

The structure of a basic batch job

To work with a batch, the user should build a batch submit file. SLURM accepts batch files that respect the following basic syntax:

```
#!/bin/bash

#

#SBATCH <OPTIONS>

(...)

srun <INSTRUCTION>
```

The `#SBATCH` directive teaches SLURM how to configure the environment for the job at matter, while `srun` is used to actually execute specific commands. It is worth now getting acquainted of some of the most useful and commonly used options one can leverage in a batch submit file.

In the following, the structure of batch jobs alongside with few basic examples will be discussed.

Submit basic instructions on SLURM with `srun`

In order to run instructions, a job has to be scheduled on SLURM: the basic `srun` command allows to execute very simple commands, such as one-liners on compute nodes.

The `srun` command can be also enriched with several useful options:

- `N<int>` allows to run the command on `<int>` nodes, for instance:

```
srun -N5 /bin/hostname
```

will run the `hostname` command on 5 nodes;

- `v` option activates verbosity (`vvv` is max verbosity);
- `w` option activates manual node selection.

If the user needs to specify the run conditions in detail as well as running more complex jobs, the user must write down a batch submit file.

#SBATCH options

A SLURM batch job can be configured quite extensively, so shedding some light on the most common sbatch options may help us configuring jobs properly.

To have a more complete list of command options you can visit the SLURM documentation. [30]

- `#SBATCH --partition=<NAME>`
This instruction allows the user to specify on which queue (partition in SLURM jargon) the batch job must land. The option is case-sensitive.
- `#SBATCH --job-name=<NAME>`
This instruction assigns a “human-friendly” name to the job, making it easier for the user to recognize among other jobs.
- `#SBATCH --output=<FILE>`
This instructions allows to redirect any output to a file (dynamically created at run-time).
- `#SBATCH --nodelist=<NODES>`
This instruction forces SLURM to use a specific subset of nodes for the batch job. For example: if we have a cluster of five nodes: `node[1:5]` and we specify `--nodelist=node[1-2]`, our job will only use these two nodes.
- `#SBATCH --nodes=<INT>`
This instructions tells SLURM to run over `<INT>` random nodes belonging to the partition.
N.B. SLURM chooses the best `<INT>` nodes evaluating current payload, so the choice is not entirely random. If we want specific nodes to be used, the correct option is the aforementioned `--nodelist`.
- `#SBATCH --ntasks=<INT>`
This command tells SLURM to use `<INT>` CPUS to perform the job. The CPU load gets distributed to optimize the efficiency and the computational burden on nodes.
- `#SBATCH --ntasks-per-node=<INT>`
This command is quite different from the former one: in this case SLURM forces the adoption of `<INT>` CPUS per node. Suppose you chose 2 nodes for your computation, writing `--ntasks-per-node=4`, you will force the job to use 4 CPUS on the first node as well as 4 CPUS on the second one.

- **#SBATCH --time=<TIME>**
This command sets an upper time limit for the job to be considered running. When this limit is exceeded, the job will be automatically held.
- **#SBATCH --mem=<INT>**
This option sets an upper limit for memory usage on every compute node in the cluster. It must be coherent with node hardware capabilities in order to avoid failures

Advanced batch job configuration

In the following we present some advanced SBATCH options. These ones will allow the user to set up constraints and use specific computing hardware peripherals, such as GPUs.

- **#SBATCH --constraint=<...>**
This command sets up hardware constraints. This allows the user to specify over which hardware the job should land or should leverage. Some examples may be: `--constraint=IB` (use forcibly Infini-Band nodes) or `--constraint=CPUTYPE` (use forcibly CPUTYPE CPUs).
- **#SBATCH --gres=<GRES>[:<#GRES>]**
This command allows (if gres are configured) leveraging general computing resources. Typical use-case: GPUs. In that case, the command looks like:
`--gres=gpu:<INT>` where `<INT>` is the number of GPUs we want to use.
- **#SBATCH --mem-per-cpu=<INT>**
This command sets a memory limit CPU-wise.
N.B. if one sets this instruction, it is mandatory for it to be the only constraint on `--mem` in the batch job configuration.

In the following, few utilization examples are given in order to practice with these concepts.

Retrieve job information

A user can retrieve information regarding active job queues with the `squeue` command for a synthetic overview of SLURM job queue status.

```
bash-4.2$ squeue
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
      8501  slurmHPC_  test_sl apascoli  R        0:01        1 hpc-200-06-05
```

The table shows 8 columns: JOBID, PARTITION, NAME, USER, ST, TIME, NODES and NODELIST(REASON).

1. **JOBID** shows the id corresponding to the submitted jobs.
2. **PARTITION** which in SLURM is a synonym of "queue" indicates to which partition the node belongs.
3. **NAME** it corresponds to the name assigned in the submit file, otherwise it will match the name of the submit file.
4. **USER** indicates the user who submitted the job.
5. **ST** indicates if the jobs are running ("R") or if it is pending ("PD").
6. **TIME** shows how long the jobs have run for using the format days-hours:minutes:seconds.
7. **NODES** indicates the number of machines running the job.
8. **NODELIST** indicates where the job is running or the reason it is still pending.

Among the information printed with the `squeue` command, the user can find the job id as well as the running time and status.
In case of a running a job, the command:

sstat -j <jobID>

will give detailed infos on the status; where the jobID is given to you by SLURM once you have submitted the job.

Then, to see specific information about a single job use:

```
bash-4.2$ sstat --format=JobID,AveCPU -j 8501
JobID      AveCPU
-----
8501.0      213503982+
```

The option **--format** allows to customise the output based on the desired features.

For instance in the example above are shown:

1. **JobID**
2. **AveCPU** Average (system + user) CPU time of all tasks in job.

Many more features are listed in the slurm manual. [\[31\]](#)

Killing a submitted job

Once submitted, a job can be killed by the command:

scancel <jobID>

It is recommended to give scancel an array of jobIDs to kill multiple job at once.

Please **DO NOT USE** scancel in for loops or shell scripts, as it can result in a degradation of performance.

More features are listed in the SLURM manual. [\[33\]](#)