GPU Programming Basics

Northeastern University *for all* Julian Gutierrez
NUCAR Laboratory David Kaeli

Hands-on Lab #1

Objective

- Setup computers to be able to access server.
- Write our first GPU program.

NOTE: Unless stated otherwise, work in pairs.

Part 1: Setting up computer

To be able to access the discovery cluster we need an ssh connection. Given most students usually have windows machines, we require additional software to be able to access the server. **NOTE:** If you have Mac or Linux, you should be able to use ssh from the terminal.

- Install PuTTY which allows you to create a connection through a terminal:

  http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

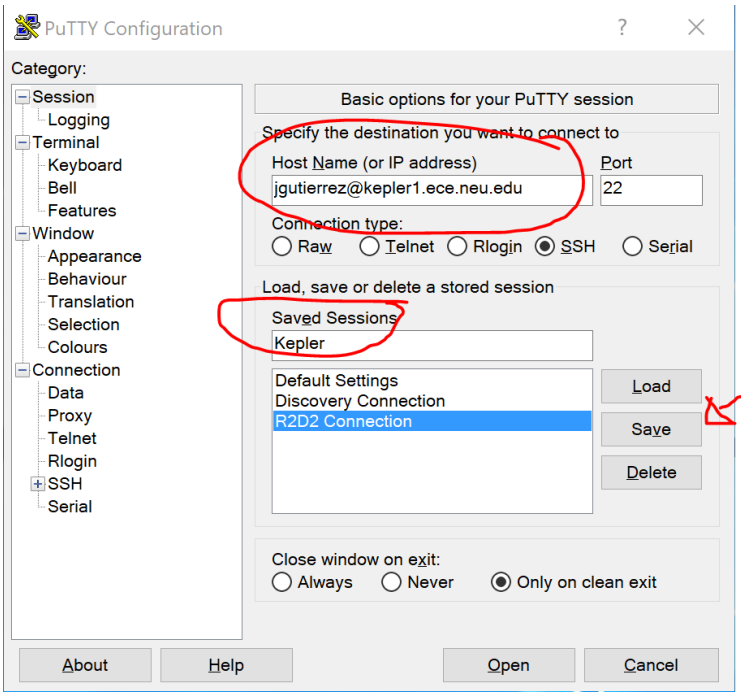
- Install FileZilla Client software:

  https://filezilla-project.org/download.php?type=client

FileZilla will allow you to copy files back and forth from the server. It's easier if you plan on editing the codes on your machine and then copying them back to the server.

How to use PuTTY

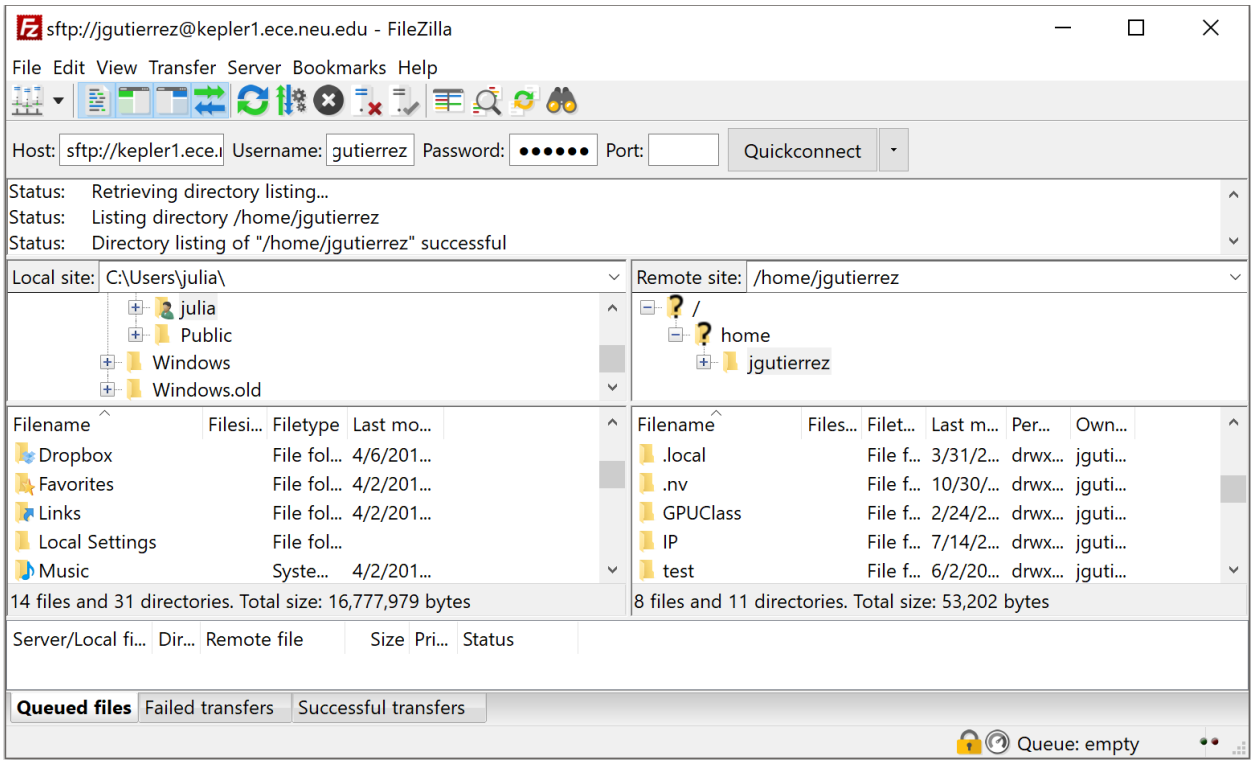
1. Open PuTTY and input your username along with the server address under Host Name: <username>@discovery.neu.edu
2. Under Saved Sessions, write "Discovery"
3. Click on Save. **NOTE: Image is just for reference.**



4. Now Discovery should appear on the list. Click on it and then click load.
5. Accept the dialog box that appears.
6. Input password.
7. You're all set. Connection to server should be successful.

How to use FileZilla

1. Open FileZilla and input the following:
    a. Host: discovery.neu.edu
    b. Username: <Username>
    c. Password: <Password>
    d. Port: 22



2. Click Quickconnect
3. And you should be done. Now you should be able to copy files from your computer (Local site to the left) to the server (Remote site to the right).

How to use the Discovery Cluster

1. First time accessing the discovery cluster (or using Putty)

```
ssh -X <USERNAME>@discovery.neu.edu
```

2. After typing the password, this will take you to a gateway node. **YOU SHOULD NOT** execute any work here. Instead you should allocate a node for yourself. To do so, you need to load the modules first.
3. You can do this by adding the following lines at the end of your ~/.bashrc file (in your home directory).
   a. Open file using an editor

```
vim ~/.bashrc
```

   b. Go to the end of the file and copy these lines.

```
# Manually load the required modules
```

```
module load slurm-14.11.8
module load gnu-4.8.1-compilers
module load fftw-3.3.3
module load openmpi-1.8.3
module load cuda-7.0
```

c. Reload the file.

```
source ~/.bashrc
```

4. Allocate the resources:

```
salloc -N 1 --exclusive -p par-gpu # Reserve the node
squeue -u $USER # Print the nodes that #USER has reserved
```

5. This will print the nodes allocated for the user. Copy the value under "NODELIST" and run:

```
  JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST
1135257   par-gpu     bash bohmagos  R       2:21      1 compute-2-132
```

6. And run the following command to access your node:

```
ssh -X compute-2-132 # your compute node may be different
```

7. From now on, you can run your work.
8. To test if everything is working fine, Run “nvidia-smi” command. It should print the information on the GPU’s available on the machine:

```
julian@kepler1:[jgutierrez]$ nvidia-smi
Thu Apr  6 14:51:19 2017
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 361.77                 Driver Version: 361.77                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla K40c          Off  | 0000:02:00.0     Off |                  Off |
| 23%   35C    P8    21W / 235W |     16MiB / 12206MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K20c          Off  | 0000:03:00.0     Off |                  Off |
| 30%   36C    P8    17W / 225W |      0MiB /  5062MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID  Type  Process name                               Usage      |
|=============================================================================|
|    0      1279    G   /usr/bin/X                                    16MiB |
+-----------------------------------------------------------------------------+
```

IMPORTANT: When you are done with everything, type the following commands to make the node available to other users.

```
exit # this will take you to the gateway computer again

squeue -u $USER # Print the nodes that USER has reserved, you should copy t
he value under "JOBID"

scancel 1135257 # free the node so another person can use it

squeue -u $USER # Just to make sure it is gone, check it again
```

Please, Remember to **ALWAYS** release the node. Always!

Part 2: Writing our first GPU Program

Copy the baseline for our code from the following directory either to your home directory in the server or into your machine (we will be editing this file).

/scratch/gutierrez.jul/GPUClass/HOL1/DELIVERABLE/VectorAdd.cu

The code is meant to do a vector add on the CPU and then do the same vector add with the GPU and compares the results at the end.

1. Open file. Find the comments and fill the code with the necessary commands to make it work.
2. The following are the templates for the commands that you'll need:

   ```
   cudaError_t cudaMalloc ( void** devPtr, size_t size )

   cudaError_t cudaMemcpy ( void* dst, const void* src, size_t count,
   cudaMemcpyKind kind )

   kind: cudaMemcpyHostToDevice or cudaMemcpyDeviceToHost

   kernel_name <<<grid, block>>>(argument list);

   cudaFree ( void* ptr);
   ```

3. Look at the kernel function and the arguments it receives, and how it does the calculation.
4. To compile the code, run the following command:

   ```
   nvcc -arch=sm_35 -O3 VectorAdd.cu -o VectorAdd
   ```

5. Make sure it gives the correct result.
6. Take the following observations into account and think of the answers to these questions while testing the code.
   a. Grid Size and Block Size should be chosen based on the variables already available.
   b. Notice cudaMalloc receives a **pointer. What does this mean?
   c. Test different vector sizes and block sizes to see if the result is still correct.
   d. Compare the performance when you increase the block size and when you decrease it.
      i. What could be affecting the behavior?
      ii. Does GPU perform better than CPU?

e. Should we include the time it takes to allocate and copy data?