NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
TDT4200 – PARALLEL COMPUTING

Problem Set 2:
CUDA Intro – Addendum

Per Magnus Veierland
permve@stud.ntnu.no

September 18, 2015

PROBLEM 2: CUDA GPGPUS

c) **Which of `kernel1()` and `kernel2()` will execute fastest, given that X and Y are *gridDim* and *blockDim* respectively, containing 3 integers with positive powers of 2 higher than 2^4?**

The kernels will both compute the same amount of effective work, i.e. they will both call the `some_other_task()` function the same number of times. When comparing the time it takes to execute one warp in `kernel1` where the `some_other_task()` function is called with the time it takes to execute one warp from `kernel2` in which the `some_other_task()` function is called, these will be the same.

The important difference between the kernels is in how they utilize the available hardware. For `kernel1` only a low number of threads in a warp will effectively do work, while the rest of the threads are wasted for the duration of the computation. The ratio of threads within each thread block which will be used effectively is given by:

$$\frac{4 \cdot 32 + 4 \cdot 32 - 4 \cdot 4}{32 \cdot 32} \approx 0.234375$$

In the case of `kernel2`, the branching occurs on block boundaries which means that a full block will either be calling `some_other_task()`, or immediately completing. This means that 32 out of 32 threads in a warp will be used productively, and in the cases

where there is no computation to be done, the warp will immediately complete – leaving room for new warps.

Given an infinite number of streaming multiprocessors, both `kernel1` and `kernel2` would take the same amount of time to complete. However in reality there is a limited number of multiprocessors, and only a given number of warps can run at a time.

This difference can easily be verified in benchmarks. Using a test case with a grid size of 32 by 32 and a block size of 32 by 32 performing some arbitrary work – the time spent by `kernel1` was 1440822 us, while the time spent by `kernel2` was 335491 us. This fits well with the ratio of threads used effectively in a thread block by `kernel1`:

$$\frac{335491 \text{ us}}{0.234375} \approx 1431428 \text{ us}$$

Hence it is clear that `kernel2` will execute the fastest.