Submission examples

Below some examples of submit files follow to help the user get comfortable with Slurm.
See them as a controlled playground to test some of the features of Slurm.

Simple batch submit

```
#!/bin/bash
#
#SBATCH --job-name=tasks1
#SBATCH --output=tasks1.txt
#SBATCH --nodelist=hpc-200-06-[17-18]
#SBATCH --ntasks-per-node=8
#SBATCH --time=5:00
#SBATCH --mem-per-cpu=100

srun hostname -s
```

To execute this script, the command to be issued is

sbatch <executable.sh>

```
bash-4.2$ sbatch test_slurm.sh
Submitted batch job 8501
```

The output, from the option --output=tasks1.txt, should be something like (hostnames and formatting may change):

```
-bash-4.2$ cat tasks1.txt
hpc-200-06-17
hpc-200-06-17
hpc-200-06-17
hpc-200-06-17
hpc-200-06-17
[...]
```

As we can see, the **`--ntasks-per-node=8`** option was interpreted by slurm as “reserve 8 cpus on node 17 and 8 on node 18 and execute the job on those cpus”.
It is quite useful to see how the output would look in case of **`--ntasks=8`** was used instead of **`--ntasks-per-cpu`**. In that case, the output should be exactly:

```
hpc-200-06-17
hpc-200-06-18
hpc-200-06-17
hpc-200-06-18
hpc-200-06-17
hpc-200-06-18
hpc-200-06-17
hpc-200-06-18
```

As we can see the execution involved 8 CPUs only and the payload was organized to minimize the burden over the nodes.

Simple MPI submit

To submit an MPI job on the cluster, the user must secure-copy the .mpi executable over the cluster.

In the following example, we prepared a C script that calculates the value of pi and we compiled it with:

```
module load compilers/openmpi-4-1-5_gcc12.3
mpicc example.c -o picalc.mpi
```

N.B: before compiling the file, you need to load the proper module as shown above. You can find the available modules by running **module avail**. Finally, it is possible to see the list of the loaded modules with:

```
-bash-4.2$ module list
Currently Loaded Modulefiles:
  1) compilers/openmpi-4-1-5_gcc12.3
```

To execute multi-node job is required to enable passwordless access inside the cluster. It can be achieved executing the following commands:

```
ssh-keygen -t ecdsa -N '' -f ~/.ssh/id_ecdsa
umask 0077
cat ~/.ssh/id_ecdsa.pub >> ~/.ssh/authorized_keys
umask 0022
```

Here's an example of a submit file:

```
#!/bin/bash
#
#SBATCH --job-name=test_mpi_picalc
#SBATCH --output=res_picalc.txt
#SBATCH --nodelist=... #or use --nodes=...
#SBATCH --ntasks=8
#SBATCH --time=5:00
#SBATCH --mem-per-cpu=1000

srun picalc.mpi
```

If the .mpi file is not available on desired compute nodes, which will be the most frequent scenario if you save your files in a custom path, computation is going to fail. That happens because Slurm does not take autonomously the responsibility of transferring files over compute nodes.

On way of acting might be to secure copy the executable over the desired nodelist, which can be feasible only if 1-2 nodes are involved. Otherwise, Slurm offers a **srun** option which may help the user.

In this case, the srun command has to be built as follows:

```
srun --bcast=~/picalc.mpi picalc.mpi
```

Where the **--bcast** option copies the executable to every node by specifying the destination path. In this case, we decided to copy the executable into the home folder keeping the original name as-is.

Simple GPU submit

The submission of a job involving GPU follows the same syntactic.

```
#!/bin/bash

#SBATCH --job-name=v100
#SBATCH --nodes=1
#SBATCH --ntasks=8
#SBATCH --gres=gpu:4


srun <gpu_script>
```

It must be specified the number of GPUs needed for the job with the option **--gres=gpu:<n_of_GPUs>**.

It is not mandatory to specify the partition where to submit the job as SLURM will find the nodes with GPUs.

Simple Python submit

The submission of a python job over the cluster follows the syntactic rules of previous use-cases. The only significative difference is that we have to specify the compiler/interpreter when we call the srun command.

In the following, we show how to execute a python job over 16 CPUs of a selected cluster node.

```
#!/bin/bash
#
#SBATCH --job-name=prova_python_sub
#SBATCH --output=prova_python_sub.txt
#SBATCH --nodelist=hpc-200-06-05
#SBATCH --ntasks=16
#SBATCH --time=5:00
#SBATCH --mem-per-cpu=100

srun python3 trial.py
```

Again, if secure-copying the executable over every node involved in our computation is an un-optimized operation, the user should add the **`--bcast`** option to the srun command.

Python submit with a virtual environment

With Slurm, we can also submit python jobs that leverage a virtual environment. This comes handy if the job needs to use packages that are not included in the standard python setup on the cluster.

In order to do so, the user has to:

1. Create a virtual environment in Python
2. Activate the environment
3. Install the packages that the job needs
4. Deactivate the environment

These 4 steps are required only once.

Henceforth, the user can add these lines to the submission script and use the desired virtual environment:

```
source PATH_TO_VENV
srun python myscript.py
deactivate
```