Using the Tufts High-Performance Compute Cluster for your GPU labs

You will be your GPU labs on the Tufts High-Performance Compute Cluster (HPCXC) rather than using Halligan machines (which, though they have GPUs, do not have the softwals /re installed to support GPU programming). The HPCC is based on a Linux command line; since it's much less graphical and visual, it can be much more reasonable to use it without a high-bandwidth low-latency internet connection. On the other hand, there are only 42 GPUs for the entire Tufts community, so resources may be very tight. Here are the basics of using it.

Logging in

In order to log in, you must be on the Tufts network. If you are not physically at Tufts, you have to use the Cisco AnyConnect VPN. You can find the download and manual at https://tufts.box.com/v/vpnsoftware . If you're not connected to the VPN, then any attempts to log into the HPCC will just hang. And note that you must use your *Tufts* UTLN and password, not your Halligan username/password

Once you're on the VPN, then use Moba Xterm as usual to connect to *yourname*@login.pax.tufts.edu, where *yourname* is your Tufts UTLN.

Files and moving files to the HPCC

- Your login directory on the HPCC is typically at /cluster/tufts/ee155class/*yourUTLN*.
- We'll have the files for the GPU labs sitting in the HPCC at /cluster/tufts/ee155class/jgrods01/files/*.
- You can move additional files to the HPCC if needed with **scp** (i.e., secure copy). If you're logged into the Halligan cluster, you can run

  scp *cxx *hxx jgrods01@xfer.pax.tufts.edu:/cluster/tufts/ee155class/you-login-name

  This would copy all .cxx and .hxx files from the current Halligan directory to my HPCC home directory. A similar syntax would copy results files in the other direction. (Note that my UTLN is *jgrods01*).
- Files are also available on the course web page as usual.

Editing files

- You can launch vim from the HPCC Linux terminal window.
- You can also launch Emacs from the terminal window, but first you must type "module load emacs/X26.3".

Compiling

- The HPCC has numerous compiler and environment versions available. Set up the correct ones with

  module load cuda/12.2
  module load gcc/11.2.0
- Compile with

  nvcc -O2 -std=c++11 matrix_mpy_user.cu matrix_mpy.cu matrix.cxx ee155_utils.cxx
- The three commands above are in the file **nvcc.sh** in the directory noted above.

Running your program

- Here's where it gets a bit more complicated. When you ssh into the HPCC, you're logged into a login server. It's OK to do editing and compiling on this machine, but it's *not* OK to do compute-

intensive jobs. Furthermore, the login server doesn't have a GPU on it, so you couldn't run your program even if you tried. Instead, you have to launch a job on a machine with a GPU. A simple way to do that is with the command

srun -p preempt --gres=gpu:1 --pty bash *run.sh*

This command will request a GPU machine, wait for one to become available (anywhere from a few seconds to almost indefinitely!), run the shell script in *run.sh*, and then return control to you. The script *run.sh* could contain the commands

```
module load cuda/10.0
module load gcc/7.3.0
./a.out > results.out
```

You can find a version of this file in the same directory noted above. You can find more detail on what is going on under the hood in the "All about Slurm" section below.

- To run the memory checker, instead of running *./a.out*, you would modify *run.sh* as per the notes in our debug-hints document at https://www.ece.tufts.edu/ee/155/misc/GPU_Linux_debug.pdf.

Performance variability

The HPCC has about 200 GPUs, spanning over half a dozen different GPU types. Your run times can vary greatly depending which GPU your job lands on. We could "fix" this variability by restricting your jobs to a particular GPU type, but then you would likely have to wait longer to get a GPU. So it seemed better to just accept the variability; at the top of each run, we print the compute capability of the machine you landed on.

Ondemand

- We have described an xterm-based interface to the HPCC. You can use a web browser instead, by pointing your browser to https://ondemand.pax.tufts.edu . After logging in (and again, it will only work if you have the VPN running), you have numerous choices.
- You can use the *Files*/*Home directory* menu item to access a simple files browser similar to a Windows My Computer window. It includes a simple editor with Emacs and Vim key bindings (all of which are built on websockets).
- The *Interactive apps* / *Xterm* menu item creates an Xterm browser tab; it is implemented via VNC and supports X11 applications. The *clusters*/*Tufts HPC shell access* and *clusters*/*Tufts fast X11 HPC shell access* menu item do the same (but not via VNC).

All about Slurm

- The HPCC is a collection of numerous interconnected computers. Each computer is called a *node*. The nodes contain over 7000 cores! At the next level up, the nodes are organized as half a dozen *partitions*, which then form the cluster. All of the nodes with GPUs belong to the "gpu" or "preempt" partitions.
- The partitions and nodes are managed via *slurm*, a commonly-used open-source cluster manager.
- **srun** is a command that waits for a node in the requested partition to become available, and then launches an interactive jobs on that node and waits for it to finish. You can get more details with, e.g., "google slurm srun." Basically, our "**srun -p preempt --gres=gpu:p100:1 --pty bash** *run.sh* requests a node in the "preempt" partition, specifically asks for a Pascal GPU, and runs the command "bash run.sh" on that node.

- It is actually possible to simply run “bash” rather than “bash run.sh,” which would log you into the given node for an interactive session. *Please do not do this* – GPUs are a scarce resource and should be used only for computing; you would be locking everyone else out of the GPU even though you’re not using it much.
- Slurm has many other features. You can also submit jobs with **sbatch** rather than **srun**, which queues the job up for batch execution and returns control to you immediately. If you prefer to do this, a sample file is at sbatch.sh.