02613 Python and High Performance Computing

Exercises for Week 1

---

1. Connecting and transferring files to the HPC

The focus of this exercise is to learn how to connect to the HPC nodes, transfer files and run code on the HPC, preferably using the terminal.

1. **Autolab** Write a Python program that writes "Hello world" to a file (e.g., called 'content.txt'). It should also print the same text to the screen.
2. Transfer your Python file to the HPC using scp, sftp, FileZilla, or other (see guide File Transfer to/from HPC on Learn).
3. Connect to an HPC terminal (see guide Connecting to an HPC Terminal on Learn). If you are *not on a DTU network*, you may need to use a VPN or set up SSH keys - see the guide. If you used SSH, make sure you called `linuxsh` so you are *not* on a login node.
4. Create and/or navigate to your working directory (on the HPC) for this exercise (see guide Linux Terminal Cheat Sheet on Learn).
5. Initialize the course conda environment (see guide Initializing the 02613 Conda Environment on Learn).
6. Run your Python program.
7. Transfer the generated file (e.g., `content.txt`) back to your local computer.

2. Job Scripts

For all scripts below, use `/bin/sleep 60` (or a longer period, like 100 or 120 seconds) as the command to run, and use 'hpc' as the queue name (except if stated otherwise).

1. **Autolab** Write a simple job script, like the one shown in the lectur and submit it.
    a) Check the status with bstat and/or bjobs. Use 'man bjobs', to get information about the options or check the online documentation.
    b) You can add a walltime limit to the script. Can you see that limit in the bstat or the bjobs output?

2. **Autolab** Write a job script that sends you notifications when the job starts and ends - see 'man bsub' for the details or check the online documentation. Take a look at the job summary, i.e. what information can you retrieve from that?

    a) To test, increase the period in the sleep command to be longer than the wall time limit, and submit the job again. What happens?

```
TERM_RUNLIMIT: job killed after reaching LSF run time limit.
Exited with exit code 140.
```

3. The default `hpc` queue has nodes of different type, e.g. with different CPUs. The CPU type can be requested as a feature in a command script.

    a) **Autolab** Use the `nodestat` command to check which CPU types are available in the `hpc` queue, and then submit a job script that requests one of the types. See the Batch Jobs under LSF 10 webpage for information on how to do that.

```
nodestat -F hpc
```

    b) Add the necessary commands to your job script, to print the CPU type - and check in the job output that your job did indeed run on a node with the requested feature. Note: there is slight difference in the type request and the type variable: the variable contains a '-', while LSF uses a '_'.

The next exercises are a preparation for later weeks, where we want to submit multi-core jobs to the batch system:

4. **Autolab** Write a job script that requests 1 node and 4 cores.
5. **Autolab** Write a job script, requesting 1 node and 16 cores. Does it run? If the job doesn't start, use 'bjobs -p' to check for the reason.

yes it will run (I have 48 nodes)

6. **Autolab** Write a job script, requesting 1 node and 64 cores. Does it run? If the job doesn't start, use the `bjobs -p` command to check for the reason.

Clean up:

7. Check all your submitted jobs with 'bstat' again. If there are any left, that still are in status 'PEND', please remove them with 'bkill JOBID' (the JOBID is the number in the first column of the 'bstat'/'bjobs' output).

Hints: to get the informations needed, use the 'man' command and take a look at the DTU Computing Center webpages https://www.hpc.dtu.dk/?page_id=2534