CUDA Programming on NVIDIA GPUs
Mike Giles

Practical 1: Getting Started

This practical gives a gentle introduction to CUDA programming using a very simple code. The main objectives in this practical are to learn about:

- the way in which an application consists of a host code to be executed on the CPU, plus kernel code to be executed on the GPU
- how to copy data between the graphics card (device) and the CPU (host)
- how to include error-checking, and printing from a kernel

The practicals are to be carried out on the `Frontera` system. Before starting, please read the notes at
`https://people.maths.ox.ac.uk/gilesm/cuda/frontera_notes.pdf`.
(If you are reading this PDF document online, the link above should appear in blue and you can click on it to go to the notes.)

What you are to do is as follows:

1. Copy all of the course files to your home directory, following the directions given in the notes.

2. The two codes `prac1a` and `prac1b` are in the same directory `prac1`. They are compiled and linked by the command

   ```
   make
   ```

   which carries out the steps within the Makefile.

3. Read through the `prac1a.cu` source file and compare it to the `prac1b.cu` source file which adds in error-checking.

4. Run both codes:

   ```
   ./prac1a
   ```

   ```
   ./prac1b
   ```

   and read them through to understand what they are doing – ask questions if anything is not clear.

5. Try introducing errors into both `prac1a.cu` and `prac1b.cu`, such as trying to allocate too much memory (e.g. by specifying an enormous value like `(long long) 500000000000`), or setting `nblocks=0` or `nthreads=10000`, and see what happens.

6. Add a `printf` statement to the kernel routine `my_first_kernel`, for example to print out the value of `tid`. Note that the new output may be written to the screen after the existing output from the main code, because it gets put into a write buffer which is flushed only intermittently.

7. Modify `prac1b.cu` to add together two vectors which you initialise on the host and then copy to the device. This will require additional memory allocation and two `memcpy` operations to transfer the vector data from the host to the device.

8. There is a third version of the original code, `prac1c.cu`, which uses "managed memory" on top of Unified Memory. Read through the code to see what it does, and try compiling and running it.

9. If you have spare time, you can browse through the online info on NVIDIA's sample codes which are on GitHub at https://github.com/nvidia/cuda-samples.