Homework #3 (HW3)

Autumn 2022 - GPU Computing

DUE: 31 October 2022, 5:59 PM

Learning Objectives

- Get experience understanding control divergence in warps
- Implement and profile kernels with warp divergence behavior
- Implement improvements to the MMULT tiled-phased algorithm kernel
- Get a better understanding of kernel launch configuration tuning.
- Get basic experience using Nsight Compute GUI for profiling kernel execution and analyze kernel performance issues and optimization.

Questions

I. If the SM of a CUDA device can take up to 1536 threads and up to 4 thread blocks. Which of the following block configurations would result in the largest number of threads in the SM?
   A. 128 threads per block
   B. 256 threads per block
   C. 512 threads per block
   D. 1024 threads per block

II. For a vector addition, assume that the vector length is 2000, each thread calculates one output element, and the thread block size is 512 threads. How many threads will be in the grid?
   A. 2000
   B. 2024
   C. 2048
   D. 2096

III. With reference to the previous question, how many warps do you expect to have divergence due to the boundary check on vector length?
   A. 1

   B. 2
   C. 3
   D. 6

IV. Assume a device that allows up to 64 blocks per SM and 2048 threads per SM. Indicate which of the following assignments per SM are possible. IN the cases in which it is possible, indicate the occupancy level.
   A. 8 blocks with 128 threads each
   B. 16 blocks with 64 threads each
   C. 32 blocks with 32 threads each
   D. 64 blocks with 32 threads each
   E. 32 blocks with 64 threads each

V. Consider a GPU with the following hardware limits: 2048 threads per SM, 32 blocks per SM, and 64K (65,536) registers per SM. For each of the following kernel characteristics, specify whether the kernel can achieve full occupancy. If not, specify the limiting factor.
   A. kernel uses 128 threads per block, and 30 registers per thread.
   B. kernel uses 32 threads per block, and 29 registers per thread.
   C. kernel uses 256 threads per block, and 34 registers per thread.

Analysis

Control Divergence

Calculate the impact of control divergence for the N tiles in the tiled MMULT algorithm

- In the same way as done in Lecture 4 slides for the M tiles
    a. refer to lecture slides
- Discuss the behavior of the Type 1 and Type 2 blocks
    a. How many warp phases have divergence for each?
    b. What is the expected performance impact %?
        - for each Type (1/2)?
        - Total?

Programming Exercises

NOTE: Always use a RELEASE (x64) build when profiling your code with Nsight Compute!

Control Divergence

- After completing the Analysis-Control Divergence section above, now implement simple CUDA kernels which exhibit control divergence behavior for each of the following cases (as discussed in Lecture 4). The kernels don't need to do any significant calculations or produce any particular result values - we are just interested in observing the warp divergence effects. *However, be sure to at least update an output result variable in each branch case - otherwise the compiler may optimize and remove the whole branch if it never actually does anything!*
    1. single branch (control flow bifurcation)
    2. nested branch scenarios (at least three different control paths per warp)
- use Nsight Compute to profile the kernels for cases with/without control divergence
    a. Save the Nsight Compute kernel profile reports (*.ncu-rep) for executions of each of the diverging kernels for the cases
        - with divergence
        - without divergence
    b. Based on the configuration of your kernels for control divergence, can you explain how the Nsight Compute metrics can capture or identify when divergence is happening?
        - Look at
            - Occupancy
                - Theoretical vs Achieved Occupancy
                - Theoretical vs Achieved Active Warps Per SM
            - Warp State Statistics
                - Avg. Active Threads Per warp
                - Avg. Not Predicated Off Threads Per Warp
            - Any other general metrics (GPU Speed of Light) which change between divergent and non-divergent runs?

MMULT Variants

- Implement the following MMULT algorithm variants
    - CPU serial reference version and the
    - GPU kernel variants (naive, all-global)
    - GPU general shared-memory tiled/phased kernel with boundary checks
        - support general nonsquare conforming input matrix dimensions

                1. P: j x l, M: j x k, N: k x l
        - support tile (= thread block) sizes not perfect multiple of input matrix dims
        - (refer to PMPP Ch 4.4 - 4.6 and lecture 4 slides)
- Provide host verification code that checks all three kernels produce consistent results, within an acceptable error tolerance threshold.
    - what is the maximum single-element error fraction observed?
        - error fraction = abs[Pcpu(x,y) - Gpu(x,y)] / Pcpu(x,y)
        - evaluate for several input/output matrix sizes:
            1. P : 100x100, k = 100
            2. P : 1000x1000, k=1000
            3. P : 2000x3000, k = 2500
- For each of the 3 input/output matrix size cases above, Use Nsight Compute to generate profiles reports. View the Roofline chart in the GPU Speed of Light section.
    - Use baselines to add a baseline for each case, so that all the variants can be viewed on a single Roofline chart.
- Experiment with different TILE_WIDTH sizes (try 8, 16, 32, 64)
    - and also different thread-block (CTA) launch configurations
        - (16, 16, 1)
        - (32, 8, 1)
        - (64, 4, 1)
        - (8, 32, 1)
    - Using baselines, try to put everything onto a single Roofline plot.
        - Note- each baseline can be renamed with a more meaningful name.
    - Which configurations (TILE_WIDTH + CTA) provide the best performance?
        - highest AI?
        - Highest compute throughput?
        - Fastest execution Duration?
        - Highest Occupancy?
    - Do these results vary with the size of the input/output matrices?
        - Try to explain the behavior in a few sentences, referring to available device resources and other Nsight profile metrics as applicable.
- Select the best performing configuration for the case with P: 2000x3000, k = 2500.
    - In your VS Project Properties -> Configuration Properties -> CUDA C/C++ ->Command Line
        - In Additional Options: enter the compiler option --fmad=false
            1. this disables use of the fused instruction equivalent (FFMA)
            2. it defaults to ‘true’
        - click OK
    - Rebuild a Release version of your code and profile it with Nsight Compute.
    - Did the performance change?
        - AI, execution duration, compute throughput, occupancy?

SUBMIT

1. Submit answers to the questions, including shown work and explained reasoning.
    a. Including answers to the control divergence analysis questions
2. Submit your CUDA host application source code
    a. .cu, .cpp and .h files only - do not submit all project & solution files and other build by-products!
3. Submit your CUDA kernel functions
    a. may be in the same file as your host code if using a single *.cu file
4. Submit your NCU profiling results for all test image cases and global + shared kernels
    a. also include your calculated kernel execution time and kernel instruction intensity values.
5. Submit a page with a discussion of your profiling analysis for the MMULT kernels and the various launch configurations and input/output matrix sizes, and the FMAD option.
    a. include the result Nsight Compute profile reports with the set of profile baselines showing all the various configurations tested.