Homework 5

CAAM 520, Spring 2019

Posted April 12, 2019. Due April 19, 2019 by 5pm.
You may turn this assignment in without penalty until 5pm on April 24.

1. Your solutions to the homework must be committed to your Github repository in a sub-directory HW05.
2. You are required to include a Makefile that creates an excecutable called "hw05' in that directory. All source code, header files, LaTeXfiles, Makefiles, etc. should be committed to your repository in the same sub-directory.
3. Use LaTeX to write and typeset your report (saved as "report.pdf" in your repository sub-directory). Document all your steps, including code snippets within your report.
4. You may only consult the instructor for assistance, but are encouraged to use textbooks and internet resources. Cite all external resources used via footnotes or a bibliography.

Assignment: Your task is to create an OpenCL code which computes the solution $\boldsymbol{u}$ to the system $\boldsymbol{Au} = \boldsymbol{b}$ arising from the finite difference discretization of Poisson's equation using a pure Jacobi iteration (weight equal to 1). You may use the change in the solution update as a stopping criterion. Use single precision for all computations (`float` instead of `double`).

Your code should include the following components:

1. separate host code and kernel codes.
2. at least two OpenCL kernels: one to perform one Jacobi iteration, and another to compute the stopping criteria using a reduction.

Your code should target a GPU implementation, and should follow guidelines on the previous assignment on efficient GPU implementations (minimizing CPU-GPU transfers, coalesced memory transfers, etc). Due to restrictions on the number of threads for OpenCL implementations on CPUs, the most optimized reduction kernel may not work properly for CPU computations. You may use another version of the reduction kernel for this case.

Documentation: Your report should include the following:

1. documented verification of your code's correctness through reported errors and comparison with serial code and GPU code (provided online).
2. runtimes (using OpenCL event timing) for all kernels involved in your implementation.

3. runtimes for your CPU and GPU (CUDA) codes for comparison. For the CPU code, please provide total runtime, and for the CUDA GPU code, please provide timings for the Jacobi and reduction kernels.

4. discussion of any discrepancies between the runtimes observed for your OpenCL implementation and for the provided CPU and GPU (CUDA) implementations.

Extra credit (10 pts): produce an OCCA implementation of the finite difference Jacobi iteration, and compare the results on a CPU and a GPU. Note that, in contrast to OpenCL, your OCCA implementation of the optimized reduction kernel should work on a CPU (so long as you use at least 32 threads).