Academic Compute Cluster

Kefei Duan

Introduction

This document provides some guidance for the usage of Academic Compute Cluster[1] provided on Engineering Compute Cluster (ENGR Cluster), which you may be able to use for your project.

All faculty, staff, and WashU students enrolled in any McKelvey Engineering course are granted access to this resource.

1 Access Hosts

The cluster can be accessed via SSH via these hosts:

- shell.engr.wustl.edu
- shell2.engr.wustl.edu

You could SSH to these hosts through your WashU Key, and the password will be your own password:

- ssh washukey@shell.engr.wustl.edu

You will be logged into your own home directory. Everyone has a quota limit under the home directory. You could use the following command to see your used space and total limits:

- quota -s

The hosts above are not meant for academic work. They are access hosts, where you could apply for compute resource.

2 Interactive Compute Jobs

2.1 CPU node

The script **qlogin** will start a session on an academic compute host. You could type the following command line in terminal to see some helpful information about **qlogin**:

- qlogin -h

Specifically, **qlogin** will start an interactive session with CPU resource. You could use **-c** to specify the number of CPU cores you want to allocate, up to 8.

[1] `https://washu.atlassian.net/wiki/spaces/EIKB/pages/261751080/Academic+Compute+Cluster`

2.2 GPU node

If you want to get access to GPU, you could use **qlogin-gpu** to start an interactive session with available GPU resource. It will start an interactive session on an academic compute host with a single GPU. You could use **-g** to specify the amount of GPU RAM you wish to use. It could be 2GB, 4GB, 10 GB or 20GB:

- qlogin-gpu -g 2

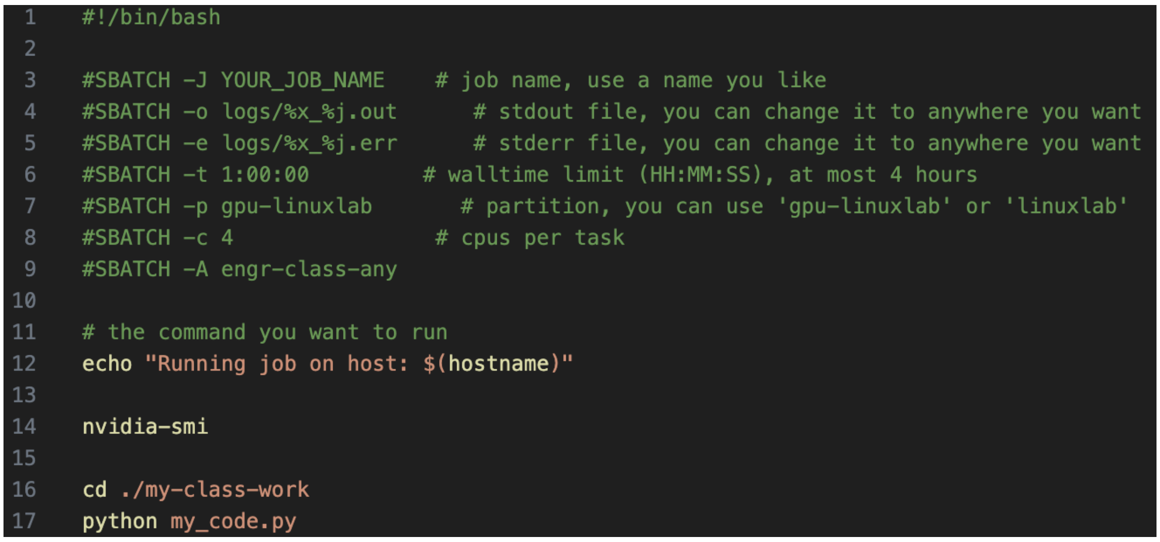
Requesting GPU RAM reserves, but does not limit, your GPU RAM usage. It is your responsibility to make sure your code or data stays within your request. Your code may fail if it uses more.

A job requesting 20GB of GPU RAM, the maximum available on the GPUs available, will request exclusive access to the GPU. It may wait longer to start.

3 Linuxlab Batch Jobs

Apart from requesting an interactive node, you can also submit jobs to run at the background rather than interactive. On the cluster, you can use `sbatch` in the SLURM[2] command to do that.

Most often, you will write a small script that contains the commands you want to run in the job, plus a set of special arguments that tell the SLURM scheduler how to handle your job. A simple example is shown below:



```
#!/bin/bash

#SBATCH -J YOUR_JOB_NAME    # job name, use a name you like
#SBATCH -o logs/%x_%j.out      # stdout file, you can change it to anywhere you want
#SBATCH -e logs/%x_%j.err      # stderr file, you can change it to anywhere you want
#SBATCH -t 1:00:00          # walltime limit (HH:MM:SS), at most 4 hours
#SBATCH -p gpu-linuxlab       # partition, you can use 'gpu-linuxlab' or 'linuxlab'
#SBATCH -c 4                # cpus per task
#SBATCH -A engr-class-any

# the command you want to run
echo "Running job on host: $(hostname)"

nvidia-smi

cd ./my-class-work
python my_code.py
```

`#SBATCH -p` specifies the partition you want to use. You can use `linuxlab` and `gpu-linuxlab` for this, which correspond to the CPU node and GPU node we mentioned above respectively.

Assume you write the content in the example in a file named `my_job.job`, you can then submit the job using the following command:

- sbatch my_job.job

[2] https://slurm.schedmd.com/documentation.html

You can use the command `squeue -u $USER$` to see the status of your running jobs.

For these batch jobs, it has the same CPU/GPU/time limits as the above interactive compute jobs.

For more information and usage of `sbatch`, you can refer to the documentation of SLURM, or you can also ask ChatGPT for help.

4 Set up Your Environments

If you use SSH to connect to the access host, and request a compute node, you should be able to use **conda** after logging in the node. You could then use conda to create your own environment for your project. You can first specify the PATH for installation of envs and pkgs:

- `export CONDA_ENVS_DIRS="$HOME/PATH_TO_ENVS_DIR/"`
- `export CONDA_PKGS_DIRS="$HOME/PATH_TO_PKGS_DIR/"`

Replace `PATH_TO_ENVS_DIRS` and `PATH_TO_PKGS_DIRS` to the path where you like.

You could then create your own environment using conda and install the packages you need.

You could start an interactive session to prepare your environment and for debugging purpose.

To submit a batch job on the background and use your own environment, you can specify the `CONDA_ENVS_DIRS` and `CONDA_PKGS_DIRS` as before in the script, and `conda activate` your environment in the script before running your code.

5 Interactive Application Access

You can also access the compute resource through the Open OnDemand interface at:
`https://linuxlab.engr.wustl.edu/pun/sys/dashboard/batch_connect/sessions`,
which will provide you with the access through your web browser.

There is a list of Interactive Apps provided, which you can open using your browser. You could explore that as you like. But I think the most helpful apps for your project would be **Academic Jupyter GPU Notebook**. You could specify the GPU RAM amount and the number of hours for the interactive session, up to 4 hours.

It will request a compute node, and while they are running you can connect/disconnect from them in your browser at will.

The environment and available applications on the **Academic Jupyter GPU Notebook** are different from the compute nodes we have applied before. There are already some python packages available. You could find the list here:
`https://washu.atlassian.net/wiki/spaces/EIKB/pages/983172085/Academic+GPU+Jupyter+Lab`.

If you want to use this **Academic Jupyter GPU Notebook** for your project, you could set up your environment by installing other packages as you like through **pip**:

- pip install –user package_name

6 Job Limits

Each interactive job, batch job, or interactive application on Open OnDemand will be a job. A single job has limits on CPU (at most 8), GPU (at most 20GB RAM), and running time (at most 4 hours). The max simultaneous jobs per academic user is 4.