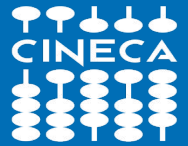
Introduction to the CINECA Leonardo HPC system

June 6, 2023

Isabella Baccarelli - i.baccarelli@cineca.it
Caterina Caravita - c.caravita@cineca.it



Outline

- CINECA infrastructure and Leonardo architecture (CPUs, GPUs, Memory, Interconnections)
- Access to the cluster and filesystems
- Software environment
- Programming environment
- Production environment (SLURM)
- Considerations and tips on the use of Leonardo
- Final remarks

2022 OVERVIEW

HPC SYSTEMS

CINECA

CINECA enables world-class scientific research by operating and supporting leading-edge supercomputing technologies and by managing a state-of-the-art and effective environment for the different scientific communities.

LEONARDO | 2022

4992 nodes
Booster Module:
32 core per node
4 GPU NVidia Ampere custom
Data Centric Module:
56 cores per node
110 PB Storage
250 PFlops

SOON IN PRODUCTION

MARCONI | 2016

3188 nodes
48 cores per node
612 TB RAM
10 PFlops

MARCONI100 | 2020

980 nodes
32 cores per node
4 GPU Nvidia V100 per node
8 PB Storage
32 PFlops

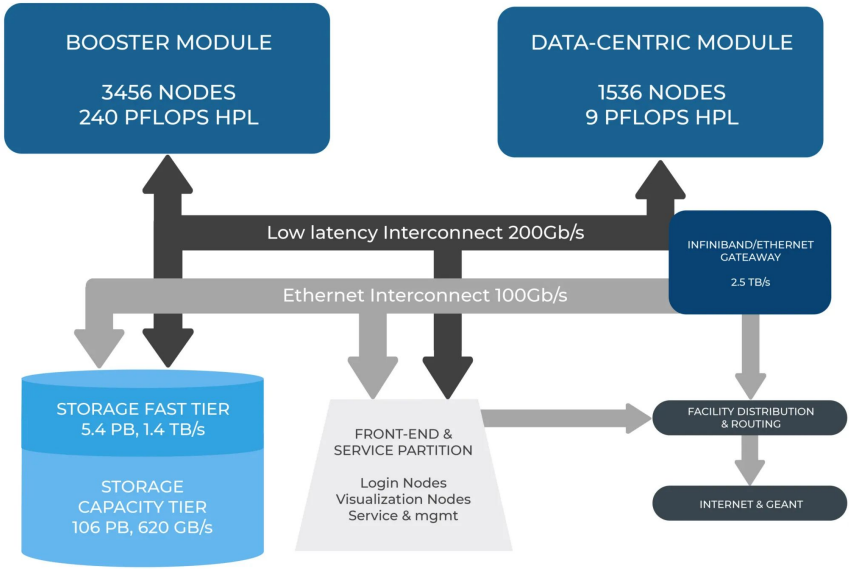
DGX | 2021

3 nodes
128 cores per node
8 GPU NVIDIA A100 per node
100 TB Storage
15 PFlops

GALILEO100 | 2021

564 nodes
48 cores per node
2 GPU NVIDIA V100 per node
~22 PB Storage
2 PFlops

Leonardo infrastructure and login nodes



Atos BullSequana X430-E6

- Processors: **2 x CPU Intel Whitley ICP06, 32 cores Intel Ice Lake, 2.4 GHz**
- RAM: 512 (16x32) GB RAM DDR4 3200 MHz
- 6TB disk in RAID1 configuration
- NO GPUs

Booster (GPU) module

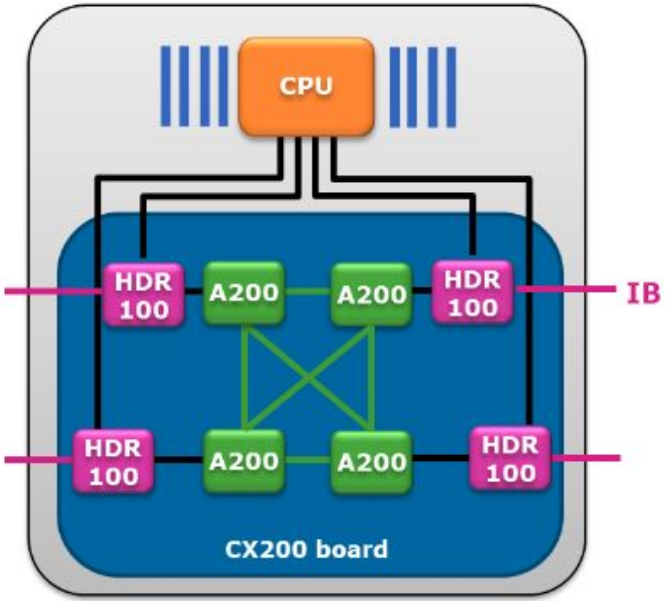
Atos BullSequana X2135 “Da Vinci” blade

- 3456 nodes
- Processors: **1 x CPU Intel Xeon 8358,**
  32 cores Intel Ice Lake, 2.6 GHz (ONE SOCKET!)
- RAM: 512 (8 x 64) GB DDR4 3200 MHz
- Accelerators: **4 x NVidia custom Ampere GPU A100 SXM4 64 GB, NVLink 3.0**
- Internal network: NVIDIA Mellanox HDR DragonFly+ 200Gb/s
- DISKLESS!!!
- Shared (via infiniband) storage space: 106 PB Capacity tier storage
  + 5.4 PB Fast tier storage

Peak performance per node: about 89,4 TFlops

Peak performance: about 309 PFlops

Booster (GPU) module



GPU performance

- 11.2 TFlops Peak FP64 per GPU or….
- 22.4 TFlops Peak FP32 per GPU or ….
- 22.4 Tflops Peak FP64 Tensor Core per GPU or…

Intra-node network

- NVLink, PCIe, GPU direct
- 200 GB/s between the GPU pairs
- Each GPU has direct 100Gb/s connection to the InfiniBand network
- PCIe Gen4 @ 31.5 GB/s

Memory

- 6.5 TB/s GPU memory bandwidth

Booster module: A100 GPUs

- **One socket** node
- The blade is being designed by Atos R&D labs in France primarily for the Leonardo project. It utilises a new GPU, Ampere A200 (ATOS NICKNAME!!!!!). The official nVIDIA name, as reported on the Top500 list, is **NVIDIA A100 SXM4 64 GB** (let's stick with that), which incorporates several new features to enhance performance and provide closer coupling of the interconnect.
- Compared to the recently announced Ampere A100 GPU, the "A200" GPU delivers higher flops per Watt and increased performance, has higher memory per GPU and improved bandwidth and latency. These features are expected to deliver a **15% performance improvement** over the standard A100 across a range of applications (tested! compared with DGX A100)

Why should you care?

- no worries for GPU-to-CPU binding. "Democratic" connections between each core and GPUs (remember the rank-by core mapping, or explicit gpu binding to tasks on M100? relaxed conditions on Leonardo concerning GPUs. For task/thread affinity, with respect to the ntasks-per-node and cpus-per-task directives, same rules apply, more later)

Booster module: interconnects

- **NVLink 3.0 between EACH pair of GPUs on nodes** (on M100 only between GPUs on socket). The four NVLINK links provides a bi-directional bandwidth of 200 GB/s between the GPU pairs. (M100: 150 GB/s). The 4 x “A200” GPUs has in total a HBM2e memory capacity of 256 GB with an expected accumulated GPU memory bandwidth of more than 6.5 TB/s.
- **NO NVLink between CPU and GPUs** (it’s Intel, no NVLink support for CPUs, CPU-GPU connections pass through the PCIe 4 (on M100: CPU-to-GPU NVLink connection on socket). Seems bad? only with respect to “one-socket” (1/2 GPUs runs). Moving towards exascale though
- **Each GPU is directly connected to a Mellanox HDR100 ConnectX6 adapter,** which, through its **PCIe passthrough functionality,** provides full speed **CPU to GPU** communications as well as **external connectivity to the HDR InfiniBand fabric**. Each of the ConnectX-6 adapters connects directly to the CPU and GPU via its integrated 32 PCIe Gen4 lanes switch (16 lanes connectivity to the CPU and 16 lanes connectivity to the GPU).

To sum up:

- ➢ Direct connection from each CPU and GPU to the network, delivering lowest latency
- ➢ Provides four HDR connections as cluster interconnect
- ➢ Enables a more power efficient node design due to the lack of external PCIe switches

More deep-down-into-hardware analysis is ongoing, with the help of NVIDIA/ATOS, stay tuned

Storage

Fast Tier

5.4 PB @ 1.4 TB/s

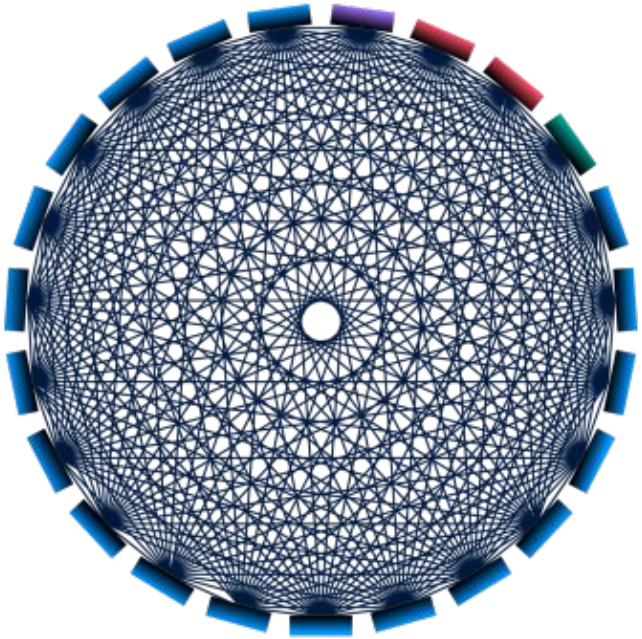
NVMe storage (SSD disks)
(home + fast scratch)

Capacity Tier

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

HDD disks
(work + large scratch + DRES)

Inter-node network topology



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

Dragonfly+ topology (as on M100, but at double bandwidth of 200 GBit/s (bidirectional)

Based on NVidia Mellanox Infiniband HDR

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

ADAPTIVE ROUTING ALGORITHM (as on M100)

- alleviates traffic congestion (slurm will take care of the “best”-possible node allocations on the dragonfly+ network with ARA enabled

Outline

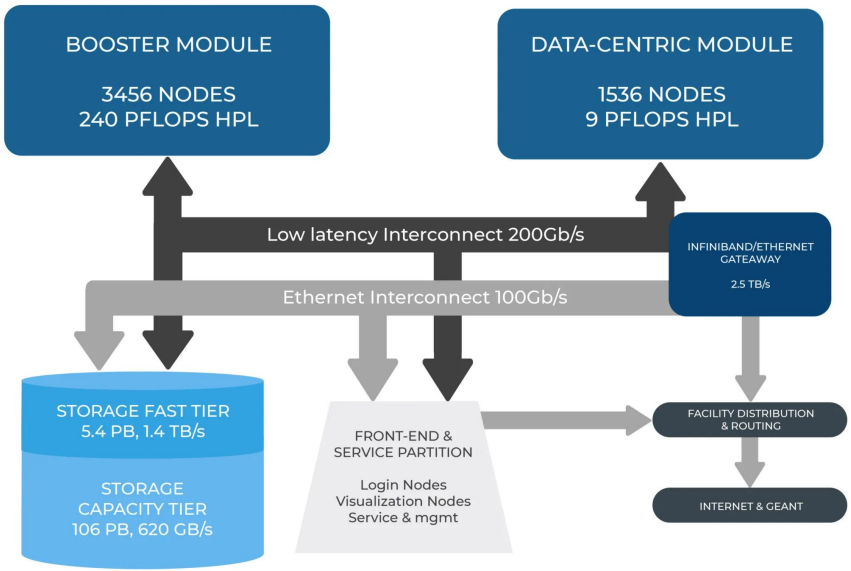
- CINECA infrastructure and Leonardo architecture

  (CPUs, GPUs, Memory, Interconnections)
- Access to the cluster and filesystems
- Software environment
- Programming environment
- Production environment (SLURM)
- Considerations and tips on the use of Leonardo
- Final remarks

Leonardo infrastructure: how to access

The access to Leonardo is granted to researchers of Italian and European centres and industries with approved projects for this platform: after a LEAP call, Iscra, EuroHPC…

Eurofusion community has **77 available nodes** of the booster module.



Become a new user

- **Register on the UserDB Portal**
  https://userdb.hpc.cineca.it/
- **Get associated to an active project on Leonardo**
  → Principal Investigator (PI): automatically associated when registered on UserDB
  → Collaborator: ask your PI to associate you to the account
- **Request the "HPC Access" on UserDB**
  → You will receive soon your credentials by mail

Leonardo infrastructure: how to access

The new mandatory method to access Leonardo (and the other 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.leonardo.cineca.it
```

A specific webinar will be held tomorrow on the 2FA!

Leonardo infrastructure: how to access

$ ssh username@login.leonardo.cineca.it

Reset password
https://sso.hpc.cineca.it/

Motto 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 and daily backed up

$PUBLIC

- 50 GB per user
- user specific (permissions **755**)
- permanent (maybe backed up)

$WORK

- not available yet, quota will be set
- account specific
- permanent (no backup)

$CINECA_SCRATCH

(also $SCRATCH)

- no quota
- user specific
- temporary (data will be removed after 40 days, and no backup)

$TMPDIR

250 GB each compute node, job specific (created when the job starts and deleted when the job ends)

Data resources (DRES)

not available yet

All the filesystems are based on **Lustre**

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

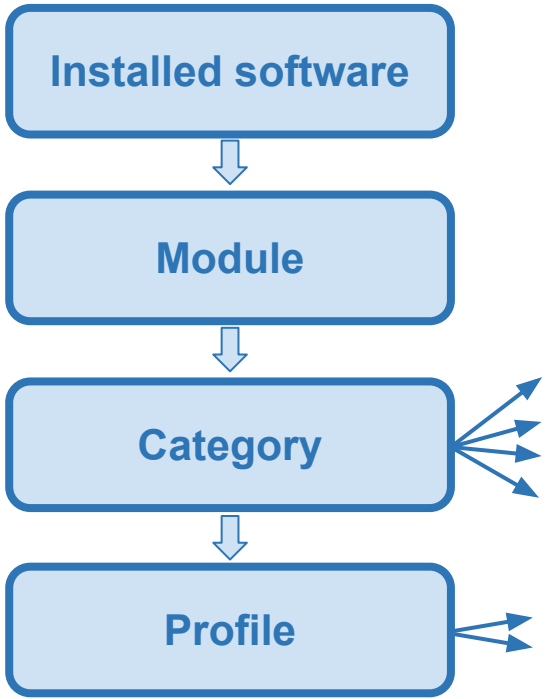
Outline

- CINECA and Leonardo infrastructure
- System architecture

  (CPUs, GPUs, Memory, Interconnections)
- Software environment
- Programming environment
- Production environment (SLURM)
- Considerations and tips on the use of Leonardo
- Final remarks

Module environment

Any available software is offered on Leonardo in a module environment (as also on Marconi and M100).
The modules are organized in functional categories and collected in different profiles.



Compilers
Libraries
Tools
Applications

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

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

Module environment

$ module av

```
------------------------------------------ /leonardo/prod/opt/modulefiles/profiles -----------------------------
profile/base  profile/chem-phys  profile/global  profile/lifesc  profile/meteo

------------------------------------- /leonardo/prod/opt/modulefiles/base/environment --------------------------
autoload

------------------------------------- /leonardo/prod/opt/modulefiles/base/libraries ----------------------------
adios/1.13.1--openmpi--4.1.4--gcc--11.3.0               nccl/2.14.3-1--gcc--11.3.0-cuda-11.8
adios/1.13.1--openmpi--4.1.4--nvhpc--23.1               netcdf-c/4.9.0--gcc--11.3.0
blitz/1.0.2--gcc--11.3.0                                netcdf-c/4.9.0--openmpi--4.1.4--gcc--11.3.0
boost/1.80.0--gcc--11.3.0                               netcdf-c/4.9.0--openmpi--4.1.4--nvhpc--23.1
boost/1.80.0--openmpi--4.1.4--gcc--11.3.0               netcdf-fortran/4.6.0--gcc--11.3.0
boost/1.80.0--openmpi--4.1.4--nvhpc--23.1               netcdf-fortran/4.6.0--openmpi--4.1.4--gcc--11.3.0
cgal/5.4.1--gcc--11.3.0                                 netcdf-fortran/4.6.0--openmpi--4.1.4--nvhpc--23.1
cgal/5.4.1--openmpi--4.1.4--gcc--11.3.0                 netlib-scalapack/2.2.0--openmpi--4.1.4--gcc--11.3.0
cudnn/8.4.0.27-11.6--gcc--11.3.0                        netlib-scalapack/2.2.0--openmpi--4.1.4--nvhpc--23.1
cutensor/1.5.0.3--gcc--11.3.0                           netlib-xblas/1.0.248--gcc--11.3.0
elpa/2021.11.001--openmpi--4.1.4--gcc--11.3.0-cuda-11.8 openblas/0.3.21--gcc--11.3.0
fftw/3.3.10--gcc--11.3.0                                openblas/0.3.21--nvhpc--23.1
```

Module environment

$ module load chem-phys
$ module av

Loaded profiles
are **added** to the environment

```
------------------------------------------ /leonardo/prod/opt/modulefiles/profiles -----------------------------
profile/base  profile/chem-phys  profile/global  profile/lifesc  profile/meteo
```

```
---------------------------------- /leonardo/prod/opt/modulefiles/chem-phys/applications ----------------------------
kokkos/3.7.00--openmpi--4.1.4--gcc--11.3.0-cuda-11.8                    yambo/5.1.1--openmpi--4.1.4--nvhpc--23.1
lammps/20220623--openmpi--4.1.4--gcc--11.3.0-cuda-11.8
nwchem/7.0.2--openmpi--4.1.4--gcc--11.3.0
quantum-espresso/7.2rc_local--openmpi--4.1.4--nvhpc--23.1-mkl-cuda-11.8
quantum-espresso/7.2rc_local--openmpi--4.1.4--nvhpc--23.1-openblas-cuda-11.8
```

$ 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

Module environment

$ modmap -m <module_name> → Detect all profiles, categories and modules available (e.g. different releases)
→ also only part of the name

$ module load <profile>

$ module load <module_name>/<version>

Autoload
not necessary
All the dependencies are automatically loaded

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

More modules (applications, libraries, tools, compilers) will be installed

Install via Spack

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

⇨ “Spack” environment provided by the **package manager Spack**.

$ module load spack/<version>

- **setup-env.sh** file is sourced
- **$SPACK_ROOT** is initialized
- **spack command** is added to your PATH, and some nice command line integration tools too
- Folder **/spack-<version>** is created into your **$PUBLIC** space 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**

Install via 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

- CINECA and Leonardo infrastructure
- System architecture

  (CPUs, GPUs, Memory, Interconnections)
- Software environment
- Programming environment
- Production environment (SLURM)
- Considerations and tips on the use of Leonardo
- Final remarks

Programming environment

Available in base profile

Compilers

- **GCC** 11.3.0 (GNU compilers: gcc, g++, gfortran)
- **NVHPC** 23.1 (ex hpc-sdk, ex PGI + CUDA → NVIDIA compilers: nvc, nvc++, nvcc, nvfortran)
- **CUDA** 11.8

Also **INTEL ONEAPI** 2023 (Intel compilers: icc, icpc, ifort…) → no Nvidia GPU support

MPI libraries

- **OpenMPI** (GNU/NVHPC compilers) → CUDA-aware
- **Intel OneAPI MPI** (Intel compilers)

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

Updated releases will be installed
(and can be installed autonomously with Spack)

Outline

- CINECA and Leonardo infrastructure
- System architecture

  (CPUs, GPUs, Memory, Interconnections)
- Software environment
- Programming environment
- Production environment (SLURM)
- Considerations and tips on the use of Leonardo
- Final remarks

Login and compute nodes

Leonardo (as the other CINECA HPC clusters) is 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*** (to be enforced)
- 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 22.05**
- Jobs can be submitted in two main ways: via **interactive mode** and via **batch mode**
- **Nodes shared**, but the allocated resources (cores, gpus, memory) are assigned in an exclusive way

Resources per node

Each node → max resources you can request per node

- ➢ 32 cores (cpus) → **ntasks-per-node * cpus-per-tasks ≤ 32**
- ➢ 4 GPUs
- ➢ 494000 MB

- ***No hyperthreading***
- **Single-socket compute nodes**

→ **tasks/threads binding not necessary**

⇨ The **accounting** considers

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

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

- N cpus
- N GPUs * 8 (= N GPUs * cores-per-node / GPUs-per-node)
- Memory / Memory-per-core (= Memory * cores-per-node / memory-per-node)

Eurofusion resources

Production partition → **boost_fua_prod** (default)

- max 16 nodes
- max walltime: 24 h

Debug QOS: **boost_qos_fuadbg**

- max 2 nodes
- max walltime: 2 h

Big production QOS: **boost_qos_fuabprod**

- min 17 full nodes
- max 32 nodes
- max walltime: 24 h

To be defined

“Low priority” qos
free, but zero queue priority, for active accounts with exhausted budget
+ **LOWPRIO account**
for active projects with non-exhausted budget
(after request to superc@cineca.it)

“Special” qos
if needed more than 32 nodes and/or 24h
(after request to superc@cineca.it and after approval by the EF Operation Committee)

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=1:00:00                  # time limit (d-hh:mm:ss)
#SBATCH --account=<account_no>          # account ($ saldo -b)
#SBATCH --partition=<partition_name>    # partition name
#SBATCH --qos=<qos_name>                # 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 01:00:00 -p <partition_name> -A <account_name> --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 01:00:00 -p <partition_name> -A <account_name>

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

Outline

- CINECA and Leonardo infrastructure
- System architecture

  (CPUs, GPUs, Memory, Interconnections)
- Software environment
- Programming environment
- Production environment (SLURM)
- Considerations and tips on the use of Leonardo
- Final remarks

Considerations and tips

- ★ **2FA method** is mandatory on Leonardo, feel free to test it on M100 before
- ★ **Login nodes and compute nodes** are different (no GPUs on login nodes)
- ★ Recommended **compilers** are gcc and Nvidia compilers (CUDA, nvhpc), but some libraries are optimized for Intel. Check the **options** required to enable OpenACC/OpenMP parallelization, GPU support…
- ★ Rely on the already available **software stack**, optimized for Leonardo architecture, and on **Spack** for autonomously installing further software you need

Outline

- CINECA and Leonardo infrastructure
- System architecture

  (CPUs, GPUs, Memory, Interconnections)
- Software environment
- Programming environment
- Production environment (SLURM)
- Considerations and tips on the use of Leonardo
- Final remarks

Final remarks

- ★ Even if not ready for the production phase yet, Leonardo is already *up & running*
- ★ Further configuration operations will be in order
- ★ More software will become available, as well as updated releases
- ★ More documentation will become available

Essential links

- UserDB: https://userdb.hpc.cineca.it/
- User Guide: https://wiki.u-gov.it/confluence/display/SCAIUS/HPC+User+Guide
  and, in particular, Leonardo, EuroFusion community, 2FA

A specific webinar will be held tomorrow on the 2FA!

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

Thanks to all EF users and to the HLST

Q&A

1.

Q: What is the actual RAM memory users can allocate out of 512 GB available on a node, if OS and other temporary files/directories have to be allocated there?

A: The actual RAM memory allocatable by users in their jobs is around 482 GB (some 15 GB for the node image + around 10 GB for the OS are reserved). We already take care of the memory needed on the diskless node reserving the needed amount of memory and limiting to 482 GB the memory for jobs.

Q&A

2.

Q: Is there an alternative way to access the clusters w/o the need to open a browser for authentication?
This would be desirable on other HPC systems where the User Network-Namespaces have been restricted for security reasons (hence, practically deactivating Firefox)?

A: Yes, you can actually download the ssh certificate and transfer it on the no-browser server.
If you will attend tomorrow's **webinar on 2FA** for EF we can discuss it more thoroughly.
→ Link to the UserGuide page with the slides and recordings

Q: Grazie. I was already considering such a solution but wasn't sure whether this would also be the preferred one on your end.

Q: Yes, that's what we were thinking of. You may also use the no-browser server as a proxy (if the ssh-agent is enabled on it) and your local ssh-agent will take care of forwarding the certificate to Leonardo via the proxy server.

Q&A

3.

Q: Will the slides of **this presentation** be available later ? (and if so, where?) Thanks

A: The slides will be available soon after the end of the webinar (and also the recordings).
I'm asking where we are supposed to upload everything, I'll update you shortly.
→ Link to the UserGuide page with the slides and recordings
→ Direct link to the slides and recordings

Q&A

4.

Q: Which is the maximum walltime for low priority jobs?

A: It's the same of the partition, i.e. 24 hours.

Q&A

5.

Q: Is it possible to do short interactive tests, e.g. using dbg queue, directly with srun?
Ok, the slide show the answer is yes.

A: Yes. You actually have two ways:
1) you "salloc" the resources with the dbg qos (you remain on the login node, and you can then launch as many "srun" as you need within the interactive job time limit);
2) you directly "srun" the application from the login node (this is a hidden-ish batch job).