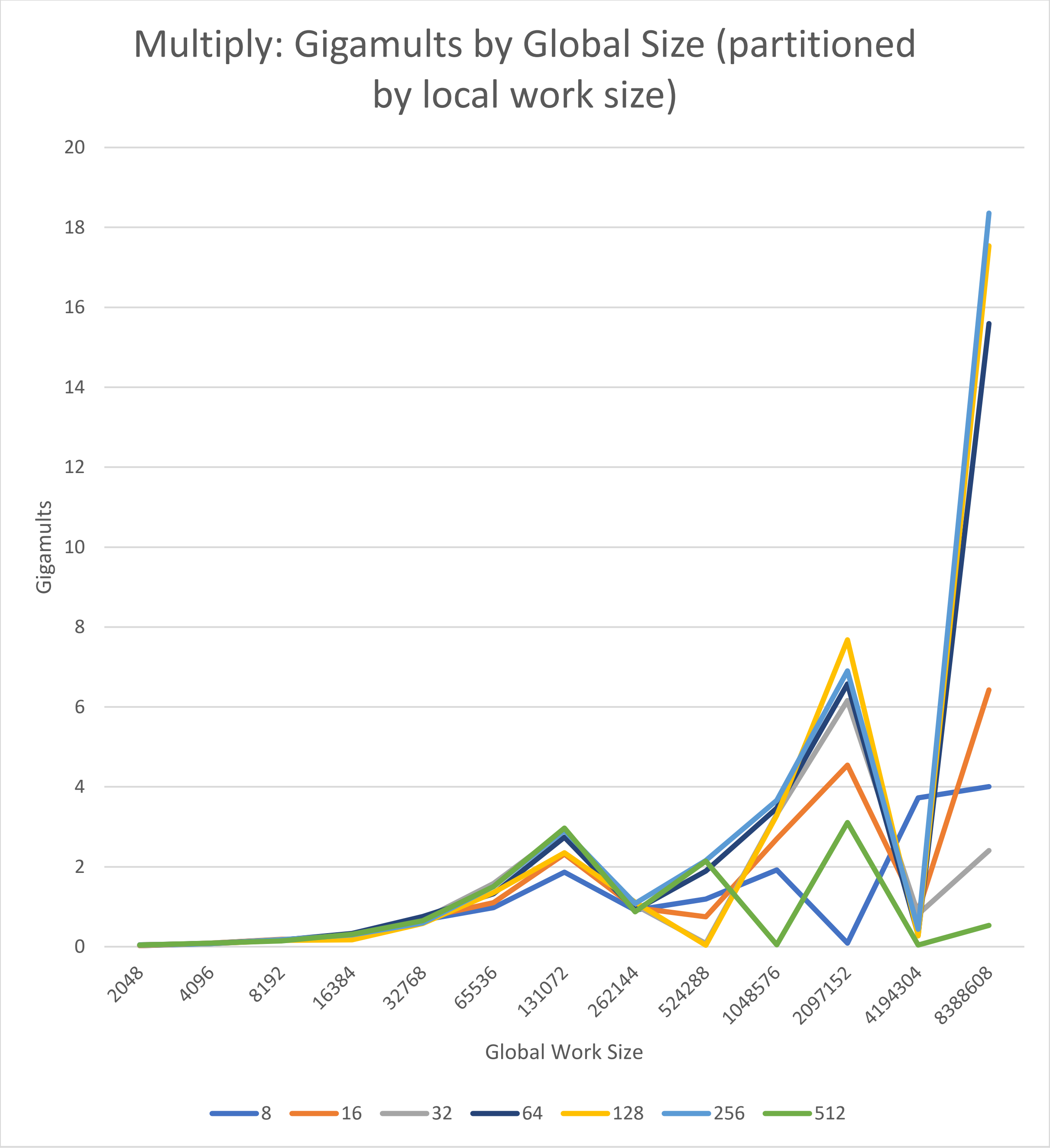
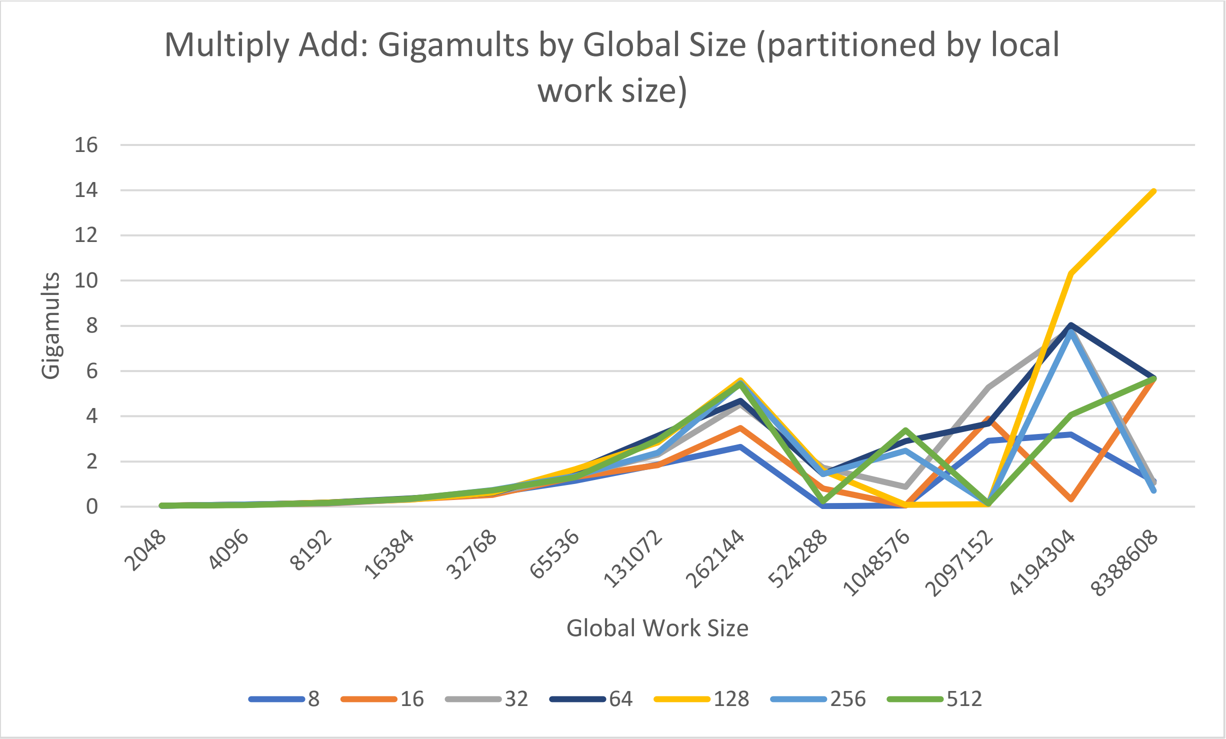
OpenCL Multiply/Add/Reduction

By Logan Saso

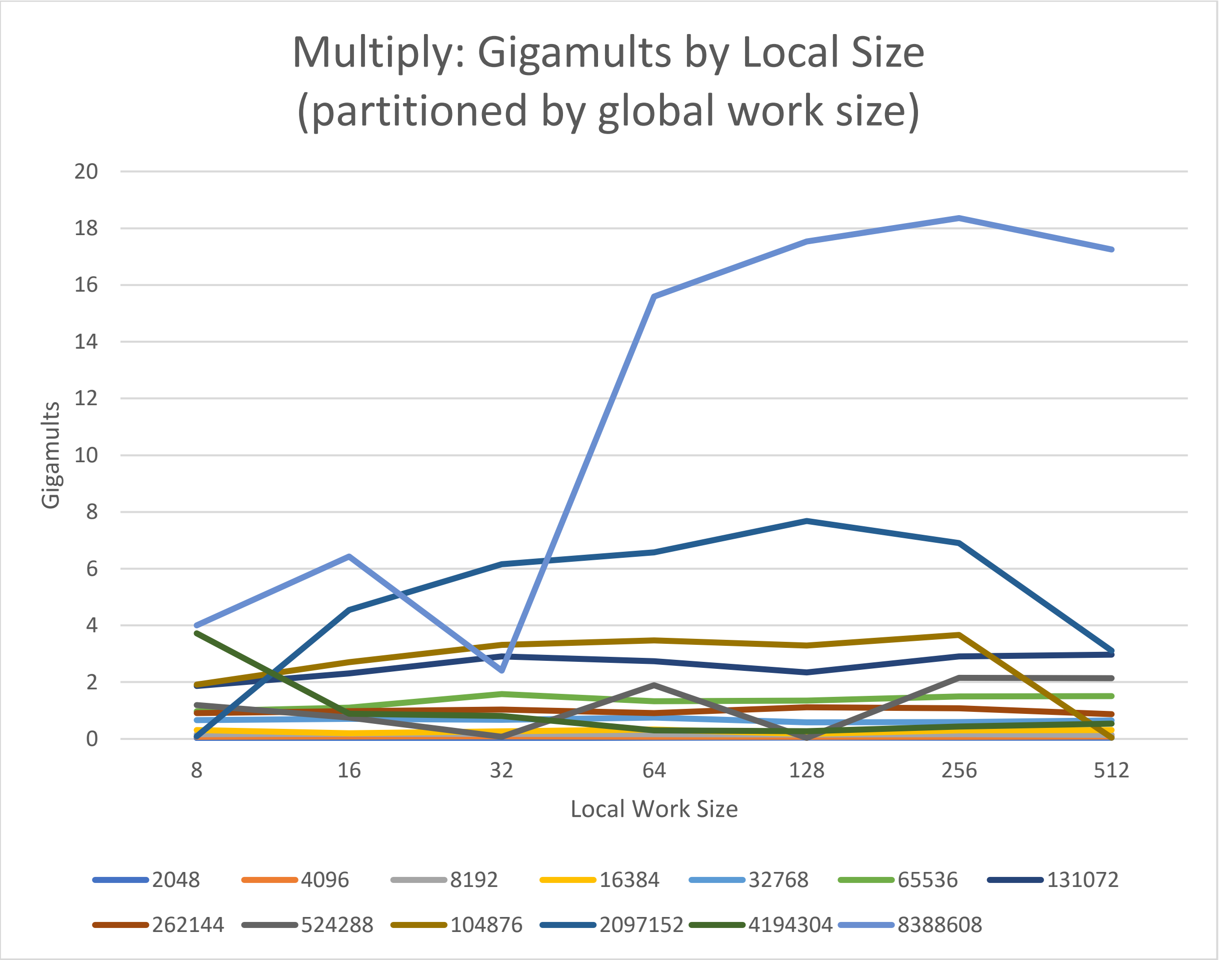
1. As I was in the middle of moving during this project, I was unable to run this on my local machine like I usually do. My thanks go to OSU for providing the hulking DGX system where I run my tests.
2. Graphs

a.

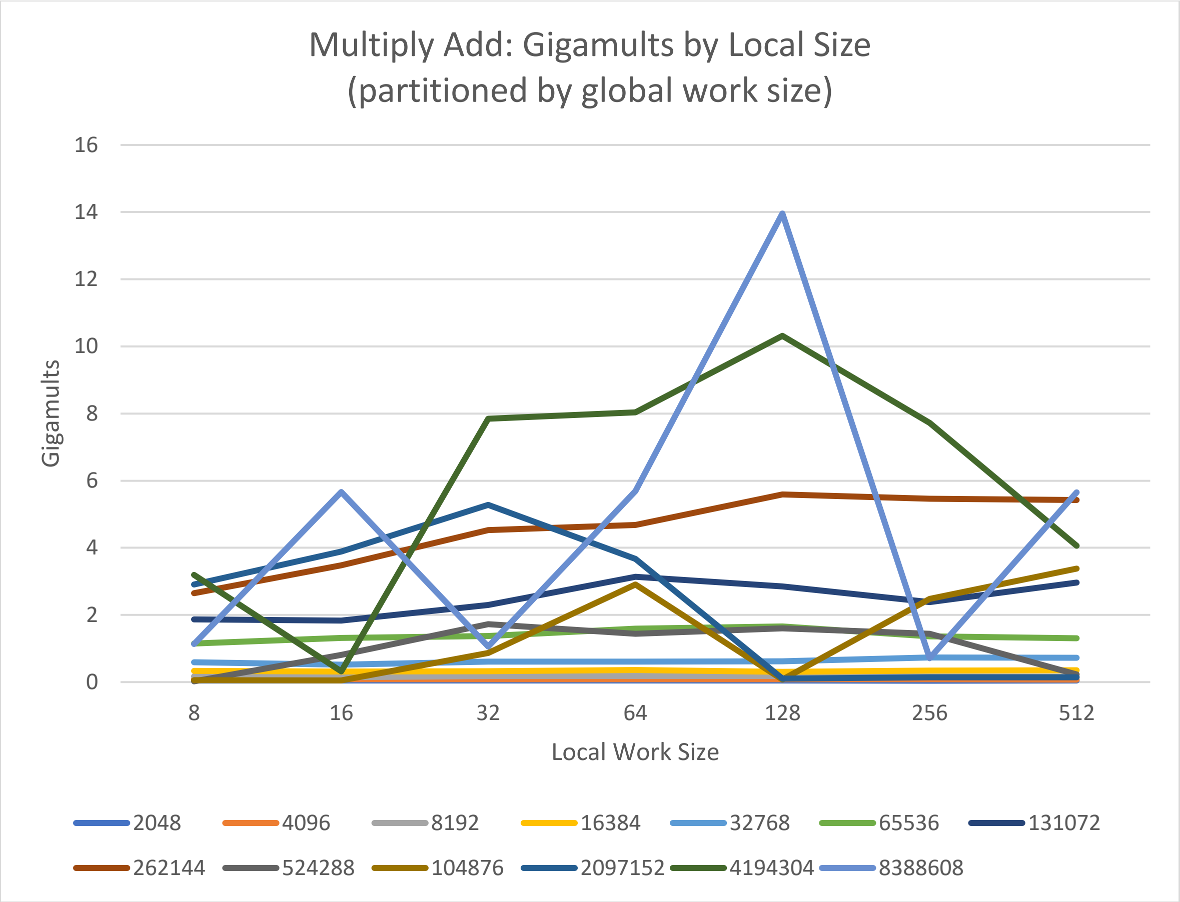



Multiply Add: Gigamults by Global Size (partitioned by local work size)
Gigamults
16
14
12
10
8
6
4
2
0
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
Global Work Size
8
16
32
64
128
256
512

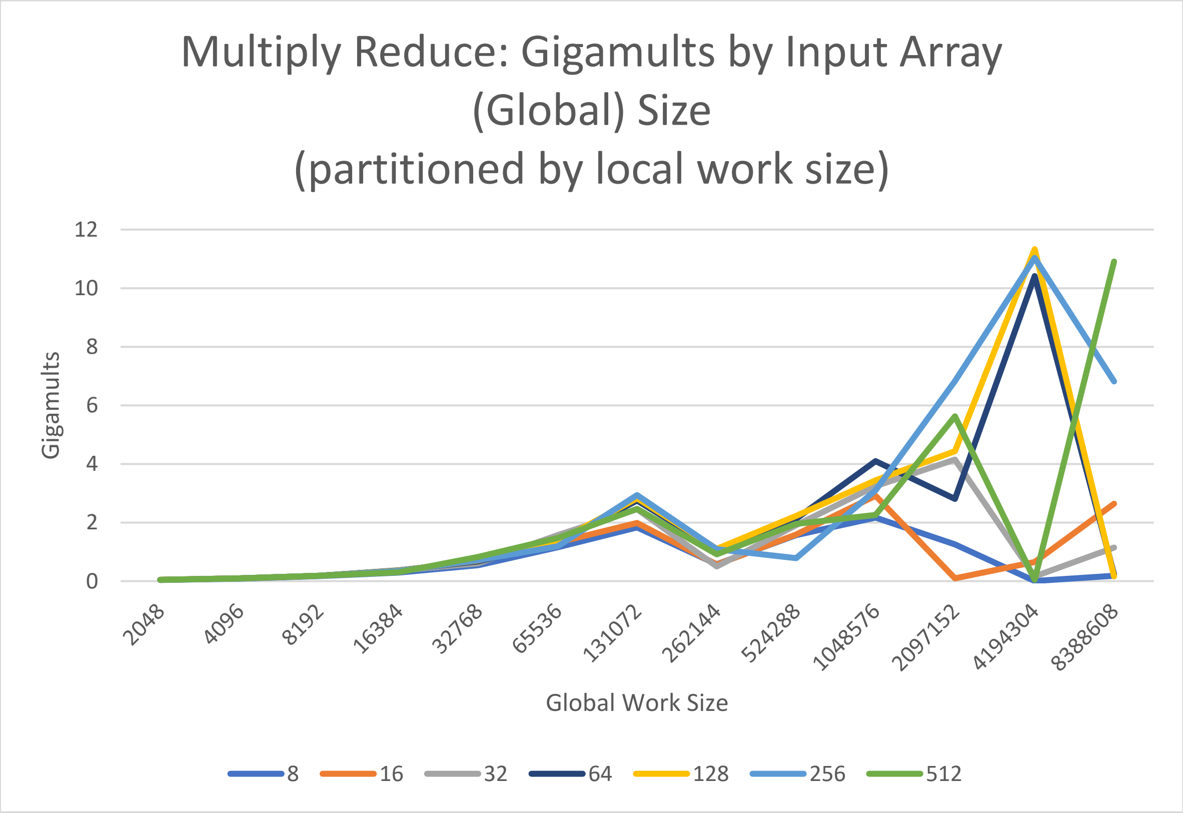
b.


Multiply: Gigamults by Local Size (partitioned by global work size)
Gigamults
20
18
16
14
12
10
8
6
4
2
0
8
16
32
64
128
256
512
Local Work Size
2048
4096
8192
16384
32768
65536
131072
262144
524288
104876
2097152
4194304
8388608

c.


Multiply Add: Gigamults by Local Size
(partitioned by global work size)
Gigamults
16
14
12
10
8
6
4
2
0
8
16
32
64
128
256
512
Local Work Size
2048
4096
8192
16384
32768
65536
131072
262144
524288
104876
2097152
4194304
8388608

d.


Multiply Reduce: Gigamults by Input Array
(Global) Size
(partitioned by local work size)
Gigamults
12
10
8
6
4
2
0
2048
4096
8192
16384
32768
65536
131072
262144
524288
1048576
2097152
4194304
8388608
Global Work Size
8
16
32
64
128
256
512

e.

f. Multiply Table

NUM_ELEMENTS	NMB	LOCAL_SIZE	WORK_GROUPS	GIGAMULTS
2048	0.001953	8	256	0.025975
2048	0.001953	16	128	0.024659
2048	0.001953	32	64	0.031346
2048	0.001953	64	32	0.046154
2048	0.001953	128	16	0.048227
2048	0.001953	256	8	0.046939
2048	0.001953	512	4	0.04175
4096	0.003906	8	512	0.076582
4096	0.003906	16	256	0.091938
4096	0.003906	32	128	0.092338
4096	0.003906	64	64	0.080459
4096	0.003906	128	32	0.076317
4096	0.003906	256	16	0.07427
4096	0.003906	512	8	0.091346
8192	0.007812	8	1024	0.165349
8192	0.007812	16	512	0.183462
8192	0.007812	32	256	0.179747
8192	0.007812	64	128	0.165889
8192	0.007812	128	64	0.159874
8192	0.007812	256	32	0.171732
8192	0.007812	512	16	0.142454
16384	0.015625	8	2048	0.316754
16384	0.015625	16	1024	0.200373
16384	0.015625	32	512	0.268182
16384	0.015625	64	256	0.332952
16384	0.015625	128	128	0.175669
16384	0.015625	256	64	0.298552
16384	0.015625	512	32	0.311979
32768	0.03125	8	4096	0.668041
32768	0.03125	16	2048	0.711744
32768	0.03125	32	1024	0.675674
32768	0.03125	64	512	0.752349
32768	0.03125	128	256	0.584497
32768	0.03125	256	128	0.597023
32768	0.03125	512	64	0.65055
65536	0.0625	8	8192	0.974987
65536	0.0625	16	4096	1.102509
65536	0.0625	32	2048	1.581498
65536	0.0625	64	1024	1.323094

65536	0.0625	128	512	1.354
65536	0.0625	256	256	1.501136
65536	0.0625	512	128	1.50589
131072	0.125	8	16384	1.866198
131072	0.125	16	8192	2.312859
131072	0.125	32	4096	2.904918
131072	0.125	64	2048	2.740803
131072	0.125	128	1024	2.349698
131072	0.125	256	512	2.909061
131072	0.125	512	256	2.970211
262144	0.25	8	32768	0.915711
262144	0.25	16	16384	0.980547
262144	0.25	32	8192	1.0399
262144	0.25	64	4096	0.915556
262144	0.25	128	2048	1.116083
262144	0.25	256	1024	1.082067
262144	0.25	512	512	0.872991
524288	0.5	8	65536	1.195213
524288	0.5	16	32768	0.748882
524288	0.5	32	16384	0.079015
524288	0.5	64	8192	1.893071
524288	0.5	128	4096	0.036432
524288	0.5	256	2048	2.153794
524288	0.5	512	1024	2.143069
1048576	1	8	131072	1.920469
1048576	1	16	65536	2.700887
1048576	1	32	32768	3.313439
1048576	1	64	16384	3.468309
1048576	1	128	8192	3.287884
1048576	1	256	4096	3.665134
1048576	1	512	2048	0.044728
2097152	2	8	262144	0.091935
2097152	2	16	131072	4.544252
2097152	2	32	65536	6.161044
2097152	2	64	32768	6.574464
2097152	2	128	16384	7.681208
2097152	2	256	8192	6.900461
2097152	2	512	4096	3.107169
4194304	4	8	524288	3.721386
4194304	4	16	262144	0.891369
4194304	4	32	131072	0.811216

4194304	4	64	65536	0.298444
4194304	4	128	32768	0.272932
4194304	4	256	16384	0.434035
4194304	4	512	8192	0.534845
8388608	8	8	1048576	4.00581
8388608	8	16	524288	6.426113
8388608	8	32	262144	2.406722
8388608	8	64	131072	15.592643
8388608	8	128	65536	17.537586
8388608	8	256	32768	18.356939
8388608	8	512	16384	17.244963

g. MultipleSum

NUM_ELEMENTS	NMB	LOCAL_SIZE	WORK_GROUPS	GIGAMULTS
2048	0.001953	8	256	0.043503
2048	0.001953	16	128	0.043846
2048	0.001953	32	64	0.045383
2048	0.001953	64	32	0.045917
2048	0.001953	128	16	0.04689
2048	0.001953	256	8	0.039762
2048	0.001953	512	4	0.045709
4096	0.003906	8	512	0.090609
4096	0.003906	16	256	0.090141
4096	0.003906	32	128	0.086729
4096	0.003906	64	64	0.083969
4096	0.003906	128	32	0.089431
4096	0.003906	256	16	0.089657
4096	0.003906	512	8	0.073876
8192	0.007812	8	1024	0.172266
8192	0.007812	16	512	0.143504
8192	0.007812	32	256	0.17899
8192	0.007812	64	128	0.176511
8192	0.007812	128	64	0.176792
8192	0.007812	256	32	0.157854
8192	0.007812	512	16	0.163578
16384	0.015625	8	2048	0.333761
16384	0.015625	16	1024	0.316281
16384	0.015625	32	512	0.321412
16384	0.015625	64	256	0.359494
16384	0.015625	128	128	0.305568
16384	0.015625	256	64	0.343102
16384	0.015625	512	32	0.351303

32768	0.03125	8	4096	0.587797
32768	0.03125	16	2048	0.519258
32768	0.03125	32	1024	0.611966
32768	0.03125	64	512	0.613022
32768	0.03125	128	256	0.623107
32768	0.03125	256	128	0.733726
32768	0.03125	512	64	0.719988
65536	0.0625	8	8192	1.149478
65536	0.0625	16	4096	1.318433
65536	0.0625	32	2048	1.367897
65536	0.0625	64	1024	1.594651
65536	0.0625	128	512	1.657724
65536	0.0625	256	256	1.358234
65536	0.0625	512	128	1.301606
131072	0.125	8	16384	1.864246
131072	0.125	16	8192	1.832162
131072	0.125	32	4096	2.297905
131072	0.125	64	2048	3.136351
131072	0.125	128	1024	2.849283
131072	0.125	256	512	2.385422
131072	0.125	512	256	2.960402
262144	0.25	8	32768	2.648254
262144	0.25	16	16384	3.480285
262144	0.25	32	8192	4.524594
262144	0.25	64	4096	4.676208
262144	0.25	128	2048	5.589479
262144	0.25	256	1024	5.464684
262144	0.25	512	512	5.42059
524288	0.5	8	65536	0.02173
524288	0.5	16	32768	0.807608
524288	0.5	32	16384	1.72493
524288	0.5	64	8192	1.43842
524288	0.5	128	4096	1.601555
524288	0.5	256	2048	1.438596
524288	0.5	512	1024	0.232859
1048576	1	8	131072	0.057553
1048576	1	16	65536	0.049994
1048576	1	32	32768	0.870847
1048576	1	64	16384	2.904888
1048576	1	128	8192	0.079285
1048576	1	256	4096	2.476792

1048576	1	512	2048	3.38034
2097152	2	8	262144	2.908629
2097152	2	16	131072	3.889287
2097152	2	32	65536	5.278926
2097152	2	64	32768	3.670966
2097152	2	128	16384	0.103598
2097152	2	256	8192	0.143371
2097152	2	512	4096	0.138429
4194304	4	8	524288	3.191914
4194304	4	16	262144	0.31963
4194304	4	32	131072	7.841129
4194304	4	64	65536	8.033162
4194304	4	128	32768	10.314997
4194304	4	256	16384	7.725573
4194304	4	512	8192	4.063073
8388608	8	8	1048576	1.135749
8388608	8	16	524288	5.658713
8388608	8	32	262144	1.058928
8388608	8	64	131072	5.693761
8388608	8	128	65536	13.958201
8388608	8	256	32768	0.710091
8388608	8	512	16384	5.656449

h. Multiply Reduce

NUM_ELEMENTS	NMB	LOCAL_SIZE	WORK_GROUPS	GIGAMULTS
2048	0.001953	8	256	0.041627
2048	0.001953	16	128	0.045239
2048	0.001953	32	64	0.037784
2048	0.001953	64	32	0.045934
2048	0.001953	128	16	0.045411
2048	0.001953	256	8	0.043943
2048	0.001953	512	4	0.045712
4096	0.003906	8	512	0.089886
4096	0.003906	16	256	0.091416
4096	0.003906	32	128	0.092868
4096	0.003906	64	64	0.093735
4096	0.003906	128	32	0.092435
4096	0.003906	256	16	0.094076
4096	0.003906	512	8	0.092017
8192	0.007812	8	1024	0.173901
8192	0.007812	16	512	0.177907
8192	0.007812	32	256	0.183031

8192	0.007812	64	128	0.1827
8192	0.007812	128	64	0.183692
8192	0.007812	256	32	0.182016
8192	0.007812	512	16	0.182219
16384	0.015625	8	2048	0.287807
16384	0.015625	16	1024	0.353662
16384	0.015625	32	512	0.36286
16384	0.015625	64	256	0.36894
16384	0.015625	128	128	0.35269
16384	0.015625	256	64	0.347706
16384	0.015625	512	32	0.306874
32768	0.03125	8	4096	0.544271
32768	0.03125	16	2048	0.708848
32768	0.03125	32	1024	0.606261
32768	0.03125	64	512	0.678816
32768	0.03125	128	256	0.703519
32768	0.03125	256	128	0.715682
32768	0.03125	512	64	0.82814
65536	0.0625	8	8192	1.161162
65536	0.0625	16	4096	1.321181
65536	0.0625	32	2048	1.567721
65536	0.0625	64	1024	1.330977
65536	0.0625	128	512	1.351218
65536	0.0625	256	256	1.188119
65536	0.0625	512	128	1.474741
131072	0.125	8	16384	1.837376
131072	0.125	16	8192	1.988857
131072	0.125	32	4096	2.452984
131072	0.125	64	2048	2.750659
131072	0.125	128	1024	2.82288
131072	0.125	256	512	2.932948
131072	0.125	512	256	2.466482
262144	0.25	8	32768	0.580482
262144	0.25	16	16384	0.564065
262144	0.25	32	8192	0.500648
262144	0.25	64	4096	1.083976
262144	0.25	128	2048	1.104282
262144	0.25	256	1024	1.094509
262144	0.25	512	512	0.91909
524288	0.5	8	65536	1.568014
524288	0.5	16	32768	1.588966

524288	0.5	32	16384	1.912558
524288	0.5	64	8192	2.123818
524288	0.5	128	4096	2.231148
524288	0.5	256	2048	0.787203
524288	0.5	512	1024	1.95388
1048576	1	8	131072	2.168395
1048576	1	16	65536	2.913684
1048576	1	32	32768	3.243463
1048576	1	64	16384	4.093408
1048576	1	128	8192	3.435638
1048576	1	256	4096	3.103978
1048576	1	512	2048	2.263625
2097152	2	8	262144	1.259526
2097152	2	16	131072	0.09407
2097152	2	32	65536	4.150232
2097152	2	64	32768	2.811113
2097152	2	128	16384	4.434938
2097152	2	256	8192	6.825911
2097152	2	512	4096	5.625632
4194304	4	8	524288	0.004569
4194304	4	16	262144	0.651959
4194304	4	32	131072	0.155874
4194304	4	64	65536	10.413838
4194304	4	128	32768	11.333203
4194304	4	256	16384	11.034637
4194304	4	512	8192	10.909857
8388608	8	8	1048576	0.186684
8388608	8	16	524288	2.641129
8388608	8	32	262144	1.14127
8388608	8	64	131072	0.232535
8388608	8	128	65536	0.164312
8388608	8	256	32768	6.821512
8388608	8	512	16384	0.581254

i.

3. What patterns are we seeing?
    a. For the simple multiplication, it seems as if the larger arrays *tend* to have the best performance. My guess is that this is because OpenCL has to do the least amount of coordination for the most amount of data. The trend isn't quite the same for multiply add, as it seems to peak on global work sizes of 262144 and the very last. Seemingly, having 128 as the local work size was best on this machine.
4. Why these patterns?

    a. Like I mentioned before, I think these patterns are being shown due to the configuration of the hardware. DGX is a virtually separated system, so there’s some coordination that occurs when using the hardware. My guess is the local work size of 128 is the best to divide up the hardware in the machine, so OpenCL has the best job of coordinating work. As for larger array sizes, I’d guess it’s a similar reason. If the hardware spends less time deciding what to multiply and more time doing the multiplication, we’re going to come up with a much better speed since it’s multiplying for a greater amount of the program runtime.

5. The performance different between mult and mult-add
    a. Multiply and Add are two completely different operations, so despite the fact that we’re already just *doing more math*, we also have to account for the results of the multiplication step before adding. Since they are ordered operations, we’re waiting on the OpenCL hardware to process each part which will take a tad longer since it has to pass the data from whatever hardware does the multiplication to the addition circuits.
6. Proper use of GPU computing
    a. Proper use of GPU computing would be to stick to one operation if possible. What you really want to do is set it up with huge amounts of data that it can churn through. Part of the reason these crazy expensive graphics cards have 38 gigs of DDR5 memory is so that the data storage and access is as fast as possible so it can spend the most time it can on math.
7. Reduction Section
    a. Table and Graph are above
    b. In this curve, we see a bonus speed boost for arrays that have a nice size of return. For example, the bigger we can break down the global array, the fewer numbers we must reduce into one sum at the end of processing. This section takes time, too, so it’s important that we want to reduce as quickly and to as small of a result array as we can.
    c. If anything, this says to me that reduction via GPU computing is definitely faster, but not the fastest thing it can do. While not necessarily a waste, there are more efficient things the GPU could be doing.