CS 575

Project #6

OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

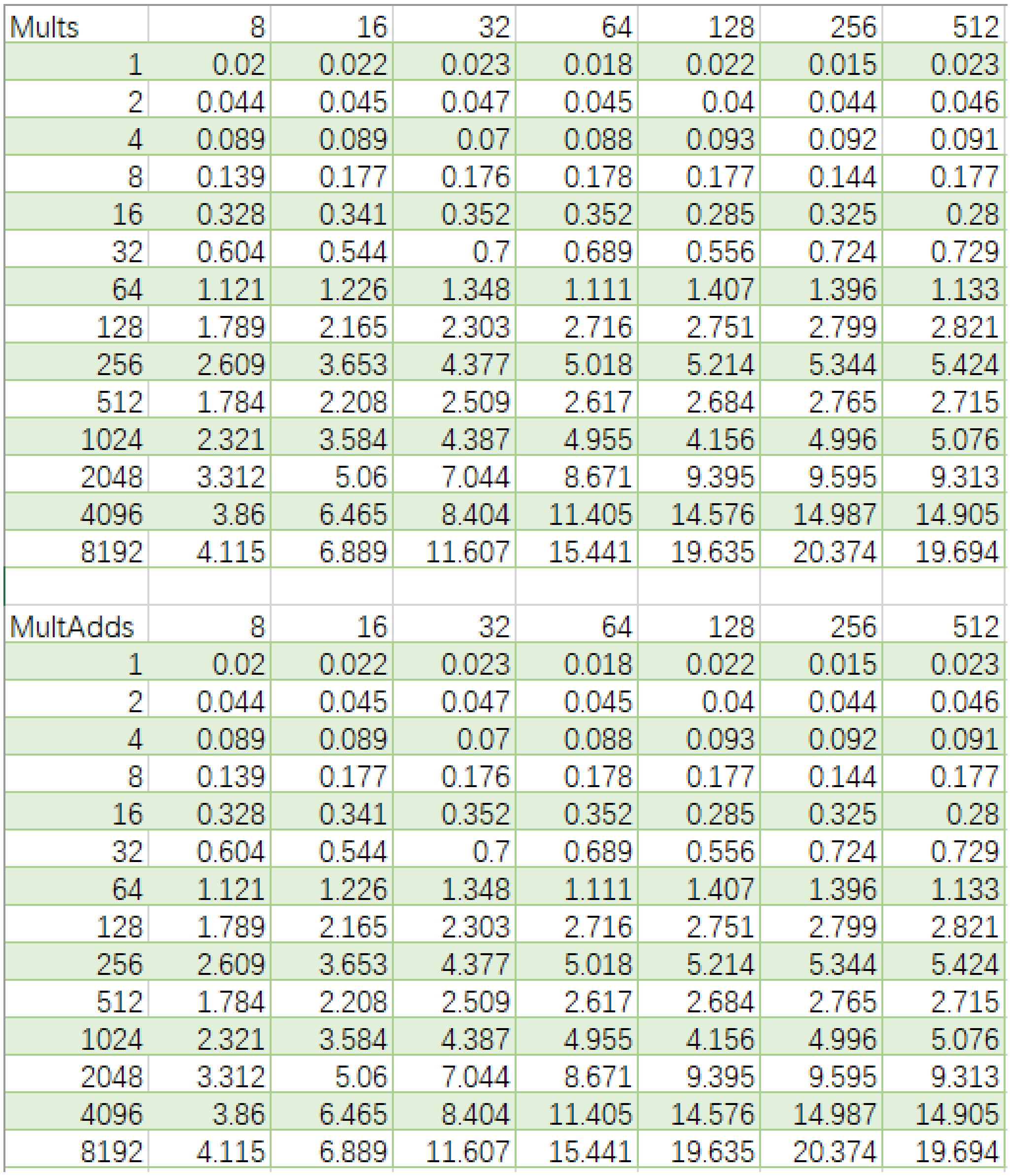
Xuming Wu

wuxum@oregonstate.edu

Code was run on the DGX Systems. Got results as:

1. The Array Multiply and the Array Multiply-Add portions

Result tables:



Mults	8	16	32	64	128	256	512
1	0.02	0.022	0.023	0.018	0.022	0.015	0.023
2	0.044	0.045	0.047	0.045	0.04	0.044	0.046
4	0.089	0.089	0.07	0.088	0.093	0.092	0.091
8	0.139	0.177	0.176	0.178	0.177	0.144	0.177
16	0.328	0.341	0.352	0.352	0.285	0.325	0.28
32	0.604	0.544	0.7	0.689	0.556	0.724	0.729
64	1.121	1.226	1.348	1.111	1.407	1.396	1.133
128	1.789	2.165	2.303	2.716	2.751	2.799	2.821
256	2.609	3.653	4.377	5.018	5.214	5.344	5.424
512	1.784	2.208	2.509	2.617	2.684	2.765	2.715
1024	2.321	3.584	4.387	4.955	4.156	4.996	5.076
2048	3.312	5.06	7.044	8.671	9.395	9.595	9.313
4096	3.86	6.465	8.404	11.405	14.576	14.987	14.905
8192	4.115	6.889	11.607	15.441	19.635	20.374	19.694

MultAdds	8	16	32	64	128	256	512
1	0.02	0.022	0.023	0.018	0.022	0.015	0.023
2	0.044	0.045	0.047	0.045	0.04	0.044	0.046
4	0.089	0.089	0.07	0.088	0.093	0.092	0.091
8	0.139	0.177	0.176	0.178	0.177	0.144	0.177
16	0.328	0.341	0.352	0.352	0.285	0.325	0.28
32	0.604	0.544	0.7	0.689	0.556	0.724	0.729
64	1.121	1.226	1.348	1.111	1.407	1.396	1.133
128	1.789	2.165	2.303	2.716	2.751	2.799	2.821
256	2.609	3.653	4.377	5.018	5.214	5.344	5.424
512	1.784	2.208	2.509	2.617	2.684	2.765	2.715
1024	2.321	3.584	4.387	4.955	4.156	4.996	5.076
2048	3.312	5.06	7.044	8.671	9.395	9.595	9.313
4096	3.86	6.465	8.404	11.405	14.576	14.987	14.905
8192	4.115	6.889	11.607	15.441	19.635	20.374	19.694

Graphs:

(1) Multiply and Multiply-Add performance versus Global Dataset Size, with a series of colored Constant-Local-Work-Size curves

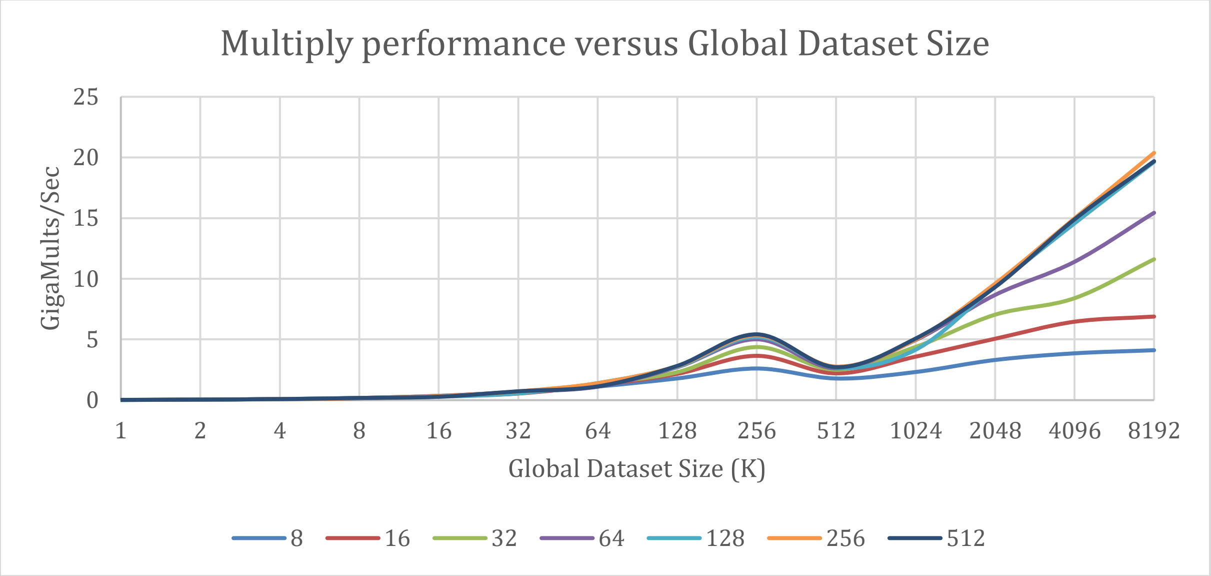


Figure 1. Multiply performance versus Global Dataset Size

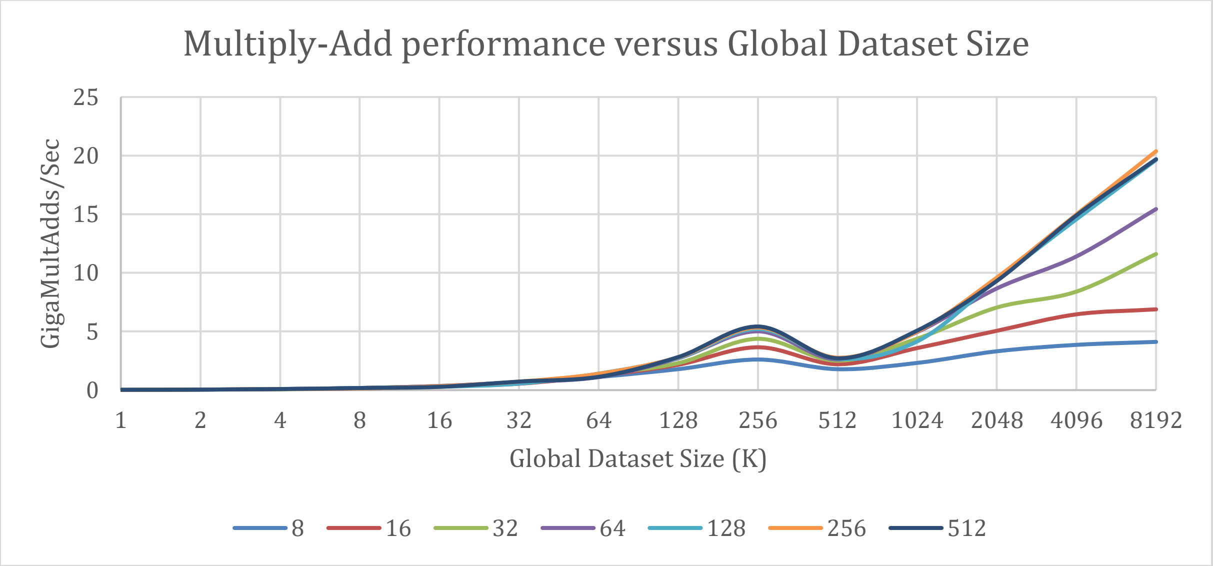


Figure 2. Multiply-Add performance versus Global Dataset Size

(2) Multiply and Multiply-Add performance versus Local Work Size, with a series of colored Constant-Global-Dataset-Size curves

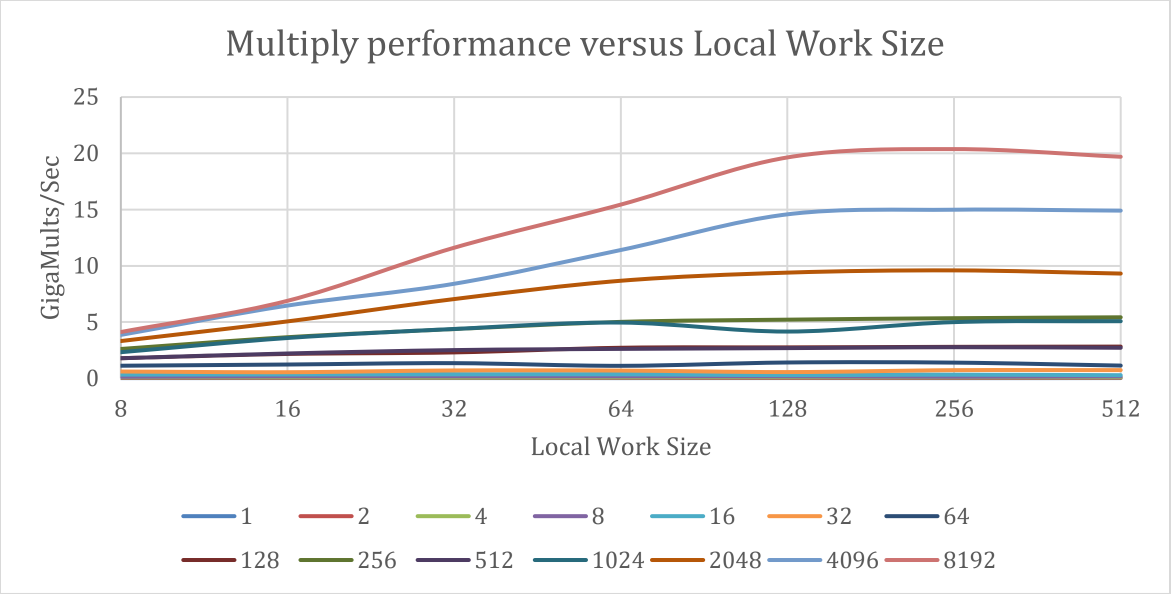


Figure 3. Multiply performance versus Local Work Size

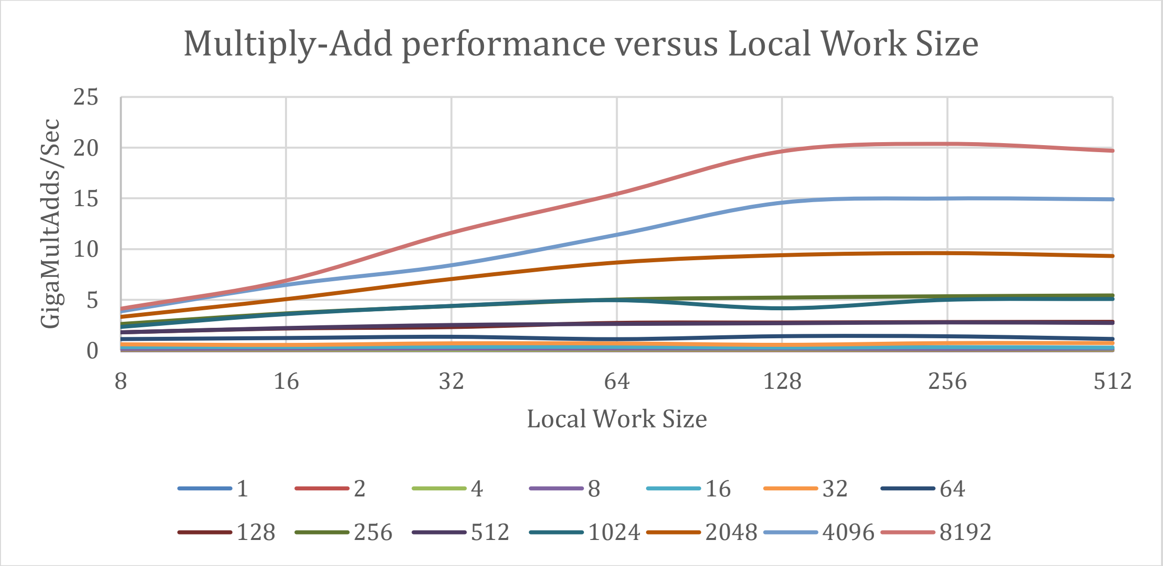


Figure 4. Multiply-Add performance versus Local Work Size

Commentary:

We can observe in the performance curves that:

1) For both Multiply and Multiply-Add, the performances grow as the Global Dataset Size increase.
2) When the Global Dataset Size is small (1K~1M), Local Work Size does not affect the performance curves in both Multiply and Multiply-Add. When the Global Dataset Size is

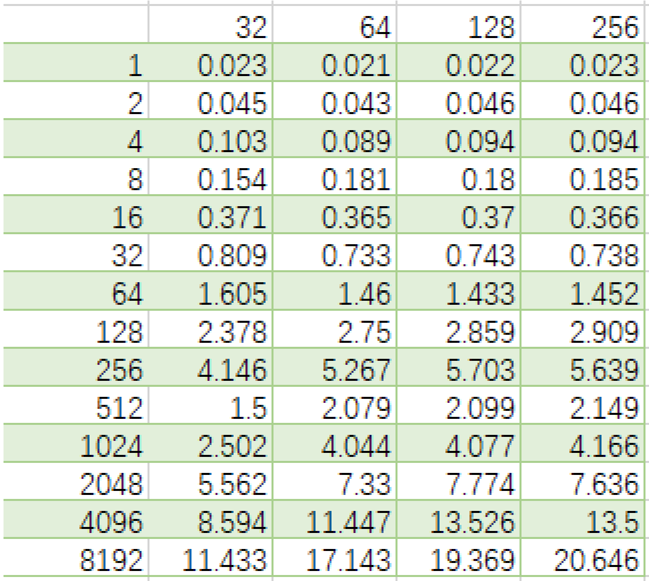
large (>1M), the performance first increases as the growth of Local Work Size and then becomes stable.

The reason of the first observation is that the growth of the Global Dataset Size increases the number of total Work-Items. More threads are assigned in the program and hence the performance increases. For the second observation, the performance curves tend to be stable because the total number of Work-items is fixed for each Global Dataset Size and will not be affected by the Local Work Size.

From the graphs above we can observe that there is no performance difference between doing a Multiply and doing a Multiply-Add. It means that the performance of GPU parallel computing is not determined by what operations are taken in the calculation. It indicates that we can use GPU parallel computing to compute complicate calculations.

2. Multiply + Reduce application

Result table:



	32	64	128	256
1	0.023	0.021	0.022	0.023
2	0.045	0.043	0.046	0.046
4	0.103	0.089	0.094	0.094
8	0.154	0.181	0.18	0.185
16	0.371	0.365	0.37	0.366
32	0.809	0.733	0.743	0.738
64	1.605	1.46	1.433	1.452
128	2.378	2.75	2.859	2.909
256	4.146	5.267	5.703	5.639
512	1.5	2.079	2.099	2.149
1024	2.502	4.044	4.077	4.166
2048	5.562	7.33	7.774	7.636
4096	8.594	11.447	13.526	13.5
8192	11.433	17.143	19.369	20.646

Graph:

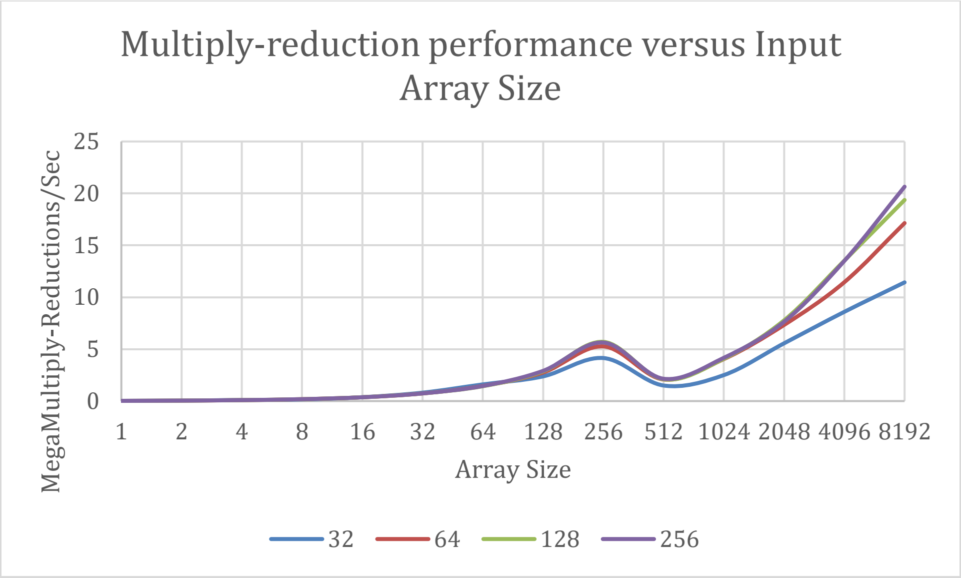


Figure 5. Multiply-reduction performance versus Input Array Size

Commentary:
We can observe that the performance of Multiply-reduction is very similar to Multiply and Multiply-Add. It is because we do summation and reduction per Work-Group, where the products of threads are available in the shared memory of the Work-Group. It means that for GPU parallel computing, doing reduction per Work-Group can provide a great performance benefit.