Login to the HPC cluster

1. If you are using a Windows machine, make sure that you have an ssh client (preferably MobaXTerm) installed. Depending on the local cluster access policies you may have to use your university's VPN to get access.
2. Copy the content of the `exercises_material/tasks-SLURM` directory to a working directory on the HPC cluster. In case your local machine runs Linux, MacOS or a current Windows version (10 or 11), you can use `scp` or `rsync` for this purpose. On Windows you can use WinSCP, FileZilla, or MobaXTerm's builtin file browser.

Prepare the exercise

1. Every cluster has its own policies, parameter defaults, and assumptions. The Slurm versions have been tested on Darmstadt's Lichtenberg cluster. If you are working on a different cluster, other or additional parameter restrictions may apply.
2. If you are unsure, try to submit the job script `simple.slurm` and note any error messages. The exact steps to submit these jobs are described in Exercise 1. Slurm's error messages are usually quite straightforward and comprehensible if there are mandatory parameters missing.
3. If you have to make any additions to `simple.slurm` to make it run correctly, remember to also apply these changes to the other example scripts.

Exercise

A Simple Job Script

You can incorporate a lot of non-trivial logic inside job scripts. However, to start out we will use a very bare-bones template. If you start to work on a new system, it is often preferable to keep the first tests very simple. This eases debugging and lets you use them as templates for more complex tasks later on.

1. Take a look at the script to find out what it is supposed to do. Use `nano`, `less`, `vim`, or any other text editor / text viewer you prefer:

   ```
   > nano simple.slurm
   ```

2. Check if there are any parameters you are not familiar with. If so, try the manual page of your cluster's job submit command, i.e.:

   ```
   > man sbatch
   ```

   or the Slurm website for the `sbatch` command.

   Hint: On manpages, you can forward-search for a term by typing `/<searchterm>`. There are more options for searching, which you can find on the manpage for `less`, the default display application for manpages.

3. If everything looks okay to you, submit the job:

   ```
   > sbatch simple.slurm
   ```

Track its state via the `squeue` command.

4. Once the job is finished, check its generated output file(s) to verify it ran without error.

Task Arrays

Task arrays are an effective and simple way to start a swarm of similar jobs through a single job script. Although it is not applicable in all situations, any workload that can be transformed into a task array can make your workflow much more efficient.

1. As in the previous exercise, take a look at the example script first before you submit them. Make sure to understand what each line and parameter is supposed to do. The script is named `array.slurm`.

2. Note that some scheduler directives are commented out by adding an extra `#`. This is to show different ways of defining task array indices and additional directions like how many tasks should be allowed to run concurrently. To test these different settings, try commenting out (adding another `#`) and commenting in (removing a `#`) different array directives. Check how many overall tasks are running, how many are concurrently running, and which indices they have.

Dependencies

Dependencies are great to set up job pipelines without having to wait for the first job to finish. This can save significant amounts of time if. For instance, if you have to partition a big calculation into several shorter steps to fit within the cluster's maximum runtime constraints.

1. As in the previous exercises, inspect the job files for this example and figure the purpose of each line. The files are `dependency_1.slurm` and `dependency_2.slurm`.

2. Start the first job, take note of its job ID, then start the second job depending on the first. Check the job status (`squeue` command) to verify that the second job does not start until the first job has finished:

```
> sbatch dependency_1.slurm
Submitted batch job <jobID>
> sbatch -d afterok:<jobID> dependency_2.slurm
```

   Note: The option `-d` is short for `--dependency` for Slurm. Both commands can be used interchangeably.

Noon Break Exercise: Putting It All Together

With the help of the slides and previous examples, design three job scripts that accomplish the following:

1. The first script should generate a task array with indices 2, 5, and 8. It should execute `./square.sh value` where `value=$((SLURM_ARRAY_TASK_ID * 1000))`. Write the output into task-specific output files named `task_<task_ID>.out`. Also add a `sleep 60` command so it takes a bit of time to finish.
   Hint: Inspect the line with the `--output` parameter in the `array.slurm` job script.

2. The second script should also generate a task array. The array should have the same array indices as the previous job array (there is a 1-to-1 correspondence between the task indices of the current job array and previous array). Each task reads from a single previous task's output file. There are multiple ways to do this in bash, for example:

   ```
   input=$(cat task_${SLURM_ARRAY_TASK_ID}.out)
   ```

   You can then use the `input` variable to execute `./add_self.sh $input`. Again, add a `sleep 60` command at the end.
   Reminder: You will need Slurm's `--dependency=aftercorr` command for this to work, and the second array must have *exactly* the same task indices as the first one. Let the second task array write its outputs in files named `task2_<task_ID>.out`

3. The third script should be a single job that reads all of the second job's output files and computes the sum of all values contained in the files. You can just execute `./summarize.sh` within the job file, as long as the output of the previous jobs are contained in the `task2_<task_ID>.out` files.

4. When submitting, chain all of the three jobs through dependency directives so you can submit them back-to-back without having to wait for them to finish.

5. Augment the `dependency_wrapper.sh` script so you can submit all three jobs in one go.

6. If you get stuck or keep getting error messages, try to find the error with the help of the course materials or the links to further documentation below!

References

- Slurm manpage for the sbatch command: `https://slurm.schedmd.com/sbatch.html`