Exercise 9

360.252 - Computational Science on Many-Core Architectures
WS 2020

December 16, 2020

The following tasks are due by 23:59pm on Tuesday, January 12, 2021. Please document your answers (please add code listings in the appendix) in a PDF document and email the PDF (including your student ID 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://gtx1080.360252.org/2020/ex9/.

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

OpenMP (2 Points)

Given vectors $x = (1, 1, \ldots, 1)$ and $y = (2, 2, \ldots, 2)$ of size N,

1. compute the dot product $\langle x + y, x - y \rangle$ with OpenMP for GPUs, (1 Point)
2. compare the execution times with your own CUDA and OpenCL implementations for values $N = 10^k$ with $k \in \{1, 2, 3, 4, 5, 6, 7\}$. (1 Point)

Dense Matrix Transpose (4 Points + 1 Bonus)

Your friend is working with a dense matrix $A \in \mathbb{R}^{N \times N}$ and needs to transpose the matrix in-place.

Inspired by your success in accelerating the multiple dot products, your friend writes a specialized CUDA kernel. Unfortunately, your friend's kernel doesn't produce correct results and you need to help out.

Please help your friend as follows:

1. Run your friend's kernel through `cuda-memcheck` and identify the problem(s). (1 Point)

2. Fix your friend's kernel to yield correct results, but do not optimize for performance. Determine the effective bandwidth (GB/sec) you achieve. (1 Point)
3. Use shared memory to load blocks of size 16×16, transpose them in shared memory, and write the result. Determine the effective bandwidth (GB/sec) you achieve. (1 Point)
4. Compare the performance you obtain for the non-optimized and the optimized kernel for $N = 512, 1024, 2048, 4096$.
5. **Bonus**: Come up with an OpenMP implementation that (ideally) matches the performance of your optimized shared memory kernel. (1 Point) *Full disclosure: Your lecturer does not know whether it is possible with OpenMP to match the performance of the optimized kernel; give it a try and see how fast you can make it.*

Feel free to use CUDA or OpenCL for tasks 2 and 3. For simplicity, assume that N is a multiple of 16.