Analysis of Next Generation Sequencing Data
Getting to know our systems

Luce Skrabanek

8 January, 2019

1 Network Access

To access the Wifi network, use the following credentials:

```
WiFi SSID: WCMEVENT
UserID:    HPCCLASS
Password:  HPCCLASS2019
```

2 Accessing SCU resources

2.1 Login nodes

The servers that we will be using for this class are hosted and maintained by the Scientific Computing Unit (`https://scu.med.cornell.edu`).

We first log in to a gateway node using `ssh`. These login nodes are monitored by an intrusion prevention software, and users can be temporarily locked out of their accounts if they fail to use the correct password, or permanently if they use an invalid username during `ssh` login.

To avoid the permanent ban, the following can be added to your `~/.ssh/config` file:

```
Host *.med.cornell.edu
user USERNAME
ServerAliveInterval 60
```

If you used the above config, you can login using:

```
ssh aristotle.med.cornell.edu
```

otherwise use:

```
ssh USERNAME@aristotle.med.cornell.edu
```

There are three login nodes available; you may use any of them.

aristotle.med.cornell.edu
pascal.med.cornell.edu
aphrodite.med.cornell.edu

If there are any issues with your password, go to https://scu.med.cornell.edu/sspr to reset it.

2.2 SSH keys

To effectively use the SCU's infrastructure, you will need to ensure that all the compute nodes have the ability to authenticate you, without the need to send your password over the network.

Setting up your ssh keys only needs to be done once, so first check if you don't already have ssh keys set up:

```
ls ~/.ssh
```

If the output shows the files id_rsa and id_rsa.pub, you already have keys in place (your private and public keys, respectively). Skip the following command and continue to authorizing your key.

If the output did not show those files, generate them with the following command:

```
ssh-keygen -t rsa
```

Follow the instructions on screen, and accept the default location.

To authorize your key, add your public key to the list of public keys that can authenticate you.

```
cd ~/.ssh
cat id_rsa.pub >> authorized_keys
```

2.3 Compute nodes

You should never run any jobs on any of the login nodes.

There are two ways to run jobs on this infrastructure:

1. Use the batch queuing system (see below). This should always be the primary choice when running jobs that will take a while.
2. Log in directly to one of the compute nodes reserved for this class.

There are two compute nodes that have been reserved for this class:

buddy.pbtech
farina.pbtech

Navigate to one of these servers using:

```
ssh buddy.pbtech
```


3 Storage space

You have several disk storage locations available to you:

1. Your home directory. This is 100 GB of backed-up storage space. In general, this should be used to store your scripts, any tools or packages that you install yourself, and anything else that you would be very sad to lose. Note that this space is limited, so use it wisely!
2. Scratch space on buddy/farina. We have 2.4 TB of local storage space here, shared between all users. This is accessible via `/scratchLocal`. This should be used for I/O-intensive operations, especially when running batch jobs.
3. Scratch space on athena. We have available a 3TB fileset on `/athena/angsd/scratch`, also shared between all users. This should be used to store datasets and other result files.

4 Installed tools

Many tools that we will be using in this class have already been installed for you by the SCU. These are accessed via `spack`. To see a list of all packages that are available on our systems, use:

```
spack find
```

Note that this command will not work on a gateway node, as those nodes are not meant to run any tools.

Make sure that you can access and use some of the tools that we will be using more frequently. Note that some tools have multiple versions available, and you will have to be explicit about which version to use.

```
spack load -r fastqc
  fastqc --help
spack load subread
  featureCounts --help
spack load -r py-rseqc
  read_duplication.py --help
spack load samtools@1.9
  samtools --help
spack load star
  STAR --help
spack load bedtools2@2.27.0
  bedtools --help
spack load -r py-deeptools
  deeptools --help
spack load bedops
  bedops --help
spack load bamtools
    bamtools --help
spack load -r py-macs2
  macs2 --help
spack load -r r@3.5.1
  R --version
```


5 Batch jobs

Slurm is used as the batch queuing system on the SCU infrastructure. While we can access it directly from our compute nodes, the preferred submit host is `curie.pbtech`.

There is a dedicated queue for this class, called `angsd_class`. To see basic information about this queue:

```
sinfo -p angsd_class
```

5.1 Interactive sessions

This is especially useful if you are debugging and testing code, or if you have a relatively simple work flow. In these cases, we can launch an interactive session.

```
srun -n1 --pty --partition=angsd_class --mem=8G bash -i
```

Explanations for the options in the above example:

- `-n 1, --ntasks=1` The number of concurrently running tasks.
- `--pty` Runs the job in a pseudo-terminal.
- `--partition=angsd_class` Specify the cluster partition you want to access. For this class, we have a dedicated partition called `angsd_class`.
- `--mem=8G` Requests 8G of memory for the job. If you use more memory than what you requested, the job will fail.
- `bash -i` The command to be run. In this case, we are running bash with an interactive terminal.

Since we can log directly to buddy and farina, we do not have to use this approach in this class, but this will be how to start an interactive session on most systems.

Make sure that you log out of an interactive session once you are done with it. Otherwise, you are tieing up valuable resources that could be used by somebody else.

5.2 Batch queuing

To submit jobs to the cluster, you need to be logged in to the submit node for the cluster you're using. For this class, we are using `curie.pbtech`.

hello_slurm.bash

```
#! /bin/bash -l

#SBATCH --partition=angsd_class
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --job-name=hi_slurm
#SBATCH --time=00:05:00    # HH/MM/SS
```



```
#SBATCH --mem=1G                # memory requested, units available: K,M,G,T

echo "Starting at:" `date` >> hello_slurm_output.txt
echo "This is job #:" $SLURM_JOB_ID >> hello_slurm_output.txt
echo "Running on node:" `hostname` >> hello_slurm_output.txt
echo "Running on cluster:" $SLURM_CLUSTER_NAME >> hello_slurm_output.
    txt
echo "This job was assigned the temporary (local) directory:" $TMPDIR
    >> hello_slurm_output.txt
touch ${TMPDIR}/test_file_$SLURM_JOB_ID

sleep 30

exit
```

The shebang line (the first line in the script) indicates the shell that we want to use to run the job. This line is absolutely required. In almost all cases, you will want to use a bash shell, and in most cases, that shell should be a login shell. This will ensure that your environment is consistent. The login shell option is specified with the `-l` flag at the end of the shebang line.

Between the shebang line and the main body of the script (which looks like any other bash script), are a set of lines that begin with `#$`. These are the slurm options that will be passed on to the scheduler. Some of them we have already seen in the interactive command.

- `--nodes=1` The number of nodes requested.
- `--job-name=hi_slurm` The name of the job. This will be used when listing the job in the queue. The name should be kept relatively short, as only the first 10 characters will be shown in many cases. Note that the name of the job is not the name of the script.
- `--time=00:05:00` The maximum amount of time requested for this job to run.
- `--mem=8G` Requests 8G of memory for the job. If you use more memory than what you requested, the job will fail.
- `bash -i` The command to be run. In this case, we are running bash with an interactive terminal.

Additionally, we are using some slurm-specific environmental variables:

- `$SLURM_JOB_ID` The job's ID number, assigned by slurm.
- `$SLURM_CLUSTER_NAME` The name of the cluster used.
- `$TMPDIR` The local temporary directory your job has access to. It's very important for your job performance, as well as cluster stability, that intense I/O (e.g. creating many temporary files) is performed in this temporary directory

To submit this job to the queue:

```
sbatch hello_slurm.bash
```

To monitor job status:

```
squeue -u USERNAME
```


More information about slurm can be found on the SCU wiki at
`https://wcmscu.atlassian.net/wiki/spaces/WIKI/pages/327731/Using+Slurm`
and at
`https://slurm.schedmd.com/pdfs/summary.pdf`

