Sept. 25th, 2025

Managing a Large-Scale HPC Infrastructure:

Production environment and monitoring

Alessandro Marani (*a.marani@cineca.it*)
Daniele Di Bari (*d.dibari@cineca.it*)
Orlenys Troconis (o.troconis@cineca.it)

User Support & Production Team
High Performance Computing Dept.

PRESENTATION

OUTLINE

- Disclaimer and Recap
- Storage Areas and Data Transfer
- Software environment
- Production environment with Slurm: the user point of view
- Production environment with Slurm: the administrator point of view
- Cluster monitoring: Sanity checks
- Cluster monitoring: Interconnect network tests

Disclaimer: the idea behind these talks

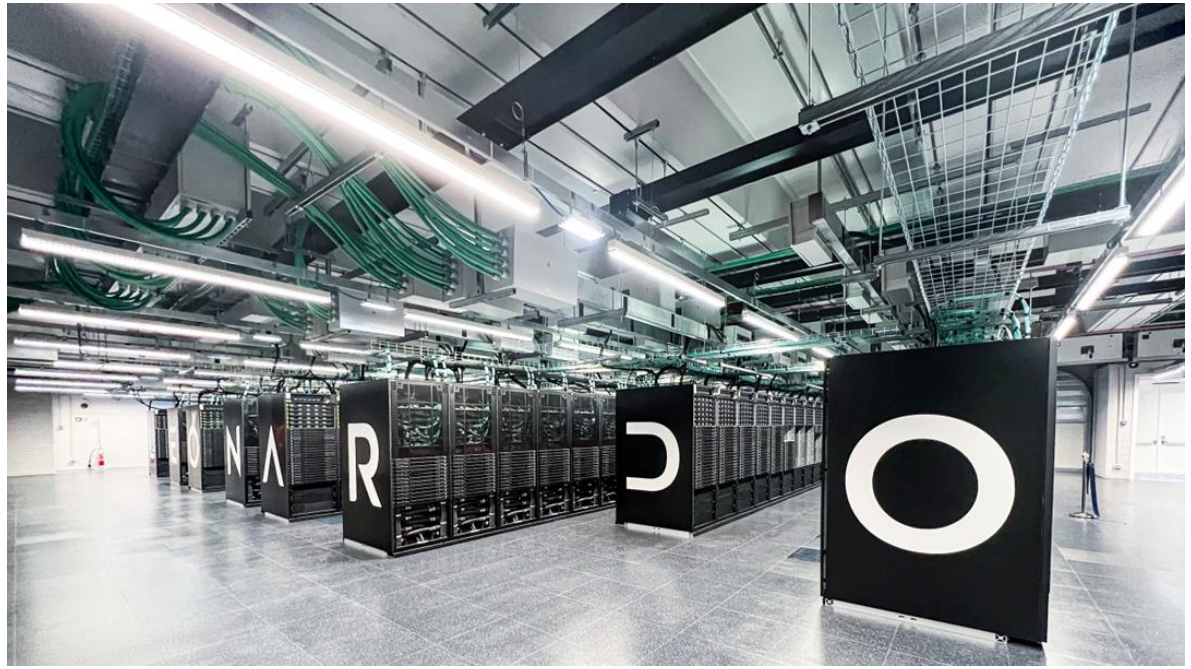
We are part of the User Support and Production team of Cineca. Our job is to:

- **Being the direct interface with our users, answering their requests and assisting them with the problems they face in our clusters;**
- **Take care of all the production aspects of our environment, deciding the use policies, setting up the configuration and make sure that all the users can have a good experience in a shared environment**

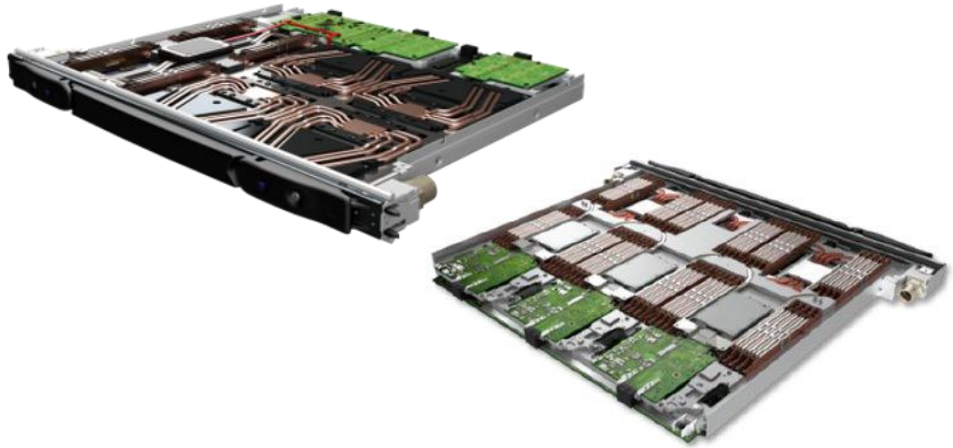
This is a Dev-Ops school, but we are not sysadmins. What we can do is to share our knowledge as User Support members, reminding that a good user experience is the final goal.
We show you our solutions to many problems: they may not be the best, or the most suited for your specific needs, but at least we hope they can be of inspiration :-)

Recap: the architecture of Leonardo

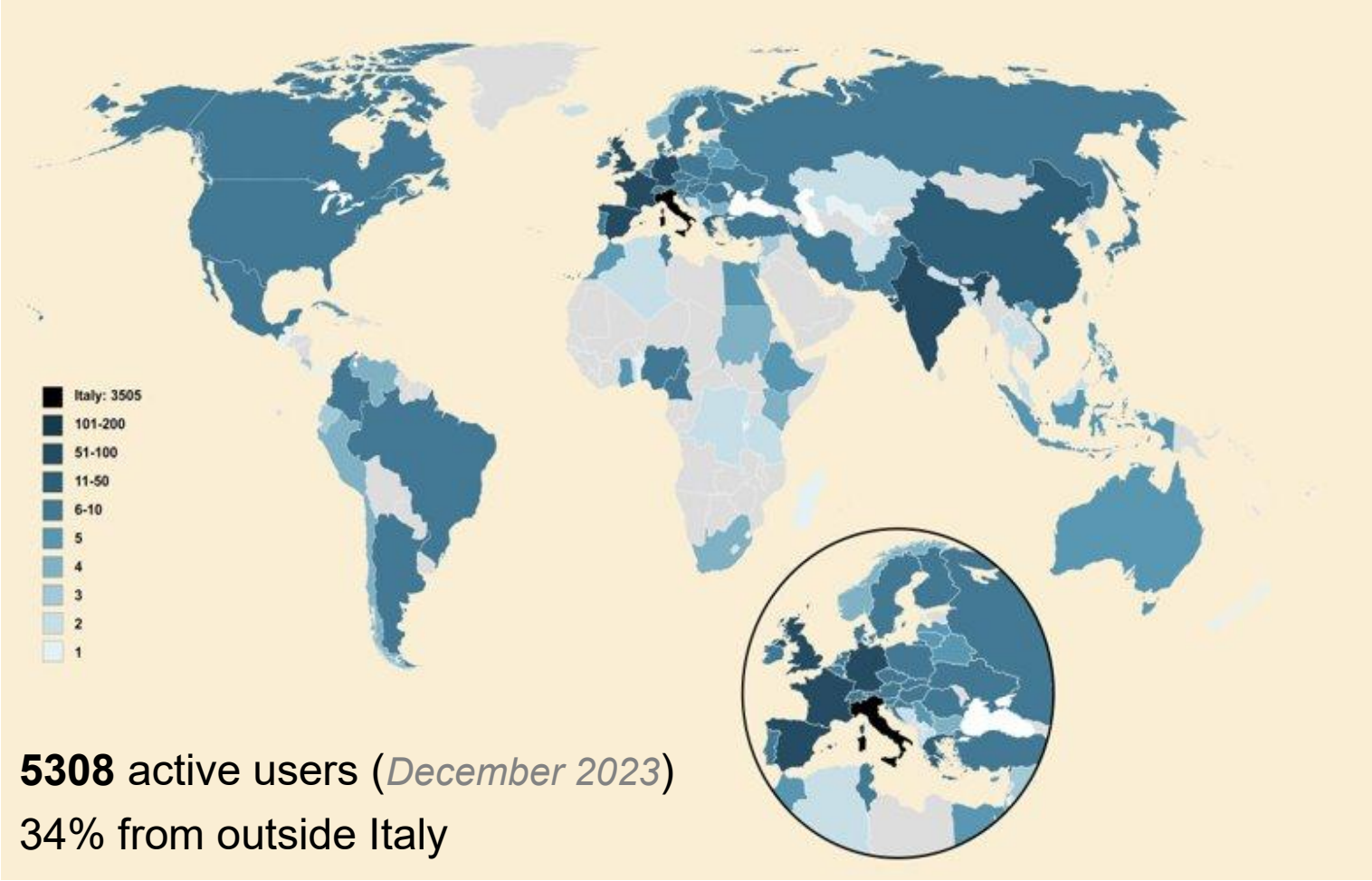
- Currently **10th Top500**
- **HPL 241,20 PF**
- **4992 nodes** based on **BullSequana XH2000** platform technology (3456 GPU + 1536 CPU)
- Computing racks: **95% Direct Liquid Cooled**
- Data storage: >**100PB** (NVMe+HDD)
- NVIDIA Mellanox **HDR 200** interconnect Dragonfly+ topology



Booster module:
1xIntel 8538 processor (32 cores) + 4 NVIDIA A100 custom GPUs
DCGP module:
2xIntel 8540+ processors (112 cores total)

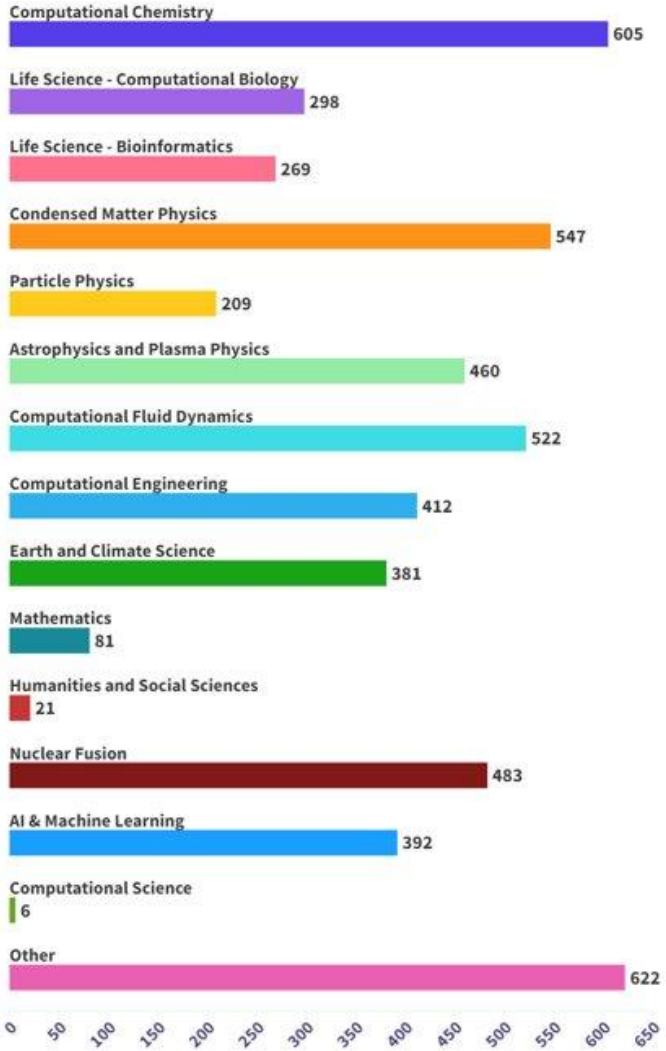


Recap: our userbase



5308 active users (*December 2023*)

34% from outside Italy



- Many important projects at an **European level** (EUROFusion, EUROHPC, Human Brain Project, …)

Recap: Becoming a user and first access

- **Ways to Get Resources:** mainly via peer-reviewed calls (national or international), with exceptions for commercial agreements.
- **Application Process:** Submission of a proposal detailing the scientific use case, resources needed, time usage, and codes to be run. Proposals are reviewed for scientific quality and technical feasibility.

Once you gain an HPC username, you can configure the **two-factor authentication process** and login via ssh to one of the frontend nodes of Leonardo, in round-robin fashion

Recap: Accounting and budget linearization

Account (different from username):

- Identifies the resource allocation which you can use for your work.
- A **budget** is associated with an account and reports how many resources (computing hours) are available on each cluster.
- An amount of storage is also associated with an account, available on the $WORK space
- The account budget and storage is shared between all the users that are associated to the account (collaborators).

Accounted Resources(cpus-h)= Reserved Cores eq.(cpus)• Elapsed Time(h)

Budget linearization (more on that later):
Every account has a **monthly quota** (total_budget / total_no_of_months). When users start to consume the account budget, the jobs submitted from the account will **gradually lose priority**, until the monthly budget is fully consumed.

When this happens, you can still run jobs (so it is possible to consume more than the monthly quota each month), but these jobs will have the lowest priority.

PRESENTATION

OUTLINE

- Disclaimer and Recap
- Storage Areas and Data Transfer
- Software environment
- Production environment with Slurm: the user point of view
- Production environment with Slurm: the administrator point of view
- Cluster monitoring: Sanity checks
- Cluster monitoring: Interconnect network tests

Filesystems

$HOME

- 50 GB per user
- User specific
- Permanent (till user is active)
- Daily backup (**soon**)

$PUBLIC

- 50 GB per user
- User specific (permissions **755**)
- Permanent (till user is active)
- **No** backup

$FAST

Same rules then WOR with Faster R/W SSD)

Data resources (DRES)
Shared area among different projects platforms.

$WORK

- Quota per account (default 1TB)
- Account specific
- Permanent (account + 6 month)
- **No** backup

$SCRATCH

- No quota
- User specific
- Temporary (data will be removed after 40 days, and no backup)

Local SSD storage (3TB)
Exploitable in jobs via $TMPDIR (SLURM directive).
Serial & DCGP only.

All the filesystems are based on **Lustre**

Filesystems

```
[mguernel@login02 ~]$ cindata
USER       AREADESCR                                AREAID                                FRESH     USED      QTA      USED%    aUSED     aQTA     aUSED%
mguernel   /leonardo_scratch/fast/cin_propro        leonardo_scratch_fast-22057231        35min       --       --       --%      4K       1T       0.0%
mguernel   /leonardo_work/IscrB_SoDi-PSV_0          leonardo_work-20059220                35min       --       --       --%     26T      35T      74.5%
mguernel   /leonardo_scratch/fast/cin_sudo          leonardo_scratch_fast-22058717        35min       --       --       --%      4K       1T       0.0%
mguernel   /leonardo_work/cin_staff                 leonardo_work-20042960                35min       --       --       --%     38T      100T     38.1%
mguernel   /leonardo_scratch/fast/cin_saldo         leonardo_scratch_fast-22058716        35min       --       --       --%      4K       1T       0.0%
mguernel   /leonardo_work/cin_propro                leonardo_work-20057231                35min       --       --       --%      4K       1T       0.0%
mguernel   /leonardo_work/cin_sudo                  leonardo_work-20058717                35min       --       --       --%      4K       1T       0.0%
mguernel   /leonardo_work/cin_saldo                 leonardo_work-20058716                35min       --       --       --%      4K       1T       0.0%
mguernel   /leonardo_scratch/fast/cin_staff         leonardo_scratch_fast-22042960        35min       --       --       --%    5.3G       1T       0.5%
mguernel   /leonardo/home/userinternal/mguernel     leonardo_home-10126046                35min     747M      50G     1.5%      --       --       --%
mguernel   /leonardo/pub/userinternal/mguernel      leonardo_pub-12126046                 34min      10G      50G    21.5%      --       --       --%
mguernel   /leonardo_scratch/large/userinternal/mguernel leonardo_scratch-11126046         33min     121G       --       --%      --       --       --%
[mguernel@login02 ~]$ _
```

Area location (full path)

ID

Last update of cindata

User occupied space

User quota

User occupied space (%)

Shared usage and quota for the whole account

Check your areas, disk usage and quota: **$ cindata**
Or the more recent **cinquota** that updates in real time

Datamover and Data transfer

Hostname: *data.<clustername>.cineca.it*

Main features

- **No cpu-time limit on processes**
- **Service is containerized and based on restricted GNU shell RUSH**
  - It is not possible to connect directly to the datamover, but the commands must be executed remotely
  - Connection allowed only with valid **SSH certificate** (2FA), no other private/public keys allowed
  - Connection from a CINECA login node does not require SSH certificate (Hostbased authentication enabled)
- **Only few commands are accepted: scp, rsync, sftp, wget, curl**
- **Environment variables are not defined ($HOME, $WORK, $CINECA_SCRATCH)**
  - You have to specify the absolute path location of the files
- **All storage areas of the cluster are visible**

Datamover and Data transfer

Usage examples: moving data to/from an external machine with rsync or scp

(e.g. your PC, a non-CINECA 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/

$ **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/

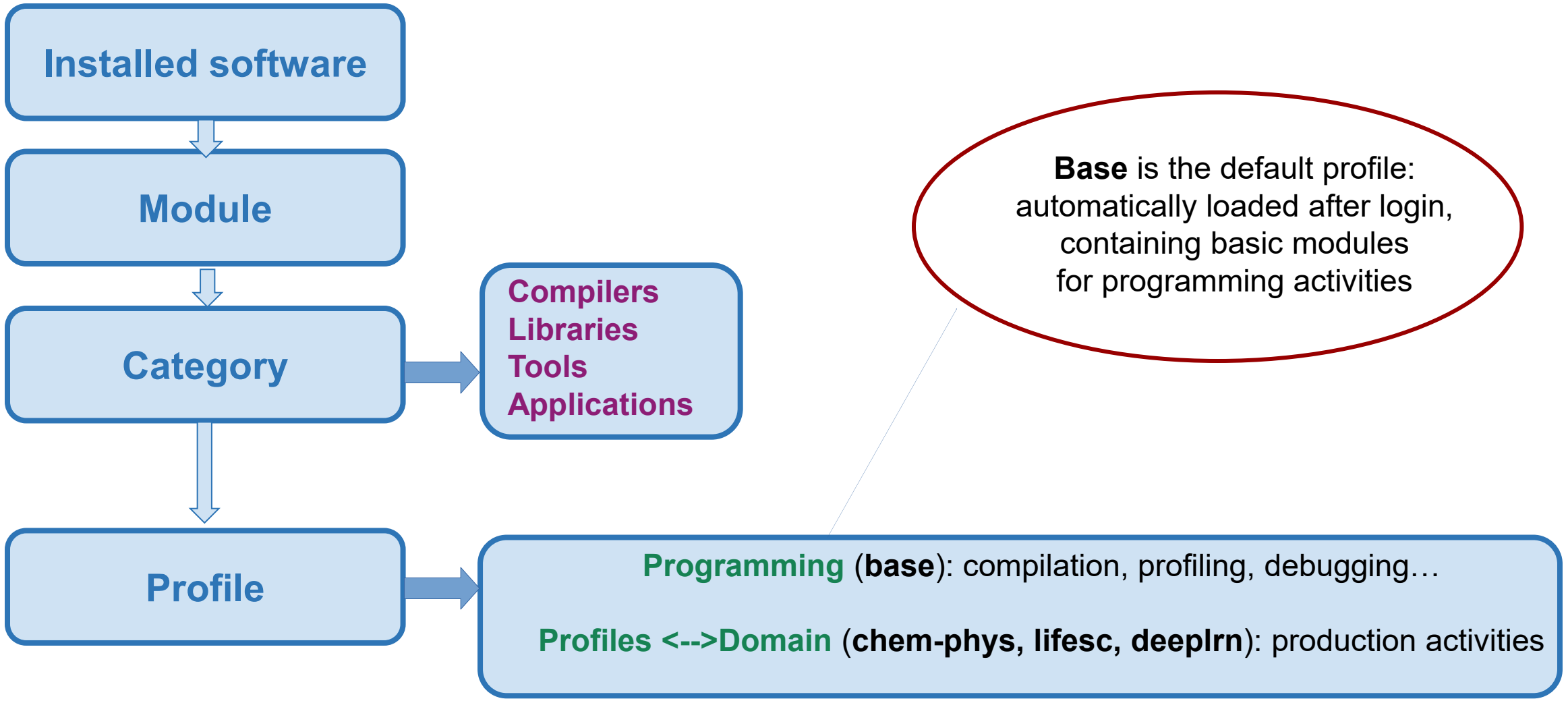
PRESENTATION

OUTLINE

- Disclaimer and Recap
- Storage Areas and Data Transfer
- **Software environment**
- **Production environment with Slurm: the user point of view**
- **Production environment with Slurm: the administrator point of view**
- **Cluster monitoring: Sanity checks**
- **Cluster monitoring: Interconnect network tests**

Module environment

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



Module environment

$ module av

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

------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/tools ----------------
anaconda3/2023.09-0                                                 intel-oneapi-vtune/2023.2.0                                 snakemake/6.15.1
cintools/1.0                                                        jube/2.4.3                                                  spack/0.21.0-68a
cmake/3.27.7                                                        maven/3.8.4                                                 spack/DCGP_0.21.0
cube/4.8                                                            ncftp/3.2.6                                                 superc/2.0
curl/8.4.0                                                          nco/5.1.6--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0    texinfo/6.5
darshan-runtime/3.4.5--intel-oneapi-mpi--2021.10.0--gcc--12.2.0     nco/5.1.6--openmpi--4.1.6--gcc--12.2.0
darshan-runtime/3.4.5--openmpi--4.1.6--gcc--12.2.0                  ninja/1.11.1
darshan-util/3.4.5--gcc--12.2.0                                     nvtop/3.1.0
dos2unix/7.4.4--gcc--12.2.0                                         openjdk/11.0.20.1_1
emacs/29.1                                                          osu-micro-benchmarks/7.3--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0
esmf/8.5.0--openmpi--4.1.6--gcc--12.2.0                             osu-micro-benchmarks/7.3--openmpi--4.1.6--gcc--12.2.0-cuda-12.1
extrae/4.0.6--openmpi--4.1.6--gcc--12.2.0-cuda-12.1                 osu-micro-benchmarks/7.3--openmpi--4.1.6--nvhpc--23.11
fenics/2019.1.0.post0--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0 paraview/5.10.1-osmesa
firefox/134.0.1                                                     py-darshan/3.4.5.0--gcc--12.2.0
ghostscript/10.03.1                                                 qibo/0.2.2
git-lfs/3.1.2                                                       root/6.28.06--gcc--12.2.0-cuda-12.1
git/2.42.0                                                          scalasca/2.6.1--intel-oneapi-mpi--2021.10.0--intel--2021.10.0
gmt/6.4.0--gcc--12.2.0                                              scalasca/2.6.1--openmpi--4.1.6--nvhpc--23.11
grads/2.2.3--openmpi--4.1.6--gcc--12.2.0                            scalasca/2.6.1--openmpi--4.1.6--nvhpc--24.3
imagemagick/7.1.1                                                   scorep/8.1--openmpi--4.1.6--nvhpc--23.11
intel-oneapi-inspector/2023.2.0                                     scorep/8.3--intel-oneapi-mpi--2021.10.0--intel--2021.10.0
intel-oneapi-itac/2021.10.0                                         scorep/8.3--openmpi--4.1.6--nvhpc--24.3
```

Module environment

$ module av

```
------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/compilers ---------
cuda/12.1  cuda/12.3  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/12.2  gcc/12.2.0  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

------------------------------------------------------------------ /leonardo/prod/opt/modulefiles/base/libraries ---------------------
adios/1.13.1--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0
adios/1.13.1--openmpi--4.1.6--gcc--12.2.0-cuda-12.1
blitz/1.0.2--gcc--12.2.0
blitz/1.0.2--oneapi--2023.2.0
boost/1.83.0--gcc--12.2.0
boost/1.83.0--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0-atomic
boost/1.83.0--oneapi--2023.2.0
boost/1.83.0--openmpi--4.1.6--gcc--12.2.0
boost/1.83.0--openmpi--4.1.6--nvhpc--23.11
cfitsio/4.3.0--gcc--12.2.0
cgal/4.11--openmpi--4.1.6--gcc--12.2.0
cgal/5.5.2--gcc--12.2.0
cgal/5.5.2--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0
cineca-hpyc/2023.05
cudnn/8.9.7.29-12--gcc--12.2.0-cuda-12.1
cutensor/1.5.0.3--gcc--12.2.0-cuda-12.1
cutensor/2.0.1.2--gcc--12.2.0-cuda-12.1
cutensor/2.0.2.5--gcc--12.2.0-cuda-12.1
elpa/2023.05.001--openmpi--4.1.6--gcc--12.2.0-cuda-12.1
fftw/3.3.10--gcc--12.2.0
fftw/3.3.10--openmpi--4.1.6--gcc--12.2.0
fftw/3.3.10--openmpi--4.1.6--nvhpc--23.11
fftw/3.3.10--openmpi--4.1.6--nvhpc--24.3
gdal/3.7.3--gcc--12.2.0
magma/2.7.2--gcc--12.2.0-cuda-12.1
metis/5.1.0--gcc--12.2.0
metis/5.1.0--oneapi--2023.2.0
nccl/2.19.1-1--gcc--12.2.0-cuda-12.1
nccl/2.19.3-1--gcc--12.2.0-cuda-12.1
netcdf-c/4.9.2--gcc--12.2.0
netcdf-c/4.9.2--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0
netcdf-c/4.9.2--oneapi--2023.2.0
netcdf-c/4.9.2--openmpi--4.1.6--gcc--12.2.0
netcdf-c/4.9.2--openmpi--4.1.6--nvhpc--23.11
netcdf-fortran/4.6.1--gcc--12.2.0
netcdf-fortran/4.6.1--intel-oneapi-mpi--2021.10.0--oneapi--2023.2.0
netcdf-fortran/4.6.1--oneapi--2023.2.0
netcdf-fortran/4.6.1--openmpi--4.1.6--gcc--12.2.0
netcdf-fortran/4.6.1--openmpi--4.1.6--nvhpc--23.11
netlib-scalapack/2.2.0--openmpi--4.1.6--gcc--12.2.0
netlib-scalapack/2.2.0--openmpi--4.1.6--nvhpc--23.11
openblas/0.3.24--gcc--12.2.0
openblas/0.3.24--gcc--12.2.0-ilp64
openblas/0.3.24--nvhpc--23.11
openblas/0.3.24--nvhpc--23.11-ilp64
openmpi/4.1.6--gcc--12.2.0
openmpi/4.1.6--nvhpc--23.11
openmpi/4.1.6--nvhpc--24.3
```

Module environment

Loading a module

Define or modify the environment variables, allowing to use the executable or libraries

```
$ module load namd/3.0--gcc--12.2.0-cuda-12.1
```

$ module show <module_name>/<version>

Prints information about the module: dependencies, paths

```
[otrocon1@login05 ~]$ module show namd/3.0--gcc--12.2.0-cuda-12.1
-------------------------------------------------------------------
/leonardo/prod/opt/modulefiles/chem-phys/applications/namd/3.0--gcc--12.2.0-cuda-12.1:

module-whatis   {NAMD is a parallel molecular dynamics code designed for high-performance simulation of large biomolecular systems.}
module          load charmpp/7.0.0--gcc--12.2.0-cuda-12.1-ulgmfjr
module          load fftw/3.3.10--gcc--12.2.0
module          load tcl/8.6.12--gcc--12.2.0-dnkdrf4
conflict        namd
prepend-path    PATH /leonardo/prod/spack/5.2/install/0.21/linux-rhel8-icelake/gcc-12.2.0/namd-3.0-pu5ghjdqy2baaltirlo4pjia2vm4hj4a/bin
prepend-path    CMAKE_PREFIX_PATH /leonardo/prod/spack/5.2/install/0.21/linux-rhel8-icelake/gcc-12.2.0/namd-3.0-pu5ghjdqy2baaltirlo4pjia2vm4hj4a/.
setenv          NAMD_HOME /leonardo/prod/spack/5.2/install/0.21/linux-rhel8-icelake/gcc-12.2.0/namd-3.0-pu5ghjdqy2baaltirlo4pjia2vm4hj4a
-------------------------------------------------------------------
```

Installed with spack

Module environment

```
$ module load profile/chem-phys
$ module load namd/3.0--gcc--12.2.0-cuda-12.1
$ module help namd/3.0--gcc--12.2.0-cuda-12.1
```

```
[otrocon1@login07 ~]$ module help namd/3.0--gcc--12.2.0-cuda-12.1
----------------------------------------------------------------------
Module Specific Help for /leonardo/prod/opt/modulefiles/chem-phys/applications/namd/3.0--gcc--12.2.0-cuda-12.1:

modulefile "namd/3.0--gcc--12.2.0-cuda-12.1"
using help from /cineca/prod/opt/helps/namd/3.0--gcc--12.2.0-cuda-12.1
Example of NAMD Job script with 4 MPI ranks, 4 GPUs and 8 OpenMP threads:

#!/bin/bash
#SBATCH --job-name jobname
#SBATCH -N1 --ntasks-per-node=4
#SBATCH --cpus-per-task=8
#SBATCH --time=24:00:00
#SBATCH --gres=gpu:4
#SBATCH --account=<account_nr>
#SBATCH --partition=boost_usr_prod

module purge
module load profile/chem-phys
module load namd/3.0--gcc--12.2.0-cuda-12.1

export OMP_NUM_THREADS=8
export OMP_PROC_BIND=true

namd3 +ppn 8 +setcpuaffinity +devices 0,1,2,3 md.namd > namd.log
```

The script example will be different in other cluster(s)

Module environment

How to find a module that I do not know in which profile is it?

$ modmap -m <module_name> — a command that looks for a module in all profiles

```
[otrocon1@login01 ~]$ modmap -m namd
Profile: archive
         applications
                 namd
                  2.14--gcc--11.3.0-cuda-11.8
Profile: astro
Profile: base
Profile: bioinf
Profile: chem-phys
         applications
                 namd
                 2.14--intel--2021.10.0
                 2.14--intel-oneapi-mpi--2021.10.0--intel--2021.10.0
                 3.0--gcc--12.2.0-cuda-12.1
Profile: deeplrn
Profile: eng
Profile: geo-inquire
Profile: lifesc
         applications
                 namd
                  2.14--intel--2021.10.0
                  2.14--intel-oneapi-mpi--2021.10.0--intel--2021.10.0
                  3.0--gcc--12.2.0-cuda-12.1
Profile: meteo
Profile: quantum
Profile: spoke7
Profile: statistics
```

Same version but different compilers

…

Is it important?

PRESENTATION

OUTLINE

- Disclaimer and Recap
- Storage Areas and Data Transfer
- Software environment
- **Production environment with Slurm: the user point of view**
- **Production environment with Slurm: the administrator point of view**
- **Cluster monitoring: Sanity checks**
- **Cluster monitoring: Interconnect network tests**

SLURM @ CINECA -1

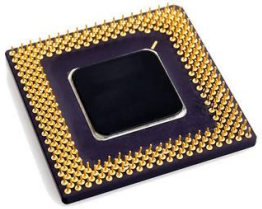
Since beginning of 2018 – migration from PBSpro to Slurm

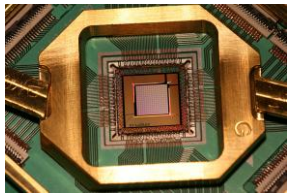
WHY?

- analysis of schedulers/resource managers proved that SLURM was **already a robust tool** to manage resources and schedule jobs in hybrid architectures (thinking of ongoing trends in HPC)

- analysis of CINECA production environment core pillars (managing different communities with specific requests, huge loads of jobs - 1000+ avg in an hour, fair use of resources) proved that SLURM was quite **easily compliant with our needs**

SLURM @ CINECA -2





- as all supercomputing centers, **CINECA follows the HPC architectures development**, resulting in adopting new, even prototype architectures

- as one cluster is dismissed, no guarantee that the new cluster will be similar to the previous one (quite the opposite actually)

- once devised the general scheduler configuration suitable to CINECA's production needs, it's now a **near zero effort** to set up the production environment (partitions, QOS, scheduler parameters, resource management) for hybrid and more and more complex architectures

Jobscript example

```
#!/bin/bash
#SBATCH -t 1:00:00
#SBATCH -N 2
#SBATCH --ntasks-per-node=16
#SBATCH --cpus-per-task=2
#SBATCH --gres=gpu:4
#SBATCH --mem=10GB
#SBATCH -o job.out
#SBATCH -e job.err
#SBATCH -p boost_usr_prod
#SBATCH -A <my_account>
module load openmpi/4.1.4--gcc--11.3.0-cuda-11.8
export OMP_PROC_BIND=true
mpirun -n 32 ./myprogram
```

Slurm directives: resource requirements

```
#SBATCH --nodes=1, -N 1
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --mem=10000        # mem=0 equals to full memory
```

nodes – number of compute nodes
ntasks-per-node – number of tasks per node (max. 32 for Booster, 112 for DCGP)
cpus-per-task – number of cpus to be assigned to each task
BOOSTER: ntasks-per-node*cpus-per-task ≤ 32
DCGP: ntasks-per-node*cpus-per-task ≤ 112
mem – memory allocated for each node (max=494000 MB).

Slurm directives:
General resources (GRES)

Booster:

#SBATCH --gres=gpu:4

gres=gpu:x – number of GPUs for each node (x=1..4)

DCGP and serial:

#SBATCH --gres=tmpfs:200G

gres=tmpfs:x – quota of temporary filesystem /tmp local to the node, that can be requested on serial and DCGP nodes (maximum: 1TB for serial, 3TB for DCGP. Default=10GB)

Slurm directives: walltime and partitions

#SBATCH --time=00:30:00, -t 00:30:00

Specifies the maximum duration of the job. The maximum time allowed depends on the partition used

Pro-tip: the less walltime you ask, the faster your job will enter in execution. Think about it!

#SBATCH --partition=boost_usr_prod, -p boost_usr_prod
#SBATCH --qos=boost_qos_dbg, -q boost_qos_dbg (optional)

Specifies the “partition”, a.k.a. the specific set of nodes among which your job can search for resources. Optionally you can specify a QoS (Quality of Service) for jobs with particular purposes, like debugging or large production

Partitions and QoS: General structure

Some definitions

- Modules or hardware partition: a group of nodes that have similar hardware features (CPUs, GPUs).
- Slurm partitions: a set of nodes that are regrouped to represent the hardware partitions at Slurm level.
- Quality of Service (QoS): parameters that set the limitations on partitions or on jobs.
- Partition QoS: QoS attached to the Slurm partition, to setup some usage limits.
- Compute nodes may belong to multiple Slurm partitions, each with their own partition QOS.
- Job QoS: a QoS that can be added to the job to overcome some limitations, but may add other (ex: no more than 1 job running with this QoS)

General structure

- **Partitions**: very few → we define a single partition with all compute nodes and rely on QOS to satisfy the request of different communities.
- **QOS:** quite a rich variety to manage the variety of jobs'types.
- **FairShares:** linearized-ish use of resources on a monthly scale to ensure that all users can use the granted hour budgets while enforcing a democratic use of resources.
-
- **Sheduler Parameters:** backfill, packing of serial jobs (flag: pack_serial_at_end), etc. to optimize/maximize the use of resources.

QOS FOR FLEXIBILITY

We rely a lot on QoS to be able to keep our clusters constantly filled with users 24/7, while looking after the special needs that some may have, asking for an help with binding some rules when necessary.

Examples include:

qos_bprod: for jobs of bigger size, a QoS is set with a minimum requirement of resources and a large new maximum. These jobs have high priority but no more than 1 or 2 are allowed at the same time, to avoid the monopoly of the cluster;

qos_lprod: for jobs of bigger walltime. Regular QoS allow for up to 24h for fairshare reasons, but for some works (e.g. molecular dynamics) it may not be enough;

qos_dbg: jobs with less than two hours of walltime and two nodes can use an high priority QoS for debugging purposes;

qos_lowprio: if your budget is depleted or your time is expired, you can use a qos for continue running with no charge: however, your priority is so low that your job is considered only if there are no other "legit" jobs in queue;

qos_special: an user can ask for this if they need a longer walltime or a very high number of nodes. We stipulate with them the number of jobs that they can submit and remove the QoS when the work is done.

Available partitions and QoS on LEONARDO Booster

SLURM partition	Job QOS	# cores/# GPU per job	max walltime	max running jobs per user/ max n. of nodes/cores/GPUs per user	priority	notes
lrd_all_serial (default)	*normal*	max = 4 physical cores (8 logical cpus) max mem = 30800 MB	04:00:00	1 node / 4 cores / 30800 MB	40	No GPUs Hyperthreading x2
boost_usr_prod	*normal*	max = 64 nodes	24:00:00		40	
	boost_qos_dbg	max = 2 nodes	00:30:00	2 nodes / 64 cores / 8 GPUs	80	
	boost_qos_bprod	min = 65 nodes max =256 nodes	24:00:00	256 nodes	60	runs on 1536 nodes min is 65 FULL nodes
	boost_qos_lprod	max = 3 nodes	4-00:00:00	3 nodes /12 GPUs	40	

Note:

- the partition **lrd_all_serial** runs on front-end nodes, and as such it is not subject to accounting and can be used for free

Available partitions and QoS on LEONARDO DCGP

SLURM partition	Job QOS	# cores/ # GPU per job	max walltime	max n. of nodes/cores/mem per user max n. of nodes per account	priority	Notes
lrd_all_serial (default)	*normal*	max = 4 physical cores (8 logical cpus) max mem = 30800 MB	04:00:00	1 node / 4 cores / 30800 MB	40	No GPUs Hyperthreading x2
dcgp_usr_prod	*normal*	max = 16 nodes	24:00:00	512 nodes per account	40	
	dcgp_qos_dbg	max = 2 nodes	00:30:00	2 nodes / 224 cores per user 512 nodes per account	80	
	dcgp_qos_bprod	min = 17 nodes max =128 nodes	24:00:00	128 nodes per user 512 nodes per account	60	GrpTRES=1536 node min is 17 FULL nodes
	dcgp_qos_lprod	max = 3 nodes	4-00:00:00	3 nodes / 336 cores per user 512 nodes per account	40	

Note:

- a maximum of **512 nodes per account** is imposed, meaning that, all the jobs associated with a particular account cannot run on more than 512 nodes at the same time.

Slurm directives: accounting

#SBATCH --account=<my_account>, -A <my_account>

Specifies the account to use the CPU hours from.

You can check the status of your account with the command "*saldo -b*", which tells you how many CPU hours you have already consumed for each account you're assigned at (a more detailed report is provided by "*saldo -r*").

Note: for DCGP projects, the command is "*saldo -b/-r --dcgp*"

```
[amarani0@login01 ~]$ saldo -b
--------------------------------------------------------------------------------------------------------------------------
account                start         end          total        localCluster   totConsumed     totConsumed     monthTotal     monthConsumed
                                                (local h)   Consumed(local h)    (local h)               %     (local h)         (local h)
--------------------------------------------------------------------------------------------------------------------------
cin_staff           20110323    20300323     200000002            16382068      50270876            25.1        864553             785338
cin_propro          20220427    20301231        500000                2568          2695             0.5          4731                  0
cin_saldo           20230524    20300323            10                   0             0             0.0             0                  0
cin_sudo            20230524    20300323            10                   0             0             0.0             0                  0
FUSIO_TEST_4        20161116    20231231          1000                 191           191            19.2            11                149
cin_external_6      20150319    20231231         20000                7695          7695            38.5           186                103
cin_extern01_3      20170210    20231231          5000                   0             0             0.0            59                  0
```

Submitting a job

You have created your jobscript! Congratulations!!!
…now, what to do with it?

sbatch

sbatch <job_script>

Your job will be submitted to the SLURM scheduler and executed when there will be nodes available (according to your priority and the partition you requested)

squeue -u

squeue -u <username>, squeue --me

Shows the list of all your scheduled jobs, along with their status (idle, running, closing, …)
It also shows you the job id required for other SLURM commands

Other Slurm commands - 1

```
[amarani0@login02 ~]$ sbatch simplejob
Submitted batch job 13781289
[amarani0@login02 ~]$ squeue -u amarani0
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          13781289 boost_usr test_sim amarani0 PD       0:00      2 (Priority)
```

scontrol show job

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 is not starting and, if it is scheduled with top priority, you will get an estimated start time

scancel

scancel <job_id>, scancel --me

Removes the job(s) (queued or running) from the scheduled job list by killing them

Other Slurm commands - 2

sinfo

sinfo -p <partition name>

sinfo -l

sinfo -N -l -p boost_usr_prod

Provides information about SLURM nodes and partitions

sacct

sacct OPTIONS <job_id>

sacct -e (for the list of format options)

displays accounting data for all jobs and job steps in the SLURM job accounting log or Slurm database. Can also see the jobs already completed or cancelled

```
[amarani0@login01 ~]$ sacct -X --format=jobid,user,account,start,nnodes,elapsed
JobID             User    Account                Start   NNodes    Elapsed
------------ --------- ---------- ------------------- -------- ----------
13851221      amarani0  cin_staff 2025-03-13T10:06:12        1   00:06:59
13851222      amarani0  cin staff 2025-03-13T10:06:12        1   00:07:03
```

Interactive batch jobs

```
[amarani0@login02 ~]$ srun -N 1 --ntasks-per-node=4 --gres=gpu:2 -A cin_staff -p boost_usr_prod --pty /bin/bash
srun: job 13781476 queued and waiting for resources
srun: job 13781476 has been allocated resources
[amarani0@lrdn3399 ~]$
```

```
[amarani0@login02 ~]$ salloc -N 1 --ntasks-per-node=4 --gres=gpu:2 -A cin_staff -p boost_usr_prod
salloc: Pending job allocation 13781517
salloc: job 13781517 queued and waiting for resources
salloc: job 13781517 has been allocated resources
salloc: Granted job allocation 13781517
salloc: Waiting for resource configuration
salloc: Nodes lrdn3399 are ready for job
[amarani0@login02 ~]$
```

Keep in mind that you are using computing nodes, meaning that you are consuming compute hours!

To exit from an interactive session, just type “*exit*”
(easy to forget with salloc)

PRESENTATION

OUTLINE

- Disclaimer and Recap
- Storage Areas and Data Transfer
- Software environment
- Production environment with Slurm: the user point of view
- **Production environment with Slurm: the administrator point of view**
- **Cluster monitoring: Sanity checks**
- **Cluster monitoring: Interconnect network tests**

GENERAL STRUCTURE (repeat)

- **partitions**: very few -> we now tend to define a single physical partition with all nodes, and rely on logical partitions and their QOS to deal with the requests of different communities

- **QOS**: quite a rich variety of them to manage the quite rich variety of jobs' types -> debug, big/long production, etc. QOS priorities and appropriate QOS's GRES limits

scheduler parameters: backfill, packing of serial jobs, preemption etc. to optimize/maximize the use of resources

- **fairshares**: linearized-ish use of resources on a monthly scale to ensure that all users can use the granted hour budgets while enforcing a democratic use of resources

- **in-house slurmd prologs/epilogs**: for temporary job's areas, for safely loading/unloading drivers when needed by jobs - e.g., intel sep drivers or system power monitoring - , for system managed nvidia mps, etc.

DYNAMIC PARTITIONING

Problem: There are two partitions insisting on the same nodes: *dcgp_usr_prod* and *dcgp_cmcc_prod*.

The first is for all users, the second is for a contracted community that needs 165 DCGP nodes always available for them

- **Standard solution**: dedicated partition. The community gets a new partition with dedicated nodes, separated from the rest.
- **Problem**: if one of those node breaks, they are left with one node less of what is contracted, until we intervene by fixing the nodelist of the partition

Our solution:
Defining both partitions so that they both insist on all the DCGP nodes, then manage the request via job qos parameters

DYNAMIC PARTITIONING

```
[amarani0@login02 SLURM]$ sacctmgr show qos dcgp_usr_prod format=name,flags,grptres%50
      Name                Flags                                              GrpTRES
---------- -------------------- --------------------------------------------------
dcgp_usr_+          DenyOnLimit                                cpu=139216,mem=614042000M
```

This is the equivalent of **1243 nodes**. Users can fill up the regular partition until that limit is touched.

```
[amarani0@login02 SLURM]$ sacctmgr show qos dcgp_cmcc_prod format=name,flags,grptres%50
      Name                Flags                                              GrpTRES
---------- -------------------- --------------------------------------------------
dcgp_cmcc+          DenyOnLimit                      cpu=18480,mem=81510000M,node=165
```

This partition can't occupy more than **165 nodes**, but most importantly, since dcgp_usr_prod is limited they will always find those 165 nodes at their disposal throughout the physical partition.

That way, even if a node breaks it isn't bound to a dedicated partition and the users can find the resources nonetheless

NOTE: broken nodes are removed from the dcgp_usr_prod count by procedures developed by the vendor technical team

Slurm user roles

A quick check with Slurm commands show that there are different levels of privileges as Slurm users on a cluster

```
[amarani0@login02 ~]$ sacctmgr show user a08trb01
      User   Def Acct     Admin
---------- ---------- ---------
  a08trb01 tra24_ope+      None
```

```
[amarani0@login02 ~]$ sacctmgr show user amarani0
      User   Def Acct     Admin
---------- ---------- ---------
  amarani0  cin staff  Operator
```

```
[amarani0@login02 ~]$ sacctmgr show user root
      User   Def Acct     Admin
---------- ---------- ---------
      root       root Administ+
```

A regular Slurm user can:
Submit jobs, monitor the queue, cancel only their own jobs, use report commands, …

A Slurm Operator (User Support) can:
Setup reservations, prioritize any job, cancel jobs of everyone, …

A Slurm Administrator (Root/Sysadmins) can:
Create partitions and qos, setup qos parameters and limits, …

You can manage your staff and alleviate part of your own workload by setting up Slurm roles within the team

Slurm.conf

Want to take a peek?

`/var/spool/slurmd/conf-cache/slurm.conf`

```
MaxJobCount=100000
MaxArraySize=300000
#MessageTimeout=100
MessageTimeout=60
OverTimeLimit=3
SchedulerParameters=spec_cores_first,pack_serial_at_end,sched_min_interval=2000000,max_rpc_cnt=150,bf_continue,bf_interval=120,bf_yield_interval=2000000,bf_yi
eld_sleep=500000,bf_max_job_part=200,bf_max_job_user_part=20,bf_max_job_test=1000,bf_resolution=300,bf_max_time=150,default_queue_depth=2000,bf_job_part_count
_reserve=40,enable_user_top
ResumeTimeout=1800
```

Part of the scheduling parameters. Check out the high presence of backfill parameters, an important system to keep the cluster occupied as much as possible

```
        PartitionName=boost_usr_prod QoS=boost_usr_prod Default=NO  DefMemPerCPU=15400 DefaultTime=00:30:00 MaxNodes=64 MaxTime=24:00:00 State=UP
        PartitionName=boost_fua_prod QoS=boost_fua_prod Default=NO  DefMemPerCPU=15400 DefaultTime=00:30:00 MaxNodes=16 MaxTime=24:00:00 State=UP
        PartitionName=boost_fua_dbg  QoS=boost_fua_dbg  Default=NO  DefMemPerCPU=15400 DefaultTime=00:10:00 MaxNodes=2  MaxTime=00:10:00 State=UP
        PartitionName=dcgp_usr_prod  QoS=dcgp_usr_prod  Default=NO  DefMemPerCPU=4400  DefaultTime=00:30:00 MaxNodes=16 MaxTime=24:00:00 State=UP
        PartitionName=dcgp_fua_prod  QoS=dcgp_fua_prod  Default=NO  DefMemPerCPU=4400  DefaultTime=00:30:00 MaxNodes=16 MaxTime=24:00:00 State=UP
        PartitionName=dcgp_fua_dbg   QoS=dcgp_fua_dbg   Default=NO  DefMemPerCPU=4400  DefaultTime=00:10:00 MaxNodes=2  MaxTime=00:10:00 State=UP DenyQos=dcgp
_qos_fuabprod
        PartitionName=dcgp_cmcc_prod  QoS=dcgp_cmcc_prod  Default=NO  DefMemPerCPU=4400  DefaultTime=00:30:00 MaxNodes=16 MaxTime=24:00:00 State=UP AllowAccou
nts=cin_sanity,CMCC_2025
```

Production partitions. Nodenames are not hard-coded, but are dynamic (part of the system setup by the vendors discussed earlier).

Check out also other interesting configuration files in that folder: **gres.conf**, **topology.conf** and many others

PRIORITY FORMULA AND PARAMETERS

```
PriorityDecayHalfLife=4-00:00:00
PriorityCalcPeriod=00:05:00
PriorityFavorSmall=No
PriorityFlags=SMALL_RELATIVE_TO_TIME,DEPTH_OBLIVIOUS,NO_FAIR_TREE,MAX_TRES
PriorityMaxAge=7-00:00:00
PriorityUsageResetPeriod=MONTHLY
PriorityWeightAge=20000
PriorityWeightAssoc=0
PriorityWeightFairShare=25000
PriorityWeightJobSize=10000000
PriorityWeightPartition=0
PriorityWeightQOS=300000
```

Formula: *QOSWeight*QOSValue* + *FSWeight*FSValue*+*AgeWeight*AgeValue*+*JobSizeWeight*JobSizeValue*

The weights are set so that the parameters have an **order of importance**

QOS

FairShare

Age

JobSize

Example:

Two jobs in the same partition and same qos ==> compete in terms of FairShare
Two jobs on different qos ==> the job in the qos with higher priority wins regardless of the other weights

Budget linearization

Why we need to implement a budget linearization ?

- High number and variety of users → **5308** active users at the end of 2023

- Variety of project with different budget sizes and duration

EUROHPC
Try projects
ISCRA's
EUROFUSION
Agreements

- We need to avoid big budget projects/users to **monopolize** the use of the clusters in order to be more democratic with all of our users.

- We want to encourage the users to spend their resources regularly, to avoid the situation where a project doesn't consume budget for months and then tries to spend all of it in a few days, without success.

Fairshare and shares administration

Budget linearization is administered via fairshare (remember the priority parameters in slurm.conf)

We wrote a procedure that assigns to each project a number of RawShares **proportional to the number of CPU hours they have to spend**. While the project families act as a "father" to the accounts related to it, we want the fair-share to not take in consideration neither the relationship between father and son, nor the relationship between the siblings, because **any account should have its own personal budget** represented by its own personal number of shares...

To guarantee fairness, the script that sums the raw shares of each account belonging to the same family, and assigns that number to the father, instead of the default "1". In this way there is a **proportion between fathers** as well as between sons, so the global proportion is respected."

```
            # Se l'account è attivo e presente nel file delle shares personalizzate, setta il suo numero di shares a
 quello indicato nel file
            r_flag=0
            for r in res:
              if r[0]==s[0]:
                r_flag=1
                v_print("Account " + s[0] + " has personalized budget: setting fair share to " + r[1])
                calc_share=int(r[1])
            if r_flag==0:
            #se l'account non ha incontrato eccezioni, le sue shares sono il suo budget in ore standard moltiplicato
per il fairshare scale (1/numero di mesi dell'account)
               fs_scale=1.0/diff
               calc_share=int(budget*fs_scale)

      #somma delle shares dell'account a quelle del progetto padre
      for f in fathers:
            search=f[0] + "_"
            if search in s[0]:
               f[2]+=calc_share
```

```
      #assegnamento effettivo delle nuove shares all'account. Questo viene fatto solo se ci sono effettive modifiche
rispetto al valore gia'registrato
      if s[1]!=str(calc_share):
          if calc_share==-1:
             if options.force:  #il default e'non fare nulla, con la flag -f le modifiche sono effettive
                print("Account " + s[0] + " changed its share from " + s[1] + " to parent: Modifying fair share value
")
                subprocess.Popen(['/opt/slurm/current/bin/sacctmgr','update','account',s[0],'set','fairshare=parent',
'where','cluster=' + clus,'-i'], stdout=subprocess.PIPE,universal_newlines=True)
                time.sleep(3)  #sleep di tre secondi perche' troppi sacctmgr di fila possono piantare lo slurmdb
             else:
                print("DRY-RUN: Account " + s[0] + " changed its share from " + s[1] + " to parent: Modifying fair sh
are value")
          else:
             if options.force:
                print("Account " + s[0] + " changed its share from " + s[1] + " to " + str(calc_share) + " : Modifyin
g fair share value")
                subprocess.Popen(['/opt/slurm/current/bin/sacctmgr','update','account',s[0],'set','fairshare=' + str(
calc_share),'where','cluster=' + clus,'-i'], stdout=subprocess.PIPE,universal_newlines=True)
                time.sleep(3)
             else:
                print("DRY-RUN: Account " + s[0] + " changed its share from " + s[1] + " to " + str(calc_share) + " :
 Modifying fair share value")
```

So many other topics...

- Populating the Slurm database via LDAP consulting
- Managing exceptions to the rules via special QoS or reservations
- Prologs, epilogues and job_submit.lua
- Slurm gres for profiling tools and Perf permissions
- ...and much more! But let's stop here for now, shall we? :-)

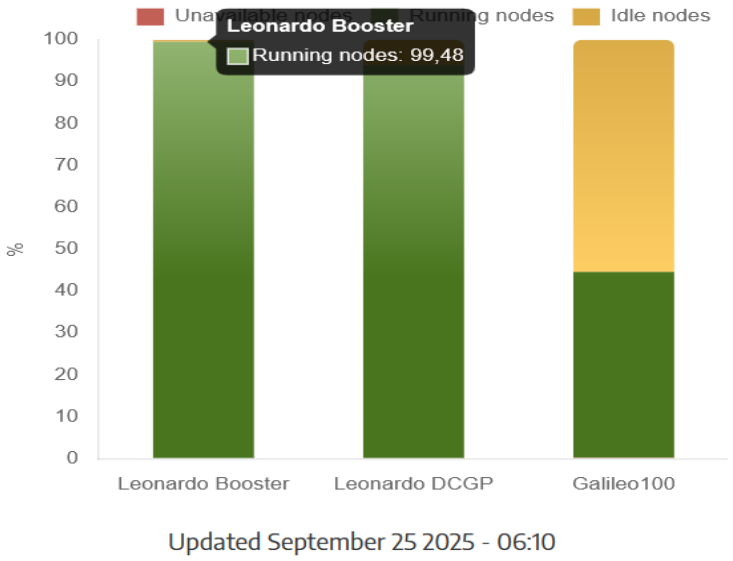
PRESENTATION

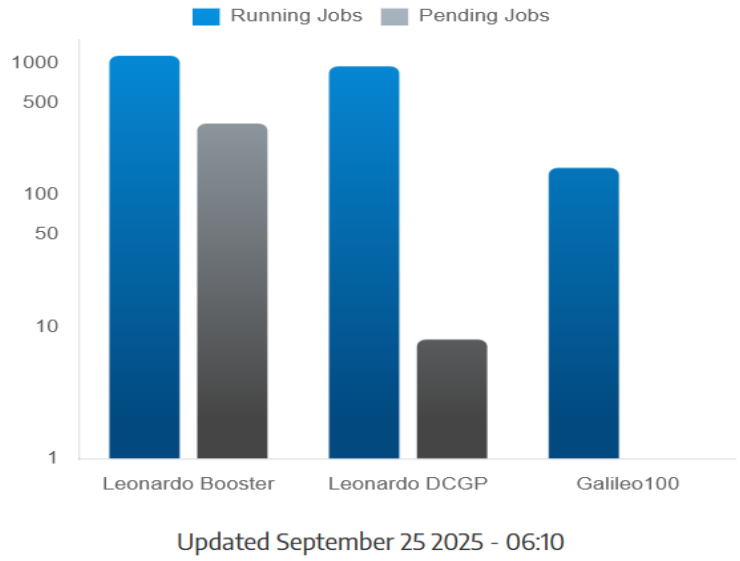
OUTLINE

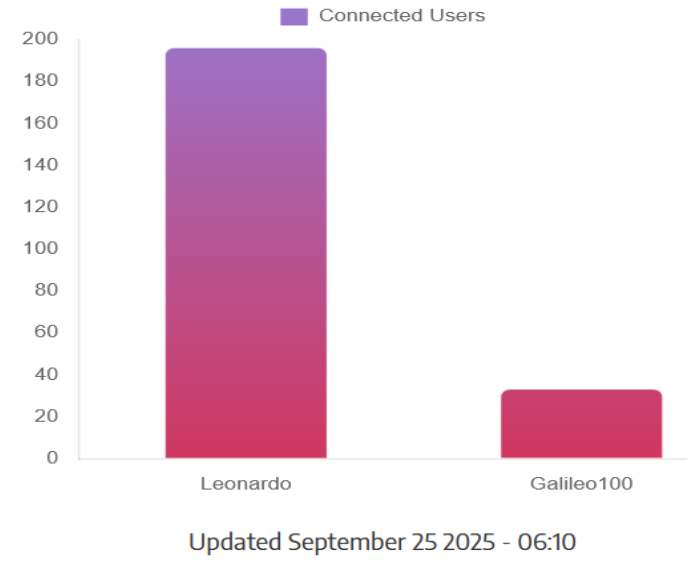
- Disclaimer and Recap
- Storage Areas and Data Transfer
- Software environment
- Production environment with Slurm: the user point of view
- Production environment with Slurm: the administrator point of view
- **Cluster monitoring: Sanity checks**
- **Cluster monitoring: Interconnect network tests**

Cluster monitoring – collecting data

https://www.hpc.cineca.it/







Data collected through repetitions of Slurm monitoring commands via cronjob, and elaborated into a graph through a dashboard

Sempahore informing users of the general health status of the cluster

Cluster monitoring – the User Support way

We surely all agree that monitoring constantly the health state of the cluster is important in order to maintain a stable environment and avoid events like general performance slowdowns or crashes for unknown reasons

While vendors and sysadmins have their own tools for checking the state and the performance of all nodes, sometimes there is something that escapes because those tests cannot detect it

It's up to the User Support team to perform regular regression tests and sanity checks through all the available nodes of the cluster, and notify system administrators of eventual nodes that have to be brought down of production for further analysis

Sanity checks

User Support performs a series of synthetic benchmarks called "Sanity checks" at the end of every maintenance time, and sometimes during production (though it takes a lot to complete with jobs on regular queues)

Jobs are launched on all the available nodes of the cluster, and results are then collected to an unique file for consulting.

The tests include:

GPU Nodes

- gpuburn for 60 seconds;
- Dgemm matrix multiplication test for single GPU performance;
- BabelStream GPU for memory bandwidth;
- Host2Device and Device2Host CPU-GPU bandwidth with nvbandwidth;
- P2p GPU-GPU communication bandwidth with nvbandwidth.

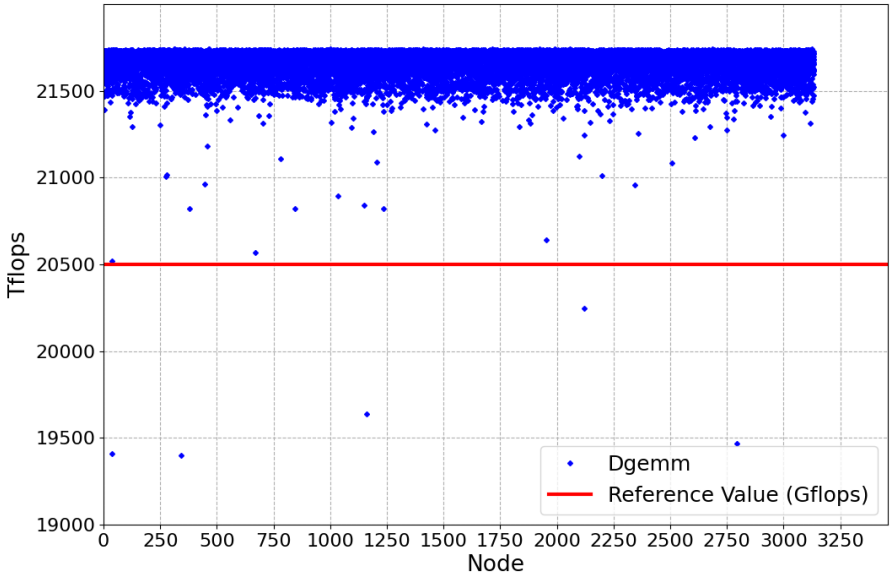
CPU and GPU nodes

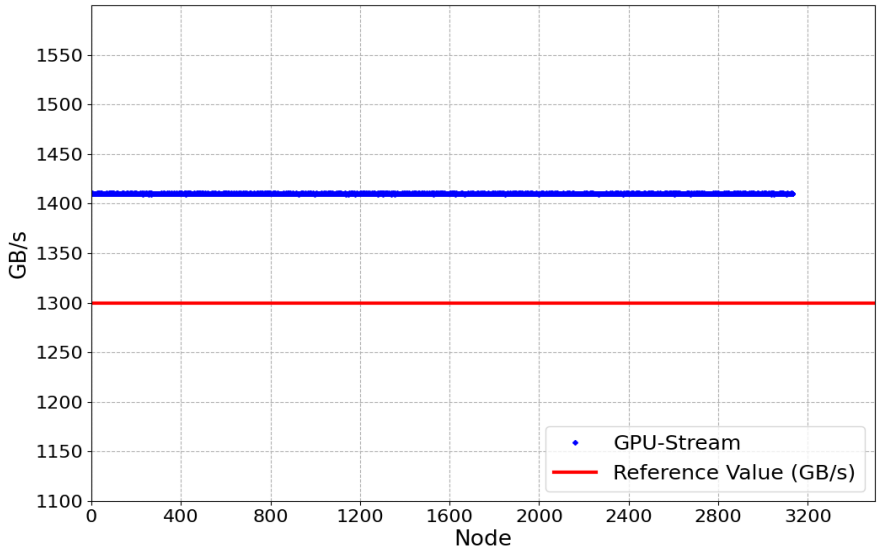
- Linpack (Intel optimized version included with mkl installation);
- Stream for memory bandwidth.

Other

- Multinode HPL on nodes connected on the same L1 switch;
- Login nodes healthness (connection attempt, basic commands working, packages installed fine);
- IOR tests for filesystem I/O stress.

Sanity checks





Eventual nodes with problems get tested again, and after the third strike are notified to the sysadmins.

The nodes will be put under reservation and out of production, and after repairment they will be retested and brough back in

If the user notifies problems on a specific node, the User Support can create a reservation on the node, test it and report the results to sysadmins in case of failures.

PRESENTATION

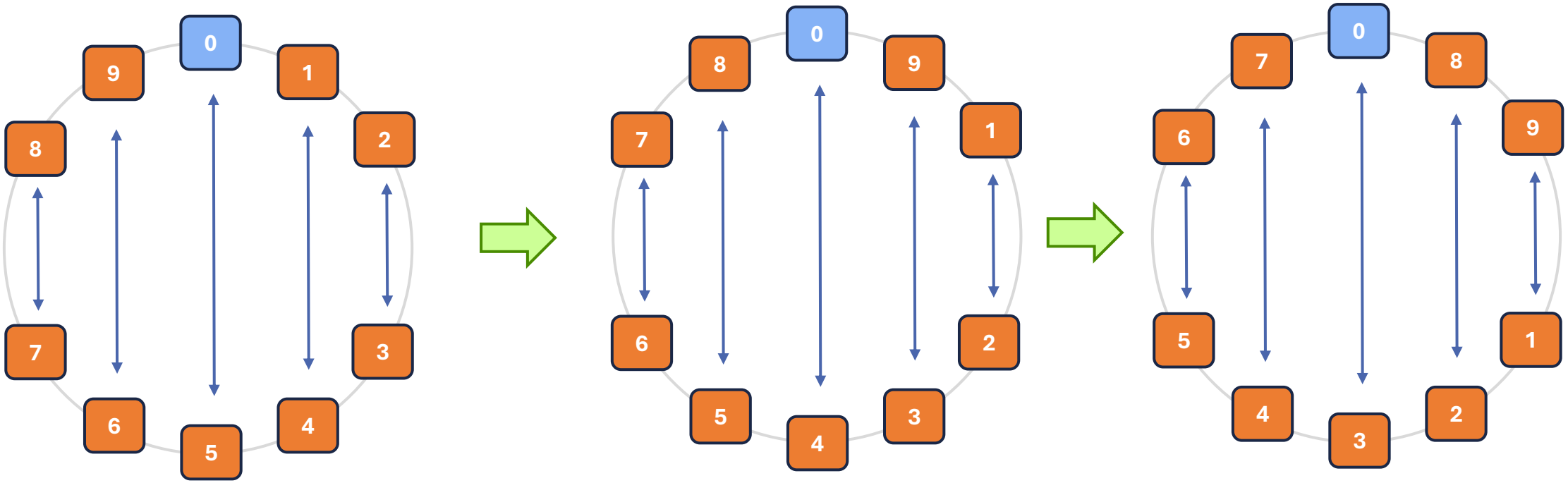
OUTLINE

- Disclaimer and Recap
- Storage Areas and Data Transfer
- Software environment
- Production environment with Slurm: the user point of view
- Production environment with Slurm: the administrator point of view
- Cluster monitoring: Sanity checks
- **Cluster monitoring: Interconnect network tests**

INTERCONNECT PERFORMANCES

Algorithm Used

n **NODEs** → ***n/2*** **PAIRs** x ***n-1*** **PERMUTATIONs** → **~n^2 COMBINATIONs**



1st ROUND **2nd ROUND** **3rd ROUND**

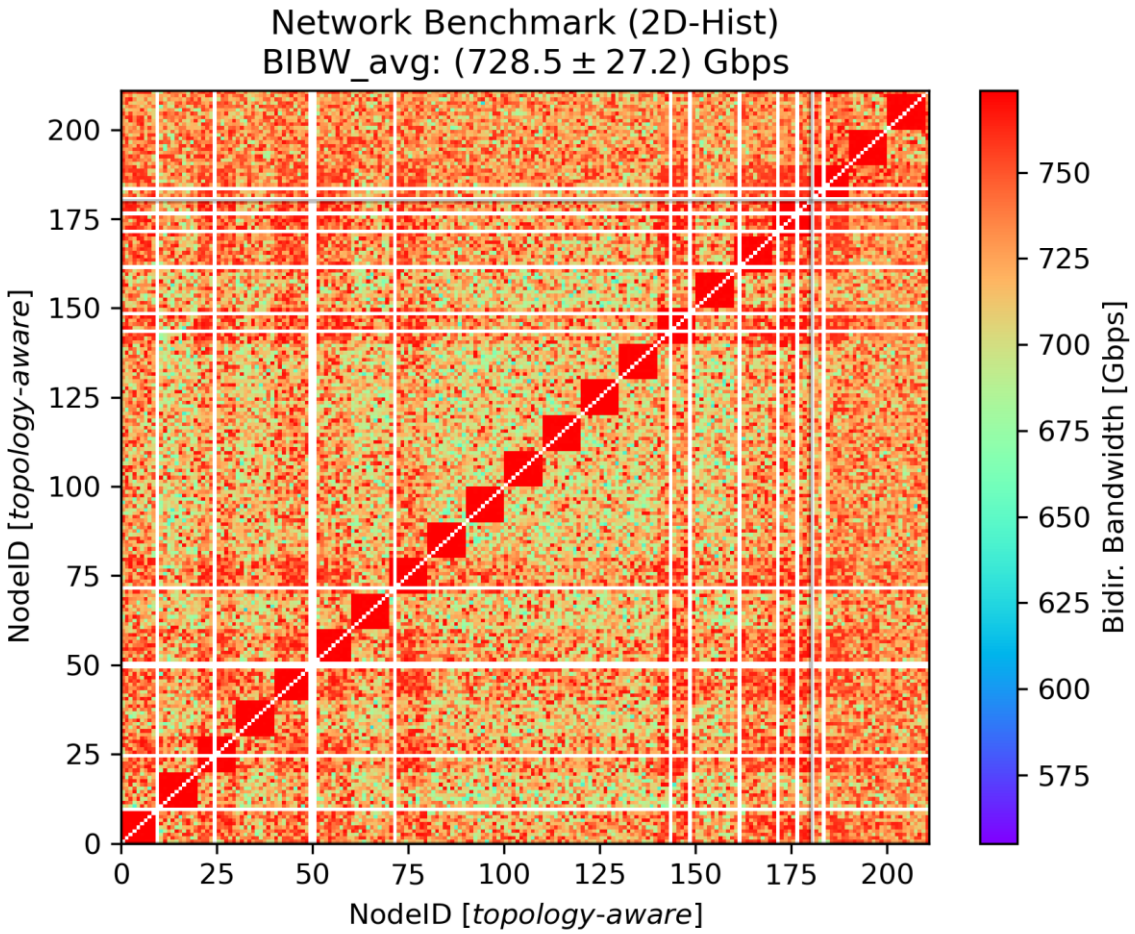
n-1 **ROUNDs*** of ***n/2*** **INDEPENDENT PAIRs** → ***n-1*** **COMPLEXITY**

**one per each permutation*

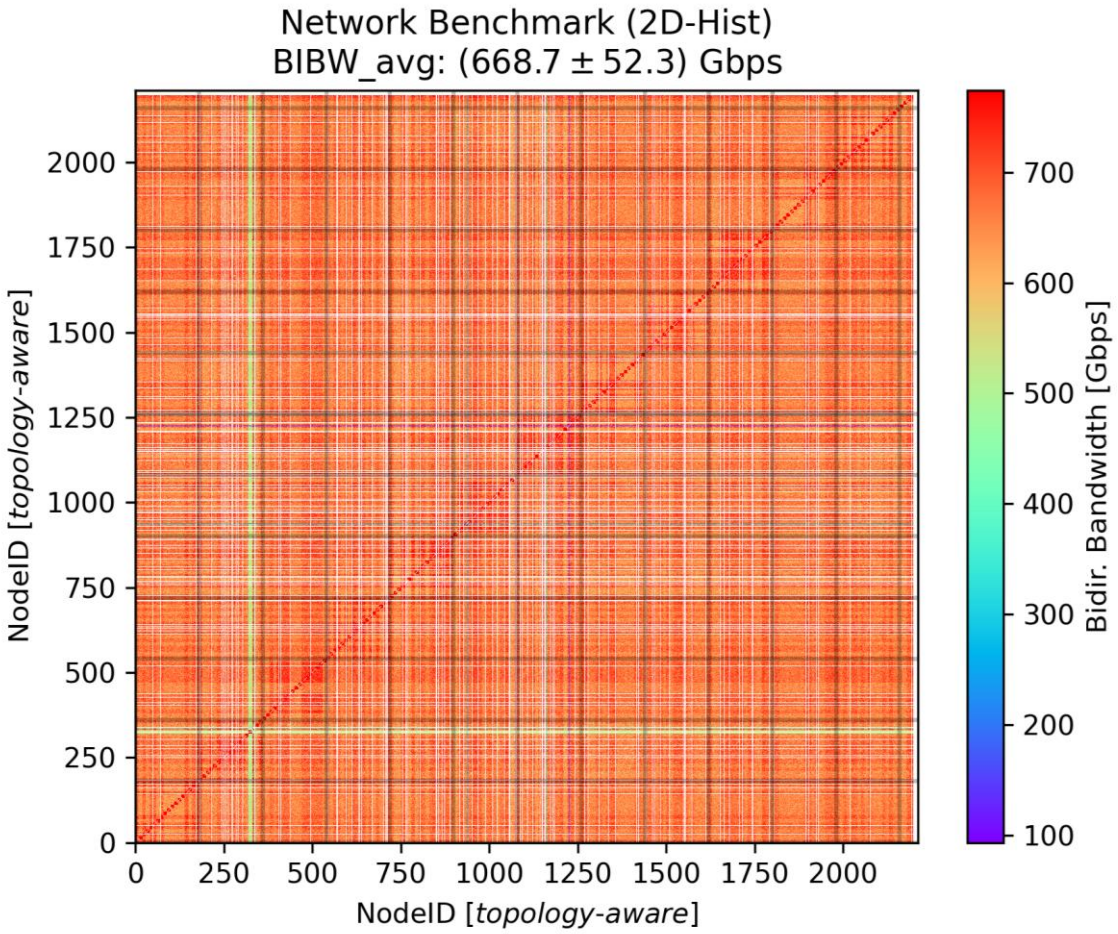
INTERCONNECT PERFORMANCES

Bidirectional Bandwith Measurements

200 NODES



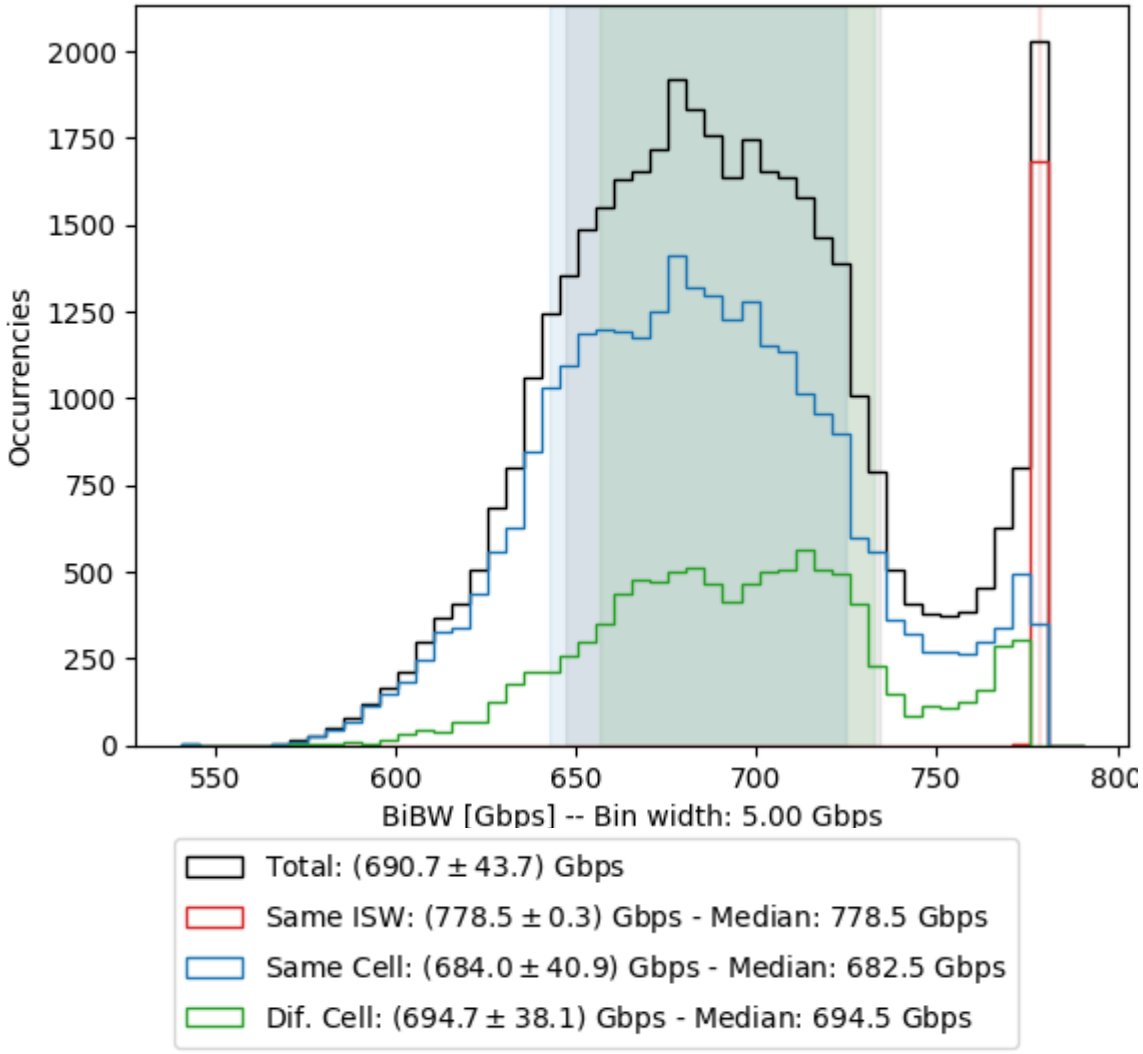
2,000 NODES



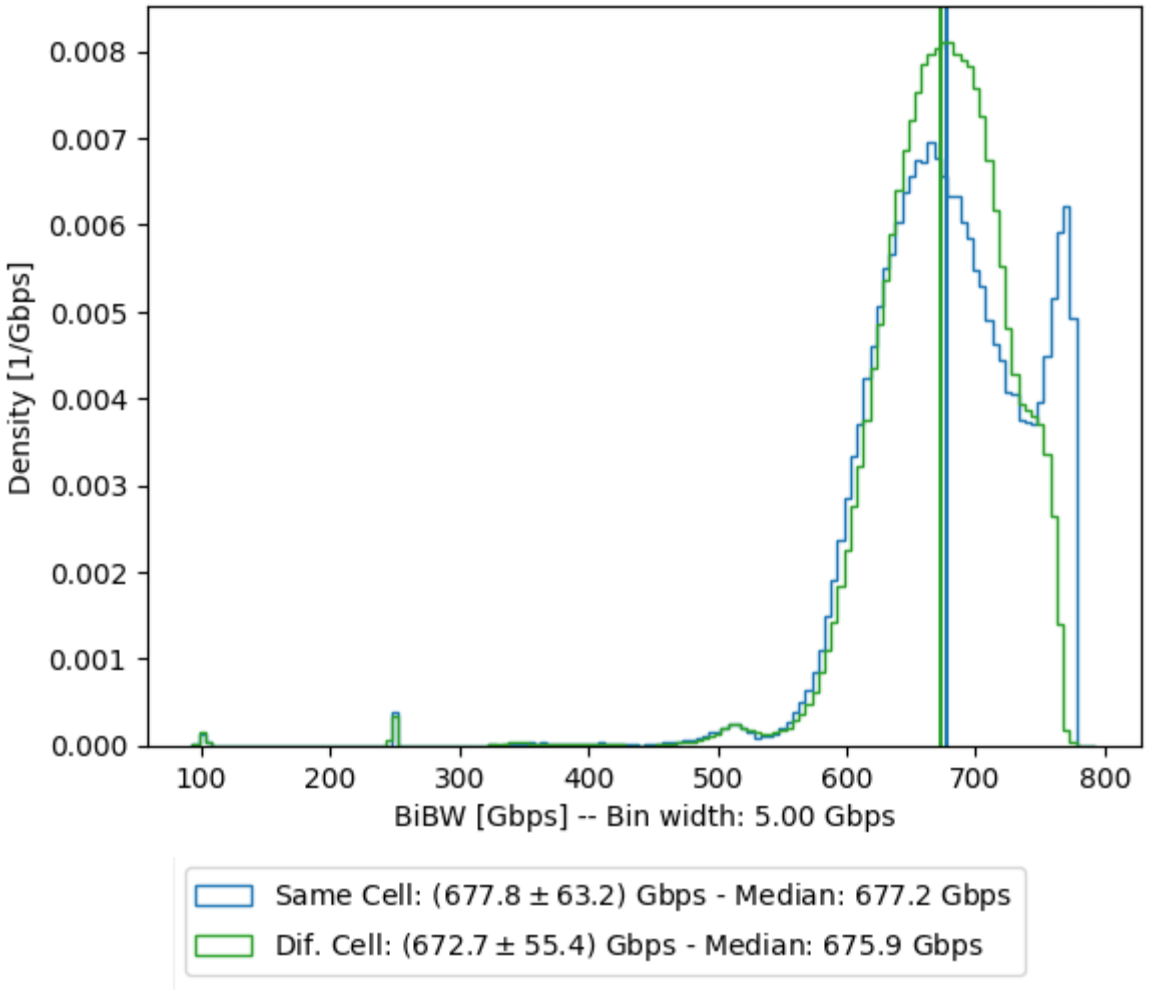
INTERCONNECT PERFORMANCES

Bidirectional Bandwith Measurements

200 NODES



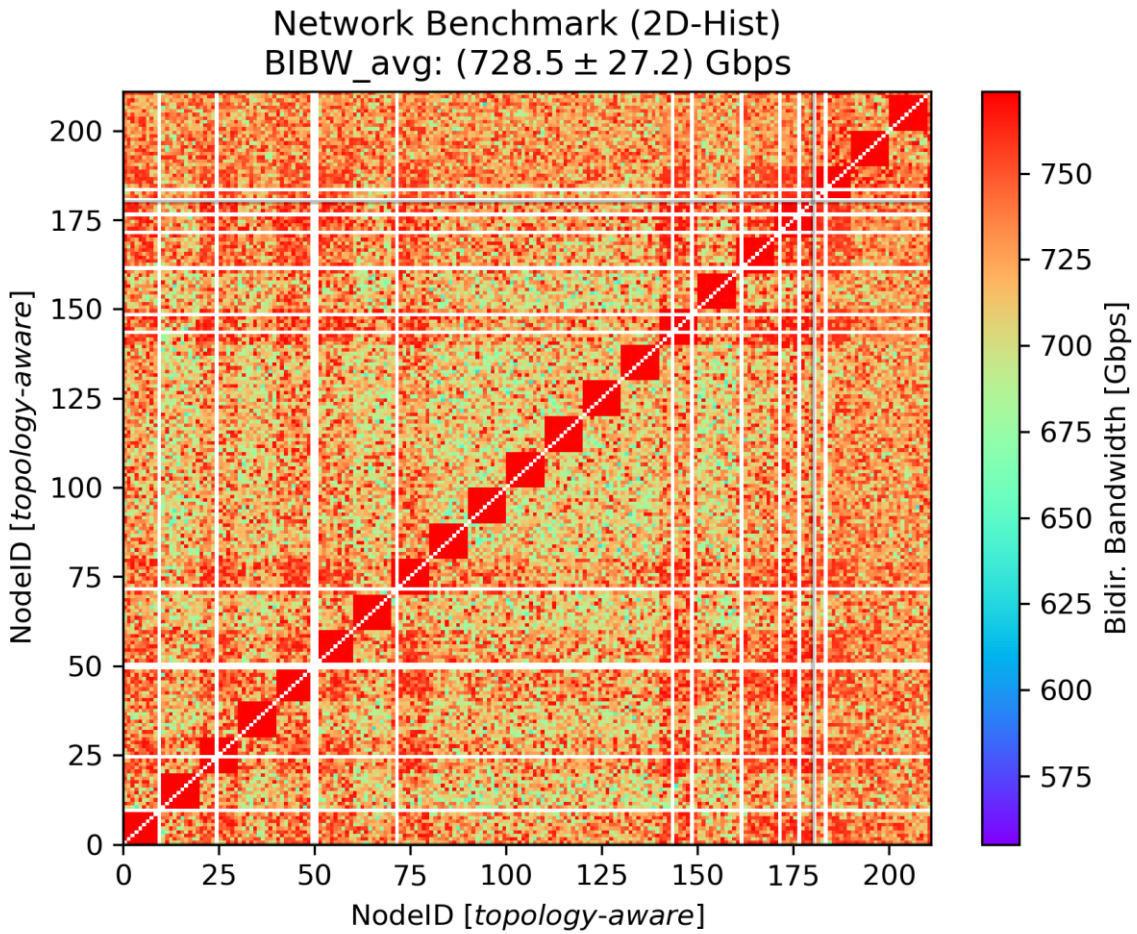
2,000 NODES

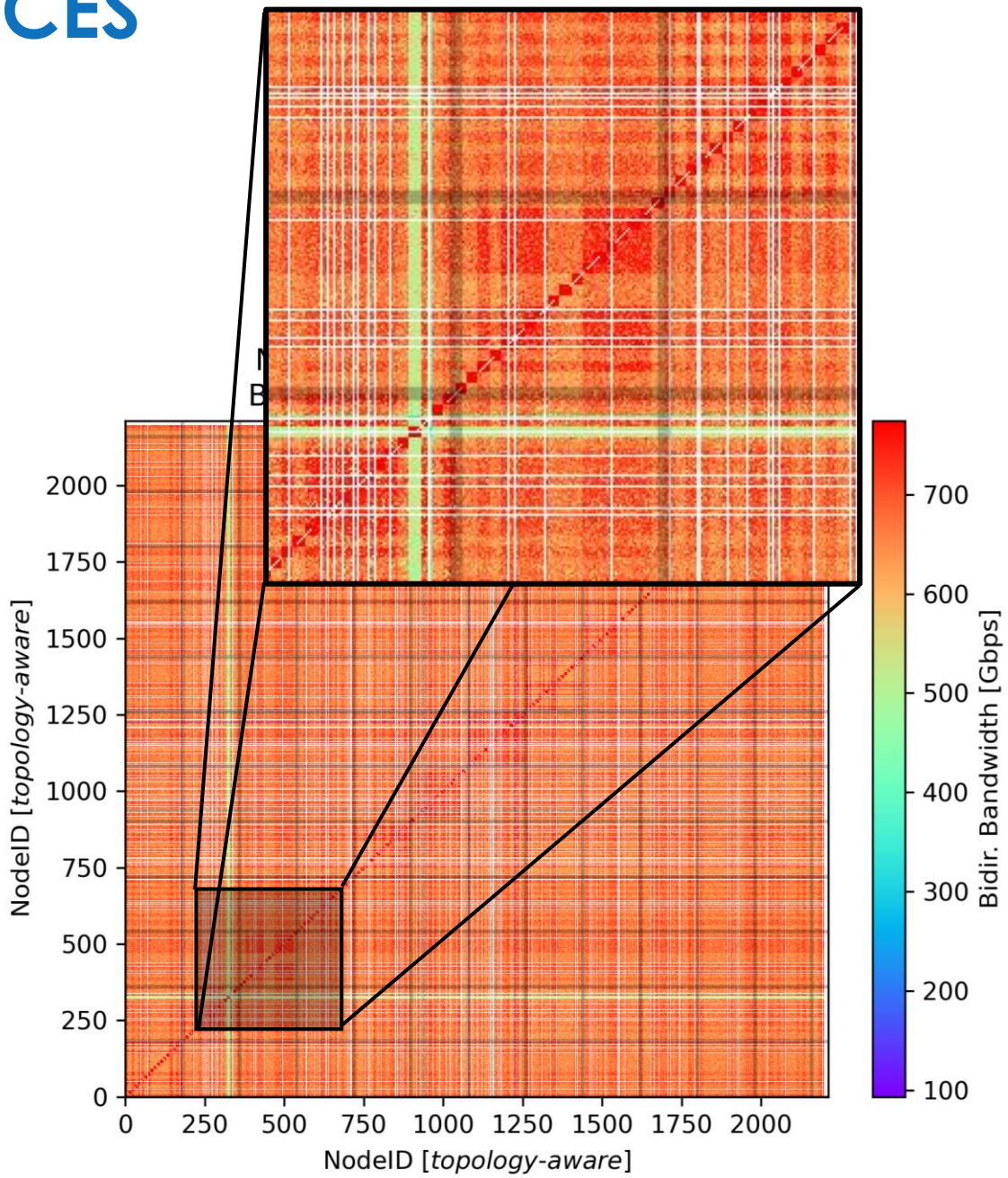


INTERCONNECT PERFORMANCES

Bidirectional Bandwith Measurements

200 NODES





Documentation

Our userguide goes into more depth about all the aspects described during this presentation. In particular, we suggest:

https://docs.hpc.cineca.it/hpc/leonardo.html#leonardo-card
Production environment on LEONARDO

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

https://docs.hpc.cineca.it/hpc/hpc_scheduler.html
Scheduler and job submission

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

Thank you