UG2.4: Accounting

In this page:

Our HPC resources can be used on a "pay for use" basis.

Currently, the cost is based on **elapsed time** and the **number of cores reserved (not used!)** by the **batch jobs**. In general, most tools and applications from our Software Catalog can be used free of charge even if the program is burdened with a licence. Only in a few cases, you need to register and pay an additional fee to access special applications. All information is reported in the specific application description: see application-software-science.

To run a batch job, a user must login to an HPC system using his/her username and password. The username must be associated with one or more active projects (Accounts) with available budgets.

Usernames and Accounts

In CINECA, the words "username" and "account" have different meanings.

USERNAME: identifies the individual connecting to the system. It is the string used (along with the password) for getting access to the system (through ssh, for example). It is 8 characters long and can be obtained, from your interactive Unix session, by the command:

```
> echo $USER
```

Login credentials are to be considered as **strictly personal**, meaning that **NO SHARING** between members of the same working group is expected to happen. Every single user entitled with login credentials is to be considered personally responsible for any misuse that should take place.

Please follow the instructions in UG2.1 Getting started to get a username and password to access our systems.

ACCOUNT: indicates the grant or resource allocation which you can use for your batch jobs. Usually, a "budget" is associated with an Account and reports how many resources (computing hours) can be used within that Account. In UG2.2 Become a user, we describe the several ways your username can be associated with an account.

You can list all the Accounts attached to your username on the current cluster, together with the "budget" and the consumed resources, with the command "saldo" (see below).
One single username can use **multiple accounts,** and one single account can be used by **multiple usernames**, all competing for the same budget.

The account that will be charged for the resource usage of a batch job is identified by a **specific line in the job script** used for the job submission. Currently, all of our clusters use the SLURM scheduler. Documentation about **how to write a job script** for SLURM, can be found in "UG2.6.1: How to submit the job - Batch Scheduler SLURM".

Every account has a **beginning** date and an **ending** date, within which the budget must be used. On our HPC platforms, **special constraints** on how to use the budget are defined (see **budget linearization** below).

The elapsed time spent by the users' batch jobs, multiplied by the number of reserved cores (core-h), is used to **decrease the account budget**; once a day, at midnight, an automatic procedure will take into account all jobs completed in the past 24 hours and update your budget accordingly.

Whenever an account **runs out of budget** (in CPU hours), or when its expiring date is met, all the usernames referring to that account won't be able to submit batch jobs anymore using the corresponding project budget (but they will still be able to do so if they are authorized on other projects account).

Nevertheless, users with no active Account will still be able to access the HPC platforms to perform some lightweight post-processing (interactive runs) and /or retrieve their data. Usernames will be kept alive **for a whole year** after their last (most recent) account has been shut down.

The **mapping** between users and Accounts is done by the CINECA staff, who is in charge of creating new projects and associating a PI to each of them. A **PI**, in turn, **can associate** other users to a project as collaborators via the UserDB page related to the project.

The "saldo" command

You can **list all the Accounts** attached to your username on the current cluster, together with the "budget" and the consumed resources, with the command:

```
> saldo -b
```

One single username can use multiple Accounts and one single Account can be used by multiple usernames (possibly on multiple platforms), all competing for the same budget.

On systems like MARCONI, where different independent partitions were available (KNL and SKL) you should specify the host you are interested in:

```
> saldo -b  (by default returns accounts active on SKL)

> saldo -b --skl (returns accounts active on SKL)
```

Another useful command is

```
> saldo -r
```

that prints **daily resource usage report** for selected usernames and/or Accounts on the local cluster.

For more information, run the "saldo" command without any option.

Billing policy

The billing procedures do not consider the time spent in **interactive** work, meaning it is free of charge.

For **batch jobs,** the billing is based on "elapsed time" (execution time, not necessarily wall clock time) and "number of cores reserved" (not used!) by the batch job. When the node is used in a "non-exclusive" mode, the memory request is also taken into account (see below).

The basic idea is:

accounted hours = ElapsedTime x **ReservedCores**

Please note that every cluster usually has a "**serial**" queue, defined on front-end nodes, that allows for serial jobs for a short time limit (maximum 4 hours). On these queues, accounting is not enabled, meaning that you can use them without being charged. Consequently, serial queues are allowed to be also used when an account is **expired** or has **exhausted** all of its budget: it is useful for post-processing or data transfer.

"Non-exclusive" nodes: memory matters!

On some clusters (for example on GALILEO100 or LEONARDO) you can choose to allocate for your job only **part of the node**. You are not forced to allocate all of it as it happens in clusters (like MARCONI) running in exclusive mode. In this case, the accounting procedure also takes into account the **am ount of memory** you request for your job. If you ask for an amount of memory that is larger than the equivalent number of cores requested, the jobs will be billed for a larger number of cores than the ones you have reserved.
The billing always follows the basic idea illustrated above, but a generalized parameter for the number of reserved cores, accounting for the memory request, is now used:

accounted hours = ElapsedTime x **ReservedCoresEquiv**

where

ReservedCoresEquiv = ReservedCores x MemFactor

- **MemFactor = 1.**
  If the memory you ask for (in terms of the equivalent number of cores) is smaller than or equal to the number of reserved cores. In this case, the amount of cpu-hours billed depends only on the number of requested cores.

- **MemFactor = (ReservedMemory / TotalMemory) / (ReservedCores / TotalCores)**
  If the memory you ask for (in terms of the equivalent number of cores) is larger than the number of reserved cores. In this case, the amount of cpu-hours billed also depend on the amount of memory requested (i.e. the actual percentage of node allocated).

For example, on GALILEO100 the TotalMemory considered to calculate the MemFactor is 375300 MB of memory (around 366 GB), and each compute node has 48 cores:

- TotalMemory = 366 GB
- TotalCore = 48

If you ask for only one core and 58 GB of memory (thus allocating for yourself half of the node even if you are using one core), the MemFactor is:

- ReservedMemory = 58 GB
- ReservedCores = 1
- MemFactor = (58 GB / 366 GB) / (1 / 48) = 8

Hence, with such a request for each hour of computation, your budget will be billed for 8 equivalent CPUs, i.e., for 8 hours.

This rule applies to each cluster based on its amount of total memory and cores.

"Non-exclusive" nodes: TMPDIR matters (LEONARDO only)!

On LEONARDO, the job's TMPDIR local area is managed by the slurm job_container/tmpfs plugin and can be explicitly requested on the diskful nodes. In this case, the accounting procedure also takes into account the **amount of space (gres/tmpfs)** you request for your job. If you ask for an amount of local storage that is larger than the equivalent number of cores requested, the jobs will be billed for a larger number of cores than the ones you have reserved.
The billing always follows the basic idea illustrated above, but a generalized parameter for the number of reserved cores, accounting for the local storage request, is now used:

accounted hours = ElapsedTime x **ReservedCoresEquiv**

where

ReservedCoresEquiv = ReservedCores x LocalStorageFactor

- **LocalStorageFactor = 1**
  If the local storage you ask for (in terms of the equivalent number of cores) is smaller than or equal to the number of reserved cores. In this case, the amount of cpu-hours billed depends only on the number of requested cores.

- **LocalStorageFactor = (ReservedLocalStorage / TotalLocalStorage) / (ReservedCores / TotalCores)**
  If the local storage you ask for (in terms of the equivalent number of cores) is larger than the number of reserved cores. In this case, the amount of cpu-hours billed also depend on the amount of local storage requested (i.e. the actual percentage of node allocated).

For example, on LEONARDO-DCGP partition, the TotalLocalStorage considered to calculate the LocalStorageFactor is 3 TB, and each compute node has 112 cores:

- TotalLocalStorage = 3 TB
- TotalCore = 112

If you ask for only one core and 1 TB of memory (thus allocating for yourself one third of the local storage of the node even if you are using one core), the LocalStorageFactor is:

- ReservedLocalStorage = 1 TB
- ReservedCores = 1
- LocalStorageFactor = (1 TB / 3 TB) / (1 / 112) = 37

Hence, with such a request for each hour of computation, your budget will be billed for 37 equivalent CPUs, i.e., for 37 hours.

At present the slurm job_container/tmpfs plugin is ONLY enabled on LEONARDO.

Accounting and accelerators

Recently, the accounting system has been extended to nodes equipped with accelerators. The principle is the same as memory accounting: asking for a number of accelerators that will make you allocate a bigger portion of the node than what is suggested by the simple number of cores requested, will increase the consumption accordingly.

For LEONARDO, every GPU will be treated as 8 cores in terms of accounting. That is because GPU nodes have 32 CPUs and 4 GPU each. So allocating 1 GPU is equivalent to allocating a quarter of the node (i.e. 8 CPUs).

Some examples based on LEONARDO (1 node):

- cpus=24, gpus=1 ==> the number of GPUs requested is equal to having requested 8 CPUs, but since 24 of them have been requested in the standard way, they are not taken into account. Thus 24 CPUs will be billed;
- cpus=6, gpu=1 ==> the number of GPUs requested is equal to having requested 8 CPUs, which is higher than the number of CPUs requested. Thus 8 CPUs will be billed;
- cpus=24, gpus=4 ==> the number of GPUs requested is equal to having requested 32 CPUs, while 24 of them have been requested in the standard way, and they are not enough to cover for the GPU request. Therefore 32 CPUs will be billed;
- cpus=24, gpus=1,mem=500GB ==> the situation is similar to the first example (so 24 CPUs billed), but the memory request is higher than what is guaranteed by the simple allocation of the CPUs or GPUs, since it is equivalent of allocating the entire node. So, 32 CPUs will be billed.

Low priority production jobs for active projects with exhausted budget

Non-expired projects with exhausted budgets may be allowed to keep using the computational resources at the cost of minimal priority. Ask superc@cineca.it to motivate your request and, in case of a positive evaluation, you will be enabled to use the qos_lowprio QOS:

```
#SBATCH --qos=qos_lowprio

#SBATCH -A <account>              # your non expired, exhausted account
```

Budget linearization

A linearization policy for the usage of project budgets is active on all clusters at Cineca. For each account, a **monthly quota** is defined as (total_budget / total_no_of_months). Starting from the first day of each month, the collaborators of any account are allowed to use the quota at full priority. As long as the budget is consumed, the jobs submitted from the account will gradually lose priority, until the monthly budget is fully consumed. At that moment, their jobs will still be considered for execution (so it is possible to consume more than the monthly quota), but with a lower priority than the jobs from accounts that still have some quota left.

The linearization effect on the priority is **fine graduated**, as the linearization parameter depends on the percentage of the monthly quota consumed. The job sorting formula also depends on other aspects, like walltime, resources requested or time spent waiting in queue, so a low priority job can still have some chance of being executed in a quick amount of time if well-tuned (but not as quick as jobs with the same tuning but advantaged in terms of linearization priority). You can check the usage of your monthly quota with the "saldo -b" command: the last two columns are about the quota defined for your account and the monthly consumption.

This policy is similar to those already applied by other important HPC centers in Europe and worldwide. The goal is to **improve the response time**, giving users the opportunity of using the cpu hours assigned to their project in relation to their actual size (total amount of core-hours). Please note that applying a sort of "linearization" of your project budget is recommended. . Each month a given percentage of your budget is guaranteed, but **non-linear usage is discouraged** for the welfare of all the users that are simultaneously hosted by our HPC systems.