1 - 7 July, 2019

CINECA

PRACE

HPC Infrastructure at CINECA:

overview and hands-on

Alessandro Marani – a.marani@cineca.it
SuperComputing Applications and Innovation Department

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

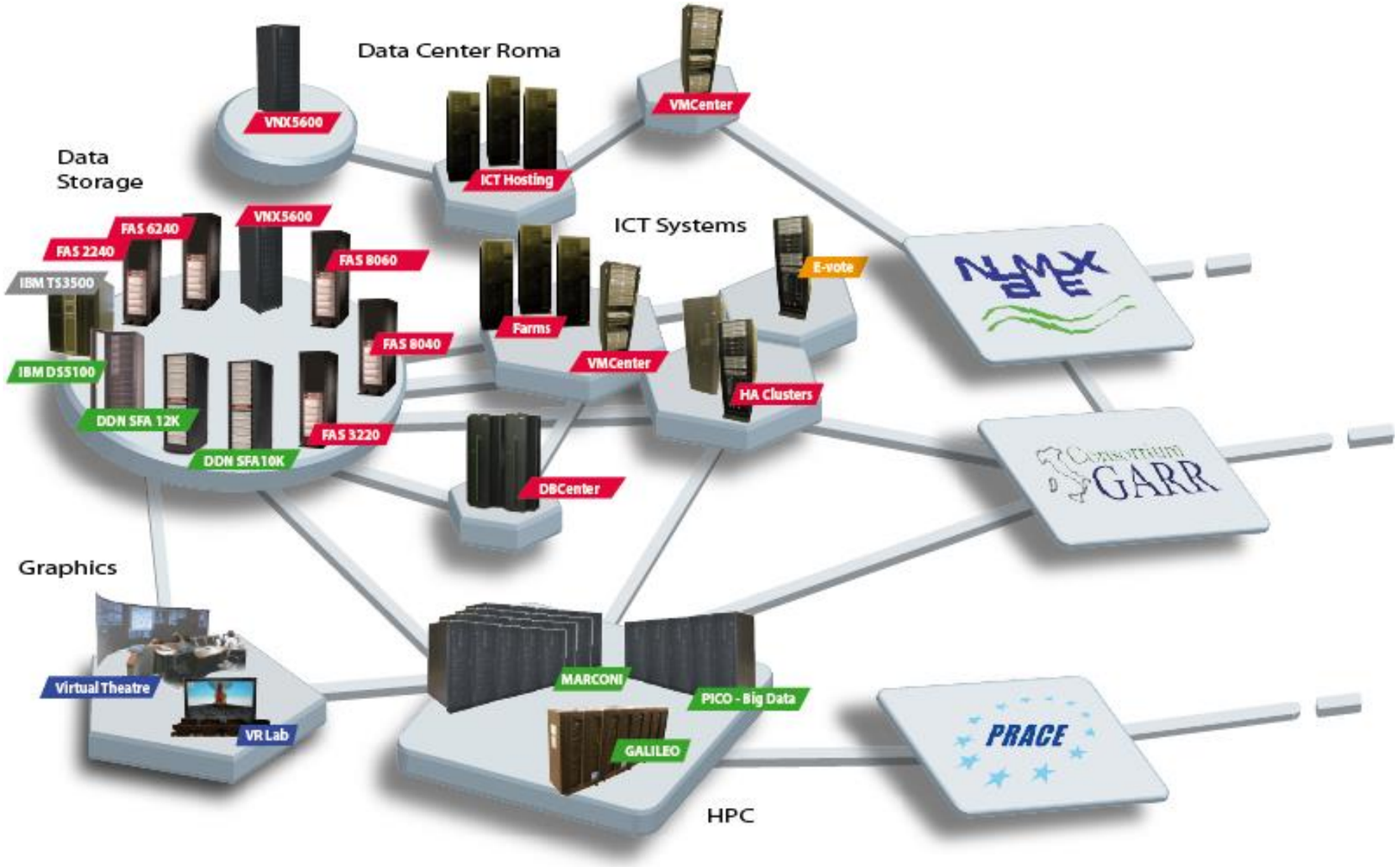
GPU environment

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, Lenovo SD530/S720AP, Intel Xeon Phi 7250 68C 1.4GHz/Platinum 8160, Intel Omni-Path	2016		348,000	10,384,900	18,816,000	**12**
Fermi - BlueGene/Q, Power BQC 16C 1.60GHz, Custom	2012		163,840	1,788,878	2,097,152	**7**
Marconi Intel Xeon - Lenovo NeXtScale nx360M5, Xeon E5-2697v4 18C 2.3GHz, Omni-Path	2016		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		50,232	684,252	1,103,066	**104**
Eurora - Eurotech Aurora HPC 10-20, Xeon E5-2687W 8C 3.100GHz, Infiniband QDR, NVIDIA K20	2013		2,688	100,900	175,667	**467**

- MARCONI (A2+A3) is currently the **21st** most powerful supercomputer in the World.
- In June of 2012 we reached **7th** place with FERMI (currently dismissed)
- MARCONI (A1 – currently dismissed), is approximately as powerful as FERMI was, but when it debuted on June 2016 it ranked at **46th** place
- 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.

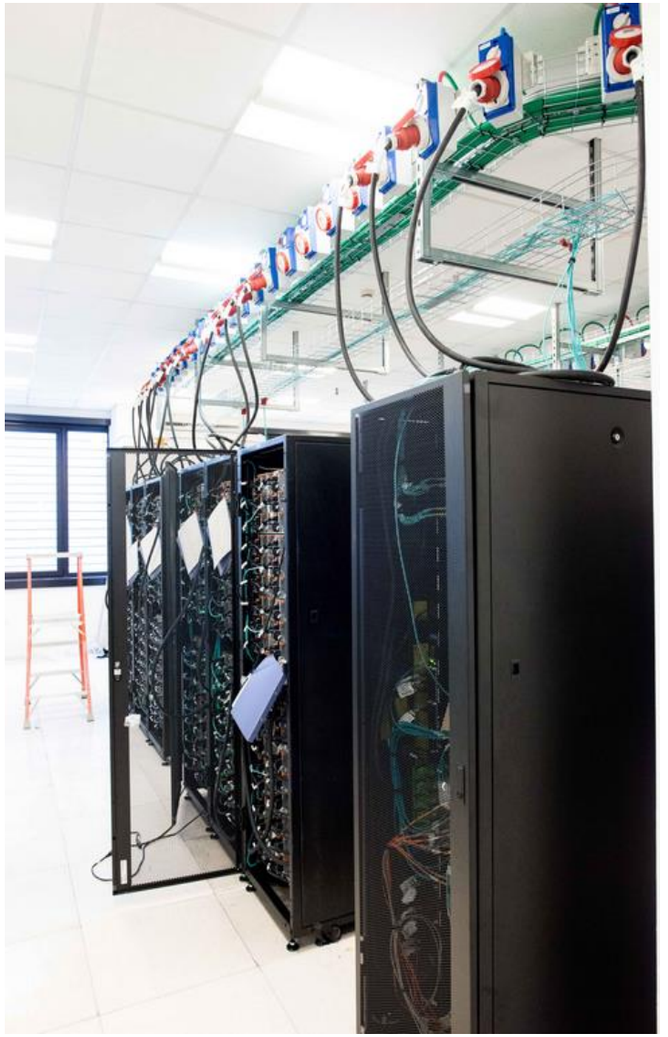
MARCONI

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 - DISMISSED
Model: Lenovo NeXtScale
Processor Type: Intel Broadwell, 2.3GHz
Peak Performance: 2 PFlop/s (presumed)

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 (presumed)

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



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.

SUMMER OF HPC 2019

PRACE

CINECA

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

SUMMER OF HPC 2019

PRACE

CINECA

GALILEO

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

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.

PRACE CINECA

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*

```
********************************************************************************
*  Welcome to  GALILEO  @ CINECA  -  NeXtScale cluster - CentOS 7.0!           *
*  (the system has been updated and opened to production on Apr 4, 2018)       *
*                                                                              *
*  360 Compute nodes (Intel Broadwell):                                        *
*  - 2*18-core Intel Xeon E5-2697 v4 @ 2.30GHz                                 *
*  - 128 GB RAM                                                                *
*                                                                              *
*  40 Compute nodes (Intel Haswell) with GPUs:                                 *
*  - 39 x 2*8-cores Intel Xeon E5-2630 v3 @ 2.40GHz + 2 nVidia  K80 GPUs       *
*  -  1 x 2*8-cores Intel Xeon E5-2630 v3 @ 2.40GHz + 2 nVidia V100 GPU        *
*  - 128 GB RAM                                                                *
*                                                                              *
*  Intel QDR (40Gb/s) Infiniband high-performance network                      *
*  SLURM 18.08                                                                 *
*                                                                              *
*  For a guide on GALILEO:                                                     *
*  https://wiki.u-gov.it/confluence/display/SCAIUS/UG3.3%3A+GALILEO+UserGuide  *
*  For support: superc@cineca.it                                               *
********************************************************************************
*     This system is in its final configuration and is in full-production      *
================================================================================
 IN EVIDENCE:
 - The modules of Intel and Intelmpi version 2017 have been moved from "base"
   profile to "archive". Consequently, all libraries and tools compiled with
   2017 Intel/Intelmpi and located under "base" and "advanced" profiles have
   been moved to "archive" profile
 - A new version of "module" is installed and based on profiles. Use the
   "modmap" command to identify the correct profile ("modmap -h" for help)
 - An automatic cleaning procedure on the $CINECA_SCRATCH is active: each day
   files older than 50 days will be removed
================================================================================
```

Check the **user guide**!

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 40 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 stands for "Simple Linux Utility for Resource Management"
open source and highly scalable job scheduling system

- Allocating access to resources
- Job starting, executing and monitoring
- Queue of pending jobs management

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=36
#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=36
#SBATCH --mem=10GB
Specify 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)

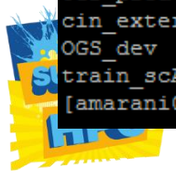
Accounting system

#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 ~]$
```



CINECA

Account for the School

The account provided for this school is “**train_sumhpc19**”
(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!

#SBATCH --account= train_sumhpc19

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

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

SLURM commands - 3

sinfo

```
sinfo -p <partition name>
sinfo -l
sinfo -N -l -p gll_usr_prod
```

Provides information about SLURM nodes and partitions

sacct

```
sacct OPTIONS  <job_id>
```

Displays accounting data for all jobs and job steps in the SLURM job accounting log or Slurm database.

Exercise 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
```

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 ./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

modmap -m <module_name>

list the module in all its available versions, indicating on which profile(s) it can be found

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 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-2018--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/1.8.5—gnu—4.9.2
```

Load a version of **IntelMPI**:

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

module load autoload intelmpi/2018--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.

So a total of 36 CPUs will be available per node. 18 of them will be MPI tasks, the others will be OpenMP threads (2 threads for each task).
Note that the environment variable **OMP_NUM_THREADS=2** has to be set inside the jobscript (otherwise, the unused CPUs will stay idle).

Execution line in job script

srun ./myprogram

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

WARNING:

In order to use srun, **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 by consequence you are consuming computing hours!

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

Exercise 2

1) Complete the first MPI exercise (MPI hello world) and compile it with the compiler (mpiicc or mpiifort) in the module intelmpi/2018--binary (default)
Find the exercise here:
https://gitlab.hpc.cineca.it/training/summer-school/tree/master/MPI/Exercise_01

2) Check with:
$ ldd <executable>
the list of required dynamic libraries.

3) Write "job.sh" (you can copy it from exercise 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 first 32 processes and then 4 processes for each select.

Exercise 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 much 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:

```
------------ /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  (up to 16 for GPU nodes!!)
#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!

Exercise 4

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

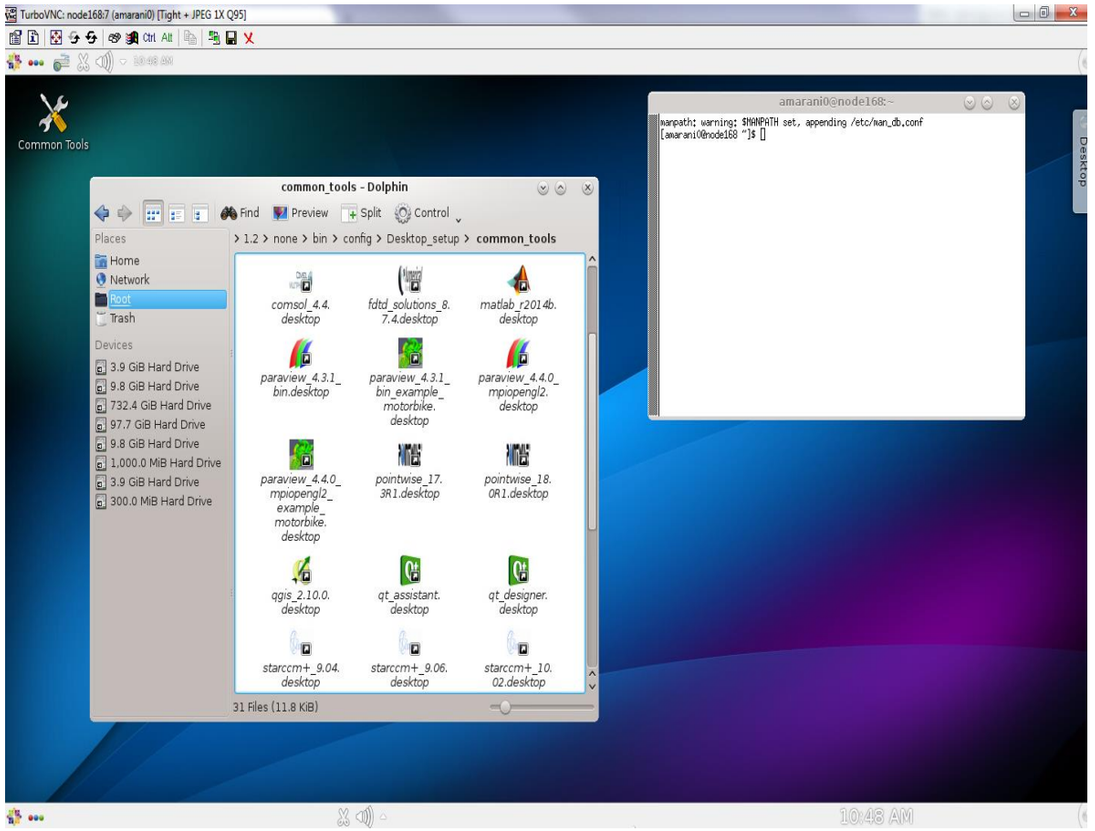
2) Write a job script for a serial execution (1 chunk, 1 cpu) that also asks for a gpu device

3) Run the job script

Graphical environment: 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**!

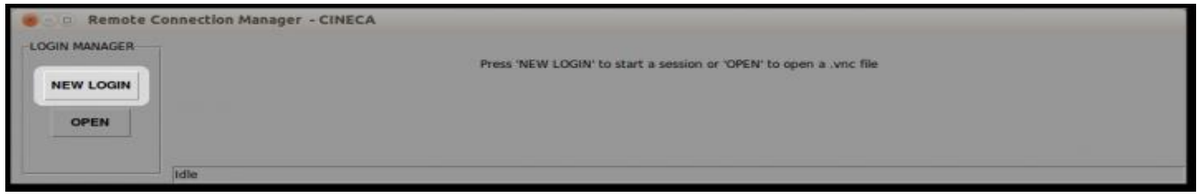
Remote Connection Manager

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

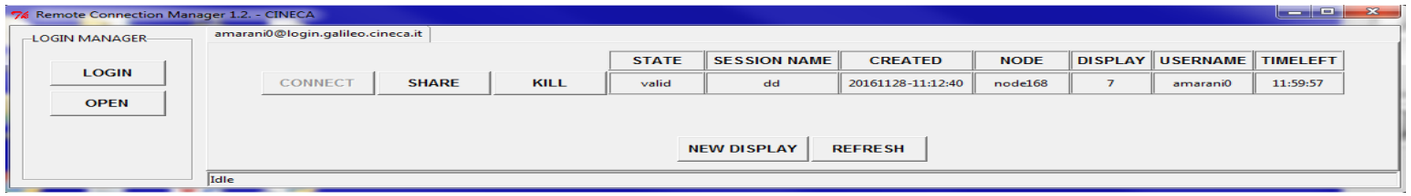
You can download here:
https://hpc-forge.cineca.it/svn/RemoteGraph/branch/multivnc/build/dist/Releases/?p=839
the version suited for your OS

Using RCM



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

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)



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 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.hpc@cineca.it