David Han
dth2126

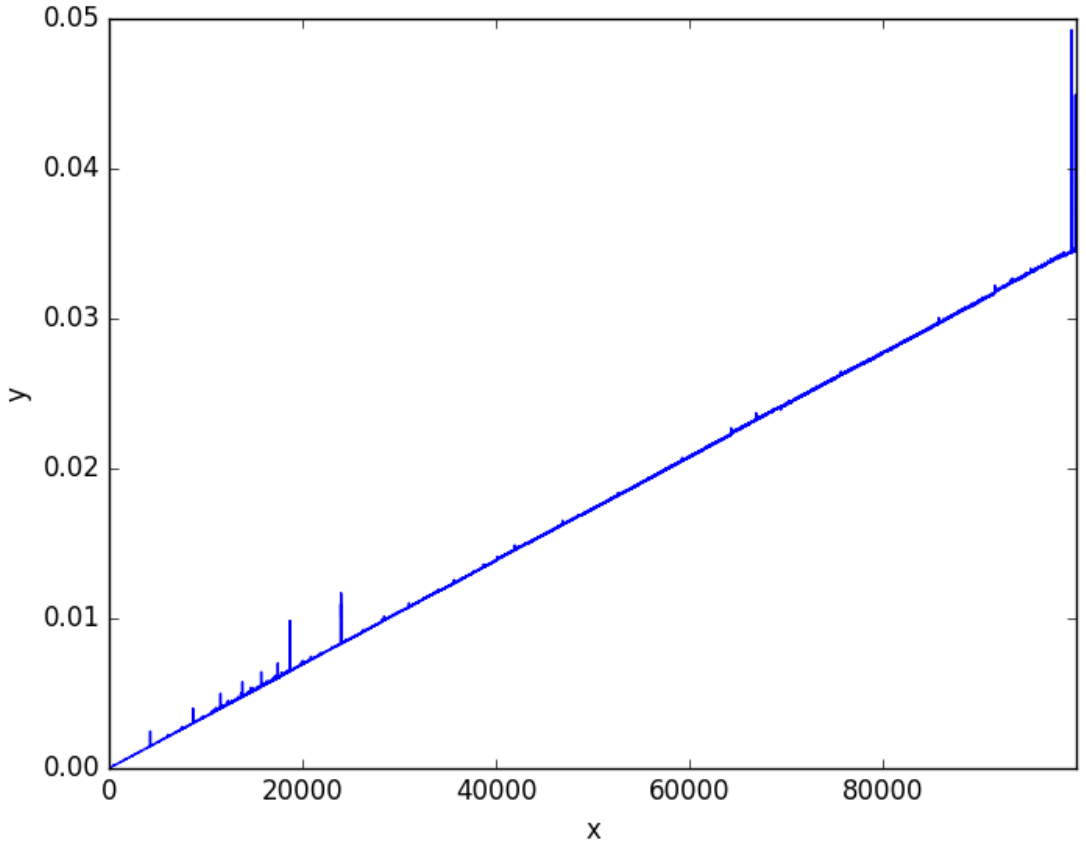
Assignment 1

Theory:

1. A process is an instance of a computer program that is being executed. A thread results from a fork of a computer program into two or more concurrently running tasks. Multiple threads can exist within the same process and share resources. A tasks a set of program instructions that are loaded into memory.
2. Concurrency means that an application is making progress on more than one task but it cannot be run at the same time as there is only one CPU and context switching or switching between different tasks can be done, but only one task can be run.
   Parallelism means that an application splits its tasks up into smaller subtasks which can be processed in parallel, for instance on multiple CPUs at the exact same time.
3. Replacing a GPU with a CPU is a bad idea because it does not allow for parallelism to be done. A CPU cannot execute multiple threads at the SAME time. It can run multiple threads but only complete work on one only. A GPU allows for multiple or several cores to run work all at the same time, which allows for things like graphics on video games to be run so efficiently and smoothly today.
4. Advantage of Python include managed memory, easy to use standard libraries.
   Disadvantages is that it's slower than C/C++ because it's not compiled but interpreted.

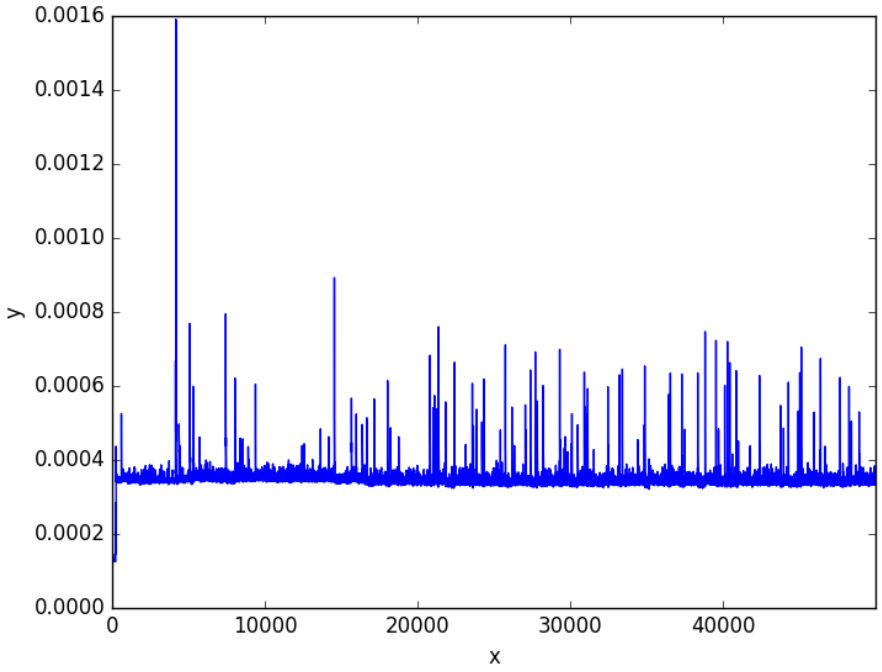
Comments on CPU, PyOpenCL, and PyOpenCUDA:

Plot for CPU (hash.py):



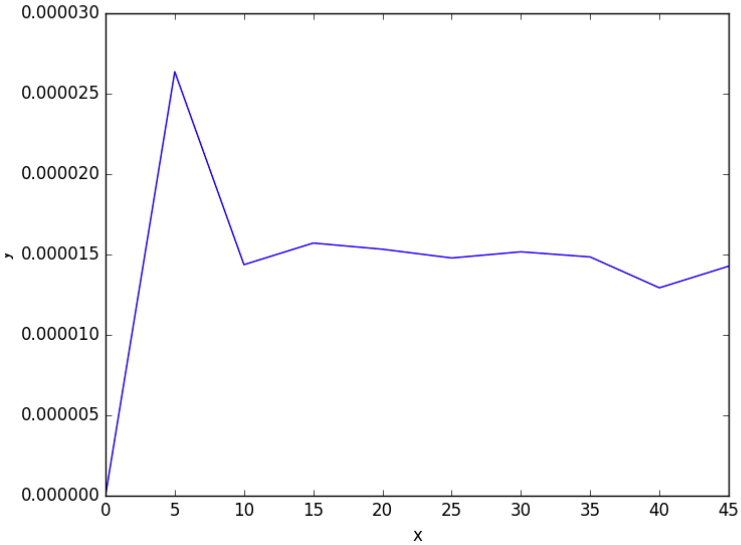
For the CPU plot, it was obvious that as the input length got larger, the amount of time also got longer. I used time library from python. Max input length was 50,000 iterations.

Plot for GPU in OpenCL using time library from python:

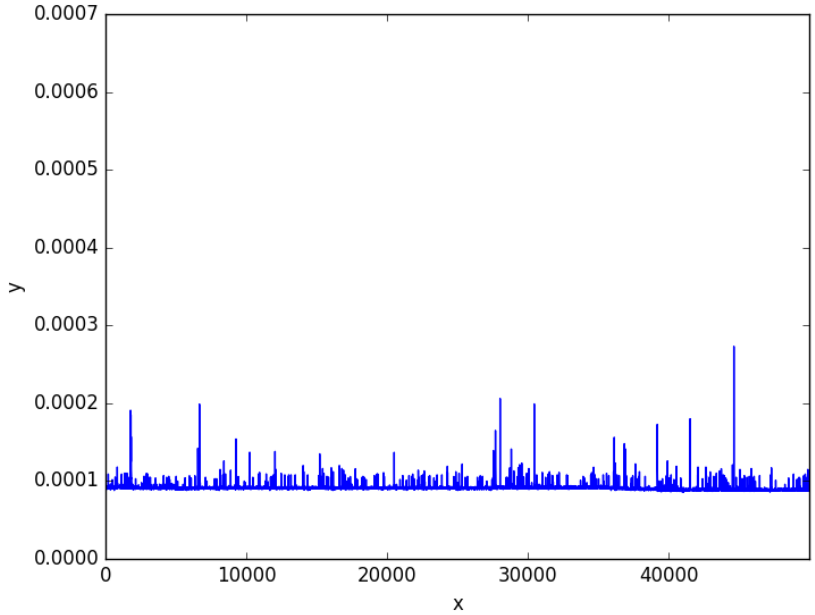


For the OpenCL plot, it leveled out around 0.0004 units which I assumed were seconds. This is definitely faster than the CPU plot. Max input length was 10,000 iterations.

However, in the beginning, I see that it had took surprisingly longer for smaller input lengths. This could be due to how there is some initial overhead or latency that is needed to move data between the different caches and areas for the GPU.



Plot for GPU in CUDA using time library from python:



For the CUDA plot, it leveled out around 0.0001 units which I assumed were seconds. This is definitely faster than the CPU plot and seems faster then OpenCL. Max input length was 10,000 iterations.

However, in the beginning, I see that it had took surprisingly longer for smaller input lengths. This could be due to how there is some initial overhead or latency that is needed to move data between the different caches and areas for the GPU.