Gretel Rajamoney
rajamong@oregonstate.edu
Project #6

Array Multiply & Array Multiply-Add Portions

Project Questions:

1. What machine did you run this on?
   = I ran my program on my Windows machine on Visual Studio Code utilizing the engineering server rabbit.engr.oregonstate.edu. To run my program in the terminal, I inputted the following lines of code:
   chmod u+x proj06.sh
   sh proj06.sh >& proj06.csv

2. Show the table and the two graphs?

   Results Table for Array Multiply:

Global Work Size	Local Work Size	Number of Work Groups	Performance (GigaMults/Sec)
1024	8	128	0.015
1024	16	64	0.014
1024	32	32	0.016
1024	64	16	0.013
1024	128	8	0.015
1024	256	4	0.014
1024	512	2	0.012
4096	8	512	0.07
4096	16	256	0.07
4096	32	128	0.074
4096	64	64	0.044
4096	128	32	0.053
4096	256	16	0.071
4096	512	8	0.052
16384	8	2048	0.187
16384	16	1024	0.269
16384	32	512	0.243
16384	64	256	0.296
16384	128	128	0.297
16384	256	64	0.215
16384	512	32	0.277
65536	8	8192	0.949
65536	16	4096	0.802
65536	32	2048	1.145
65536	64	1024	0.887
65536	128	512	0.938
65536	256	256	0.942
65536	512	128	1.16
262144	8	32768	0.719
262144	16	16384	1.023
262144	32	8192	0.985
262144	64	4096	1.101
262144	128	2048	1.105
262144	256	1024	1.092
262144	512	512	1.014
1048576	8	131072	1.893
1048576	16	65536	2.25
1048576	32	32768	3.062
1048576	64	16384	3.963
1048576	128	8192	3.329
1048576	256	4096	3.845
1048576	512	2048	4.315
2097152	8	262144	2.274
2097152	16	131072	3.384
2097152	32	65536	5.196
2097152	64	32768	5.783
2097152	128	16384	5.899
2097152	256	8192	5.033
2097152	512	4096	5.749
4194304	8	524288	2.643
4194304	16	262144	4.412
4194304	32	131072	6.636
4194304	64	65536	8.132
4194304	128	32768	7.721
4194304	256	16384	9.622
4194304	512	8192	7.694
8388608	8	1048576	2.666
8388608	16	524288	4.627
8388608	32	262144	8.119
8388608	64	131072	11.009
8388608	128	65536	12.41
8388608	256	32768	11.162
8388608	512	16384	10.712

   Pivot Table of Performance in GigaMults/Second for Array Multiply:

		Global Work Size								
		1024	4096	16384	65536	262144	1048576	2097152	4194304	8388608
Local Work Size	8	0.015	0.07	0.187	0.949	0.719	1.893	2.274	2.643	2.666
	16	0.014	0.07	0.269	0.802	1.023	2.25	3.384	4.412	4.627
	32	0.016	0.074	0.243	1.145	0.985	3.062	5.196	6.636	8.119
	64	0.013	0.044	0.296	0.887	1.101	3.963	5.783	8.132	11.009
	128	0.015	0.053	0.297	0.938	1.105	3.329	5.899	7.721	12.41
	256	0.014	0.071	0.215	0.942	1.092	3.845	5.033	9.622	11.162
	512	0.012	0.052	0.277	1.16	1.014	4.315	5.749	7.694	10.712

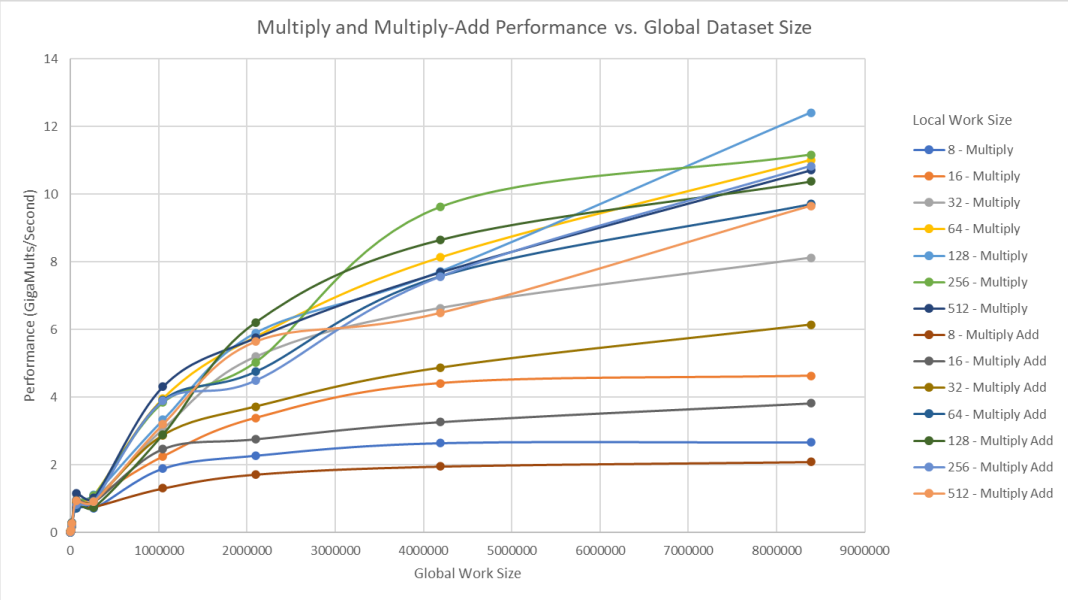
Results Table for Array Multiply-Adds:

Global Work Size	Local Work Size	Number of Work Groups	Performance (GigaMult-Adds/Sec)
1024	8	128	0.012
1024	16	64	0.009
1024	32	32	0.017
1024	64	16	0.015
1024	128	8	0.01
1024	256	4	0.012
1024	512	2	0.019
4096	8	512	0.05
4096	16	256	0.068
4096	32	128	0.041
4096	64	64	0.047
4096	128	32	0.066
4096	256	16	0.056
4096	512	8	0.061
16384	8	2048	0.197
16384	16	1024	0.195
16384	32	512	0.179
16384	64	256	0.289
16384	128	128	0.27
16384	256	64	0.202
16384	512	32	0.262
65536	8	8192	0.755
65536	16	4096	0.954
65536	32	2048	0.86
65536	64	1024	0.722
65536	128	512	0.898
65536	256	256	0.832
65536	512	128	0.935
262144	8	32768	0.753
262144	16	16384	0.895
262144	32	8192	0.928
262144	64	4096	0.929
262144	128	2048	0.735
262144	256	1024	0.912
262144	512	512	0.903
1048576	8	131072	1.304
1048576	16	65536	2.463
1048576	32	32768	2.879
1048576	64	16384	3.914
1048576	128	8192	2.901
1048576	256	4096	3.885
1048576	512	2048	3.204
2097152	8	262144	1.713
2097152	16	131072	2.754
2097152	32	65536	3.722
2097152	64	32768	4.755
2097152	128	16384	6.214
2097152	256	8192	4.49
2097152	512	4096	5.646
4194304	8	524288	1.952
4194304	16	262144	3.261
4194304	32	131072	4.87
4194304	64	65536	7.582
4194304	128	32768	8.654
4194304	256	16384	7.569
4194304	512	8192	6.496
8388608	8	1048576	2.086
8388608	16	524288	3.814
8388608	32	262144	6.137
8388608	64	131072	9.711
8388608	128	65536	10.376
8388608	256	32768	10.835
8388608	512	16384	9.658

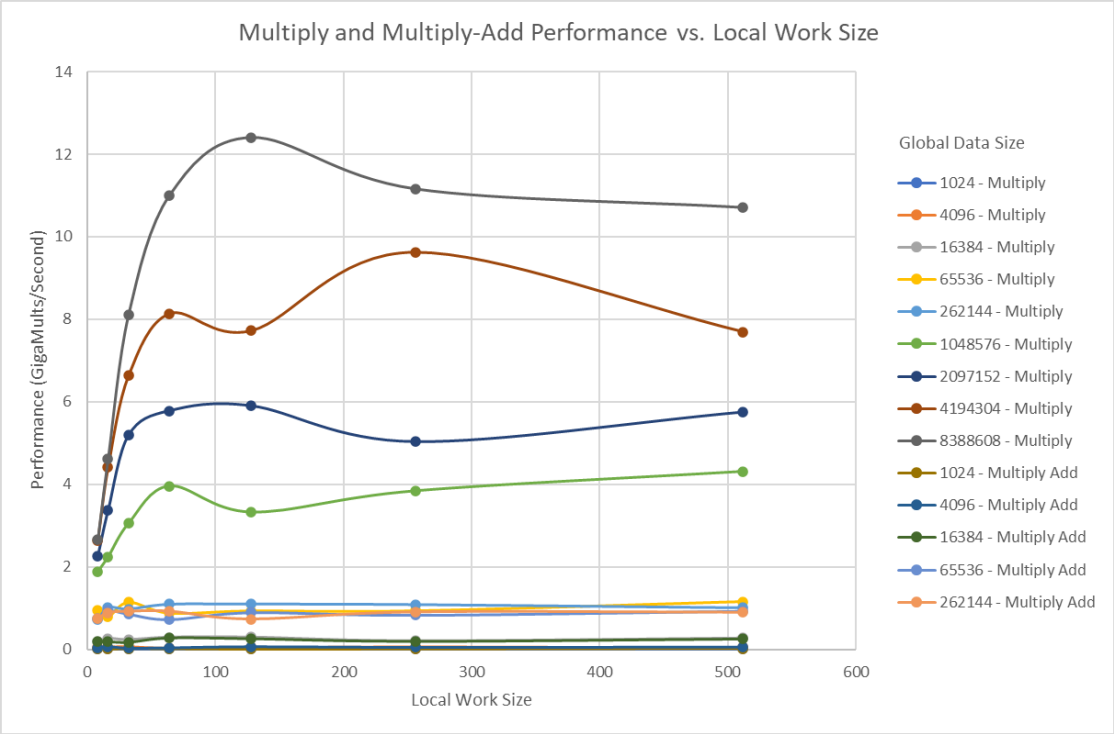
Pivot Table of Performance in GigaMult-Adds/Second for Array Multiply-Adds:

		Global Work Size								
		1024	4096	16384	65536	262144	1048576	2097152	4194304	8388608
Local Work Size	8	0.012	0.05	0.197	0.755	0.753	1.304	1.713	1.952	2.086
	16	0.009	0.068	0.195	0.954	0.895	2.463	2.754	3.261	3.814
	32	0.017	0.041	0.179	0.86	0.928	2.879	3.722	4.87	6.137
	64	0.015	0.047	0.289	0.722	0.929	3.914	4.755	7.582	9.711
	128	0.01	0.066	0.27	0.898	0.735	2.901	6.214	8.654	10.376
	256	0.012	0.056	0.202	0.832	0.912	3.885	4.49	7.569	10.835
	512	0.019	0.061	0.262	0.935	0.903	3.204	5.646	6.496	9.658

Graph of Multiply and Multiply-Add Performance vs. Global Dataset Size:



Graph of Multiply and Multiply-Add Performance vs. Local Work Size:



3. What patterns are you seeing in the performance curves?
   = In the first graph representing the Multiply and Multiply-Add Performance versus Global Dataset Size, there appears to be a direct correlation between global dataset size and performance. As seen within the graph, as the global work size increases, the performance of all local work size curves increases as well. This positive correlation between the variables is present within all curves on the graph, telling us that a higher global dataset size leads to a higher overall performance. In the second graph representing the Multiply and Multiply-Add Performance versus Local Work Size, there appears to be an increase as local work size increases with a plateau once the local work size reaches a size of approximately 128GB. The global dataset size curves shown on the graph of size greater than 1048576 in the multiply category appear to increase drastically prior to leveling out, meanwhile all other curves shown appear to be horizontal lines meaning that their performances do not increase as local work size increases.

4. Why do you think the patterns look this way?
   = In the first graph representing the Multiply and Multiply-Add Performance versus Global Dataset Size, there is a positive direct correlation between global work size and performance. This pattern is prevalent within the graph because as we are adding more items to be worked on by our program, the higher the

number of calculations needing to be calculated the more resources that can be utilized to calculate them. Essentially our program is capable of performing on a larger global work size since it is able to efficiently utilize its resources to handle them. In the second graph representing the Multiply and Multiply-Add Performance versus Local Work Size, our curves level out at a local work size of 128GB since any local work size larger no longer improves the program's performance. The reasoning behind this plateau in the graph is due to the system's limit, although our program is still able to perform with local work sizes of 512GB, 128GB is the point in our graph in which the curves appear to hit this system performance plateau. This essentially means that although higher global work sizes increase the program's performance, the same can not be said for local work sizes.

5. What is the performance difference between doing a Multiply and doing a Multiply-Add?
   = In both the first graph representing the Multiply and Multiply-Add Performance versus Global Dataset Size as well as the second graph representing the Multiply and Multiply-Add Performance versus Local Work Size, the overall performance of the Multiply curves outperform all of the Multiply-Add curves. This difference in performance between computing a Multiply versus a Multiply-Add is very prominent in the graph displaying local work size since the curves representing the Multiply-Add are horizontal showing no increase in performance. This difference is also significantly clear in the first graph displaying global work sizes, since all curves representing the Multiply are above all of the curves representing the Multiply-Add.

6. What does this mean for the proper use of GPU parallel computing?
   = The two programs calculating both Multiply as well as Multiply-Add have a great impact on understanding the proper use of GPU parallel computing. Firstly, as we understood with the first graph, increasing the global dataset size of our program greatly benefits the overall performance of the program. By understanding that our program is able to efficiently utilize resources in order to compute Multiply and Multiply-Add calculations for larger dataset sizes, we can properly use our GPU by providing it with more global data. We also understood the impact that local sizes have on our programs computing performance. Local sizes that are greater than 128GB do not improve the performance of our program, instead our performance levels out into a horizontal line. Through understanding the impacts that these two variables have on the performance of our program as it computes Multiply and Multiply-Add calculation, we can properly assign array sizes that efficiently make use of our GPU.

Array Multiply-Reduction Portions

Project Questions:

1. Show this table and graph?

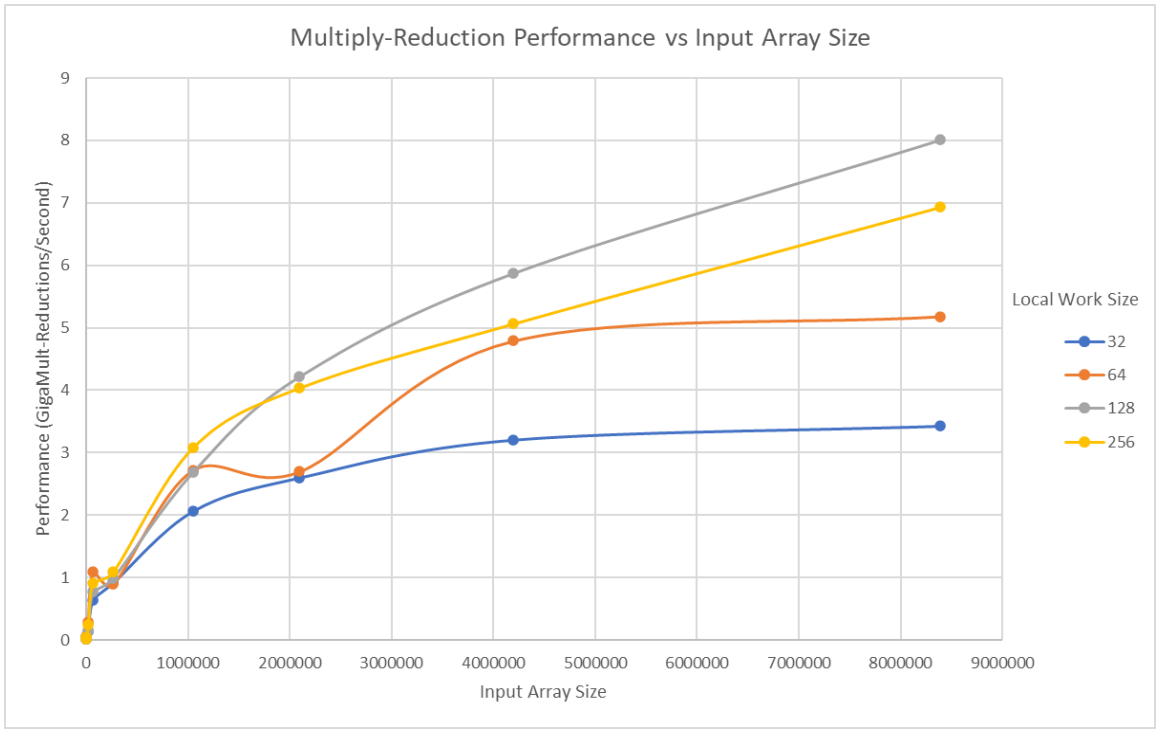
   Results Table for Part 3:

Global Work Size	Local Work Size	Number of Work Groups	Performance (GigaMult-Reductions/Sec)
1024	32	32	0.014
1024	64	16	0.014
1024	128	8	0.015
1024	256	4	0.013
4096	32	128	0.053
4096	64	64	0.063
4096	128	32	0.042
4096	256	16	0.05
16384	32	512	0.139
16384	64	256	0.286
16384	128	128	0.154
16384	256	64	0.236
65536	32	2048	0.641
65536	64	1024	1.092
65536	128	512	0.769
65536	256	256	0.912
262144	32	8192	0.919
262144	64	4096	0.895
262144	128	2048	0.994
262144	256	1024	1.091
1048576	32	32768	2.065
1048576	64	16384	2.724
1048576	128	8192	2.693
1048576	256	4096	3.083
2097152	32	65536	2.6
2097152	64	32768	2.701
2097152	128	16384	4.222
2097152	256	8192	4.038
4194304	32	131072	3.205
4194304	64	65536	4.791
4194304	128	32768	5.871
4194304	256	16384	5.062
8388608	32	262144	3.426
8388608	64	131072	5.182
8388608	128	65536	8.006
8388608	256	32768	6.929

   Pivot Table of Performance in GigaMult-Reductions/Second:

		Global Work Size								
		1024	4096	16384	65536	262144	1048576	2097152	4194304	8388608
Local Work Size	32	0.014	0.053	0.139	0.641	0.919	2.065	2.6	3.205	3.426
	64	0.014	0.063	0.286	1.092	0.895	2.724	2.701	4.791	5.182
	128	0.015	0.042	0.154	0.769	0.994	2.693	4.222	5.871	8.006
	256	0.013	0.05	0.236	0.912	1.091	3.083	4.038	5.062	6.929

   Graph of Multiply-Reduction Performance vs. Input Array Size:



2. What pattern are you seeing in this performance curve?
   = In the graph above displaying the Multiply-Reduction Performance versus Input Array Size, there appears to be a direct positive correlation between the variables. As the input array size increases, the performance also appears to increase. The curve that performs the best out of all four curves present on the graph, is the curve representing a local work size of 128GB.

3. Why do you think the pattern looks this way?
   = Similarly to what was stated in the questions above, adding more items to be worked on by our program, the higher the number of calculations needing to be calculated the more resources that can be utilized to calculate them. Essentially our program is capable of performing on a larger global work size since it is able to efficiently utilize its resources to handle them. Our curve showing a local work size of 128GB has the highest performance, because our system's limit affects the performance of the program when work sizes are larger than that value. Again as stated above in the prior question pertaining to graph patterns, although higher global work sizes increase the program's performance, the same can not be said for local work sizes since 128GB outperforms the other curves present within the graph.

4. What does that mean for the proper use of GPU parallel computing?
   = Repeating what was said above, increasing the global dataset size of our program greatly benefits the overall performance of the program. By understanding that our program is able to efficiently utilize resources in order to compute Multiply-Reduction calculations for larger dataset sizes, we can properly use our GPU by providing it with more global data. We also understood the impact that local sizes have on our programs computing performance. Local sizes that are greater than 128GB do not improve the performance of our program. Through understanding the impacts that these two variables have on the performance of our program as it computes the Multiply-Reduction calculation, we can properly assign array sizes that efficiently make use of our GPU.