In this lab, you’ll investigate an important factor in CUDA performance.

Strange timing results

Copy lab4.cu from /home/courses/cs180j/ and look through this program. You’ll find a fairly simple main that times a kernel call, plus two different kernels (unimaginatively called kernel1 and kernel2). Although significantly different in length, the kernels appear to be equivalent. (The actual operation performed is not useful or meaningful.)

Run this program on huygens. Then modify main to run kernel2 instead and compare the times. You should get significantly different running times. Of course, that there is a difference isn’t surprising since the switch statement clearly increases the number of instructions executed. However, if you change the number of cases in kernel2, you’ll see that its running time varies. Since switch on small ranges of values is implemented as a table lookup, the running time of the switch itself cannot explain what we see.

How do you explain the different performance of the two kernels? Think about this; the key observation is how the different threads behave. When you think you know the answer or are thoroughly stuck, talk with me. (The book talked about the relevant phenomenon, but not in great detail so you could have missed it.)

The same phenomenon can be demonstrated via the following other experiments as well:

- Switch both of the given kernels to determine tid based on blockIdx rather than threadIdx.
- Compare kernels with loops: one kernel should loop for a number of iterations dependent on the thread index while the other should always loop for the maximum number of iterations.

Run each of these experiments and explain your results.

If you have more time, spend it either designing other experiments to illustrate this phenomenon or continuing to “CUDAify” the Game of Life.