SLURM Exercises

Programs and scripts mentioned in this document are already available at the *Escritorio/alumnos/SLURM* folder. Copy them to your local account and use them from your local folders.

Besides the specific scripts/programs required in each case, collect screenshots and output files that illustrate their correct operation and that the correct results are generated in each case.

1. Write a SLURM job that solves the following problem:

   We have several text files names_1.txt, names_2.txt,…. For each file we want to generate a new file in upper case letters (with a different extension). (Hint: you can use the *tr* command to perform this action: i.e. *tr [:upper:][:lower:]*).

2. Write a script (either in bash or Python) that submits a workflow that carries out the following operation with the *names_*.txt*:
   a) Generates upper case versions of each file (as in the previous exercise).
   b) All files (original and upper case) are sorted alphabetically (Hint: use the *sort* command from Linux).
   c) A summary file with the first and last name of each file is generated using the script *summary.sh* (available at *Escritorio/alumnos/SLURM/*). Take a look at it and run it at the command line as a stand-alone command to understand how it works.

      ```
      summary.sh file_1 file_2 ….file_N
      ```

   Your script should try to submit as many parallel jobs as possible to complete the whole task in the minimum time if computing resources are available at the cluster. You can use SLURM dependencies, job arrays or any other feature.

   There are two main additional issues to solve:

   a. calling *sbatch* commands from a bash script / Python program. Document "How to run Linux commands with Python" might be helpful, in the case of Python. In Bash, there are many websites with information about assigning output of commands to shell variables (see for instance: https://www.cyberciti.biz/faq/unix-linux-bsd-appleosx-bash-assign-variable-command-output/).
   b. get the job id generated by SLURM and use that value to create new *sbatch* commands using such id as a dependency. Using the *parsible* option in the sbatch command would be the easiest way to do it. However, provide an alternative solution that does not use such option (this requires some string manipulation).