GPU Programming 2018/19

Assignment 2

Due: 04/01/2019, 22:00

In this assignment, you will implement reduction, which on data-parallel architectures is one of the most important algorithms, in CUDA. Recall that reduction determines

$$r = a_0 \otimes a_1 \otimes \cdots \otimes a_{n-1} \tag{1}$$

for an input array $a = \{a_0, \cdots, a_{n-1}\}$ and a binary, associative operation $\otimes$ using a tree based algorithm that hierarchically combines two adjacent elements. For simplicity, the binary operation will in our implementation be addition and $a_i \in \mathbb{Z}$, i.e. we will compute the sum of a list of integers. We furthermore assume that the result of the reduction is required on the host so that it is admissible to perform some computations there.

Your tasks are as follows:

1.) Download the skeleton code and generate the build system using `cmake`.

2.) Implement a serial version of Eq. 1 that computes a reference solution. (1 Point)

3.) Complete `version1()` and `reduction1()` so that they implement a version of reduction that directly operates on global memory and that provides the correct result for at least $n = 16384$. Your implementation should use a number of threads that is appropriate for the chosen input size and that exploits the available parallelism on the device. (3 Points)

4.) Complete `version2()` and `reduction2()` so that they implement reduction using shared memory for at least $n = 16384$. Document the speed up that this version provides compared to the global memory one with a graph where the performance of both is plotted as a function of n. (3 Points)

5.) Complete `version3()` and `reduction3()` so that they implement a version of reduction that uses shared memory and that avoids divergent branches to the extent possible. Document the performance as in the previous question. (3 Points)

Bonus: Use templates to implement a version of reduction where all loops are unrolled and branching is minimized. Again document the performance as in the previous questions and if necessary document your implementation in the code. (3 Points)

Submission: Please submit your implementation as well as the graphs before the deadline to gpucourse@isg.cs.ovgu.de. Your code has to compile and run with the given CMake file.

Plagarism policy: You are free to discuss the assignment with your classmates and consult available resources (books, websites, ...). However, every student writes her / his own implementation. Violations are handled according to the study regulations.