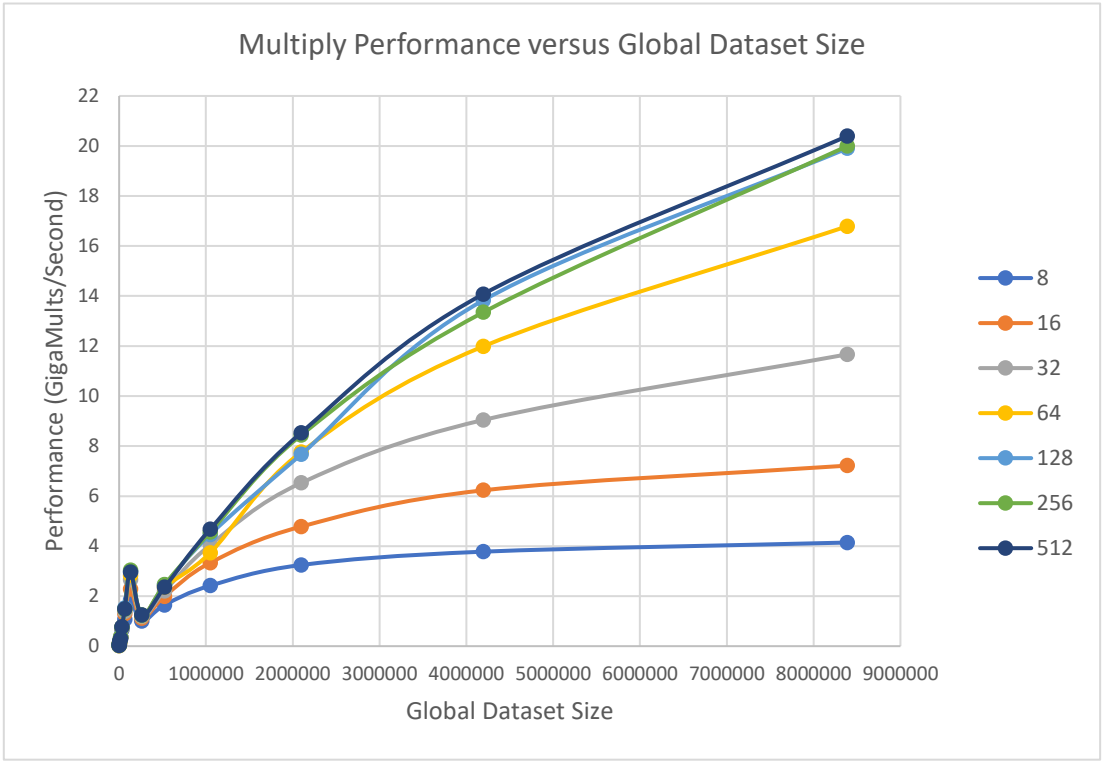
Xiaoying Li
lixiaoyi@oregonstate.edu
Project #6 OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

- **Machine:**
  I ran the project on OSU's DGX system.

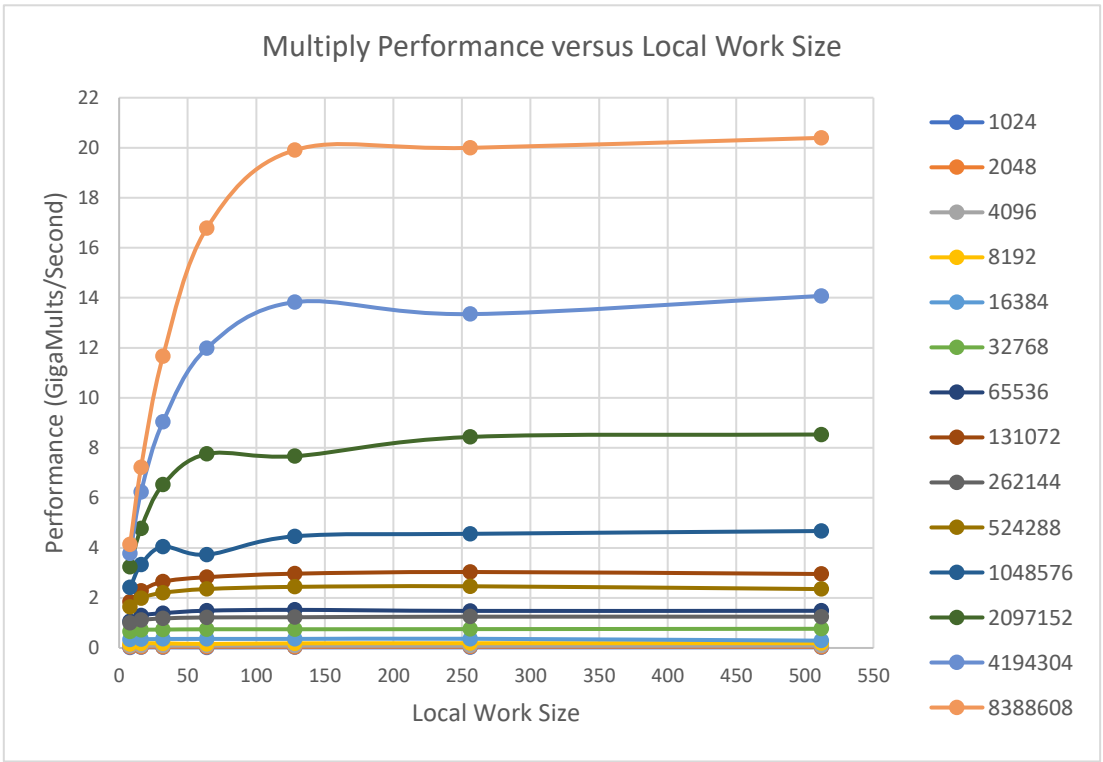
- **Table of Multiply Performance versus Global Dataset Size and Local Work Size:**

Global Dataset Size \ Local Work Size	8	16	32	64	128	256	512
1024	0.021	0.023	0.023	0.021	0.023	0.023	0.024
2048	0.046	0.046	0.046	0.047	0.045	0.048	0.046
4096	0.081	0.091	0.092	0.094	0.094	0.087	0.094
8192	0.154	0.184	0.185	0.16	0.184	0.189	0.189
16384	0.342	0.347	0.358	0.361	0.361	0.367	0.292
32768	0.661	0.723	0.734	0.748	0.747	0.753	0.766
65536	1.063	1.299	1.389	1.49	1.523	1.481	1.486
131072	1.843	2.276	2.65	2.829	2.969	3.037	2.957
262144	1.001	1.117	1.18	1.22	1.228	1.251	1.25
524288	1.646	1.993	2.203	2.356	2.442	2.466	2.354
1048576	2.417	3.333	4.044	3.73	4.463	4.563	4.675
2097152	3.245	4.781	6.526	7.757	7.667	8.436	8.534
4194304	3.778	6.232	9.043	11.986	13.826	13.348	14.076
8388608	4.143	7.218	11.668	16.789	19.904	19.995	20.395

- **Graph of Multiply Performance versus Global Dataset Size, with a series of colored Constant-Local-Work-Size curves:**



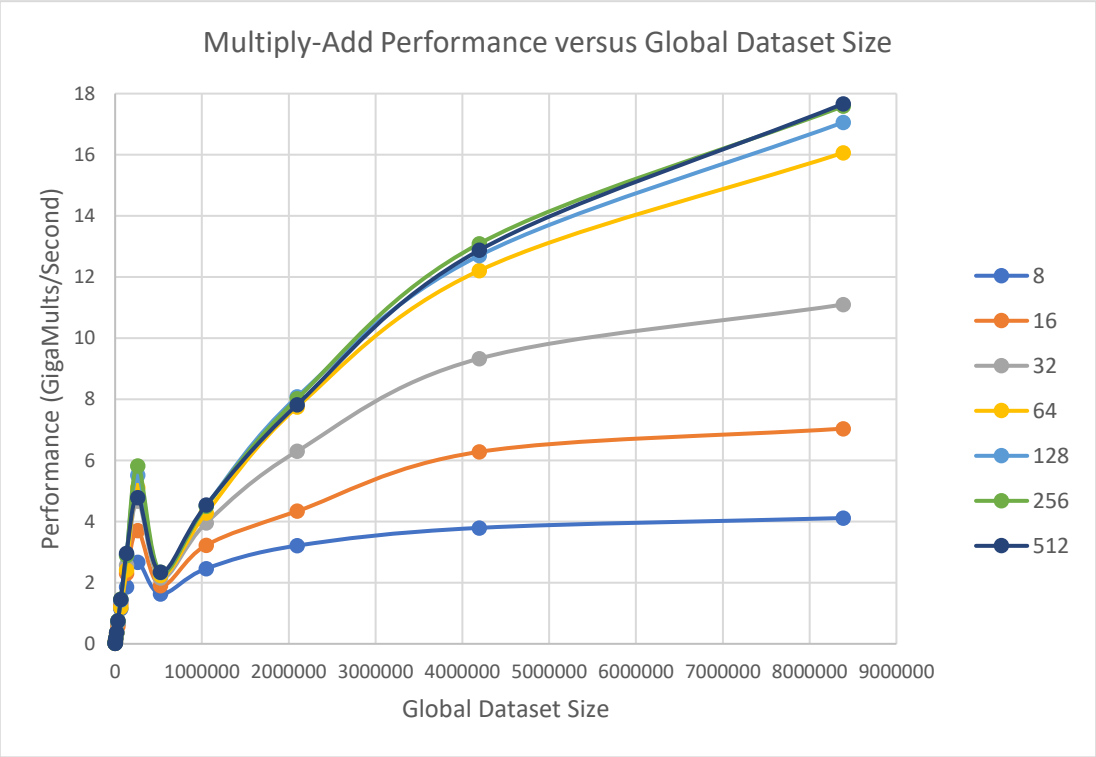
- **Graph of Multiply Performance versus Local Work Size, with a series of colored Constant-Global-Dataset-Size curves:**



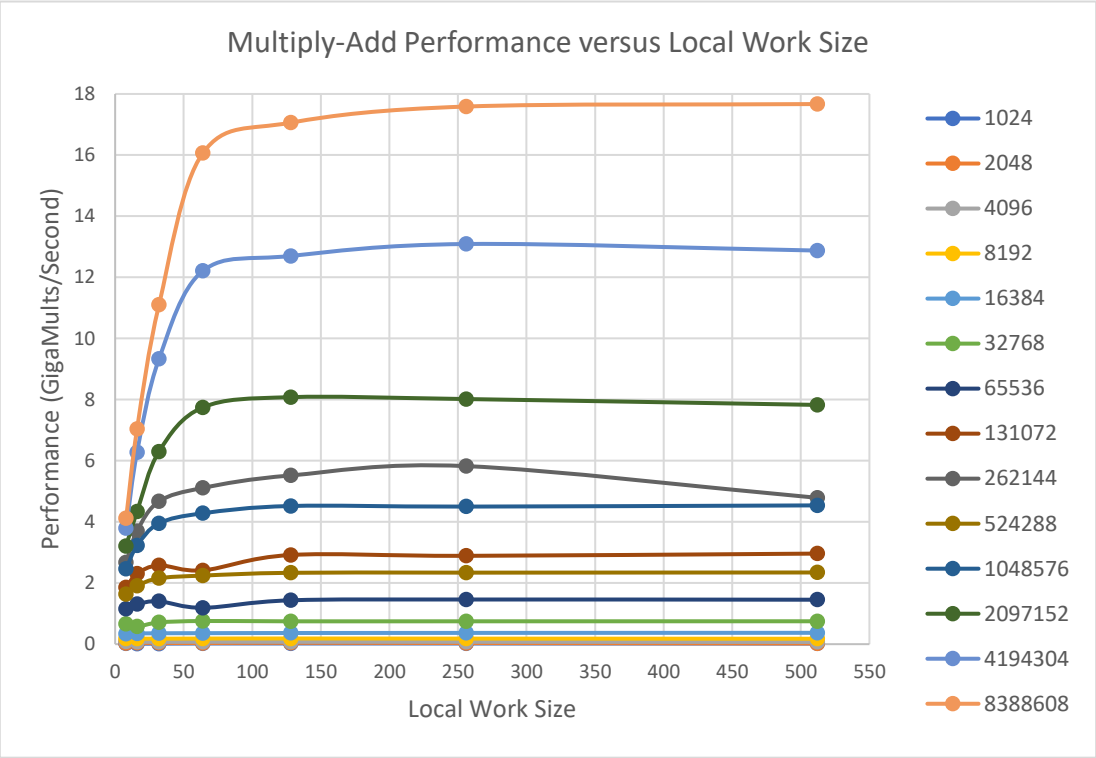
- **Table of Multiply-Add Performance versus Global Dataset Size and Local Work Size:**

Global Dataset Size \ Local Work Size	8	16	32	64	128	256	512
1024	0.023	0.016	0.017	0.022	0.023	0.022	0.02
2048	0.037	0.041	0.04	0.038	0.045	0.038	0.024
4096	0.088	0.081	0.074	0.078	0.077	0.077	0.072
8192	0.181	0.182	0.18	0.184	0.186	0.182	0.177
16384	0.34	0.35	0.356	0.36	0.365	0.368	0.371
32768	0.66	0.575	0.711	0.758	0.747	0.748	0.749
65536	1.155	1.307	1.403	1.19	1.435	1.461	1.454
131072	1.86	2.302	2.577	2.415	2.917	2.89	2.963
262144	2.674	3.704	4.669	5.112	5.519	5.825	4.788
524288	1.633	1.909	2.156	2.244	2.335	2.34	2.345
1048576	2.461	3.225	3.953	4.282	4.52	4.501	4.54
2097152	3.212	4.341	6.302	7.747	8.081	8.019	7.827
4194304	3.794	6.28	9.331	12.21	12.7	13.091	12.876
8388608	4.115	7.04	11.1	16.065	17.06	17.588	17.669

- **Graph of Multiply-Add Performance versus Global Dataset Size, with a series of colored Constant-Local-Work-Size curves:**



- **Graph of Multiply-Add Performance versus Local Work Size, with a series of colored Constant-Global-Dataset-Size curves:**



- **Patterns in the performance curves and why:**

For both multiply and multiply-add performance curves, with the same number of global dataset size, the larger the local work size is, the higher the performance is. But the performance difference among local work sizes of 128, 256 and 512 is not very big. I think the reason is larger local work size can make better use of local memory. An OpenCL program is organized as a grid of Work-Groups. Each Work-Group is organized as a grid of Work-Items. In terms of hardware, a Work-Group runs on a Compute Unit and a Work-Item runs on a Processing Element. All work items in a work group execute on the same compute unit, share resources of the compute unit. Thus, larger local work size means more work items can execute synchronously, and less time is spent on accessing global memory, which brings better performance, while less local work size leaves some of the streaming processors unutilized. But when the local work size gets too large, there may be lesser work-groups than the number of streaming processors in the GPU, then some streaming processors will run idle, which makes the performance under local work sizes of 128, 256 and 512 not increase much.

For the same number of local work size, the bigger the global dataset size is, the higher the performance is. I think the reason is that larger global dataset size means larger amounts of data parallelism. Larger global dataset size lowers memory latency and overheads, which also brings better performance.

- **The performance difference between doing a Multiply and doing a Multiply-Add:**

Under same Global Dataset Size and Local Work Size, performance of Multiply is a little bit better than Multiply-Add.

- **The meaning for the proper use of GPU parallel computing:**

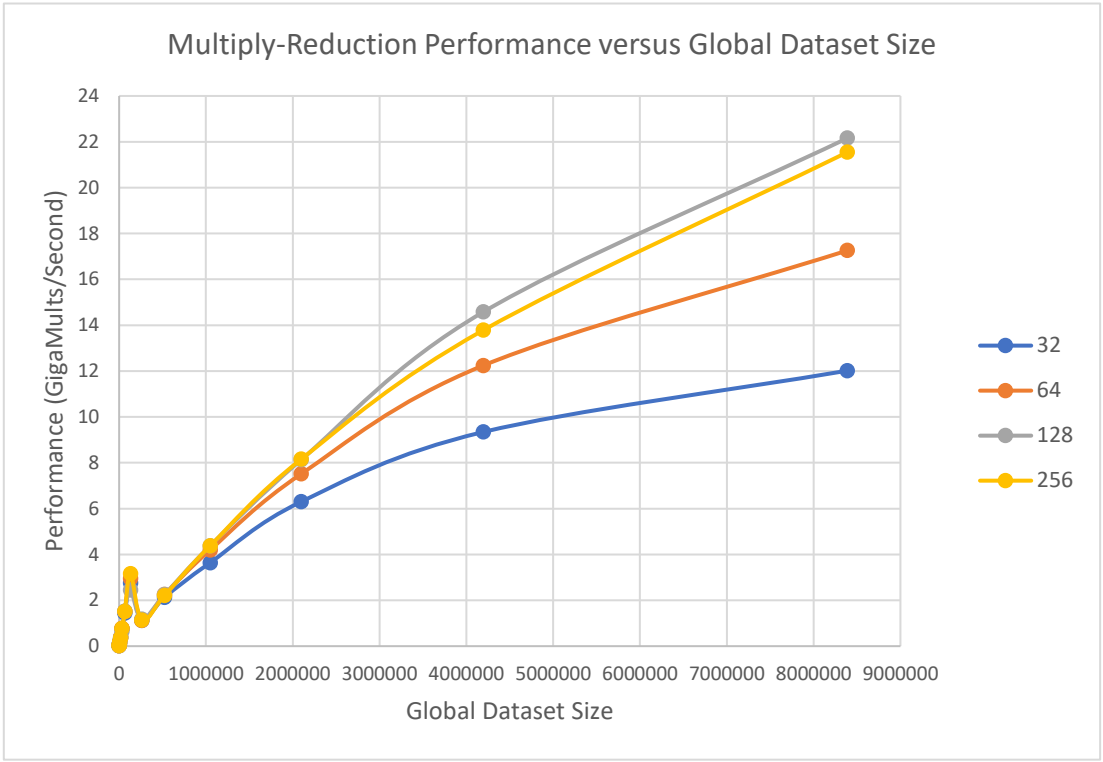
When using OpenCL to do GPU parallel computing, it's at its best on compute devices with large amounts of data parallelism. Therefore, larger global data size can bring better performance.

But in order to get best performance and improve efficiency, the local work size should not be too small or too large. In this proper scope, the larger the local work size is, the better the performance is. So, in order to make the best use of local memory, it's important to find the maximum work-group size.

- **Table of Multiply-Reduction Performance versus Global Dataset Size and Local Work Size:**

Local Work Size / Global Dataset Size	32	64	128	256
1024	0.023	0.023	0.024	0.023
2048	0.046	0.047	0.047	0.046
4096	0.093	0.093	0.092	0.094
8192	0.181	0.171	0.188	0.186
16384	0.375	0.378	0.371	0.372
32768	0.759	0.772	0.641	0.774
65536	1.437	1.505	1.527	1.514
131072	2.754	2.95	2.453	3.153
262144	1.126	1.115	1.175	1.129
524288	2.132	2.247	2.249	2.212
1048576	3.641	4.197	4.376	4.378
2097152	6.299	7.512	8.127	8.154
4194304	9.342	12.241	14.576	13.782
8388608	12.016	17.258	22.158	21.543

- **Graph of Multiply-Reduction Performance versus Global Dataset Size, with a series of colored Constant-Local-Work-Size curves:**



- **Graph of Multiply-Reduction Performance versus Local Work Size, with a series of colored Constant-Global-Dataset-Size curves:**

- **Pattern in this performance curve and why:**

For the same number of global dataset size, the larger the local work size is, the higher the performance is. But the performance difference between local work sizes of 128 and 256 is not very big. Performances of local work sizes of 128 are even better than local work sizes of 256 under some global dataset sizes. I think the reason is similar to the one I explained in the "Multiply and Multiply-Add" part. Global memory access is time-consuming, and larger local work size can make better use of local memory. Larger local work size means more work items can execute synchronously, and less time is spent on accessing global memory, which brings better performance, while less local work size leaves some of the streaming processors unutilized. But when the local work size gets too large, there may be lesser work-groups than the number of streaming processors in the GPU, then some streaming processors will run idle, which makes the performances under local work sizes of 128 and 256 don't have significant difference.

For the same number of local work size, the bigger the global dataset size is, the higher the performance is. I think the reason is that larger global dataset size means larger amounts of data parallelism. Larger global dataset size lowers memory latency and overheads, which also brings better performance.

- **The meaning for the proper use of GPU parallel computing:**

The meaning for the proper use of GPU parallel computing is also similar to the "Multiply and Multiply-Add" part.

When using OpenCL to do GPU parallel computing, larger global data size can bring better performance. But the local work size should not be too small or too large. In this proper scope, the larger the local work size is, the better the performance is. In order to make the best use of local memory, it's important to find the maximum work-group size.