Kempner FAS RC

By Manos Theodosis

Create an FAS RC account

Sign up for an account here: https://portal.rc.fas.harvard.edu/request/account/new. Demba needs to approve, so ping him to do so. Then you need to reset your password here: https://portal.rc.fas.harvard.edu/p3/pwreset/. Finally, you need to enable 2FA, the instructions are here: https://docs.rc.fas.harvard.edu/kb/openauth/.

Use the FASRC VPN

This is not entirely necessary, but they strongly recommend using their own VPN instead oh the default SEAS VPN. Information is here: https://www.rc.fas.harvard.edu/kb/vpn-setup/.

*Note: you need to use your FASRC username and password **and** the authorization code from the authenticator app.*

Ask to be added to Demba's account

Since you most likely created your account after Demba was appointed to Kempner, you need to email the FAS RC team (rchelp@rc.fas.harvard.edu) to be added to Demba's account in order to access the Kempner cluster. (But try to see if you can access it before emailing them).

Using the cluster

To connect to the cluster, you need to run

```
ssh <username>@login.rc.fas.harvard.edu
```

It will ask for your *FAS RC password* and your 2FA token. After connecting, you are landed at a login node. **DO NOT RUN CODE ON THE LOGIN NODES.** (i.e., do not run `python main.py`.) You need to submit jobs to the cluster as they are using a job scheduler called SLURM. To do that, you need to submit scripts using `sbatch` that will submit your

job to the scheduler. There are multiple servers (or partitions) that you can run things at. For our purposes, we will always run things on the Kempner partitions. There is a list of available partitions and their requirements here: https://docs.rc.fas.harvard.edu/kb/running-jobs/#Slurm_partitions.

The form of the scripts, say `schd.sh`, that we use to schedule the jobs is:

```
#!/bin/bash                     # allows it to be run as a bash script
#SBATCH -c 1                    # number of cores you are requesting
#SBATCH -t 0-00:10              # time in D-HH:MM (no less than 10m)
#SBATCH -p kempner_requeue      # partition (kempner/kempner_requeue)
#SBATCH --mem=100               # memo per node in MB (mostly 4g)
#SBATCH --open-mode=append      # append when writing files
#SBATCH -o hostname_%j.out      # where to save the console output
#SBATCH -e hostname_%j.err      # where to save error messages
#SBATCH --mail-type=FAIL        # email if the job failed
#SBATCH --account=kempner_ba_lab # use demba's resources

<code to run>
```

where `%j` is replaced with the job ID, for note-keeping. Note that the memory is per CPU, so if you request more CPUs, that will be split between them. If you want to specify the number of CPUs, or you need GPU access as well, you can add to your file:

```
#SBATCH -n 1                    # number of cpus you are requesting
#SBATCH --gres=gpu:1            # request gpus and how many
```

Finally, if you want to run something interactive or test things, you can use

`salloc -p test --mem 500 -t 0-06:00`

FAS cluster commands

- `spart`: shows you which partitions you have access to (great for making sure you have the correct memberships).
- `sshare --account=ba_lab -a`: shows the usage share of every user.
- `sacct -j JobID`: shows accounting information for completed jobs.
- `squeue -j JobID`: shows accounting information for running jobs.

- `showq -o -p <partition_name>`: see what else is running in the partition.
- `sbatch <your_file>.sh`: submit your code to the queue.
- `sbatch --test-only <your_file>.sh`: will give you an estimated on when your code will run.

Storage

Storage options

link

You need very careful with *where* you store things. In short:

- Every user has 100GB of storage in their home directory that gets backed up. Most likely you can keep code and results there and they should be safe.
- `/n/holyscratch01`: Also accessible via $SCRATCH is a large, high-performance file system. You should use this as your many working area. Make a folder with your name under `/n/holyscratch01/ba_lab`. Note that this gets wiped every 90 days, so do not store files there that are not backed up. It's most efficient for storing data.
- `/scratch`: On-node scratch environment, very large and fast. You can use it for intermediate results that you can then move to permanent storage. However, you **need** to move the results to permanent storage within your job, otherwise they will be lost. This environment is only accessible through the node.

You will be tempted to run everything on your home directory. **DO NOT DO THAT**. FASRC will be really mad at you because this significantly degrades performance for **everyone** in the cluster. If you are adamant about using only one storage location, use `/n/holyscratch01`. The caveats are that your I/O will be slightly slower than if you used `/scratch` for intermediate results and you **need** to move your results to your permanent storage otherwise it will be wiped.

Storage commands

To move files between your local (or another remote) environment and the cluster, you will most likely use `scp`. The format is

```
scp [username@server:][source] [username@server:][destination]
```

To be concrete, the most common examples you will use are transferring data from your local environment to the cluster and vice versa. These look like:

- **Local-to-cluster**
  You need to run the following on your local environment:
  `scp ~/source.py etheodosis@login.rc.fas.harvard.edu:~/`
- **Cluster-to-local**
  You need to run the following on your local environment:
  `scp etheodosis@login.rc.fas.harvard.edu:~/source.py ~/`

You can use the `-r` flag to recursively copy all folders and files in a directory. It is good practice to compress your files before transfer, if possible.

To move files between your home directory in the cluster and the working environments, you will use `rsync` most of the time:

`rsync -avx --progress folder/ /n/holyscratch01/<lab>/folder/`

You can add the flag `-z` to compress the files.

PyTorch on the cluster

Creating environment

It is highly recommended that you create the environment and install packages in your lab space (see the discussion above about storage options). To make sure your environments are create in the correct place you need to add to your `.condarc` and `.mambarc`:

```
envs_dirs:
 - /n/holyscratch01/<lab>/<user>/conda/envs

pkgs_dirs:
 - /n/holyscratch01/<lab>/<user>/conda/pkgs
```

When installing packages, use the `--prefix` option, i.e., `mamba install -y --prefix=/n/holyscratch01/lab/user/conda/envs/ENV_NAME PACKAGE_LIST` if you want to do the installation outside of the environment.

To run PyTorch on the cluster there are a few extra steps you need to take:

- Load the Python module: `module load python/3.10.X-fasrc01`
- Create a Mamba (Anaconda) environment:
  `mamba create -n ENV_NAME PACKAGE_LIST` (`-n` specifies the name)
- You can activate the environment with `source activate ENV_NAME`.
- You can install more packages using `mamba install -y PACKAGE_LIST` (the `-y` flag suppresses confirmation prompts).
- To deactivate, simply do `mambda deactivate`.

It will make your life significantly easier if you are keeping track of your code's requirements. To save your requirements:

`mamba list --export > requirements.txt`

To install the requirements, when creating the environment do:

`mamba create -n ENV_NAME --file requirements.txt` (you can use a `.yml` file)

To see a list of your environments do `mamba env list`. To export an environment (to share) do

`mamba ENV_NAME export > environment.yml`

To remove an environment do `mamba remove -n ENV_NAME --all`.

Submitting a PyTorch job

Assuming everything has been done correctly up to here, to submit a training script:

```
#!/bin/bash
#SBATCH -c 1
#SBATCH -t 0-12:00
#SBATCH -p kempner_requeue
#SBATCH --mem=12g
#SBATCH -n 4 # if you use workers, this NEEDS to be the same number
#SBATCH --gres=gpu:1
#SBATCH --open-mode=append
#SBATCH -o hostname_%j.out
#SBATCH -e hostname_%j.err
```

```
#SBATCH --mail-type=FAIL
#SBATCH --account=kempner_ba_lab

# copy files and move to the correct directory
rsync -avx ~/folder/ /n/holyscratch01/<lab>/<user>/folder/
cd /n/holyscratch01/<lab>/<user>/folder/

# setup the environment
module load python/3.10.X-fasrc01
mamba activate ENV_NAME

# run training
python train.py

mamba deactivate

# your program should be saving things to /scratch/
rsync -avx /n/holyscratch01/<lab>/<user>/folder/results/ ~/folder/results/
```

Helpful resources

For more information, read the quickstart guide (https://docs.rc.fas.harvard.edu/kb/quickstart-guide/) and, most importantly, the guide about running jobs on the cluster (https://docs.rc.fas.harvard.edu/kb/running-jobs/).

For extra resources, check the following links regarding proper etiquette, transferring data, and not hogging the cluster resources.

- https://docs.rc.fas.harvard.edu/kb/responsibilities/
- https://docs.rc.fas.harvard.edu/kb/transferring-data-on-the-cluster/
- https://docs.rc.fas.harvard.edu/kb/kempner-partitions/
- https://docs.rc.fas.harvard.edu/kb/fairshare/