Intro to using the cluster

Luke Beddow 22/09/22

This is an instruction document with some useful commands which can be run in the cluster terminal. Throughout this entire document angle brackets are used to indicate arguments that you should substitute eg <username>@example -> lbeddow@exmample. This guide is with linux in mind.

This guide makes reference to some example scripts which are also provided. This example scripts are indicated in bold, eg '**example_script.sh**'. If you read this whole document, it should explain the majority of the code in the example scripts.

Connecting to the cluster:

I use ssh to connect to the cluster via a terminal.

```
> ssh -J <username>@<ucl_login_node> <username>@<cluster_login_node>
> ssh -J lbeddow@tails.cs.ucl.ac.uk lbeddow@vic.cs.ucl.ac.uk
```

J is a double jump (hence the two addresses), you will need to enter your password twice. My example is for the computer science cluster, see the docs for myriad if needed: https://www.rc.ucl.ac.uk/docs/Clusters/Myriad/

Transferring data back and forth:

I use a tunnel to transfer data, for this open a new terminal and run the following:

```
> ssh -L <port_num>:<cluster_login_node>:22 <username>@<ucl_login_node>
> ssh -L 3333:vic.cs.ucl.ac.uk:22 lbeddow@tails.cs.ucl.ac.uk
```

Rather than enter your password every time, install sshpass: https://linux.die.net/man/1/sshpass

You can give sshpass a password directly with the -p flag, or better is to make a textfile and put the password on the first line, then give the name of this file with the -f flag. Now add the textfile to your .gitignore (if using git) and you won't leak your password!

```
> sudo apt-get install sshpass
> sshpass -p <yourpassword> ssh -L <port_num>:<cluster_login_node>:22
<username>@<ucl_login_node>
> sshpass -f <filename> ssh -L <port_num>:<cluster_login_node>:22
<username>@<ucl_login_node>
> sshpass -f key.txt ssh -L 3333:vic.cs.ucl.ac.uk:22
lbeddow@tails.cs.ucl.ac.uk
```

As this is a single jump, sshpass works nicely, it doesn't seem to work for a double jump. Once you run this command in a seperate terminal then you don't need to touch it.

I have written a script called '**connect_cluster.sh**' which sets up the connection and the tunnel, but note that it relies on a terminal emulator called 'terminator'.

With the tunnel running in its own terminal, you can now use 'scp' to copy across the connection.

To transfer a file from your pc to the cluster:

```
> scp -P <port_num> <local/path/to/file>
<username>@localhost:<cluster/path/to/file>
> scp -P 3333 /home/luke/mymujoco/Makefile
lbeddow@localhost:~/mymujoco/Makefile
```

To transfer a file from the cluster to your pc just swap the order:

```
> scp -P <port_num> <local/path/to/file>
<username>@localhost:<cluster/path/to/file>
> scp -P 3333 lbeddow@localhost:~/mymujoco/Makefile
/home/luke/mymujoco/Makefile
```

To transfer a folder just use the '-r' recursive flag:

```
> scp -P <port_num> -r <local/path/to/folder>
<username>@localhost:<cluster/path/to/destination>
> scp -P 3333 -r lbeddow@localhost:~/mymujoco/Makefile
/home/luke/mymujoco/Makefile
```

‘scp’ can be used alongside sshpass to make this faster. I have a script that does this called ‘**transfer_cluster.sh**’. I have set up three possible input flags:

- -d direction   specifiy either ‘up’ or ‘down’ to/from cluster, required
- -c command   specify a transfer command to do multiple files in batches, see the code
- -f filename   specify /path/to/filename.xyz

I would recommend editing this file for your own needs so that it is simple to upload and download. When I download, I download to a seperate folder as to not overwrite local copies.

NOTE: this file does NOT automatically create folders which do not yet exist when given a path, this will yield an error. Before you transfer, ensure all paths to files/folder exist.

Setting up the cluster:

When you connect to the cluster, you connect to a login_node. This is shared between many people and is used to submit jobs. After submission, jobs join a queue and are sent to a compute_node. It is bad practice to run a lot of things on the login_node as it will use up all the resources so other people will not be able to submit jobs. Hence, there is a command called 'qrsh' where you can get your terminal commands assigned to a compute_node. This is a great way to test your jobs, profile them, and set up libraries/python etc. Typically, I never queue for more than 5mins and I test all new scripts here. However, if I have made only small modifications I still compile on the login_node as it only takes a couple mins.

To get a 1hr interactive session with ‘qrsh’:

```
> qrsh -l tmem=<RAM>,h_vmem=<RAM>,h_rt=<hr:min:sec>
> qrsh -l tmem=16G,h_vmem=16G,h_rt=1:0:0
```

<u>Compiling</u>

I was advised that the default gcc version was very old (on the computer science cluster). You can check with:

```
> gcc -v
```

On the comp. sci. cluster there are scripts to upgrade gcc version, and the following was good for c++11:

```
> scl enable devtoolset-8 bash
```

This may not be an issue for you, but it is worth checking!

To use libararies, you will need to add them to the $LD_LIBRARY_PATH environment variable every time you want to use it. For example:

```
> export LD_LIBRARY_PATH=/share/apps/python-3.6.9/lib:$LD_LIBRARY_PATH
> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/path/to/your/shared/library
```

I have a script called ‘**configure_cluster.sh**’ that prepares the cluster before I need to compile.

Python

You will likely need to make a python virtual environment if you want to install your own python packages. On the comp. sci. cluster various python versions are found at:

/share/apps/

I then used the 'venv' python module: https://docs.python.org/3/library/venv.html

Since you have to run python to initialise a virtual environment, it could look like this (run from wherever you want your virtual environment folder to appear):

```
> /share/apps/python-3.6.9/bin/python3 -m venv <venv-folder-name>
```

Then activate the virtual environment, you will need to do this everytime you want to use it!

```
> source <venv-folder-name>/bin/activate
```

Once activated, you can use the ‘python3’ alias as normal:

```
> python3 -m pip install <package>
```

Submitting jobs:

To submit a job use qsub, and give the path to a bash script of commands to be run. It should include some #$ tokens to set options, please see the tutorial on Sun Grid Engine (SGE).

```
> qsub ~/mymujoco/cluster_job.sh
```

To see information about the jobs you have queuing/running use 'qstat'

```
> qstat
```

Here is an example job script which runs one job only (**cluster_job.sh)**:

```
# These are flags you must include - Two memory and one runtime.
# Runtime is either seconds or hours:min:sec

#$ -l tmem=4G
#$ -l h_vmem=4G
#$ -l h_rt=72:0:0
```

```
# These are optional flags but you probably want them in all jobs

#$ -S /bin/bash
#$ -j y
#$ -N cluster_job

# The code you want to run now goes here.

hostname
date

cd

# source python and export the library location
source mypython/python3/bin/activate
export LD_LIBRARY_PATH=/share/apps/python-3.6.9/lib:$LD_LIBRARY_PATH
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/clusterlibs/mujoco/mujoco210/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/clusterlibs/mujoco/mujoco-
2.1.5/lib

cd ~/mymujoco/rl

# run the script and pass in flags
python3 array_training_DQN.py "$@"

date
```

Here is an example which runs an array job, that is one script that spawns many different jobs (**array_job.sh**):

```
# These are flags you must include - Two memory and one runtime.
# Runtime is either seconds or hours:min:sec (1day=24, 2days=48,
3days=72, 4days=96, 5days=120)

#$ -l tmem=4G
#$ -l h_vmem=4G
#$ -l h_rt=120:0:0

# Some important notes
#dollar -t X-Y -> submit array job inclusive of both X and Y
#dollar -t X-Y:Z -> submit with stride Z, eg 2-10:2 means { 2, 4, 6, 8,
10 }
#dollar -tc Z -> Z is max number of concurrent jobs at once
```

```
#$ -S /bin/bash
#$ -j y
#$ -N Array35_baseline
#$ -t 1-30

# The code you want to run now goes here.

hostname
date

# safety measure to stagger processes starting
sleep ${SGE_TASK_ID}

cd

# source python and export the library location
source mypython/python3/bin/activate
export LD_LIBRARY_PATH=/share/apps/python-3.6.9/lib:$LD_LIBRARY_PATH
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/clusterlibs/mujoco/mujoco210/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/clusterlibs/mujoco/mujoco-
2.1.5/lib

# fix for multithreading issues on cluster
export OPENBLAS_NUM_THREADS=1

cd ~/mymujoco/rl

# run the script and pass in flags
# NB "@" is required: see
https://unix.stackexchange.com/questions/129072/whats-the-difference-
between-and
python3 array_training_DQN.py \
-j ${SGE_TASK_ID} \
"$@"
date
```

Notice above the variable ${SGE_TASK_ID}, when the array spawns many jobs, each of them is given this variable such that it equals 1 for the first job, 2 for the second, 3 for the third etc. Hence, here I pass it into my main python script so that I can run slightly different code for each array job.