Parallel and Distributed Processing
Assignment 04
GPU Programming Using OpenCL and CUDA

Attique Dawood

23-04-2025
Due: 04-05-2025 2359 Hours
Last Modified: Thursday 24th April, 2025, 07:26

Practice: Create OpenCL and CUDA versions of 1-D and 2-D matrix addition. Compare speed-up against CPU single-threaded and multi-threaded (OMP) implementations. Also compare against CPU version for correctness by summing the difference of CPU and GPU versions (sum should be zero). For debugging, you can check if difference of individual indices is zero (or not).
Question 1: Create OpenCL and CUDA versions of naive/simple matrix multiplication. Compare against single-threaded and multi-threaded (OMP) versions for performance and correctness. Justify your choice of workgroup size(s).
Question 2: Create OpenCL and CUDA versions of Laplace solver. For boundary conditions, use $top = 5$ V, $bottom = -5$ V, $left = right = 0$ V. This is effectively a capacitor. Compare against single-threaded version for performance and correctness. Justify your choice of workgroup size(s).
Challenge: Create OpenCL and CUDA versions of Laplace solver that utilises shared/local memory. Compare against the regular GPU version for performance and correctness.
Extra Challenge: If you have multiple devices, you can utilise OpenCL to do *heterogeneous computing*. This will divide the problem to be run on two devices simultaneously. Could be any combination of CPU/GPU, CPU/Integrated GPU, etc. You'll have to use multi-threading to have two devices running in parallel.
Note 1: Please do mention the hardware specs and plot graphs to show performance speed-up using single-threaded CPU implementation as the gold standard.
Note 2: Appropriately package your submission so that the end-user can easily compile and run your code (makefile, cmake, VS build, etc.).