HPC3 Setup and Usage Guide

CS 277: Control and Reinforcement Learning

1 HPC3

HPC3 is a shared-computing cluster at UCI that you can use for exercises in this course. HPC3 efficiently manages the execution and prioritization of the jobs running on it using a queue system.

2 Getting an Account

To begin your journey with the server, obtaining an account is the initial step. If you have already enrolled in the class, your account should be ready for use, allowing you to proceed to the next step. In case your account is yet to be created, kindly reach out via email to hpc-support@uci.edu. Please include **your name** and your **UCINetID**.

3 Login to the Server

Once your account is established, use **ssh** for establishing a connection to the server. You will need the server name, username, and password. The server name is **hpc3.rcic.uci.edu** and the username and password are your usual UCINetID credentials. Connect to the server by executing the following command (with your username instead of `panteater`)

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

in the terminal (or PuTTY for windows). Note that throughout these instructions we are using 'panteater' for the username and you should replace this with your UCINetID.

4 Server Configuration

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

4.1 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. Get an interactive 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 conda into your environment variables by

   ```
   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.

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=3.10
```

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.

4.2 Installing and starting JupyterHub

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

```
conda install jupyterlab
```

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

```
jupyter lab --no-browser --ip=$(hostname) --port=8989
    ...
    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-i17:8989/?token=xxx
     or http://127.0.0.1:8989/?token=xxx
     ...
```

This initiates the server on the hostname 'login-i17' (highlighted above) at port 8989 on your server. Bear in mind that the hostname may vary each time you log in.

Now, you can connect to the JupyterHub server hosted by HPC3 by executing the following command from your **local machine** (replace the hostname accordingly).

```
ssh -N -L 8989:login-i17:8989 panteater@hpc3.rcic.uci.edu
```

This launches the JupyterHub server on your localhost at port 8989. To access the server, visit http://127.0.0.1:8989 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.

5 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.

5.1 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.

5.2 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 CLASS_CS277_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 this link 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 CLASS_CS277_GPU. For a sample `main.py` file to run you can use the following example.

```
import gymnasium as gym
from stable_baselines3 import A2C

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

The provided example utilizes the `stable_baselines3` package to learn a policy with Advantage Actor Critic method (covered in a later lecture) for the cart pole environment. To install the package, use the following command:

```
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)
```

To **stop a job**:

```
scancel <jobid>
```

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

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 5.2. 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.

7 Reference

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