Worksheet

Documentation

The software and systems available at the CHPC are documented on the CHPC's wiki:

```
wiki.chpc.ac.za
```

An overview of how to login, the software installed on the cluster, example job scripts and how to submit and check jobs can be found in the ***Quick Start Guide***:

```
http://wiki.chpc.ac.za/quick:start
```

Logging in to CHPC systems and shell basics

Use the `ssh` command to connect

```
ssh username@lengau.chpc.ac.za
```

Replace *username* with the login **user name** given to you by the CHPC.

Look around and follow the demo. The lecturer will show you the **file systems** you have access to and how to list files, check where you are, move around, create sub-directories and copy files. This will use Linux commands you already know but will show specifics of how the CHPC cluster is configured and where and what software is available.

Quiz: What directory does `~/..` refer to?

Copy the example files from `student00` into your `example` directory on your **Lustre file system** directory *(the full command will be given by the lecturer, you may make note of it in the box below)*. Remember to create the `example` sub-directory first.

Interactive Job

When you log into the CHPC systems you connect to a **login node** (usually called `login1` or `login2`) but you can not run intensive programs here. When you need to do anything computationally demanding but still require to work interactively (see the output after typing commands) you need to request an **interactive job** from the **scheduler**

```
qsub -I -P WCHPC -q R1026350 -W group_list=training
```

The lecturer will explain this command and you can make notes in the spaces below.

`qsub`	
`-I`	
`-P WCHPC`	
`-q R1026350`	
`-W group_list=training`	

Software Modules

Software (libraries, programming tools, and applications) is installed on the CHPC cluster in `/apps` and, to make them easy to access, the CHPC uses GNU modules.

`module avail`	
`module load` *`Module`*	
`module list`	
`module unload` *`Module`*	

Quiz: What does the shell operator **`|&`** do?

Load the **modules** needed to run parallel python into your interactive session:

```
module load chpc/BIOMODULES
module load chpc/python/2.7.13_gcc-6.2.0
```

Now you will be able to run the parallel python example:

```
mpirun python hello-mpi.py
```

The output from that command is not very exciting, because your interactive session has access to only one CPU **core**. In order to access more cores, you will need to **submit** this command in a **job script** that requests more cores from the PBSPro **scheduler**.

Job Scripts

In your `example` sub-directory on your Lustre scratch file system you should have the `python1.pbs` example job script:

```
#!/bin/bash

#PBS -N python-1
#PBS -l select=1:ncpus=4:mpiprocs=4
#PBS -l walltime=0:5:00
```

```
#PBS -P WCHPC
#PBS -q R1026350
#PBS -W group_list=training
```

```
WD=/mnt/lustre/users/$USER/example
cd $CWD
```

```
module load chpc/BIOMODULES
module load chpc/python/2.7.13_gcc-6.2.0
```

```
nproc=`cat $PBS_NODEFILE | wc -l`
```

```
mpirun -n $nproc  python hello-mpi.py
```

Quiz: What does the shell expression **`$CWD`** mean?

What do the shell 'back tick' operators in `` `*statement*` `` do?

Now submit the job script to the PBSPro scheduler

```
qsub python1.pbs
```

To check if your job is running, use the `qstat` command, with the appropriate options and/or arguments:

```
qstat <job-id>
```

```
  -u <username>
```

```
  -x
```

```
  -w
```

Job Output

The **standard output** and the **standard error** of your *job script* are captured in the `.o` and `.e` files. These can be redirected to specific files with the `-e` and `-o` PBSPro commands.

```
#PBS -o /mnt/lustre/users/student_____/_____/python2.o
```

```
#PBS -e /mnt/lustre/users/student_____/_____/python2.e
```

The scheduler can also notify you of when your job **b**egins, **e**nds or **a**borts via email with the `-m` and `-M` commands.

```
#PBS -m _____
```

```
#PBS -M _____@_____
```

The `python2.pbs` job script file contains these extra commands. Edit this file to (1) put in the *full path* to your scratch directory for the `.o` and `.e` files, and (2) add *your* email address in the appropriate place (otherwise the messages only appear on `login1`) and submit this script to the scheduler. When it finishes you should find new files `python2.o` and `python2.e` in your working directory.

Quiz: What is the main disadvantage of specifying exactly the name of the output files?

The third example job script has some more shell commands in it to provide useful information about the job environment. Submit it and look at the output files. What do you see? And what does it mean?

PBSPro predefines environment (shell) variables that allow your job script to access information about the job. You've seen one variable used to find the **machine file** which contains the list of compute nodes allocated to your job. This is used by the `mpirun` command when it launches its MPI ranks. The same file of host names can be used by the `ssh` or GNU `parallel` commands for **throughput computing** (HPC jobs that don't use MPI).

PBS_NODEFILE	
PBS_JOBID	
PBS_JOBNAME	
PBS_TASKNUM	