Exercise 10

360.252 - Computational Science on Many-Core Architectures
WS 2022/23

December 23, 2022

The following tasks are due by 23:59pm on Tuesday, January 17, 2023. Please document your answers (please add code listings in the appendix) in a PDF document and submit the PDF together with the code in TUWEL.

You are free to discuss ideas with your peers. Keep in mind that you learn most if you come up with your own solutions. In any case, each student needs to write and hand in their own report. Please refrain from plagiarism!

"Anticipatory plagiarism occurs when someone steals your original idea and publishes it a hundred years before you were born."
— Robert Merton

There is a dedicated environment set up for this exercise:

`https://k40.360252.org/2022/ex10/`
`https://rtx3060.360252.org/2022/ex10/`

To have a common reference, please run all benchmarks for the report on both machines in order to see differences across GPU generations.

DIVOC simulator (7 Points + 2 Bonus)

The prediction of COVID-19 cases has been a prominent theme in news since 2020. This exercise deals with one approach to simulate the spreading of COVID-19. Since we make only very crude assumptions, we consider the simulation of a hypothetical *disease of very immediate concern (DIVOC)* virus.

In a loop over each day of the year, we model each Austrian individual and hence have to track 9 million individuals. In every iteration we check whether each individual is infected and if so, whether it transmits the virus. Further details can be found right in the code.

Please work on the following:

(a) The simulator includes a random number generator based on `rand()`. However, there is no simple `rand()` call available on the GPU. As a first step, generate a large enough sequence of random numbers and copy that over to the GPU for use as a random number pool by GPU threads. (1 Point)

(b) Implement a mechanism to generate random numbers within GPU threads based on pseudo-random numbers generated on the GPU[1]. Describe how your implementation works and justify your choice. Also, comment on the performance difference compared to a pre-generated sequence on the CPU as in (a). (1 Point)

(c) Port the simulator to the GPU using either CUDA, HIP, or OpenCL. Port the initialization phase (2 Points) as well as the simulation phase (2 Points). For each simulated day only the current number of infections should be communicated back from the GPU; all other data should remain on the GPU.

(d) Develop a simple performance model and compare it to the observed execution times. (1 Point)

(e) **Bonus**: Implement a non-trivial[2] refinement of your choice in the DIVOC simulator (CPU-only suffices, but GPU-implementations preferred). (1 Point)

[1]see for example `https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-37-efficient-random-number-generation-and-application`

[2]Simply changing existing parameters is considered trivial. It should be at least a few lines of code. Good ideas with little code are better than a lot of code implementing a bad idea. ;-)