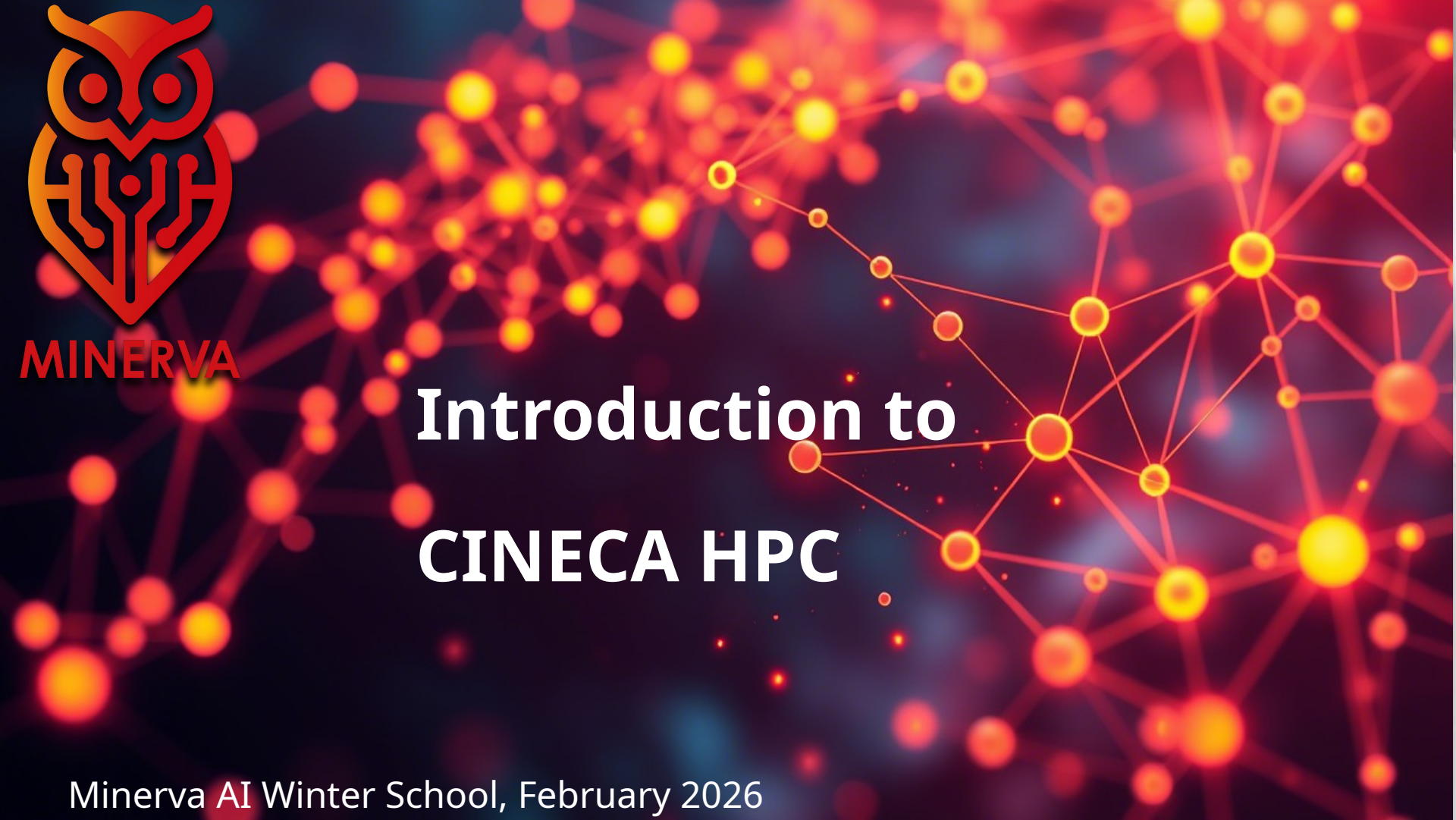
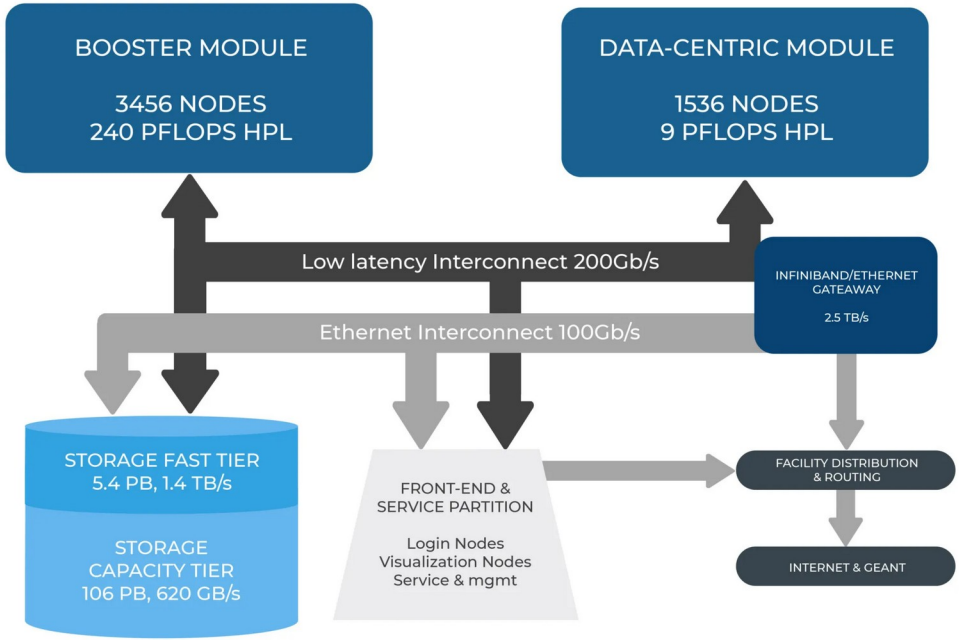

MINERVA
Introduction to
CINECA HPC
Minerva AI Winter School, February 2026

Outline

- **Leonardo infrastructure**
- Access HPC resources and filesystems
- Software environment
- Programming environment
- Production environment
- Data transfer
- Containers
- Final remarks

Leonardo infrastructure and login nodes



Atos BullSequana X430-E6

- Processors (dual-socket): **2x CPU Intel *Whitley* ICP06, 32 cores Intel *Ice Lake*** (64 cores/node), **2.4 GHz**
- RAM: 512 (16x32) GB RAM DDR4 3200 MHz
- Disk: 14 TB HDD
- **NO GPUs**

Booster
(GPU) partition

TOP 500 The List.

4th - November 2022

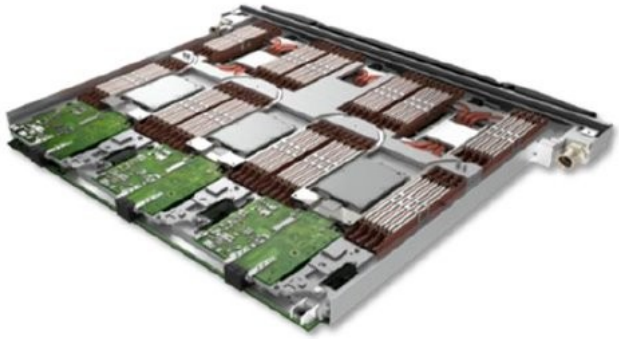
10th - June 2025

Atos BullSequana X2135 “Da Vinci” blade

- 3456 nodes: **lrdn[0001-3456]**
- Processors: **1 x CPU Intel Xeon 8358,**
  32 cores Intel Ice Lake, 2.6 GHz (one socket)
- Accelerators: **4 x NVidia custom Ampere GPU A100 SXM4 64 GB**
- RAM: 512 (8 x 64) GB DDR4 3200 MHz
- **Diskless**
- Internal network: NVLink 3.0 200 GB/s GPU-to-GPU, PCIe Gen4 GPU-to-CPU, each GPU has direct 100Gb/s connection to the InfiniBand network

Peak performance: about 306 PFlops

Data Centric and General Purpose (CPU-only) partition

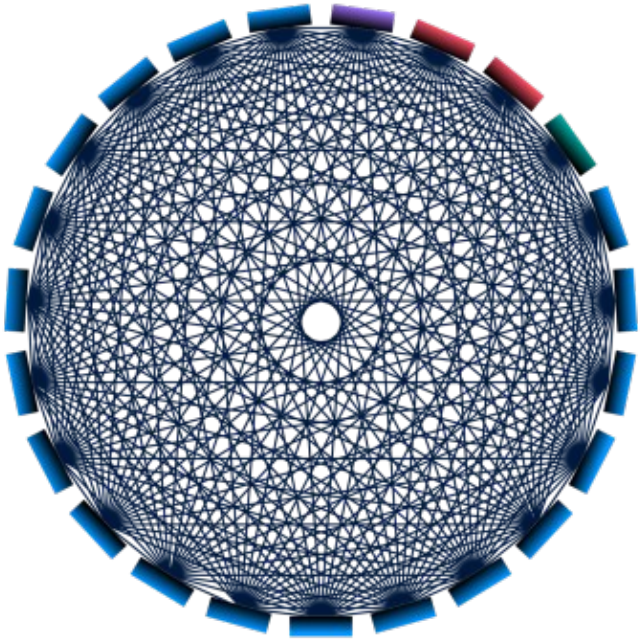


BullSequana X2140 three-node CPU Blade

- 1536 nodes: **lrdn[3457-4992]**
- Processors (dual-socket): **2x CPU Intel *Xeon* 8480+,**
  56 cores Intel *Sapphire Rapids* (**112 cores/node**),
  3.8 GHz (turbo enabled)
- RAM: 512 (16 x 32) GB DDR5 4800 MHz
- **Disk: 1x SSD 3.84 TB M.2 NVMe**
- Internal network: PCIe Gen5,
  1x port HDR100 100Gb/s network interface

Peak performance: about 13 PFlops

Inter-node network topology



Booster Module nodes
I/O cell
Data-Centric cells
Hybrid cell (Booster + Data-Centric nodes)

Dragonfly+ topology

based on **Nvidia Mellanox Infiniband HDR, bidirectional bandwidth of 200 Gb/s**
(shared between Leonardo Booster and DCGP)

- All nodes are divided into cells
- Non-blocking, two-layer **Fat Tree** within the cells
- All to all connection between cells

- **Adaptive routing algorithm**: SLURM will take care of the “best”-possible node allocations

Storage

Fast Tier

5.4 PB, 1.4 TB/s

NVMe storage (SSD disks)

- **HOME, PUBLIC, FAST SCRATCH**

Capacity Tier

106 PB, read 744 GB/s - write 620 GB/s

HDD disks

- **WORK, LARGE SCRATCH, DRES**

Outline

- Leonardo infrastructure
- **Access HPC resources and filesystems**
- Software environment
- Programming environment
- Production environment
- Data transfer
- Containers
- Final remarks

Become a new HPC user

- **Register on the UserDB Portal:** https://userdb.hpc.cineca.it/

- **Get associated to an active Project Account**

  → Principal Investigator (PI): we create the account and set you as PI on the UserDB

  → Collaborator: ask your PI to associate you to the account on the UserDB

- **Request the “HPC Access” on UserDB**

  → You will receive soon your credentials by mail

https://docs.hpc.cineca.it/general/users_account.html

How to apply for HPC resources (and get a Project Account)

ISCRA calls for users (PI) affiliated to an **Italian** Institute

for CINECA HPC systems (Leonardo Booster, Leonardo DCGP, G100, ADA cloud)

- **Class B**: large size account, duration of 12 months, 2 calls/year (November and May)
- **Class C:** small size account, duration of 9 months, continuous submission (10 selections per year)

- **Class D:** storage related to HPC simulations, duration 36 months

\+ **Test:** very small account, duration of 3 months, on demand (superc@cineca.it)

https://www.hpc.cineca.it/hpc-access/access-cineca-resources/iscra-projects/

iscra@cineca.it

How to apply for HPC resources (and get a Project Account)

EuroHPC calls for users (PI) affiliated to an **European** Institute

for EuroHPC systems (Leonardo Booster, Leonardo DCGP)

- **Extreme scale**
- **Regular scale**
- **Development scale**
- **Benchmark scale**

https://eurohpc-ju.europa.eu/access-our-supercomputers/eurohpc-access-calls_en

Access a cluster

The mandatory method to access CINECA HPC systems is via **two-factor authentication (2FA)**.

First time

- **Activate the 2FA**: authenticate on our **Identity Provider** at https://sso.hpc.cineca.it using username and password you use to connect to CINECA clusters.
  → You will need an **app to generate authentication codes** (e.g. Google Authenticator)
- **Install and configure the smallstep client** (depending on your OS)

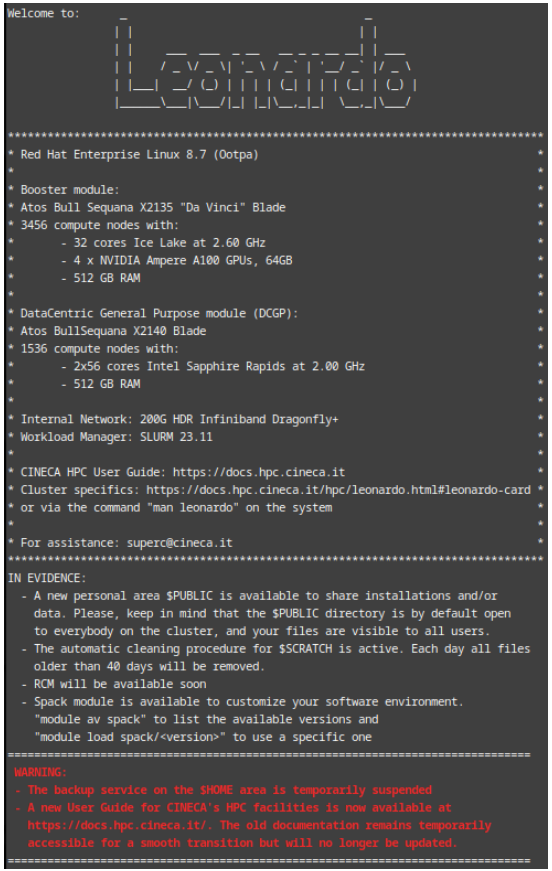
Any access to the cluster

- **Request the ssh certificate** to our Identity Provider via the smallstep client
  → A web page will open on the **browser** and you will be asked to insert a One-Time Password (OTP) from the app
  → ***Valid for 12 hours***
- **Access to the cluster via ssh:**

```
$ ssh <username>@login.<cluster>.cineca.it
```

https://docs.hpc.cineca.it/general/access.html

Access a cluster



For this course, you will access Leonardo with temporary training usernames with **password**.

$ ssh <username>@login.leonardo.cineca.it

Message of the day

- ➔ Short system description
- ➔ System status
- ➔ “In evidence” messages
- ➔ “Important” messages

  (e.g. scheduled maintenances)

Filesystems

$HOME

- 50 GB per user
- user specific
- permanent
- daily backup (soon)

$PUBLIC

- 50 GB per user
- user specific (permissions **755**)
- permanent
- **no** backup

$SCRATCH

- no quota
- user specific
- temporary (data removed after 40 days)
- **no** backup

Filesystems

$HOME

- 50 GB per user
- user specific
- permanent
- daily backup (soon)

$PUBLIC

- 50 GB per user
- user specific (permissions **755**)
- permanent
- **no** backup

$SCRATCH

- no quota
- user specific
- temporary (data removed after 40 days)
- **no** backup

$WORK

- quota per account (default 1 TB)
- account specific
- permanent
- **no** backup

$FAST

- like $WORK
- **fast I/O**
- only on Leonardo

Filesystems

$HOME
- 50 GB per user
- user specific
- permanent
- daily backup (soon)

$PUBLIC
- 50 GB per user
- user specific (permissions **755**)
- permanent
- **no** backup

$SCRATCH
- no quota
- user specific
- temporary (data removed after 40 days)
- **no** backup

$WORK
- quota per account (default 1 TB)
- account specific
- permanent
- **no** backup

$FAST
- like $WORK
- **fast I/O**
- only on Leonardo

$TMPDIR
- local on nodes
- job specific

DRES
- long storage on demand
- shared among accounts and platforms (not Leonardo)

All the filesystems are based on **Lustre**

→ Check your areas, disk usage and quota: **$ cindata**

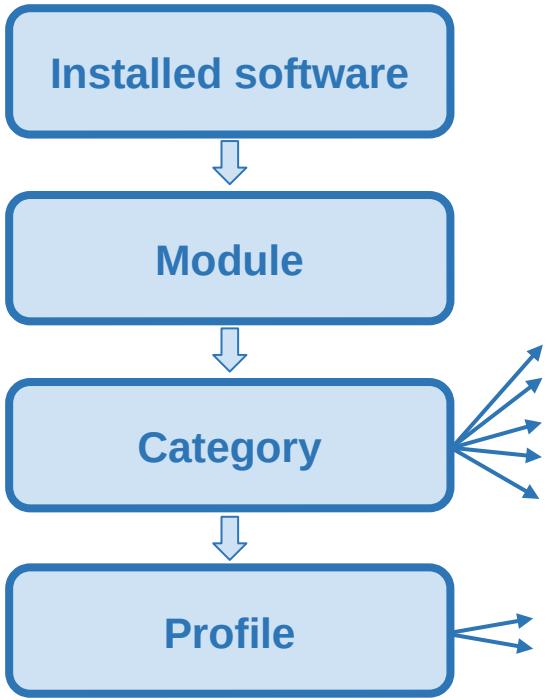
https://docs.hpc.cineca.it/hpc/hpc_data_storage.html

Outline

- Leonardo infrastructure
- Access HPC resources and filesystems
- **Software environment**
- Programming environment
- Production environment
- Data transfer
- Containers
- Final remarks

Module environment

Any available software is offered on the clusters in a module environment.
The modules are organized in functional categories and collected in different profiles.



Compilers
Libraries
Tools
Applications
Data

Base is the default profile:
automatically loaded after login,
containing basic modules
for programming activities

Programming (**base**, **advanced**): compilation, profiling, debugging…
Domain (**chem-phys, astro, bioinf…**): production activities

Module environment

$ module av

```
------------------------------------------------------ /leonardo/prod/opt/modulefiles/profiles ------------------------------------------------------
profile/archive  profile/base       profile/chem-phys  profile/geo-inquire  profile/meteo    profile/spoke7
profile/astro    profile/candidate  profile/deeplrn    profile/lifesc       profile/quantum  profile/statistics
```

```
---------------------------------------------- /leonardo/prod/opt/modulefiles/base/libraries ----------------------------------------------
adios/1.13.1--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0          metis/5.1.0--gcc--12.2.0
adios/1.13.1--openmpi--4.1.6--gcc--12.2.0-cuda-12.1                  metis/5.1.0--oneapi--2023.2.0
blitz/1.0.2--gcc--12.2.0                                             nccl/2.19.1-1--gcc--12.2.0-cuda-12.1
blitz/1.0.2--oneapi--2023.2.0                                        nccl/2.19.3-1--gcc--12.2.0-cuda-12.1
boost/1.83.0--gcc--12.2.0                                            netcdf-c/4.9.2--gcc--12.2.0
boost/1.83.0--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0-atomic   netcdf-c/4.9.2--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0
boost/1.83.0--oneapi--2023.2.0                                       netcdf-c/4.9.2--oneapi--2023.2.0
boost/1.83.0--openmpi--4.1.6--gcc--12.2.0                            netcdf-c/4.9.2--openmpi--4.1.6--gcc--12.2.0
boost/1.83.0--openmpi--4.1.6--nvhpc--23.11                           netcdf-c/4.9.2--openmpi--4.1.6--nvhpc--23.11
cfitsio/4.3.0--gcc--12.2.0                                           netcdf-fortran/4.6.1--gcc--12.2.0
```

```
------------------------------------------------ /leonardo/prod/opt/modulefiles/base/tools ------------------------------------------------
anaconda3/2023.09-0                jube/2.4.3                spack/0.21.0-68a
cintools/1.0                       maven/3.8.4               spack/DCGP_0.21.0
```

```
---------------------------------------------- /leonardo/prod/opt/modulefiles/base/compilers ----------------------------------------------
cuda/12.1  gcc/12.2.0           intel-oneapi-compilers/2023.2.1     nvhpc/23.11  perl/5.36.0--gcc--8.5.0  python/3.10.8--gcc--8.5.0
cuda/12.3  gcc/12.2.0-cuda-12.1 llvm/14.0.6--gcc--12.2.0-cuda-12.1  nvhpc/24.3   perl/5.38.0--gcc--8.5.0  python/3.11.6--gcc--8.5.0
```

Almost all the modules on Leonardo have been installed with **Spack**, and they report the Spack package name.

Module environment

$ module load profile/astro
$ module av

Loaded profiles
are **added** to the environment

```
------------------------------------------------- /leonardo/prod/opt/modulefiles/profiles -------------------------------------------------
profile/archive  profile/base       profile/chem-phys  profile/geo-inquire  profile/meteo    profile/spoke7
profile/astro    profile/candidate  profile/deeplrn    profile/lifesc       profile/quantum  profile/statistics

------------------------------------------------ /leonardo/prod/opt/modulefiles/astro/libraries ------------------------------------------------
cfitsio/4.3.0--gcc--12.2.0
```

$ module show <module_name>/<version> → Print information about the module, such as dependencies, paths

$ module help <module_name>/<version> → Print the help of the software, its brief description and examples of the use

https://docs.hpc.cineca.it/hpc/hpc_enviroment.html

Module environment

$ modmap -m <module_name> → Detect all profiles, categories and modules available (e.g. different releases)

$ module load <profile>

$ module load <module_name>/<version> → all the dependencies are automatically loaded

$ module list → List all the profiles and modules loaded so far

Important!

You will find **modules compiled to support GPUs and modules suitable only for CPUs**.
You can check the compiler in the full name of the module, where the version is specified (e.g. gromacs/2022.3--intel-oneapi-mpi--2021.10.0--oneapi–2023.2.0). Remind that modules compiled with gcc, nvhpc, cuda should be used only on the Booster partition (and g100 if you use the GPUs), while modules compiled with intel oneapi are suitable for running on the DGCP partition (and g100 if you do not use GPUs).

https://docs.hpc.cineca.it/hpc/hpc_enviroment.html

Install new software

In case you don't find a software, you can choose to install it by yourself.

- **Install without sudo permissions**
- **Install with pip in virtual env**

  $ module load python/3.11.7

  $ python3 -m venv <env_name>

  $ source <env_name>/bin/activate

- **Install with Spack**

Write to superc@cineca.it if you need help on the installation or if you would like to request a new module.

Install with Spack

A “Spack” environment is provided by the **package manager Spack** and available as **module**.
$ module load spack/<version>

- **setup-env.sh** file is sourced
- **$SPACK_ROOT** is initialized
- **spack command** is added to your PATH, and some nice command line integration tools as well
- Folder **/spack-<version>** is created into your **$PUBLIC** area (on Leonardo, and $WORK on the other clusters) and it contains some subfolders created and used by spack during the phase of the packages installation:
  - sources cache: **/cache**
  - software installation root: **/install**
  - modulefiles location: **/modules**
  - user scope: **/user_cache**

https://docs.hpc.cineca.it/hpc/hpc_enviroment.html#spack

Install with Spack

Some fundamental Spack commands

$ spack list <package_name> → Check if the package is available for installation with Spack

$ spack info <package_name> → Show available versions, building variants and dependencies

$ spack spec -Il <package_name> → Show version, compiler, dependencies, building variants with which the package will be installed (-Il for installation status and *hash*)
→ options can be specified

e.g. $ spack spec -Il scorep

```
Concretized
--------------------------------
 -   ijz2tvy  scorep@7.1%gcc@11.3.0+mpi+papi~pdt~shmem~unwind build_system=autotools arch=linux-rhel8-icelake
 -   5bnn3tg      ^cubelib@4.6%gcc@11.3.0 build_system=autotools arch=linux-rhel8-icelake
```

$ spack install <package_name> → Install the package
→ options as spec command

$ spack load <package_name> → Load the package installed to use it (you can also create a **module**)

Outline

- Leonardo infrastructure
- Access HPC resources and filesystems
- Software environment
- **Programming environment**
- Production environment
- Data transfer
- Containers
- Final remarks

Programming environment

Compilers and MPI libraries are available as modules in profile/base.
Use the ones suitable for the architecture!

Check with commands
modmap -m,
module av,
module show,
module help,
and **man**

Compilers

- **GCC** (GNU compilers: gcc, g++, gfortran)
- **NVHPC** (ex hpc-sdk, ex PGI + CUDA → NVIDIA compilers: nvc, nvc++, nvcc, nvfortran)
- **CUDA**
- **INTEL ONEAPI** (Intel compilers: icc, icpc, ifort. Oneapi compilers: icx, icpx, ifx) → **no** Nvidia GPU support

MPI libraries

- **OpenMPI** (GNU/NVHPC compilers)
- **Intel Oneapi MPI** (Intel compilers) → **no** CUDA-aware

https://docs.hpc.cineca.it/hpc/hpc_enviroment.html#compilers

Outline

- Leonardo infrastructure
- Access HPC resources and filesystems
- Software environment
- Programming environment
- **Production environment**
- Data transfer
- Containers
- Final remarks

Login and compute nodes

CINECA HPC clusters are shared among many users, so **a responsible use is crucial!**

Login nodes

- Interactive runs on login nodes are strongly discouraged and should be limited to short test runs
  → ***10 minutes cpu-time limit***
- Avoid running large and parallel applications on login nodes
- ***No GPUs on login nodes***

Compute nodes

- Long production jobs should be submitted on compute nodes using the **scheduler** → **SLURM**
- Jobs can be submitted in two main ways: via **batch mode** and via **interactive mode**
- **Nodes shared**, but the allocated resources (cores, GPUs, RAM, $TMPDIR) are assigned in an exclusive way

Submit jobs with SLURM

Batch mode

- Write a batch script like the example
- Launch the batch script
  $ sbatch [options] start.sh
- The job is queued and scheduled

```
#!/bin/bash

#SBATCH --nodes=1                       # nodes
#SBATCH --ntasks-per-node=4             # tasks per node
#SBATCH --cpus-per-task=8               # cores per task
#SBATCH --gres=gpu:4                    # GPUs per node
#SBATCH --mem=494000                    # mem per node (MB)
#SBATCH --time=00:30:00                 # time limit (d-hh:mm:ss)
#SBATCH –account=<account_name>         # account
#SBATCH --partition=<partition_name>    # partition name
#SBATCH --qos=<qos_name>                # quality of service

module load <module_name>

 srun my_application
```

Submit jobs with SLURM

Batch mode

- Write a batch script like the example
- Launch the batch script
  $ sbatch [options] start.sh
- The job is queued and scheduled

shell →

#SBATCH directives →
(also contracted syntax,
e.g. -N for --nodes)

Loading modules and setting variables →

Launch executable →
(for parallel applications, use **srun** or **mpirun**)

```
#!/bin/bash

#SBATCH --nodes=1                       # nodes
#SBATCH --ntasks-per-node=4             # tasks per node
#SBATCH --cpus-per-task=8               # cores per task
#SBATCH --gres=gpu:4                    # GPUs per node
#SBATCH --mem=494000                    # mem per node (MB)
#SBATCH --time=00:30:00                 # time limit (d-hh:mm:ss)
#SBATCH --account=tra26_minwinsc        # account
#SBATCH --partition=boost_usr_prod      # partition name
#SBATCH --qos=boost_qos_dbg             # quality of service

module load <module_name>

srun my_application
```

Submit jobs with SLURM

Interactive mode

- Ask for the needed resources with the same **SLURM directives** with srun or salloc
- The job is queued and scheduled but, when executed, the standard input, output, and error streams are connected to the **terminal session** from which srun or salloc were launched
- **Run your application from that prompt**
- Exit from the terminal session: **$ exit**

Non MPI programs (one process using one or more GPUs)

$ srun -N 1 --ntasks-per-node=8 --cpus-per-task=4 --gres=gpu:4 -t 00:30:00 -p boost_usr_prod -q boost_qos_dbg -A tra26_minwinsc --pty /bin/bash

The session starts on the compute node: **[username@lrdn0053 ~]$**

Also **MPI programs** (using one or more GPUs)

$ salloc -N 1 --ntasks-per-node=8 --cpus-per-task=4 --gres=gpu:4 -t 00:30:00 -p boost_usr_prod -q boost_qos_dbg -A tra26_minwinsc

A new session starts on the login node: **[username@login14 ~]$**

Submit jobs with SLURM

#SBATCH –account= tra26_minwinsc or **-A tra26_minwinsc**

Specifies the account with a **budget** of core-hours available to run jobs.
As a user, you have access to a limited number of core-hours to spend. They are not assigned to User Accounts, but to **Project Accounts**, and are shared among users on the same project (i.e. your research partners).

On Leonardo, you can check the status of your accounts with

$ saldo -b → **Leonardo Booster**

$ saldo -b --dcgp → **Leonardo DCGP**

Accounts defined on Booster can only be used on **Booster partition** (boost_usr_prod),
and accounts defined on DCGP can only be used on **DCGP partition** (dcgp_usr_prod)**.**

Submit jobs with SLURM

#SBATCH --partition=boost_usr_prod or **-p boost_usr_prod**
Specifies the “partition”, i.e. the specific set of nodes among which your job can search for resources.

#SBATCH --qos=boost_qos_dbg or **-q boost_qos_dbg**
The Quality of Service is used to modify the limits of a partition and its priority, or to access selected partitions.

Partition	QOS	#Cores/#GPU per job	Walltime	Max Nodes/cores/GPUs/user	Priority
boost_usr_prod	boost_qos_dbg	2 nodes	00:30:00	2 nodes / 64 cores / 8 GPUs	80

Submit jobs with SLURM

Only on Leonardo “diskful” nodes, it’s possible to increase the space of the **$TMPDIR** area.
Remind that the area is **local to nodes**, and **job specific** (i.e. “temporary”): created at the begging of a job and deleted at its end, and accessible only by the user who launched the job.

Specify the space on $TMPDIR=/tmp (default=10GB):

#SBATCH --gres=tmpfs:200GB

on the local disks on **lrd_all_serial** nodes (max 1 TB) and **dcgp_usr_prod** compute nodes (max 3 TB).

It is possible to use the $TMPDIR=/scratch_local space also on the **login** nodes (14 TB shared among users, remove your files once they are no more necessary).

On the diskless **boost_usr_prod** compute nodes, the $TMPDIR=/tmp area is hosted on the RAM, with a fixed size of 10 GB (no increase is allowed, and the gres=tmpfs resource is disabled).

Remind that for the DCGP jobs the requested amount of gres=tmpfs resource contributes to the consumed budget, changing the number of accounted equivalent core hours.

https://docs.hpc.cineca.it/hpc/leonardo.html#file-systems-and-data-managment

Resources per Booster node

Each node → max resources you can request per Booster node

- 32 cores (**cpus**) → **ntasks-per-node * cpus-per-tasks ≤ 32**
- 4 GPUs (**gres=gpus**)
- 494000 MB of RAM (**memory**)

⇨ The **accounting** considers

- the requested number of CPUs
- the requested number of GPUs
- the requested memory on RAM

and calculates the **number of equivalent cores** → it takes the **maximum** among

- number of cpus
- number of GPUs * 8 (= number of GPUs * cores-per-node / GPUs-per-node)
- memory / memory-per-core (= requested memory / memory-per-node * cores-per-node)

Resources per DCGP node

Each node → max resources you can request per DCGP node

- 112 cores (**cpus**) → **ntasks-per-node * cpus-per-tasks ≤ 112**
- 494000 MB of RAM (**memory**)
- 3 TB of temporary local memory on $TMPDIR (**gres=tmpfs**)

⇨ The **accounting** considers

- the requested number of CPUs
- the requested memory on RAM
- the requested memory on $TMPDIR

and calculates the **number of equivalent cores** → it takes the **maximum** among

- number of cpus
- memory / memory-per-core (= requested memory / memory-per-node * cores-per-node)
- tmpfs / tmpfs-per-core (= requested tmpfs / tmpfs-per-node * cores-per-node)

Example of job submission

Prepare a jobscript.sh file:

```
#!/bin/bash

#SBATCH --nodes=1                    # nodes
#SBATCH --ntasks-per-node=4          # tasks per node
#SBATCH --cpus-per-task=8            # cores per task
##SBATCH --gres=gpu:4                # GPUs per node
#SBATCH --mem=494000                 # mem per node (MB)
#SBATCH --time=00:30:00              # time limit (d-hh:mm:ss)
#SBATCH --account=tra26_minwinsc     # account
#SBATCH --partition=boost_usr_prod   # partition name
#SBATCH --qos=boost_qos_dbg          # quality of service

module load python/3.11.7
module load openmpi/4.1.6--gcc--12.2.0-cuda-12.2

srun python hello_world_area.py
```

hello_world_area.py

```
import math

radius = 2

area = math.pi * (radius**2)

print("Hello, world!")
print(f"In addition, area of your circle
is: {area}")
```

Monitor your jobs with SLURM

$ squeue -u <username> or **$ squeue --me**

Shows the list of all your scheduled jobs, along with their status (pending, running, closing, …).
Also, shows you the **jobID** required for other SLURM commands.

$ 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 has not started yet and, if it is scheduled with top priority, you will get an **estimated start time**.

$ scancel <job_id>

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

$ sinfo (e.g. **$ sinfo -o "%10D %a %20F %P"**)

Provides information about SLURM nodes and partitions.

$ sacct <options> <job_id> (e.g. **$ sacct -Bj <job_id>**)

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

Outline

- Leonardo infrastructure
- Access HPC resources and filesystems
- Software environment
- Programming environment
- Production environment
- **Data transfer**
- Containers
- Final remarks

Data transfer in CINECA

Users can use login nodes to transfer small files, but we strongly suggest to use the dedicated services: CINECA provides a data transfer service based on two main tools: **data movers** and **GridFTP**.

Data movers are dedicated, containerized nodes without interactive access, supporting only a limited set of commands (scp, rsync, sftp, wget, curl, rclone, aws s3 and s3). User authentication is done via SSH certificates with 2-Factor Authentication or host-based authentication from within CINECA clusters.

GridFTP is also available on these nodes but can only be used through the globus-url-copy client, which must be run from the user's local machine.

We will stick with Data movers, for the time being.

https://docs.hpc.cineca.it/hpc/hpc_data_storage.html#data-transfer

Data movers

alias: data.leonardo.cineca.it/data.g100.cineca.it

Datamovers are dedicated nodes on each HPC cluster that are designed for transferring data FROM/TO a cluster. On datamovers, there is no CPU time limit, that allows long data transfers. Unlike, on login nodes, there is a 10-minute of CPU time limit that usually interrupts the transfer of a large amount of data.

By construction, the shell is not available, so it is not possible to open interactive sessions. The only available commands are scp, rsync, sftp, wget, curl, rclone, s3 and aws s3.

IMPORTANT: on a datamover, the environment variables $HOME, $WORK and $CINECA_SCRATCH (as well as ~ or *) are not defined:

- if you want to transfer files FROM/TO your cluster personal areas, you have to specify the absolute path
- You cannot make use of the SSH configuration files stored in your remote ~/.ssh/ directory (such as $HOME/.ssh/config).

https://docs.hpc.cineca.it/hpc/hpc_data_storage.html#data-transfer

Example: how to use data movers (1)

- **Rsync**

- 1. You need to upload or download data FROM/TO your local machine TO/FROM a CINECA HPC cluster
  - *rsync -PravzHS /absolute/path/from/file <username>@data.<cluster_name>.cineca.it:/absolute/path/to/*
  - *rsync -PravzHS <username>@data.<cluster_name>.cineca.it:/absolute/path/from/file /absolute/path/to/*

- 2. You need to transfer files between 2 CINECA HPC clusters
  - *ssh -xt <username>@data.<cluster_name_1>.cineca.it rsync -PravzHS /absolute/path/from/file <username>@data.<cluster_name_2>.cineca.it:/absolute/path/to/*
  - *ssh -xt <username>@data.<cluster_name_1>.cineca.it rsync -PravzHS <username>@data.<cluster_name_2>.cineca.it:/absolute/path/from/file /absolute/path/to/*

https://docs.hpc.cineca.it/hpc/hpc_data_storage.html#data-transfer

Example: how to use data movers (2)

- **Scp**
- 1. You need to upload or download data FROM/TO your local machine TO/FROM a CINECA HPC cluster
  - *scp /absolute/path/from/file <username>@data.<cluster_name>.cineca.it:/absolute/path/to/*
  - *scp <username>@data.<cluster_name>.cineca.it:/absolute/path/from/file /absolute/path/to/*
- 2. You need to transfer files between 2 CINECA HPC clusters
  - *ssh -xt <username>@data.<cluster_name_1>.cineca.it scp /absolute/path/from/file <username>@data.<cluster_name_2>.cineca.it:/absolute/path/to/*
  - *ssh -xt <username>@data.<cluster_name_1>.cineca.it scp <username>@data.<cluster_name_2>.cineca.it:/absolute/path/from/file /absolute/path/to/*

https://docs.hpc.cineca.it/hpc/hpc_data_storage.html#data-transfer

Outline

- Leonardo infrastructure
- Access HPC resources and filesystems
- Software environment
- Programming environment
- Production environment
- Data transfer
- **Containers**
- Final remarks

Introduction to containers in CINECA

On CINECA's HPC clusters, the containerization platforms available can be either **Singularity** or **Apptainer**

- Singularity: the commercial software supported by Sylabs.
- Apptainer: the fully open-source Singularity port under the Linux Fundation.

"Apptainer IS Singularity"

The Apptainer project makes no changes at the image format level.

Singularity as a command line link, is provided, and maintains as much of the CLI and environment functionality as possible.

https://docs.hpc.cineca.it/services/singularity.html#singularity-and-apptainer-containers

Build a Singularity or Apptainer container image on your local machine

Simplest command used to build:

$ sudo singularity build [local options...] <IMAGE PATH> <BUILD SPEC>

build command can produce containers in 2 different output formats. Format can be specified by passing the fllowing build option:

- *default: a compressed read-only Singularity Image Format (SIF), suitable for production. This is an immutable object.*
- *sandbox: a writable (ch)root directory called sandbox, used for interactive development. To create those kind of output format use the --sandbox build option.*

https://docs.hpc.cineca.it/services/singularity.html#singularity-and-apptainer-containers

Containers in HPC environment

On CINECA's HPC clusters, build with sudo privileged commands cannot be executed. Solutions are:

- move locally built SIF images on CINECA's clusters (DataMovers⇒ File Systems and Data Management)
  - IMPORTANT: since Singularity's sandbox containers are writable (ch)root directories, accessing them requires root privileges so they will not be usable on HPC Cluster: consider creating a SIF image starting from the sandbox
- pulling existing container images from container registries:
  - *singularity pull registry://path/to/container_img[:tag]*

https://docs.hpc.cineca.it/services/singularity.html#singularity-and-apptainer-containers

Parallel MPI container

We can run parallel application using the host MPI installation (in CINECA clusters, OpenMPI)

IMPORTANT: the two installations (container and host) of OpenMPI have to be compatible!

To launch a parallel containerized application:

mpirun -np $nnodes singularity exec <container_img> <container_cmd>

The following code snippet will launch the application using MPI inside the container on a single node:

singularity exec <container_img> mpirun -np $nnodes <container_cmd>

If GPU support needed (full documentation is available here), just add the --nv or the --nvccli flag:

mpirun -np $nnodes singularity exec --nv <container_img> <container_cmd>

https://docs.hpc.cineca.it/services/singularity.html#singularity-and-apptainer-containers

GPU aware container

IMPORTANT: check that the container image holds a compatible version of CUDA, and the CUDA compatibility!

Best way to obtain a container image provided with a CUDA installation: bootstrap from an NVIDIA HPC SDK docker container, which already comes equipped with CUDA, OpenMPI and the NVHPC compilers (available at the NVIDIA catalog).
Their tag follows a simple structure, *$NVHPC_VERSION-$BUILD_TYPE-cuda$CUDA_VERSION-$OS*:

- $BUILD_TYPE: can either take the value devel or runtime. The first ones are usually heavier and employed to compile and install applications. The second ones are lightweight containers for deployment, stripped of all the compilers and applications not needed at runtime execution.

- $CUDA_VERSION: an either take a specific value (e.g.) or be a multi.

https://docs.hpc.cineca.it/services/singularity.html#singularity-and-apptainer-containers

Outline

- Leonardo infrastructure
- Access HPC resources and filesystems
- Software environment
- Programming environment
- Production environment
- Data transfe
- Containers
- **Final remarks**

Final remarks

- ★ **2FA method** is mandatory on CINECA HPC systems (not for training usernames).

- ★ **Login nodes** should only be used for installation (connection to external network!), compilation, and small tests. **No** GPUs on login nodes!

- ★ Consider to use **Leonardo Booster for your applications on GPUs** and **g100/Leonardo DCGP for applications only on CPUs.**

- ★ Recommended **compilers** are gcc and Nvidia compilers (nvhpc, cuda) for Leonardo Booster, and gcc and Intel (intel, oneapi) for Leonardo DCGP.
  Check the **options** required to enable OpenACC/OpenMP parallelization, GPU support…

- ★ Rely on the already available **software stack**, tested and optimized for the cluster architecture, and on **Spack** for autonomously installing additional software.

https://docs.hpc.cineca.it

Write to superc@cineca.it in case of need!

Reach out to us @
info@minerva4ai.eu

Thank you

Co-funded by the European Union

EuroHPC Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101182737. The JU receives support from the Digital Europe Programme.

57