JONATHAN PERRY
6/3/18

PROJECT #6

For this assignment, I created a bash script to help test my program with several different combinations of parameters – such as setting the local and global work size. To run the script, adjust permissions to execute the script and then simply enter in the following command: ./script

1. Tell what machine you ran this on
    - A Linux system on OSU’s FLIP server was used to run Project #3.

2.
    - **Table of Results**

Performance - Mult.cl

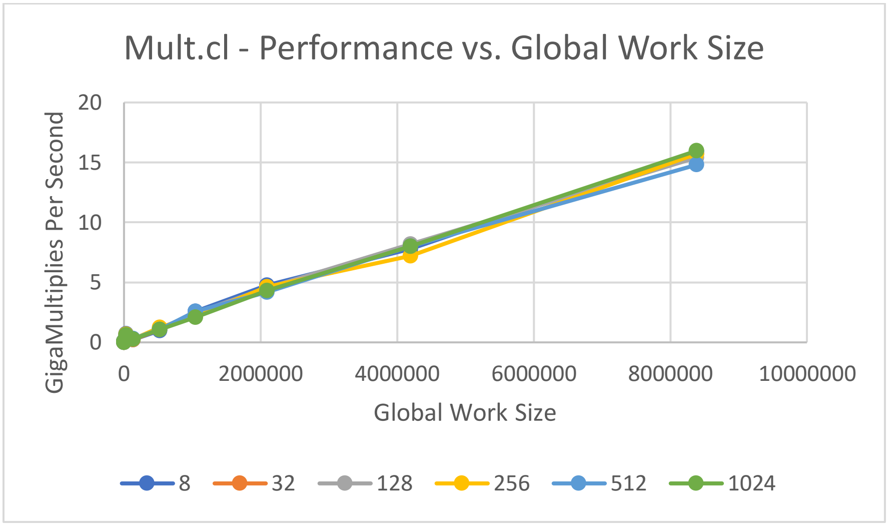
		Global Work Size									
		1024	2048	4096	32768	131072	524288	1048576	2097152	4194304	8388608
Local Work Size	8	0.014	0.039	0.073	0.724	0.211	0.969	2.598	4.774	7.792	15.957
	32	0.022	0.028	0.075	0.424	0.21	1.069	2.21	4.38	8.054	15.7
	128	0.023	0.023	0.088	0.468	0.287	1.14	2.457	4.553	8.196	15.411
	256	0.023	0.045	0.055	0.68	0.289	1.272	2.19	4.637	7.211	15.728
	512	0.022	0.027	0.085	0.52	0.315	1.073	2.594	4.176	8.014	14.808
	1024	0.014	0.027	0.091	0.642	0.286	1.065	2.074	4.337	8.015	15.977

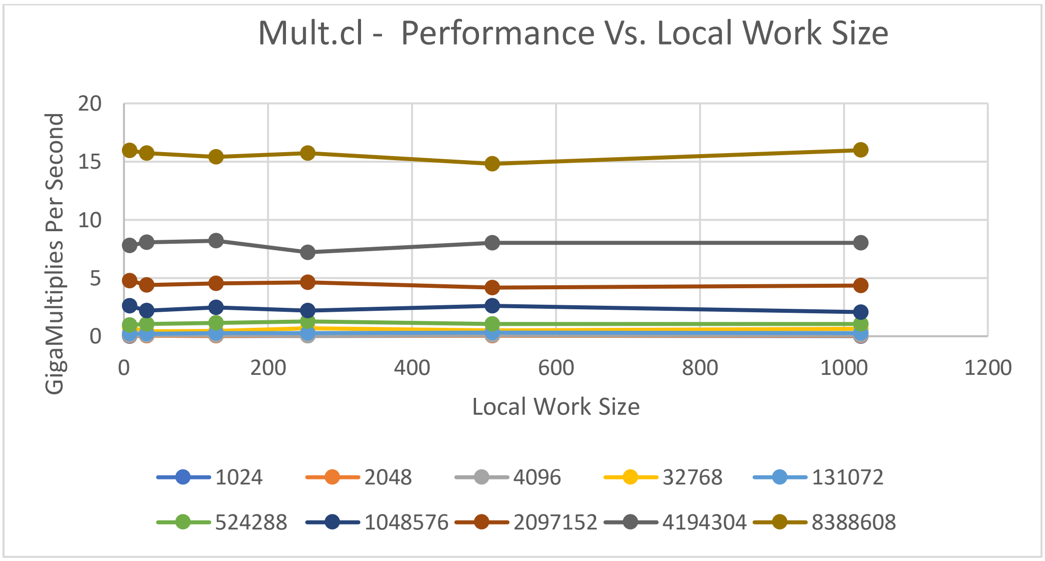
Performance - MultAdd.cl

		Global Work Size									
		1024	2048	4096	32768	131072	524288	1048576	2097152	4194304	8388608
Local Work Size	8	0.015	0.032	0.08	0.627	2.376	1.086	2.609	4.463	8.311	15.543
	32	0.016	0.037	0.075	0.432	2.55	1.308	2.283	4.288	8.225	14.226
	128	0.02	0.035	0.055	0.502	1.771	0.994	2.352	4.314	8.222	16.413
	256	0.021	0.039	0.062	0.718	2.402	1.115	2.118	4.422	8.204	14.205
	512	0.021	0.038	0.054	0.629	2.301	1.33	2.456	3.906	8.277	15.916
	1024	0.017	0.044	0.084	0.632	1.658	1.28	2.476	4.493	8.369	15.406

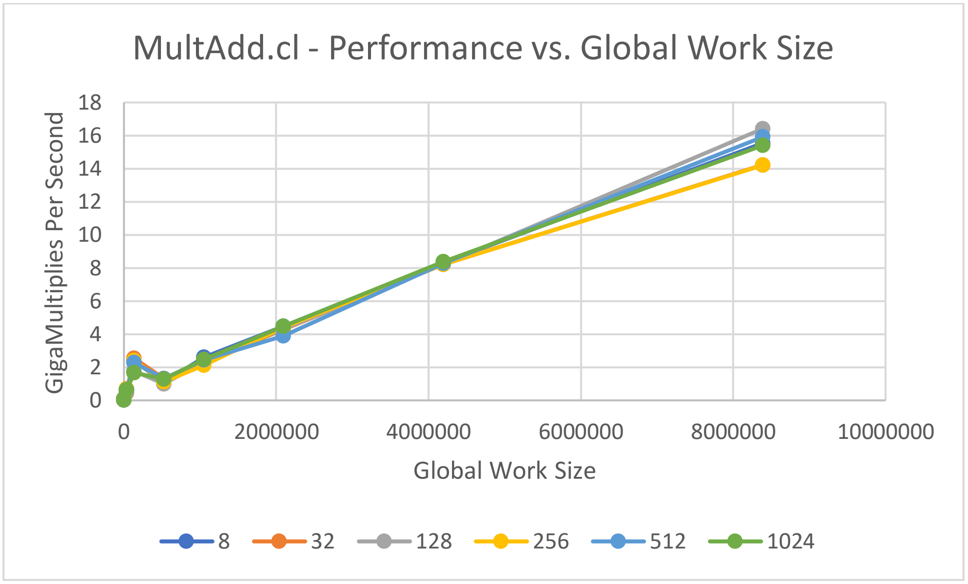
- **Graphs**

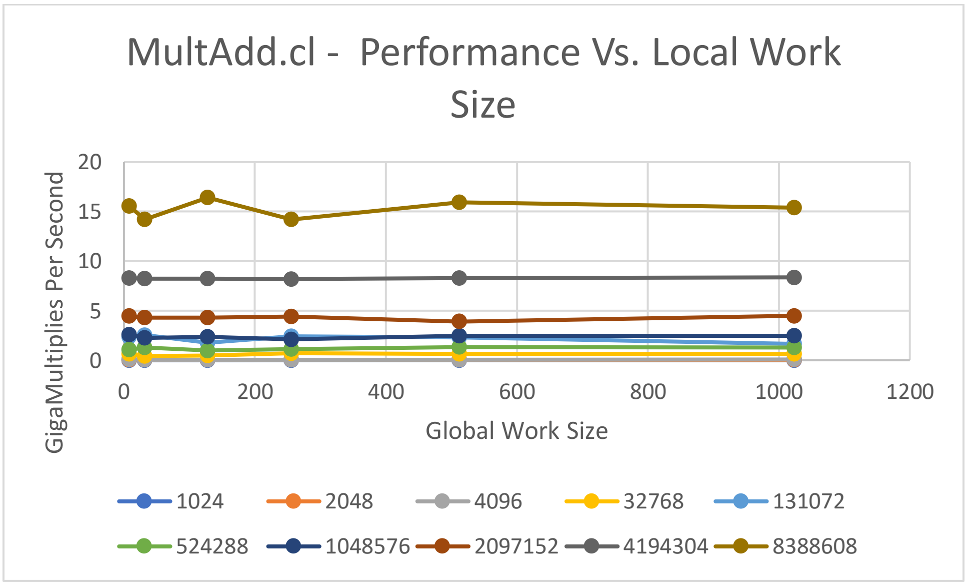
PROJECT #6


Mult.cl - Performance vs. Global Work Size
GigaMultiplies Per Second
Global Work Size
0
5
10
15
20
0
2000000
4000000
6000000
8000000
10000000
8
32
128
256
512
1024


Mult.cl - Performance Vs. Local Work Size
GigaMultiplies Per Second
Local Work Size
0
5
10
15
20
0
200
400
600
800
1000
1200
1024
2048
4096
32768
131072
524288
1048576
2097152
4194304
8388608

PROJECT #6





3. What patterns are you seeing in the performance curves?

I'm seeing that both Mult and MultAdd show patterns of a linear graph (apart from a slight hiccup in performance with a low global work size in MultAdd), whereas the total number of elements increases, we see a steady increase in performance. I'm also seeing that the local work size has no affect on performance as the local work size increases.

4. Why do you think the patterns look this way?

The patterns look this way because performance is dependent on the number of elements being processed by the GPU, and not the local work size. So, any increase in the number of elements results in an increase in performance.

5. What is the performance difference between doing a Multiply and doing a Multiply-Add?

There is no performance difference between doing a Multiply and doing a Multiply-Add when adding in the extra operation of performing addition.

6. What does this mean for the proper use of GPU parallel computing?

For operations that don’t require branch-prediction, or the need to process instructions out-of-order, GPUs offer a lot more compute power versus CPUs regarding computational problems that can be broken up into numerous pieces, where each piece is farmed out to threads on the GPU.

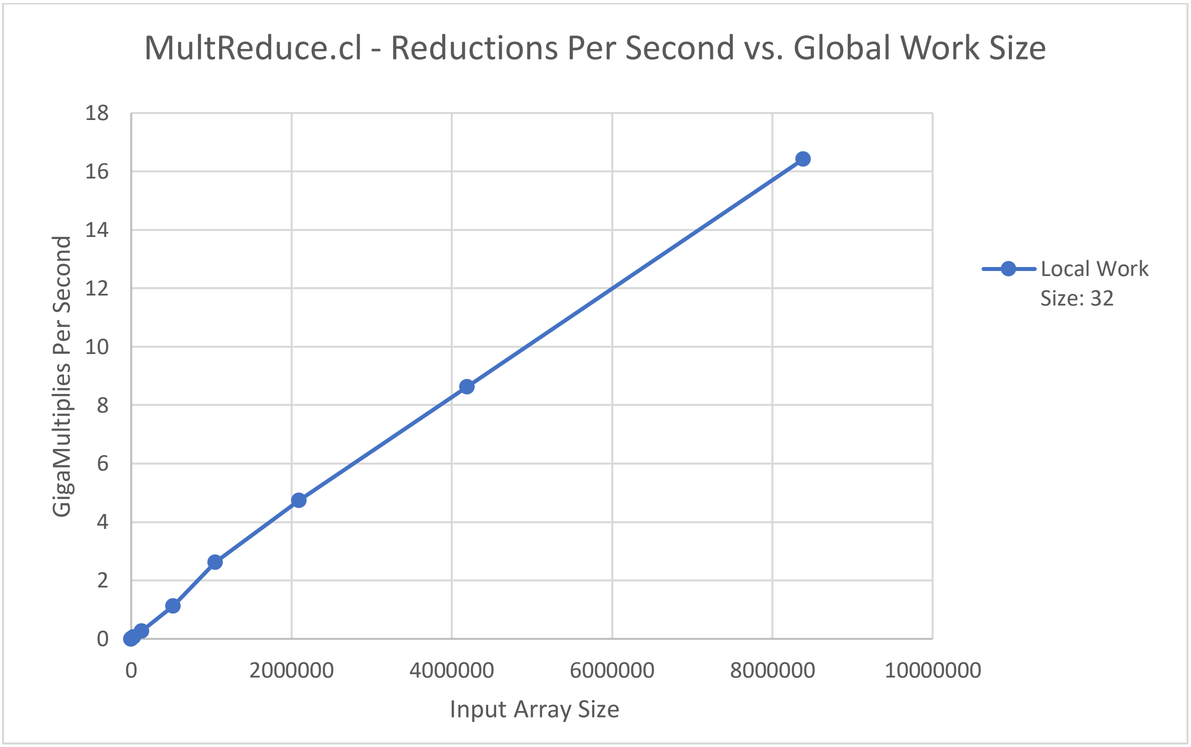
Multiply-Reduce Section

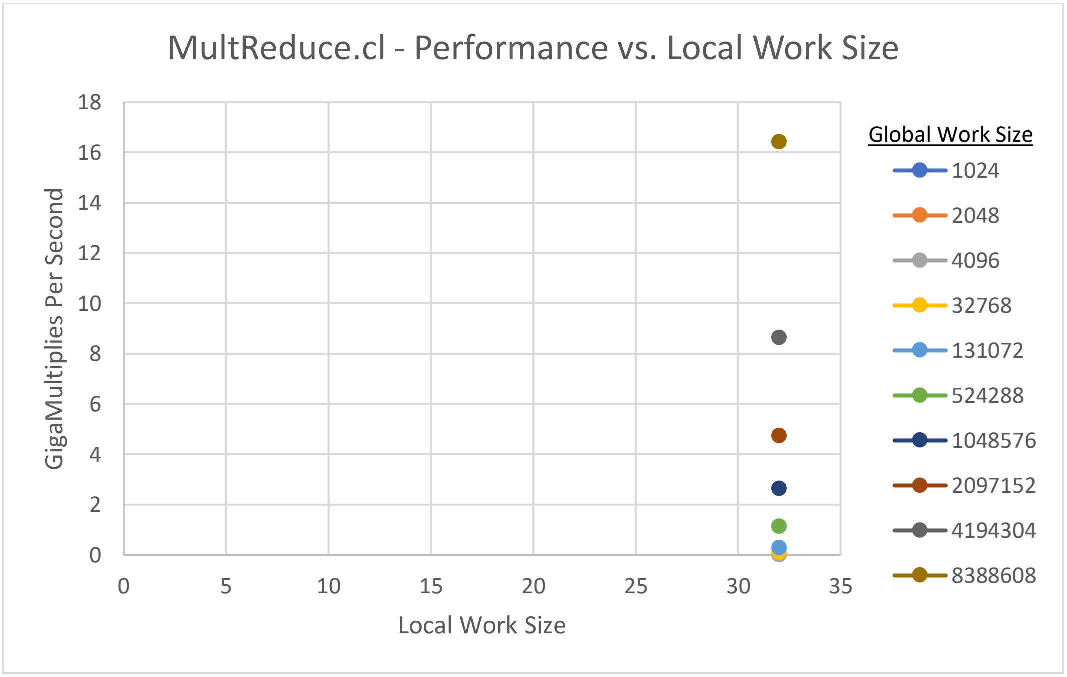
1. Tables and Graphs

Performance - MultReduce.cl

Local Work Size	**Global Work Size**										
		1024	2048	4096	32768	131072	524288	1048576	2097152	4194304	8388608
	32	0.002	0.005	0.01	0.075	0.275	1.133	2.625	4.737	8.634	16.423

PROJECT #6





2. What pattern are you seeing in the performance curve?

PROJECT #6

I'm seeing that MultReduce is showing patterns of a linear graph for performing reductions in OpenCL.

3. Why do you think the pattern looks this way?

By using reduction, we complete our computation in log(# of items) steps. We get complete the computation in log(# of items) steps by taking the result of each multiplication operation and storing it in local memory on the GPU that each work group has access to, and finishing up by performing a power of 2 summing. From this statement, the graph should reflect that of a logarithmic function, but instead shows a linear curve. I suspect that the input array size is too small to accurately depict a logarithmic curve.

4. What does this mean for the proper use of GPU parallel computing?

This means that using GPU parallel computing for a large input size will reduce your overall execution time of your program versus using a CPU. For example, a CPU paired with OpenMP may not produce results anywhere close to using a GPU for parallel programming, as the CPU may split the program up to 8 or more threads, but the GPU has over a thousand possible threads active, where each thread executes the same kernel and they're all working on a small piece of the overall problem.