Introduction to Pitagora

June 23th, 2025

Jonathan Frassineti
j.frassineti@cineca.it

CINECA - High Performance Computing Department

Outline

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

Pitagora login nodes

Connect with:

login{01-06}-ext.pitagora.cineca.it

- CPU (01, 03, 05): same as compute nodes (see later)

- GPU (02, 04, 06): same as compute nodes, only GPU differs at the moment (same GPU, less memory, air cooled)

Data Centric and General Purpose (CPU-only) partition

Lenovo SD665V3 DWC CPU compute node

- 1008 nodes
- Processors (dual-socket): **2x AMD Turin Dense 128c - Zen5 microarch** (**128 cores/node per CPU**), **2.4 GHz**
- RAM: 768 (24 x 32) GB (744 GB available) DDR5 6400 MHz
- Internal network: Nvidia ConnectX-7 NDR 200Gbit/s network interface
- No storage
- **Performance: R_{max} = 17 Pflops, R_{peak} ~ 20 Pflops**

BOOSTER (CPU+GPU) partition

Lenovo SD650-N V3 GPU compute node

- 168 nodes
- Processors (dual-socket): **2x Intel Emerald Rapids 6548Y 32c** (**32 cores/node per CPU**), **2.4 Ghz**
- GPU: **4x NVIDIA H100 SXM 100GB HBM2e**
- RAM: 512 (16 x 32) GB (494 GB available) DDR5 5200 MHz
- Internal network: Nvidia ConnectX-7 NDR200 200Gbit/s network interface
- Storage: **894 GB**
- **Performance: R_{max} ~ 28 Pflops, R_{peak} ~ 37 Pflops**

Storage (DDN)

- 1x ES200NVX2E appliance for metadata handling housing:

- 12 SSD NVMe 3,84TB

- 3x ES400NVX2E-S appliances for data storage with 5x SS9024 expansion JBOD each appliance housing:

- 24 SSD NVMe 3,84TB (ES400NVX2E controller)

- 440 Hard Disk 12TB SAS Enterprise Edition (JBOD SS9024)

Raw total HDD Capacity (WORK, SCRATCH): 15.7 PB

RAW flash Capacity (HOME, PUBLIC): 0.27 PB (3x 24 SSD NVMe 3,84TB (ES400NVX2E controller))

Sequential Read/Write HDD: ~150/190 GB/s

Outline

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

Become a new HPC user

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

- **Get associated to an active 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 two mails:

  one with your HPC username, and one to set an HPC password and configure the 2FA

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

Access to Pitagora

The access to CINECA HPC systems requires a **two-factor authentication (2FA)**.

First time

- **Activate the 2FA**: authenticate on our **Identity Provider** at https://sso.hpc.cineca.it using your HPC username and password.
  → 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 from your local shell.
  → 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.pitagora.cineca.it
```

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

Access to Pitagora

```
********************************************************************************
*                                                                              *
* Pitagora is still in a pre-production phase.                                 *
*                                                                              *
* Please note that:                                                            *
*                                                                              *
* - The configuration of the Slurm scheduler is still being finalized, so you  *
*   may encounter some anomalies likely related to the current configuration.  *
*   At this stage, the following temporary partitions are available:           *
*     gpu (default)                                                            *
*     cpu                                                                      *
*   In the coming days, we will create the definitive partition structure.     *
* - Accounting is currently not active, so it is not necessary to specify the  *
*   budget account in the slurm scripts. For this reason the saldo command is  *
*   still not available.                                                       *
* - For MPI users: At this stage, we recommend using mpirun instead of srun,   *
*   as the partial configuration of Slurm may result in unexpected behavior.   *
* - There are already four software stacks available as modules, compiled      *
*   with gcc, NVHPC, AMD and Intel-Oneapi.                                     *
*   For nvhpc suite two software stacks are available: one based on            *
*   nvhpc@24.5 and one based on nvhpc@24.9.                                    *
* - If you encounter this error with job submission:                           *
*   sbatch: error: Batch job submission failed: Unexpected message received    *
*   you can try cleaning your module environment with "module purge" and then  *
*   resubmit. Load all the modules you need inside the jobscript itself        *
* - Being this an early preproduction configuration, it is possible that the   *
*   system or some components may be temporarily shut down at any time for     *
*   reconfigurations.                                                          *
********************************************************************************
Last login: Thu Jun 19 17:04:25 2025 from 130.186.19.188
09:11:05 [jfrassi1@login01 ~]$
```

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

Motto of the day

- ➔ **Short system description**
- ➔ **“In evidence” messages**
- ➔ **“Important” messages**

  (e.g. scheduled maintenances)

Filesystems

$HOME

- 50 GB per user
- user specific
- permanent
- Backup (in future)

$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
- Backup (in future)

$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

Filesystems

$HOME
- 50 GB per user
- user specific
- Permanent
- Backup (in future)

$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

$TMPDIR
- local on nodes
- job specific
- **fast I/O**

DRES
- To be defined
- No backup

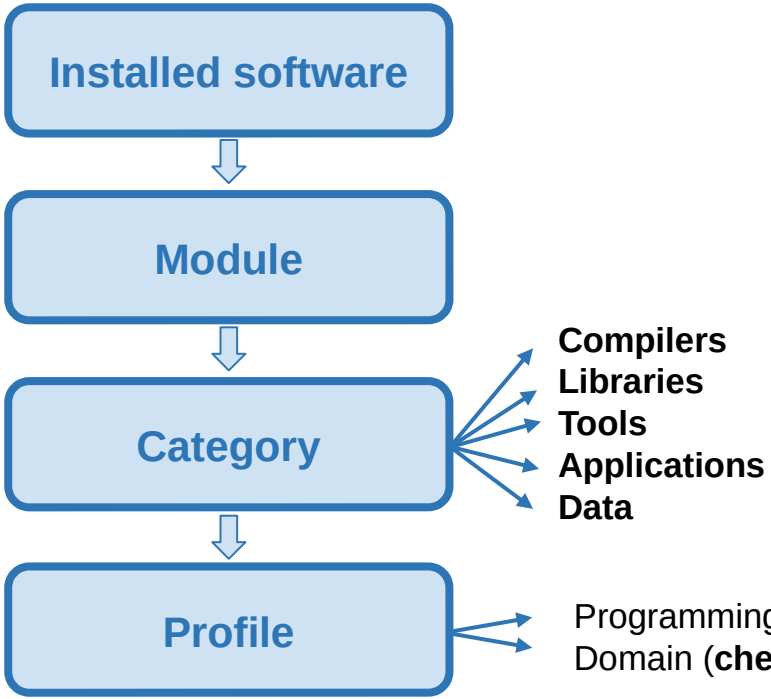
All the filesystems are based on **Lustre**

Outline

- Pitagora infrastructure
- Access HPC resources and filesystems
- **Software environment**
- Programming environment
- Production environment
- 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.



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

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

Module environment

$ module avail

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

------------------------------------------ /pitagora/prod/opt/modulefiles/base/libraries ------------------------------------------
adios/1.13.1--hpcx-mpi--2.20--nvhpc--24.5                    libszip/2.1.1--oneapi--2024.1.0
adios/1.13.1--hpcx-mpi--2.20--nvhpc--24.9                    metis/5.1.0--aocc--4.2.0
adios/1.13.1--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0  metis/5.1.0--gcc--12.3.0
adios/1.13.1--openmpi--4.1.6--aocc--4.2.0                    metis/5.1.0--nvhpc--24.5
adios/1.13.1--openmpi--4.1.6--gcc--12.3.0                    metis/5.1.0--nvhpc--24.9
amdblis/4.2--aocc--4.2.0                                     metis/5.1.0--oneapi--2024.1.0
amdfftw/4.2--aocc--4.2.0                                     nccl/2.22.3-1--gcc--12.3.0
amdfftw/4.2--openmpi--4.1.6--aocc--4.2.0                     netcdf-c/4.9.2--aocc--4.2.0
amdlibflame/4.2--aocc--4.2.0                                 netcdf-c/4.9.2--gcc--12.3.0
amdscalapack/4.2--openmpi--4.1.6--aocc--4.2.0                netcdf-c/4.9.2--hpcx-mpi--2.20--nvhpc--24.5
blitz/1.0.2--aocc--4.2.0                                     netcdf-c/4.9.2--hpcx-mpi--2.20--nvhpc--24.9
blitz/1.0.2--gcc--12.3.0                                     netcdf-c/4.9.2--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0
blitz/1.0.2--oneapi--2024.1.0                                netcdf-c/4.9.2--oneapi--2024.1.0
boost/1.85.0--aocc--4.2.0                                    netcdf-c/4.9.2--openmpi--4.1.6--aocc--4.2.0
boost/1.85.0--gcc--12.3.0                                    netcdf-c/4.9.2--openmpi--4.1.6--gcc--12.3.0
boost/1.85.0--hpcx-mpi--2.20--nvhpc--24.5                    netcdf-fortran/4.6.1--aocc--4.2.0
boost/1.85.0--hpcx-mpi--2.20--nvhpc--24.9                    netcdf-fortran/4.6.1--gcc--12.3.0
boost/1.85.0--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0  netcdf-fortran/4.6.1--hpcx-mpi--2.20--nvhpc--24.5
boost/1.85.0--oneapi--2024.1.0                               netcdf-fortran/4.6.1--hpcx-mpi--2.20--nvhpc--24.9
```

```
------------------------------------------- /pitagora/prod/opt/modulefiles/base/tools -------------------------------------------
binutils/2.42                                          intel-oneapi-itac/2022.1.0
cdo/2.4.0--gcc--12.3.0                                 intel-oneapi-vtune/2024.1.0
cdo/2.4.0--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0  jube/2.6.1
cmake/3.27.9                                           maven/3.8.4
cuda-samples/12.5                                      ncftp/3.2.6
curl/8.7.1                                             nco/5.1.9--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0
eccodes/2.34.0--aocc--4.2.0                            nco/5.1.9--openmpi--4.1.6--gcc--12.3.0
eccodes/2.34.0--gcc--12.3.0                            ninja/1.11.1
eccodes/2.34.0--oneapi--2024.1.0                       openjdk/11.0.20.1_1
emacs/29.3                                             osu-micro-benchmarks/7.4--hpcx-mpi--2.20--nvhpc--24.5
esmf/8.6.0--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0  osu-micro-benchmarks/7.4--hpcx-mpi--2.20--nvhpc--24.9
esmf/8.6.0--openmpi--4.1.6--aocc--4.2.0                osu-micro-benchmarks/7.4--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0
esmf/8.6.0--openmpi--4.1.6--gcc--12.3.0                osu-micro-benchmarks/7.4--openmpi--4.1.6--aocc--4.2.0
git/2.45.1                                             osu-micro-benchmarks/7.4--openmpi--4.1.6--gcc--12.3.0
gmt/6.4.0--aocc--4.2.0                                 snakemake/8.5.2
gmt/6.4.0--gcc--12.3.0                                 spack/0.22.2_6.1
intel-oneapi-advisor/2024.1.0                          superc/2.0
intel-oneapi-inspector/2024.1.0                        texinfo/7.0.3

------------------------------------------ /pitagora/prod/opt/modulefiles/base/compilers ------------------------------------------
aocc/4.2.0  gcc/12.3.0                          intel-oneapi-compilers/2023.2.4  nvhpc/24.5  perl/5.38.0
cuda/12.6   intel-oneapi-compilers-classic/2021.10.0  intel-oneapi-compilers/2024.1.0  nvhpc/24.9  python/3.11.7
```

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

Module environment

$ module load profile/chem-phys
$ module avail

Loaded profiles
are **added** to the environment

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

```
------------------------------------- /pitagora/prod/opt/modulefiles/chem-phys/applications --------------------------------
gromacs/2024.2--aocc--4.2.0                                  namd/3.0--gcc--12.3.0
gromacs/2024.2--gcc--12.3.0                                  namd/3.0--openmpi--4.1.6--gcc--12.3.0
gromacs/2024.2--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0  plumed/2.9.2--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0
gromacs/2024.2--oneapi--2024.1.0                             plumed/2.9.2--openmpi--4.1.6--aocc--4.2.0
gromacs/2024.2--openmpi--4.1.6--aocc--4.2.0                  plumed/2.9.2--openmpi--4.1.6--gcc--12.3.0
gromacs/2024.2--openmpi--4.1.6--gcc--12.3.0                  quantum-espresso/7.4.1--hpcx-mpi--2.20--nvhpc--24.9-openblas
namd/3.0--aocc--4.2.0                                        yambo/5.3.0--hpcx-mpi--2.20--nvhpc--24.9
```

$ 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) → TO BE INSTALLED SOON

$ module load <profile>

$ module load <module_name>/<version> → **all the dependencies are automatically loaded; we recommend to specify the module version!**

$ 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/2024.2--intel-oneapi-mpi--2021.12.1--oneapi--2024.1.0).
Remind that modules compiled with nvhpc, cuda should be used only on the **GPU** partition, while modules compiled with gcc, aocc are suitable for running on the CPU partition.

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 guidance on the installation or if you want to request a new module.

Install with Spack

⇨ “Spack” environment provided by the **package manager Spack** and available as **modules**.

$ module avail spack

[jfrassi1@login01 ~]$ module avail spack
---------- /pitagora/prod/opt/modulefiles/base/tools ----------
spack/0.22.2_6.1

Install with Spack

Load the module:

$ module load spack/0.22.2_6.1

- **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 Pitagora and 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

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
--------------------------------
 -   w6b2un3  scorep@8.4%gcc@12.3.0~cuda~hip+mpi+papi~pdt~shmem~unwind build_system=autotools arch=linux-rhel9-zen4
[^]  qsgkuxr      ^binutils@2.42%gcc@11.4.1~gas+gold~gprofng~headers~interwork+ld~libiberty~lto~nls~pgo+plugins build_system=autotools c
ompress_debug_sections=zlib libs=shared,static arch=linux-rhel9-zen4
```

$ 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

- Pitagora 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: on Pitagora DCGP, AMD aocc compilers and libraries are recommended.

Check with commands
module av,
module show,
module help,
and **man**

Compilers

- **GCC** (GNU compilers: gcc, g++, gfortran)
- ***gcc*** *was installed to offload the NVIDIA GPUs through the nvptx target. However, if the device is not present, as in the* ***CPU*** *partition, it runs on the CPU.*
- **NVHPC** (ex hpc-sdk, ex PGI + CUDA → NVIDIA compilers: nvc, nvc++, nvcc, nvfortran)
- **CUDA**

- **INTEL ONEAPI** (**Oneapi** compilers: icx, icpx, ifx, ifort) → **no** Nvidia GPU support
- *From the 2024 version,* ***intel-oneapi-compilers contains*** *only the oneAPI compilers (the x ones) and the classic Fortran (ifort), which will no longer be available from 2025 version.*
- **INTEL ONEAPI CLASSIC** (**Intel** compilers: icc, icpc, ifort) → **no** Nvidia GPU support
- *Load* ***intel-oneapi-compilers-classic*** *to use the above compilers*
- **AMD AOCC** → **no** Nvidia GPU support

MPI libraries

- **OpenMPI** (**GNU** compiler) → CUDA-aware
- **hpcx-mpi** (**NVHPC** compiler) → CUDA-aware

- **OpenMPI** (**AOCC** compiler) → **no** CUDA-aware
- **Intel Oneapi MPI** (**Intel** compilers) → **no** CUDA-aware

Outline

- Pitagora 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
- ***GPUs only on certain login nodes (login02, login04 and login06)***

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, RAM, $TMPDIR) are assigned in an exclusive way

Resources per node

Each node → max resources you can request per node

- 64 cores (**GPU**), 256 cores (**CPU**) → **ntasks-per-node * cpus-per-tasks ≤ 64 (BOOSTER), 256 (DCGP)**
- 495 GB (**GPU**), 744 GB (**CPU**) of RAM (**memory**)
- 894 GB of temporary local memory on $TMPDIR (**GPU**) (**gres=tmpfs**)

Accounting is currently not active, so it is not necessary to specify the budget account in the slurm scripts!

⇨ The **accounting** will consider

- 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)

Eurofusion resources

Production partition → **dcgp_fua_prod/boost_fua_prod**

1) Normal QOS: **normal**

- max 16 nodes (BOOSTER) – max 64 nodes (DCGP)
- max walltime: 24 h

2) Big production QOS:
dcgp_qos_fuaprod/boost_qos_fuaprod

640 (DCGP) – 96 (BOOSTER) compute nodes dinamically allocated

- min 65 full, max 128 nodes (DCGP) – min 17 full, max 32 nodes (BOOSTER)
- max walltime: 24 h

3) Long production QOS:
dcgp_qos_fualprod/boost_qos_fualprod

- max 3 (BOOSTER), 3 (DCGP) nodes
- max walltime: 4 days

Debug partition → **dcgp_fua_dbg/boost_fua_dbg**

Normal QOS: **normal**

- max 2 nodes
- max walltime: 30 min

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 –mem=<x>MB                      # memory on RAM
#SBATCH –gres=tmpfs:<x>GB               # memory on $TMPDIR
#SBATCH --time=1:00:00                  # 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

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

$ srun -N 1 --ntasks-per-node=8 --cpus-per-task=4 -t 01:00:00 -p <partition_name> -A <account_name> --pty /bin/bash

The session starts on the compute node: **[username@r328c03s09 ~] $**

Also **MPI programs**

$ salloc -N 1 --ntasks-per-node=8 --cpus-per-task=4 -t 01:00:00 -p <partition_name> -A <account_name>

A new session starts on the login node: **[username@login02 ~]$**

Submit jobs with SLURM

Only on Pitagora “diskful” nodes (**GPU**), 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 **GPU** compute nodes (**max 894 GB**).

On the diskless **CPU** 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 GPU jobs the requested amount of gres=tmpfs resource contributes to the consumed budget, changing the number of accounted equivalent core hours.

Submit jobs with SLURM

#SBATCH --account=<account_name> or **-A <account_name>**

Specifies the account with a **budget** of core-hours available to run jobs.

Remind also that, on Pitagora, you can check the status of your accounts with “saldo” (TO BE INSTALLED SOON)

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

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

Final remarks

- ★ **Login nodes** should only be used for installation (connection to external network!), compilation, and small tests. GPUs only on certain login nodes!

- ★ Consider to use Pitagora **GPU/BOOSTER for applications on GPUs** and Pitagora **CPU/DCGP for applications only on CPUs.**

- ★ Recommended **compilers** are gcc and Nvidia compilers (nvhpc, cuda) for Pitagora Booster, and gcc and AMD (aocc) for Pitagora DCGP.

- ★ 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/index.html

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

Additional info

SW architecture

- Pitagora CPU (AMD) = zen5
- Pitagora GPU (Intel) = sapphire rapids
- Software stack = zen4. Zen 4 includes instruction set extensions that are compatible with both AMD and Intel architectures, allowing us to maintain a unified GCC software stack across both AMD and Intel partitions. Zen 4 supports AVX-512, but compared to Intel’s Sapphire Rapids architecture, it lacks AMX (Advanced Matrix Extensions) instructions. Additionally, compared to Zen 5, it misses a minor extension that is not particularly critical.

Additional info (2)

INTEL-ONEAPI-MPI on AMD

- job that fail with the error (es. Ascot):

  = *BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES*

  = *RANK 1920 PID 2666886 RUNNING AT r332c06s03*

  = *KILLED BY SIGNAL: 9 (Killed)*

  srun: error: r333c01s06: task 15: Broken pipe

  [mpiexec@r332c04s12] wait_proxies_to_terminate (../../../../../src/pm/i_hydra/mpiexec/intel/i_mpiexec.c:554): downstream from host r332c04s12 exited with status 141

  The issue appears to be related to the default Mellanox provider. For now, it can be resolved by explicitly selecting the verbs provider (using ***export I_MPI_FABRICS=shm:ofi*** and ***export FI_PROVIDER=verbs***). After some investigation, we found that the code compiled with Intel oneAPI MPI using the Mellanox provider runs without issues on the Intel CPU side. So we attribute the problem to Intel MPI on AMD, where it can be resolved by selecting the fabrics provider, which slightly reduces performance. We still don't know exactly by how much.

- job that hang (es. starwall): still investigating.
- jobs that concern PETSc, which produces incorrect output at runtime. In this case too, it appears to be a problem with the Mellanox provider, which, however, is not resolved even when using the verbs provider. It actually works only with the tcp provider, which uses the service network and is therefore even more inefficient.

Additional info (2)

INTEL-ONEAPI-MPI on AMD

To summarize:

Intel MPI on AMD might (because there are no issues with OSU benchmarks and presumably other applications) show problems with the mlx provider. The workaround (which doesn't apply in all cases) is to use the verbs provider, with some performance penalty (NOTE: the fabrics is always ofi, both for mlx and verbs), or even tcp, which we do not recommend since it uses the service network and is significantly less efficient.

Additional info (3)

SLURM

- SLURM version is not definitive, possible errors may occur
- If you encounter this error:

  sbatch: error: Batch job submission failed: Unexpected message received

  you can try cleaning your module environment with "module purge" and then resubmit. Load all the modules you need inside the jobscript itself

Q&A

1.

Q: how can we install software/libraries so that they are available to all the members of one project?

A: You can install them in your $PUBLIC (with open access) or in your WORK (shared with collaborators) with Spack. You can also create your own modules.

See more information regarding installing with SPACK on CINECA clusters at https://docs.hpc.cineca.it/hpc/hpc_enviroment.html#spack

Q&A

2.

Q: if I am already user and with active projects, when can i start running on Pitagora?

A: Pitagora is open to projects selected for EUROfusion cycle 9. Projects related to cycle 8 (started in March 2024), will run on Leonardo until July. The projects of cycle 9 are already defined on the cluster. To access the cluster you must be associated with one of these.

Q&A

3.

Q: If no memory or tempfs specified, what's happening with that resources?

A: The default for memory is the memory-per-core. The default for tmpfs is 10 GB.

Q&A

4.

Q: Who do I have to contact to ask for operation and allocation resources?

A: In case of doubt: richard.kamendje@euro-fusion.org (operation), Duarte.Borba@euro-fusion.org (allocation, see call)