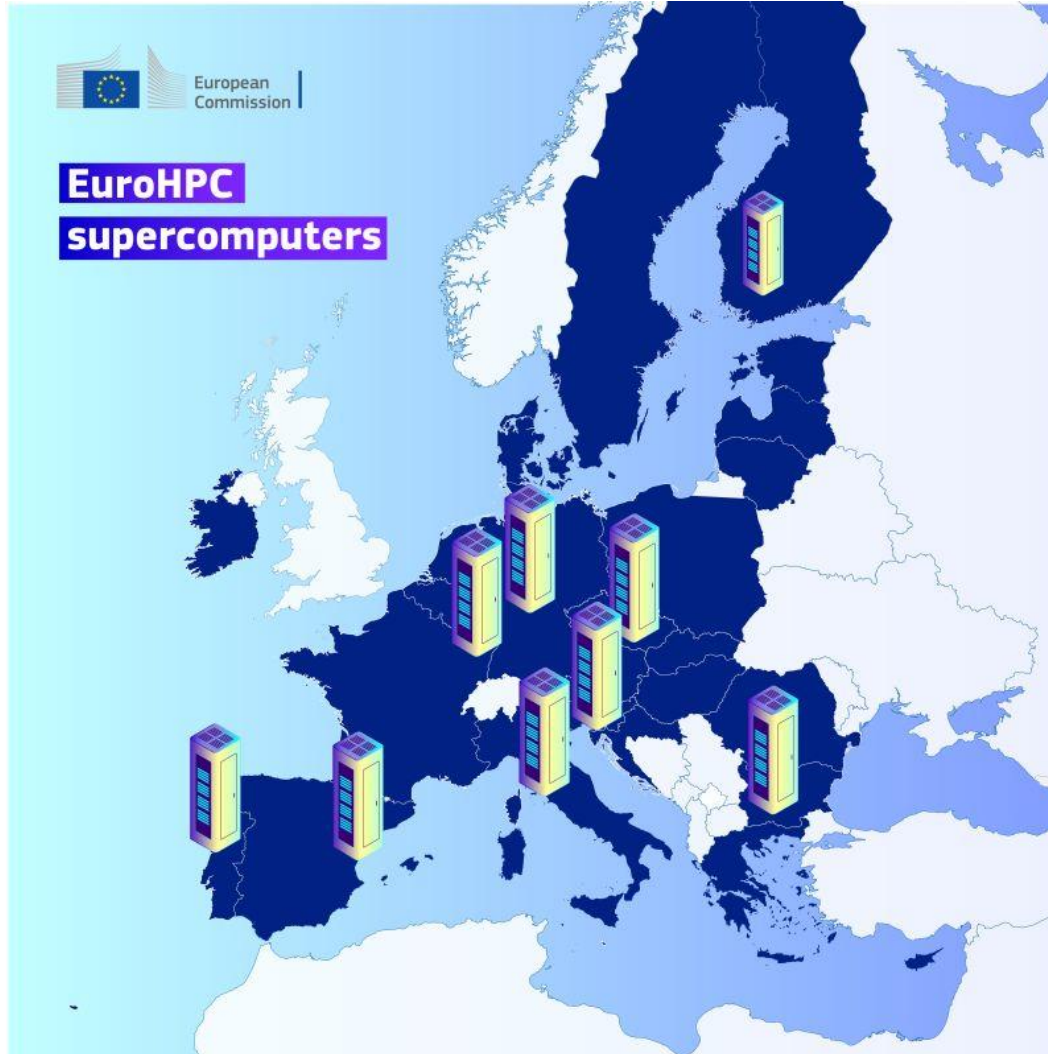




EuroHPC JU



Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
1	El Capitan - HPE Cray EX255a, AMD 4th Gen EPYC 24C 1.8GHz, AMD Instinct MI300A, Slingshot-11, TOSS, HPE DOE/NNSA/LLNL United States	11,039,616	1,742.00	2,746.38	29,581
2	Frontier - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Cray OS, HPE DOE/SC/Oak Ridge National Laboratory United States	9,066,176	1,353.00	2,055.72	24,607
3	Aurora - HPE Cray EX - Intel Exascale Compute Blade, Xeon CPU Max 9470 52C 2.4GHz, Intel Data Center GPU Max, Slingshot-11, Intel DOE/SC/Argonne National Laboratory United States	9,264,128	1,012.00	1,980.01	38,698
4	Eagle - Microsoft NDv5, Xeon Platinum 8480C 48C 2GHz, NVIDIA H100, NVIDIA Infiniband NDR, Microsoft Azure United States	2,073,600	561.20	846.84	
5	HPC6 - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, RHEL 8.9, HPE Eni S.p.A. Italy	3,143,520	477.90	606.97	8,461
6	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
7	Alps - HPE Cray EX254n, NVIDIA Grace 72C 3.1GHz, NVIDIA GH200 Superchip, Slingshot-11, HPE Cray OS, HPE Swiss National Supercomputing Centre (CSCS) Switzerland	2,121,600	434.90	574.84	7,124
8	LUMI - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE EuroHPC/CSC Finland	2,752,704	379.70	531.51	7,107
9	Leonardo - BullSequana XH2000, Xeon Platinum 8358 32C 2.6GHz, NVIDIA A100 SXM4 64 GB, Quad-rail NVIDIA HDR100 Infiniband, EVIDEN EuroHPC/CINECA	1,824,768	241.20	306.31	7,494



Leonardo Infrastructure

Booster

- 3456 nodes
- Processors: 1 x CPU Intel Xeon 8358, 32 cores Intel Ice Lake, 2.6 GHz
- Accelerators: 4 x NVidia Ampere GPU A100
- Internal network: NVIDIA Mellanox HDR DragonFly+

Data Centric & General Purpose

- 1536 nodes
- Processors: 2 x CPU Intel Xeon 8480+, 56 cores Intel Sapphire Rapids, 2.0 GHz
- Infiniband: 1 x NVIDIA HDR cards 100 Gbps via PCIe Gen 5

Leonardo: Access to the system

- Access through the [2FA procedure](#) (not yet for training-users)
- Leonardo can be reached with SSH (Secure Shell) protocol using the "collective" hostname:

```
$ ssh -X username@login.leonardo.cineca.it
```

Or explicit login-node

```
$ login01-ext.leonardo.cineca.it; $ login02-ext.leonardo.cineca.it;
```

```
$ login05-ext.leonardo.cineca.it; $ login07-ext.leonardo.cineca.it;
```

- Parallel storage, accessible from all the nodes of a given cluster:

`$HOME` **`$PUBLIC`** **`$WORK`** **`$CINECA_SCRATCH`**

<https://wiki.u-gov.it/confluence/display/SCAIUS/LEONARDO+User+Guide>

Leonardo: Production Environment

The software are divided in **profiles** and **categories** (for each profile):

Categories:

compilers
environment
libraries
tools
data
applications

Profiles

base
astro
bioinf
chem-phys
deeplrn
lifesc
advanced
archive
candidate

On the cluster you will find already several common software (**module**) installed

<https://wiki.u-gov.it/confluence/display/SCAIUS/LEONARDO+User+Guide>

Leonardo: Production Environment

To see all the available profiles/modules use the command *\$ module av*

To find a specific module use the command: *\$ modmap -m <module_name>*

To load the module you need with all the dependencies:

\$ module load profile/<profile_name>
\$ module load autoload <module_name>

To check the module loaded
\$ module list

To check details about the module:
\$ module show <module name>

To have examples of job scripts to run the code
\$ module help <module name>

to clean the environment:
\$ module purge

Leonardo: Production Environment

```
$ module load cineca-hpyc
```

```
$ python -m venv --system-site-packages my_venv      #create virtual env
```

```
$ source my_venv/bin/activate  #activate virtualenv
```

```
$ pip list
```

```
$ pip install <package_name>
```

```
$ deactivate      #exit from a virtualenv
```

```
$ pip freeze > requirements.txt
```

```
$ pip install -r requirements.txt
```

Leonardo: Interactive Batch Job

In case you need to “interact” with your running job (open notebook, tuning of input parameters, debugging etc.) you can submit an “Interactive” SLURM batch job

- Ask for the needed resources (cores, gpus, memory, time) with *srun*

```
srun -N 1 --ntasks-per-node=1 -A tra25_PythSci -p boost_usr_prod --reservation s_tra_py1 -t 02:00:00 --pty /bin/bash
```

- The job is queued and scheduled as any other job

- The session starts on the compute node (look at the prompt)

- You can then run your application from that terminal

Leonardo: Slurm Directives

- Leonardo allows you to run your simulations by submitting “jobs” to the compute nodes
- Your job is managed by a scheduler, **SLURM**, that adds it to a queue and allows its execution when the resources required are available

SLURM: "Simple Linux Utility for Resource Management"

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

```
#!/bin/bash
#SBATCH -t 1:00:00
#SBATCH -N 1
#SBATCH --ntasks-per-node=16
#SBATCH --cpus-per-task=4
#SBATCH --gres=gpu:4
#SBATCH --mem=10GB
#SBATCH -o job.out
#SBATCH -e job.err
#SBATCH -p boost_usr_prod
#SBATCH -A <account name>

module load <module_name>
.....
mpirun -np 16 ..... ./myprogram
```

Jupyter Notebook On Leonardo

1. On **localhost** (your laptop) open ssh session to Leonardo login node from the shell with the command:

```
ssh USERNAME@login01-ext.leonardo.cineca.it
```

2. Once you are on Leonardo login, submit an interactive job to get a compute node:

```
srun -N 1 --ntasks-per-node=1 -A tra25_PythSci -p boost_usr_prod --reservation s_tra_py1 -t 02:00:00 --pty /bin/bash
```

3. Load the specific module containing python libraries

```
module load cineca-hpyc
```

Jupyter Notebook On Leonardo

4. Open a new shell from your laptop open ssh tunnel to login node and from login node to compute node:

```
ssh -L 9999:localhost:9999 USERNAME@login01-ext.leonardo.cineca.it ssh -L 9999:localhost:9999 -N ldrxxx
```

5. Go back to the shell on the remote host (Leonardo) and open the jupyter notebook on the selected port with the

```
Jupyter notebook --port=9999 --no-browser
```

6. To access the notebook, open a browser on localhost and copy and paste the URL that will appear on the shell

```
http://localhost:9999/?token=75f9c6d4611a636b3249cd79fe10b218ab1f1c267d4c53d13
```

All the steps are described on gitlab page, file README.md

How to get resources

Italian SuperComputing Resource Allocation (ISCRA)

Calls ISCRA C – Small projects for research, test and development

Calls ISCRA D – Data Storage

Calls ISCA B – Standard Project

Eurohpc Access

Applicants can request machine time on the EuroHPC JU supercomputers available via:

- Benchmark and access to development
- Regular access (one-year projects)
- Extreme Scale Access (one-year or 2-year projects)

<https://leonardo-supercomputer.cineca.eu/access-leonardo-hpc-resources/>