# Using the HPCC

Prepared by Chris Fietkiewicz

November 28, 2018

## Contents



## 1. Introduction

This document provides a brief introductory tutorial for using the high performance computing cluster (HPCC) at Case. Main website: https://sites.google.com/a/case.edu/hpcc/. You will SSH to the HPCC using *ID*@rider.case.edu, where *ID* is your Case network ID. This tutorial assumes that you are familiar with using SSH and transferring files using SFTP. On Mac and Linux, you can use the Terminal application with the "ssh" and "scp" commands. On Windows, you can use PuTTY as a terminal and WinSCP or Filezilla.

You will use a job management system called *SLURM* (Simple Linux Utility for Resource Management). Information on using SLURM is available here:

https://sites.google.com/a/case.edu/hpcc/jobs
https://computing.llnl.gov/linux/slurm/overview.html

## 2. Executing a program in "interactive" mode

**IMPORTANT: All programs should be run on compute nodes and not on the head node.** This is because all users interact with the head node, and you will be sharing it with them. To use a compute node interactively, use the "srun" command. The following example requests a compute node with 4 cores for 1 hour. Type the following after you have SSH'd into the head node as described above:

```
srun -c 4 --time=1:00:00 --pty /bin/bash
```

Various options are described here:
https://sites.google.com/a/case.edu/hpcc/jobs/interactive-session

Note that you can omit "-c 4" to get a single core by default. You can also omit "--time=1:00:00", and it will request 10 hours by default. There are tradeoffs regarding the requested time. Longer times may cause you to sit in the queue longer if the system is really busy. However, your session will be terminated *without warning* when your time has expired, which will terminate any programs or commands that are running. So if you request a time, be sure that you are ready to be kicked off at that exact time. Any files you saved will still be available back on the head node.

## 3. Executing a program in "batch" mode

General work such as simulations should always be done on compute nodes and not on the head node. This is because all users interact with the head node, and you will be sharing it with them. Time consuming tasks should be done on a compute node.

The basic process is to submit a script to the job manager. The execution of the script is called a *job*, and the head node will take care of assigning each job to an available compute node to be executed. This tutorial describes how to use job management system called SLURM (Simple Linux Utility for Resource Management). By convention, the name of job script will end with the extension .slurm. A job will be submitted using the "sbatch" command. (Note that the HPCC website mostly has documentation on using a different job management system, the commands for which may also work on the SLURM cluster using wrappers.) There is also a website for SLURM:

https://computing.llnl.gov/linux/slurm/overview.html

The following instructions demonstrate a simple process of running the program "hello.c" using a SLURM script "hello.slurm". These files are provided with this document.

A. The basic approach to running a program is to have a .slurm script. This script is executed on a compute node using the syntax "sbatch *scriptname*.slurm". For this demo, you will use the provided file named "hello.slurm". Transfer the two files to the HPCC.

B. Here we will discuss the lines in this file.
   1. The following line determines which shell the script should be run in:
      ```
      #!/bin/bash
      ```

   2. The following line specifies that we want 1 node, which essentially means 1 CPU. If you are not doing any special message passing, such as MPI, you can omit this line. However, it's provided here as an example.
      ```
      #SBATCH --nodes=1
      ```

   3. The following line requests 4 cores. Don't be fooled by the fact that it says "cpus". In this case, the "cpu" is a single core.
      ```
      #SBATCH --cpus-per-task=4
      ```

   4. The following line requests 5 minutes of compute time.

```
#SBATCH --time=0-00:05:00
```

If this line is omitted, you will get 10 hours by default. As with an interactive session (see previous section), there are tradeoffs regarding the requested time. Longer times may delay your job if the system is really busy, but jobs that expire will have their active programs immediately terminated! Therefore, be sure to request a safe amount of time.

5. The following line defines a file for console output:
   ```
   #SBATCH --output=my_output.txt
   ```

6. In the following lines, the script copies the program from the head node to the compute node and then changes the current directory to the compute node:
   ```
   cp hello.c $PFSDIR/.
   cd $PFSDIR
   ```

7. The following lines compile the program on the compute node and execute:
   ```
   gcc hello.c -o hello
   ./hello
   ```

C. Submit the job as follows:

```
sbatch hello.slurm
```

NOTE: Files created on non-Linux computers (e.g. Windows) may have incompatible linebreaks. If this occurs, you will see the following error:

```
sbatch: error: Batch script contains DOS line breaks (\r\n)
sbatch: error: instead of expected UNIX line breaks (\n).
```

This can be corrected by using the "dos2unix" command as follows:

```
$ dos2unix hello.slurm
dos2unix: converting file hello.slurm to UNIX format ...
```

D. Your job should finish quickly. List the directory contents, and you should see a new file named "my_output.txt". The script defined this as the filename for saving console output. Open this file to see your program output.

E. Note that console output is different than file output. If your program creates additional output files, you need to copy them from the compute node to the head node. Otherwise, they will be lost. If you need to do this, use the path "$SLURM_SUBMIT_DIR/." for the head node. The following is an example that could be added to the .slurm script to copy *my_file* from the compute node to the head node (note that "-u" is the *update* option for "cp"):

```
cp -u my_file $SLURM_SUBMIT_DIR/.
```

## 4. Managing jobs

When a job is running, you can check the statuses using the "squeue" command. You can actually get a list of *all* jobs running for *all* users as follows:

```
% squeue
```

To see only your own jobs, use the "-u" option to specify a particular user as follows:

```
% squeue -u networkID
```

Where *networkID* is your Case network ID (username such as "abc123"). Note that using the up-arrow key will recall the previous command, so you can also use that approach instead of retyping the command. List your own job as explained above. If you want to delete one of your jobs, this can be done using the "scancel" command as shown below:

```
% scancel jobID
```

Where *jobID* is the ID number for your job that appears in the first column displayed by the "squeue" command. An example of where you would delete a job is a program that obviously is running too long due to an error.

For long jobs, there are a couple of ways to conveniently monitor when they are finished. The online documentation shows how to schedule an email alert. Another approach is to use the Linux command "watch" to rerun the squeue command every second as follows:

```
% watch squeue -u networkID
```

Using the approach above, you can occasionally check your screen to see when the job is no longer on the list. Then use CTRL-c to terminate the "watch" command.