Introduction to CINECA HPC systems

November 17th, 2025

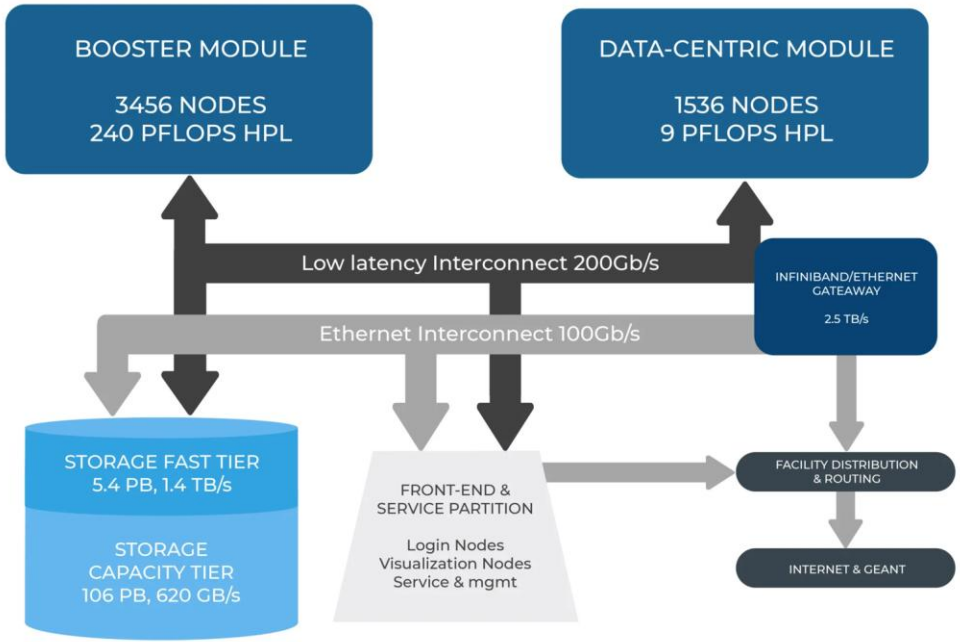
Michael Redenti
m.redenti@cineca.it

Outline

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

LEONARDO's ARCHITECTURE

SYSTEM OVERVIEW & LOGIN NODES



LOGIN NODES:

Processors:
2x Intel Xeon Platinum 8358 Processor
(*Intel Ice Lake – 32 cores, 3.4 GHz with Turbo*)

- Hyper Threading (×2) is enabled
- RAM: 512 GiB RAM DDR4 3200MHz
- 14TiB disk in RAID1 configuration
- **NO GPUs**
- Open to outside network
- *Serial* partition on two login node

BOOSTER MODULE (GPU-Accelerated)

Key Features & Specitications

Atos BullSequana X2135 "Da Vinci" blade

- 3456 nodes
- 1 × Intel Xeon Platinum 8358 Processor (32 cores)
- RAM: 512 (8 x 64) GB DDR4 3200 MHz
- Accelerators: 4 × GPU NVIDIA A100 custom - **15% performance improvement over the standard A100**
- Internal network: NVIDIA Mellanox HDR DragonFly+ 200Gb/s
- **DISKLESS!!!**
- Shared storage (*InfiniBand-Connected*): 106 PiB Capacity tier storage + 5.4 PiB Fast tier storage

Rmax per node: ~70 TFLOPS
Rmax: ~241 PFLOPS

IN PRODUCTION SINCE AUGUST 2023

*Rmax = *Maximal LINPACK performance achieved (TOP500)*

BOOSTER MODULE (GPU-Accelerated)

Processor Details

Intel Xeon Platinum 8358 Processor

- **32 cores**, each with 1.25 MiB of L2 cache and 48+32 KiB of L1 cache.
- 2 threads per core if hyperthreading is enable (only on login nodes).
- 2 of AVX-512 FMA Units.
- **48 MiB** of **L3 cache**, shared across all cores.
- 503 GiB of available RAM, divided into **2 NUMA nodes**.
- Processor Base Frequency: 2.60 GHz
- Max Turbo Frequency: 3.40 GHz

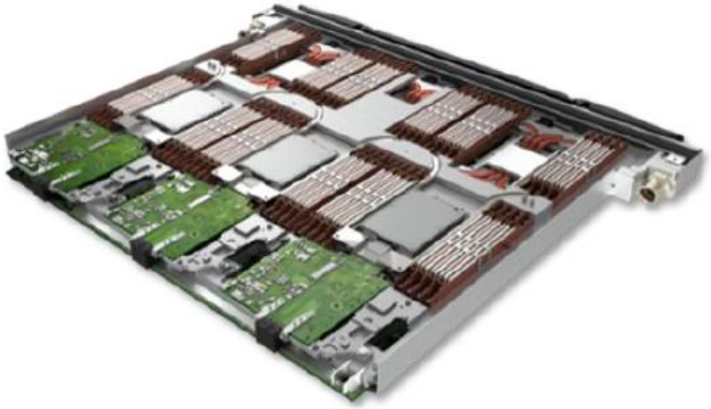
More information is available using the following commands:

```
$ lstopo-no-graphics
$ numactl –H
```

The **RAM available for user jobs is 494,000 MiB** (*slightly over 482 GiB*), as approximately 20 GiB is reserved for the operating system.

DCGP MODULE (CPU)

Key Features & Specitications



BullSequana X2140 three-node CPU Blade

- 1536 nodes (512 blades)
- **2 ×** Intel Xeon Platinum 8480+ Processor (56 cores each)
- RAM: 512 GB DDR5 4800 MHz
- Infiniband: 1 × NVIDIA HDR cards 100 Gbps via PCIe Gen 5
- Disk: 1 M.2 SSD 3.84 TB

Rmax per node: ~8.5 TFLOPS
Rmax: ~13 PFLOPS

IN PRODUCTION
SINCE FEBRUARY 2024

*Rmax = *Maximal LINPACK performance achieved (TOP500)*

DCGP MODULE (CPU)

Processor Details

Compute Node:

- **112 cores** in total (56 cores per socket).
- 503 GiB of available RAM, divided into **8 NUMA nodes** (4 per socket).

Intel Xeon Platinum 8480+ Processor:

- **56 cores**, each with 2 MiB of L2 cache and 48+32 KiB of L1 cache.
- 2 of AVX-512 FMA Units.
- **105 MiB** of **L3 cache**, shared across all cores.
- Processor Base Frequency: 2.00 GHz
- Max Turbo Frequency: 3.80 GHz

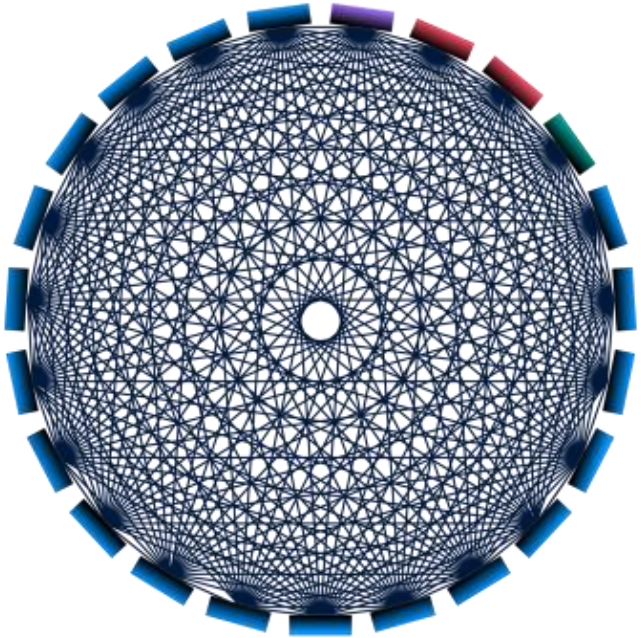
More information is available using the following commands:

```
$ lstopo-no-graphics
$ numactl –H
```

The **RAM available for user jobs is 494,000 MiB** (*slightly over 482 GiB*), as approximately 20 GiB is reserved for the operating system.

INTER-NODE NETWORK TOPOLOGY

Key Features & Specitications



Dragonfly+ topology

Based on NVidia Mellanox InfiniBand High Data Rate (HDR) and NVIDIA Quantum QM8700 switches

- All nodes are divided into cells.
- Cells are connected in an all-to-all topology with 18 independent connections between two different cells.
- Within each cell, a non-blocking two-layer fat tree topology is employed.

Slurm Optimization & Network Adaptability

- Slurm is aware of the topology and tune the node allocations on the dragonfly+ network.
- Adaptive Routing Algorithm enabled to alleviate network congestion.

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**

TOP500 SUPERCOMPUTERS

www.top500.org

November 2024

#	SYSTEM	COUNTRY	Rmax* [PFLOPS]	POWER [kW]
1	Frontier	USA	1,742.00	29,581
2	Aurora	USA	1,353.00	24,607
3	Aurora	USA	1,012.00	38,698
4	Eagle	USA	561.20	--
5	**HPC6**	**Italy**	**477.90**	**8,461**
6	Fugaku	Japan	442.01	29,899
7	**Alps**	**Switzerland**	**434.90**	**7,124**
8	**LUMI**	**Finland**	**379.70**	**7,107**
9	**Leonardo**	**Italy**	**241.20**	**7,494**

*Rmax = *Maximal LINPACK performance achieved*

On November 2022 Leonardo was in the 4th place in the top 500 classification for the first time, been after Frontier, Fugaku and LUMI

Outline

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

Become a new HPC user

(This is ***not*** something you need to do for this course)

- **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)

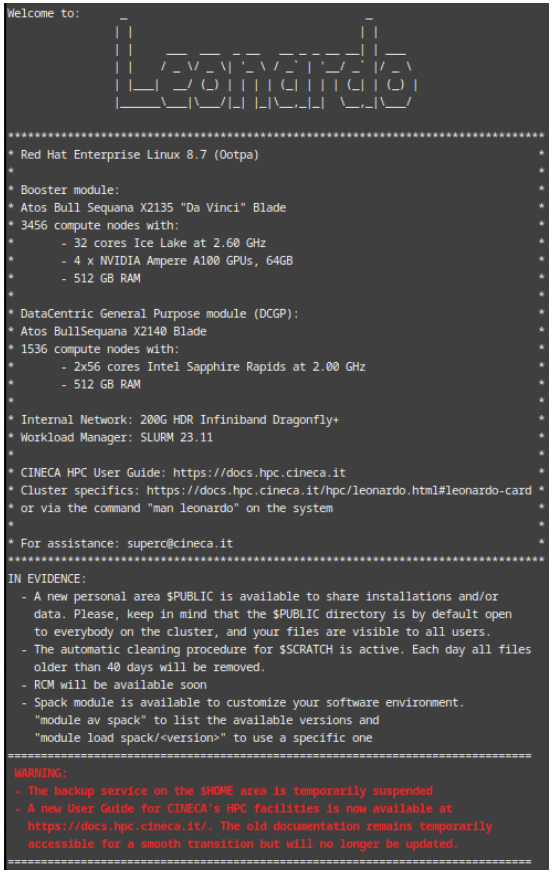
EuroHPC calls for users (PI) affiliated to an **European** Institute

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



For this course, you will access Leonardo with temporary training **usernames** with **password** that you received via email.

$ ssh a08traXX@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
- Final remarks

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

Outline

- Leonardo infrastructure
- Access HPC resources and filesystems
- Software environment
- **Programming environment**
- Production environment
- 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**
- 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=1             # tasks per node
#SBATCH --cpus-per-task=8               # cores per task
#SBATCH --gres=gpu:1                    # GPUs per node
#SBATCH --mem=494000                    # mem per node (MB)
#SBATCH --time=00:10:00                 # time limit (d-hh:mm:ss)
#SBATCH --partition=boost_usr_prod      # partition name
#SBATCH --account=tra25_gputs           # account
#SBATCH --reservation=s_tra_gputs       # reservation DAY 1

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=1             # tasks per node
#SBATCH --cpus-per-task=8               # cores per task
#SBATCH --gres=gpu:1                    # GPUs per node
#SBATCH --mem=494000                    # mem per node (MB)
#SBATCH --time=00:10:00                 # time limit (d-hh:mm:ss)
#SBATCH --partition=boost_usr_prod      # partition name
#SBATCH --account=tra25_gputs           # account
#SBATCH --reservation=s_tra_gputs2      # reservation DAY 2

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=1 --cpus-per-task=4 --gres=gpu:1 -t 00:10:00 -p boost_usr_prod -A tra25_gputs --reservation=s_tra_gputs --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 00:10:00 -p boost_usr_prod --A tra25_gputs –reservation=s_tra_gputs

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

Submit jobs with SLURM

#SBATCH --account=tra25_gputs or **-A tra25_gputs**

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**

account	start	end	total (local h)	localCluster Consumed(local h)	totConsumed (local h)	totConsumed %	monthTotal (local h)	monthConsumed (local h)
tra25_astrophd	20250919	20251005	3000	0	0	0.0	5625	0

$ 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).

Your training Project Account and usernames will be active until Sunday, November 23rd.

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)

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
- **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!


LEONARDO
EuroHPC
Joint Undertaking
CINECA
Founded by
the European Union
L E O NARDO

Thank you