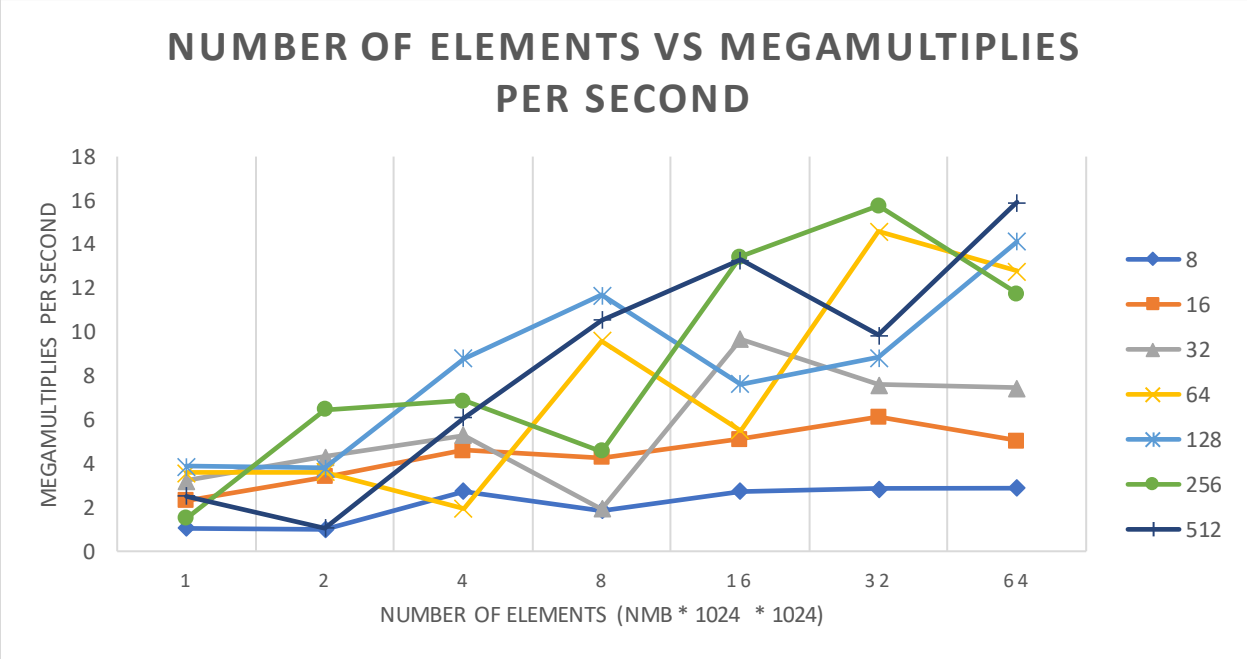
Coulby Nguyen

5/21/19

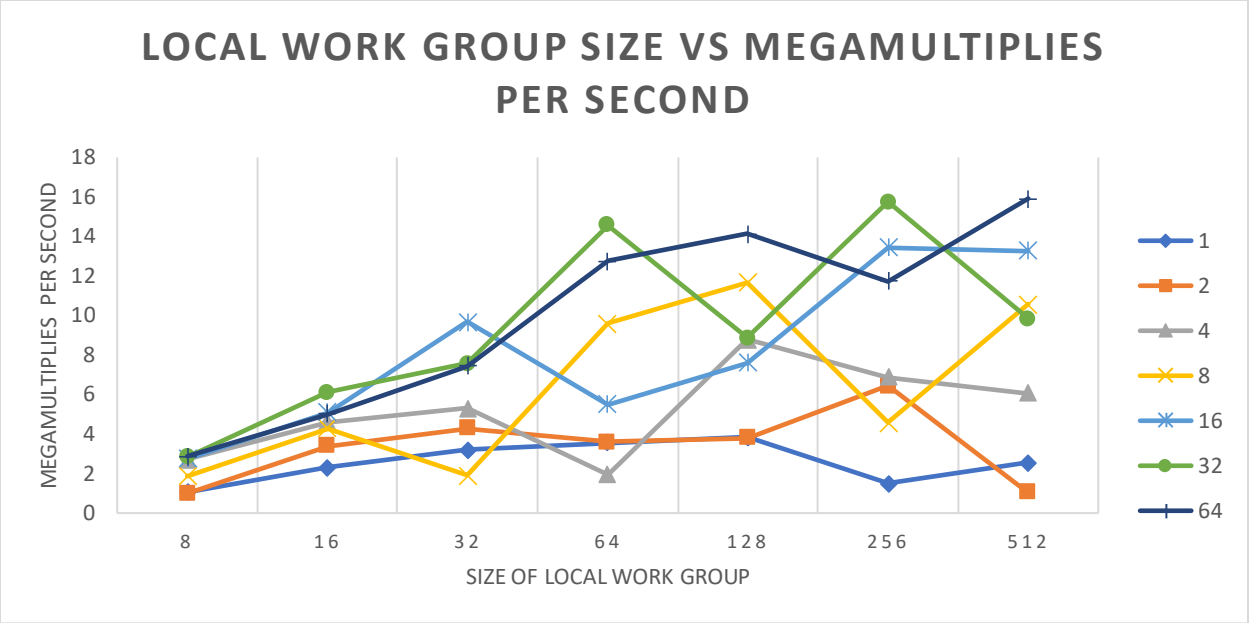
CS 475

Project 5 Write Up

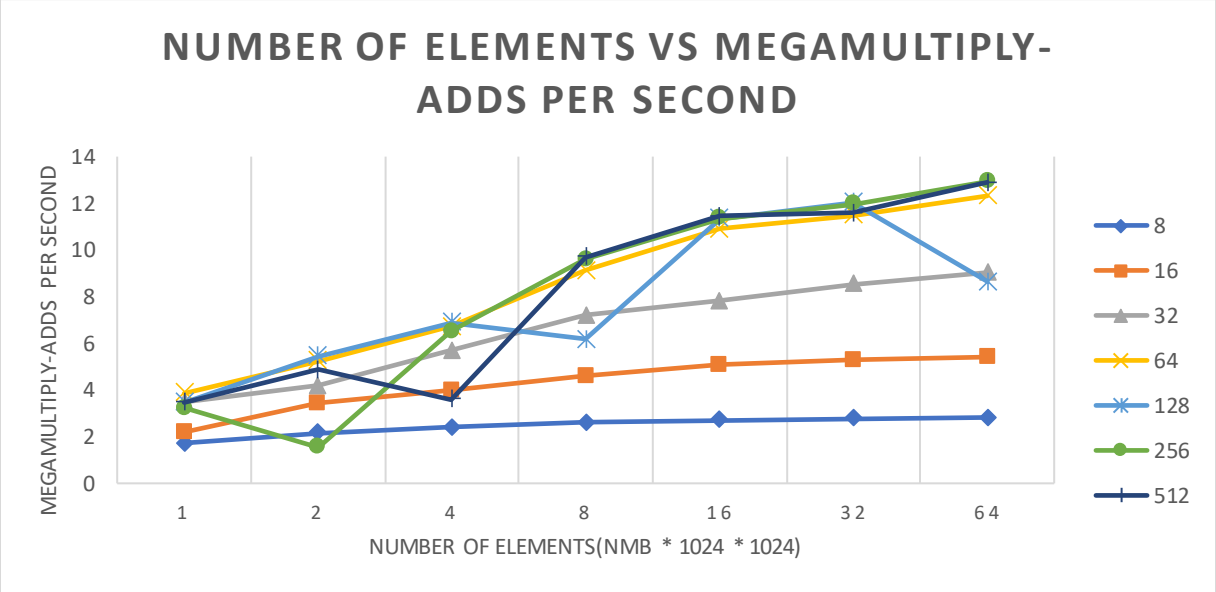
1. What machine you ran this on
    a. I ran this on the rabbit.engr.oregonstate.edu server. I used puTTy to establish a connection and was able to run it during the day when the uptime of the server was less than 2 for all 3 times.
2. Show the Tables and Graphs
    a. Multiply (Number of Elements vs MegaMultiplies per Second)



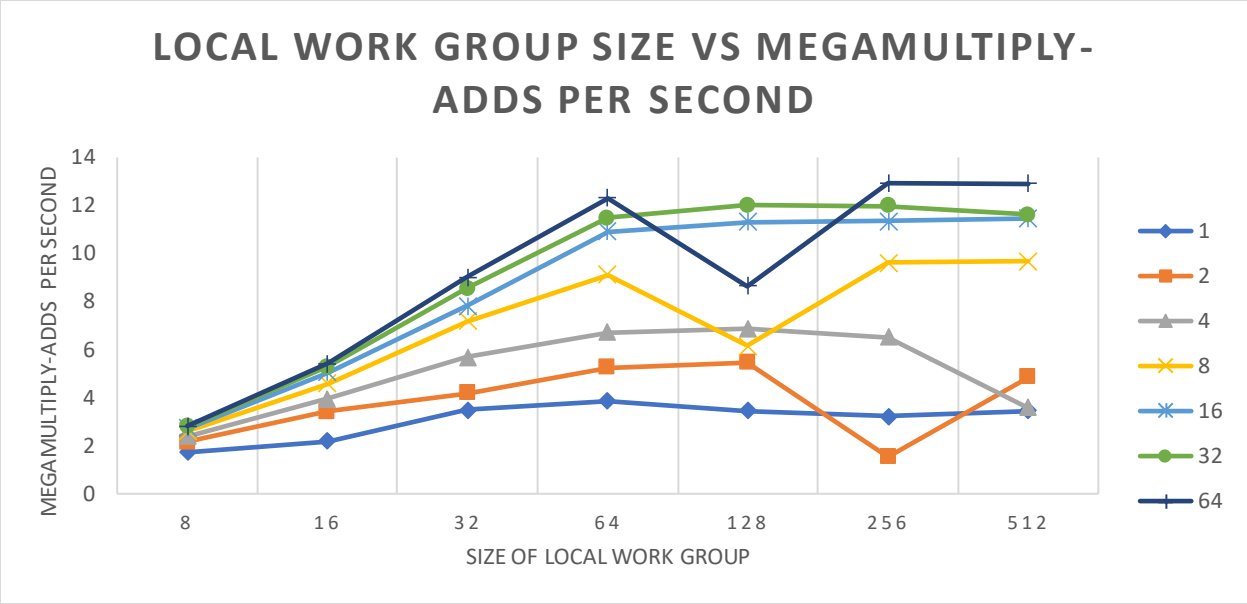
    b. Multiply (Size of Local Work Group vs MegaMultiplies per Second)



c. Multiply-Add (Number of Elements vs MegaMultiply-Adds per Second)



d. Multiply-Add (Size of Local Work Group vs MegaMultiply-Adds per Second)



e. Raw Data for Multiply

Number of elements (NMB * 1024 * 1024)	Local Work Group Size	# of Work Groups	MegaMultiplies Per Second
1	8	131072	1.05926
2	8	262144	0.996724
4	8	524288	2.72367
8	8	1048576	1.86664
16	8	2097152	2.75394
32	8	4194304	2.83688
64	8	8388608	2.87786
1	16	65536	2.30506
2	16	131072	3.40538
4	16	262144	4.60408
8	16	524288	4.28869
16	16	1048576	5.10103
32	16	2097152	6.12861
64	16	4194304	5.02194
1	32	32768	3.19858
2	32	65536	4.30548
4	32	131072	5.28929
8	32	262144	1.9195
16	32	524288	9.66869
32	32	1048576	7.58284
64	32	2097152	7.42933
1	64	16384	3.56695
2	64	32768	3.61681
4	64	65536	1.94453

8	64	131072	9.59748
16	64	262144	5.51004
32	64	524288	14.5902
64	64	1048576	12.7584
1	128	8192	3.85455
2	128	16384	3.80618
4	128	32768	8.79609
8	128	65536	11.6659
16	128	131072	7.62226
32	128	262144	8.84919
64	128	524288	14.1133
1	256	4096	1.50002
2	256	8192	6.45348
4	256	16384	6.87463
8	256	32768	4.57891
16	256	65536	13.4317
32	256	131072	15.7601
64	256	262144	11.7428
1	512	2048	2.53782
2	512	4096	1.05015
4	512	8192	6.07885
8	512	16384	10.5532
16	512	32768	13.2721
32	512	65536	9.83972
64	512	131072	15.9214

f. Raw Data for Multiply-Add

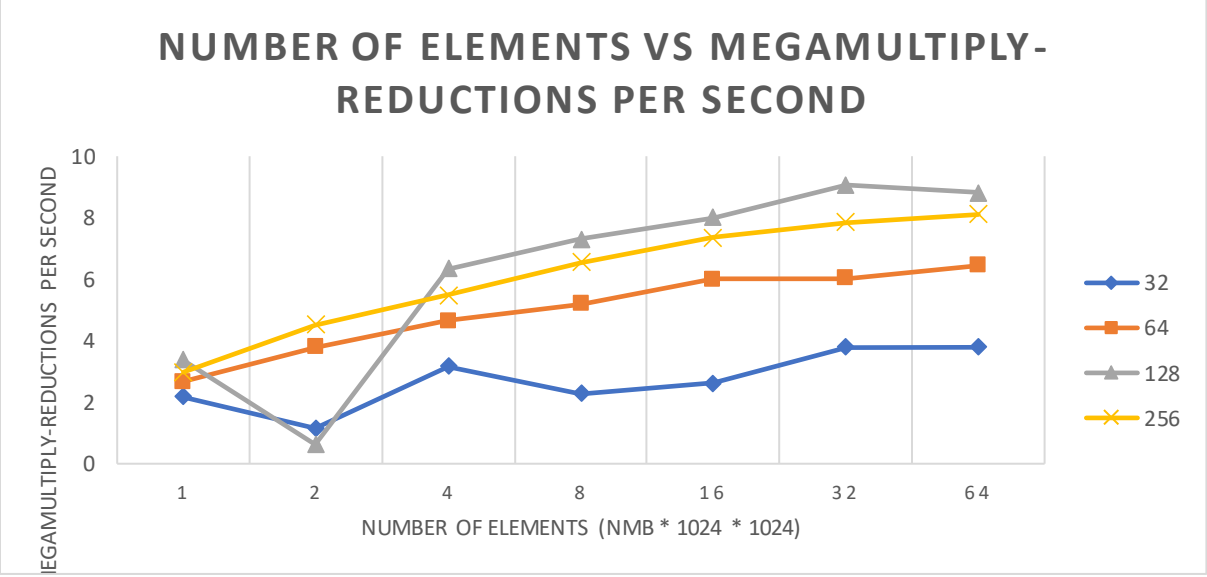
Number of elements (NMB * 1024 * 1024)	Local Work Group Size	Number of Work Groups	Mega-Multiply Adds per Second
1	8	131072	1.72472
2	8	262144	2.15327
4	8	524288	2.38183
8	8	1048576	2.60432
16	8	2097152	2.70816
32	8	4194304	2.77978
64	8	8388608	2.82183
1	16	65536	2.19354
2	16	131072	3.43732
4	16	262144	3.97115
8	16	524288	4.59386
16	16	1048576	5.06578
32	16	2097152	5.27997

64	16	4194304	5.41152
1	32	32768	3.48222
2	32	65536	4.17668
4	32	131072	5.6749
8	32	262144	7.18782
16	32	524288	7.82484
32	32	1048576	8.53575
64	32	2097152	9.03206
1	64	16384	3.85455
2	64	32768	5.24201
4	64	65536	6.70945
8	64	131072	9.11985
16	64	262144	10.915
32	64	524288	11.4682
64	64	1048576	12.3184
1	128	8192	3.4603
2	128	16384	5.46002
4	128	32768	6.87463
8	128	65536	6.15865
16	128	131072	11.3279
32	128	262144	12.0227
64	128	524288	8.64693
1	256	4096	3.23624
2	256	8192	1.53644
4	256	16384	6.50358
8	256	32768	9.60797
16	256	65536	11.337
32	256	131072	11.9756
64	256	262144	12.9307
1	512	2048	3.46303
2	512	4096	4.8651
4	512	8192	3.58147
8	512	16384	9.68733
16	512	32768	11.4588
32	512	65536	11.5986
64	512	131072	12.9005

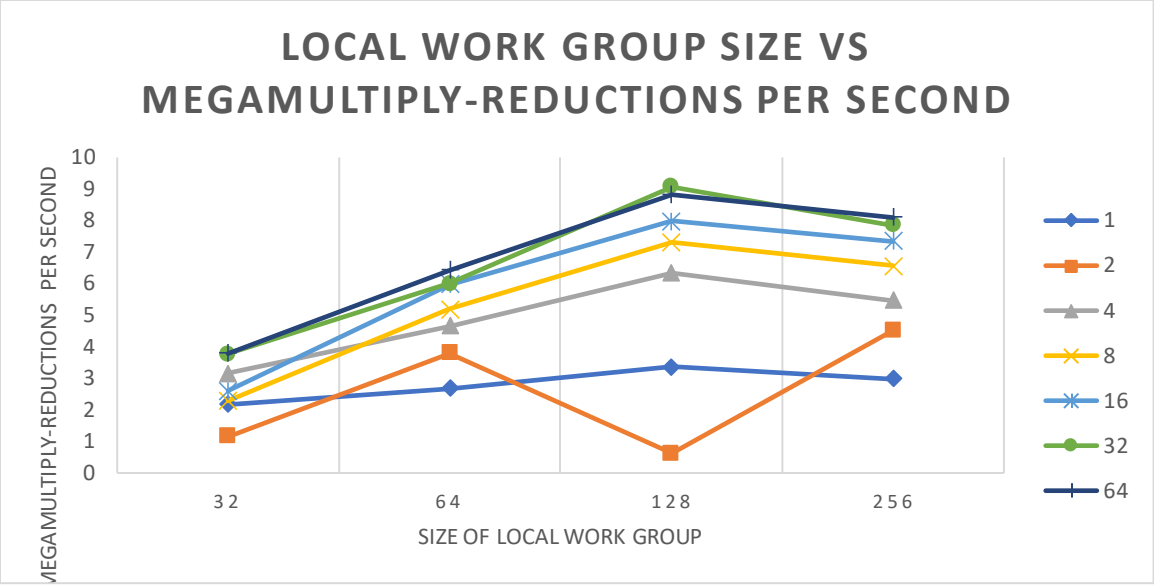
3. What patterns are you seeing in the performance curves?
    a. One pattern that I noticed immediately was that the data I recorded for only the array multiplication was very sporadic in regards the Number of elements vs MegaMultiplies per second. It is sporadic in the Local Group Size vs MegaMultiplies per second graph as well, but as the work group size increases there is the trend that the performance increases as well. However, in MegaMultiply-Add graphs there are clear trends that with

larger number of element sizes as well as larger work group sizes, there is greater performance. There are some exceptions in which performance dips, but for the most part all lines are gradually increasing.

4. Why do you think the patterns look this way?
    a. For the Multiply-Adds I believe that the graphs look this way as with larger data sets the multiply and addition are treated as a single instruction under the concept of a fused multiply add, and therefore by running this on a large data set, you are able to capitalize on this instruction simplicity. By increasing the number of work group size, we can reduce the number of work groups and run all those work groups at the same time and be able to increase the number of mega evaluations per second for both the multiply and add.
5. What is the performance difference between doing a Multiply and doing a Multiply-Add?
    a. In the data that I collected, the difference between doing a multiply and a multiply add is that a multiply-add provides greater performance as the number of elements increase. However, I believe the real difference between the Multiply and Multiply-Add is that the two should be relatively close, with the Multiply-Add yielding slightly less performance as it has to do a FMA, instead of a single multiply.
6. What does that mean for the proper use of GPU parallel computing?
    a. I believe this means that the proper use of GPU parallel computing has the capacity to perform complex operations very fast and simultaneously. This also provides the opportunity to do much more work as there are more processing units that are located on a GPU and therefore can perform simultaneous operations on many work groups
7. Show the graphs and table for Multiply-Reduction
    a. Multiply-Reduction (Number of Elements vs MegaMultiply-Reductions per second)



    b. Multiply-Reduction (Local Work Group Size vs MegaMultiply-Reductions per second)



c. Raw Data for Multiply-Reduction

Number of Elements (NMB * 1024 * 1024)	Size of Local Work Group	Number of Work Groups	MegaMultiply-Reductions per second
1	32	32768	2.16653
2	32	65536	1.15223
4	32	131072	3.1652
8	32	262144	2.28886
16	32	524288	2.60799
32	32	1048576	3.78164
64	32	2097152	3.78632
1	64	16384	2.67521
2	64	32768	3.79305
4	64	65536	4.65525
8	64	131072	5.18791
16	64	262144	5.98934
32	64	524288	6.01545
64	64	1048576	6.44166
1	128	8192	3.37274
2	128	16384	0.625122
4	128	32768	6.33724
8	128	65536	7.31332
16	128	131072	7.99282
32	128	262144	9.06405
64	128	524288	8.8209
1	256	4096	2.97769
2	256	8192	4.51777

4	256	16384	5.46851
8	256	32768	6.55447
16	256	65536	7.35844
32	256	131072	7.85059
64	256	262144	8.1049

8. What pattern are you seeing in this performance curve?
    a. One clear trend that is present in the graph of Local Work group sizes vs MegaMultiply-Reductions per second is that the performance is increasing through local work group size of 128 but then drops at work group size of 256. In the graph of Number of elements vs MegaMultiply-Reductions per second, there is a clear trend of increasing performance as number of elements increase.
9. Why do you think the pattern looks this way?
    a. I think the main reason that performance drops past local work group size of 128, is that the maximum number of work groups is exceeded after that point, and therefore when the wait is called to make sure all threads are finished, a second set of threads is waited on. This is shown in both graphs, but in the graph of number of elements vs MegaMultiply-reductions per second, we can see that for every point in the line of work group size of 128, it is greater than every point in the line of work group size of 256.
10. What does that mean for the proper use of GPU parallel computing?
    a. This shows that in GPU parallel computing, when you have less work groups doing more work, and waiting for all threads to finish computing can cause time to increase waiting for a single thread to finish computing. Therefore when applied to GPU parallel computing, this can have an effect as the number of work groups and/or the size of the work group can limit the performance.