Check of CUDA environment

- **Login:** `ssh username@oakley.osc.edu`.
- **Ensure CUDA is available:** `module load cuda`.
- **Acquire and unpack example source code:**

```
mkdir CUDAex
cd CUDAex
wget http://developer.nvidia.com/sites/default/files/akamai/cuda\
/files/cuda_by_example.zip
# Alternative location:
wget http://slac.stanford.edu/~rolfa/cuda_by_example.zip
unzip cuda_by_example.zip
```

- **Compile find-the-GPUs example:**

```
cd chapter03
nvcc -o enumGPUs enum_gpu.cu
```

Batch submission

- **First run the enum program in batch; this requires a submission script:**

```
#PBS -l walltime=0:10:00
#PBS -l nodes=1:ppn=12:gpus=2
#PBS -N my_job
#PBS -S /bin/bash
#PBS -j oe
module load cuda
pbsdcp $HOME/CUDAex/chapter03/enumGPUs $TMPDIR
cd $TMPDIR
./enumGPUs
```

- **Submit and check status:**

```
qsub subenumscript
qstat | grep username
```

- **Other useful commands:**

```
showstart jobname
qpeek jobname
qdel jobnumber
```

Some comments

- **Compute node has access to your home directory, but this is slow and will make you unpopular if abused.**
- **Move input data and executables to $TMPDIR before using them with `pbsdcp` command (Portable Batch System Distributed Copy). For jobs running on one CPU plain `cp` will also work.**
- **Log output will go in file `JobName.oJobNumber` in the directory from which you run qsub.**
- **If your program writes to a file, or otherwise has output that's not going to the logfile, copy it from $TMPDIR to $HOME at the end of your batch script.**
- **Batch queue can be kind of slow. This is fine if you're submitting something that will take three hours; go have lunch. If it'll take thirty seconds and then you change the code, it's annoying to wait ten minutes per submission.**

Interactive sessions

- **For quick testing, small jobs, anything that requires rapid feedback, use an *interactive batch* session instead.**
- **In effect we are logging in to a compute node:**

```
qsub -I -l nodes=1:ppn=6:gpus=1 -l walltime=01:00:00
```

- **Now it works just like any other login. Load modules, compile, run. But note that access to your home directory is slow! Copy things across just as before.**

```
module load cuda
cd $TMPDIR
pbsdcp $HOME/CUDAex/chapter03/enumGPUs .
./enumGPUs
```

- **When done with an interactive session, do `exit`.**
- **For rapid iteration, it may be better to copy the code across and do compiling in the interactive session.**

Coding exercise

- **CUDA code is conventionally put in files labelled foo.cu to distinguish it from plain C++. You may want to add this to your .emacs file:**

```
(setq auto-mode-alist (cons '("\\.cu$" . c++-mode) auto-mode-alist))
```

- **Copy the code from ~ucn1122/goofitcourse/exercises/exercise1.cu or from www.slac.stanford.edu/~rolfa/goofitcourse/exercise1.cu and solve the exercises in it. My process for compiling and running the exercise:**

```
qsub -I -l nodes=1:ppn=12:gpus=2 -l walltime=01:30:00
# Wait a minute for job to start
cd $TMPDIR
module load cuda
cp $HOME/goofitcourse/exercises/exercise1.cu .
nvcc -arch=sm_20 -o lab1 exercise1.cu
./lab1 256
```

- **Here sm_20 indicates that the target has compute capability at least 2.0. Support for double precision was introduced in 1.3, but as we have a reasonably modern card we may as well use 2.0 instead.**