An Introduction to HPC and Scientific Computing

CWM, Department of Engineering Science

University of Oxford

Practical 5: Practical examples of CUDA libraries.

This practical will review some of the CUDA libraries and there uses which were discussed in the lecture "An introduction to GPUs and how to use them".

The learning outcomes of this practical are:

- To have some understanding of CUDA libraries and there uses.
- To understand how memory is allocated on the GPU.
- To understand how data is transferred to and from the GPU.
-

All practicals for this course will be carried out on the Universities ARCUS-B computer. To understand how to use ARCUS-B see the slides from lecture 3. As a reminder log in using ssh as follows:

```
ssh –CX teachingXY@arcus-b.arc.ox.ac.uk
```

Where teachingXY is the account that we have issued you with.

When logged into the Arcus-B head node, you can create an interactive session on one of the K80 GPU compute nodes by issuing the following command:

```
salloc -pgpu --ntasks-per-node=1 srun --pty --x11 --preserve-env /bin/bash -l
```

and once you are then put onto one of the K80 nodes, issue the commands

```
module load gpu/cuda
```

and

```
export CUDA_VISIBLE_DEVICES=0,1,2,3
```

If you have not done so clone the github repo for this CWM. To do this, at the command prompt type:

```
$ git clone https://github.com/wesarmour/CWM-in-HPC-and-Scientific-Computing.git
```

```
Or

        $git pull
```

To update your local repo.

Instructions for this practical

Part A (The first part of this practical is courtesy of Mike Giles)

1. Navigate to the code director in the prac5 directory.

2. Make both applications, simpleBLAS and simpleFFT by typing make.

3. Run them – the output should show that the error between the library result and the corresponding CPU “Gold” code is very small.

4. Read through the source files to see how the library routines are used, referring to the online documentation for both CUBLAS and CUFFT.

Part B

1. Read through the code provided in 'prac5/code/cuFFT/cuFFT_C2C.cu'. The main() function is preparing data which will be Fourier transformed by the GPU. Locate where we set the size of the FFT and number of series we want to transform. Find where we initialize our series which we want to be Fourier transformed. Is there a problem?

2. We have written a function 'Do_FFT_C2C_forward'. It provides an example of how to perform an FFT using cuFFT library. Read through comments in the code. Can we reuse the same cuFFT plan with different data? Where would you place a ‘for’ loop which would do that? Can we reuse same cuFFT plan if we change number of FFTs we would like to calculate? What about the FFT size?

3. Complete function 'Do_FFT_C2C_inverse_inplace' based on reading through function 'Do_FFT_C2C_forward'. In function 'Do_FFT_C2C_inverse_inplace' we would like to perform inverse and in-place FFT by using cuFFT.

4. Run the program and look what it prints to the console. If you have written your code correctly the difference between input and output should be 0 (or very close to it).

Part C

1. Take the information given in the lecture notes and use it to write a code that uses cuRAND to generate a normal distribution of numbers.

2. Add a function to your host code that will calculate the mean and standard deviation of the random number distribution that you generate using cuRAND.

3. Write a function that will generate a histogram of the randoms.

4. Use printf() to output your histogram to a file and then using a plotting tool (such as gnuplot) to view the results.

Bonus questions

a. Generate different random number distributions using cuRAND and plot the results.
b. Implement the Box-Muller transform on the CPU and compare the results to those generated by cuRAND (https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform)

Do not worry if you don't complete all of the above. The aim of this practical is to encourage you to write your own C code and become familiar with some of the common functions.