Exercise 7

360.252 - Computational Science on Many-Core Architectures
WS 2022

November 25, 2022

The following tasks are due by 23:59pm on Sunday, December 11, 2022. 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!

> "Taking something from one man and making it worse is plagiarism." — George A. Moore

There is a dedicated environment set up for this exercise:

https://k40.360252.org/2022/ex7/
https://rtx3060.360252.org/2022/ex7/

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

Pipelined Conjugate Gradients (5 Points total)

In the lecture we discussed the following pipelined conjugate gradient implementation:

1 Choose x_0;
2 $p_0 = r_0 = b - Ax_0$;
3 Compute and store Ap_0;
4 $\alpha_0 = \langle r_0, r_0 \rangle / \langle p_0, Ap_0 \rangle$;
5 $\beta_0 = \alpha_0^2 \langle Ap_0, Ap_0 \rangle / \langle r_0, r_0 \rangle - 1$;
6 **for** $i = 1$ *to convergence* **do**
7 $\quad x_i = x_{i-1} + \alpha_{i-1} p_{i-1}$;
8 $\quad r_i = r_{i-1} - \alpha_{i-1} Ap_{i-1}$;
9 $\quad p_i = r_i + \beta_{i-1} p_{i-1}$;
10 $\quad$ Compute and store Ap_i;
11 $\quad$ Compute $\langle Ap_i, Ap_i \rangle$, $\langle p_i, Ap_i \rangle$;
12 $\quad$ Compute $\langle r_i, r_i \rangle$;
13 $\quad \alpha_i = (r_i, r_i \rangle / (p_i, Ap_i \rangle$;
14 $\quad \beta_i = \alpha_i^2 \langle Ap_i, Ap_i \rangle / \langle r_i, r_i \rangle - 1$;
15 **end**

Implement this algorithm in CUDA using double precision arithmetic and data types. Make sure only two kernels are launched per iteration. (3 points)

A reference implemenation of a classical conjugate gradient implementation can be found at the URL above for reference. Feel free to use the initial guess $x_0 \equiv 0$.

Finally, compare the execution time per iteration for different unknowns (10^3 to about 10^7) for the classical implementation and your pipelined implementation. Which performance benefits do you identify? (1 Point per GPU)