Project #6
OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

Michael Smith
Email: smitmic5@oregonstate.edu

Machine this ran on:

- I ran this on my MacBook Pro running on OS Monterey version 12.0.1 while on the DGX and Rabbit OSU server:
  CPU: 2 GHz Dual-Core Intel Core i5
  Memory: 8 GB 1867 MHz LPDDR3

Table of Data for Multiply:

NUM_ELEMENTS	LOCAL_SIZE	WORK_GROUP_SIZE	Performance
1024	32	32	0.017
2048	32	64	0.032
4096	32	128	0.071
8192	32	256	0.125
16384	32	512	0.249
32768	32	1024	0.559
65536	32	2048	1.001
131072	32	4096	2.017
262144	32	8192	0.302
524288	32	16384	0.999
1048576	32	32768	1.769
2097152	32	65536	3.152
4194304	32	131072	4.836
8388608	32	262144	7.632
1024	64	16	0.015
2048	64	32	0.034
4096	64	64	0.072
8192	64	128	0.128
16384	64	256	0.28
32768	64	512	0.488
65536	64	1024	1.119
131072	64	2048	2.041
262144	64	4096	0.53
524288	64	8192	0.843
1048576	64	16384	1.843
2097152	64	32768	1.974
4194304	64	65536	5.206
8388608	64	131072	9.397

1024	128	8	0.017
2048	128	16	0.035
4096	128	32	0.068
8192	128	64	0.123
16384	128	128	0.234
32768	128	256	0.573
65536	128	512	1.155
131072	128	1024	2.217
262144	128	2048	0.258
524288	128	4096	0.767
1048576	128	8192	2.013
2097152	128	16384	2.1
4194304	128	32768	7.102
8388608	128	65536	12.815
1024	256	4	0.015
2048	256	8	0.036
4096	256	16	0.068
8192	256	32	0.135
16384	256	64	0.272
32768	256	128	0.488
65536	256	256	1.15
131072	256	512	1.969
262144	256	1024	0.129
524288	256	2048	1.019
1048576	256	4096	1.155
2097152	256	8192	3.479
4194304	256	16384	6.532
8388608	256	32768	11.185
1024	512	2	0.017
2048	512	4	0.03
4096	512	8	0.066
8192	512	16	0.137
16384	512	32	0.238
32768	512	64	0.486
65536	512	128	0.969
131072	512	256	1.985
262144	512	512	0.498
524288	512	1024	0.98
1048576	512	2048	1.754

2097152	512	4096	0.986
4194304	512	8192	6.907
8388608	512	16384	7.115

Multiply Graph:

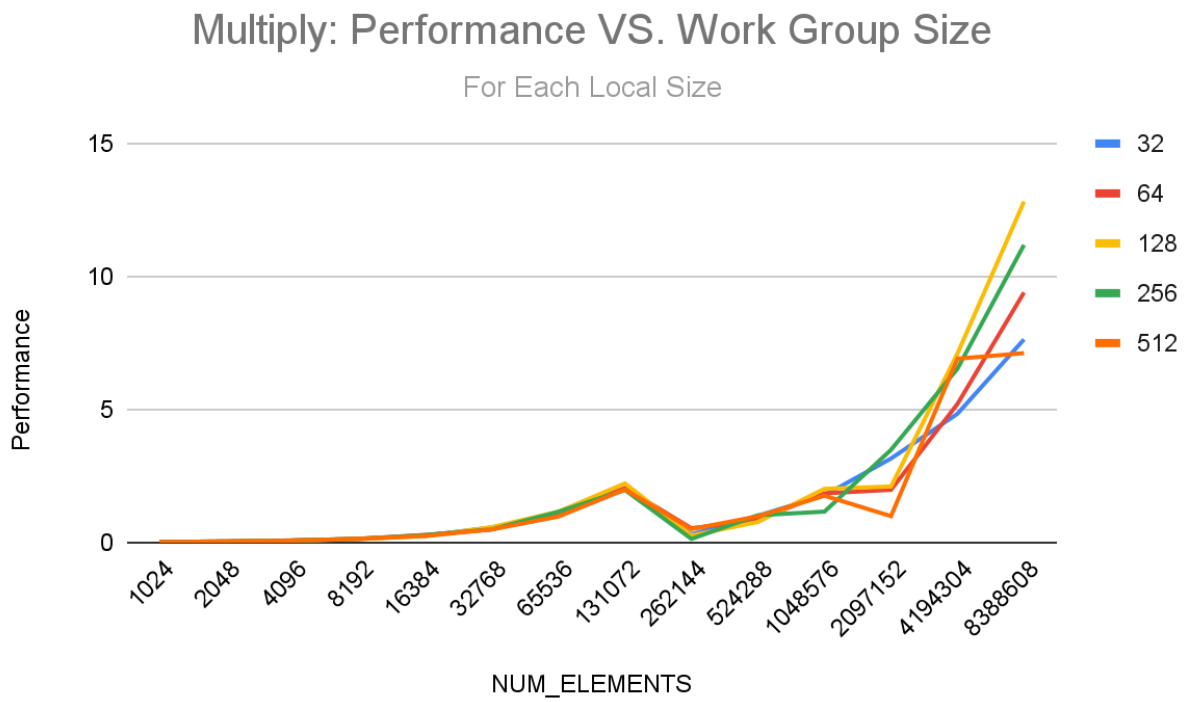


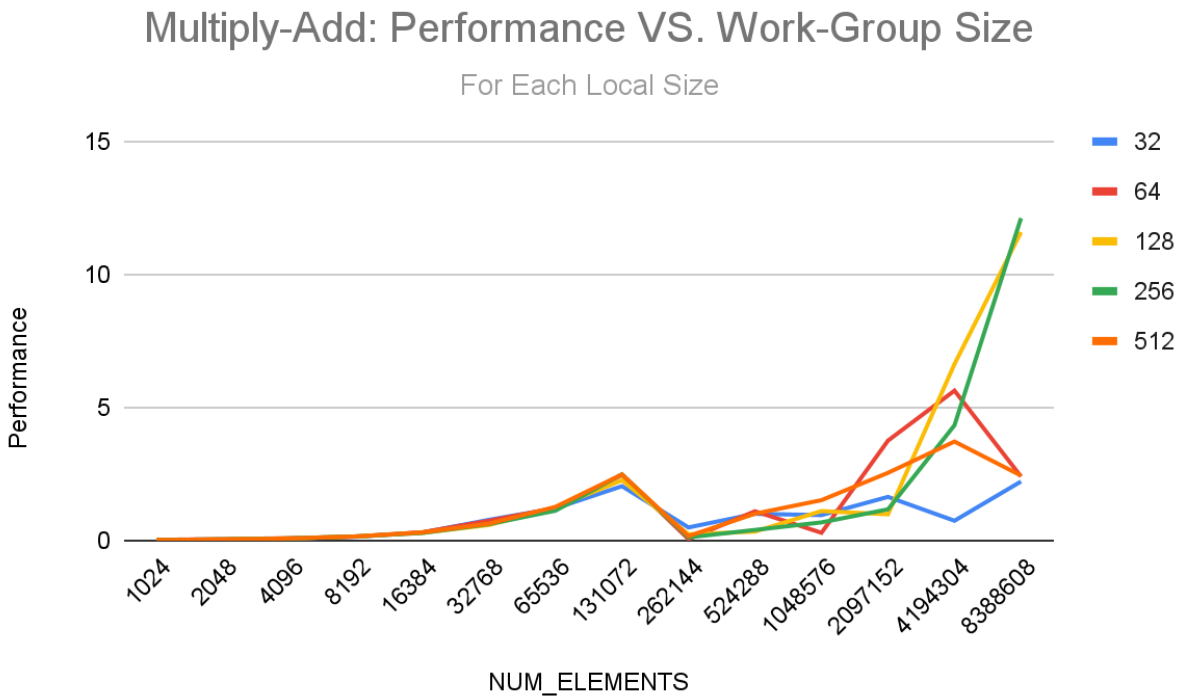
Table of Data for Multiply-Add:

NUM_ELEMENTS	LOCAL_SIZE	WORK_GOUP_SIZE	Performance
1024	32	32	0.018
2048	32	64	0.035
4096	32	128	0.069

8192	32	256	0.133
16384	32	512	0.296
32768	32	1024	0.77
65536	32	2048	1.204
131072	32	4096	2.036
262144	32	8192	0.485
524288	32	16384	0.997
1048576	32	32768	0.949
2097152	32	65536	1.634
4194304	32	131072	0.737
8388608	32	262144	2.212
1024	64	16	0.02
2048	64	32	0.045
4096	64	64	0.08
8192	64	128	0.143
16384	64	256	0.303
32768	64	512	0.736
65536	64	1024	1.213
131072	64	2048	2.457
262144	64	4096	0.045
524288	64	8192	1.093
1048576	64	16384	0.284
2097152	64	32768	3.75
4194304	64	65536	5.63
8388608	64	131072	2.4
1024	128	8	0.017
2048	128	16	0.036
4096	128	32	0.072
8192	128	64	0.147
16384	128	128	0.281
32768	128	256	0.576
65536	128	512	1.241
131072	128	1024	2.26
262144	128	2048	0.225
524288	128	4096	0.329
1048576	128	8192	1.1
2097152	128	16384	0.98
4194304	128	32768	6.629
8388608	128	65536	11.593

1024	256	4	0.017
2048	256	8	0.036
4096	256	16	0.073
8192	256	32	0.147
16384	256	64	0.278
32768	256	128	0.61
65536	256	256	1.115
131072	256	512	2.484
262144	256	1024	0.111
524288	256	2048	0.391
1048576	256	4096	0.675
2097152	256	8192	1.165
4194304	256	16384	4.328
8388608	256	32768	12.118
1024	512	2	0.013
2048	512	4	0.033
4096	512	8	0.07
8192	512	16	0.145
16384	512	32	0.298
32768	512	64	0.628
65536	512	128	1.259
131072	512	256	2.477
262144	512	512	0.16
524288	512	1024	1.002
1048576	512	2048	1.511
2097152	512	4096	2.539
4194304	512	8192	3.715
8388608	512	16384	2.436

Multiply-Add Graph:



Patterns:

- The biggest pattern I see is that the variation in performance is more as the Number of elements and the work group size increases. You can see this by looking at the beginning of each graph. All the performances are relatively the same for every Local Size for the first 6 group sizes.
- I would assume these patterns are happening because of the extra time being lossed while making calculations. This would explain why smaller local sizes for the Multiply-add have significantly less performance when compared to just the Multiply graph.

Performance Differences:

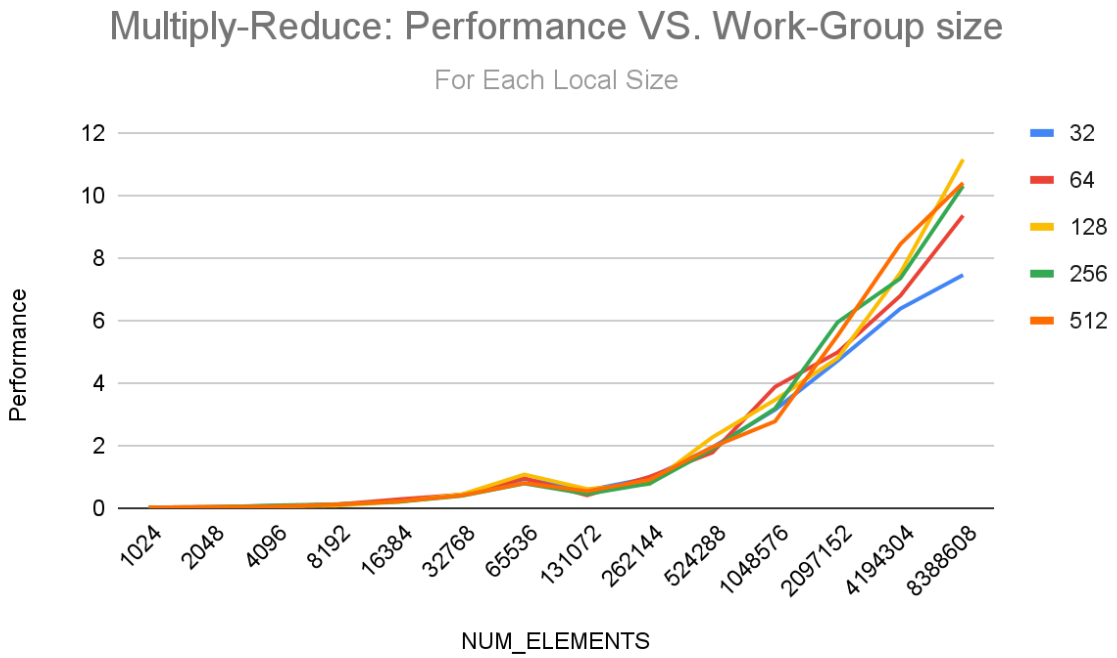
- The performance differences between Multiply-add and Multiply really can only be seen as the Work group size increases. Because Multiply-add requires more calculations, there is a significant decrease in performance when compared to the Multiply performances. I would assume if you kept on increasing the number of elements, the Multiply graph would show a plateau as well, just at a higher performance.

Multiply-Reduce Table:

NUM_ELEMNTS	LOCAL_SIZE	WORK_GOUP_SIZE	Performance
1024	32	32	0.013
2048	32	64	0.032
4096	32	128	0.064
8192	32	256	0.101
16384	32	512	0.217
32768	32	1024	0.427
65536	32	2048	0.928
131072	32	4096	0.562
262144	32	8192	0.958
524288	32	16384	1.939
1048576	32	32768	3.143
2097152	32	65536	4.712
4194304	32	131072	6.377
8388608	32	262144	7.454
1024	64	16	0.018
2048	64	32	0.029
4096	64	64	0.048
8192	64	128	0.111
16384	64	256	0.277
32768	64	512	0.424
65536	64	1024	0.947
131072	64	2048	0.404
262144	64	4096	1
524288	64	8192	1.772
1048576	64	16384	3.875
2097152	64	32768	4.981
4194304	64	65536	6.792
8388608	64	131072	9.357
1024	128	8	0.019
2048	128	16	0.028
4096	128	32	0.054
8192	128	64	0.077
16384	128	128	0.198
32768	128	256	0.435

65536	128	512	1.069
131072	128	1024	0.606
262144	128	2048	0.768
524288	128	4096	2.261
1048576	128	8192	3.459
2097152	128	16384	4.796
4194304	128	32768	7.523
8388608	128	65536	11.152
1024	256	4	0.013
2048	256	8	0.023
4096	256	16	0.08
8192	256	32	0.118
16384	256	64	0.198
32768	256	128	0.388
65536	256	256	0.782
131072	256	512	0.447
262144	256	1024	0.783
524288	256	2048	1.884
1048576	256	4096	3.181
2097152	256	8192	5.945
4194304	256	16384	7.346
8388608	256	32768	10.305
1024	512	2	0.013
2048	512	4	0.03
4096	512	8	0.047
8192	512	16	0.112
16384	512	32	0.211
32768	512	64	0.399
65536	512	128	0.793
131072	512	256	0.537
262144	512	512	0.901
524288	512	1024	1.95
1048576	512	2048	2.77
2097152	512	4096	5.518
4194304	512	8192	8.443
8388608	512	16384	10.4

Multiply-Reduce Graph:



Patterns:

- Compared to the other two graphs, this graph has generally less variation for each Local Size. Unlike the other graphs, a local size of 32 does not plateau as the number of elements increases which is interesting.
- I think this pattern shows the advantages of reduction because you are able to get a higher performance with a smaller local size. I would assume this is because there is time being saved while using the reduction method when compared to the alternatives. This would intern, increase the overall performance.

GPU Computing:

- For proper GPU computing, this would mean that we are still able to increase the performance where we would not have been able to when not using a GPU. It also means that reducing calculation time will always help improve performance even when computing on a GPU.