HPC Cluster Cheat Sheet

Cheatsheet is based on chapters 3,4 and 6 of this documentation: VSC HPC tutorial for UGent Mac users

Important note: In order to use the cluster you need a vsc account and ssh keys. Here we use `vsc40000` to refer to a standard vsc user. Please modify it with your vsc username. If you don't have an account, follow the instructions on chapter 2 How to get an account? .

Basics

Connect to the cluster

```
ssh vsc40000@login.hpc.ugent.be
```

Close the connection

```
exit
```

Moving files

Copy files from local to cluster (use `scp -r` for directories)

```
scp localfile.txt vsc40000@login.hpc.ugent.be:path/
```

Copy file from cluster to local (use `scp -r` for directories)

```
scp vsc40000@login.hpc.ugent.be:/path/to/clusterfile localpath/
```

Copy **large** files from local to cluster

```
rsync -rzv localfile vsc40000@login.hpc.ugent.be:path/
```

Copy **large** files from cluster to local

```
rsync -rzv vsc40000@login.hpc.ugent.be:to/clusterfile localpath/
```

List of rsync flags

`-z`	compress file data during the transfer
`-r`	*recursively*, allows to copy directories
`-v`	add verbosity
`-P`	display a progress bar
`-n`	simulate transfer without actually executing it, allows to preview which file(s) will be transferred

Manage Modules

Environment Modules is a system for dynamically loading software packages into your environment

Display all available modules

```
module av
```

Search a specific module from the list of available modules. Replace `softwarename` with, e.g., `matlab`.

```
module av | grep -ie softwarename
```

Load a module

```
module load softwarename
```

Unload a module

```
module unload softwarename
```

Unload all modules at once

```
module purge
```

Create a collection of modules

```
module load software 1 software2 software3 ...
module save my_collection
```

Load a collection of modules

```
module reload my_collection
```

List collections of modules

```
module savelistr
```

Display all modules of a collection of modules

```
module describe my_collection
```

Running Jobs

Each time you want to execute a program on the HPC you'll need 2 things:

- **The executable**: The program to execute from the end-user, together with its peripheral input files, databases and/or command options. (e.g.: a python script)
- **A batch job script**: file which will define the computer resource requirements of the program (a .pbs file)

Example of a simple .pbs file (file.pbs) .In this example you move in the directory my_directory and run the script *myscript.py*.

```
# !/bin/bash -l
cd my_directory
python myscript.py
```

The command `qsub` is used to run jobs on the HPC. The flags after `qsub` or the `"# PBS"` headers at the start of a .pbs file can be used to specify the amount of computational power/memory for the job. In this example the job requests 5 compute nodes with two cores on each node and 4GB of memory via `qsub`.

```
qsub -l nodes=5:ppn=2,mem=4gb file.pbs
```

Alternatively add `"# PBS"` headers within file.pbs and run `qsub file.pbs` without any flag in the command line

```
# !/bin/bash -l
# PBS -l nodes=5:ppn=2
# PBS -l mem=4gb
cd my_directory
python myscript.py
```

List of `# PBS` and `qsub` flags

`-l nodes=X:ppn=Y`	Request X compute nodes each with Y cores
`-l mem=Xgb`	Request X GB of RAM memory
`-l walltime=X:00:00`	Set estimated execution time to X hours 0 minutes and 0 seconds. If a job exceeds this time will be automatically terminated
`-m abe`	Receive e-mail notification when the job (a) aborts, (b) begins, (e) ends

Manage active jobs

Retrieve the status of all your jobs that are submitted and are not yet finished

```
qstat
```

Remove a job from the queue or stop it. Replace `$JOB_ID` with the ID of the job you want to kill (you can get the `JOB_ID` via the command `qstat`, looking at the first column.

```
qdel $JOB_ID
```

A more extended guide on the commands of this cheat sheet is available in this notebook