CINECA

SchedMD

slurm workload manager

SITE REPORT: CINECA EXPERIENCE WITH SLURM

Alessandro Marani
a.marani@cineca.it

12-13 September 2023

WHAT IS CINECA

1969

SUPERCOMPUTING

4 founding Universities
Bologna, Florence, Padua, Venice

’80

MINISTRIES

IT Systems for the Italian Ministry of Universities and Research

‘90

UNIVERSITIES

IT Systems for the Italian Academic System

2000

PA & INDUSTRY

Technologial Transfer
to Healthcare
Public Administration
Industry

2020

HPC PRE-EXASCALE

Artificial Intelligence
Big Data
Quantum Computing

50 YEARS OF SUPERCOMPUTERS

CINECA

TIMELINE OF CINECA'S SUPERCOMPUTERS

1969	1975	1985	1989	1993	1994	1995	1996	1998	1999
CDC 6600	CDC 7600	CRAY X-MP / 48	CRAY Y-MP / 48 IBM 3090-600 VF	CRAY C-90/2128	CRAY T30 64	CRAY T3D 128	CRAY T3D 128	CRAY T3E 256 SGI ORIGIN 16	CRAY T3E 256 SGI ORIGIN 64 IBM SP POWER 3
	1° SC IN ITALY	1° VECTOR SC			1° PARALLEL SYSTEM		1° MPP SYSTEM		
	1 TERAFLOP/S			10 TERAFLOP/S	100 TERAFLOP/S	2 PETAFLOP/S		32 TERAFLOP/S	PRE-EXASCALE

2000	2002	2004	2005	2007	2009	2012	2016	2020	2022
VIRTUAL THEATRE	IBM SP4 612	IBM SP5 512	IBM CLX XEON 1024	IBM BCX AMD 5120	IBM SP6	FERMI	MARCONI	MARCONI100	LEONARDO

2023 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 | 2023

Booster Module:
32 core per node
4 GPU NVIDIA Ampere custom

Data Centric Module:
56 cores per node
SOON IN PRODUCTION

110 PB Storage
250 PFlops

MARCONI | 2017

48 cores per node
612 TB RAM
10 PFlops

DGX | 2021

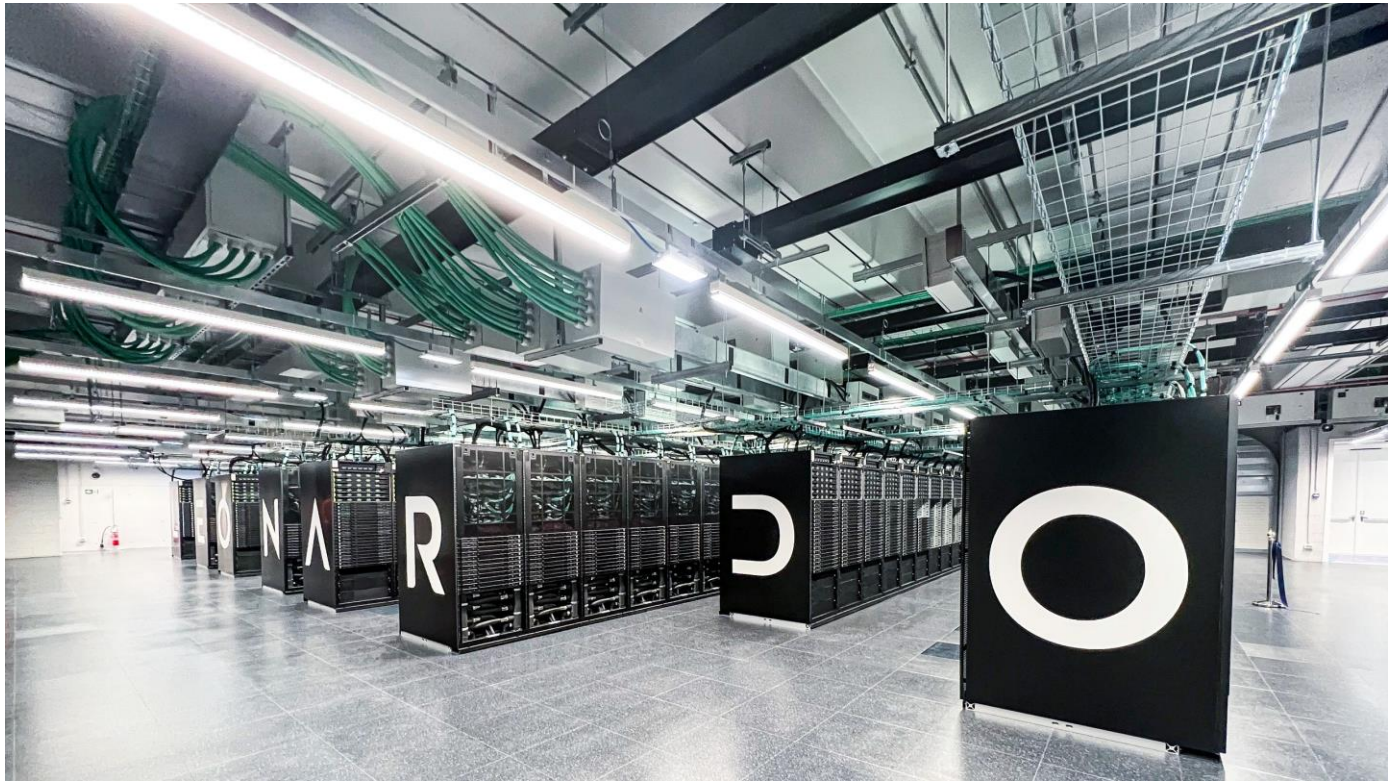
128 cores per node
8 GPU NVIDIA A100 per node
100 TB Storage
15 PFlops

GALILEO100 | 2021

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

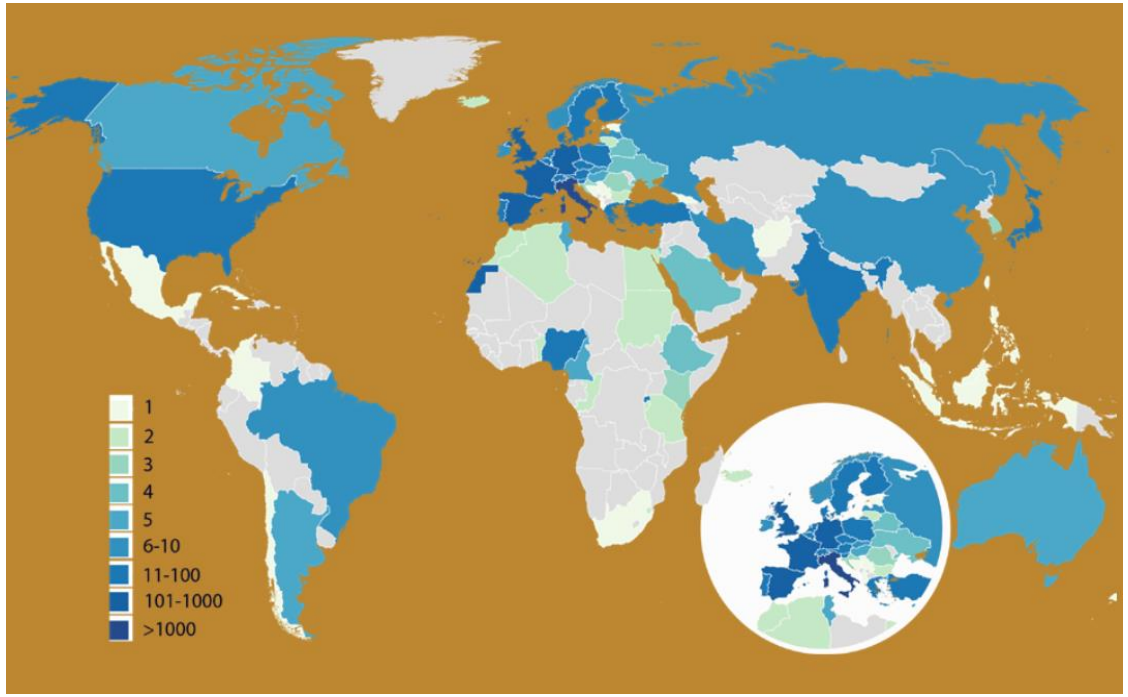
Leonardo System

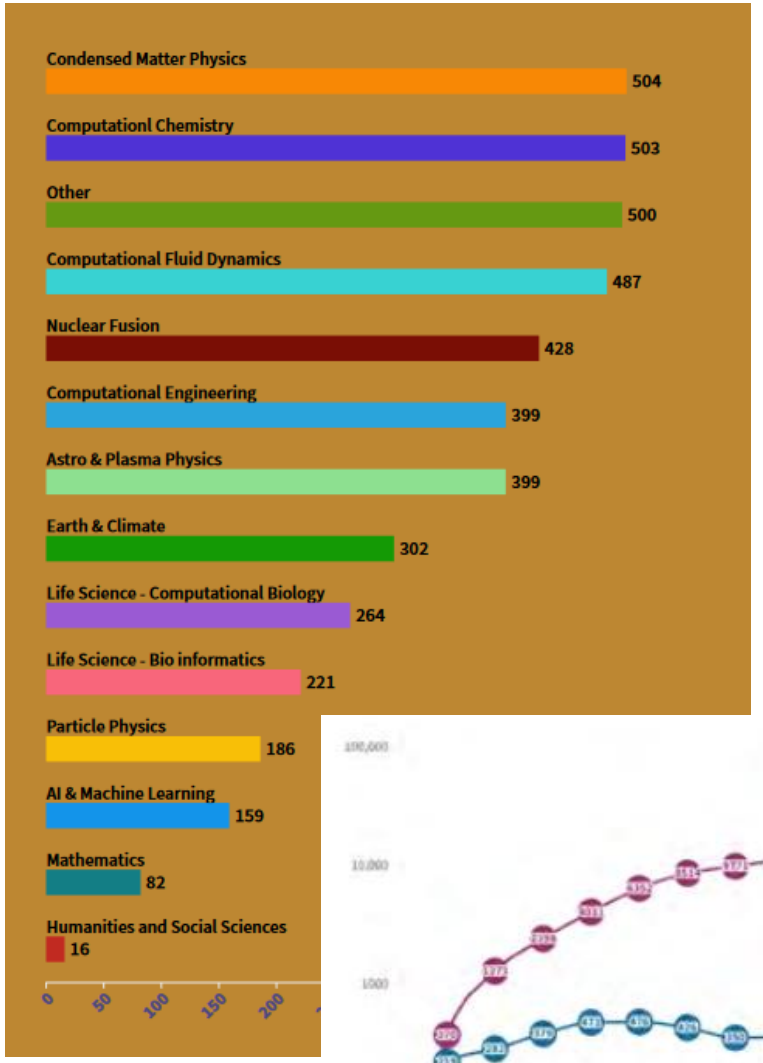
- **4th Top500**
- **HPL 240 PF + 9 PF (currently 170PF)**
- TCO Investment: **240M€** (120M€ Capex + 120M€ Opex)
- **5000 nodes** based on **BullSequana XH2000** platform technology (3500 GPU + 1500 CPU)
- Computing racks: **95% Direct Liquid Cooled**
- Data storage: >**100PB** (NVMe+HDD)
- **Warm water**: Inlet temperature of 37 degrees
- NVIDIA Mellanox **HDR 200** interconnect
  - Dragonfly+ topology



Equipped with Slurm 22.05.7-Atos.1.0
A customized version with Atos in charge of the support

OUR USERS





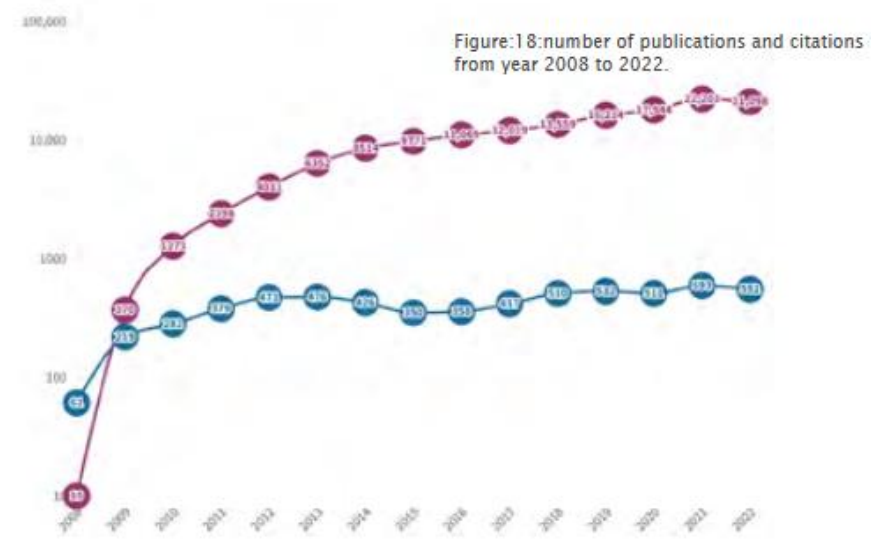


Figure:18:number of publications and citations from year 2008 to 2022.

- **4450** active users at the end of 2022
- **33%** users from outside the italian institutions
- Many important projects at an **European level** (EUROFusion, Cheese, LIGATE, …)

SLURM @ CINECA -1

Since beginning of 2018 – migration from PBSpro to Slurm

WHY?

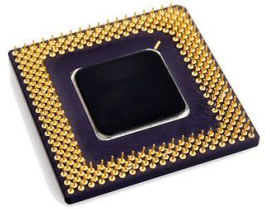
- analysis of schedulers/resource managers proved that SLURM was **already a robust tool** to manage resources and schedule jobs in hybrid architectures (thinking of ongoing trends in HPC)

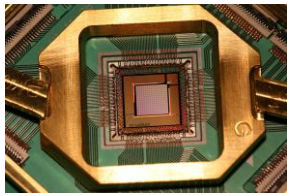
- analysis of CINECA production environment core pillars (managing different communities with specific requests, huge loads of jobs - 1000+ avg in an hour, fair use of resources) proved that SLURM was quite **easily compliant with our needs** (more work needed for linearized used of resources? see later)

IS IT STILL TRUE?

Fortunately, **yes**. Since Marconi100, DGX, and now Leonardo we exploited slurm's dealing of GPUs technologies (NVIDIA MPS, MIG) to fully exploit **GPUs power towards the exascale world** (in Leonardo, just opened to production, still work in progress)

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

GENERAL STRUCTURE

- **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.

Some of these points are better discussed in the following slides...

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.

BUDGET LINEARIZATION

The reasoning: due to the number and the variety of users and project sizes and duration, it is necessary to implement a mechanism that grants a certain degree of fairness with the usage of the HPC clusters.
In the past we had issues with big budget users "monopolizing" the resources and impeding other users to run their simulations.

Every month, we set a quota that is the total amount of CPU hours per budget divided by the total number of months of the project's duration. While you have all the monthly quota to spend, **your jobs will have full priority** and get executed quite fast.

As your quota depletes, **your priority will drop** and, when the monthly quota is fully consumed, it will reach its lowest. Jobs will still be able to enter, but they will have to wait more.

At the beginning of the next month, **all quotas are restored** and all jobs will have full priority again

PRIORITY 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
```

The weights are set so that the parameters have an **order of importance**:
qos, fairshare, age, jobsize.

For example, two jobs in the same partition and with the same qos compete in terms of fairshare, while if the qos are different the one with the highest priority wins regardless of the fairshare and other weights.

ADMINISTERING THE SHARES

Remember **Bug #5212**? (05/25/18)
https://bugs.schedmd.com/show_bug.cgi?id=5212

"So 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."

...

"What we did is to implement a 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")
```

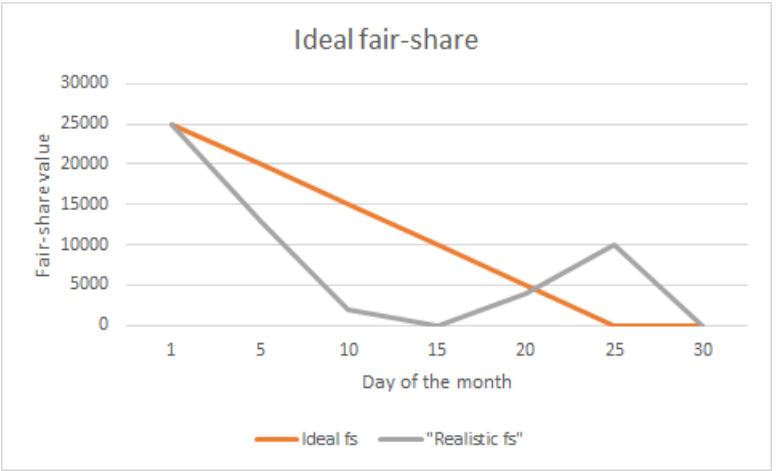
admin_shares.py

WISHING LIST...

Will SLURM_JOB_QOS be available in prolog/epilog in addition to SrunProlog/Epilog?
A common use case: the **Intel vtune sep drivers**
- for security reasons our sys admins require us to monitor and control the users' capability to load the sep drivers
- the QOS comes to our rescue! Users needing the drivers are granted a specific QOS (and registered) to be required in their jobs
- the job's prolog has to detect the request and load the driver, and the epilog has to unload it. But.... the SLURM_JOB_QOS is not available to the prolog
- we recurred to the SLURM_JOB_CONSTRAINT: the user has to request both the specific QOS and the constraint, and a check is performed by the job_submit.lua script on the coherent request of the two parameters

Intel VTune

"Linearized" fair-share: with its formula, fair-share is still an instrument that we can use to simulate a linearized priority, but we can never truly achieve it. What would be best for us is for a way to set the fair-share contribute to priority so that it is maximum when the monthly quota is full, and minimum when the monthly quota is depleted.



THANK YOU!!

Alessandro Marani
a.marani@cineca.it

Massimiliano Guarrasi
m.guarrasi@cineca.it

Isabella Baccarelli
i.baccarelli@cineca.it

CINECA HPC User Support and Production Team
superc@cineca.it