Introduction to Computing on an HPC System

MATH1900: Machine Learning

Location: http://people.sc.fsu.edu/~jburkardt/classes/ml_2019/hpc_intro/hpc_intro.pdf

BRIDGES
NEXT-GENERATION SUPERCOMPUTING
FOR NEW COMMUNITIES AND BIG DATA

BRIDGES is the main system at the Pittsburgh Supercomputing Center (PSC).

HPC Basics

How do I create an account at the Pittsburgh Supercomputing Center, log in, transfer files, and do some simple tasks?

1 Overview

Around 1985, the National Science Foundation (NSF) set up several supercomputing centers across the country. Individual universities could no longer afford the cost of buying, installing, running, and upgrading such systems. These new centers were meant to offer, at no cost, the best computing facilities for researchers and students. There are now four such centers, NCSA (Urbana, Illinois), PSC (Pittsburgh), SDSC (San Diego), and TACC (Austin, Texas).

To do some interesting exercises with neural networks, we will want to use the keras library on the graphical processing units (GPUs) that are available at the PSC. The GPUs are one part of a large computer cluster of more than 800 separate machines, known as *Bridges.*

Professor Schneier has been authorized to create accounts on Bridges for every student in this class. In this lab, we will go over the steps necessary to get permission from XSEDE, to set up your PSC account, to connect from your computer to Bridges, to transfer a file from your personal computer to Bridges, to understand the Unix operating system used on Bridges, to run some small, simple jobs as tests, and to transfer a result file back to your personal computer.

Parts or all of this lab exercise may fail for you. Today, that's OK. I will try to get through all the activities on my own; we will try to identify the problems people are having and work them out before our later lab, where we will try to do the GPU computing.

The PSC web page is

```
https://www.psc.edu/
```

and you will find information about Bridges under

`RESOURCES / Computing / Bridges`

2 Create an XSEDE account

The NSF supercomputing centers are administered by an agency called XSEDE. In order to get access to any NSF center, you must first set up an XSEDE account. This is done by going to the website

`portal.xsede.org`

Under the menu item **MY XSEDE** there is an option **accounts**. On the accounts page, there is an option **create account**. Choosing that option takes you to the new account form, which will gather some information from you and then allow you to set up your account. For your XSEDE identifier or username, you should specify your Pitt email ID, such as **joeuser25**. You will also need to specify an XSEDE password.

Once you have created an XSEDE account, you need to notify professor Schneier, so that he can create a PSC account for you, using the same XSEDE identifier as your username.

3 Your PSC Account

Access to your PSC account requires specifying a username and password. The username will be the same as your XSEDE identifier (that is, probably your Pitt email ID). The PSC password must be created by going to the PSC password change utility at:

`apr.psc.edu`

although you might have to specify the full address:

`https://apr.psc.edu/autopwdreset/autopwdreset.html`

Shortly after your password is set up, you will be able to access your PSC account. You will typically do this from your laptop machine, running a program that allows you to issue commands interactively to a PSC system, or to transfer files back and forth.

Everyone in this class belongs to the same group. The group name is

`ac5pibp`

For some activities on Bridges, it may be necessary to specify this group name.

4 Interactive access to Bridges

From a Mac or Linux system, you can use the `ssh` program to connect to your PSC account:

```
ssh joeuser25@bridges.psc.xsede.org

--- Interactive session with Bridges ---

logout
```

If we are doing anything involving graphics on Bridges, then the command should be slightly modified:

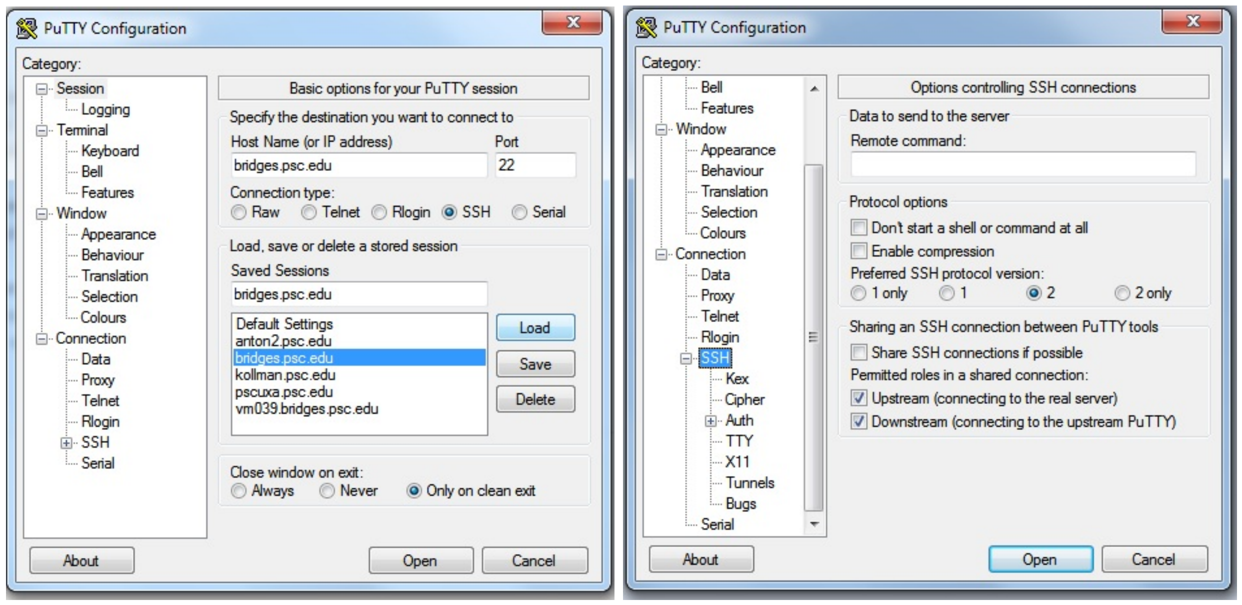
```
ssh -X joeuser25@bridges.psc.xsede.org
```

On a Windows 10 system, if you have set up WSL (Windows Subsystem for Linux), then you can essentially open a Linux environment from the Start menu, and you will have access to the `ssh` command and other Linux commands to be discussed later.

Otherwise, your best option may be to install the `putty` program, available from

`https://www.chiark.greenend.org.uk/~sgtatham/putty/`

In that case, here is a version of the settings that would be useful:



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): bridges.psc.edu

Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: bridges.psc.edu

Default Settings
anton2.psc.edu
bridges.psc.edu
kollman.psc.edu
pscuxa.psc.edu
vm039.bridges.psc.edu

Close window on exit: Always Never [✓ Only on clean exit]

Options controlling SSH connections

Data to send to the server

Remote command:

Protocol options

☐ Don't start a shell or command at all

☐ Enable compression

Preferred SSH protocol version: 1 only 1 [✓ 2] 2 only

Sharing an SSH connection between PuTTY tools

☐ Share SSH connections if possible

Permitted roles in a shared connection:

☑ Upstream (connecting to the real server)

☑ Downstream (connecting to the upstream PuTTY)

PUTTY configuration and SSH settings for PSC.

You will have to type `bridges.psc.edu` in the `Host name` field, and you should make sure that the `Port` field is set to 22.

Try to connect to your account on Bridges now! I assume some, or many, of you will have some trouble doing this.

5 Unix commands:

Bridges uses a version of the Unix operating system called Linux.

If you are unfamiliar with Unix or Linux systems, there are a few things to know that will make life bearable. When you log in, you are placed in your home directory. Your directory will contain files. To organize your work, you can create subdirectories that contain all the files associated with a particular task.

Some useful commands include:

- **ls**: lists the files in the current directory;
- **mkdir** *name* creates a subdirectory called *name*;
- **cd** *name* moves to the subdirectory called *name*;
- **cd** .. (two dots) moves "up" to the directory above this one;
- **cd** moves back to your home directory;
- **pwd** prints the name of your current directory;

- **more** *name* prints out the file *name*;
- **rm** *name* deletes the file *name*;
- **cp** *current new* makes a copy of a file;
- **mv** *current new* moves a file from one place to another;
- **display** *filename* displays a graphic file;
- **logout** logs you off the system;

Right now, if you type `ls`, you should probably see nothing - you have no files. That will soon change.

6 Exercise: Use the nano editor

To create, modify, or view a file on Bridges, you need an editor. We will look at using the `nano` editor to modify a file that customizes your environment. The file has the peculiar name **.bashrc** - note the initial period! - and it is a "hidden file" (because of that initial period) in your home directory.

Invoke the editor with the command:

```
nano .bashrc
```

You should see something like this, the text of your file, and a few lines at the bottom that are a menu of commands. Each command involves the control key and a letter. The symbol **^G** for *Get Help* indicates that the CTRL ("Control") key should be pressed down along with the letter G.

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
  . /etc/bashrc
fi

# Uncomment the following line if you don't like systemctl's auto-paging feature
:
# export SYSTEMD_PAGER=

# User specific aliases and functions


^G Get Help   ^O WriteOut   ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is    ^V Next Page   ^U Uncut Test  ^T To Spell
```

Use the arrow keys to navigate to just after the last line of text and insert this line:

```
# User specific aliases and functions
PS1="PSC \h: "
```

Then invoke the `^O` command to write the file out, and the `^X` command to exit from `nano`.

In order to make the change effective immediately, type

```
source .bashrc
```

Now your prompt string should be something like

```
PSC br018:
```

which will let you know you're talking to the PSC, and which PSC node you are currently logged in on.

You can use `nano` to view or modify any text file on the PSC system.

7 Exercise: Create and run hello.py

Now we will create a new file. Start `nano` and you will see a blank screen. Type the following text:

```
print ( "Hello, Bridges!" )
```

Then invoke the `^O` command to write the file out as *hello.py*, and the `^X` command to exit from `nano`.

You should now be able to run your 1-line program by typing

```
python hello.py
```

You can save the output of the program using the `>` symbol:

```
python hello.py > hello.txt
```

Typing the **ls** command should now display at least these two files in your directory.

We have just run a very small program on the front end or login node. In general, you should not run programs here! The login nodes are shared by many users, who will be disrupted if you run a big program. After this first small test, your class programs should be run on a dedicated node, using the `interact` command.

Users who run large jobs on the front end may have their account frozen by system administrators. All the remaining tasks for this class will be done on computational nodes, using the `interact` command, or else, possibly, through the non-interactive batch system.

8 The module command

PSC has many libraries of software, whose availability is controlled by the **module** command, which you can think of as "checking out" copies of the software you specify. When you log in, just a bare minimum of this software is automatically checked out and available to you. To see the current list, type:

```
module list
```

To see a list of all the available software (there is a lot!) type

```
module avail
```

To see a list of the available versions of `python`:

```
module avail python
```

To actually make a software library available, use the `load` option. Usually, the only library we will ask for is `keras`, but we will get it as part of the `anaconda` release:

```
module load anaconda3/2019.03
```

When you load `anaconda`, you actually get a bunch of libraries To see what you've just checked out, try again:

```
module list
```

If, for some reason, you don't want to have a particular software library loaded anymore, you can `unload` it:

```
module unload anaconda3/2019.03
```

9 File transfers to and from Bridges

From a Mac or Linux system, you can use the `sftp` program to transfer files to and from Bridges. This command may also be available to Windows users via the PowerShell. If you are already using `ssh` to connect to Bridges, you will need to open a separate terminal window on your home system in which to start `sftp`.

Note that `sftp` will require a different address than that used by `ssh`. Once you start `sftp`, you can use `put` and `get` commands to move files.

```
 ---  Start this command on your home system! ---

 sftp joeuser25@data.bridges.psc.edu
 put montana.py      <-- copies montana.py FROM  your system  TO  Bridges
 get hello.txt       <-- copies hello.txt  FROM  Bridges      TO  your system
 quit
```

- `sftp` *user@address* connects your local machine to the remote machine, starting in the user's remote home directory;
- `put` *filename* will transfer a copy of the local file to the remote directory;
- `get` *filename* will transfer a copy of the remote file to the local directory;
- `quit` terminates the `sftp` session;

You can also use a limited number of unix commands. The following commands refer to the remote system, which in this example would be Bridges:

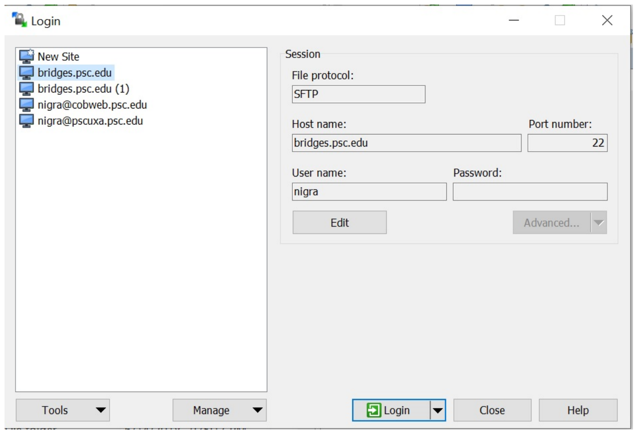
- `ls` to list the files in the remote directory;
- `pwd` to report the remote directory;
- `cd` *directoryname*, to change to a new directory;

Using an **l** prefix, (for "local"), you can also issue commands that refer to your own system:

- `lls` to list the files in the local directory;
- `lpwd` to report the local directory;
- `lcd` *directoryname*, to change to a new directory;

For `putty` users, I don't know the details of file transfer. I believe that `putty` comes with a program `psftp` which may run from the command line similarly to `sftp`.

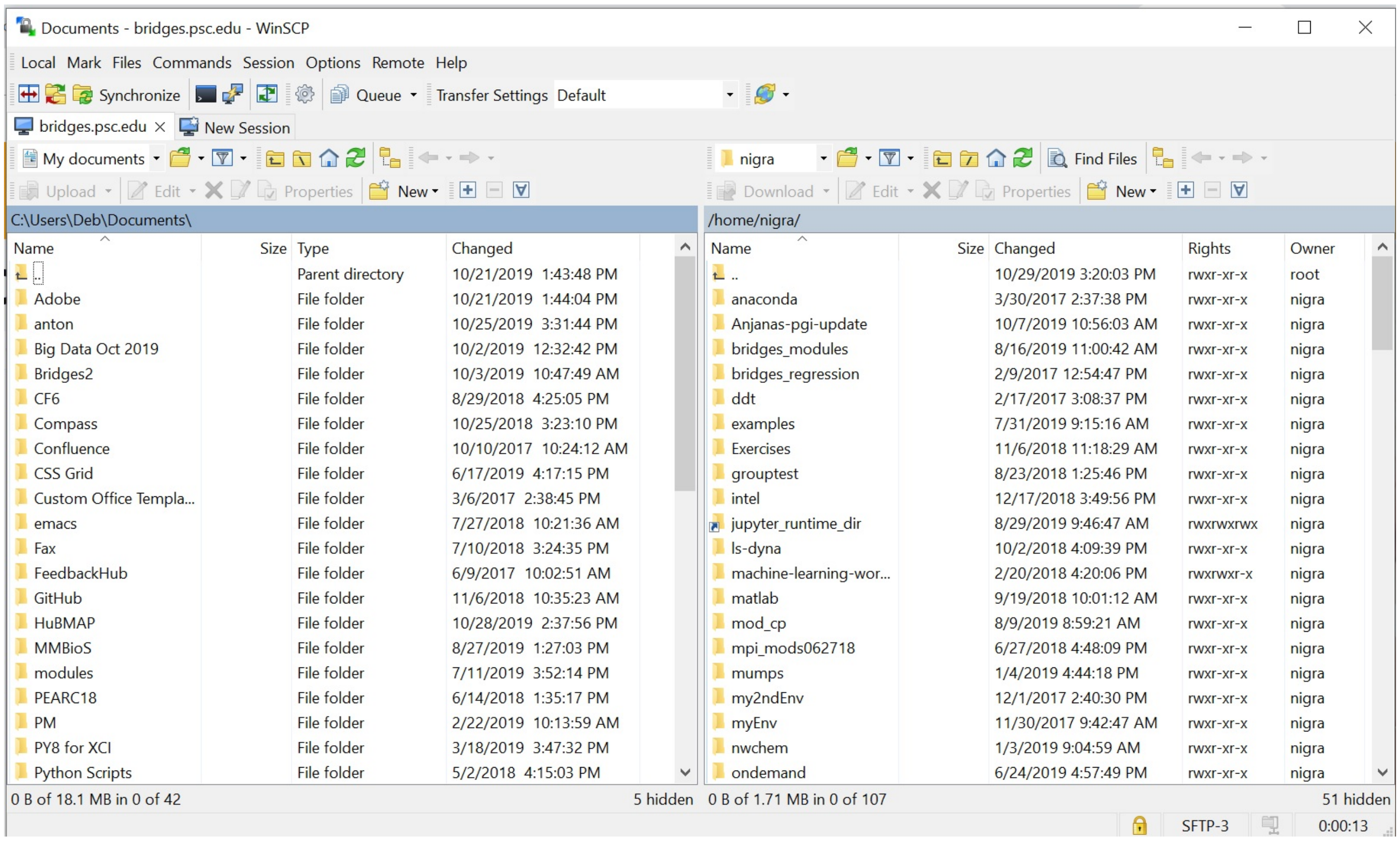
Alternatively, there is a Windows program called WinSCP which gives you a visual interface showing your local and remote directories



New Site
bridges.psc.edu
bridges.psc.edu (1)
nigra@cobweb.psc.edu
nigra@pscuxa.psc.edu

Session
File protocol: SFTP
Host name: bridges.psc.edu
Port number: 22
User name: nigra
Password:

WinSCP configuration for PSC.



C:\Users\Deb\Documents\

Name	Size	Type	Changed
..		Parent directory	10/21/2019 1:43:48 PM
Adobe		File folder	10/21/2019 1:44:04 PM
anton		File folder	10/25/2019 3:31:44 PM
Big Data Oct 2019		File folder	10/2/2019 12:32:42 PM
Bridges2		File folder	10/3/2019 10:47:49 AM
CF6		File folder	8/29/2018 4:25:05 PM
Compass		File folder	10/25/2018 3:23:10 PM
Confluence		File folder	10/10/2017 10:24:12 AM
CSS Grid		File folder	6/17/2019 4:17:15 PM
Custom Office Templa...		File folder	3/6/2017 2:38:45 PM
emacs		File folder	7/27/2018 10:21:36 AM
Fax		File folder	7/10/2018 3:24:35 PM
FeedbackHub		File folder	6/9/2017 10:02:51 AM
GitHub		File folder	11/6/2018 10:35:23 AM
HuBMAP		File folder	10/28/2019 2:37:56 PM
MMBioS		File folder	8/27/2019 1:27:03 PM
modules		File folder	7/11/2019 3:52:14 PM
PEARC18		File folder	6/14/2018 1:35:17 PM
PM		File folder	2/22/2019 10:13:59 AM
PY8 for XCI		File folder	3/18/2019 3:47:32 PM
Python Scripts		File folder	5/2/2018 4:15:03 PM

0 B of 18.1 MB in 0 of 42 5 hidden

/home/nigra/

Name	Size	Changed	Rights	Owner
..		10/29/2019 3:20:03 PM	rwxr-xr-x	root
anaconda		3/30/2017 2:37:38 PM	rwxr-xr-x	nigra
Anjanas-pgi-update		10/7/2019 10:56:03 AM	rwxr-xr-x	nigra
bridges_modules		8/16/2019 11:00:42 AM	rwxr-xr-x	nigra
bridges_regression		2/9/2017 12:54:47 PM	rwxr-xr-x	nigra
ddt		2/17/2017 3:08:37 PM	rwxr-xr-x	nigra
examples		7/31/2019 9:15:16 AM	rwxr-xr-x	nigra
Exercises		11/6/2018 11:18:29 AM	rwxr-xr-x	nigra
grouptest		8/23/2018 1:25:46 PM	rwxr-xr-x	nigra
intel		12/17/2018 3:49:56 PM	rwxr-xr-x	nigra
jupyter_runtime_dir		8/29/2019 9:46:47 AM	rwxrwxrwx	nigra
ls-dyna		10/2/2018 4:09:39 PM	rwxr-xr-x	nigra
machine-learning-wor...		2/20/2018 4:20:06 PM	rwxrwxr-x	nigra
matlab		9/19/2018 10:01:12 AM	rwxr-xr-x	nigra
mod_cp		8/9/2019 8:59:21 AM	rwxr-xr-x	nigra
mpi_mods062718		6/27/2018 4:48:09 PM	rwxr-xr-x	nigra
mumps		1/4/2019 4:44:18 PM	rwxr-xr-x	nigra
my2ndEnv		12/1/2017 2:40:30 PM	rwxr-xr-x	nigra
myEnv		11/30/2017 9:42:47 AM	rwxr-xr-x	nigra
nwchem		1/3/2019 9:04:59 AM	rwxr-xr-x	nigra
ondemand		6/24/2019 4:57:49 PM	rwxr-xr-x	nigra

0 B of 1.71 MB in 0 of 107 51 hidden

WinSCP interface for PSC.

10 Creating and viewing graphics files

If you are hoping to view graphics directly on the PSC Bridges system, you will probably need to login with the extra **-X** switch, as in

```
 ssh –X joeuser25@bridges.psc.xsede.org
```

This should be enough for Linux and Mac systems, but I have not heard whether it will take care of Windows as well.

A common way to create graphics in a Python program is to issue the command

```
import matplotlib.pyplot as plt
```

and then build a plot starting with the basic command:

```
plt.plot ( x, y )
```

To actually display the plot, we might then issue the command

```
plt.show ( )
```

I can demonstrate this procedure by running, on my laptop, the program *bulgaria_interactive.py*.

Interactive graphics display is NOT a good idea when using an HPC system; it presumes that the user is available to interactively hit RETURN so that the program can proceed. Also, you need to have logged in with the `ssh -X` switch in order to set up graphics transmission.

If you're having trouble getting PSC graphics to appear on your home system, there is a way to separate your program execution and your graphics viewing.

Replace the `show()` command with the `savefig()` command, which saves the plot to a file:

```
plt.savefig ( 'bulgaria.jpg' )
```

Once your program has executed, your graphics is saved in a file and we can try at least two ways to view it:

1. directly: on Bridges, try typing:

   ```
     display bulgaria.jpg
   ```

2. indirectly: from your laptop, use an `sftp` command like

   ```
     sftp get bulgaria.jpg
   ```

   to copy the file from Bridges, and then view it using some display program on your home system;

On the web page for this lab, you can find *bulgaria.py*, *mario.py* and *montana.py*, which all create graphics files in this way.

You can copy any of the graphics files from my directory and see if you can view them with `display`:

```
  cp ~jburkard/bulgaria.jpg bulgaria.jpg
  cp ~jburkard/mario.png mario.png
  cp ~jburkard/montana.png montana.png
```

11 Exercise: Using keras with the `interact` command

Our goal in a later lab will be to do some interesting neural network computations on a GPU using `keras`. Today, it will be "interesting" just to do a little sample calculation this way.

Copy the files *montana.py* and *montana_psc.sh* into your home directory at PSC. There are two ways to do this:

- Download the files from the lab webpage to your own system. Then transfer them to PSC;
- OR...;
- Log in to your PSC account, and copy my version, by typing

  ```
      cp ~jburkard/montana.py montana.py   <-- "twiddle" or "tilde" character
      cp ~jburkard/montana_psc.sh montana_psc.sh
  ```

Try interactive execution. You may have to wait some time to get it:

```
interact -gpu
module load anaconda3/2019.03
source activate keras-gpu   <-- This logs you onto a gpu.
python montana.py

  (Now repeat the command, but save your output to a file.)

python montana.py > montana.txt

exit             <-- Control-D or "exit" logs you off the gpu
```

12 Exercise: Using keras with the `sbatch` command

As it turns out, it's usually not easy or quick to get access to a GPU using the `interact` command. There are only a few GPU nodes available this way, and we are sharing these resources with many other people. (To get an idea of how many people are logged in right now, type **who** to see a list.)

Most computing on the PSC Bridges system is done by creating a **shell script** that lists all the commands you would normally type interactively. These commands are preceded by some special lines that explain how much time you are asking for, what kind of machine you want to use, and some other information.

The system that accepts all these job requests is called `SLURM` and so the job request file is often called a "SLURM script". We will give you a SLURM script to work with, so that you don't have to worry about creating one. However, it can be useful to take a quick look at such a script, and see whether anything makes sense.

```
#! /bin/bash
#
#SBATCH --gres=gpu:k80:4
#SBATCH -J montana_psc
#SBATCH -N 1
#SBATCH --ntasks-per-node 4
#SBATCH --output=montana_psc.log
#SBATCH -p GPU
#SBATCH -t 0:01:00
#
set -x
#
#  Set up the environment.
#
module load anaconda3/2019.03
source activate keras-gpu
#
#  Run the program.
#
python3 montana.py > montana_psc.txt
#
echo "Normal end of execution."
```

Listing 1: montana_psc.sh the SLURM script for the montana job.

You can see the section that corresponds to the commands we could have given interactively. The lines that begin with **#SBATCH** are messages to the job scheduler. For now, note that:

- `-J montana_psc` means that while our job is running, it can be identified by that name;
- `--output=montana_psc.log` means that any "comments" or error messages from the system will appear in this "log" file;
- `-p GPU` means our job should run on the GPU partition;
- `--gres=gpu:k80:4` means our job wants 4 Kepler GPU's;
- `-t 0:01:00` means our job wants 1 minute of time;

To submit our job to be run, we type

```
sbatch montana_psc.sh
Submitted batch job 6761295  <-- The system gives your request a number
```

To check what jobs I have in the queue, I specify my username `jburkard` in the `squeue` command:

```
  squeue -u jburkard
```

which might result in the report:

```
  JOBID  PARTITION      NAME      USER ST        TIME  NODES NODELIST(REASON)
 6761295       GPU  bulgaria jburkard PD        0:00      1 (Priority)
```

If for some reason I want to cancel my job, I need the number:

```
  scancel 6761295
```

Eventually, my job will run. I expect three output files:

- *montana_psc.log*, system messages, warnings, errors;
- *montana_psc.txt*, created by my program;
- *montana.png*, a graphics file my program created.

While I cannot guarantee when my job will run, the good thing is that I can log out at any time, and come back later and check to see if my job has completed.

13 Computing Assignment #9

Place a copy of *hw9.py* into your Bridges directory, either by:

- downloading *hw9.py* from today's lab web page and then transferring that file from your computer to your Bridges account;
- OR
- copying the file from my Bridges account, via the command

```
  cp ~jburkard/hw9.py hw9.py    <-- That's a "twiddle" or "tilde" character
```

Then execute this script with python:

```
python hw9.py > hw9.txt
```

Then transfer the file *hw9.txt* back to your home directory and email it to Dr Schneier **mhs64@pitt.edu** before Wednesday, 13 November.

14 Computing Assignment #10

Get a copy of the file *montana.py* into your PSC Bridges directory, either from my PSC Bridges directory, or by downloading a copy from the lab web page to your home system, and then using `sftp`.

Run the `montana.py` program on the PSC Bridges system, either:

- with the `interact -gpu` command;
- OR by submitting a batch job (for which you will also need to copy the file *montana_psc.sh*);

The graphics file *montana.png* will be created by your program. Use an `sftp` command to copy it back to your home system. Then email it to Dr Schneier **mhs64@pitt.edu** before Wednesday, 20 November.