Project #6 – OpenCL Matrix Multiplication

Young-Joon Park

parky8@oregonstate.edu

1. What machine you ran this on

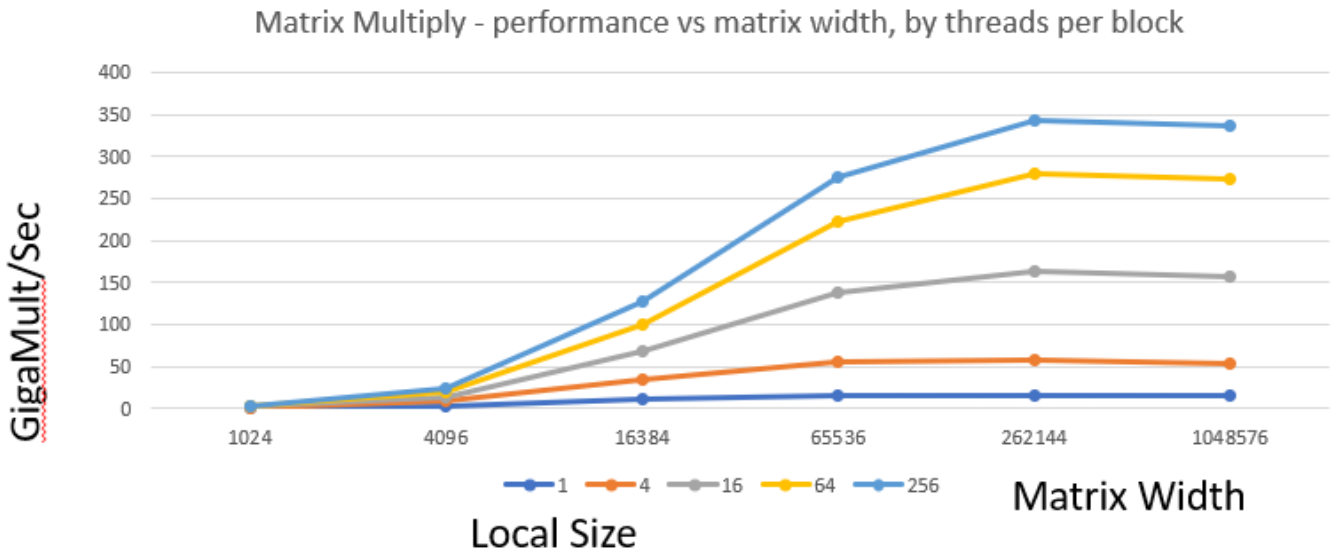
The simulation was run locally on a desktop PC with a Ryzen 5600 and an RTX 2080.

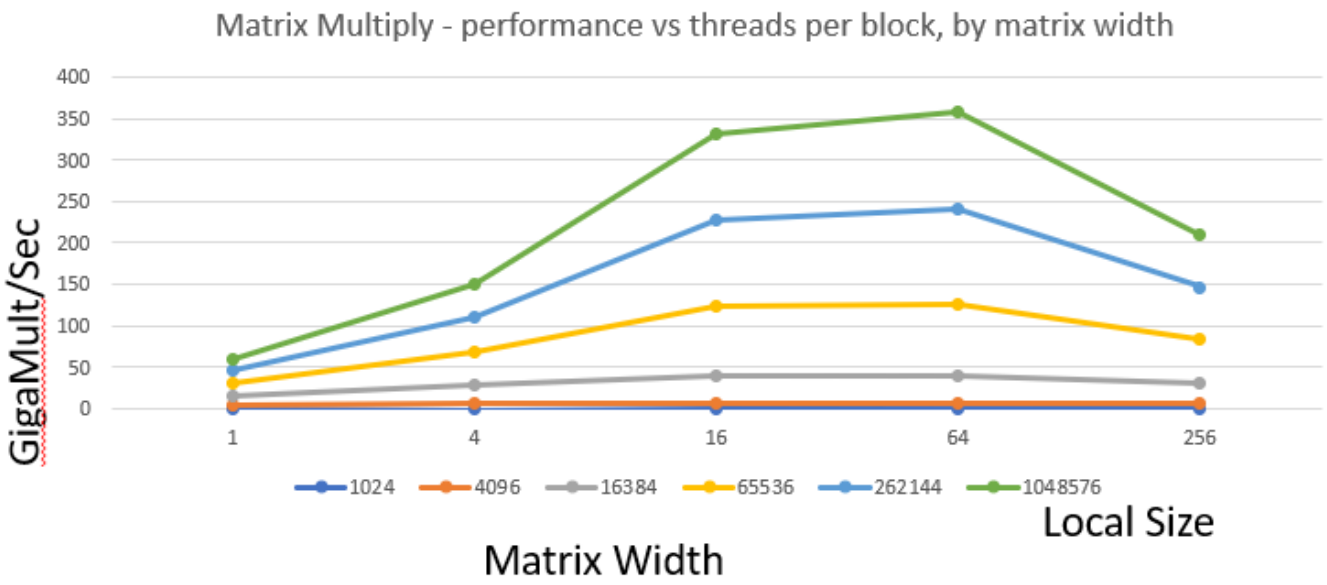
2. Show the table and graphs

(Local Size = LocalGroup * LocalGroup)

(matrix size = MATW * MATW)

	1	4	16	64	256
1024	0.65	0.55	0.7	0.69	0.63
4096	3.4	5.03	5.43	5.48	4.97
16384	11.57	22.03	33.99	33.35	25.45
65536	14.82	39.85	82.52	85.69	52.66
262144	14.93	43.26	105.15	116.52	63.1
1048576	14.38	38.9	104.59	115.45	63.1





3. What patterns are you seeing in the performance curves? What difference does the size of the matrices make? What difference does the size of each work-group make?

OpenCL's performance is not good in smaller matrix sizes. Only after a certain threshold, does its performance peak out.

The performance gains from local size increase incrementally decrease after a certain threshold, and in big numbers, the performance significantly decreases.

4. Why do you think the patterns look this way?

OpenCL's performance is not good in smaller matrix sizes. Only after a certain threshold, does its performance peak out.

>>> This is because there is overhead in setting up OpenCL. This overhead dominates the operation in smaller matrix sizes. If the matrix size is too small, there is simply not enough work for the GPU. Once there is a certain amount of data parallelism possible due to a larger matrix size, one can see high performance.

The performance gains from local size increase incrementally decrease after a certain threshold, and in big numbers, the performance significantly decreases.

>>> In the hardware level, operations on the GPU are conducted in warps. 32 threads make up a warp, and if there are less threads in a work-group, the unused remainder of the 32 threads are wasted. If there are too many threads in a work-group, the time to swap out work-groups could overshadow the benefits gained from GPU parallel computing.