CUDA Programming and Using the SCC (Shared Computing Cluster)

(based on slides from Research Computing Services)

CUDA programming can be done directly on the "VLSI" machines in PHO 305 and 307. They are more than enough for labs 7 and 8 in EC527. *If you are starting lab 7, skip ahead to the CUDA PROGRAMMING INSTRUCTIONS section about 5 pages past this page.*

If you intend to use GPU computing for your **project**, or if you just want to try out faster GPUs for lab 8, you can use GPUs by connecting to the SCC. The first several pages of this document are about the SCC.

SCC Links

Here are the links to full documentation for the things described later in this PDF:

Overview of what SCC is:
https://www.bu.edu/tech/support/research

Connecting:
https://www.bu.edu/tech/support/research/system-usage/using-scc/access-security
https://www.bu.edu/tech/support/research/system-usage/getting-started/connect-ssh

SCC Open OnDemand (web browser connection method):
https://scc-ondemand.bu.edu
https://www.bu.edu/tech/support/research/system-usage/scc-ondemand

Using programs and commands on the SCC itself:
https://www.bu.edu/tech/support/research/system-usage/scc-quickstart
https://www.bu.edu/tech/support/research/system-usage/getting-started/x-forwarding/
https://www.bu.edu/tech/support/research/software-and-programming/software-and-applications/modules

Getting a GPU on the SCC:
https://www.bu.edu/tech/support/research/software-and-programming/programming/multiprocessor/gpu-computing/

Connecting to SCC from Different Platforms

	SSH	X-Windows (for MATLAB)	SFTP (file transfer)
Microsoft Windows	**PuTTY** (see 3rd column)	**MobaXterm** https://mobaxterm.mobatek.net	**PuTTY** https://www.ssh.com/ssh/putty/download
Apple Mac OS X	**Terminal** (built in)	**XQuartz** https://www.xquartz.org	**CyberDuck** https://cyberduck.io
Linux	**Terminal** (built in)	**X11** (built in)	**Various** (built in)

SSH - Login to the SCC (with X-Forwarding)

(Complete instructions are at **www.bu.edu/tech/support/research/system-usage/using-scc/access-security** *and at* **www.bu.edu/tech/support/research/system-usage/getting-started/connect-ssh** *)*

The -X or -Y is not needed if you are going to use a "plain text" editor but most people prefer a "GUI" editor

> Linux (from Terminal) or Windows (MobaXterm) both use "-X":

```
[local_prompt]$ ssh -X username@scc1.bu.edu
username@scc1.bu.edu's Password:
```

Use your BU Kerberos name and password (same as for Student Link, email, etc.).

> Apple Mas OS X (from Terminal) uses "-Y":

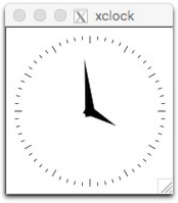
```
[local_prompt]$ ssh -Y username@scc1.bu.edu
username@scc1.bu.edu's Password:
```

Use your Kerberos name and password

> Once logged in, verify X Windows (graphical) apps work:

```
[username@scc1 ~]$ xclock &
```

> After 10 seconds or so, you should see a little clock appear (illustrated to the right)



NOTES:

> After this first SSH command you are connected to a cluster login node (which has no GPU)

> The file server is independent of the compute nodes. All nodes will access the same versions of your files and directories. Any file uploads and edits you do will be visible when you get onto a GPU node.

SCC Open OnDemand (access to GUI apps through a web browser)

(Complete instructions are at **www.bu.edu/tech/support/research/system-usage/scc-ondemand** *)*

If you want to try things quickly (and do not need a shell or SSH session) you can visit

`https://scc-ondemand.bu.edu/`

This will ask for a Kerberos username and password, then give you a web page with menus across the top: **Files**, **Quoteas**, **Login Nodes**, **Interactive Apps**. "Files" allows you to upload and download files to/from the SCC. Most of the useful programs are in "Interactive Apps"; the ones probably most familiar to ENG students are MATLAB, Mathematica, and Jupyter Notebook.

This is potentially useful for your projects, for example if you wish to use MATLAB to do a visualisation of data you got from simulation on the GPU. However there is no shell access, so no way to compile or run your program; you'll still need SSH for that.

SFTP - Transferring Files to the SCC

> Connect using your Kerberos name and password

```
[local_prompt]$ sftp username@scc1.bu.edu
username@scc1.bu.edu's Password:
```

Use your Kerberos password (same as BU email)

```
connected to scc1.bu.edu
sftp>
```

> Create a project directory

```
sftp> mkdir EC527
sftp> cd EC527
sftp> mkir lab7
sftp> cd lab7
```

> Navigate to appropriate directory on local machine
('L' commands like lcd, lls are used to do this)

```
sftp> lcd Documents
sftp> lcd lab7
sftp> lls
cuda_test.cu
```

> Copy the file from local directory to remote directory

```
sftp> pwd
Remote working directory: <remote_path>/EC527/lab7
sftp> lpwd
Local working directory: <local_filepath>\Documents\lab7
sftp> put cuda_test.cu
Uploading cuda_test.cu
cuda_test.cu
sftp> ls
cuda_test.cu
sftp>
```

Files can also be copied from remote to local
Type 'help' to get a full list of commands

Request a GPU Node

> Log in as described in the earlier section "Login to the SCC"

> Request access to a node with a GPU, specifying the `ec327` "project" group:

```
[username@scc1 ~]$ qrsh -l gpus=1 -P ec527
```

> If you get an error message like *`job rejected: no access to project "ec527" for user "username"`* then you should let a TA or the professor know. You may also check what other "projects" you belong to with the `id` command:

```
[username@scc1 ~]$ id -nG
scc-lite ec527
```

However it is best to use the ec527 project for accounting purposes (so your usage is charged to the course)

If you cannot get a GPU node on the SCC for whatever reason, the machines in PHO 305 have GPUs. If you decide to work on one of the PHO 305 machines you can just skip to the CUDA PROGRAMMING INSTRUCTIONS section.

> The `qrsh` command can take up to about 60 seconds. A message like this should be displayed:

```
********************************************************************************
          This machine is owned and administered by Boston University.

               This machine is governed by Boston University's
               Conditions of Use and Policy on Computing Ethics.
      https://www.bu.edu/policies/conditions-of-use-policy-computing-ethics/

  Information about Research Computing Services (RCS) facilities and services:
                              https://rcs.bu.edu/

                     Information about using the SCC:
            https://www.bu.edu/tech/support/research/system-usage/

                Please send questions and report problems to "help@scc.bu.edu".

********************************************************************************
[username@scc-k02]$
```

Editing Source Code on the SCC

Plain-text style editors

nano - "Nano's ANOther editor"	nano `cuda_test.cu`
emacs - "Editor MACroS" use the -nw option:	`emacs -nw cuda_test.cu`
vi - vim - "Visual Editor" / "Vi IMproved"	vim `cuda_test.cu`

Graphical editors (requires that you used -X or -Y option when connecting)

gedit - "Gnome EDITor"	`gedit cuda_test.cu &`
emacs - "Extensible MACroS"	`emacs cuda_test.cu &`
gvim - GUI vim	`gvim cuda_test.cu &`

The "&" at the end makes it so you can continue typing other commands in the SSH window while the editor is running in its own window.

More Info, Help, etc. for SCC

Getting Started (SSH, SFTP, etc):
https://www.bu.edu/tech/support/research/system-usage/getting-started/

SCC Quick Start Guide
https://www.bu.edu/tech/support/research/system-usage/scc-quickstart/

Page with links to the topics listed here:
https://www.bu.edu/tech/support/research/system-usage/

Running Jobs:
https://www.bu.edu/tech/support/research/system-usage/running-jobs/

Also of use (PDFs on Blackboard):
Linux_SCC_CheatSheet-20181218.pdf
SCC_CheatSheet-20180705.pdf

Help:
Research Computing Services Website: https://www.bu.edu/tech/support/research/
help@scc.bu.edu

CUDA PROGRAMMING INSTRUCTIONS

These instructions apply to everything - the lab machines (in PHO 305/307) and the SCC.

Verify CUDA Compiler, (and enable if needed)

> Find out shat type of GPU you have with the "`nvidia-smi`" command:

```
[username@scc-c01 ~]$   nvidia-smi
...
|=============================+======================+======================|
|   0  Tesla K40m          On   | 00000000:03:00.0 Off |                    0 |
| N/A   29C    P8    20W / 235W |      0MiB / 11441MiB |      0%   E. Process |
+-----------------------------+----------------------+----------------------+
...
```

A lot of information is listed, and there might be multiple GPUs. You just care about the model number.

> Use this table to find what version of `nvcc` you need, and what compiler flags to use.

GPU Model	*module load command*	*compiler options*
SCC (for project work):		
Tesla V100-SXM2	`module load cuda/10.0`	`-arch compute_70 -code sm_70`
Tesla P100	`module load cuda/9.2`	`-arch compute_60 -code sm_60`
Tesla K40m	`module load cuda/9.2`	`-arch compute_35 -code sm_35`
Tesla M2070	`module load cuda/8.0`	`-arch compute_20 -code sm_20`
Grid (PHO 305,307) (for labs 7 and 8):		
Quadro P1000	`module load cuda`	`-arch sm_35`
Quadro K610M	`module load cuda`	`-arch sm_30`

> Now, pick an `nvcc` compiler version using the appropriate "`module load`" command for your GPU type:

```
[username@scc-c01 ~]$ module load cuda/9.2
```

> Confirm that the Nvidia compiler is available and version number matches:

```
[username@scc-c01 ~]$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Tue_Jun_12_23:07:04 CDT 2018
Cuda compilation tools, release 9.2, V9.2.148
[username@scc-c01 ~]$
```

(If you see something like "*`nvcc: command not found`*", make sure to do a "`module load`" command each new time you log in)

Compile and Run

> Navigate to the directory with cuda_test.cu

```
[username@scc1 ~]$ cd EC527
[username@scc1 EC527]$ cd lab7
[username@scc1 lab7]$
```

> Compile with the `nvcc` command using the appropriate options for your GPU type:

```
[username@scc1 lab7]$ nvcc -arch compute_35 -code sm_35 cuda_test.cu -o cuda_test
```

> Run the program.

```
[username@scc1 lab7]$ ./cuda_test
Length of the array = 50000

Initializing the arrays ...   ... done

GPU time: 0.128916 (msec)

TEST PASSED: All results matched
[username@scc1 lab7]$
```

This test program is very small and simple and should always work. For help with error messages getting `cuda_test.cu` to compile or run, see the following pages.

Troubleshooting Errors from `cuda_test.cu`

If your GPU is not listed on the previous page (SCC adds new hardware every year) try each line in the table. Start with the high cuda versions and work down from there, becase the newly-added hardware is probably using newer, better GPUs. You can also use the "`module avail cuda`" command (described below) to find out if newer versions of the CUDA software have been installed.

> "*`sm_10 architecture is deprecated`*"
> "*`Double is not supported, demoting to float`*"
> These errors are seen on the ENG-Grid "VLSI" machines (in PHO 305) if you don't give a `-arch` option or choose a really old version like `sm_10`. The minimum that works is `-arch sm_13`
>
> "*`CUDA_SAFE_CALL: invalid device function  cuda_test.cu 120`*"
> These are seen on the ENG-Grid machines when giving the wrong `-arch` and/or `-code` options (for example if you try the Tesla M2070 options "`-arch compute_20 -code sm_20`")
>
> "*`no kernel is available to run`*"
> "*`CUDA driver version is insufficient for CUDA runtime version`*"
> These errors are seen on the SCC if your `-arch` option is missing or is too low.
>
> "*`The 'compute_20', 'sm_20', and 'sm_21' architectures are deprecated`*"
> This error is seen on some SCC nodes and it appears it can be ignored. Unfortunately, the newer options like `sm_30` will not run on the Tesla M2070.

You can search online for error solutions too. Find out more about your compiler and hardware, and use this information to investigate your error messages online.

The version numbers for "`module load`" and the `nvcc -arch` option correspond to different generations of GPU products.

To find what compiler you have currently selected:

```
[username@scc-sc1 ~] nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2018 NVIDIA Corporation
Built on Tue_Jun_12_23:07:04_CDT_2018
Cuda compilation tools, release 9.2, V9.2.148
```

To find what other versions are available:

```
[username@scc-sc1 ~] module avail cuda
---------------------- /share/module.7/programming ----------------------
cuda/10.0 cuda/10.1 cuda/8.0  cuda/9.0  cuda/9.2
```

To find out what type of GPU you have:

```
[username@scc-sc1 ~] nvidia-smi
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.40.04    Driver Version: 418.40.04    CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| 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 K40m          On   | 00000000:03:00.0 Off |                    0 |
| N/A   29C    P8    20W / 235W |      0MiB / 11441MiB |      0%   E. Process |
+-------------------------------+----------------------+----------------------+
|   1  Tesla K40m          On   | 00000000:83:00.0 Off |                    0 |
| N/A   30C    P8    19W / 235W |      0MiB / 11441MiB |      0%   E. Process |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Another way to find out what version of the Nvidia driver is running:

```
[username@scc-sc1 ~] cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX x86_64 Kernel Module  390.116  Sun Jan 27 07:21:36 PST 2019
```

To investigate further you would then search online for something like **nvidia driver 390 cuda**

This page lists valid `nvcc` compiler options for different GPUs:

```
https://arnon.dk/matching-sm-architectures-arch-and-gencode-for-various-nvidia-cards
```

This CUDA Wikipedia page also has lists "compute capability" versions for different GPUs. For example it states that the K610M is version 3.0. This corresponds to the option "`-arch sm_30`".

```
https://en.wikipedia.org/wiki/CUDA
```