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

Haoxiang Wang; Student ID: 932359049

1. Machine Running the Test

I used school's server "Rabbit" to run all of the tests. At the time my tests are running, I'm the only one who is using the server, so my results are sort of reliable.

2. Tables and Graphs for Multiply and Multiply-Add

2.1 Multiply Test

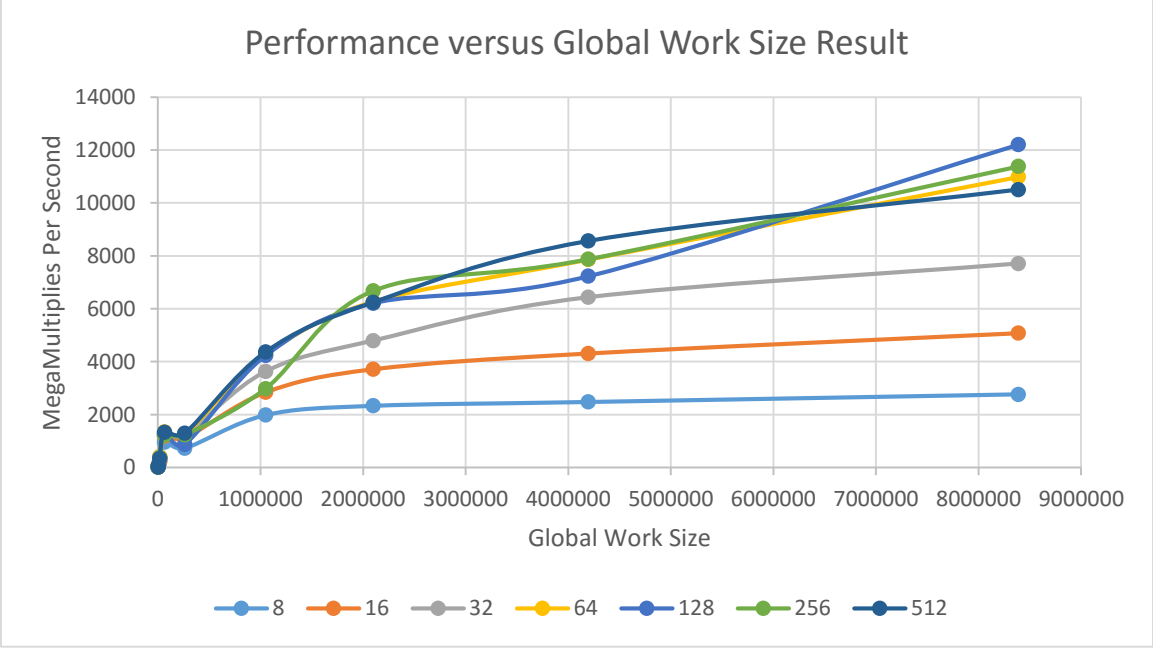
The table for Multiply test is shown below. The Global Work Size is set from 1K to 8M, and the Local Work Size is set from 8 to 512.

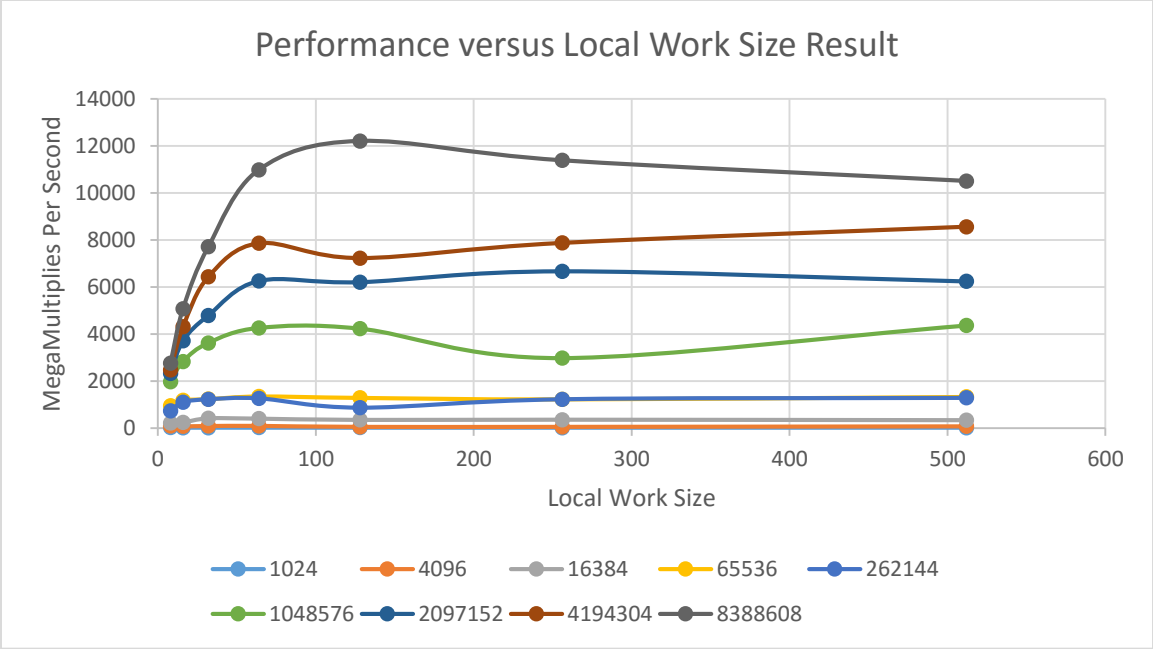
	1024	4096	16384	65536	262144	1048576	2097152	4194304	8388608
8	19.236	94.152	211.465	956.669	728.794	1976.747	2327.798	2472.658	2762.145
16	14.173	75.677	241.333	1180.763	1106.723	2835.363	3713.586	4310.226	5079.584
32	14.019	89.668	414.441	1231.515	1226.77	3624.826	4792.573	6432.574	7708.106
64	20.107	94.884	402.383	1345.689	1267.243	4259.609	6260.913	7857.82	10981.28
128	18.948	54.493	348.581	1285.603	871.321	4228.256	6208.71	7229.31	12207.19
256	13.708	54.407	357.885	1222.444	1227.669	2975.706	6672.158	7872.38	11383.94
512	13.443	75.521	337.429	1326.411	1287.861	4361.384	6239.816	8560.284	10506.23

Based on this table, two graphs are drawn below.

The first one is the Performance versus Global Work Size, with a series of Constant Local Work Size curves.

The second one is the Performance versus Local Work Size, with a series of Constant Global Work Size curves.





2.2 Multiply-Add Test

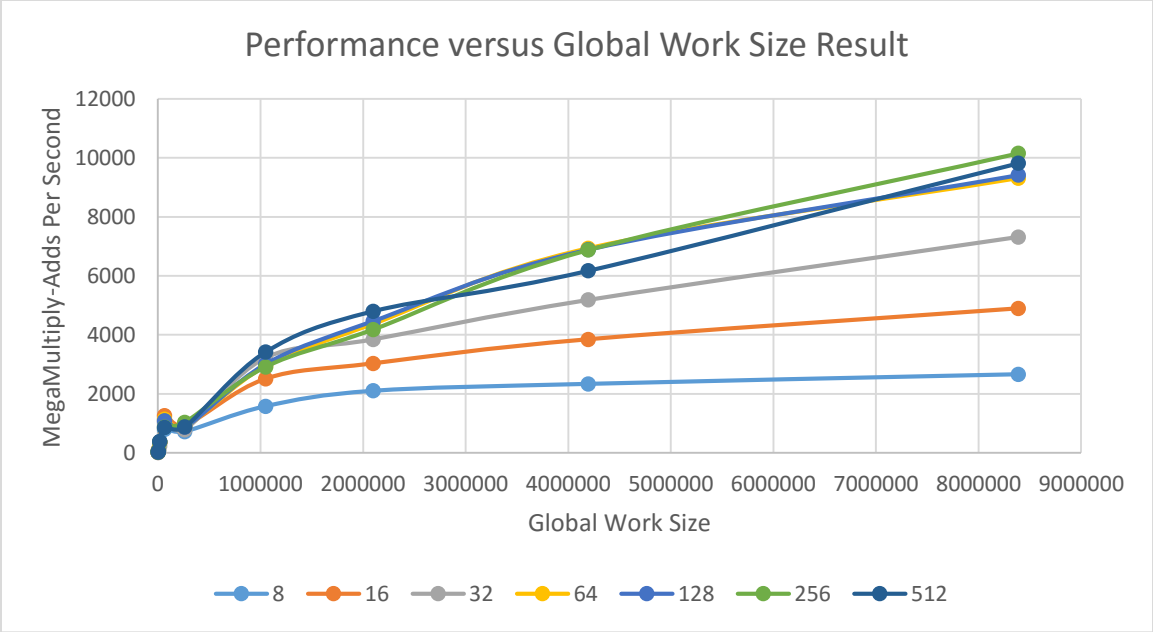
The table for Multiply-Add test is shown below. The Global Work Size is set from 1K to 8M, and the Local Work Size is set from 8 to 512.

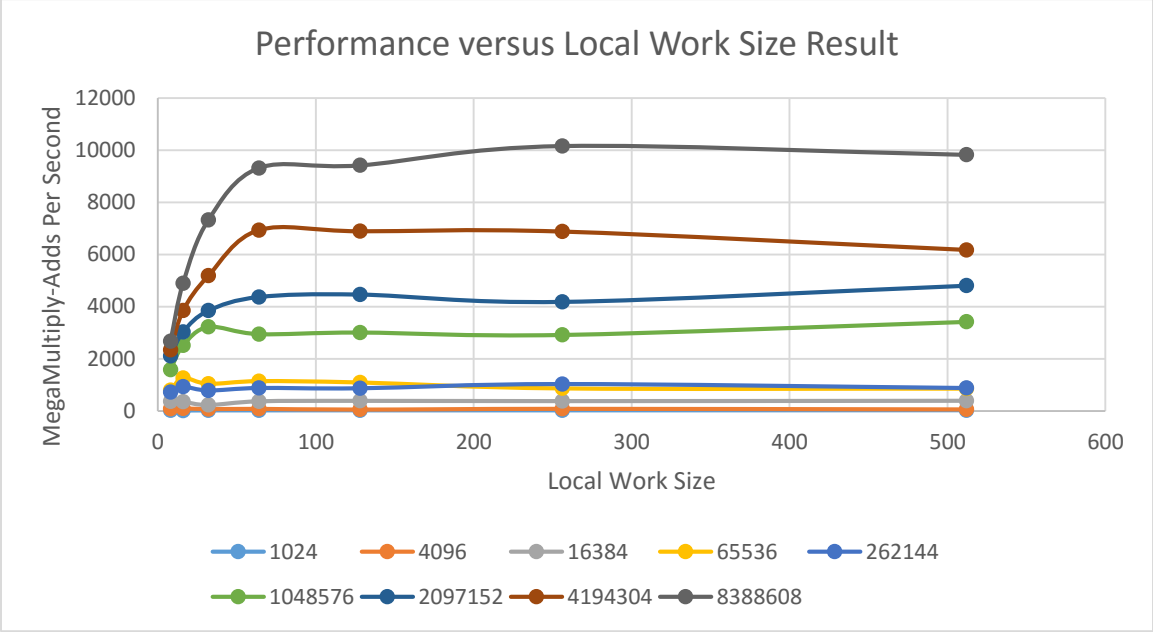
	1024	4096	16384	65536	262144	1048576	2097152	4194304	8388608
8	13.828	82.213	363.055	790.802	722.835	1578.651	2102.804	2338.05	2664.568
16	13.629	95.843	347.342	1255.24	923.402	2508.421	3029.805	3846.968	4895.994
32	17.556	69.493	230.978	1040.219	782.475	3219.322	3845.234	5182.794	7318.333
64	18.795	78.002	364.318	1139.538	877.996	2946.333	4368.016	6931.686	9312.346
128	19.978	52.001	382.14	1089.03	866.269	3007.276	4463.995	6886.575	9413.879
256	18.918	78.903	372.274	864.778	1031.588	2913.578	4181.988	6876.859	10155.27
512	22.346	57.236	384.68	856.693	877.361	3417.037	4802.016	6173.171	9818.095

Based on this table, two graphs are drawn below.

The first one is the Performance versus Global Work Size, with a series of Constant Local Work Size curves.

The second one is the Performance versus Local Work Size, with a series of Constant Global Work Size curves.





3. Behavior Explanation

3.1 Patterns

Multiply test and the Multiply-Add test both have similar patterns on their behaviors. When using the fixed local work size, while the global work size is increasing, the performances are increased under most of the local work size condition. However, when the local work sizes are small, e.g. 8 or 16, the performance increases really slowly while the global work size increasing. Also, when dealing with the small global work size, no matter how large the local work

size is, they perform similarly, and when dealing with large global work size, different local work size will have much different performances.

When using the fixed global work size, while the local work size is increasing, the performance first increases but after a certain size, it remains in a constant value. Some small global work size even don't have an obviously increasing while the local work size increases. Also, when dealing with small local work size, no matter how large the global work size is, they perform similarly, and when dealing with large local work size, different global work size will have much different performances.

3.2 Analysis

The patterns I resulted from the tests really make sense in my opinion. Under the fixed local work size condition, for one local work size, under the performance limitation, when the global work size is increased, there are more work for computing units to do, so the performance also increases. Once the performance reaches the limitation, the local work units will be fully used to finish the work, there are no more computing units in local work space, so the performance stays in a constant value. When dealing with problems with small global work size, all of the local work sizes haven't reached their limitation, so they perform in similar result. However, when dealing with problems with larger global work size, the different limitation will result in different performances.

Under the fixed global work size condition, for one global work size, while the local work size increasing, the performances are increased till some certain point. Before this point, the smaller local work sizes all reach their performance limitations, so the larger local work size is, the better it performs. After the point, the larger local work sizes won't reach their performance limitations, so they perform similarly. Comparing different global work size, the larger it size is, the better performance it has, since larger local work sizes don't reach their limitations, the more work they have, the better performance they will show. As for the small global work size, none of the local work size reach its performance limitation, so they perform similarly all the way till the end.

3.3 Performance Differences

Multiply test and Multiply-Add test have similar pattern in graphs but they have different value ranges. It could be found in the tables that Multiply-Add test has smaller value in performance then Multiply test does. This is because Multiply-Add test has an extra addition operation in the code. The extra operation causes more computation in the process, and this will sure cost more running time to finish the work, which will result in a lower performance for sure.

3.4 GPU parallel computing Explanation

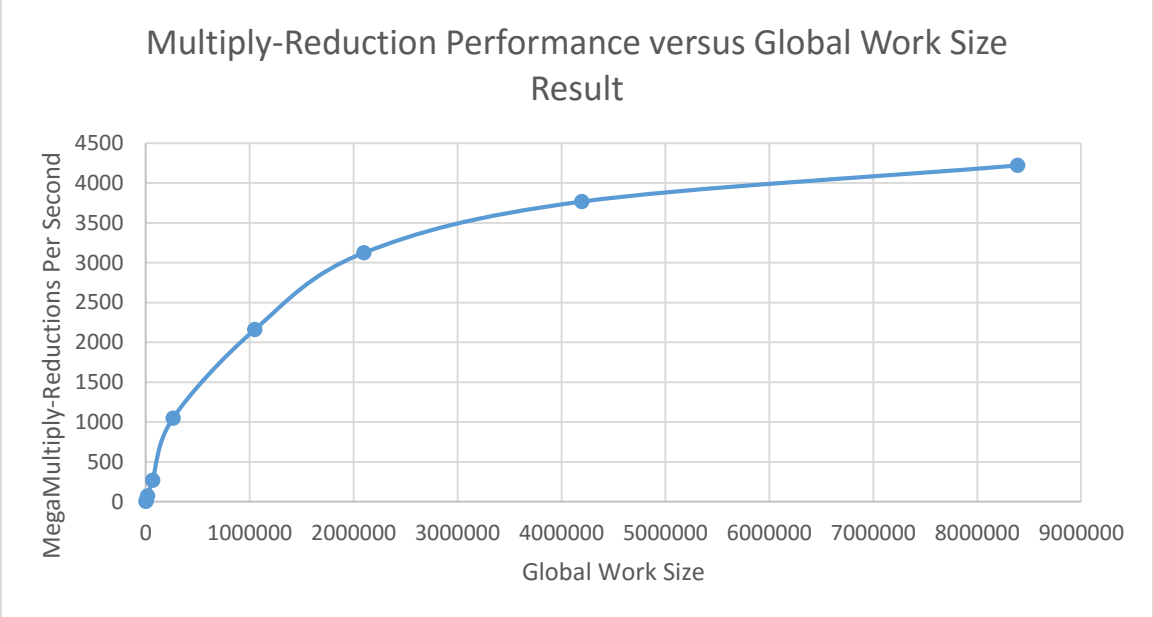
From the Graphs listed above, and comparing with the OpenMP multiple thread computation, we could find that GPU parallel computing has much better performance than multithread. This is because GPU has a lot more computation units. Therefore, GPU could be used for computing very large size problem in a short time but it also requires a good local work size setting. We could find in the graphs that when the local work size is small, the performance is poor compared to larger sizes, and we could also find that the biggest local work size, which is 512, doesn't have the best performance, so choosing a good local work size is important to let GPU parallel computing performs well.

4. Multiply-Reduction Table and Graph

The table for Multiply-Reduction test is shown below. The Global Work Size is set from 1K to 8M, and the Local Work Size is set to a constant number 32.

	1024	4096	16384	65536	262144	1048576	2097152	4194304	8388608
32	3.674	14.895	73.402	268.125	1048.652	2159.958	3123.318	3766.736	4220.624

Based on this table, one graph are drawn below. It shows how performance changes based on the global work size.



5. Multiply-Reduction Behavior Explanation

5.1 Patterns

The performance increases really fast in the beginning with the small global work size, then when dealing with larger global work size the performance increases slower than before. I believe if the global work size keep increasing, the performance will stay in a constant range and keep performs in that way.

5.2 Analysis

Similar to the previous Multiply and Multiply-Add tests, since the local work size is fixed to 32, when the global size is too small, the performance is low since the work groups have not been fully used. When the global work size becomes larger, the performance increases, and will finally reach the limitation then performs steady. Also, since we are dealing with the reduction, the performance also looks like a log function, and this also makes sense to me.

5.3 GPU parallel computing Explanation

Just like what discussed in the previous paragraph, GPU is good for dealing with large size computing problems. At the same time, reduction also is a good method to dealing with large size summation problem. Also, the way GPU do the computation fits to the reduction way. In this test, GPU calculates and stores the multiply result in a local array and then do the summation, this lets the reduction computation could be finished in a really short time. So, with a properly good choice of local work size, GPU performs really well on large size of problems and reduction problems.