Exercise 2

360.252 - Computational Science on Many-Core Architectures
WS 2022

October 12, 2022

The following tasks are due by 23:59pm on Tuesday, October 18, 2022. Please document your answers in a PDF document and email the PDF (including your student ID in the file name to get due credit) to karl.rupp@tuwien.ac.at.

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!

> "To steal ideas from one person is plagiarism;
> to steal from many is research." — Steven Wright

There is a dedicated environment set up for this exercise:

https://k40.360252.org/2022/ex2/

To have a common reference, please run all benchmarks for the report on this machine.

Basic CUDA (5 Points total)

Write a program that initializes two CUDA vectors (double precision floating point numbers) of length N: One consisting of the numbers

$$0, 1, 2, 3, \ldots, N-1$$

and the other consisting of the numbers

$$N-1, N-2, \ldots, 0\,.$$

Investigate the following:

(a) Measure the time to allocate (`cudaMalloc`) and free (`cudaFree`) a CUDA array for different sizes N. (1 Point)

(b) Compare the following two options to initialize the vectors: (1 Point)

- Initialize directly within a dedicated CUDA kernel
- Copy the data via cudaMemcpy() from a host array (e.g. from a malloc'ed array or from `std::vector<double>`).

(c) Write a CUDA kernel that adds the two vectors and writes the result to a third vector. Make sure that the kernel works for different values of N. (1 Point)

(d) Measure and plot the execution time of the kernel for different values of N (e.g. 100, 300, $1\,000$, ..., $1\,000\,000, 3\,000\,000$). What do you observe for small values of N and large values of N, respectively? (1 Point)

(e) Try different grid sizes and block sizes as kernel launch parameters. For simplicity, consider the values 16, 32, 64, 128, 256, 512, and 1024. Which values lead to a significant reduction in performance for large (N = 10^7) vectors? (1 point)

Dot Product (4 Points total)

Write a program that computes the dot-product of two vectors of variable size N in different ways:

(a) Use two GPU stages: In the first kernel, each thread block computes the partial dot-product and writes the partial result to a temporary array in GPU RAM. Then, a second kernel sums the obtained partial results in GPU RAM. (2 Points)

(b) Use a GPU and a CPU stage: In the first kernel, each thread block computes the partial dot-product and writes the partial result to a temporary array in GPU RAM. Then, copy the temporary array to the CPU and sum it there. (1 Point)

Compare the performance of the two versions. When is each of the three versions most appropriate? (1 Point)

Bonus: Early Submission

Submit your report by 23:59 on Monday, October 17, 2022. (1 Point)

Hints

- Example code is provided in the online enviroment. Use this as a starting point.
- The CUDA toolkit documentation[1] provides details on the CUDA API calls. The routines `cudaMalloc()` and `cudaFree()` are described in Section 6.9.
- A timer class is provided in the online environment. Be aware of statistical fluctuations in each measurement.
- A good version control system is very handy. :-)

[1] https://docs.nvidia.com/cuda/cuda-runtime-api/