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

James McDermott james.mcdermott@nuigalway.ie

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

If you get stuck, or if you have problems with any steps and then figure out the solutions, please contact James McDermott.

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`. You will eventually receive an email stating that your account has been created, or possibly one asking you to choose a new username. Then you can proceed to 2.
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 and MScAI Online students will use a Class-C project (go to 4, below); other classes will use Condominium access (go to 3, below)**.
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, you need to know the `account_name` for your project. (It is `ngcom018c` for MScAI **Online** 2020/21. It is `ngcom019c` for MScAI (**classroom**) 2020/21.) For other classes please email James McDermott and state your ICHEC username, student number, and programme name.) Go to this page: `https://register.ichec.ie/login/` and log in. You should see a big list of projects. Find the project with your 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
   ```

   If it asks you for a passphrase, you can press return to say no. If it asks you for a filename, give `$(HOME)/.ssh/id_ed25519`. Then find the public key file: `$(HOME)/.ssh/id_ed25519.pub`, and send this file to (goar.sanchez@ichec.ie and simon.wong@ichec.ie). In the email, include your ICHEC username, `account_name`, and NUIG email, requesting off-campus ssh access. Do not send the private key (the private key has no `.pub` suffix.) They will reply to say it has been added.

   Contact James McDermott if you have problems creating the ssh keypair or finding the public key.
6. Confirm that you can login to the `kay` supercomputer, e.g. using `ssh`:

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

If you see an error `Access denied: public key` or similar, you could try:

```
$ ssh username@kay.ichec.ie -i ~/.ssh/id_ed25519
```

If you need to trouble-shoot, try following this guide:

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

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
   ```

   You will see a message like `To load the default (base) ....` It is not an error! It is fine, just continue.

   ```
   conda create --name myenv python=3.7
   conda activate myenv
   conda install matplotlib # our fruit-picking example code will use it
   pip install cma # a library you don't need to know about, used in our example later
   conda install 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 `conda activate myenv` as shown above.

   In future sessions (e.g. Step 9 below), you don't need to `conda create` the environment again or reinstall the libraries, but you do need to run the `module load conda/2` (and `module load cuda/10.0` if you need GPU), and `conda activate myenv`, in every future session.

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

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

   If it brings you to a prompt such as `[username@n360 ~]$` then you have an interactive node. You can try out some Python code. When finished playing around, type `exit` or just `Ctrl-D` to exit from your interactive node back the login node. (But note, this ends the interactive session and you cannot get back into the same session.)

   Or if you wait, after 10 minutes it will exit by itself with a message like `srun: Force Terminated job step 643191.0`.

   If it says `srun: error: Unable to allocate resources: Invalid account or account/partition combination specified`, that is probably because you didn't put in the correct `account_name` above.

   Otherwise it might wait until a node becomes available, with a message like `srun: job 642735 queued and waiting for resources`. In that case, if you want you can type `Ctrl-C` to give up, and try the interactive node later. But you can still proceed to the Batch jobs and Taskfarming steps (Step 11 etc. below).

Acquiring a GPU node

If you don't need a GPU you can skip these steps for now.

9. If you need GPU for training deep neural networks, then we need some extra steps. First, on a login node we will install `tensorflow-gpu`.

   ```
   module load cuda/10.0
   module load conda/2
   conda activate myenv
   conda install tensorflow-gpu
   ```

10. Next, acquire an interactive GPU node as in step 8 above, but using `GpuQ` instead of `DevQ` this time. If/when it gives you a node, test that Tensorflow can be imported and sees the GPU:

    ```
    module load cuda/10.0
    module load conda/2
    conda activate myenv
    python -c 'import tensorflow as tf; tf.test.gpu_device_name()'
    ```

    You will see a lot of messages about the GPU device.

    As before, type `exit` or just `Ctrl-D` to exit from your interactive node back the login node.

    If you later need a GPU for batch mode, then use the batch instructions below but include always the `module load cuda/10.0` command, and use `GpuQ` instead of `DevQ` or `ProdQ`.

Batch jobs and Taskfarming

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 representing our job, and submit it to a queue. Many other users are doing the same. When your batch job reaches the front of the queue it is allocated sole use of a node or nodes depending on the script. Kay has multiple queues including `DevQ` for quick initial testing, `ProdQ` for the main experiments, and `GpuQ` for experiments requiring GPU. `DevQ` and `ProdQ` use the same nodes, just a different queue.

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. A taskfarming script is a shell script of many lines. Each line will be run independently, on its own CPU, so many can run at the same time on one node. 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.

11. 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, using `#SBATCH` commands. 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. It runs the command `taskfarm taskfarm.sh`. See `taskfarm.sh` to see the format it should be in: a list of shell commands.

12. For every project we will need to customize these files. 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.

13. We have to copy these files from our local machine (laptop) to kay. **From your laptop**, 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>`.

Think of `scp` as `copy from to`. E.g.: `scp -i ~/.ssh/id_ed25519 -r ichec_example username@kay.ichec.ie:/ich` (Substitute your username, of course). Here, `ichec_example` is **local**, i.e. on your laptop. `username@kay.ichec.ie:` tells `scp` to copy to a **remote** machine, `kay` (and the username there); then `/ichec/home/users/username/` gives the destination on that remote machine.

14. Now 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` while you wait, by typing `Ctrl-D` at the shell prompt.

15. Several commands allow you to query the batch job system, while on `kay`.

    ```
    $ mybalance
    $ sinfo
    $ squeue # eg `squeue | grep username` # substitute your 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: probably only a few minutes if you have only requested one node for a few minutes, or a day or two if you have requested more. 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.

16. When you receive an email to say the job is finished, log back in to `kay` if not there already. In the same directory you should now have a `.out` file, e.g. `slurm-643296.out`. This is plain text and you can read it with `less slurm-643296.out`.

    If it says something like `CommandNotFoundError: Your shell has not been properly configured to use 'conda activate'.`, try editing `submit.sh` to replace `conda activate myenv` with new lines `conda init` and `source activate myenv`.

    Otherwise, if your `.out` file looks ok, type `ls` to see if `.png` files have been created by the `fruit_picking.py` program. Then **from your own machine**, copy all the output files and images back to your own machine. Use Putty again, or something like:

    ```
    $ scp -i ~/.ssh/id_ed25519 -r username@kay.ichec.ie:/ichec/home/users/username/ichec_example/
    .
    ```

    (Notice the trailing '`.`', which is really on the same line, not on a new line. It means copy to the current directory on the current machine, i.e. your laptop.)

    Once they are on your laptop, you can look at the `.png` files.

Applying for your own Class-C project

For typical assignments, Condominium access or Class-C 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. Read the following and then talk to your supervisor and James McDermott.

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 such as 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