Parallelle beregninger - TDT4200 - Problem Set 4

Ola Kristoffer Hoff

18. October 2022

1 CUDA implementation

Serial CUDA

Done, se code.

Parallel CUDA

1-4 done, see code.

5 How much speedup did you get for this parallelized CUDA solution compared with the serial implementation? Optionally, how does this compare with your solutions from previous assignments?

In table 1 we can see the benchmark results. Here we can see a speedup of 0.55, 1.82 and 5.56, respectivly for the different "n"-sizes. For $n = 1024$ we actually had a decrease in speed, probably due to to much overhead compared to the workload. However, both $n = 4096 \wedge 16384$ gave a good amount of speedup and it seems to increase further with the workload size.

Program \n	1024	4096	16384	Speedup
Serial	1.638s/1.386s/0.024s	5.852s/5.496s/0.036s	23.065s/22.569s/0.025s	1.00x/1.00x/1.00x
CUDA	2.964s/1.795s/0.881s	3.219s/2.027s/0.849s	4.152s/2.545s/1.109s	0.55x/1.82x/5.56x

Table 1: Table showing speed of execution with serial and CUDA. The entries are in the format "real/user/sys" and is generated from the *"benchmark_solution.sh"*-script. The speedup column is show the serial "real" result divided by the CUDA "real" result, and it respectivly show it for n = 1024, 4096 and 16384.

2 Questions

1 Why is it necessary to seperate the time_step function into two distinct kernals?

From what I can gather it is because the second part uses values calculated in the first part. This means that the whole first part must be executed before the second part and to ensure this we must synchronize between them so no threads start to use the old values and not the new ones.

2 In which scenarios could a CUDA implmentation outperform a CPU-based implementation?

The requirements for even bothering to implement something in CUDA should be as follows: the problem must be parallelizable so that there is a benfit to execute the code on the GPU and not the CPU. Also the speedup gained from using the GPU must be greater than the overhead of sending data between the CPU and GPU. So as we saw in this problem set, when the size of the problem grew the time gained by using the CPU was became greater than the overhead and we had pretty significant speedups from using CUDA.

3 In which scenarios could a CUDA implmentation be inferior to a CPU-based implementation?

This was answer indirectly in the previous question. So I just refer you to that. In short for smaller problems and problems with low paralleizablility, CUDA will most likly be inferior to just running it on the CPU.