Table of Contents

Introduction

This page gives best practices and tips on how to use the clusters **Baobab** and **Yggdrasil**.

An HPC cluster is an advanced, complex and always-evolving piece of technology. It's easy to forget details and make mistakes when using one, so don't hesitate to check this section every now and then, yes, even if you are the local HPC guru in your team! There's always something new to learn !

First steps

You are now ready to start using the HPC clusters. But how do you start ?

For your first steps we recommend the following :

- Check the Rules and etiquette.
- Connect to the login node of the cluster you are planning to use : Access : SSH, X2GO
- Check the rest of this page for best practices and smart use of the HPC resources.
  - This page contains important information! You can hog resources and/or wait much longer than you could if you don't request the right amount of time for your job, if you ask too much (or not enough) resources, if you are not using the right partition, etc.
- Understand how to load your libraries/application with `module`: Applications and libraries
- Learn how to write a Slurm `sbatch` script: Slurm and job management

Rules and etiquette

As a user, you are sharing a fabulous technological beast with hundreds of other researchers. Any research is important (and *yes* we realise *your* research is more important!), nevertheless we expect all of you to respect following rules and etiquette.

- Have basic knowledge of the Linux command line (without GUI) and Bash scripting.
- **Do not use the login node to execute any code**
  - If you do, you are disturbing all other users and this is unacceptable. When this happens we will most likely kill your process and send you an email to make you feel bad.
  - The login node should only be used to compile your code and submit a Slurm job. You must even use Slurm to run your tests. The `debug-cpu` and `debug-gpu` partitions are dedicated for small tests.
- Make a fair use of the resources and **do not waste resources** for others.
  - When people waste resources, it increases the waiting time for everyone.
  - The best way to avoid that is by making sure you have a **reasonable estimation** for the resources you need in your sbatch (CPU, memory, duration, etc.) and that you are using the right partition for the job.
- Read more in the section Stop wasting resources!
  - It's fine to ask your colleagues to give you pointers, BUT adapting a script they wrote 5 years ago might not be good for your project.
- Loading applications and libraries should always be done using `module`
  - Pro tip: you can even force the version to have consistent results or to survive an OS migration, for example.
  - When an application is not available through `module`, you can compile binaries in your `$HOME` directory and use them on any node of the cluster (since your `$HOME` is accessible from any node). Make sure you load the compiler with `module`.
  - You can request the HPC team to install a new software or version of a library in order to load it through `module`. Check this page
- You **cannot** install new software with `yum` (and don't bother with `apt-get`, we are not running Ubuntu).
- **You are not sudo** and never will be.
  - N.B. this is only available for the sacred HPC engineers (Linux-type beard and hoody mandatory)
- Think green : wasting compute resources is **not environmental friendly** as a wasted resource still uses electricity to run and AC to cool down our data centre.

Think green

By design, an HPC cluster is not the most environmental friendly thing in the world : dozens or sometimes hundreds of servers (compute nodes) are running 24/7. But, it doesn't mean there's nothing you can do about it!

If you are not familiar with IT infrastructures, this can be simplified to this :

- Computer are powered with **electricity** and produce **heat**. A powerful computer usually produces more heat.

- **Baobab** has 250+ nodes and **Yggdrasil** has 130+ nodes
- The heat produced by the nodes needs to be cooled down with Air conditioning (AC) to avoid hardware failure.

The power consumption of Baobab and Yggdrasil is more of less **80kW each** (that doesn't include the AC).

The best we can do about it is to make the most of the clusters by using all the resources available and not wasting them.

The same goes for the storage. As you already know you shouldn't store any personal files on the clusters.

But even with scientific data, if you are storing thousands of files and using hundreds of GB that you don't really need, at some point we will have to buy more storage. The storage servers are no different than compute nodes : they also need electricity to run and AC to be cooled down. So deleting useless files from time to time is a good habit.

Besides the quantity of data, remember it is important *where* you store your data. For instance, we back up the content of your `$HOME` every day. Once again, backing up large quantities of data relies on a high speed network, backup tapes, a robot to read/load the tapes, etc.

We still want you to store your files and use the clusters! But we would appreciate if you could keep this in mind next time you wonder if you should keep this 500GB failed output from 3 years old job.

Stop wasting resources!

Based on the post Good usage of resources on HPC cluster , this section gives you information on how to avoid wasting resources on the clusters.

On a HPC cluster such as Baobab or Yggdrasil, the resources are shared between all the users. For this reason it is important to use them wisely. An allocated resource is unavailable for others during the complete lifespan of the job. The issue arise when the resource is allocated but not used.

N.B. in the HPC world, when we talk about CPU, we usually mean core. In Baobab a standard compute node has 2 physical CPUs and 20 cores. So we say that the node has 20 CPUs.

Which resources ?

The resources available on our clusters are:

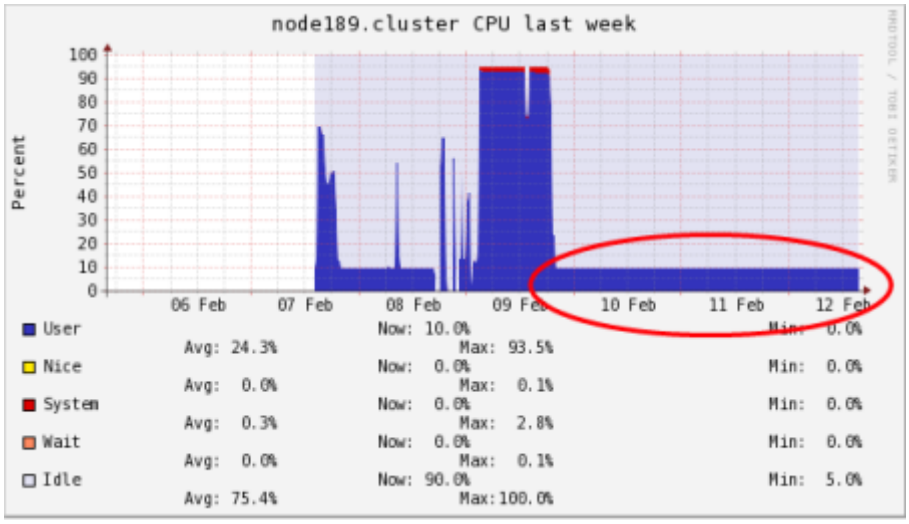
- CPUs, which are grouped in partitions
- GPGPUs which are accelerator for software that support them
- memory (RAM) per core or per node, 3GB by default
- disk space
- time, duration of the computation

Single thread vs multi thread vs distributed jobs

See here to be sure you specify the correct configuration for your job type

Bad CPU usage

Let's take an example of a **single threaded job**. You should clearly use a partition which allows to request a single CPU, such as `public-cpu` or `shared-cpu` and ask one CPU only. If you ask too much CPUs, the resources will be reserved for your job but only one CPU will be used. See the screenshot below of such a bad case where 90% of the compute node is idle.

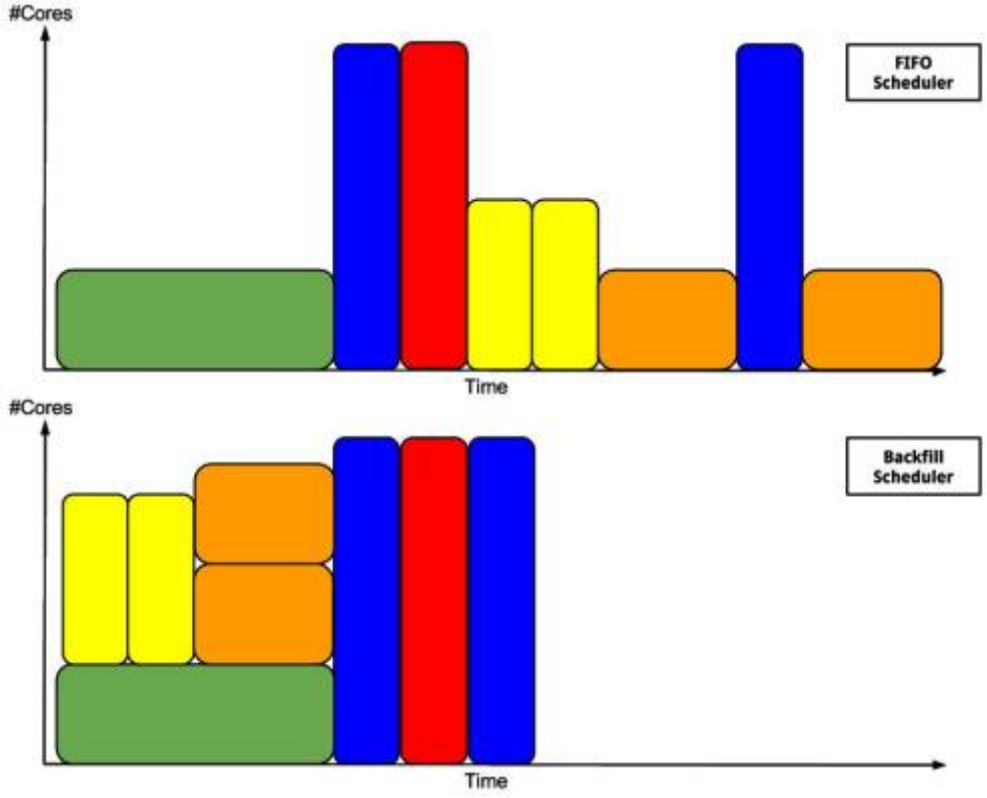


Bad memory usage

Another misuse of HPC resources, is to request too much RAM without using it. Baobab has many older compute nodes with 48GB. If you request for example 40GB of RAM and 1 CPU, only two other CPU of this compute node may be allocated for another job, the remaining will stay unused by lack of memory. This is ok as long as you really use the memory. But in this case, if you job can benefit of more CPU, feel free to request all the other cores of this compute node. You can check the memory usage during the job execution or after.

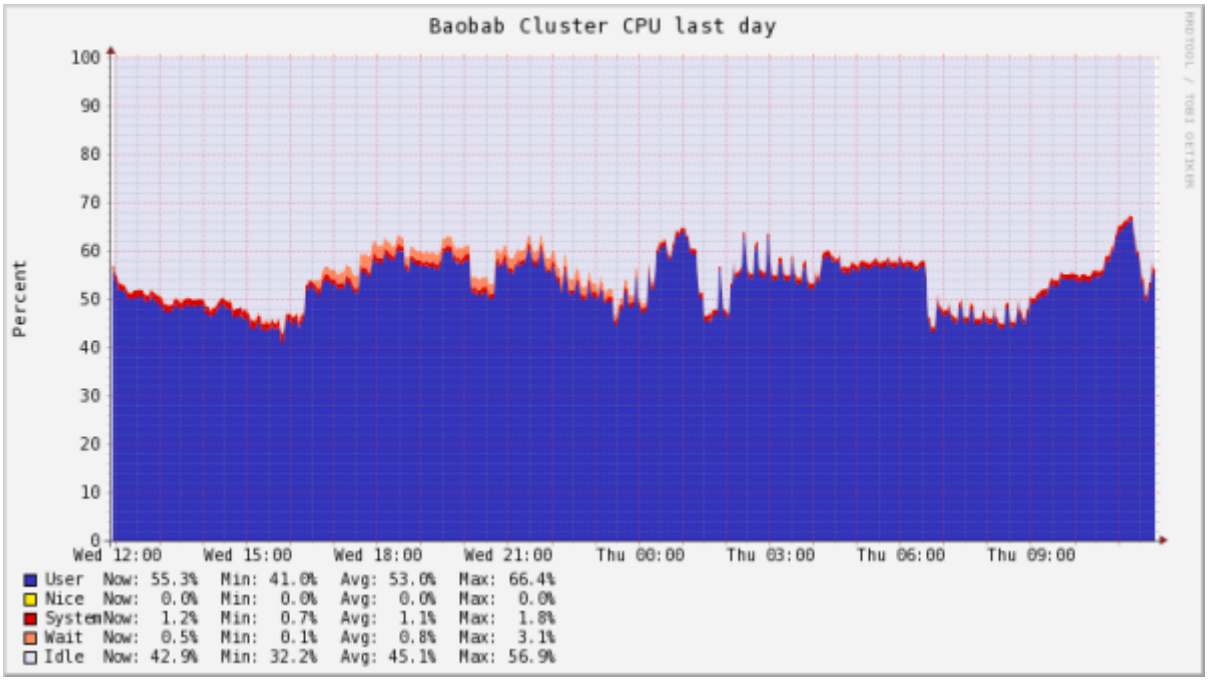
Bad time estimation

If you submit a job and request 4 days of compute time and your job run effectively for 10 minutes, your time estimation is bad. It is important for the scheduler (Slurm) that your time estimation is correct. A couple of hours of overestimation is ok. It is also important to have more or less correct time estimation due to the fact that slurm doesn't only put new job in a fifo queue, but has a backfill mechanism that allows to launch a job that is later in the queue sooner if it doesn’t disturb other job. See the picture below [1)] for an explanation.



Conclusion

You can see on the screenshot below the overall usage of the CPU resource during a whole day. The goal is to increase this usage to around 80% which is considered the maximum on a general purpose cluster. So, help us to reach this goal and, use the resource wisely!



How to write a good Slurm sbatch script

You need to use the cluster(s) and after reading the rest of the documentation, you are now convinced about making the most of the resources whithout wasting them!

So, what is the best way to do that ?

First of all, a “good” sbatch script doesn't need to be complicated. It needs to reflect your needs and your job's needs.

You should find most answers on the Slurm page and in the official Slurm documentation.

In any case, those are the questions you should ask yourself :

- What type of job am I running (single threaded, multi threaded or distributed) ?
  - This will help you choose the parameters `--ntasks` and `--cpus-per-task`
- What partition should I run my job on ?
  - This will help you choose the parameters `--partition`
- How much memory does my job need ?
  - This will help you choose the parameters `--mem`, or `--mem-per-cpu`
  - If you need a large amount of RAM, you might also need to choose a `*-bigmem` partition.
- For how long does my job needs to run ?
  - This will help you choose the parameters `--time`
- Do I want to receive email notification ?
- This is optional, but you can specify the level of details you want with the `--mail-type` parameter

Transfer data from cluster to another with

Rsync

This help assumes you want transfer the directory `$HOME/my_projects/the_best_project_ever` from baobab to yggdrasil at the same path. You can adapt your case by changing the variables.

Rsync options:

- `-a, --archive`This is equivalent to `-rlptgoD`. It is a quick way of saying you want recursion and want to preserve almost everything (with `-H` being a notable omission). The only exception to the above equivalence is when `--files-from` is specified, in which case -r is not implied.
- `-i` turns on the itemized format, which shows more information than the default format
- `-b` makes rsync backup files that exist in both folders, appending ~ to the old file. You can control this suffix with `--suffix .suf`
- `-u` makes rsync transfer skip files which are newer in dest than in src
- `-z` turns on compression, which is useful when transferring easily-compressible files over slow links
- `-P` turns on –partial and –progress

- `--partial` makes rsync keep partially transferred files if the transfer is interrupted
- `--progress` shows a progress bar for each transfer, useful if you transfer big files
- `-n, --dry-run` perform a trial run with no changes made

1) Go to your directory containing `the_best_project_ever`:

```
(baobab)-[toto@login2 ~]$cd $HOME/my_projects/
```

2) Set the variables (or not)

```
(baobab)-[toto@login2 my_projects]$ DST=$HOME/my_projects/
(baobab)-[toto@login2 my_projects]$ DIR=the_best_project_ever
(baobab)-[toto@login2 my_projects]$ YGGDRASIL=login1.yggdrasil
```

3) Run the rsync

```
(baobab)-[toto@login2 my_projects]$ rsync -aviuzPrg ${DIR}
${YGGDRASIL}:${DST}
```

SWITCHfilesender from the cluster

Switch Filesender Filesender is a service provided by SWITCH for transferring files via http. Normally, files are downloaded via a web browser, but using the CLI is more efficient and we benefit from the Login node's high bandwidth.

In order to avoid having to transfer the files to your local computer it is possible to use the Filesender command line tools as explained below

Configuring the CLI tools

Connect to https://filesender.switch.ch then go to the profile tab

Then click on “Create API secret” to generate a code that will be used to allow you to authenticate

This will generate a long string like

`ab56bf28434d1fba1d5f6g3aaf8776e55fd722df205197`

Then connect to Curnagl and run the following commands to download the CLI tool and the configuration

```
cd

mkdir ~/.filesender

wget https://filesender.switch.ch/clidownload.php -O filesender.py

wget https://filesender.switch.ch/clidownload.php?config=1 -O
~/.filesender/filesender.py.ini
```

You will then need to edit the ~/.filesender/filesender.py.ini file using your preferred tool

You need to enter your username as show in the Filesender profile and the API key that you generated

Note that at present, unlike the other Switch services this is not your EduID account!

```
[system]
base_url = https://filesender.switch.ch/filesender2/rest.php
default_transfer_days_valid = 20

[user]
username = Charizard@unige.ch
apikey = ab56bf28434d1fba1d5f6g3aaf8776e55fd722df205197
```

Transferring files

Now that we have done this we can transfer files - note that the modules must be loaded in order to have a python with the required libraries.

```
(baobab)-[alberta@login2 ~]$ module load gcc python
```

You may need to install some python package before:

```
(baobab)-[alberta@login2 ~]$ pip3 install requests urllib3
```

```
(baobab)-[alberta@login2 ~]$ python3 filesender.py -p -r mewtwo@unige.ch
master.tar.gz
Uploading: /home/users/a/alberta/master.tar.gz 0-18408 0%
Uploading: /home/users/a/alberta/master.tar.gz 18408 100%
```

A mail will be sent to Mewtwo@unige.ch who can then download the file

1)

Source: Scheduling. Slurm Training15. Salvador Martin & Jordi Blasco (HPCNow!).