CS 475/575 -- Spring Quarter 2021

Project #6

OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

Name: Chun-Yu, Chen

E-mail: chench6@oregonstate.edu

1. **Tell what machine you ran this on**

   Machine: **DGX**

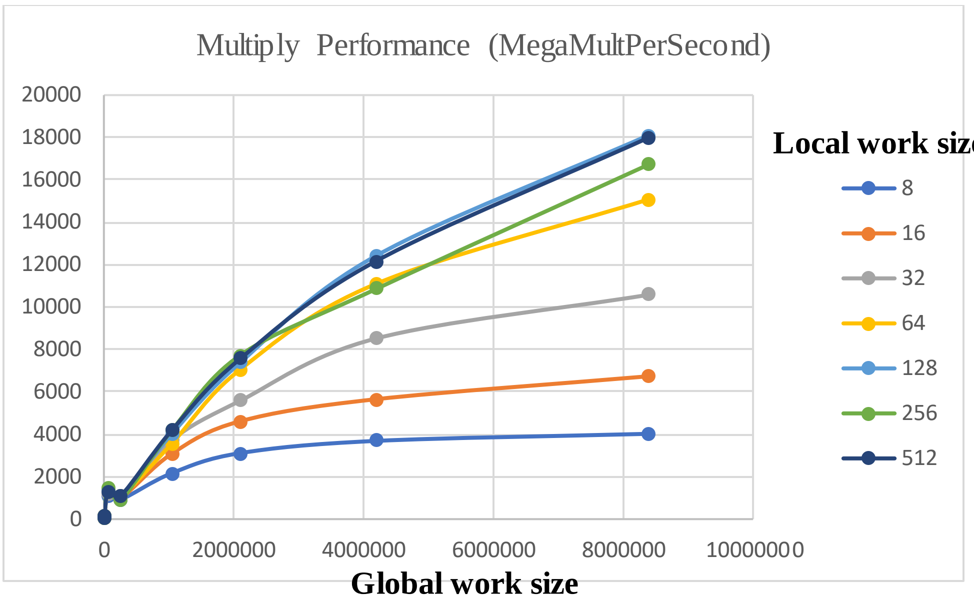
2. **Show the table and the graphs (Multiply & Multiply-Add)**

GLOBAL / LOCAL	1024	8192	65536	262144	1048576	2097152	4194304	8388608
8	18.7732	179.894	1096.91	930.096	2171.77	3113.57	3690.23	4024.41
16	23.3874	162.367	1256.61	1001.44	3091.82	4614.83	5657.56	6738.14
32	22.2821	183.573	1173.38	1056.75	3727.96	5578.62	8521.64	10579.5
64	23.2371	180.964	1386.74	1071.83	3571.97	7050.16	11100.5	15078.7
128	17.3866	181.221	1418.87	1121.1	4030.64	7376.35	12423.5	18106.1
256	21.1611	179.417	1461.24	923.35	4193.44	7723.63	10856.5	16721
512	20.6967	157.481	1275.91	1109.18	4200.21	7558.66	12173.3	17990.6

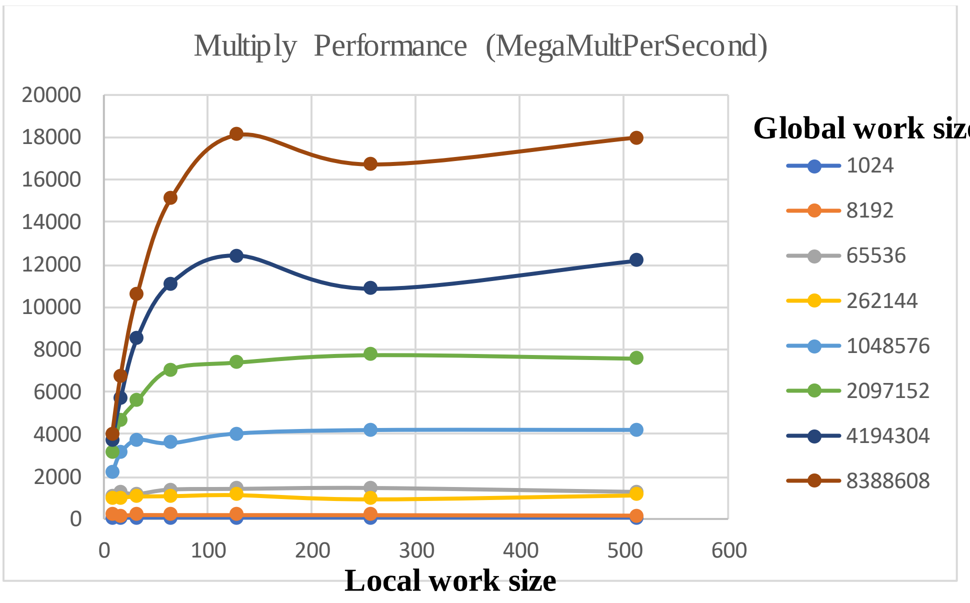
Unit: Mega Multiplies Per Second

GLOBAL / LOCAL	1024	8192	65536	262144	1048576	2097152	4194304	8388608
8	21.8935	170.917	1126.26	2611.79	2345.07	3129.65	3570.88	4002.54
16	19.9621	183.527	1213.44	3623.33	3085.43	4556.34	5848.22	6795.02
32	20.9435	159.399	1210.6	4573.48	3327.8	5966.91	8757.44	10517.3
64	21.7256	181.771	1421.71	5004	4047.02	7005.98	10798.9	14672.7
128	20.7447	162.755	1406.45	5383.89	4147.27	7138.64	10772.4	16095.8
256	22.6097	179.979	1416.61	5432.41	4062.71	7463.63	11913.2	15862.1
512	20.5805	180.792	1230.95	5343.92	4163.02	7526.42	11434.7	16005

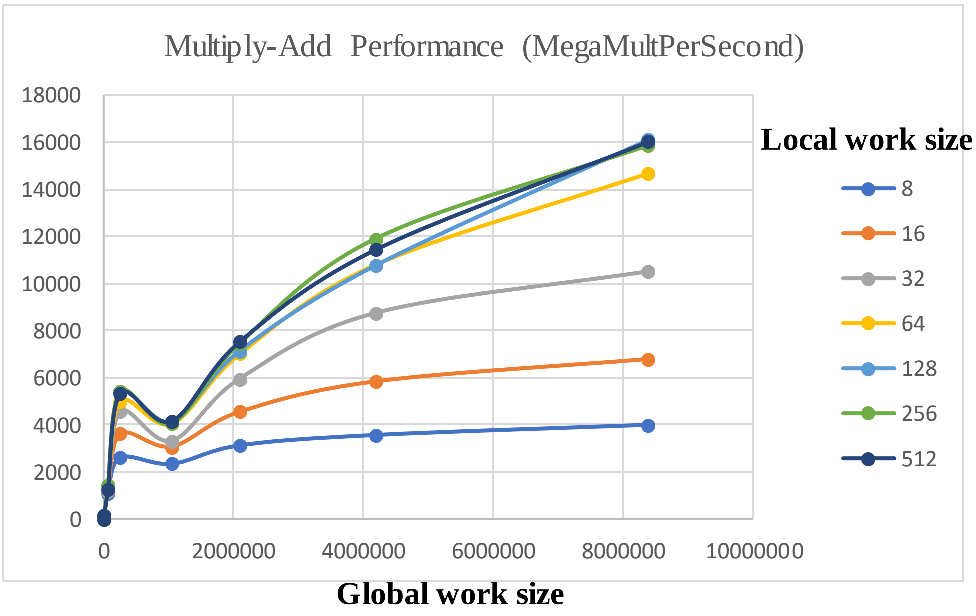
Unit: Mega Multiply-Adds Per Second



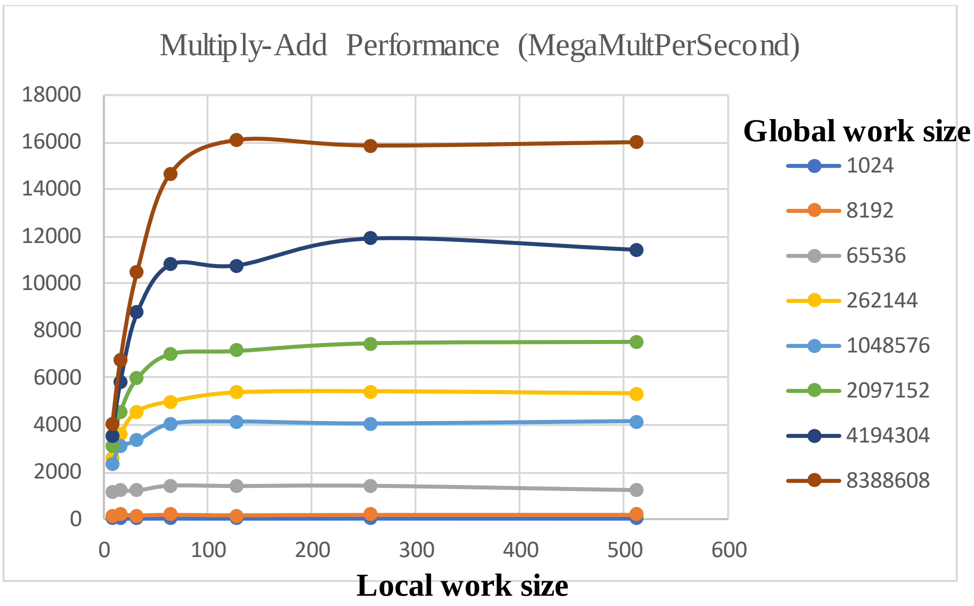
Multiply Performance (MegaMultPerSecond)
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
2000000
4000000
6000000
8000000
10000000
Global work size
Local work size
8
16
32
64
128
256
512



Multiply Performance (MegaMultPerSecond)
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
100
200
300
400
500
600
Local work size
Global work size
1024
8192
65536
262144
1048576
2097152
4194304
8388608



Multiply-Add Performance (MegaMultPerSecond)
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
0
2000000
4000000
6000000
8000000
10000000
Global work size
Local work size
8
16
32
64
128
256
512



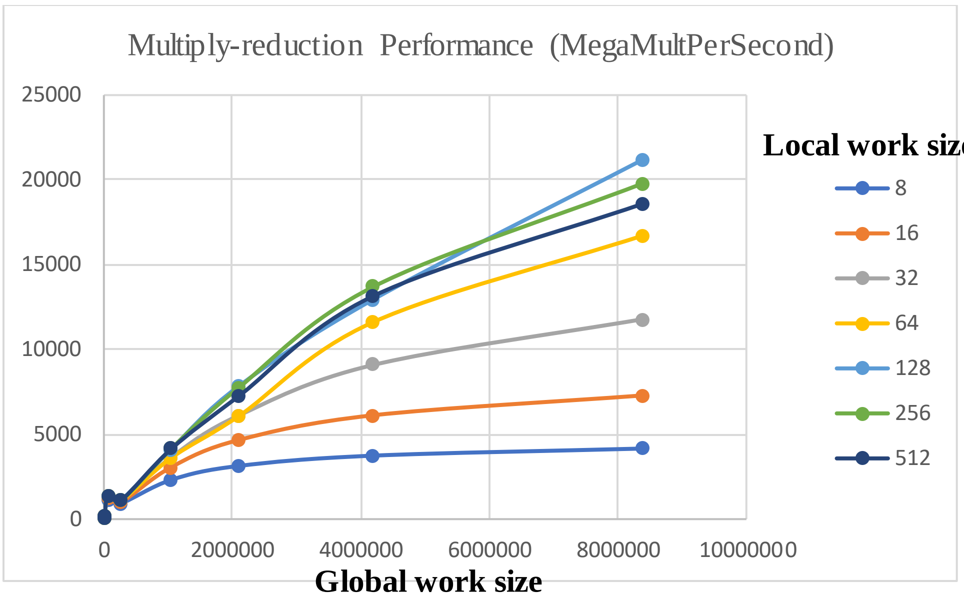
3. Commentary

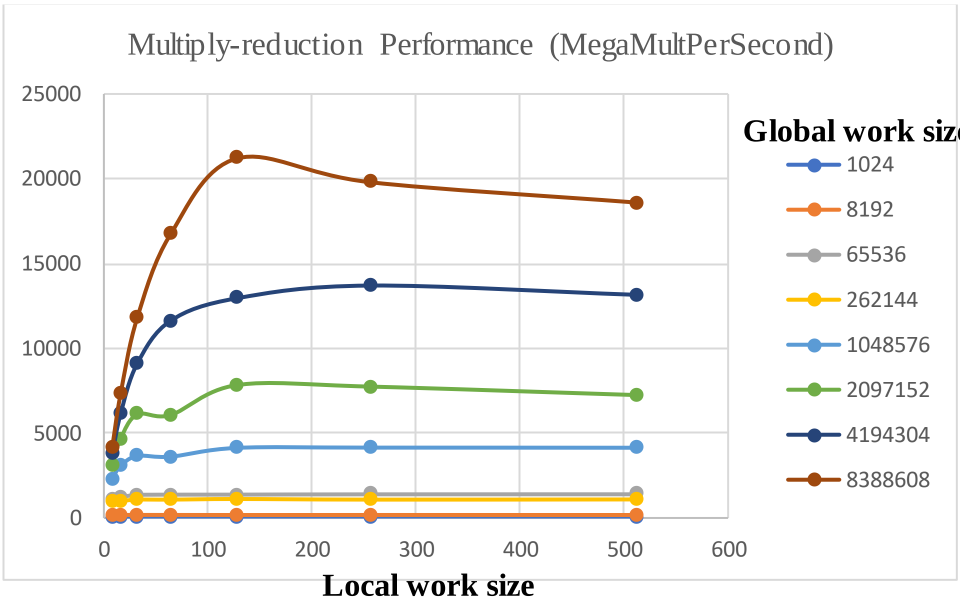
The result shows that we can get the far better performance while the global or local work group size increasing. In addition, it can be found that the performance of Multiply is a little better than the performance of Multiply-Add because Multiply was used in conjunction; however, Multiply-Add required the machine to switch gears between multiplying and adding. It charges a bunch of cost to do the extra operation. On the other hands, the result demonstrates that GPUs are good at handling huge data. If the workgroup is too small, it means that GPU is underutilized since there are many cores not being used. Furthermore, if there are too little data computed, lots of overhead are wasted in setting up the GPU for computation.

4. Show the table and the graphs (Multiply+Reduce)

GLOBAL / LOCAL	1024	8192	65536	262144	1048576	2097152	4194304	8388608
8	21.5904	157.698	1103.81	920.235	2316.47	3135.68	3744.55	4161.78
16	22.2984	174.89	1226.24	991.734	3067.77	4664.65	6125.28	7293.95
32	21.8991	177.907	1353.92	1081.02	3652.4	6106.41	9093.72	11789.5
64	10.8501	178.311	1374.09	1082.78	3613.34	6067.27	11620.9	16719.4
128	18.5662	174.232	1376.81	1118.72	4131.59	7834.8	12955.9	21198
256	21.1367	176.168	1393.52	1082.09	4148.29	7744.82	13702.4	19795.7
512	19.3617	174.702	1396.45	1093.56	4140.04	7244.87	13156.4	18595.3

Unit: MegaMultiply-Reductions Per Second





5. Commentary

The graphic pattern of the Multiply-reduction is similar to the above ones. The performance is getting better while the global or local workgroup size increasing. As far as I am concerned, the advantages of the parallelization on the GPU is offset by the performance penalty from doing the “CPU types of actions” like loop structures. I think GPU is not designed to do this kind of things effectively.