ECE277 Final Project Proposal

Henrik Larsson Hestnes

November 2021

GPU accelerated 2D matrix multiplication with Pybind11

My plan is to make a simple console/terminal frontend in Python where the user can specify the dimensions of two 2D matrices, and if desired the user can also specify the elements of the matrices. I want to make the matrix dimensions configurable during runtime, to make the code more flexible to use. Python will thereafter make the matrices using NumPy, and do a call to a binded C++ function, which will be binded using Pybind11. The binding will consist of one C++ module with one or more C++ function to make it possible to compare different CUDA implementations. This call will pass the matrices along with information about the matrix dimensions from Python to C++. The C++ function will then allocate the needed memory on the host and device using CUDA, and then transfer the data to the device. After that it will invocate the CUDA kernel. Since the matrix multiplication is of 2D matrices, the dimension of both the grid and thread-blocks will be arranged in 2D. The kernel then does the matrix multiplication both via global and shared memory. The advantage of shared memory is that it can speed up the multiplication by reducing the amount of global memory accesses. If time allows I will also try to utilize the shared memory to avoid uncoalesced global memory accesses. After that, the data is transferred back to the host, and the memory on the device is deallocated. The Python function can then read the new matrix, and the execution time for the different implementations will be printed to the screen. If desired the resulting matrix will also be printed to the simple GUI.