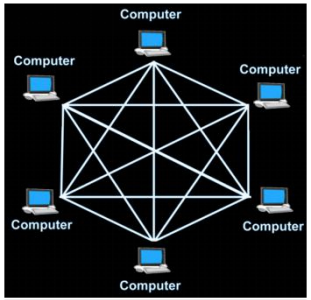
Parallel Machines:
A brief HOW-TO

Luca Tornatore, I.N.A.F.

2025 INAF Course on HPC

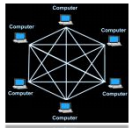
September, 22nd - 26th, OACT, Catania

Outline



Discovering the machine

Submitting jobs



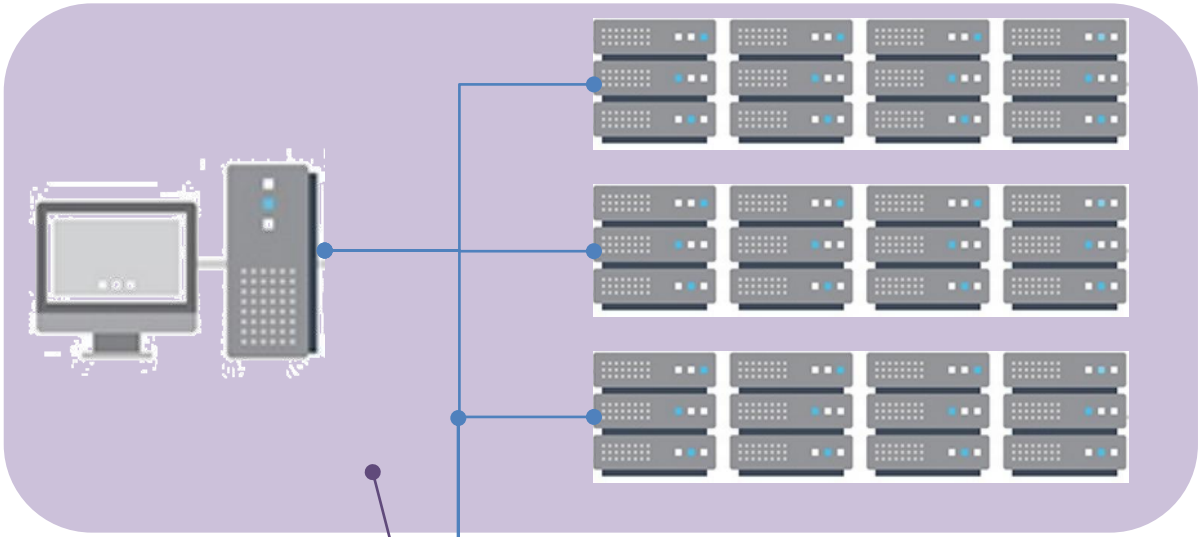
Welcome to your HPC machine

As soon as you login to the machine, you land on a ***login node*** (usually there are many of them).
Those are nodes devoted to interface you to the machine and are distinguished from the ***computational nodes***.

The login nodes are meant to allow the users performing all the managing activities:
- upload / download files
- organize your folders
- build archives
- *small (!!!)* post-processing tasks

and, of course, to access the computational nodes:
- submitting jobs
- inspecting job status

Machine's topology



$HOME
small, solid filesystem

PRODUCTION AREA
large, very fast, parallel filesystem

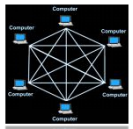
LONG-TERM STORAGE
very large, robust, redundant filesystem

login nodes

middle-ware
resource-manager, job scheduler

partitions (in slurm)
or
queues (in PBS)

storage

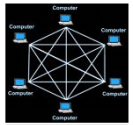


Machine's topology

« An HPC cluster is made up of a number of compute nodes, each with a complement of processors, memory and GPUs. The user submits jobs that specify the application(s) they want to run along with a description of the computing resources needed to run the application(s).

The processing units on nodes are the cores. With the advent of Simultaneous Multithreading (SMT) architectures, single cores can have multiple hardware threads (sometimes known as hyper-threads). The processing elements are generically called a CPU. For systems without SMT, a CPU is a core. For systems with SMT available and enabled, a CPU is a hardware thread. »

from https://hpc.llnl.gov/banks-jobs/running-jobs/slurm-user-manual

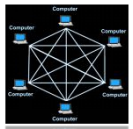


Resources manager & job scheduler

« The batch scheduler and resource manager work together to run jobs on an HPC cluster. The batch scheduler, sometimes called a workload manager, is responsible for finding and allocating the resources that fulfill the job's request at the soonest available time. When a job is scheduled to run, the scheduler instructs the resource manager to launch the application(s) across the job's allocated resources.
This is also known as "**running the job**".

The user can specify conditions for scheduling the job. One condition is the completion (successful or unsuccessful) of an earlier submitted job. Other conditions include the availability of a specific license or access to a specific file system. »

from https://hpc.llnl.gov/banks-jobs/running-jobs/slurm-user-manual



Resources manager & job scheduler

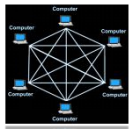
There is a large variety of resource managers & job scheduler.
However, by far the two most used and common are

- SLURM :: https://slurm.schedmd.com/overview.html

- PBS :: https://www.openpbs.org/

Both of them play both roles, serving both as resource managers and job schedulers.

We'll cover he most basic slurm usage, since it is the most commonly found on HPC platforms and it is that running on the cluster that we'll be using in the next days

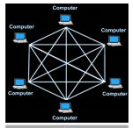


1 / what partitions are present

Normally, with your account comes a bunch of specifications of the machine on which you are accounted - read them carefully.

However, as first, you may want to determine **what partitions exist on the system.**
This information is provided by the command `sinfo`.

- the * indicates what partition is the default
- the column **`avail`** tells you whether the partition is up or not
- the column **`timelimit`** informs about the default time-limit for a job
- the column **`nodes`** indicates how many nodes belong to the partition
- the **`state`** column carries information about the state of the nodes it refers to
- finally, you have the list of the nodes under the **`nodelist`** column



2 / what jobs are running

If you wanna inspect what jobs are currently running on a machine, **`squeue`** is the command suited for that.

```
squeue -u yourusername
squeue --me
```

will inform you about the status of your jobs.

see the man page for more info.

`scontrol` can be used for a more detailed reporting; check the manula page.
A typical usage may be **`scontrol show job`** ***`jobid`***

2 / what jobs are running

squeue's output

```
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
423696      base mcmc_bao afumagal  R 17-15:55:09      1 gen10-01
424428   pleiadi  AL_1006  almagal  R 12-13:04:34      1 len55
424699      base     bash   nbiava  R 9-17:42:27       1 gen10-12
425225      base mcmc_ok_ afumagal  R 6-00:40:43       1 gen10-09
425277   pleiadi   AL_995  almagal  R 5-19:46:12       1 len24
425336   pleiadi   AL_994  almagal  R 5-11:18:20       1 len57
425439      base     bash fubertos  R 3-19:30:30       1 gen10-14
425622      base pretar22 abotteon  R   19:09:50       1 gen10-02
425671      base     bash   nbiava  R   18:51:45       1 gen10-13
425692      base pretar26 abotteon  R   18:33:53       1 gen10-05
425724   pleiadi     Ek40 ggranato  R   14:08:22       3 len[17-19]
425725   pleiadi   b1Ek40 ggranato  R   14:05:11       3 len[01-03]
425727   pleiadi   b3Ek40 ggranato  R   13:59:44       3 len[04-06]
425728   pleiadi   b4Ek40 ggranato  R   13:53:05       3 len[07-09]
425729   pleiadi   b1Ek40 ggranato  R   13:27:54       3 len[45-47]
425730   pleiadi Ek40noBH ggranato  R   13:26:56       3 len[48-50]
425731   pleiadi b3Ek40no ggranato  R   13:26:02       3 len[51-53]
425732   pleiadi b4Ek40no ggranato  R   13:25:12       3 len[25-27]
425734   pleiadi noBHnoco ggranato  R   11:39:35       3 len[28-30]
425736   pleiadi    ncs30 ggranato  R   11:37:22       3 len[31-33]
425745    euclid     bash   frizzo  R    2:16:26       1 gen09-18
425746      base  BoxNoBH gmurante  R      18:38       1 gen10-03
425747      base     bash   ftomba  R       7:15       1 gen10-11
```

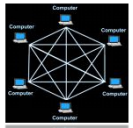
JOBID
is a unique
id for the job

PARTITION
On what partition
it is running

STATUS
R = running, PD = pending

NODELIST
On what nodes our job is running

REASON
why it is pending

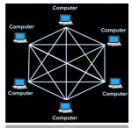


3 / immediate execution

`srun` allows you to immediately allocate resources and launch the tasks in a single command

`srun -n3 /bin/hostname` is running /bin/hostname on 3 cores
`srun -N3 /bin/hostname` is running /bin/hostname on 3 cores each of which will be on a different node (note the lower- and upper-case n and N)

Of course srun has options to set what resources are allocated and how the tasks are distributed on the resources - see the man page for the details.



4 / interactive jobs

A useful `srun` option is the interactive job, where you ask for some computational resources to be allocated for you:

```
srun -nx -Ny --partition=ppp -A account  --pty bash
```

This will allocate ***x*** cores distributed over ***y*** nodes chosen in the partition ***ppp***, and will execute a shell.
The time will be accounted on the account ***account.***

Try:

```
srun -n4 -N1 --partition=oats_guests --pty bash
```

Use this access with caution, it may be unfair and not compliant with the netiquette

Submitting jobs

Our preferred way to execute codes will be by submitting jobs - which is also the standard way since it is about fai sharing of available resources.

The concept is that you describe what “job” you want to execute by specifying:

- the *resources* you want to allocate (how many cores)
- how the resources are *distributed* (on how many nodes)
- how much *memory* you need (per-resource or per-node)
- how much *time* the job will run
- the *account* to be used
- some instructions for the shell - like env variables, file operations, cd, …
- the execution command

How to define a job

see documentation on https://slurm.schedmd.com/

Submitting a job means that you “send” to the scheduler a text file, named “batch files”, that specifies the properties of the run itself

- How many resources
  - how many nodes
  - how many processes per nodes
  - how many cores per process
  - how much memory
  - how much time
- What modules
- What mpirun/srun options and bindings

How to define a job

```
#!/bin/bash
#SBATCH --partition=hpc_school
#SBATCH --account=hpc_school

#SBATCH --nodes=_NN_
#SBATCH --ntasks-per-node=_TT_
#SBATCH --cpus-per-task=_CC_
#SBATCH --mem=_GG_GB
#SBATCH --exclusive

#SBATCH --job-name=janedoe
#SBATCH -t 0-0:20 # time (D-HH:MM)
#SBATCH -o slurm.%N.%j.out_9.3.0 # STDOUT
#SBATCH -e slurm.%N.%j.err_9.3.0 # STDERR

module purge
module load default-gcc-11.2.0

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
export OMP_PLACES=_PP_
export OMP_PROC_BIND=_BB_

./my_wonderful_code arg1 arg2 ... > output
```

where you want to run on the cluster

what account is paying for the time

how many nodes you want

how many tasks per nodes

how many cores per task

how much memory

you do not anyone else on this node

The environment

Your codes will normally use some libraries, either standard (i.e. routinely found on all platforms) or not (i.e. you may have to ask for installation, or you may have to install them in your space).

In any case, for sure you'll need a compiler and a software stack based upon it (the MPI and the libraries).

Since there may be, and there will be, many different software stacks - for instance: based on different versions of the same compiler, or based on different compilers - a common way to organize that is through ***modules***.

`Modules` are organized collections of software stacks that coherently direct the compilation and the dynamic loading at run time to the correct locations.

The environment

The fundamental commands that you need are

`modules av`	shows the available modules and theri dependencies
`modules load` *`mmm`* `shortcut:` **`ml`**	load module *mmm* (in some cnfigurations it also loads the dependencies)
`modules li`	list the modules that you have loaded
`modules purge`	unload modules and reset the environment
In these days it will be	**`modules load`**

Netiquette

There are few rules that *must* be respected on a cluster.

The more the cluster is “local” and run “informally” - for instance our INAF’s in-house PLEIADIs - the more these rules are to be respected

(I mean.. on big tiers they make you repsect the rules at the cost of some freedom)

Netiquette

- **Be aware of the environment (as you always should do):** the fact that it is an "immaterial" environment does not mean that it does not have its own equilibrium. There also other living beings, there.

- **do not use the login node for your jobs**
  - the login nodes serve all the user
  - only small codes should run on the login node (potentially no ones) "small" in terms of memory occupancy and in terms of run-time
  - yes, you may have that python stuff.. post-processing... -.- No. Now you do not have any more reason: learn the Jedi way of parallelism, move your codes to parallel. Or at least, encapsulate them on a job and launch the job.

Netiquette

- **Pay attention to your disk occupancy.**

  Ask for the rules.
  Is the file-system meant for long-resident data?
  Are there separated area for production (= fast parallel file systems) and long-term storage?
  Is the file system backed-up? which area?

  If the file system is not for long-term storage, you should take for granted, *before flooding it with data*, that you will have to move the data elsewhere
  (there are specific separate calls for computation and for storage)

Our set-up

Cluster: pleiadi @ OACT

pleiadi.oact.inaf.it

partition: **hpc_school**
account: **hpc_school**

then, to get a node

```
srun -N1 -A hpc_school  --partition hpc_school --pty bash
```