# HW2 - Accelerators and Accelerated Systems

Shay Agroskin 305664039
Tomer Abraham 316219849

June 9, 2019

## Stream implementation

a. Run the program in streams mode with load = 0 (unlimited rate) and report the throughput in the report (will later on be refered as maxLoad.

   **A:** The thorughput is $25782.950\frac{req}{sec}$

b. vary the load from load from $\frac{maxLoad}{10}$ to $maxLoad * 2$ in $\sim 10$ equal steps. In each run write down the load, latency and throughput in a table in a report.

   **A:**

| $Load\left[\frac{req}{sec}\right]$ | $Throughput\left[\frac{req}{sec}\right]$ | $Latency\,[msec]$ |
|---|---|---|
| 2578.295 | 2563.965530 | 0.047707 |
| 7477.055 | 7467.483535 | 0.051045 |
| 12375.815 | 12272.541976 | 0.051699 |
| 17274.575 | 17079.256141 | 0.052335 |
| 22173.335 | 19529.621886 | 0.053035 |
| 27072.095 | 23030.351666 | 0.053453 |
| 31970.855 | 23320.037872 | 0.057989 |
| 36869.615 | 25371.636216 | 0.059060 |
| 41768.375 | 25717.055695 | 0.060661 |
| 46667.135 | 26023.707617 | 0.081855 |

Table 1: latency, load, thorughput

c. Show the sample in a graph and explain it

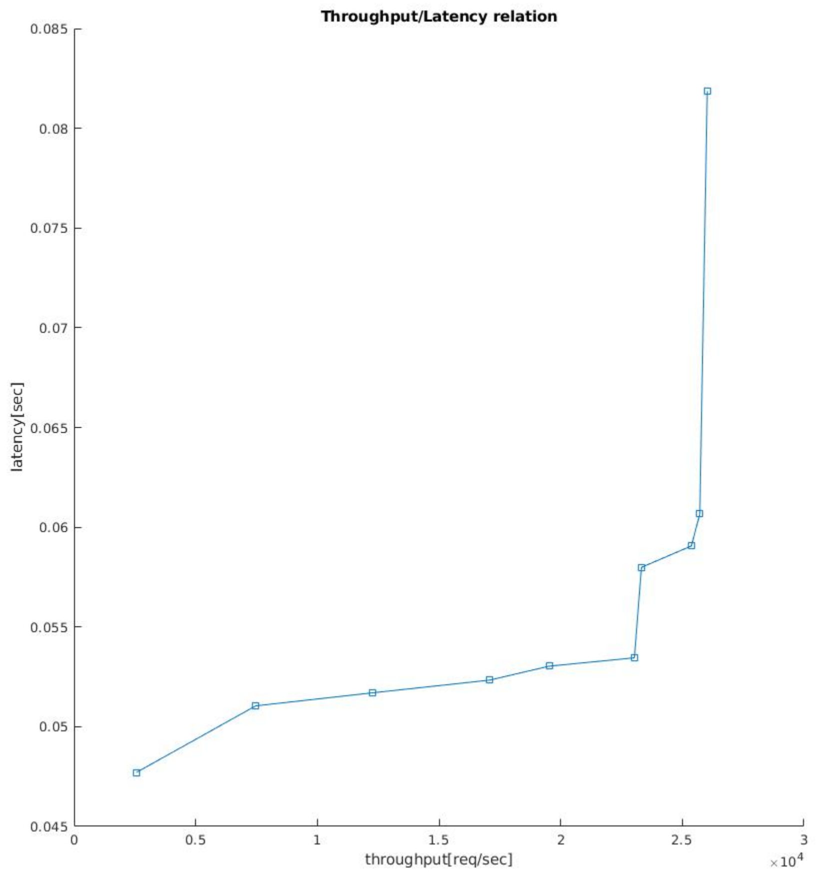

Figure 1: Latency-Throughput graph streams implementation

As we can see from the graph, the more load (and thus thorughput) we add to the system, the longer the latency is.

## Producer Consumer Queues

a. How many thread blocks can we run ?

**A:** For a single $SM$ the amount of thread-blocks we can run is the minimum of the following values:

1) Maximum number of threads per SM divided by the number of threads we run in a single thread-block
2) Max amount of shared memory per SM divided by the shared memory a single thread-block uses.
3) Max number of registers per SM divided by the number of registers each thread-block uses (number of registers per threads times number of threads in thread-block).

We multiply the number of thread-blocks per SM by the number of SMs in the device and get total number of thread-blocks that we can run.

b. Run the program in queue mode with $\#threads = 1024$ and $load = 0$ and report throughput in the report. We'll refer to the throughput you get here as maxLoad.

**A:** $maxLoad = 147962.25511 \left[\frac{req}{sec}\right]$

c. vary the load from load from $\frac{maxLoad}{10}$ to $maxLoad * 2$ in $\sim 10$ equal steps. For $nr\_threads = 1024$

**A:**

| $Load\left[\frac{req}{sec}\right]$ | $Throughput\left[\frac{req}{sec}\right]$ | $Latency\,[msec]$ |
|---|---|---|
| 14797.225 | 14861.8229 | 0.1649 |
| 42911.952 | 42665.791 | 0.1747 |
| 71026.680 | 70006.47 | 0.2021 |
| 99141.408 | 97015.346 | 0.2224 |
| 127256.135 | 123322.216 | 0.3010 |
| 155370.863 | 145165.337 | 0.647 |
| 183485.591 | 147490.64 | 1.527 |
| 211600.319 | 147568.7602 | 1.5612 |
| 239715.0468 | 148161.131 | 1.56849 |
| 267829.7745 | 149344.631 | 1.5517 |

Table 2: latency, load, thorughput, nr_threads=1024

d. Same but for $nr\_threads = 256$

| $Load\left[\frac{req}{sec}\right]$ | $Throughput\left[\frac{req}{sec}\right]$ | $Latency\,[msec]$ |
|---|---|---|
| 14797.225 | 14731.435520 | 0.483365 |
| 42911.952 | 42120.119159 | 0.564211 |
| 71026.680 | 68798.507347 | 0.568970 |
| 99141.408 | 95548.155026 | 0.770850 |
| 127256.135 | 121950.711285 | 0.779161 |
| 155370.863 | 147265.028092 | 1.009903 |
| 183485.591 | 157489.570207 | 1.501819 |
| 211600.319 | 164587.820191 | 1.800987 |
| 239715.0468 | 173768.992816 | 2.230139 |
| 267829.7745 | 177765.239077 | 2.483280 |

Table 3: latency, load, thorughput, nr_threads=256

e. same but for $nr\_threads = 256$

| $Load\left[\frac{req}{sec}\right]$ | $Throughput\left[\frac{req}{sec}\right]$ | $Latency\,[msec]$ |
|---|---|---|
| 14797.225 | 14904.436086 | 0.272244 |
| 42911.952 | 42074.867179 | 0.308531 |
| 71026.680 | 69426.331228 | 0.344472 |
| 99141.408 | 96724.897733 | 0.405332 |
| 127256.135 | 122132.890211 | 0.436986 |
| 155370.863 | 145342.684663 | 0.674260 |
| 183485.591 | 161502.451371 | 1.016936 |
| 211600.319 | 171306.614720 | 1.643606 |
| 239715.0468 | 173511.274437 | 2.550382 |
| 267829.7745 | 173990.995337 | 2.598047 |

Table 4: latency, load, thorughput, nr_threads=512

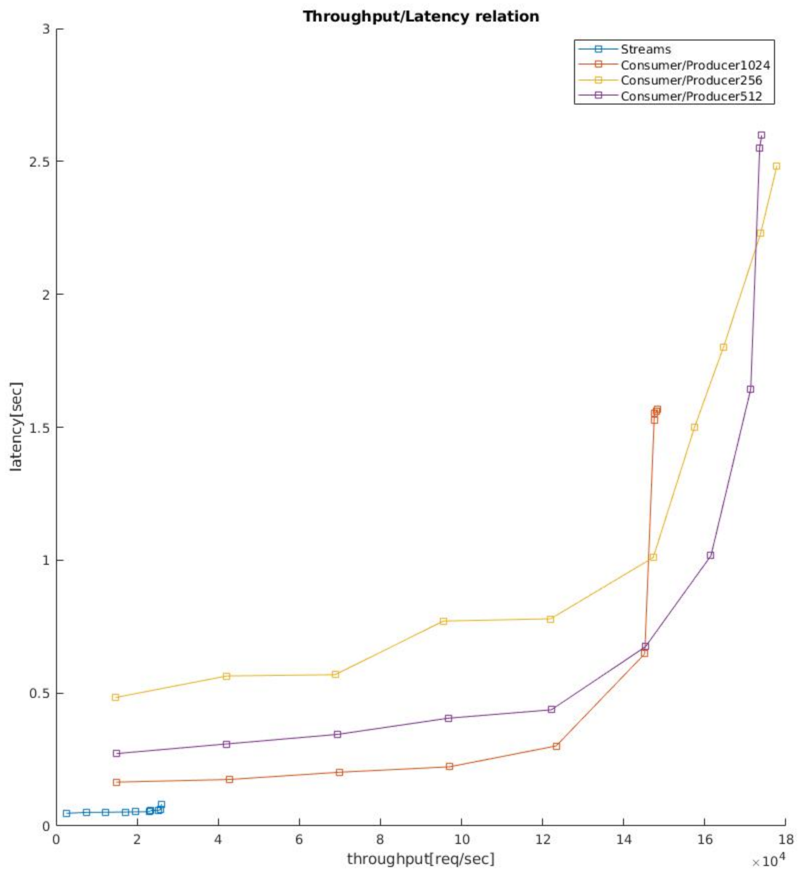

Figure 2: Latency, load, throughput, nr_threads=256,512,1024 and streams

f. Explain the difference in the graphs between different number of threads

**A:** As we can see, the more threads we run, the more throughput we get (some values may differ, but it's the general line).

g. A wise man suggested to move the CPU-to-GPU queue to the memory of the GPU (assume it is still accessible by the CPU), and to keep the GPU-to-CPU queue in the CPU memory. He claimed it might result in better performance. Explain why.

**A:** the input data (passed by the CPU-GPU queue) doesn't need to be accessed by the CPU after it is written, and hence it makes more sense to make it local for the GPU that can access this memory several times.

The same logic apply for the GPU-CPU queue where the GPU writes the data only once, where the CPU might access it several times. The improvement depends on the implementation. We might not get any improvements by it.

h. In order to place the CPU-to-GPU queue in the GPU memory, we will need to make it accessible by CPU. Explain roughly what should be done for this to happen (in terms of PCIe MMIO)

   **A:** We will map a virtual address to the device's memory by creating a map from address to PCIe address. This way, a CPU can make memory access to GPU's memory by accessing its virtual address.