Optimization in CUDA

By Shin Kee Chung

CAASTRO (UWA)

Why we need optimization?

- Optimization is a very crucial part in CUDA.
  - Without optimization in your program, you are not fully utilizing the advantage of using GPU.
  - Sometimes you might get 2x or 3x speed-up (compared to a single core CPU implementation), or possibly even slower than CPU.

One Realistic Example

Basic algorithm: IIR Filters

$$y_0[k] = a_0 y_0[k-1] + b_0 x[k]$$
$$y_1[k] = a_1 y_1[k-1] + b_1 x[k]$$
$$y_2[k] = a_2 y_2[k-1] + b_2 x[k]$$

$$y_n[k] = a_n y_n[k-1] + b_n x[k]$$

Final output,

$$y = y_0 + y_1 + y_2 + \ldots + y_n$$

Speed-up Obtained

Method	Overall Speedup
Straightforward Design with `atomicAdd()`	7
Parallel Sum Reduction	25
+Use Texture Memory	42
+Avoid Bank Conflicts	48
+Tuning Resource Usage (registers and shared memory)	58

Parallel Sum Reduction

- Assign multiple threads to perform summation for an array of numbers.
  - Example can be found in 5.4.3 of CUDA C Programming Guide

Texture Memory

- Memory is still allocated with `cudaMalloc()`.
  - But you “bind” the memory space to texture, explicitly telling the compiler to treat this as texture memory.
  - Explanation in 3.2.10.1.1 CUDA C Programming Guide (functions in Appendix B.8).
  - Also example code in CUDA-SDK, simpleTexture

Tuning Resource Usage

- It is possible to limit the register usage in kernel functions, can be done by an `nvcc` compiler flag `-maxregcount <N>`
- Why we want to do it?
  - It might be slower in reading some of the variable.
  - But it improves occupancy, if one thread uses a huge number of registers, it might reduce the number of active warps

Warp Size

- Warp is the collection of threads that get executed as a whole.
- There will be an exercise introducing the concept of warp.

- Due to time constraint:
  - The exercises that we should do or not do
    - Part 2.1 do (CUBLAS)
    - Part 2.2 skipped (FFT)
    - Part 2.3 skipped (atomic operations)
    - Part 3.1 do (CUDA profiler)
    - Part 3.2 do (Choice of blocks and threads)
    - Part 3.3 skipped (Accumulate more data)
    - Part 3.4 do (Aligned memory access)
    - If you can finish these, then you can proceed on from part 3.5 onward.