Lab 3: Optimizing memory usage

Advanced Computer Architecture

Stijn Crul

Thursday 14th November, 2019

Lab session 3 deadline	20/12/2019 23:59

1 Introduction

The GPU has its own RAM, also called global memory. So far we have always used the global memory. However the CUDA SoC also contains shared memory. This on-chip memory is much faster than the global memory. A schematic overview of the CUDA memories is shown in Figure 1. Furthermore the memory gets split into blocks. If a thread accesses a memory location, the GPU reads the whole block of memory. This means if other threads need to access other memory locations, it would be good for performance if the data was stored in the same memory block. Threads addressing data in the same block are called coalesced. If there are equal intervals between memory adresses for thread access, the memory is strided. At worst, memory access for threads is random and will lead to bad performance. This is illustrated in Figure 2.

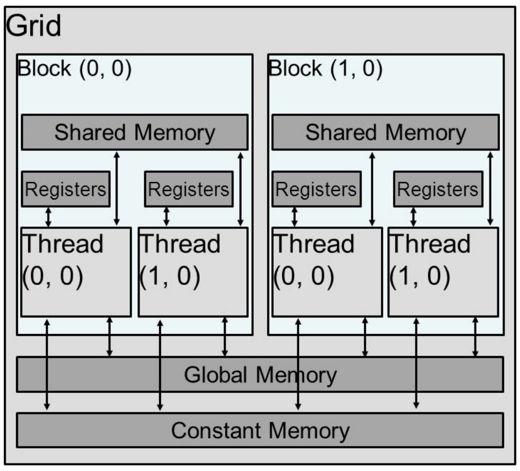

Figure 1: Schematic overview of CUDA memories. Shared memory is on the SoC while the global memory are accessed on the GPU through a bus.

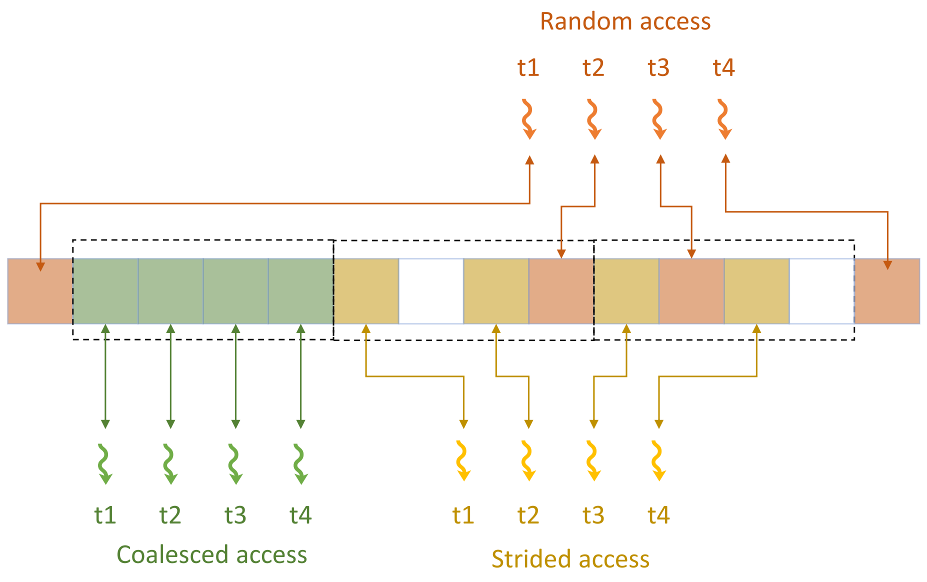

Figure 2: This figure illustrates the difference between coalesced, strided and random memory access. Memory blocks are indicated by the dotted lines. The green threads are coalesced and access memory all from the same memory block. The yellow threads are strided. The red threads are random. Coalesced threads will read to the best performance.

2 Assignment

2.1 Shared memory and coalescing

Create a kernel to transpose a matrix. Research the syntax for shared memory access. Investigate if there is a gain between an implementation using global or shared memory. Furthermore, investigate if your kernel uses coalesced memory access. If not, optimized your kernel and investigate performance gains.

2.2 Optimizing grayscale

Adapt your grayscale-kernel to use shared memory instead of global memory. Investigate performance gains or losses. Explain these differences. Furthermore, investigate if your grayscale kernel is coalesced (either in global or shared memory). If not, optimize your kernel so its memory access is coalesced.

3 Goals

- Research the memory architecture for CUDA GPUs.
- Write a working program and test it excessively.
- Gain insight in proper memory addressing and CUDA programming.

4 Report

Every group writes a report. The following elements must be present in the report:

- Analyse the problem: What are we trying to research? What is the goal? Why is this useful?
- Solution: The solution consists out of two elements: a flowchart and code. The flowchart must explain the code clearly.
- Testruns: Clearly show your program works by means of screenshots or output.
- Analyse the results: Create graphs as necessary and link them to the CUDA architecture.
- Conclusion: What did you achieve? What works? What doesn't work? What could you improve? Why does it work and what are the lessons learned.