Table of Contents

Utilization and accounting

When you submit jobs, they are using physical resources such as CPUs, Memory, Network, GPUs, Energy etc. We keep track of the usage of some of those resource. On this page we'll let you know how to consult your usage of the resource. We have several tools that you can use to consult your utilization: sacct, sreport, openxdmod

Comparison of sreport, sacct, and sshare

We use **sreport** as our primary accounting reference. However, you may find other tools useful for specific purposes. Here's a comparison:

- **sacct**: Displays only account jobs, excluding time allocated via reservations. If duplicate jobs exist, only one is shown.
- **sreport**: By default, jobs with wall times overlapping the report's time range are truncated. For reservation-based jobs, the requested idle time is distributed among all users with access to the reservation.
- **sshare**: Not recommended for accounting purposes; displayed values are adjusted based on fairshare calculations.

Resource accounting uniformization

We charge usage uniformly by converting GPU hours and memory usage into CPU hour equivalents, leveraging the TRESBillingWeights functionality provided by SLURM.

A CPU hour represents one hour of processing time by a single CPU core.

For GPUs, SLURM assigns a conversion factor to each GPU model through TRESBillingWeights (see below the conversion table), reflecting its computational performance relative to a CPU. Similarly, memory usage is also converted into CPU hour equivalents based on predefined weights, ensuring that jobs consuming significant memory resources are accounted for fairly.

For example, a job using a GPU with a weight of 10 for 2 hours and memory equivalent to 5 CPU hours would be billed as 25 CPU hours. This approach ensures consistent, transparent, and fair resource accounting across all heterogeneous components of the cluster.

You can see the detail of the conversion by looking at the parameter of a random partition on any of the clusters. We are using the same conversion table everywhere.

```
(bamboo)-[root@slurm1 ~]$ scontrol show partition debug-cpu | grep
TRESBillingWeights | tr "," "\n"
   TRESBillingWeights=CPU=1.0
Mem=0.25G
GRES/gpu=1
GRES/gpu:nvidia_a100-pcie-40gb=5
GRES/gpu:nvidia_a100_80gb_pcie=8
GRES/gpu:nvidia_geforce_rtx_2080_ti=2
GRES/gpu:nvidia_geforce_rtx_3080=3
GRES/gpu:nvidia_geforce_rtx_3090=5
GRES/gpu:nvidia_geforce_rtx_4090=8
GRES/gpu:nvidia_rtx_a5000=5
GRES/gpu:nvidia_rtx_a5500=5
GRES/gpu:nvidia_rtx_a6000=8
GRES/gpu:nvidia_titan_x=1
GRES/gpu:tesla_p100-pcie-12gb=1
```

Here you can see for example that using a gpu nvidia_a100-pcie-40gb for 1 hour is equivalent in term of cost to use 5 CPUhour.

Resources available for research group

Research groups that have invested in the HPC cluster by purchasing private CPU or GPU nodes benefit from high priority access to these resources.

While these nodes remain available to all users, owners receive priority scheduling and a designated number of included compute hours per year.

To check the details of their owned resources, users can run the script `ug_getNodeCharacteristicsSummary.py`, which provides a summary of the node characteristics within the cluster.

Example:

```
ug_getNodeCharacteristicsSummary.py --partitions private-<group>-gpu
private-<group>-cpu --cluster <cluster> --summary
host    sn             cpu    mem    gpunumber    gpudeleted  gpumodel
gpumemory  purchasedate      months remaining in prod. (Jan 2025)    billing
------  -----------  -----  -----  -----------  ------------  --------------
------------  -----------  --------------  ----------------------------------
-----  ---------
```

```
cpu084  N-20.02.151     36    187            0             0
0  2020-02-01                                           1          79
cpu085  N-20.02.152     36    187            0             0
0  2020-02-01                                           1          79
cpu086  N-20.02.153     36    187            0             0
0  2020-02-01                                           1          79
cpu087  N-20.02.154     36    187            0             0
0  2020-02-01                                           1          79
cpu088  N-20.02.155     36    187            0             0
0  2020-02-01                                           1          79
cpu089  N-20.02.156     36    187            0             0
0  2020-02-01                                           1          79
cpu090  N-20.02.157     36    187            0             0
0  2020-02-01                                           1          79
cpu209  N-17.12.104     20     94            0             0
0  2017-12-01                                           0          41
cpu210  N-17.12.105     20     94            0             0
0  2017-12-01                                           0          41
cpu211  N-17.12.106     20     94            0             0
0  2017-12-01                                           0          41
cpu212  N-17.12.107     20     94            0             0
0  2017-12-01                                           0          41
cpu213  N-17.12.108     20     94            0             0
0  2017-12-01                                           0          41
cpu226  N-19.01.161     20     94            0             0
0  2019-01-01                                           0          41
cpu227  N-19.01.162     20     94            0             0
0  2019-01-01                                           0          41
cpu228  N-19.01.163     20     94            0             0
0  2019-01-01                                           0          41
cpu229  N-19.01.164     20     94            0             0
0  2019-01-01                                           0          41
cpu277  N-20.11.131    128    503            0             0
0  2020-11-01                                          10         251
gpu002  S-16.12.215     12    251            5             0  NVIDIA TITAN X
(Pascal)           12288  2016-12-01
0       84
gpu012  S-16.12.216     24    251            8             0  NVIDIA GeForce
RTX 2080 Ti        11264  2016-12-01
0      108
gpu017  S-20.11.146    128    503            8             0  NVIDIA GeForce
RTX 3090           24576  2020-11-01
10       299
gpu023  S-21.09.121    128    503            8             0  NVIDIA GeForce
RTX 3080           10240  2021-09-01
20       283
gpu024  S-21.09.122    128    503            8             0  NVIDIA GeForce
RTX 3080           10240  2021-09-01
20       283
gpu044  S-23.01.148    128    503            8             0  NVIDIA RTX
A5000                  24564  2023-01-01
```

```
36        299
gpu047  S-23.12.113    128    503           8             0  NVIDIA RTX
A5000                  24564  2023-12-01
47        299
gpu049  S-24.10.140    128    384           8             0  NVIDIA GeForce
RTX 4090           24564  2024-10-01
57        291

============================================================ Summary
============================================================
Total CPUs: 1364 Total CPUs memory[GB]: 6059 Total GPUs: 61 Total GPUs
memory[MB]: 142300 Billing: 1959 CPUhours per year: 10.30M
```

How to read the output:

- **host**: the hostname of the compute node
- **sn**: the serial number of the node
- **cpu**: the number of CPUs available in the node
- **mem**: the quantity of memory on the node in GB
- **gpunumber**: the number of GPU cards on the node
- **gpudeleted**: the number of GPU cards out of order
- **gpumodel**: the GPU model
- **gpumemory**: the GPU memory in MB per GPU card
- **purchasedate**: the purchase date of the node
- **months remaining in prod. (Jan 2025)**: the number of months the node remains the property of the research group, the reference date is indicated in parenthesis. In this example it is January 2025.
- **billing**: the billing value of the compute node

You can modify the reference year if you want to “simulate” the hardware you'll have in your private partition in a given year. To do so, use the argument `--reference-year` of the script.

Job accounting

OpenXDMoD

We track the job usage of our clusters here: https://openxdmod.hpc.unige.ch/

We have a tutorial explaining some of the features: here

Openxdmod is integrated into our SI. When you connect to it, you'll get the profile “user” and the data are filtered by your user by default. If you are a PI, you can ask us to change your profile to be PI.

OpenXDMoD doesn't support the “billing” metrics (yet?) but only CPUh and GPUh. For this reason, you need to use the sreport or our script.

sacct

You can see your job history using `sacct`:

```
[sagon@master ~] $ sacct -u $USER -S 2021-04-01
       JobID    JobName  Partition    Account  AllocCPUS      State ExitCode
------------ ---------- ---------- ---------- ---------- ---------- --------
45517641        jobname  debug-cpu   rossigno          1     FAILED      2:0
45517641.ba+      batch              rossigno          1     FAILED      2:0
45517641.ex+     extern              rossigno          1  COMPLETED      0:0
45517641.0            R              rossigno          1     FAILED      2:0
45518119        jobname  debug-cpu   rossigno          1  COMPLETED      0:0
45518119.ba+      batch              rossigno          1  COMPLETED      0:0
45518119.ex+     extern              rossigno          1  COMPLETED      0:0
```

Report and statistics with sreport

To get reporting about your past jobs, you can use `sreport` (https://slurm.schedmd.com/sreport.html).

We wrote a helper that you can use to get your past resource usage on the cluster. This script can display the resource utilization

- for each user of a given account (PI)
- total usage of a given account (PI)

```
(baobab)-[sagon@login1 ~]$ ug_slurm_usage_per_user.py -h
usage: ug_slurm_usage_per_user.py [-h] [--user USER] [--start START] [--end
END] [--pi PI] [--group GROUP] [--cluster {baobab,yggdrasil,bamboo}] [--
all_users] [--report_type {user,account}] [--time_format
{Hours,Minutes,Seconds}]
                                  [--verbose]

Retrieve HPC utilization statistics for a user or group of users.

options:
  -h, --help            show this help message and exit
  --user USER           Username to retrieve usage for.
  --start START         Start date (default: first of month).
  --end END             End date (default: now).
  --pi PI               Specify a PI manually.
  --group GROUP         Specify a group name to get all PIs belonging to it.
  --cluster {baobab,yggdrasil,bamboo}
                        Cluster name (default: all clusters).
  --all_users           Include all users under the PI account.
  --report_type {user,account}
                        Type of report: user (default) or account.
  --time_format {Hours,Minutes,Seconds}
                        Time format: Hours (default), Minutes, or Seconds.
```

```
  --verbose             Verbose output.
```

By default when you run this script, it will print your past usage of the current month, for all the accounts you are member of.

Usage example to see the resource usage from the beginning of 2025 for all the PIs and associate users of the group private_xxx. The group private_xxx owns several compute nodes:

```
(baobab)-[sagon@login1 ~]$ ug_slurm_usage_per_user.py --group private_xxx --
start=2025-01-01 --report_type=account
--------------------------------------------------------------------------------
----

Cluster/Account/User Utilization 2025-01-01T00:00:00 - 2025-08-21T14:59:59
(20095200 secs)

Usage reported in TRES Hours

--------------------------------------------------------------------------------
----

Cluster    Login    Proper Name    Account    TRES Name       Used
---------  -------  -------------  ---------  -----------  -------
baobab                             PI1        billing        56134
yggdrasil                          PI1        billing       105817
bamboo                             PI2        billing         5416
baobab                             PI2        billing      1517001
yggdrasil                          PI2        billing        23775
bamboo                             PI3        billing            0
baobab                             PI3        billing      1687963
yggdrasil                          PI3        billing      1344599
[...]
Total usage: 7.36M
```

sreport examples

Here are some examples that can give you a starting point :

To get the number of jobs you ran (you ⇔ $USER) in 2018 (dates in yyyy-mm-dd format) :

```
[brero@login2 ~]$ sreport job sizesbyaccount user=$USER PrintJobCount
start=2018-01-01 end=2019-01-01

--------------------------------------------------------------------------------
----
Job Sizes 2018-01-01T00:00:00 - 2018-12-31T23:59:59 (31536000 secs)
Units are in number of jobs ran
--------------------------------------------------------------------------------
----
  Cluster   Account     0-49 CPUs   50-249 CPUs  250-499 CPUs  500-999 CPUs
```

```
>= 1000 CPUs % of cluster
--------- --------- ------------ ------------ ------------ ------------
------------ -----------
   baobab      root           180            40             4            15
0      100.00%
```

You can see how many jobs were run (grouped by allocated CPU). You can also see we specified an extra day for the *end date* `end=2019-01-01` in order to cover the whole year :

```
Job Sizes 2018-01-01T00:00:00 - 2018-12-31T23:59:59''
```

You can also check how much CPU time (seconds) you have used on the cluster between since 2019-09-01 :

```
[brero@login2 ~]$ sreport cluster AccountUtilizationByUser user=$USER
start=2019-09-01 -t Seconds
--------------------------------------------------------------------------
----
Cluster/Account/User Utilization 2019-09-01T00:00:00 - 2019-09-09T23:59:59
(64800 secs)
Usage reported in CPU Seconds
--------------------------------------------------------------------------
----
  Cluster         Account     Login     Proper Name     Used   Energy
--------- -------------- --------- -------------- -------- --------
   baobab        rossigno     brero   BRERO Massimo     1159        0
```

In this example, we added the time `-t Seconds` parameter to have the output in seconds. *Minutes* or *Hours* are also possible.

Please note :

- By default, the CPU time is in Minutes
- It takes up to an hour for Slurm to upate this information in its database, so be patient
- If you don't specify a start, nor an end date, yesterday's date will be used.
- The CPU time is the time that was allocated to you. It doesn't matter if the CPU was actually used or not. So let's say you ask for 15min allocation, then do nothing for 3 minutes then run 1 CPU at 100% for 4 minutes and exit the allocation, then 7 minutes will be added to your CPU time.

Tip : If you absolutely need a report including your job that ran on the same day, you can override the default end date by forcing tomorrow's date :

```
sreport cluster AccountUtilizationByUser user=$USER start=2019-09-01
end=$(date --date="tomorrow" +%Y-%m-%d) -t seconds
```

From:
https://doc.eresearch.unige.ch/ - **eResearch Doc**

Permanent link:
https://doc.eresearch.unige.ch/hpc/accounting?rev=1764840784

Last update: **2025/12/04 09:33**