CS575(Introduction To Parallel Programming)

Project5

Project Title : OpenCL Array Multiply, Multiply-Add,and Multiply-Reduce(Project 6)

Name : Si Thu Lin

ID : 933-957-884

Email : linsi@oregonstate.edu

The code was run on the DGX.

The rabbit was also used during writing the code.

The Giga rather than Mega is used as the performance unit because Giga is also appropriate here.

In the zip file, the submit.bash,first.cpp and first.cl are for the Multiply and Multiply-Add.

The submit3.bash, third.cpp and third.cl are for the Multiply-Reduction.

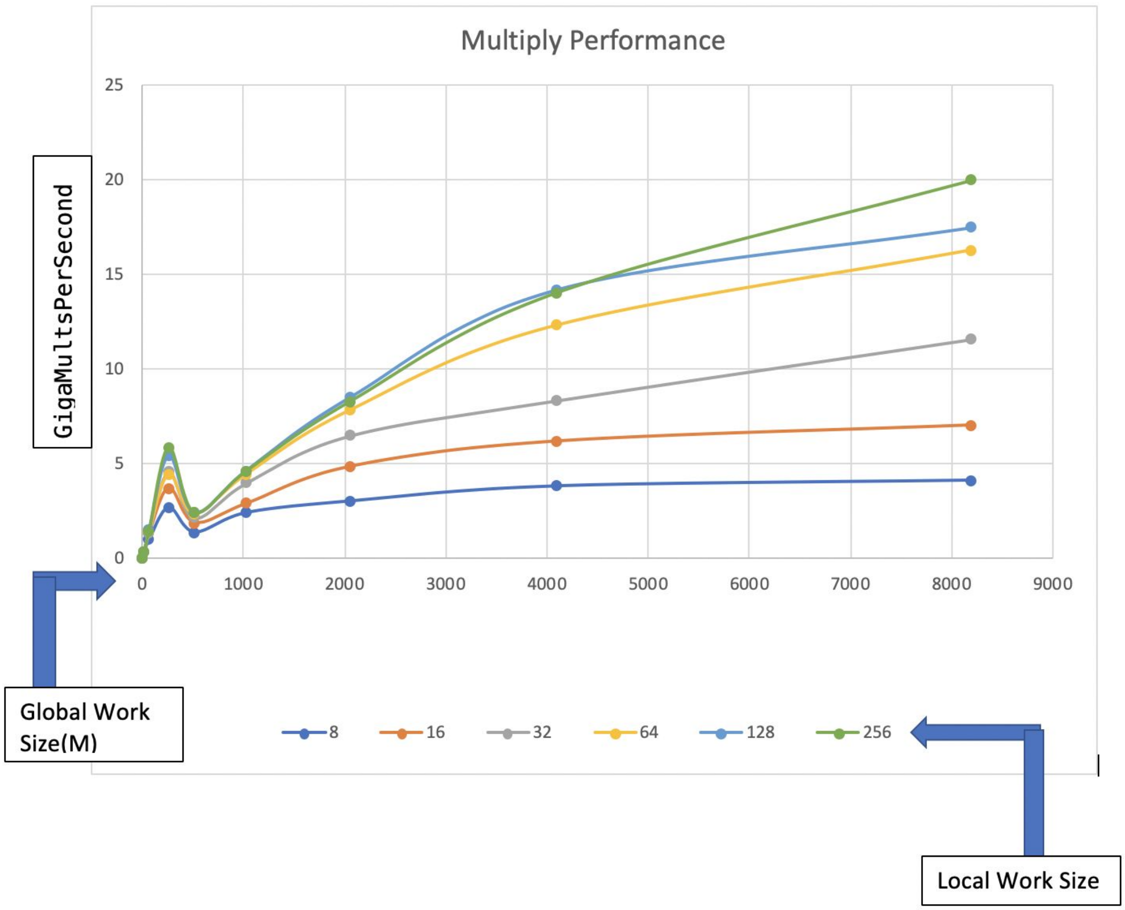
The Performance Table for the Multiply

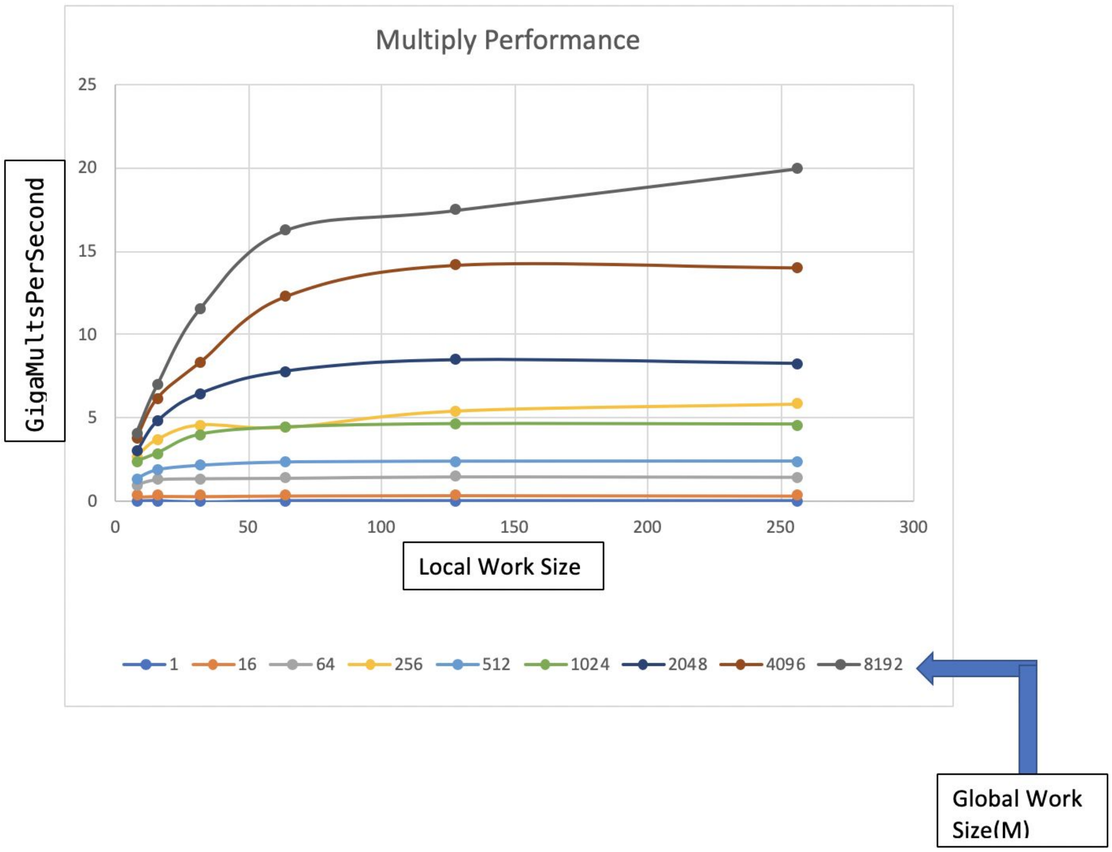
Local Work Size	Global Work Size(M)	Performance
8	1	0.022
8	16	0.344
8	64	0.98
8	256	2.67
8	512	1.375
8	1024	2.395
8	2048	3.01
8	4096	3.808
8	8192	4.113
16	1	0.023
16	16	0.358
16	64	1.313
16	256	3.722
16	512	1.889
16	1024	2.894
16	2048	4.854
16	4096	6.205
16	8192	7.054

32	1	0.018
32	16	0.355
32	64	1.375
32	256	4.582
32	512	2.155
32	1024	3.996
32	2048	6.489
32	4096	8.35
32	8192	11.58
64	1	0.023
64	16	0.366
64	64	1.405
64	256	4.423
64	512	2.348
64	1024	4.426
64	2048	7.811
64	4096	12.303
64	8192	16.269

128	1	0.023
128	16	0.373
128	64	1.499
128	256	5.424
128	512	2.401
128	1024	4.621
128	2048	8.492
128	4096	14.196
128	8192	17.483
256	1	0.023
256	16	0.365
256	64	1.473
256	256	5.85
256	512	2.409
256	1024	4.597
256	2048	8.276
256	4096	14.041
256	8192	19.978

The Performance Graph for the Multiply




Multiply Performance
GigaMultsPerSecond
25
20
15
10
5
0
0
50
100
150
200
250
300
Local Work Size
1
16
64
256
512
1024
2048
4096
8192
Global Work Size(M)

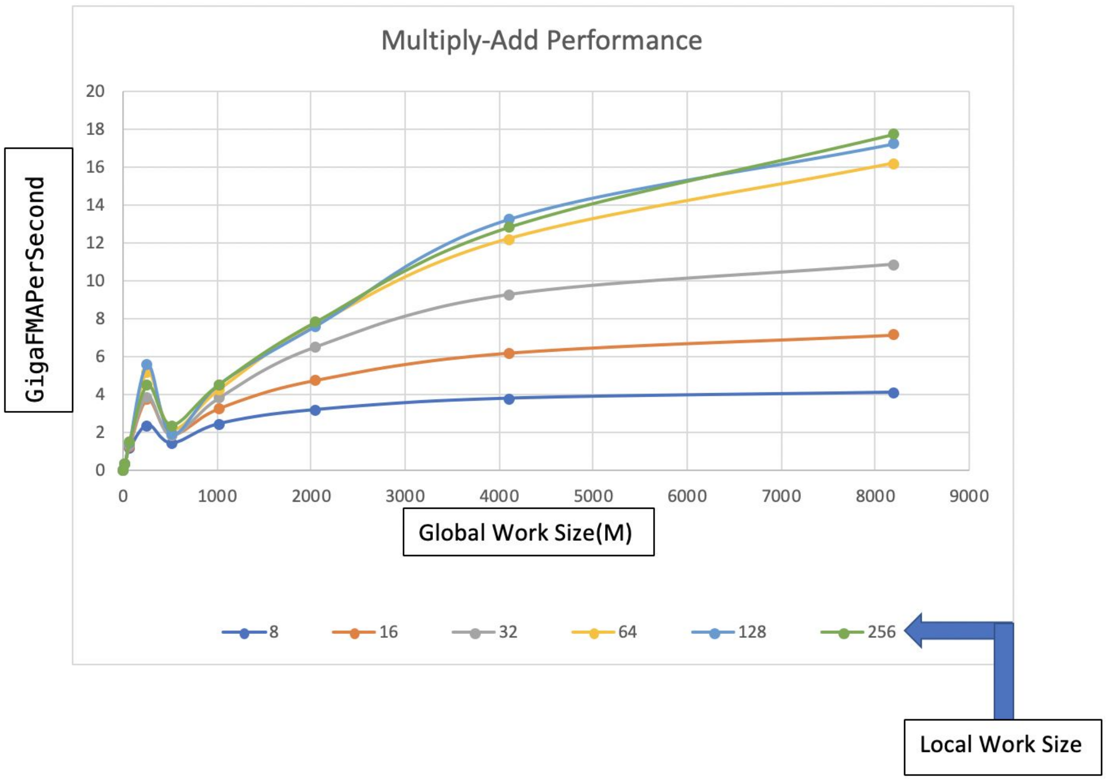
The Performance Table for the Multiply-Add

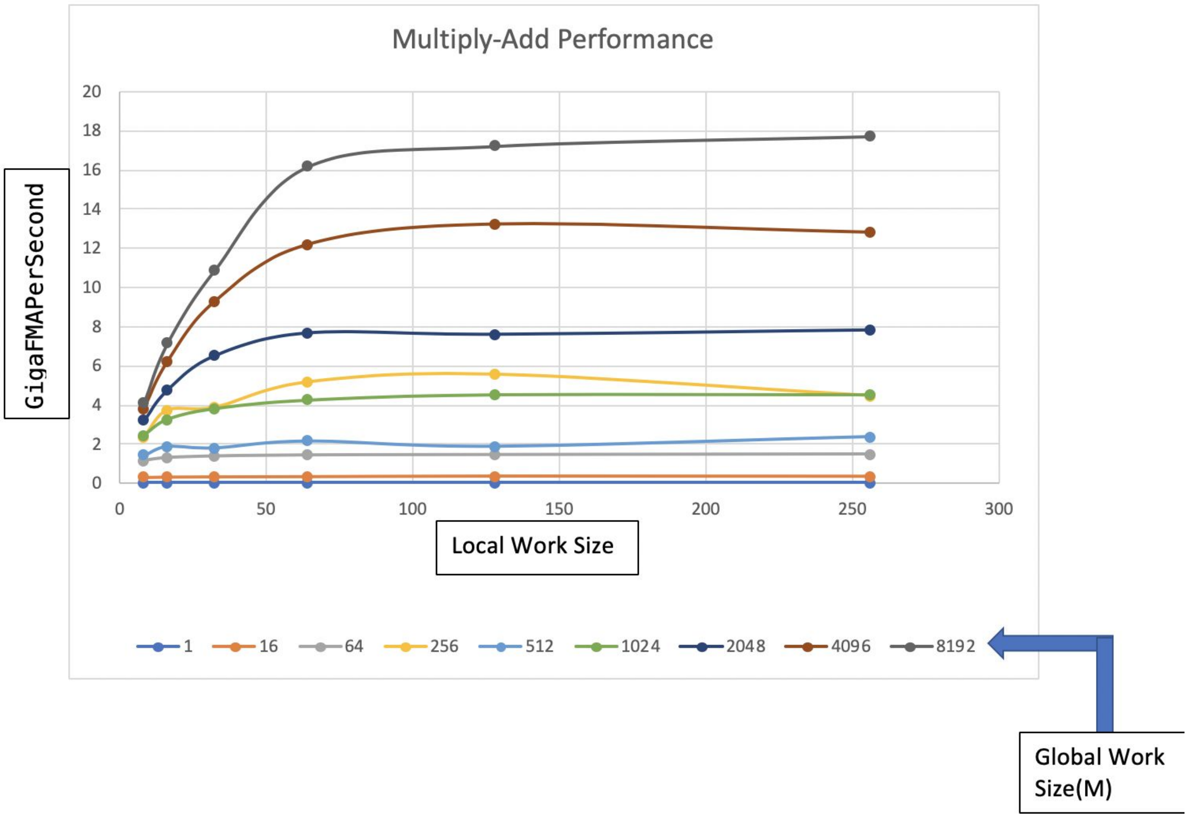
Local Work Size	Global Work Size(M)	Performance
8	1	0.023
8	16	0.34
8	64	1.145
8	256	2.385
8	512	1.435
8	1024	2.464
8	2048	3.2
8	4096	3.801
8	8192	4.117
16	1	0.023
16	16	0.348
16	64	1.292
16	256	3.733
16	512	1.886
16	1024	3.273
16	2048	4.761
16	4096	6.202
16	8192	7.156

32	1	0.023
32	16	0.352
32	64	1.372
32	256	3.878
32	512	1.816
32	1024	3.824
32	2048	6.501
32	4096	9.26
32	8192	10.861
64	1	0.023
64	16	0.356
64	64	1.428
64	256	5.171
64	512	2.162
64	1024	4.267
64	2048	7.677
64	4096	12.205
64	8192	16.185

128	1	0.023
128	16	0.366
128	64	1.447
128	256	5.562
128	512	1.9
128	1024	4.545
128	2048	7.608
128	4096	13.242
128	8192	17.231
256	1	0.023
256	16	0.363
256	64	1.477
256	256	4.476
256	512	2.385
256	1024	4.548
256	2048	7.83
256	4096	12.822
256	8192	17.73

The Performance Graph for the Multiply-Add





The overall performance for the multiply and multiply-add are nearly the same because the FMA(Fused Multiply-Add) is used in the multiply-add calculation of the Multiply-Add code.

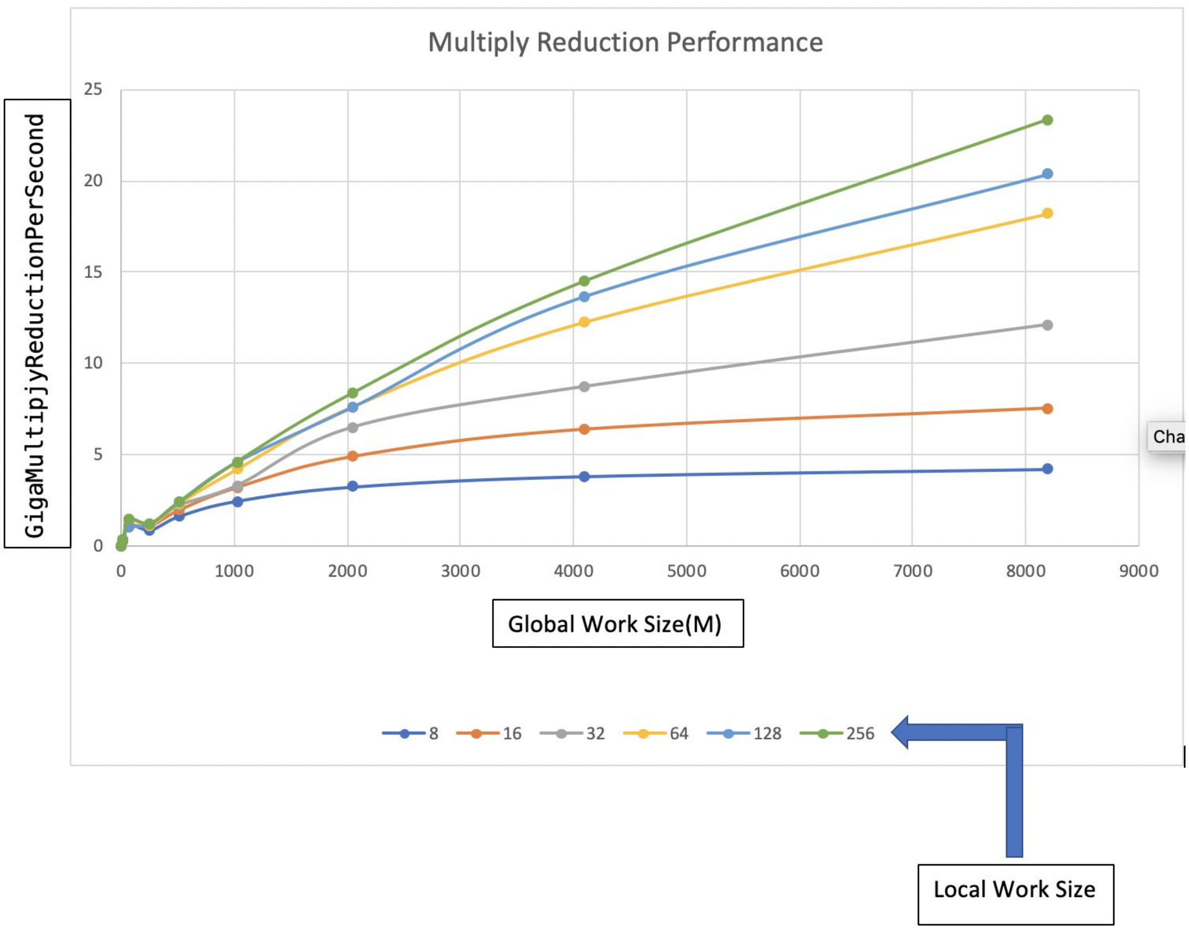
The Performance Table for the Multiply Reduction

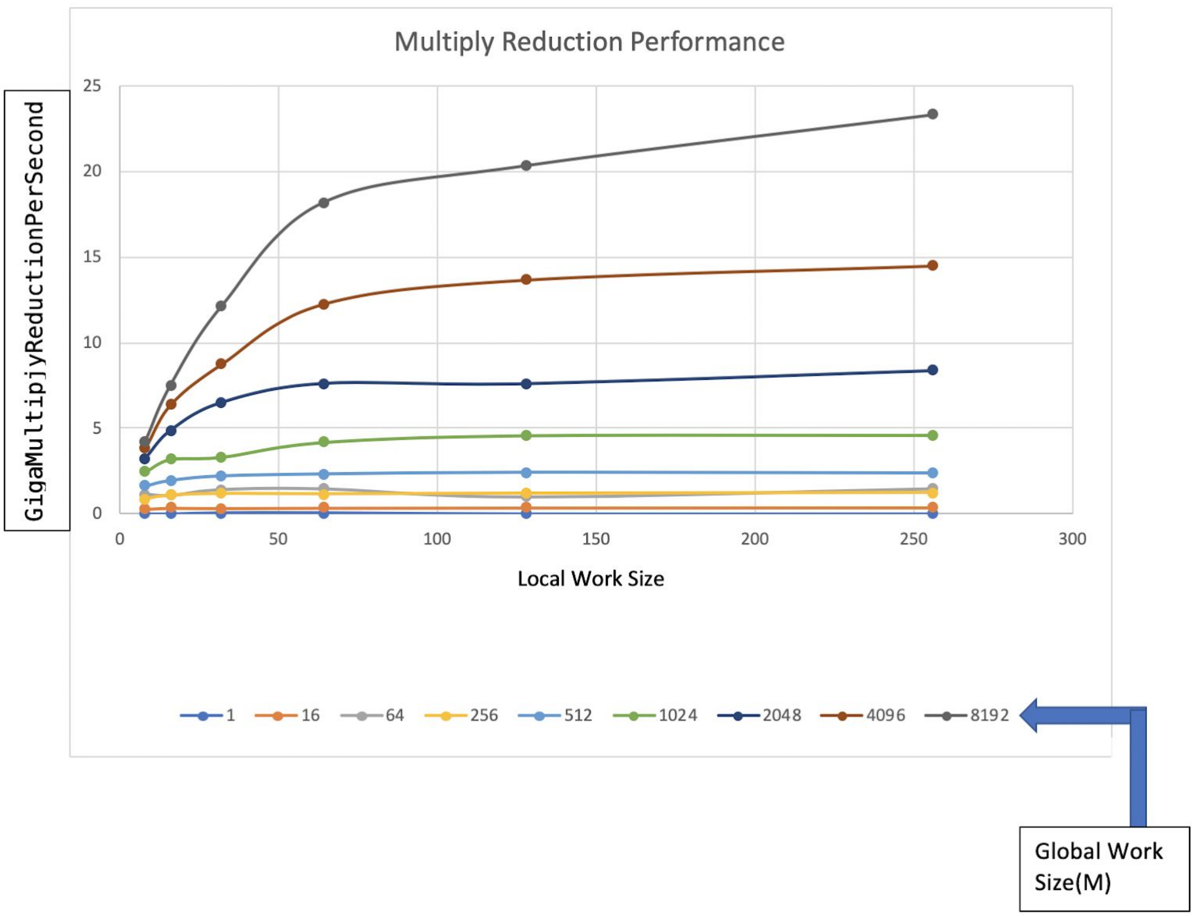
Local Work Size	Global Work Size(M)	Performance
8	1	0.022
8	16	0.282
8	64	1.151
8	256	0.868
8	512	1.622
8	1024	2.459
8	2048	3.24
8	4096	3.819
8	8192	4.217
16	1	0.022
16	16	0.341
16	64	1.081
16	256	1.089
16	512	1.956
16	1024	3.19
16	2048	4.886
16	4096	6.377
16	8192	7.531

32	1	0.023
32	16	0.327
32	64	1.406
32	256	1.179
32	512	2.218
32	1024	3.322
32	2048	6.513
32	4096	8.739
32	8192	12.131
64	1	0.023
64	16	0.345
64	64	1.445
64	256	1.16
64	512	2.333
64	1024	4.192
64	2048	7.629
64	4096	12.258
64	8192	18.2

128	1	0.022
128	16	0.358
128	64	0.996
128	256	1.193
128	512	2.431
128	1024	4.595
128	2048	7.615
128	4096	13.672
128	8192	20.364
256	1	0.022
256	16	0.366
256	64	1.449
256	256	1.24
256	512	2.393
256	1024	4.62
256	2048	8.381
256	4096	14.495
256	8192	23.346

The Performance Graph for the Multiply Reduction





In the above graphs,the performance for the local work size 64,128 and 256 are noticeably better than other local work sizes. That is because warps,which contains 32 threads, can be swapped to keep the cores busy when there are multiple warps, which results in the high performance.

To conclude ,the performance for every local work size becomes better when the global work sizes become larger because there are a considerable number of cores in the GPU and more parallelism can be done by having more global workload. The more work which is done in parallel, the better the performance is. Therefore, the advantage of a large number of cores from the GPU should be taken to improve the performance of a program. However, the number of cores is finite, which means that the performance improvement will halt at a particular global work size.