**HPC Scripting Documentation**

**Walkthrough**

1. Insert SSH keys to bypass two factor authentication (server -> HPC)
2. Local machine script to run script in the HPC (step 3)
3. Node initiating script on the HPC to request a node and run training script (step 4)
4. Script to run program on the HPC to run train.py
5. Modify python code to accommodate HPC performance
   1. `sudo` not allowed
   2. `keyboard` module seems to be erroneous
   3. GUI based commands not allowed
      1. E.g.: matplotlib .show()

**Generating ssh keys**

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

**Insert SSH keys to bypass two factor authentication**

<https://www.ssh.com/academy/ssh/copy-id> is great and easy to use to transfer any ssh keys to a remote machine.

**Local machine script**

Take first argument and initiate train/test/clear on it

Call script on remote machine to initiate node

Pass arguments to the script that initiates nodes to be passed to the python script

**Node initiating script on the HPC**

Call srun to request for a node, arguments can be used to specify node configurations and requests such as:

-p gpu,dgxh #Partition(takes comma separated array)

--gres=gpu:1 #Generic resources required per node

--time=1-00:00:00 #Time per node

**Script to run program on the HPC**

arguments can be used to specify node configurations and requests such as:

#!/bin/bash

#SBATCH -J recycling # name of my job

#SBATCH -p gpu,dgxh # name of partition/queue to use

#SBATCH -o recycling.out # name of output file for batch script

#SBATCH -e recycling.err # name of error file for batch script

#SBATCH -c 1 # number of cores per task

#SBATCH -t 0 # number of threads per task

#SBATCH --time=1-00:00:00 # time needed for job

#SBATCH --mem=10g # memory needed for job

#SBATCH --gres=gpu:1 # consumable resources needed for job

#SBATCH --mem=10g # memory needed for job

Choosing partitions can be seen under “The Hardware” at <https://it.engineering.oregonstate.edu/hpc/about-cluster#:~:text=The%20College%20of%20Engineering%20HPC,or%20HDR%20InfiniBand%20network%20connection>

We ensure that required modules are loaded with `module load <module>` or simply `module restore <module\_name>` to restore a saved module.

We then load the python environment because OSU servers do not allow for installation of modules outside of virtual environments. Run ` pip install -q -r ./requirements.txt` so that when code changes, the new modules are also installed.

Run training with `python3 <file.py> <arguments>` and save the output to a file

scp saved file back to Austin’s machine

` rsync -a --ignore-existing <directory> <remote host>` when we want to sync entire directory

**Modify python code to accommodate HPC performance**

Added additional argument of `--hpc` with default as false, when true we avoid using the `keyboard` module. Also set device to “0” when using `--hpc` to utilize GPUs in codes that is not defaulted to gpu

**Issues encountered**

Issue #1: invalid node time

Solution: Some partitions have a time limit of 1 day instead of 7, limit all nodes to 1 day

issue #2: script not loading PATH variables and also .bashrc

solution: <https://superuser.com/questions/306530/run-remote-ssh-command-with-full-login-shell>

issue #3: HPC no gpus available

solution: choose other partitions; don’t use preempt

Issue #4: HPC old gpu drivers

Solution: Issue not resolved; refrain from using the V100s

Issue #5: python version remains the initial 2.7 despite `module load python/3.10`

Solution: use python3 instead

Issue #6: unable to pip install requirements on the HPC

Solution: Create virtual environment with ` python3 -m venv env`and pip install after activating it