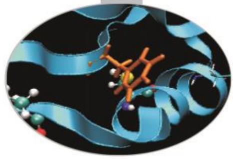
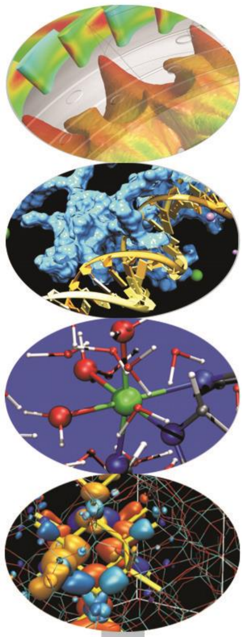


Parallel architectures and production environment



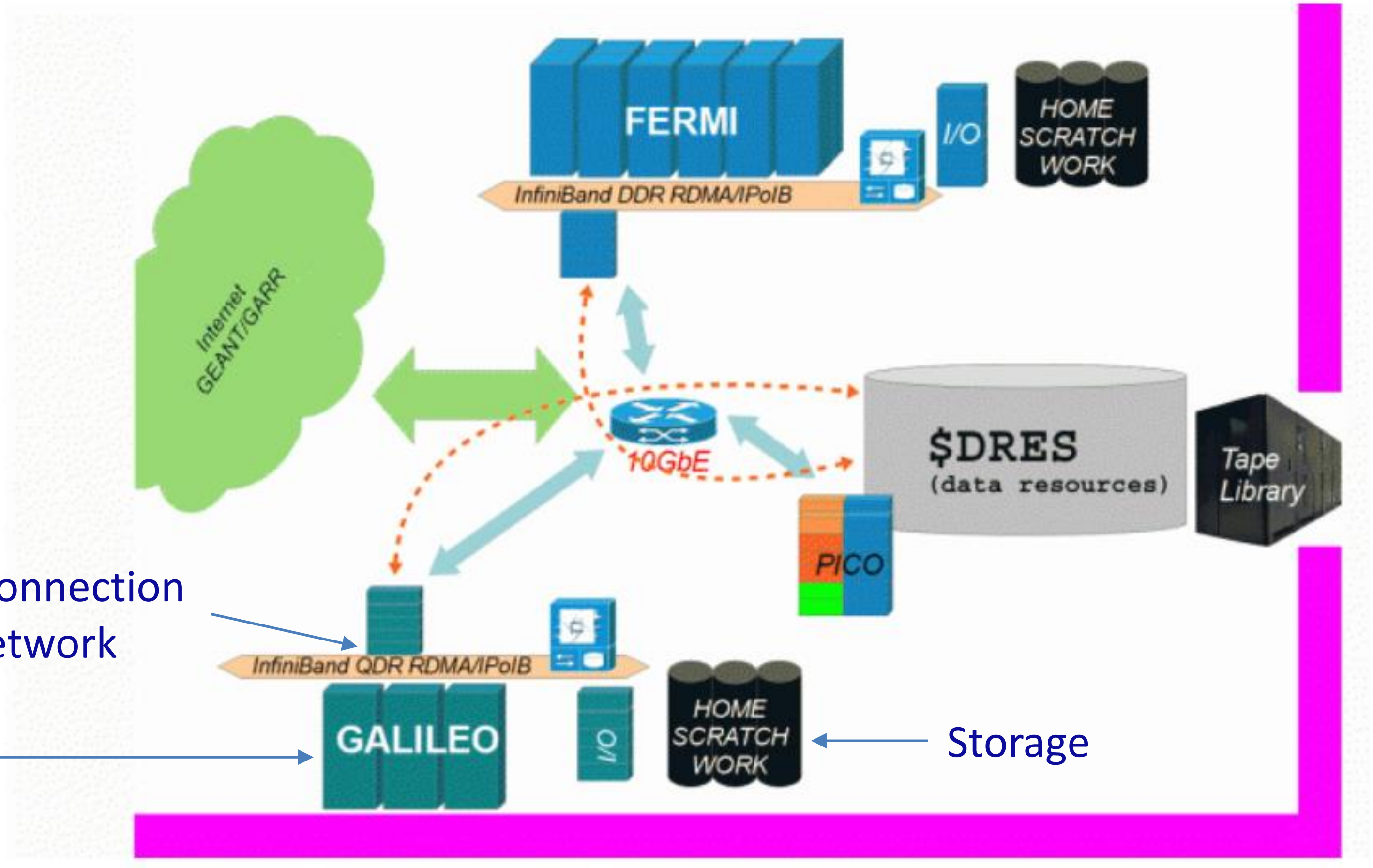
Introduction to Parallel Computing with MPI and OpenMP

P. Dagna

Segrate, November 2016

The HPC infrastructure

Cineca HPC infrastructure



Interconnection Network

CPUs

Storage

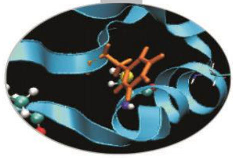
The HPC infrastructure

The HPC infrastructure

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

The HPC infrastructure



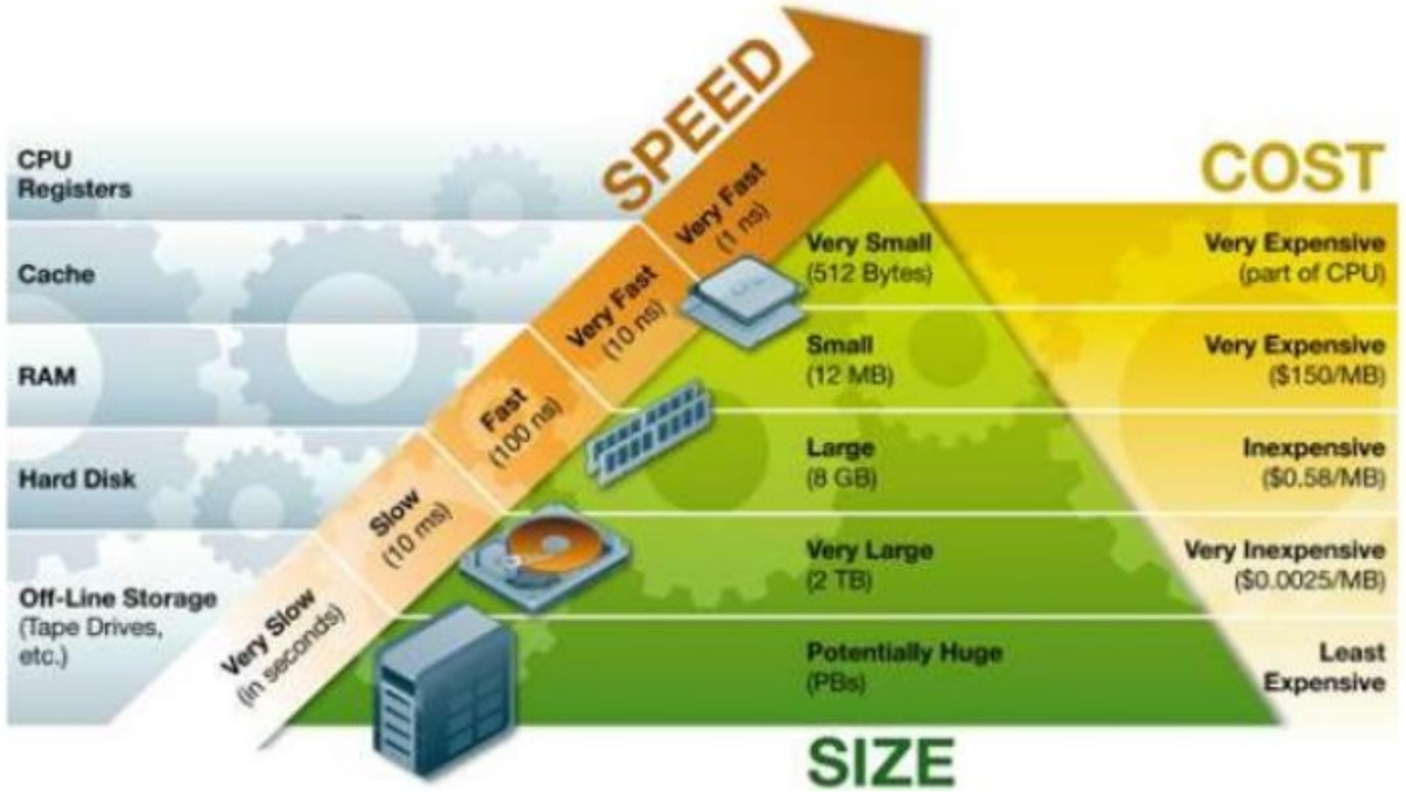
MARCONI – A1 CHARACTERISTICS

Model: Lenovo NeXtScale
Architecture: Intel OmniPath Cluster
Nodes: 1.512
Processors: 2 x 18-cores Intel Xeon E5-2697 v4 (Broadwell) at 2.30 GHz
Cores: 36 cores/node, 54.432 cores in total
RAM: 128 GB/node, 3.5 GB/core
Internal Network: Intel OmniPath
Disk Space: 17PB (raw) of local storage
Peak Performance: 2 PFlop/s

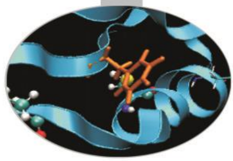
- **A2**: by the end of 2016 a new section will be added, equipped with the next-generation of the Intel Xeon Phi product family (Knights Landing), enabling an overall configuration of about 250 thousand cores with expected additional computational power of approximately 11Pflop/s.
- **A3**: finally, in July 2017, the system should reach a total computational power of about 20Pflop/s utilizing future generation Intel Xeon processors (Sky Lakes).

Memory hierarchy

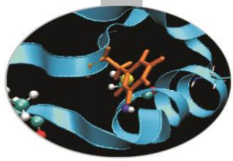
Storage organization and speed access can be thought of as a pyramid



In a shared working environment storage access changes depending to the load of the cluster



Production Environment



Login and working areas

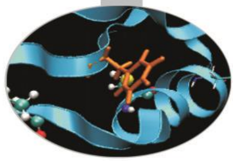
How to login

- 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!

Working 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.



Module System

All the optional software on the machine is made available through the "module" system

- provides a way to rationalize software and its environment variables

Modules are divided in profiles

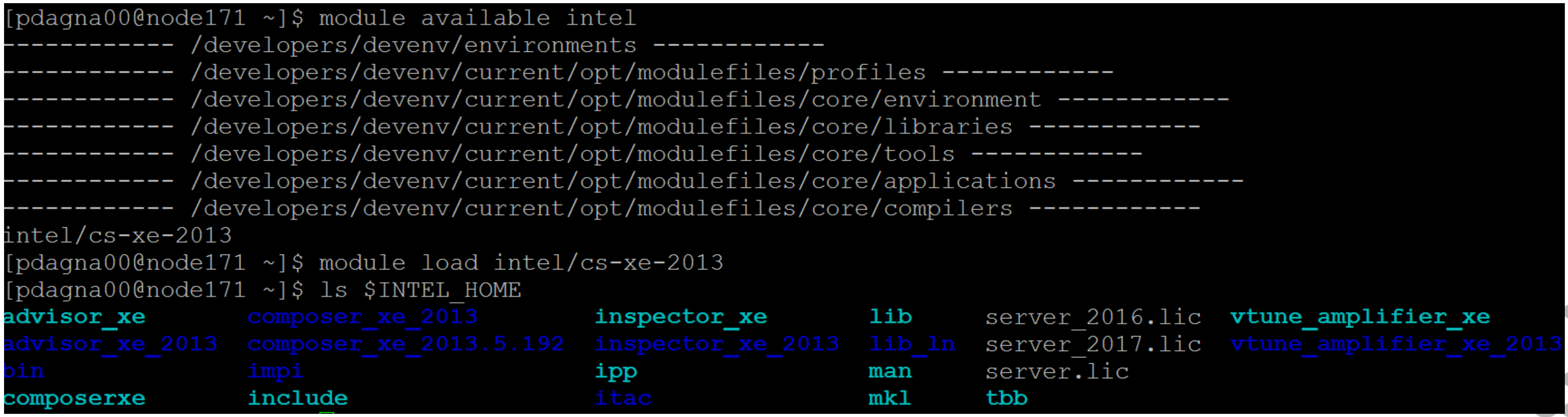
- profile/core (default - stable and tested modules)
- profile/extra (more recent versions non completely tested)

Profiles are divided in 4 categories

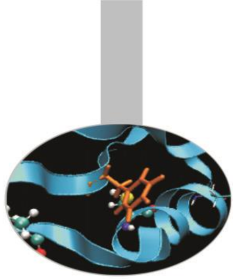
- compilers (GNU, intel, openmpi)
- libraries (e.g. LAPACK, BLAS, FFTW, ...)
- tools (e.g. Scalasca, GNU make, VNC, ...)
- applications (software for chemistry, physics, ...)

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

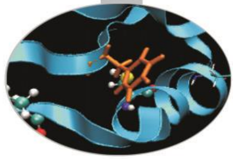


```
[pdagna00@node171 ~]$ module available intel
------------ /developers/devenv/environments ------------
------------ /developers/devenv/current/opt/modulefiles/profiles ------------
------------ /developers/devenv/current/opt/modulefiles/core/environment ------------
------------ /developers/devenv/current/opt/modulefiles/core/libraries ------------
------------ /developers/devenv/current/opt/modulefiles/core/tools ------------
------------ /developers/devenv/current/opt/modulefiles/core/applications ------------
------------ /developers/devenv/current/opt/modulefiles/core/compilers ------------
intel/cs-xe-2013
[pdagna00@node171 ~]$ module load intel/cs-xe-2013
[pdagna00@node171 ~]$ ls $INTEL_HOME
advisor_xe        composer_xe_2013          inspector_xe        lib      server_2016.lic  vtune_amplifier_xe
advisor_xe_2013   composer_xe_2013.5.192    inspector_xe_2013   lib_ln   server_2017.lic  vtune_amplifier_xe_2013
bin               impi                      ipp                 man      server.lic
composerxe        include                   itac                mkl      tbb
```



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

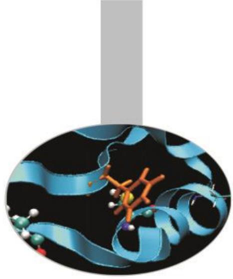


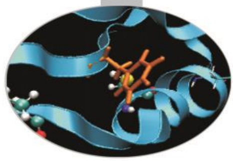
Launching Jobs

- As in every HPC cluster, users are allowed to run their own simulations by submitting “jobs” to the **compute nodes.**
- The **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.**
- The scheduler has a proprietary scripting language necessary to submit jobs

PBS Job script

- The scheme of a PBS job script is as follows:
  - #!/bin/bash
  - #PBS keywords
  - variables environment
  - execution line

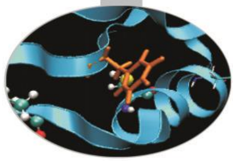




PBS Job script

- Example of parallel job:

```
#!/bin/bash
#PBS -N <jobname>
#PBS -o job.out
#PBS -e job.err
#PBS -l walltime=1:00:00
#PBS -l select=2:ncpus=16:mpiprocs=16:mem=10GB
#PBS -q <queuename>
cd $PBS_O_WORKDIR # points to the folder you are actually working
into
module load autoload openmpi
mpirun ./myprogram
```



PBS Job script

- PBS Keyword Analysis:

#PBS -N myname

- Defines the name of your job

#PBS -o job.out

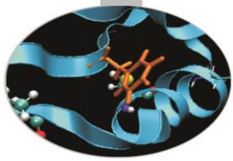
- Specifies the file where the standard output is directed (default=jobname.o<jobID>)

#PBS -e job.err

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

#PBS -l walltime=00:30:00

- Specifies the maximum duration of the job (queue dependency).



PBS Job script

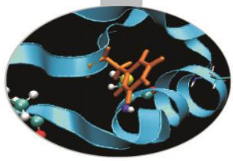
- PBS Keyword Analysis:

#PBS -l select=2:ncpus=16:mpiprocs=16: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)

#PBS -q <queuename>

- Specifies the queue requested for the job.



PBS Commands

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 <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.

qstat -f <job_id>

- Provides a long list of information 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 scheduled jobs by killing it