CUDA Streams

Class: CS315 – Distributed Scalable Computing
Whitworth University, Instructor: Scott Griffith

Last Modified: 11/8/2018

Part 1: Two Dimensional Matrix Streams (50 pts)

This is going to be added on to LM6. Previously you implemented a matrix multiply function. For this learning module you are going to use that kernel (along with a matrix add kernel) to experiment with streams.

First you are going to need to implement a square matrix addition kernel. This kernel should take in two matrices, add each element and store them in an output matrix. This should be pretty straightforward after writing the multiply kernel.

Make sure you test this kernel before implementing the next part.

Once you have that done, you are going to put it all together. You are going to make a total of four input matrices, you are going to compute two intermediate products, then add them together to find your resulting answer. In total your matrices should **fully** (or as close to fully as you can get) **occupy your devices memory**.

Each matrix should operate in its own stream, this means you are going to have to utilize some cudaEvents to synchronize between kernels.

You also should mask some of your memory transfers, in particular at least one transfer from host to device at the start. This is done by utilizing different streams and asynchronous memory copies.

10 pts: Is there a benefit from using streams? Run your code on one stream and report your total memory copy and computation time, then compare to running streamed code. Some things should be different, others should be the same. What does it mean if the computation time is the same either way? Be careful about timing events with multiple streams.