Introduction

The CS 211 GPU cluster is an HPC (High Performance Computing) cluster using a SLURM scheduler, similar to many other computing facilities at universities and supercomputing centers. The remainder of this documentation provides information on how to access the cluster, the cluster environment, how to run jobs on it, and how to install software on it.

Accessing the Cluster

Logging in to the cluster for the first time require the following steps:

(1) Ensure that you have a CS account. If you already have a CS account, you can skip this step. Otherwise, you can follow the instructions on Obtaining an Account.

(2) Once you have obtained a CS account, log in to bolt.cs.ucr.edu. You can do so via either SSH or X2Go.

(3) On bolt.cs.ucr.edu, open a terminal and run:

```
rssh hpc-001
```

This command will automatically log you in to the head node of the cluster using a pregenerated SSH key specific to your account. (Note: if there are any issues at this stage, it could be due to you adding the course a bit late and your access to the cluster has not been set up - in that case email systems@cs.ucr.edu regarding the issue.)

Note: there is an Appendix on setting up SSH keys and using Visual Studio code to access the cluster. We recommend following it once you have verified that you have access to the cluster and can run basic commands on it.

The Cluster Environment

The cluster is an HPC (High Performance Computing) cluster running Rocky Linux 8. Rocky Linux 8 is a clone of enterprise Linux. You log in to the head node of the cluster, and run commands to see the status of the cluster and schedule jobs to run your code. The jobs are run on internal compute nodes. They are not run on the head node, it is intended as the gateway to the cluster.

The jobs can be either interactive (meaning you log in directly to the node that you want to run code on interactively) or batch (meaning you specify a job and run it asynchronously) In HPC computing, interactive jobs are typically used to test and debug and get code working, and

batch jobs are used to run the final version of the code with more resources, in a way where you don't have to continue to pay attention to it.

Typical commands to interact with the cluster:

```
sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
batch*       up    1:00:00      5   idle cluster-001-compute-[001-005]
```

Compute nodes are organized into partitions with a number of identically configured nodes in each partition. The `sinfo` command shows information about the partitions on the cluster.

The default partition (denoted by the * after the partition name) is batch, there are 5 compute nodes. Their names are cluster-001-compute-001 through cluster-001-compute-005.

Example Jobs

Each example job shows how to use the cluster for some specific purpose. They don't take long to run and produce deterministic results as long as the cluster is working properly.

Batch Jobs

Batch jobs are stored in script files. The format of the file includes some headers which are directives to the scheduler, followed by the commands that should run. Individual jobs are submitted to the scheduler using the `sbatch` command. If resources are available to run the job immediately, it will run. If there are no resources available, the job will be queued until there are resources available.

Note that the `sbatch` command does not launch tasks; it requests an allocation of resources such as GPUs and memory and then submits an associated batch script. Directives put in as comments at the top of the script advise the Slurm controller what resources the batch script might need so that it can provide sufficient resources.

The batch script itself can contain any valid commands, *including* calls to other Slurm-related executables such as srun (for running commands dynamically).

Example: Batch Script - Bash Commands

First, edit a file. Name it bash_commands.sh.

```
#!/bin/bash -l

#SBATCH --nodes=1 # Allocate *at least* 1 node to this job.
#SBATCH --ntasks=1 # Allocate *at most* 1 task for job steps in the job
#SBATCH --cpus-per-task=1 # Each task needs only one CPU
#SBATCH --mem=12G # This particular job won't need much memory
#SBATCH --time=0-00:20:00  # 20 minutes
#SBATCH --job-name="batch job test"
#SBATCH -p batch # You could pick other partitions for other jobs
#SBATCH --wait-all-nodes=1  # Run once all resources are available
#SBATCH --output=output_%j-%N.txt # logging per job and per host in the
current directory. Both stdout and stderr are logged.

# Place any commands you want to run below
hostname
date
lsof
```

Now, run the job as a terminal command:.

```
sbatch -p batch -t 00:10:00 bash_commands.sh
-p argument is for the partition of your choice
-t argument is for the amount of time you would like to allocate
bash_commands.sh is the name of the script you are running. It could
be named something other than bash_commands.sh
```

Each job will have a unique job identifier to keep track of it, called the JOBID. This job will automatically run on the first available node. NOTE: all the files on your home directory on the head node are also available on all of the compute nodes; any change on any node will reflect on all nodes.

You can check whether the job has been completed by running squeue:

```
squeue -u $USER
```

There will be no remaining jobs in your queue once the job has completed.
Based on the relevant SBATCH directive from the example, the standard output and standard error from the job will be stored in a file in the current directory named in the format `output_%j-%N.txt`, where %j is the JOBID and %N is the host. This file can be opened to see the output from the job.

If the job has not been completed, it may actually be queued due to the current lack of resources. In that case, you can obtain more information about when it will start by running:
`squeue --start -u $USER`

Another reason for a job not completing might be if you specified unavailable resources using SBATCH directives in your batch file or in command line options provided to sbatch. This would never happen with the example above, but it might if you are copying and pasting SBATCH directives you found somewhere on the internet. For example, a SBATCH directive that requested to use 9 nodes on an 8-node cluster means that the associated job will simply never run. The job would need to be canceled using scancel:
`scancel **job_id**`

Interactive Jobs

Interactive jobs open a shell on a node where you can test your code. We don't recommend doing this in CS 211 unless your TA specifically asks you to do so.

The following command runs a shell on a remote host, using the batch partition:

```
srun -p batch --mem=12g --time=0:30:00 --pty bash -l
```

Using Scontrol to Monitor Jobs

If a job is taking a surprising amount of time, you can monitor the job using scontrol:
`scontrol show JOBID **job_id**`

Modules

If you are coming from other HPC clusters, you might be used to using modules to load in specific libraries. There is a set of modules available for the course, consult your TA as to the details.

Installing Your Own Software

You can install any software that you want in your home directory. Since your home directory is available both on the cluster's head node and also on all of the compute nodes used to run jobs, the software would be available everywhere you needed it.

Obtaining Support

Your TA will provide support for any issues that are specific to CS 211. The IT staff will provide support for anything that's not working on the cluster, that needs to be solved at a system level. If in doubt whether the issue is a system issue or a course related issue, please email both the TA and the IT staff since it may require everyone working together to determine what the issue is and resolve it.

If you do email either the TA or the IT staff, please provide a clear description of the issue so that they can assist you. The email to use for the TA is provided as part of your course material. The email for the IT staff is systems@cs.ucr.edu. This will reach all administrators of the cluster. The sorts of things that the IT staff might assist with:

- Issues with not having an account to access the cluster.
- Issues related to jobs not running or otherwise having trouble.
- Operating system issues on the cluster, for example slow I/O performance.
- Installation of software that is needed by all students in the course but is not installed yet.

Appendix A: Remote Access Using Visual Studio Code

The following instructions will set it up so that you can access the cluster using Visual Studio Code on your computer. If you want to use them, please read and follow them carefully.

(A) Create an SSH keypair on your remote computer that you use to connect to bolt. If you have already done this, you can skip ahead to (B).

The following command will work on Linux and Mac, and it will work on Windows 10 or 11 if the OpenSSH Client feature is installed (this is the default on the newest versions of Windows, otherwise you would need to install it (typically via **Settings** -> **Apps & features** -> **Manage optional features** -> **Add a feature** -> **OpenSSH Clien**t)

ssh-keygen -t rsa -b 4096

When you run this command, accept all defaults and do not change the location of the key. Leave the keys in the location they were generated, with the default names they were created with. This is so that SSH will use them automatically.

(B) Append the contents of the public key in **~/.ssh/id_rsa.pub** that you just created on your remote computer to **~/.ssh/authorized_keys** on the servers you want to connect to:
bolt.cs.ucr.edu
hpc-001.cs.ucr.edu

You should be able to modify **authorized_keys** in bolt.cs.ucr.edu using the VSCode's text editor. However, you may have to use Vim to make modifications in hpc-001.cs.ucr.edu.

1. Use Vim to open **authorized_keys.**
   vim ~/.ssh/authorized_keys
2. If you make some mistakes in the following steps and don't know how to fix them, press **ESC** several times, and then type **:q!** to force quit vim without saving.
3. Copy the contents of **id_rsa.pub.**
4. Press **ESC** to get in the normal mode.
5. Use arrow keys to move the cursor to the place you want to paste (usually at the end of the current content).
6. Use **Shift+Insert** to paste.
7. Use **:wq** to save and quit Vim.

After you've done this, you should use **cat ~/.ssh/authorized_keys** to see if the operation is successful, and guarantee that the new public key is pasted completely.

If the **~/.ssh** directory does not exist in your account on bolt, then create it as a part of this process.

After you have done so, run this command on each server after you have done so in order to set permissions correctly.

chmod -Rv 0700 ~/.ssh

This restricts permission on **~/.ssh** and its contents to only your user, and is what is needed in order to ensure that key based SSH access will work.

(C) Test that key based access is working by running the following command on your computer:

ssh -J USERNAME@bolt.cs.ucr.edu USERNAME@hpc-001.cs.ucr.edu

where **USERNAME** is your CS username. This should log you in to hpc-001 if all of the steps have been followed successfully.

(D) Assuming that you have (a) Visual Studio Code installed on your computer and (b) Have the Remote SSH extension installed in Visual Studio Code, then you should be able to edit your SSH configuration file on your computer to use bolt as a ProxyHost to access hpc-001:
(a) Run Visual Studio Code.
(b) Type F1 to bring up the command palette and select **Remote SSH: Open SSH Configuration File...** In that file, add these lines, modify them to replace **YOUR_CS_USERNAME** with your UCR NetID, save the file, and then close it.

Host bolt
HostName bolt.cs.ucr.edu
User YOUR_CS_USERNAME

Host hpc-001
HostName hpc-001.cs.ucr.edu
User YOUR_CS_USERNAME
ProxyJump bolt

If the commands in (C) above worked, then this should as well. You should be able to select hpc-001 as a remote SSH host in Visual Studio Code and access it that way.

You will be able to open a remote terminal in Visual Studio Code on hpc-001 and run batch and interactive jobs from that terminal.

You may find that the ‘File Explorer’ or ‘File Manager’ is unavailable in VSCode after you get connected. You can click ‘Open Folder’ and select or type the directory path you want to access. The VSCode will reload the page and the ‘File Explorer’ should be showing files in the specified directory.