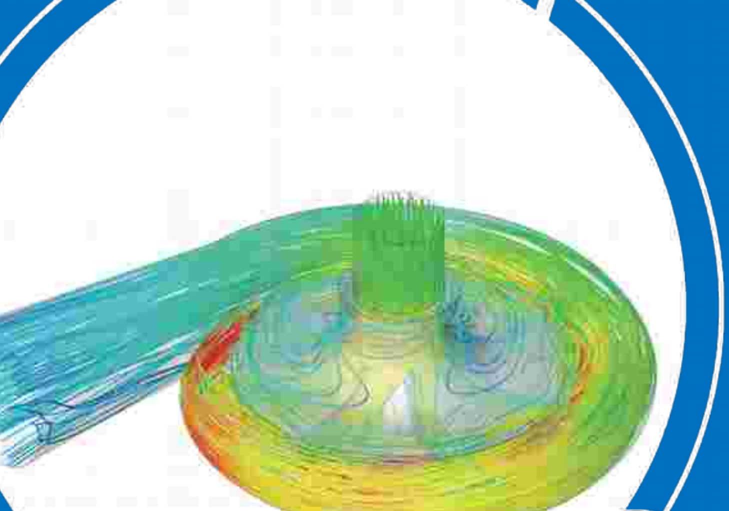
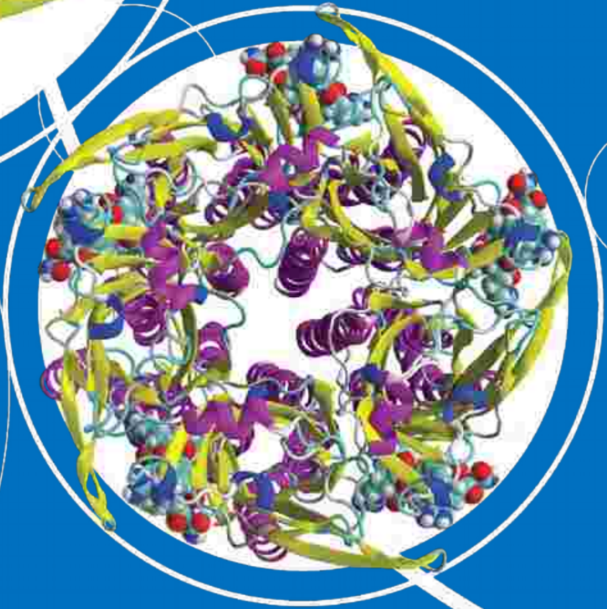
HPC Infrastructure at CINECA: overview and hands-on

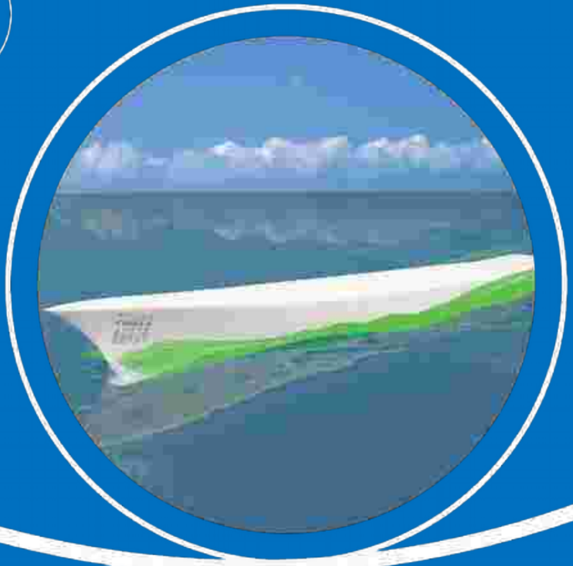
Neva Besker – n.besker@cineca.it

Alessandro Grottesi – a.grottesi@cineca.it

High Performance Computing Department - CINECA







Outlook

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 HPC cluster

OUTLINE

A first step:

- **HPC infrastructures at CINECA**
- **GALILEO: system overview**
- **Login**
- **Work environment**

Production environment

- **Our first job!!**
- **Creating a job script**
- **Accounting and queue system**
- **SLURM commands**

Programming environment

- **Module system**
- **Serial and parallel compilation**
- **Interactive session**

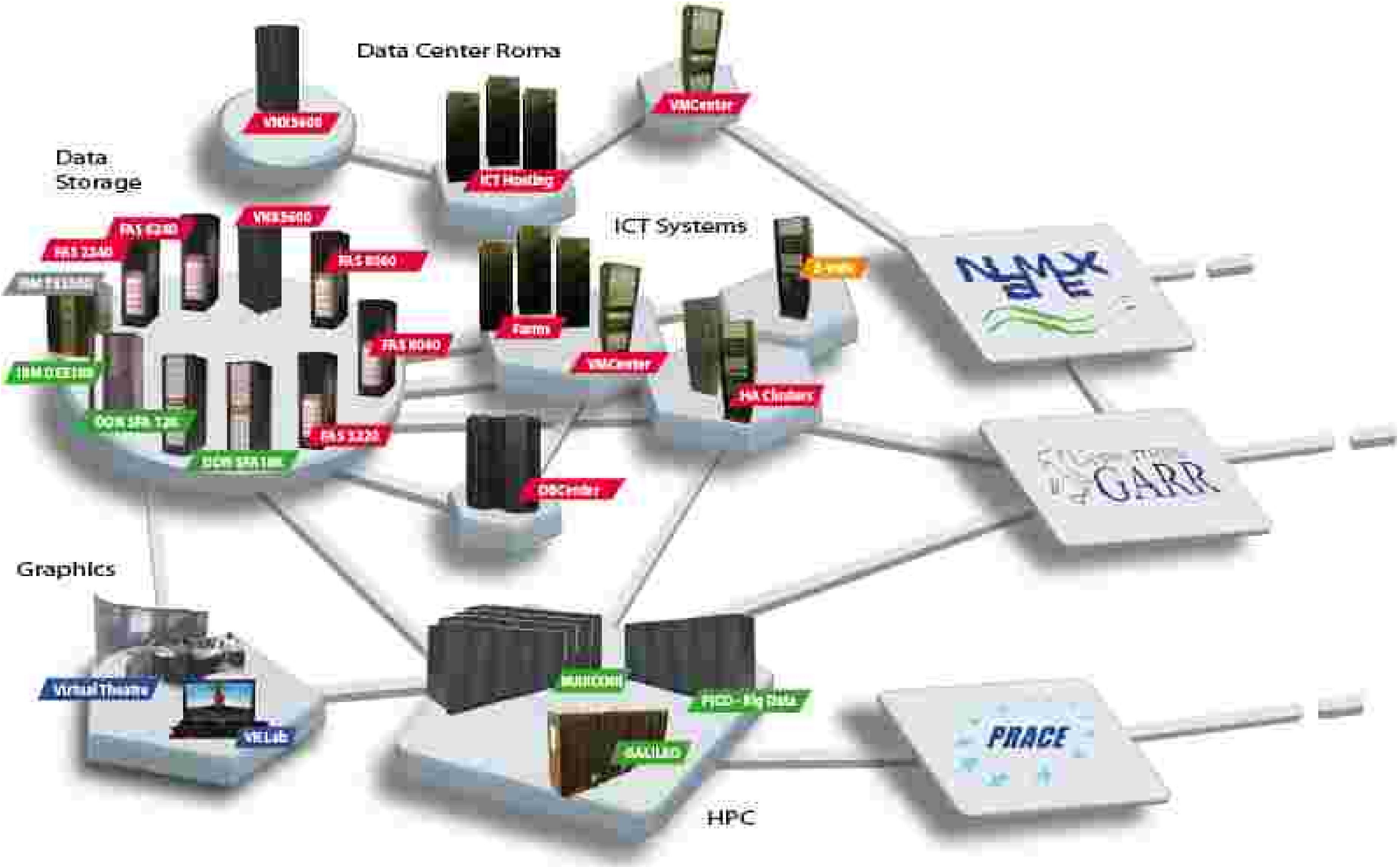
GPU environment

Graphical session with RCM

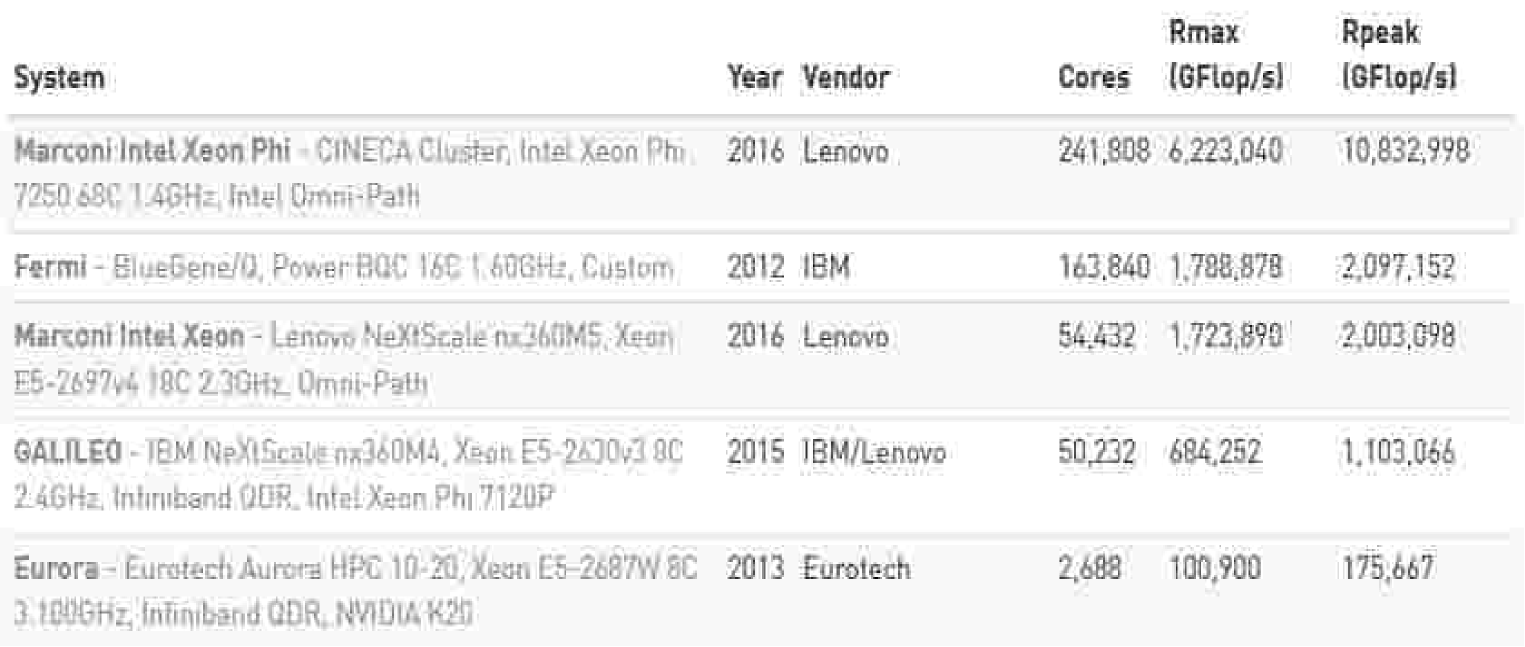
For further info...

- **Useful links and documentation**

CINECA Infrastructure Map



CINECA & Top500



System	Year	Vendor	Cores	Rmax (GFlop/s)	Rpeak (GFlop/s)	Best placement
Marconi Intel Xeon Phi - CINECA Cluster, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path	2016	Lenovo	241,808	6,223,040	10,832,998	**12**
Fermi - BlueGene/Q, Power BQC 16C 1.60GHz, Custom	2012	IBM	163,840	1,788,878	2,097,152	**7**
Marconi Intel Xeon - Lenovo NeXtScale nx360M5, Xeon E5-2697v4 18C 2.3GHz, Omni-Path	2016	Lenovo	54,432	1,723,890	2,003,098	**46**
GALILEO - IBM NeXtScale nx360M4, Xeon E5-2630v3 8C 2.4GHz, Infiniband QDR, Intel Xeon Phi 7120P	2015	IBM/Lenovo	50,232	684,252	1,103,066	**105**
Eurora - Eurotech Aurora HPC 10-20, Xeon E5-2687W 8C 3.100GHz, Infiniband QDR, NVIDIA K20	2013	Eurotech	2,688	100,900	175,667	**467**

- MARCONI (A2+A3) is currently the **21st** most powerful supercomputer in the World. It is ranked first among EU nations.
- In June of 2012 we reached **7th** place with FERMI (currently dismissed)
- MARCONI (A1), is approximately as powerful as FERMI was, but when it debuted on June 2016 it ranked at **46th** place
- GALILEO: (currently under major upgrade) ranked at **105th** in 2015 with a sustained performance of 684.3 TFlops.
- EURORA (currently dismissed) ranked in Top500 only once, at **467th** place, but that time it was ranked **1st** in the Green500, making it the greenest supercomputer in the World at the time.

MARCONI

MARCONI is currently our top-tier supercomputer. It is divided in three partitions (A1, A2 and A3) and can be considered as three different HPC systems in one

PARTITION A1
Model: Lenovo NeXtScale
Processor Type: Intel Broadwell, 2.3GHz
Peak Performance: 2 PFlop/s

PARTITION A2
Model: Lenovo Adam Pass
Processor Type: Intel Knights Landing, 1.4GHz
Computing Nodes: 3.600 with 68 cores each
Peak Performance: 11 PFlop/s

PARTITION A3
Model: Lenovo Stark
Processor Type: Intel SkyLake, 2.3GHz
Computing Nodes: 1512 with 40 cores each
Peak Performance: 4,5 PFlop/s

Network: all the nodes are interconnected through a custom Intel Omnipath network that can go up to 100Gb/s, making MARCONI the largest Omnipath cluster in the World.

D.A.V.I.D.E.

D.A.V.I.D.E. (**D**evelopment of an **A**dded **V**alue **I**nfrastructure **D**esigned in **E**urope) is an energy-aware Petaflops Class High Performance Cluster based on Power Architecture and coupled with NVIDIA Tesla Pascal GPUs with NVLink. The innovative design of D.A.V.I.D.E. has been developed by E4 Computer Engineering for PRACE.

Model: E4 Cluster Open rack
Architecture: OpenPower NViDIA NVLink
Nodes: 45 x (2 Power8+4Tesla P100) + 2 (service&login nodes)
Processors: OpenPower8 NVIDIA Tesla P100 SXM2
Internal Network: 2xIB EDR, 2x1GbE
Cooling: SoC and GPU with direct hot water
Cooling capacity: 40kW
Heat exchanger: Liquid-liquid, redundant pumpsModel: E4 Cluster Open rack
Storage: 1xSSD SATA
Max Performances: 22 TFLOPs (double precision), 44 TFLOPs single precision
Peak Performance: ~1 PFlop/s

Galileo

GALILEO is a smaller cluster that will be your «home» during this week. Unlike MARCONI, it is equipped with accelerators, especially GPUS which will be a major topic during the school

Model: IBM NeXtScale
Architecture: Linux Infiniband Cluster
Processors: 16-cores Intel Broadwell 2.30 GHz (2 per node)
Number of Nodes: 360
Internal Network: Infiniband
Accelerators: 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 Overview

Compute Nodes: 400 16−core compute cards (nodes)
- 360 nodes contain 2 Intel Broadwell compute cards, 36 cores per node
- 40 nodes contain 2 Intel Haswell compute cards, 16 cores per node, and 2 nVIDIA Tesla K80 "Kepler" per node (being 4 the total number of K40 visible devices)

-The nodes have 128GB of memory, but the allocatable memory on the node is 115 GB (actually 118000 MB).
- Not all nodes are available for all the users. A partition of the cluster (including 26 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.

How to log in

Establish a ssh connection

```
ssh <username>@login.galileo.cineca.it
```

Remarks:

- **ssh** is 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**! (still incomplete though, informations about GPU access are not available yet)

https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.3%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 policy will delete all your files older than 30 days.

$WORK:

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

1 Tb of quota (no cleaning policy). 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:

- HPC infrastructures at CINECA
- GALILEO: system overview
- Login
- Work environment

Production environment

- **Our first job!!**
- **Creating a job script**
- **Accounting and queue system**
- **SLURM commands**

Programming environment

- Module system
- Serial and parallel compilation
- Interactive session

GPU environment

Graphical session with RCM

For further info...

- Useful links and documentation

Jobs & Schedulers

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 **SLURM**

SLURM job script scheme

The scheme of a SLURM job script is as follows:

#!/bin/bash

#SLURM directives

variables environment

execution line

Jobscript example

```
#!/bin/bash
#SBATCH --job-name=myname
#SBATCH --output=job.out
#SBATCH --error=job.err
#SBATCH --mail-type=ALL
#SBATCH --mail-user=user@email.com
#SBATCH --time=00:30:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=16
#SBATCH –ncpus-per-task=1
#SBATCH --ntasks-per-socket=8
#SBATCH --mem=10GB
#SBATCH --partition=gll_usr_prod
#SBATCH --account=<my_account>

echo “I’m working on GALILEO!”
```

SLURM directives - 1

#SBATCH --job-name=myname, -J myname

Defines the name of your job

#SBATCH --output=job.out, -o job.out

Specifies the file where the standard output is directed (default=slurm-<Pid>)

#SBATCH --error=job.err, -e job.err

Specifies the file where the standard error is directed (default=slurm-<Pid>)

#SBATCH --mail-type=ALL (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 (NONE, BEGIN, END, FAIL, REQUEUE, ALL)

#SBATCH --mail-user=user@email.com (optional)

Specifies the e-mail address for the keyword above

SLURM directives - 2

#SBATCH --time=00:30:00, -t 00:30:00

Specifies the maximum duration of the job. The maximum time allowed depends on the partition used

#SBATCH --nodes=1, -N 1

#SBATCH --ntasks-per-node=16

#SBATCH --mem=10GB

Specifies the resources needed for the simulation.

nodes – number of compute nodes ("chunks")

ntasks-per-node – number of cpus per node (max. 36)

mem – memory allocated for each node (default= 3000MB, max= 118000 MB)

#SBATCH --partition=gll_usr_prod, -p gll_usr_prod

Specifies the "partition", a.k.a the specific set of nodes among which your job can search for resources.

Only two partitions on GALILEO: **gll_usr_prod** (regular nodes) and **gll_usr_gpuprod** (GPU nodes)

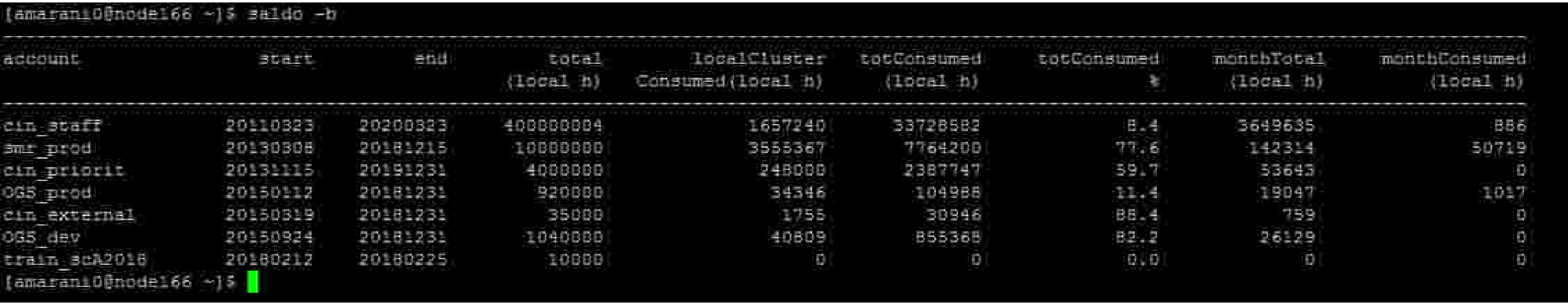
Accounting System

#SBATCH --account=<my_account>, -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@node166 ~]$ 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    400000004            1657240      33728582            8.4      3649635             886
smr_prod          20130308   20181215     10000000            3555367       7764200           77.6       142314           50719
cin_priorit       20131115   20191231      4000000             248000       2387747           59.7        53643               0
OGS_prod          20150112   20181231       920000              34346        104988           11.4        19047            1017
cin_external      20150319   20181231        35000               1755         30946           88.4          759               0
OGS_dev           20150924   20181231      1040000              40809        855368           82.2        26129               0
train_scA2018     20180212   20180225        10000                  0             0            0.0            0               0
[amarani0@node166 ~]$
```

Account for the School

The account provided for this school is “train_scR2019”
(you have to specify it on your job scripts).
It will expire a week 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!

```
#SBATCH --account= train_scR2019
```

SLURM Commands - 1

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

sbatch

```
sbatch <job_script>
```

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

squeue -u

```
squeue -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 SLURM commands

SLURM Commands - 2

scontrol show job -f

```
scontrol show job <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 the reason it is not starting and, if it is scheduled with top priority, you will get an estimated start time

scancel

```
scancel <job_id>
```

Removes the job (queued or running) from the scheduled job list by killing it

Tutorial #1

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 sbatch

3) Check its state with squeue

4) Check its state again with squeue after having increased the sleep to 60, namely:

```
hostname
echo 'Hello World'
Sleep 60
```

5) You can find the tutorials either by:

- git clone https://gitlab.hpc.cineca.it/training/summer-school-2019-rome.git
- cp -r /galileo/home/userinternal/agrottes/Cineca-HPC-systems-Introduction/ .

Outline

A first step:

- HPC infrastructures at CINECA
- GALILEO: system overview
- Login
- Work environment

Production environment

- Our first job!!
- Creating a job script
- Accounting and queue system
- SLURM commands

Programming environment

- **Module system**
- **Serial and parallel compilation**
- **Interactive session**

GPU environment

Graphical session with RCM

For further info...

- Useful links and documentation

Parallel Job Environment

```
#!/bin/bash
#SBATCH -t 1:00:00
#SBATCH -N 2
#SBATCH --ntasks-per-node=18
#SBATCH -c 2
#SBATCH --mem=10GB
#SBATCH -o job.out
#SBATCH -e job.err
#SBATCH -A <my_account>

module load autoload intelmpi
srun --mpi=pmi2 ./myprogram
```

Module System - 1

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.

Modules are divided in several *profiles*:

- **profile/base** - default/stable and tested compilers, libraries, tools
- **profile/advanced** - libraries and tools compiled with different setups than the default
- **profile/chem** (phys, bioinf, astro,...) - “domain” profiles with the application softwares specific for each research field
- **profile/archive** - old or outdated versions of our module - we don't throw away anything!

Each profile is divided in 4 categories:

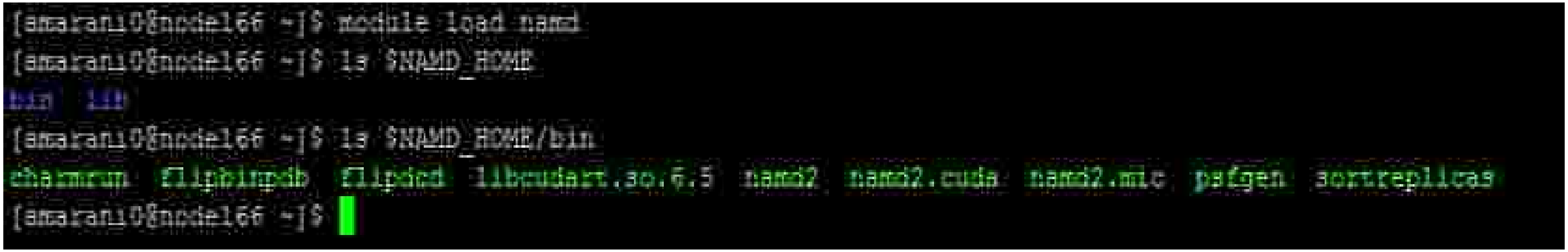
compilers (GNU, intel, openmpi)
libraries (e.g. LAPACK, BLAS, FFTW, ...)
tools (e.g. Scalasca, GNU make, VNC, ...)
applications (software for chemistry, physics, ...)

Module System - 2

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

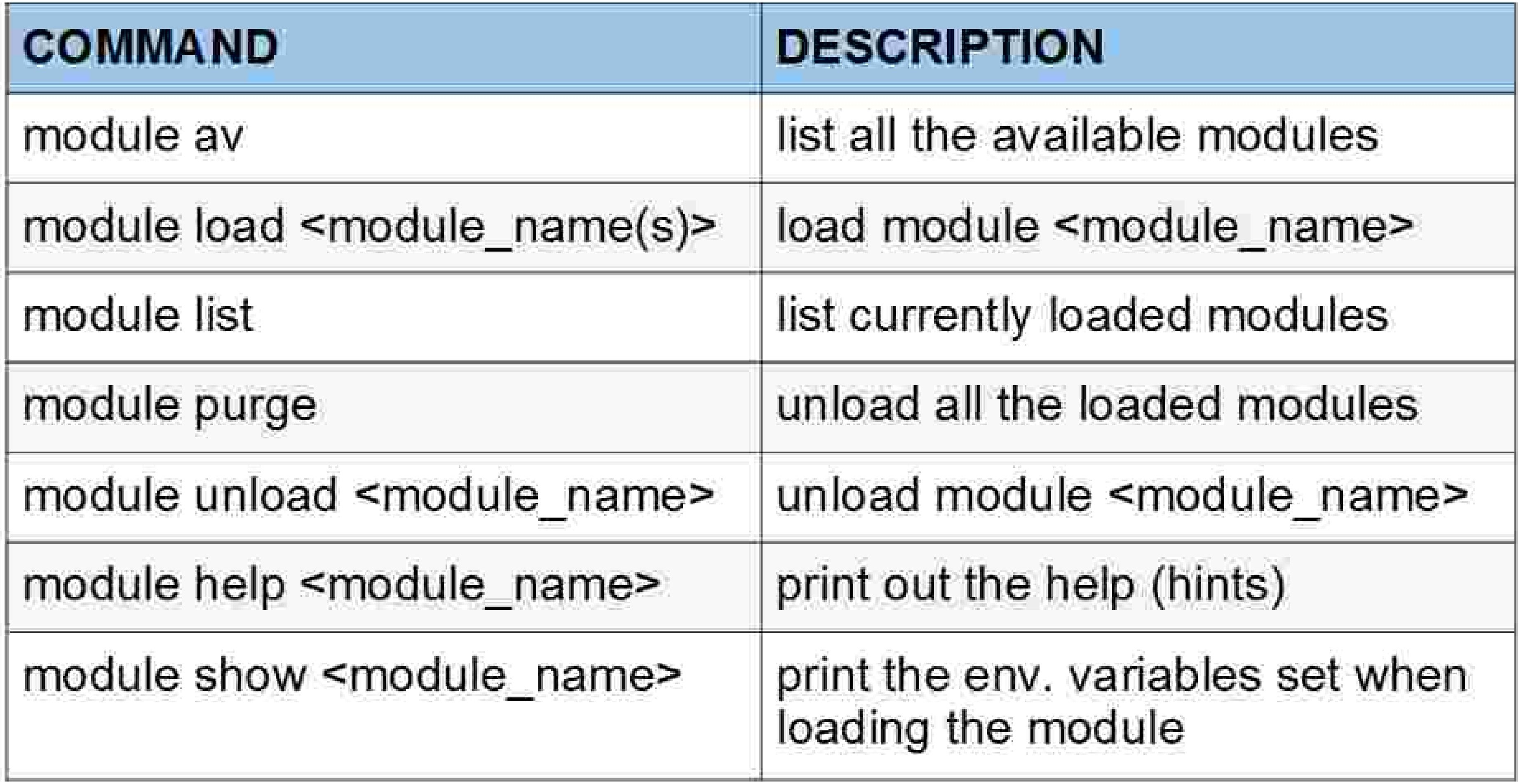
“Loading” a module means that a series of (useful) shell environment variables will be set

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



```
[amarani0@node166 ~]$ module load namd
[amarani0@node166 ~]$ ls $NAMD_HOME
bin  lib
[amarani0@node166 ~]$ ls $NAMD_HOME/bin
charmrun  flipbinpdb  flipdcd  libcudart.so.6.5  namd2  namd2.cuda  namd2.mic  psfgen  sortreplicas
[amarani0@node166 ~]$
```

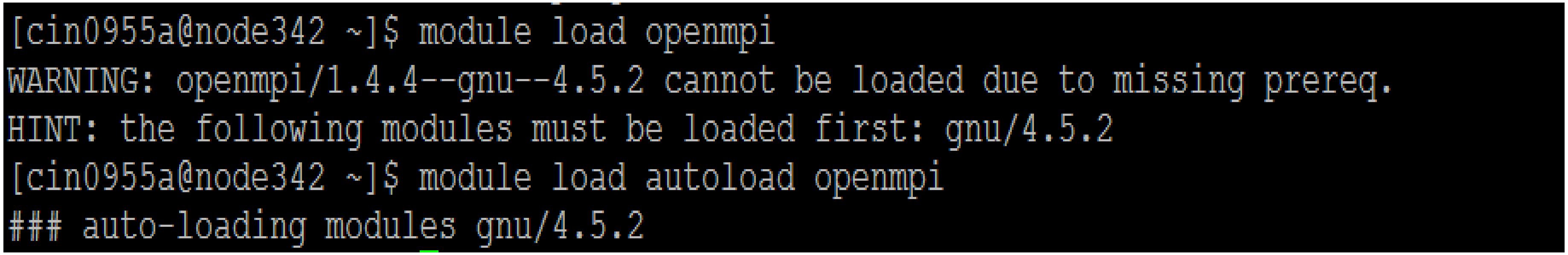
Module Commands



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 dependencies 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 modules 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
```

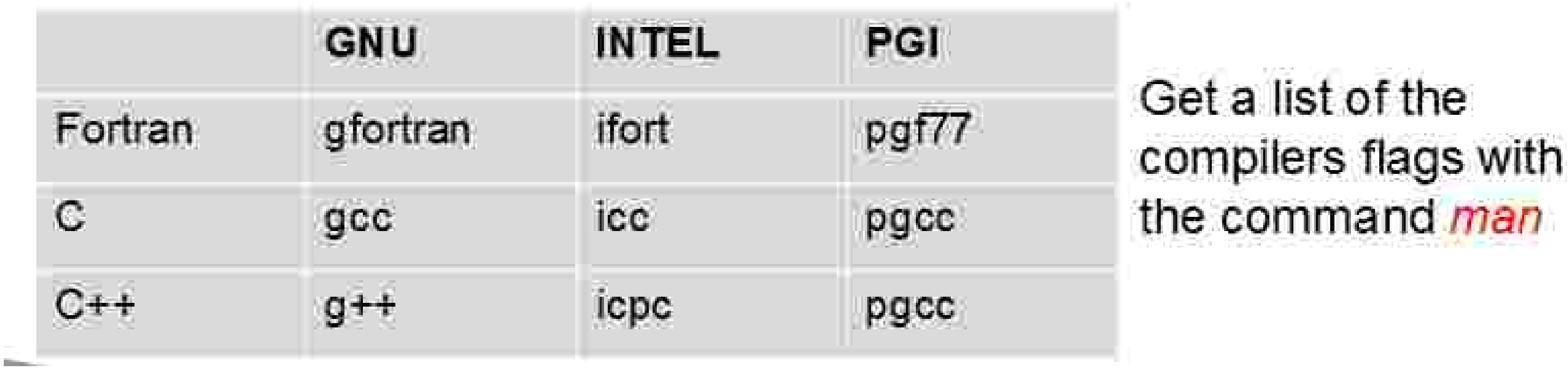
You may also get a "conflict error" if you load a module not suited for working together with other modules you already loaded (e.g. different compilers). Unload the previous module with "module unload"

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-2017--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 - 1

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

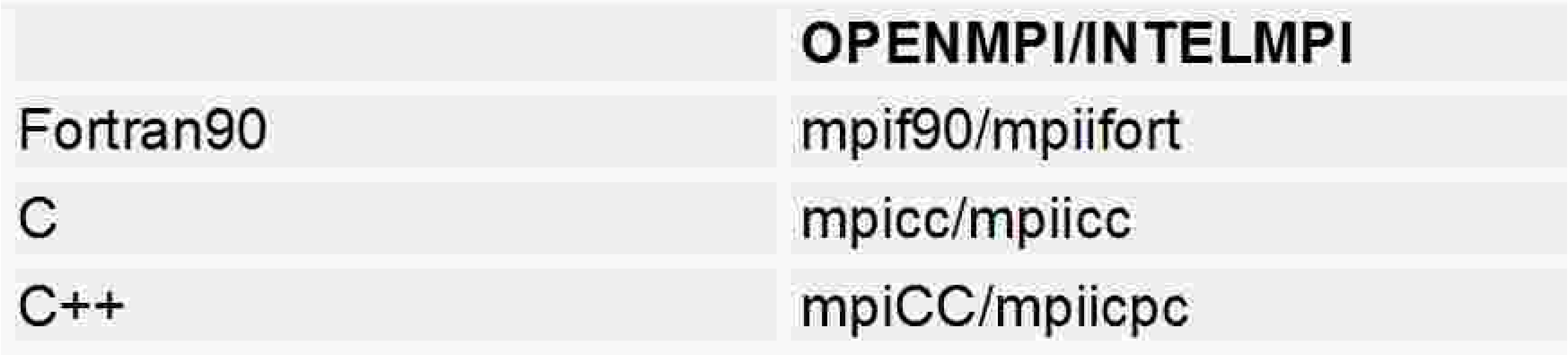
Load a version of OpenMPI (in profile/advanced):

```
module av openmpi
openmpi/2.1.1--gnu--6.1.0

module load autoload openmpi/2.1.1--gnu--6.1.0
```

Load a version of IntelMPI:

```
module av intelmpi
intelmpi/2017--binary
intelmpi/2018--binary

module load autoload intelmpi/2017--binary
```

Parallel compiling on Galileo - 2



	OPENMPI/INTELMPI
Fortran90	mpif90/mpiifort
C	mpicc/mpiicc
C++	mpiCC/mpiicpc

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

OpenMP is provided with the following compiler flags:
gnu: -fopenmp
intel : -qopenmp
pgi: -mp

Job script for parallel execution

Let’s take a step back…

```
#SBATCH -N 2
#SBATCH --ntasks-per-node=18
#SBATCH -c 2
```

This example means “allocate 2 nodes with 36 CPUs each. At any node are assigned 18 MPI tasks, and at any task are assigned 2 CPUs.”

```
#SBATCH --cpus-per-task=2, -c 2
```

This is the number of cores that can be treated as a single MPI task. They can then become OpenMP threads.

Execution line in job script

```
srun --mpi=pmi2 ./myprogram
```

Your parallel executable is launched on the compute nodes via the command *“srun”*, with all the tasks requested via resource allocation.

The “--mpi=pmi2” flag is the method the MPI library uses to launch the tasks. Just trust us on pmi2 being the recommended choice on GALILEO.

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!!

Interactive jobs

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 SLURM. The node will be reserved to you as if it was requested by a regular batch job

Basic interactive submission line (to be launched directly on command shell):

```
srun –N 1 –A <account_name> -p gll_usr_prod ... --pty /bin/bash
```

Other SLURM keywords can be added to the line as well (walltime, tasks-per-node,…)

Keep in mind that you are using computing nodes, and consequently you are consuming computing hours!

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

Tutorial #2

1) Complete the first MPI exercise (MPI hello world) and compile it with the compiler (mpiicc or mpiifort) in the module intelmpi/2017--binary (default)

Find the exercise here:

```
/galileo/home/userinternal/agrottes/Cineca-HPC-systems-Introduction
```

2) Check with:

```
$ ldd <executable>
```

the list of required dynamic libraries.

3) Write "job.sh" (you can copy it from Tutorial #1), asking for two nodes and modifying the --ntasks-per-node with the following requests:

```
#SBATCH --ntasks-per-node=16
#SBATCH --ntasks-per-node=2
```

Run firstly 32 processes and then 4 processes.

Tutorial #3

1) Launch an interactive job. You just need to write the same SLURM directives, without "#SBATCH" and on the same line, as arguments of "srun". Remember to finish the line with "--pty /bin/bash"

```
$ srun ... <arguments> --pty /bin/bash
```

2) Check whether you are on a different node

3) Check that there's an interactive job running

Outline

A first step:
- HPC infrastructures at CINECA
- GALILEO: system overview
- Login
- Work environment

Production environment
- Our first job!!
- Creating a job script
- Accounting and queue system
- SLURM commands

Programming environment
- Module system
- Serial and parallel compilation
- Interactive session

GPU environment

Graphical session with RCM

For further info...
- Useful links and documentation

Compiling for GPUs

DISCLAIMER: In this presentation I will stay as basic as it is possible. Better and more aimed teachings about the subject will come from the teachers of the following days

To compile codes suited for GPU application (*.cu), we have to load the specific compiler module, **CUDA**
CUDA is not available on profile/base, but you can find it on profile/advanced, as well as older versions on profile/archive:

```
------------ /cineca/prod/opt/modulefiles/base/compilers ------------
cuda/10.0   cuda/8.0.61 cuda/9.0
```

Load the module (it doesn’t have dependencies) and use the CUDA compiler, nvcc:

```
nvcc mycudacode.cu -o mycudaexe.x
```

SLURM & GPUs

```
#!/bin/bash
#SBATCH --time=1:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem=10GB
#SBATCH --partition=gll_usr_gpuprod
#SBATCH --gres=gpu:kepler:4
#SBATCH --account=<my_account>

./mycudaexe.x
```

You can ask for up to 4 GPUs for each node (there are 2 K80s that the system can see as 4 K40s). If they aren’t asked, SLURM won’t allocate them for you even if they are in the node you are using

You don’t have to load the cuda module inside your jobscript!

Tutorial #4

1) Compile the provided cuda program, after having loaded the required modules

2) Write a job script for a serial execution (1 chunk, 1 cpu) that also asks for a gpu device

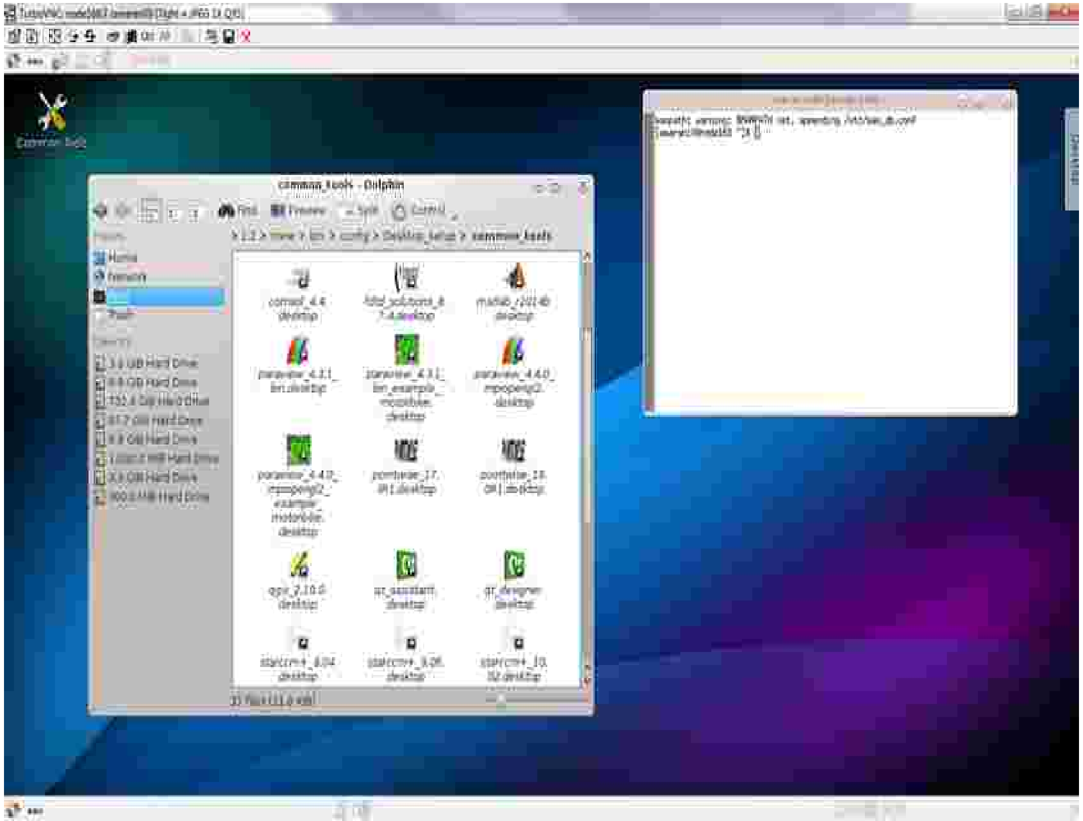
3) Reservation:s_tra_gpuR

3) Run the job script

Graphic Interface on Galileo: RCM

It is possible to login in GALILEO and work with a Graphical User Interface, in a more user-friendly environment

In this environment, some of the most common tools for post-processing and visualization are available



This can be done thanks to **RCM**!

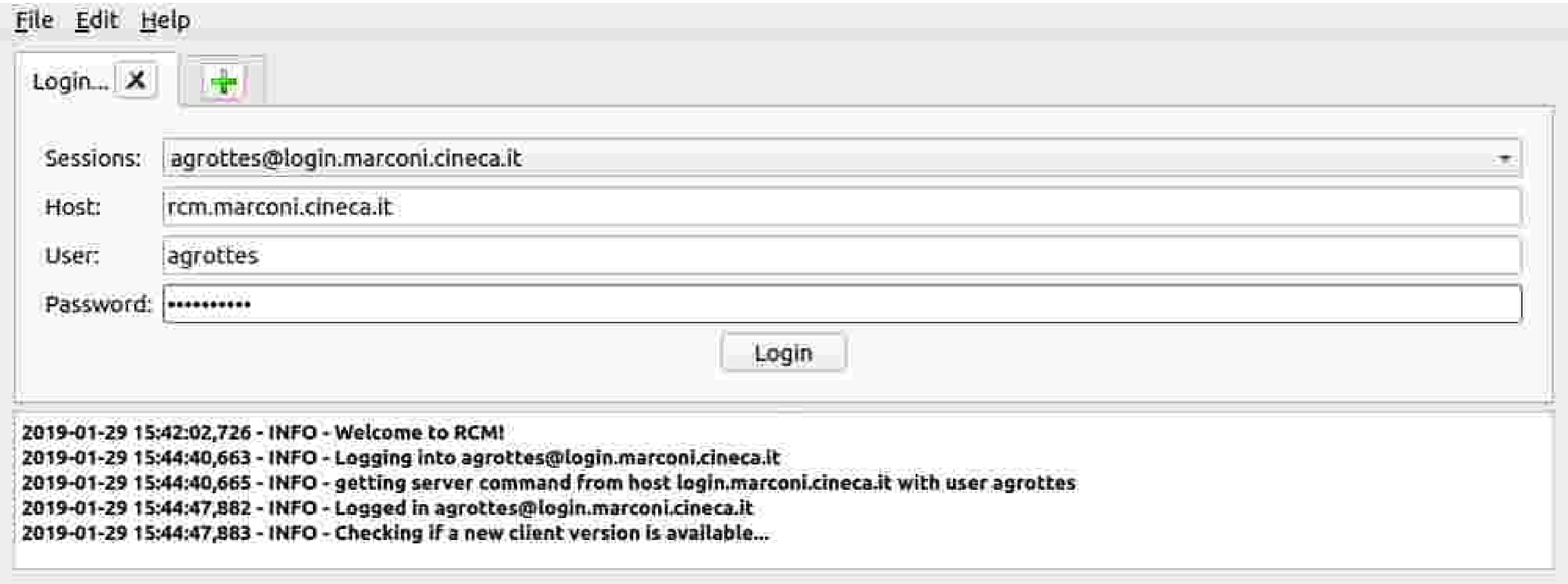
Remote Connection Manager

RCM (**R**emote **C**onnection **M**anager) is a tool developed by CINECA staff for allowing the opening of a graphical session inside our HPC clusters.

It runs by submitting a job on the budget-free “visual” queue, and starting an interactive session on special visual nodes

You can download here:
https://hpc-forge.cineca.it/svn/RemoteGraph/branch/multivnc/build/dist/Releases/?p=839
the version suited for your OS

Using RCM (I)



Sessions: agrottes@login.marconi.cineca.it

Host: rcm.marconi.cineca.it

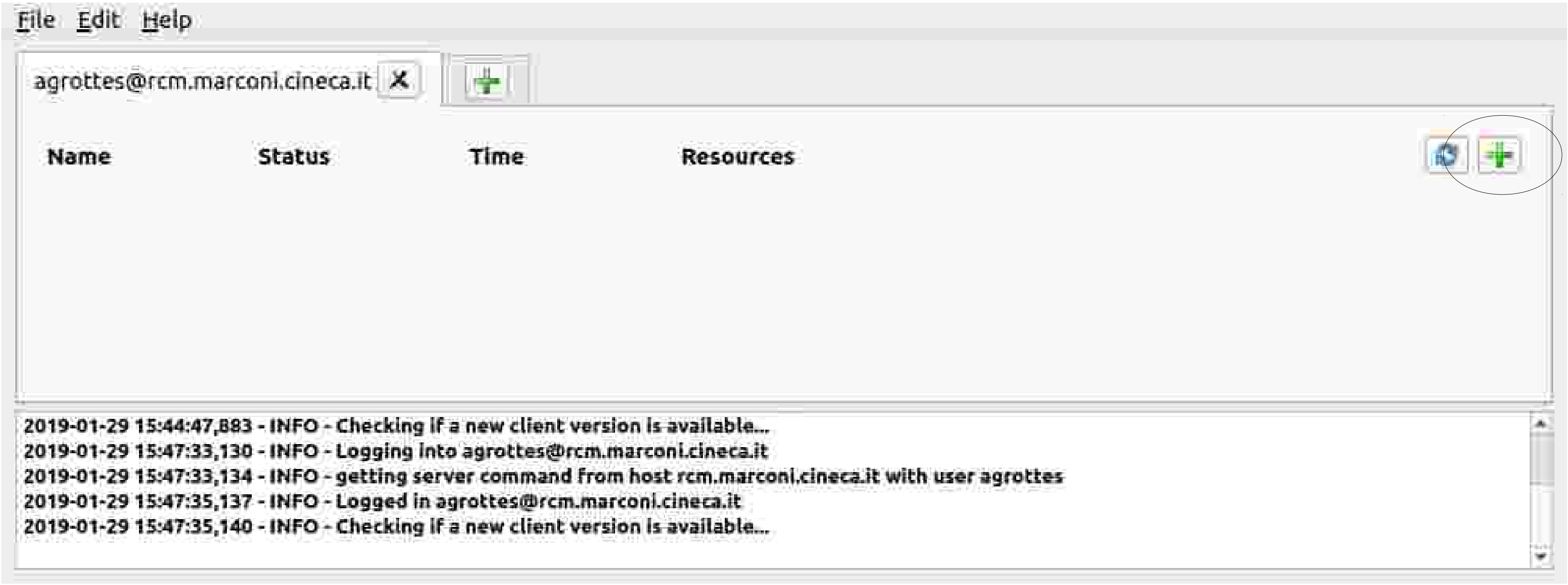
User: agrottes

Password: •••••••••••

```
2019-01-29 15:42:02,726 - INFO - Welcome to RCM!
2019-01-29 15:44:40,663 - INFO - Logging into agrottes@login.marconi.cineca.it
2019-01-29 15:44:40,665 - INFO - getting server command from host login.marconi.cineca.it with user agrottes
2019-01-29 15:44:47,882 - INFO - Logged in agrottes@login.marconi.cineca.it
2019-01-29 15:44:47,883 - INFO - Checking if a new client version is available...
```

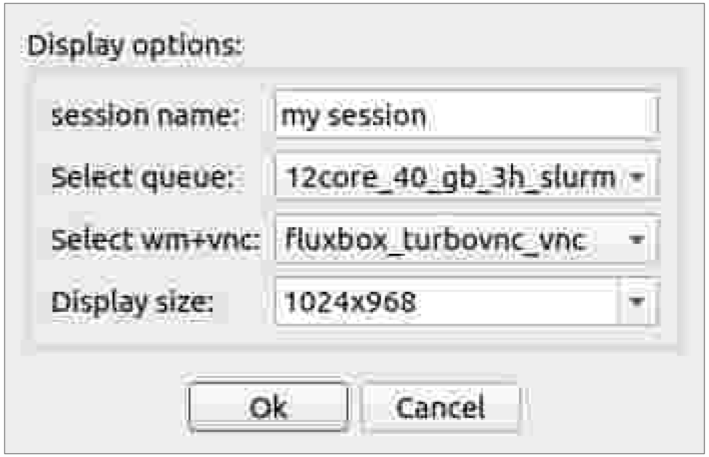
Login on the cluster via the proper RCM button (credentials: same as regular login)

Using RCM (II)



```
2019-01-29 15:44:47,883 - INFO - Checking if a new client version is available...
2019-01-29 15:47:33,130 - INFO - Logging into agrottes@rcm.marconi.cineca.it
2019-01-29 15:47:33,134 - INFO - getting server command from host rcm.marconi.cineca.it with user agrottes
2019-01-29 15:47:35,137 - INFO - Logged in agrottes@rcm.marconi.cineca.it
2019-01-29 15:47:35,140 - INFO - Checking if a new client version is available...
```

Open a new session by creating a new display. This will reserve some resources on the visual nodes (depending on the options selected). GALILEO has 2 nodes dedicated to visualization, if they are full the new display won't open (because the visualization job would be in queue)



Display options:

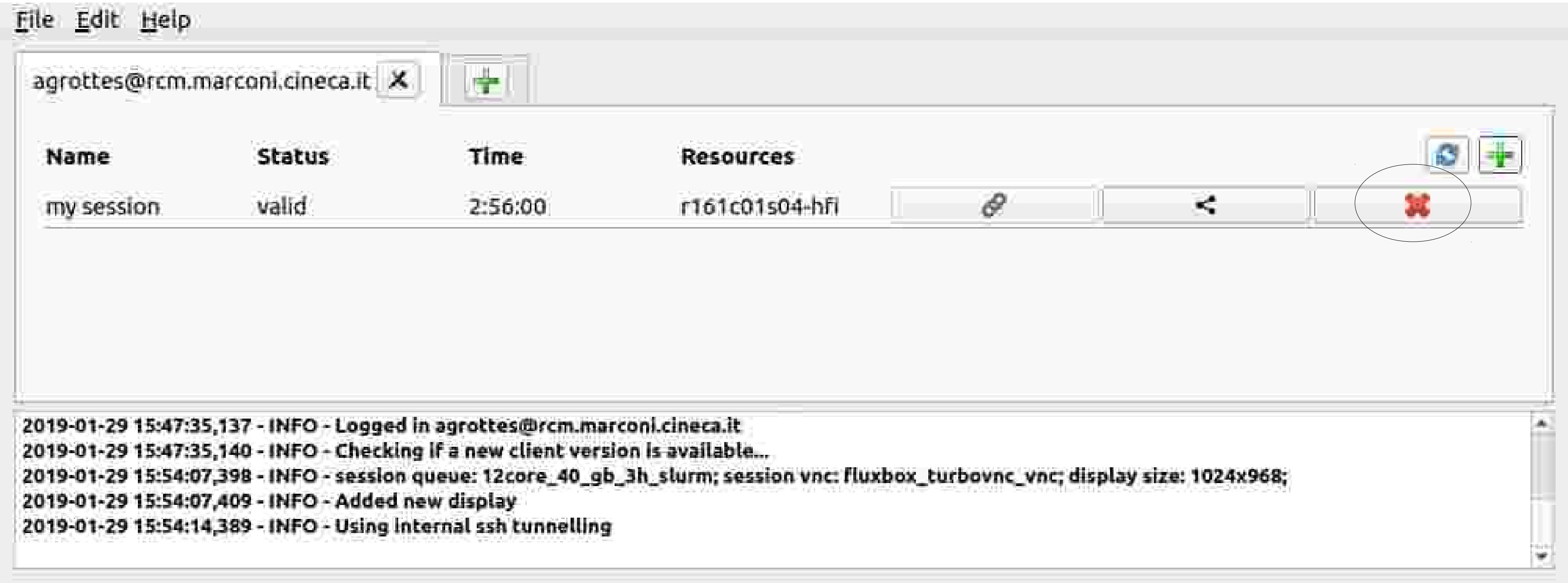
session name: my session

Select queue: 12core_40_gb_3h_slurm

Select wm+vnc: fluxbox_turbovnc_vnc

Display size: 1024x968

Using RCM (III)



Name	Status	Time	Resources
my session	valid	2:56:00	r161c01s04-hfi

```
2019-01-29 15:47:35,137 - INFO - Logged in agrottes@rcm.marconi.cineca.it
2019-01-29 15:47:35,140 - INFO - Checking if a new client version is available...
2019-01-29 15:54:07,398 - INFO - session queue: 12core_40_gb_3h_slurm; session vnc: fluxbox_turbovnc_vnc; display size: 1024x968;
2019-01-29 15:54:07,409 - INFO - Added new display
2019-01-29 15:54:14,389 - INFO - Using internal ssh tunnelling
```

This windows opens together with your display and is for displays management.
You can kill a graphical session with the “kill” button (this will also kill the job)

Outline

A first step:

- HPC infrastructures at CINECA
- GALILEO: system overview
- Login
- Work environment

Production environment

- Our first job!!
- Creating a job script
- Accounting and queue system
- SLURM commands

Programming environment

- Module system
- Serial and parallel compilation
- Interactive session

GPU environment

Graphical session with RCM

For further info...

- **Useful links and documentation**

Useful links and further documentation

Reference guide:
https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.3%3A+GALILEO+UserGuide
https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.5.1%3A+Batch+Scheduler+SLURM
https://wiki.u-gov.it/confluence/display/SCAIUS/UG2.4%3A+Data+storage+and+FileSystem
https://wiki.u-gov.it/confluence/display/SCAIUS/Remote+Visualisation

GPU computing *http://www.nvidia.com/object/GPU_Computing.html*

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