a low latency/high bandwidth interconnection.

A LOOK AT THE FUTURE: MARCONI

PHASE A1 (July 2016)

Model: Lenovo NeXtScale

Processor Type: Intel Broadwell, 2.3GHz

Peak Performance: 2 PFlop/s (ranked #46 in Top500 list)

PHASE A2 (Fall 2016)

Model: Lenovo Adam Pass

Processor Type: Intel Knights Landing BIN1, 1.4GHz

Computing Nodes: 3.600 with 68 cores each

Peak Performance: 11 PFlop/s (presumed)

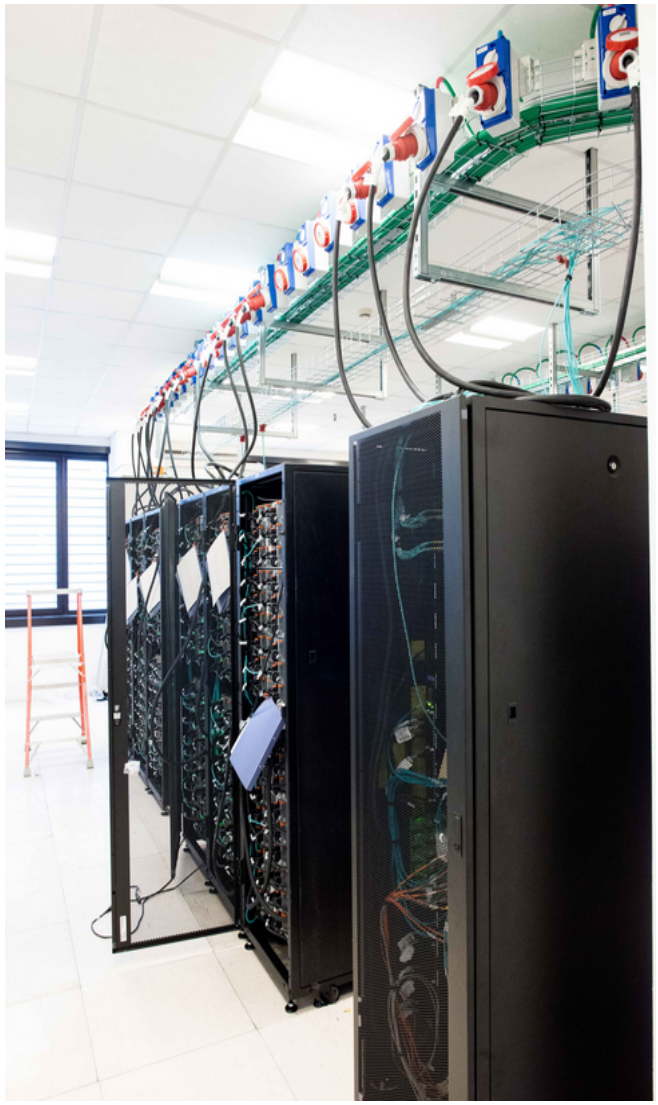
PHASE A3 (Spring 2017)

Model: Lenovo Stark

Processor Type: Intel SkyLake, 2.3GHz

Computing Nodes: 1512 with 40 cores each

Peak Performance: 4,5 PFlop/s (presumed)



How to log in

- Establish a ssh connection

ssh <username>@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.

$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=10GB
#PBS -q <queue_name>
#PBS -A <my_account>
#PBS -W group_list=<my_account>

echo “I’m working on GALILEO!”
```

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 -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=10GB

Specifies the resources needed for the simulation.

select – number of compute nodes (“chunks”)

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

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

mem – memory allocated for each node (default=8GB, max.=120 GB)

ACCOUNTING SYSTEM

#PBS -A <my_account>

Specifies the account to use the CPU hours from.

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.

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@fen08 ~]$ saldo -b

--------------------------------------------------------------------------------------------------------
account              start         end          total       localCluster    totConsumed      totConsumed
                                                (local h)   Consumed(local h)   (local h)              %
--------------------------------------------------------------------------------------------------------
cin_staff          20110323     20200323     1000000000           30365762      30527993            3.1
cin_totview        20130123     20130213          50000                  0             0            0.0
train_sc32013      20130211     20130411        1250000              87458         87458            7.0
train_cnl12013     20130311     20130411         100000                  0             0            0.0
```

ACCOUNT FOR THE SCHOOL

The account provided for this school is "**train_scR2016**"

(you have to specify it on your job scripts).

It will expire two weeks after the end of the school and is shared between all the students; there are plenty of hours for everybody, but don't waste them!

#PBS -A train_scR2016

QUEUING SYSTEM

#PBS -q R1400212

#PBS -q R1400354 → **nodi di Galileo con GPU Giovedì 21 Luglio!!**

Specifies the queue requested for the job. As a regular user, you don't have to specify anything, as you will be redirected to the queue most suited for the size of your job.

#PBS -W group_list=train_scR2016

Specifies that you are member of a group authorized to use the selected queue. This keyword is not required when you are a regular GALILEO user, and must not be added to your jobscript.

QUEUING SYSTEM - SCHOOL EXCEPTIONS

The rules of the slides above can be broken for this school!

You have the access to 4 nodes reserved for the days of the Summer School. To use them, you have to specify the queue:

#PBS -q R1400212

In order to be recognized as an user allowed to have access to the reservation, you have to specify the group_list as well:

#PBS -W group_list=train_scR2016 (the same as your account)

After the school, you can use the account for other 2 months (up to September 22nd), but you will lose your privileges and be able to run your jobs as a regular user. Remember to delete the queue and group_list keywords on your job scripts!!

PBS COMMANDS

After the job script is ready, all there 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.

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

EXERCISE 01

1) Write a job script with "walltime" of 3 minutes that asks for 1 node and 1 core. Copy-paste the following in the execution section

hostname

echo 'Hello World'

sleep 4

Now add the automatic sending of the email in case of ending and abort of the job.

2) Launch the job with qsub

3) Check its state with qstat

4) Check its state again with "qstat -f jobid" after having increased the sleep to 60, namely:

hostname

echo 'Hello World'

sleep 60

5) Add a memory request to the "select" line in the job script (rember that each processor has a quota of 850 MB of memory). Please check the new requirements with "qstat -f jobid"

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=4
#PBS -o job.out
#PBS -e job.err
#PBS -q <queuename>
#PBS -A <my_account>
#PBS -W group_list=<my_account>
cd $PBS_O_WORKDIR    # points to the folder you are actually working into
module load autoload openmpi
mpirun –np 8 ./myprogram
```

MODULE SYSTEM

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

- Modules are divided in 2 *profiles*
  - **profile/base** (default - stable and tested modules)
  - **profile/advanced** (software not yet tested or not well optimized)

- Each profile is divided in 4 categories
  - **compilers** (GNU, intel, openmpi)
  - **libraries** (e.g. LAPACK, BLAS, FFTW, ...)
  - **tools** (e.g. Scalasca, GNU make, VNC, ...)
  - **applications** (software for chemistry, physics, ...)

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

```
[amarani0@fen07 ~]$ module load namd
[amarani0@fen07 ~]$ ls $NAMD_HOME
backup  flipbinpdb  flipdcd  namd2  namd2_plumed  namd2_remd  psfgen  sortreplicas
```

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”

```
[cin0955a@node342 ~]$ module load openmpi
WARNING: openmpi/1.4.4--gnu--4.5.2 cannot be loaded due to missing prereq.
HINT: the following modules must be loaded first: gnu/4.5.2
[cin0955a@node342 ~]$ module load autoload openmpi
### auto-loading modules gnu/4.5.2
```

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”

COMPILING ON GALILEO

- On Galileo you can choose between three different compiler families: **gnu, intel** and **pgi**
- You can take a look at the versions available with "*module av" and then* load the module you want.

module load intel *# loads default intel compilers suite*

module load intel/pe-xe-2016--binary *# loads specific compilers suite*

	GNU	INTEL	PGI
Fortran	gfortran	ifort	pgf77
C	gcc	icc	pgcc
C++	g++	icpc	pgcc

Get a list of the compilers flags with the command *man*

PARALLEL COMPILING ON GALILEO

- MPI libraries available: **OpenMPI/IntelMPI**
  - The library and special wrappers to compile and link the personal programs are contained in several modules, one for each supported suite of compilers
- Load a version of **OpenMPI** (in profile/advanced):

```
module av openmpi
    openmpi/1.8.4--gnu--4.9.2 (the only one in profile/base)
    openmpi/1.8.4--intel--cs-xe-2015--binary
    openmpi/1.8.5--gnu--4.9.2
    openmpi/1.8.5--pgi--15.3
module load autoload openmpi/1.8.5—gnu—4.9.2
```

- Load a version of **IntelMPI** (in profile/advanced):

```
module av intelmpi
        intelmpi/5.0.1—binary
        intelmpi/5.0.2—binary
        intelmpi/5.1.1—binary
        intelmpi/5.1.3--binary
module load autoload intelmpi/5.0.1--binary
```

PARALLEL COMPILING ON GALILEO

	OPENMPI/INTELMPI
Fortran90	mpif90
C	mpicc
C++	mpiCC

Compiler flags are the same of the basic compiler (since they are basically MPI wrappers of those compilers)

OpenMP is provided with following compiler flags:

gnu: -fopenmp

intel : -openmp

pgi: -mp

PARALLEL COMPILING ON GALILEO

To compile a parallel code using openmpi/intelmpi:

```
>module load autoload openmpi/intelmpi
>module load <some_library>
>mpicc my_code.c -O2 -L$my_library -llibname -o my_code.x
```

To compile a parallel code using OpenMP:

```
>module load intel
>module load <some_library>
>icc my_code.c -L$my_library -llibname -o my_code.x -openmp
```

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.

WARNING:

In order to use mpirun, **openmpi-intelmpi has to be loaded.**

module load autoload openmpi

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

DEVELOPING IN COMPUTE NODES: INTERACTIVE SESSION

It may 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 it was requested by a regular batch job

Basic interactive submission line:

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

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

Keep in mind that you are using computing nodes, and by consequence you are consuming

computing hours!

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

EXERCISE 02

1) Compile "test.c" with the compiler (mpicc) in the module intelmpi/4.1.1--binary

2) Check with:

$ ldd <executable>

the list of required dynamic libraries.

3) Write "job.sh" (you can copy it from exercise 1), modifying the "select" line with the following requests:

#PBS -l select=2:ncpus=16:mpiprocs=16:mem=12gb

#PBS -l select=2:ncpus=16:mpiprocs=1:mem=12gb

Run first 32 processes and then 2 processes for each select.

EXERCISE 03

1) Launch an interactive job. You just need to write the same PBS directives, without "#PBS" and on the same line, as arguments of "qsub -I"

$ qsub -I ... <arguments>

2) Check whether you are on a different node

3) Check that there's an interactive job running

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
  https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.1%3A+MARCONI+UserGuide (WIP)

- **GPU computing** *http://www.nvidia.com/object/GPU_Computing.html*
  MIC programming http://software.intel.com/en-us/mic-developer

- **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