CS 175 (W25): Project in Artificial Intelligence
Exercise 1

Due date: Wednesday, January 22, 2024 (Pacific Time)

Roy Fox
https://royf.org/crs/CS175/W25

In this exercise, you will familiarize yourself with HPC3. HPC3 is a shared-computing cluster at UCI that you can use for project experiments in this course. HPC3 efficiently manages the execution and prioritization of the jobs running on it using the Slurm job scheduling system.

Part 1 Getting an HPC3 Account

To begin your journey with HPC3, obtaining an account is the initial step. If you have already enrolled in the class, your account should be already be created and available to use, allowing you to proceed to the next step. If your account has yet to be created for some reason, reach out to us on the discussion boards. Please include **your name** and your **UCINetID**.

Part 2 Login to the Server

With an HPC3 account, use **ssh** for establishing a connection to the server. You will need the login server name, username, and password. The login server name is **hpc3.rcic.uci.edu** and the username and password are your UCINetID credentials (*not your ICS credentials)*. Connect to the server by executing the following command (with your username instead of `panteater`) in the terminal (or PuTTY for windows). Subsequently, you will be prompted to enter '1' for DUO authentication. If you're not on UCI's network, you'll likely to need to connect to it first using the UCI VPN.

```
ssh panteater@hpc3.rcic.uci.edu
```

Note that throughout these instructions we are using 'panteater' for the username and you should replace this with your UCINetID.
Once you log in, you will have a bash session on HPC3's login node. The login node `hpc3.rcic.uci.edu` should be primarily used to schedule jobs on the compute nodes via Slurm. If you try to run compute-heavy operations on the login node, they will most likely have issues or fail.

Part 3 Server Configuration

The most straightforward approach for managing and installing software and packages on the server is through Conda.

Part 3.1 Creating Conda Environments

Conda comes pre-installed on your server by default. To get access to conda, you must load its module. The provided instructions are derived from this source, albeit with some modifications.

1. Since Conda environment creation may be compute-intensive, start an interactive Slurm job on a compute node by running

```
srun -c 2 -p free --pty /bin/bash -i
```

   This can take a few seconds for the system to schedule this job.

2. Load the Conda dependency with

```
module load anaconda/2022.05
```

   At this step, to confirm whether Conda environment variables have been configured for your account, execute the following command, and the output should resemble the following.

```
conda --version
conda 4.10.3
```

3. Verify conda info by running the following command

```
conda info
    active environment :  None
      user config file :  /data/homezvol0/panteater/.condarc
 populated config files :
        conda version :  4.10.3
   conda-build version :  3.21.5
        python version :  3.9.7.final.0
      virtual packages :  __linux=3.10.0=0
                          __glibc=2.17=0
                          __unix=0=0
                          __archspec=1=x86_64
     base environment :  /opt/apps/anaconda/2021.11 (read only)
    conda av data dir :  /opt/apps/anaconda/2021.11/etc/conda
 conda av metadata url :  None
         channel URLs :  https://repo.anaconda.com/pkgs/main/linux-64
                         https://repo.anaconda.com/pkgs/main/noarch
                         https://repo.anaconda.com/pkgs/r/linux-64
                         https://repo.anaconda.com/pkgs/r/noarch
        package cache :  /opt/apps/anaconda/2021.11/pkgs
                         /data/homezvol0/panteater/.conda/pkgs
     envs directories :  /data/homezvol0/panteater/.conda/envs
```

```
                    /opt/apps/anaconda/2021.11/envs
          platform :  linux-64
        user-agent :  conda/4.10.3 requests/2.26.0 CPython/3.9.7 Linux/3.10.0-1160.53
           UID:GID : 1234567:1234567
        netrc file :  None
      offline mode :  False
```

Inspect the entries for package cache and environment directories in the preceding output. Each category should consist of two lines: one indicating the system's installed location (lines beginning with /opt/apps) and the other corresponding to your home path (lines beginning with /data/homezvol. . .). If any entries are absent, pointing to your user area, you'll need to create a file in your $HOME directory using your preferred text editor. Name the file '.condarc', and substitute the placeholders with your version, home directory, and username in the following content:

```
pkgs_dirs:
  - /data/homezvol0/panteater/.conda/pkgs
  - /opt/apps/anaconda/2021.11/pkgs
envs_dirs:
  - /data/homezvol0/panteater/.conda/envs
  - /opt/apps/anaconda/2021.11/envs
```

4. Initialize Conda for your shell by

```
conda init bash
```

This appends several lines to your ~/.bashrc. These lines should be positioned at the end of your ~/.bashrc file, delineated by the commencement and conclusion markers "conda initialize," encapsulating all lines within this delineation.

```
# >> conda initialize >>
# !!  Contents within this block are managed by 'conda init' !!
   <some lines are cut here>
__conda_setup="$('/opt/apps/anaconda/2021.11/bin/conda' 'shell.bash' 'hook'
2> /dev/null)"
if [ $?  -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/opt/apps/anaconda/2021.11/etc/profile.d/conda.sh" ]; then
        .  "/opt/apps/anaconda/2021.11/etc/profile.d/conda.sh"
    else
        export PATH="/opt/apps/anaconda/2021.11/bin:$PATH"
    fi
fi
```

```
unset __conda_setup
# «< conda initialize «<
```

Please be aware that your lines may exhibit slight variations contingent upon the conda module utilized. This ensures that conda is incorporated into your environment variables every time you log in to your server.

`conda init` may have directed you to close and create a new shell for changes to take effect. Close the current shell and create a new interactive Slurm job (as in step 1).

5. Now, you are prepared to create your local Conda environment. Proceed to create a local environment named `rl` as below.

```
conda create -n rl python=3.11
```

In case you encounter a permission error, ensure that Step 3 has been executed accurately.

6. Finally, activate your environment and commence the installation of requisite packages.

```
conda activate rl
```

For an exhaustive guide on managing Conda environments, refer to this link.

Part 3.2 Installing and starting JupyterHub

Throughout the course, there may arise a need to install new packages. For now, we will install 'jupyterlab' to initiate a Jupyter Notebook server on your account. Following the activation of your local Conda environment, execute the following command to install 'jupyterlab'.

```
conda install jupyterlab
```

Once 'jupyterlab' has been successfully installed on your server, you can start a Jupyter server by executing the following command.

```
jupyter lab --no-browser --ip=$(hostname)
...
To access the server, open this file in a browser:
    file:///data/homezvol2/panteater/.local/share/jupyter/runtime/jpserver-3512235-open.html
Or copy and paste one of these URLs:
    http://login-i17:8888/?token=xxx
    or http://127.0.0.1:8888/?token=xxx
...
```

This initiates the server on the hostname 'login-i17' (highlighted above) at port 8888 on your server.

Bear in mind that the hostname may vary each time you log in, and the port may change depending on which ports are already in use.

You may not be able to access the server's hostname directly from your machine. In which case, you can connect to the JupyterHub server hosted by HPC3 by executing the following command from your **local machine** (replace the hostname and port accordingly).

```
ssh -N -L 8888:login-i17:8888 panteater@hpc3.rcic.uci.edu
```

This creates an ssh tunnel so that any traffic sent to localhost:8888 on your local machine is forwarded to login-i17:8888. To access the server, visit http://127.0.0.1:8888 on your browser. Enter the provided token (highlighted above) for server access on each occasion. Alternatively, you can establish a password initially, enabling subsequent access without the need for a token each time you connect.

JupyterHub server provides you a simple way to upload, download and edit the files on the server. Also, you can use terminals provided by the server to run your programs.

Part 4 Slurm

HPC3 employs Slurm to execute submitted jobs within the cluster. Jobs fall into two categories: free or allocated. Free jobs may be terminated at any time to release resources for allocated jobs, whereas allocated jobs are assured completion and operate based on the current queue status. Further details about the Slurm workload manager can be found at this link.

Part 4.1 Partitions

There are multiple free and allocated partitions available on HPC3. The complete list of partitions along with the available resources can be found here. For this course, we only need **free-gpu** or **gpu**. **free-gpu** is a free partition with GPU cores that you can use anytime. However, jobs in **free-gpu** might be terminated by the Slurm workload manager to free resources for allocated jobs. **gpu** is an allocated partition, and each student enrolled in this course is assigned **50 hours** for completing the exercises throughout the quarter. If you run out, please let the course staff know.

Important. It is recommended to utilize either free partitions (gpu or cpu) or your local machine for the debugging process, and to use free-gpu as much as possible unless your jobs get terminated, to prevent unnecessary consumption of your allocated GPU quota.

Part 4.2 Submitting a sample Job

Here, we provide a sample bash file that you can use for submitting your jobs to the cluster. Create a file named `submit.sh` and place the following content in the file.

```
#!/bin/bash
#SBATCH -A cs175_class_gpu   ## Account to charge
#SBATCH --time=04:00:00      ## Maximum running time of program
#SBATCH --nodes=1            ## Number of nodes.
                             ## Set to 1 if you are using GPU.
#SBATCH --partition=free-gpu ## Partition name
#SBATCH --mem=8GB            ## Allocated Memory
#SBATCH --cpus-per-task=16   ## Number of CPU cores
#SBATCH --gres=gpu:V100:1    ## Type and the number of GPUs
                             ## Don't change the GPU numbers.
                             ## Follow https://rcic.uci.edu/hpc3/specs.html#specs
                             ## to see all available GPUs.

srun python main.py
```

The bash file above defines the resources required for your program (in this case, `python main.py`). To utilize your quota for allocated jobs on the GPU, you can replace the 'free-gpu' partition with 'gpu' and make sure to set the account to charge to `cs175_class_gpu`.

In cases where your compute-intensive jobs don't use GPU, you can use the `cs175_class` account with the `free`, `standard`, `highmem`, or `hugemem` partitions. The `cs175_class` account has it own quota tracked separately from the `cs175_class_gpu` account.

Use the following `main.py` as an example RL python script:

```
import gymnasium as gym
from stable_baselines3 import A2C

env = gym.make("CartPole-v1", render_mode="rgb_array")
model = A2C("MlpPolicy", env, verbose=1, device="cuda")
model.learn(total_timesteps=10_000)
```

The provided example utilizes the `stable_baselines3` package to learn a policy with Advantage Actor Critic method for the cart pole environment. To install the package, use the following command with the `rl` Conda environment activated:

```
pip install stable_baselines3
```

Then you can submit your job to the cluster by

```
sbatch submit.sh
```

This will submit your job to the cluster, and you can view the program's output in a file named `slurm-<job_id>.out`. To determine if your program utilized a GPU (when using gpu partitions), inspect the first line of the output; it should indicate "Using cuda device."

To view the **status of your jobs**:

```
squeue -u $USER
JOBID PARTITION NAME USER ACCOUNT ST TIME CPUS NODE NODELIST(REASON)
26788573 gpu submit.sh panteater class_cs27 PD 0:00 1 1 (Resources)
```

If you want a live display of your current jobs, you can pair `squeue` with `watch` to rerun the command every few seconds:

```
watch squeue -u $USER
```

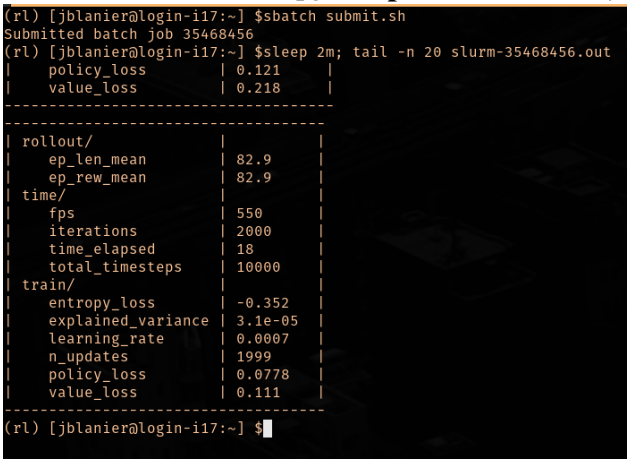
To **stop a job**:

```
scancel <jobid>
```

For a complete guide to Slurm commands you can visit here.

Part 5 Exercise Submission

To submit this assignment and receive credit for it, on Canvas, submit a .jpg or .png screenshot of your terminal showing both your username in the shell prompt and the output for the last iteration of the `main.py` script like below. 10,000 total timesteps should have been collected:



Part 6 Visual Studio Code (Optional)

Connecting to your server through VSCode is also possible, there are two options for using VSCode to connect to the server and run your code.

- Requesting an interactive node to run programs through VSCode is an option, and detailed instructions for this approach are provided in this link.
- Using VSCode primarily as an editor and file manager to connect to the server. However, to execute programs, you still need to submit the job via ‘sbatch‘ as illustrated in Section Part 4.2. To connect to your server using VSCode, refer to this link, and set the hostname as `panteater@hpc3.rcic.uci.edu` with the password being your UCINetID password. Subsequently, you will be prompted to enter ‘1’ for DUO authentication.

Part 7 Reference

https://rcic.uci.edu/index.html