ICHEC Access for NUI Galway Postgrad students (MScAI, MScAI Online, MScDA, PhD)

James McDermott

This document is intended to get our students up and running with ICHEC supercomputing resources.

If you get stuck, contact: `james.mcdermott@nuigalway.ie`.

Registration

1. First, you need to have a login for ICHEC. Create an account here if you don't already have one: `https://register.ichec.ie/register`.
2. Are you going to use NUI Galway *Condominium* access, or an ICHEC Class-C project? If you're not sure, ask your Lecturer, Supervisor or Programme Director. **NB MScAI Online students will use a Class-C project (see 4, below); other classes will use Condominium**.
3. If using Condominium access, send an email request to Noreen Goggin `noreen.goggin@nuigalway.ie`, who will give you a Word form to be filled-in. In this case your `account_name` will be `nuig02`.
4. Instead, if using a Class-C project, ask your Programme Director for the appropriate `account_name` for your project, e.g. `ngcom018c` for MScAI Online 2020/21. Include your ICHEC username, student number, and programme name in the email. Then go to this page: `https://register.ichec.ie/login/` and log in. You should see a big list of projects. Find the project with that account name and click `Apply` and then `Confirm`. Your Programme Director will receive an email notifying them of your application and will add you to the project.
5. We need ssh keys to login to ICHEC. If you don't already have an ssh keypair on your laptop, create one:

   ```
   ssh-keygen -t ed25519 # -t specifies the cryptography method
   ```

   Then find the public key, probably named: `$(HOME)/.ssh/id_rsa.pub`, and send this file to ICHEC by email, requesting off-campus ssh access. Do not send the private key (no `.pub` suffix.) For more, see:

https://www.ichec.ie/academic/national-hpc/documentation/tutorials/setting-ssh-keys.

6. Confirm that you can login to the kay supercomputer, e.g. using ssh:

   ```
   $ ssh username@kay.ichec.ie # use your ICHEC username
   ```

Creating an environment for Python scientific computing

7. Create a new Conda environment called myenv, and install some libraries in it:

   ```
   module load conda/2
   module load cuda/10.0
   conda create --name myenv python=3.7
   source activate myenv
   conda install matplotlib cma # our fruit-picking example will use them
   conda install tensorflow-gpu
   conda scikit-learn
   ```

   Installing Python packages using conda like this (or using pip) can only be done from the login node – not from a node you might acquire using srun as below. Also, it can only be done after running module load conda/2 and source activate myenv as shown above.

8. Try to acquire an interactive GPU node for 10 minutes like this (use your account_name from 3 or 4 above – don't confuse this with your username):

   ```
   srun -p GpuQ -N 1 -A account_name -t 0:10:00 --pty bash
   ```

   If it brings you to a prompt such as [username@n1 ~]$ then you have an interactive node and can proceed. Otherwise it might wait until a node becomes available. In that case, if you want you can type Ctrl-C and try again later.

9. Test that Tensorflow can be imported and sees the GPU.

   ```
   # Do these three commands once at the start of the session on a login node
   module load cuda/10.0
   module load conda/2
   source activate myenv
   # From now on, you can run Python and import Tensorflow
   python -c 'import tensorflow as tf; tf.test.gpu_device_name()'
   ```

10. Note: if you are using Tensorflow or another library that will take advantage of the GPU, then use GpuQ as above. Otherwise, use DevQ for normal CPU-only nodes, and omit the cuda line.

Batch jobs and Taskfarming

11. Above we acquired a node for short-term interactive use. For long-running jobs, we instead use *batch jobs*. That means we write a shell script and

submit it to a queue. Many users submit to these queues and when there is capacity, your job is taken off the queue and run.

See `submit.sh` for an example batch job. It has a specific format: first it gives details such as your account name and the queue you are submitting to. Then it loads some modules and your Python environment. Then it runs one or more commands. These could be Python scripts, e.g. `python hello.py`, or *taskfarming* commands. In the example `submit.sh`, there is a single taskfarming command.

Several commands allow you to query the batch job system:

```
$ mybalance
$ sinfo
$ squeue # eg `squeue | grep username`
$ scancel # use `man scancel` to read options
```

In particular, if you have submitted a job and it doesn't seem to have started, you can use `squeue | grep username` to look for it. Usually, you just have to wait a day or two. But you might see `AssocGrpBillingMinutes` in the final column: this means that the project has exceeded its resource limits and your job may not run. E.g. on `nuig02`, it will not run until the next month. If this happens on `nuig02`, you can consider applying for your own Class-C project – see later.

12. Taskfarming means running many similar commands, taking advantage of multiple CPUs on a single node, or even multiple nodes.

    For example, if we have a machine learning algorithm with some hyperparameters, we could try many values for the hyperparameters using taskfarming. In the example, `submit.sh` runs the command `taskfarm taskfarm.sh`. See `taskfarm.sh` to see the format it should be in: a list of shell commands. Each line will be run independently, on its own CPU, so many will run at the same time. If there are more lines than CPUs, then a new line is run automatically whenever a CPU becomes free. It's like a smaller version of the batch-job queueing system we already discussed, but now it refers to multiple *tasks* (commands) being queued on nodes which already "belong" to you, as opposed to multiple *batch jobs* from multiple users being queued for access to nodes.

13. Next we will run an example taskfarming batch job. On your laptop, edit `submit.sh` to include your NUI Galway email address and check the `account_name` is correct. Look at `fruit_picking.py`: you don't need to understand this example experiment, but notice it doesn't use Tensorflow, hence `submit.sh` requests `DevQ`, not `GpuQ`. Similarly, it doesn't load `cuda` since that is for GPU only.

    Use `scp`, `Putty`, or another remote file-copying program to copy the fruit-picking code and data files, and the `submit.sh` and `taskfarm.sh` files to `kay.ichec.ie:/ichec/home/users/<username>`. E.g.: `scp -r fruit_picking username@kay.ichec.ie:/ichec/home/users/username`.

14. Back on the `kay` login node (not on an interactive one), type `ls` to confirm your files are present. If you know how to use `cd`, `mv`, etc., then you can create a sensible working directory structure here. The files will be mirrored from the login node to interactive nodes and batch nodes.

    Make sure you are in the directory where you have the files `submit.sh`, `taskfarm.sh`, `fruit_picking.py`, and `fruit_picking.dat`. Then type `sbatch submit.sh`. It will tell you that the batch job has been submitted. However, it might not run immediately. You will receive an email both when the job begins executing and when it ends (look at `submit.sh` to see where you requested this). This very small job will take only a couple of minutes, and we have requested 1 node for 10 minutes (look at `submit.sh` to see how). If you want, you can exit from `kay` by typing `Ctrl-D` at the shell prompt.

15. When you receive an email to say the job is finished, log back in to `kay` and have a look at the output files. Then copy all the output files and images back to your own machine.

Applying for your own Class-C project

For typical assignments, `nuig02` condominium access should be enough. However, if you have a larger Capstone project or PhD research, you have the option to create your own Class-C project. It's quite easy: you can do it in a couple of hours and receive the approval within a week. Talk to your supervisor first.

Go here: https://www.ichec.ie/academic/national-hpc/national-service-projects, read the part about Class-C, and download the project template, a Word doc. In the doc, describe your research project very briefly, justifying the need for high-end computing.

Use the Core Hour Calculator on the same page to create a table estimating your ICHEC usage, and add that to your doc.

Then go to https://register.ichec.ie/login/project_apply, fill in the form and upload your doc.

ICHEC documentation for further reading

- Quick start guide: https://www.ichec.ie/academic/national-hpc/documentation
- Tutorials https://www.ichec.ie/academic/national-hpc/tutorials
- Kay user guide https://www.ichec.ie/academic/national-hpc/kay-user-guide
- Python, Anaconda, and environments on ICHEC: https://www.ichec.ie/academic/national-hpc-service/software/python-conda

- Conda environments: https://www.ichec.ie/academic/national-hpc/documentation/tutorials/conda-environments
- More on SLURM: https://www.ichec.ie/academic/national-hpc/kay-documentation/slurm-workload-manager
- SLURM commands: https://www.ichec.ie/academic/national-hpc/kay-documentation/slurm-commands
- SLURM for PBS users (still has some useful clues): https://www.ichec.ie/academic/national-hpc/kay-documentation/pbs-slurm
- Task farming: https://www.ichec.ie/academic/national-hpc/documentation/task-farming
- Hardware available: https://www.ichec.ie/about/infrastructure/kay
- Software available (only relevant for hardcore HPC (simulation, weather forecasting, etc, not relevant to those using the Python scientific stack): https://www.ichec.ie/academic/national-hpc-service/software
- Applying for your own project: https://www.ichec.ie/academic/national-hpc/national-service-projects