Blue Waters Usage for CS546 Spring 2018

Many of your projects this semester will require significant computational resources (memory, access to GPUs, etc.). Therefore, students in this class will have access to the Blue Waters computing environment. Running programs on Blue Waters is somewhat different than running code on your local machine, so this document is meant as a quick-start guide to getting yourself acclimated with the Blue Waters environment with as little hassle as possible. For more details on any of the topics discussed here, please see the Blue Waters User Guide [https://bluewaters.ncsa.illinois.edu/user-guide](https://bluewaters.ncsa.illinois.edu/user-guide)

As a general note, we expect you to use the following workflow when running programs on Blue Waters. While there are other ways to run programs on Blue Waters, the goal of providing this workflow is to simplify this process so you can spend more time on your project, rather than on configuring the environment.

1. Code in your own (local) environment
2. Copy your code (and necessary data) to Blue Waters (see Section 2)
3. Test your code in an Interactive Session (see Section 3)
4. When you’re fairly confident your program won’t crash, submit all long-running jobs to the queue (see Section 4)

Blue Waters is organized as a collection of nodes. *Login nodes* are the points at which users access the environment. You may not run applications on login nodes (it’s against Blue Waters policy). Instead, your code will be run on *compute nodes*, either explicitly (via interactive sessions – Section 3) or implicitly, when you queue jobs to run (Section 4).

### 1 Accessing Blue Waters

In order to access Blue Waters, you need to log in to a login node with the login credentials that you’ve been given in class. To do so, open a terminal (while on the University network) and enter the following

```
ssh <user>@bwbay.ncsa.illinois.edu
```

where `<user>` refers to your user name. You’ll then be prompted with your password, and (during your first login) you will be prompted with a few additional questions regarding their Terms of Use Policy.

By default, this will place you in your home directory:

```
/u/training/<user>/
```
which should contain a scratch/ directory. While you should feel free to use this directory for temporary storage, note that the system periodically clears scratch directories, so you shouldn’t keep anything important there.

2 Data Transfer

In order to transfer data to your Blue Waters home directory, you must run the scp command from a login node to a machine running an ssh server. Assuming you are already logged in (Section 1), you can use the following

```bash
scp <remote_user>@<remote_address> <bw_directory>
```

where the `<remote_user>` and `<remote_address>` refer to the machine you’re trying to copy files from, and `<bw_directory>` refers to the location on Blue Waters you’re trying to copy files to.

It is important to note that scp is only permissible with individual files smaller than 1GB or directories smaller than 10GB. If you need files larger than this, please contact the TAs to make special arrangements.

Note that all EWS machines should be accessible from Blue Waters, so at a minimum you can move files to Blue Waters that way. If you prefer, you may be able to install an ssh server on your local machine (e.g. `sudo apt-get install openssh-server` on Linux, for example).

3 Interactive Sessions

Interactive sessions allow you to run your code on a compute node as if it were a regular Linux machine. You should only use interactive sessions for short-running jobs (e.g. testing your code). To start one of these sessions from a login node, you need to use a sequence of commands similar to the following:

```bash
qsub -I -l gres=ccm -l nodes=1:ppn=16:xk -l walltime=01:00:00
module add ccm
cmlogin
```

where `-I` enables interactive mode, `gres=ccm` enables cluster compatibility mode (which sets up the compute node to act like a normal Linux installation), `nodes=1` refers to the number of nodes being requested (typically 1 is sufficient), `ppn=16` refers to the number of cores being requested on the node, `xk` specifies that you want GPU-enabled nodes, and `walltime` specifies the amount of time before the interactive session will time out.

After running the first command, you will be prompted to wait (which can take minutes or hours, depending on how many jobs there are in the queue, etc.). The next two commands will take you from a login node to a compute node where you can run your code.

If you’re using python (and TensorFlow, for that matter), you may also want the following

```bash
module load bwpy
module load cudatoolkit
```

which loads the Blue Waters python libraries (which include TensorFlow) and the CUDA toolkit, which should enable TensorFlow to leverage the GPUs on the node.
4 Submitting Jobs

The process for executing long-running jobs on Blue Waters is as follows

1. Set up a .pbs script that calls a .sh script
2. Specify your python script (and arguments) in the .sh script
3. Run `qsub <script_name>.pbs`, which submits your job to the queue

To make this process easier, we’ve given you some templates – `launch.pbs` and `launch.sh` – based on the Blue Waters examples at [https://github.com/asaxton/ncsa-bluewaters-tensorflow](https://github.com/asaxton/ncsa-bluewaters-tensorflow). Our templates are located at `/projects/training/baoz/`. If you’d like to use these files, please do the following

1. Copy both `launch.pbs` and `launch.sh` to your home directory
2. Modify `launch.sh` to reflect your python script location and arguments (should be line 54)

You can then use `qsub launch.pbs` to enqueue your job, as described above. You may want to change the directory to which your log files are written (currently specified as `~/scratch/`) or the number of nodes / etc.; these arguments are all in the `launch.pbs` script.

Once your job is submitted to the queue, you can check on its status with

```bash
qstat | grep <user>
```

where `<user>` is your Blue Waters user name. The code next to your job should either be Q or R for enqueued and running, respectively.

If you’d like to kill a job that you’ve submitted, run

```bash
qdel <job_id>.bw
```

where you can find the job ID with `qstat` (and it should’ve printed when you ran `qsub`).