Shawn Berg
bergsha@oregonstate.edu
May 21st, 2019
CS 475

Project 5
OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce
Writeup

Machine:

This project was run on a Google Cloud Platform Compute Engine instance. The details of this machine are below:

```
bergsm@more-threads:~/CS475/Project2$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                24
On-line CPU(s) list:   0-23
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) CPU @ 2.00GHz
Stepping:              3
CPU MHz:               2000.172
BogoMIPS:              4000.34
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              56320K
NUMA node0 CPU(s):     0-23
```

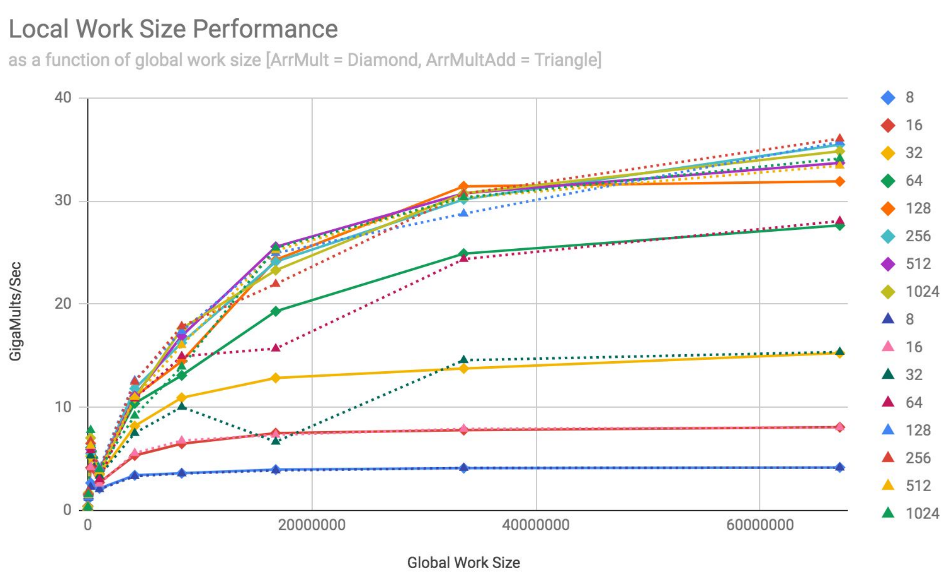
GPU information is below:

```
bergsm@more-threads-2:~/CS475/Project5$ ../OpenCL/utils/printinfo
Number of Platforms = 1
Platform #0:
        Name    = 'NVIDIA CUDA'
        Vendor  = 'NVIDIA Corporation'
        Version = 'OpenCL 1.2 CUDA 9.1.84'
        Profile = 'FULL_PROFILE'
        Number of Devices = 4
        Device #0:
                Type = 0x0004 = CL_DEVICE_TYPE_GPU
                Device Vendor ID = 0x10de (NVIDIA)
                Device Maximum Compute Units = 56
                Device Maximum Work Item Dimensions = 3
                Device Maximum Work Item Sizes = 1024 x 1024 x 64
                Device Maximum Work Group Size = 1024
                Device Maximum Clock Frequency = 1328 MHz
```

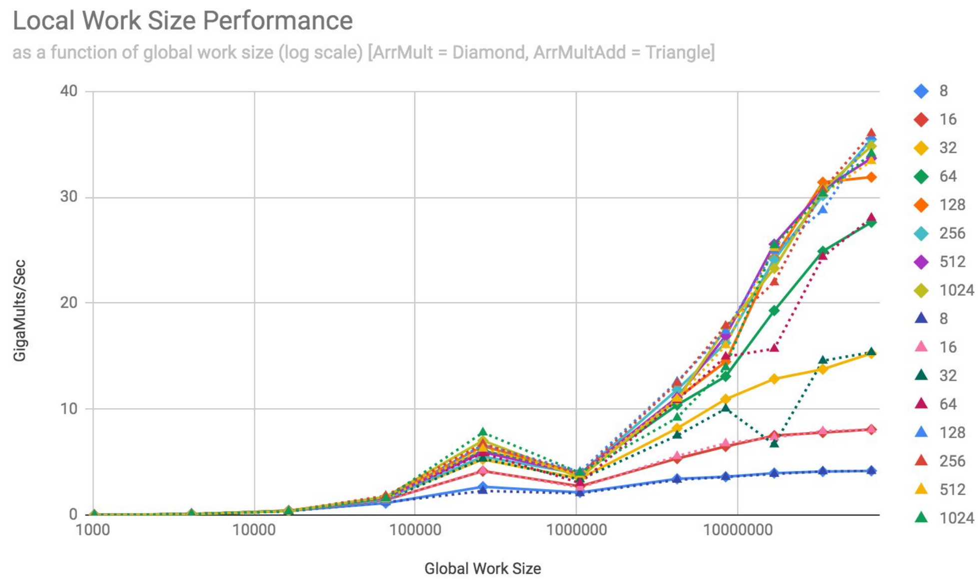
Table 1 (Performance as a function of Local Work Size vs Global Work size)

Mult	1024	4096	16384	65536	262144	1048576	4194304	8388608	16777216	33554432	67108864
8	0.026	0.101	0.43	1.143	2.683	2.153	3.428	3.627	3.975	4.119	4.186
16	0.023	0.103	0.402	1.439	4.16	2.732	5.345	6.487	7.51	7.804	8.089
32	0.024	0.102	0.407	1.6	5.27	3.467	8.194	10.941	12.854	13.774	15.266
64	0.03	0.105	0.408	1.614	6.097	3.962	10.393	13.099	19.305	24.896	27.634
128	0.027	0.081	0.436	1.575	6.516	3.943	11.046	14.478	24.29	31.413	31.896
256	0.025	0.097	0.404	1.538	5.539	3.898	11.83	16.254	24.117	30.148	35.454
512	0.027	0.109	0.413	1.636	5.944	3.888	11.109	16.92	25.564	30.737	33.679
1024	0.026	0.103	0.4	1.733	7.041	3.569	10.905	17.632	23.283	30.738	34.813
Mult + Add											
8	0.025	0.102	0.369	1.219	2.286	2.067	3.336	3.603	3.894	4.131	4.173
16	0.025	0.119	0.414	1.399	4.207	2.613	5.578	6.783	7.344	7.926	8.068
32	0.015	0.117	0.395	1.429	5.328	3.367	7.511	10.046	6.671	14.572	15.37
64	0.026	0.103	0.396	1.621	5.851	3.04	10.86	14.957	15.7	24.363	28.054
128	0.025	0.104	0.424	1.281	6.655	4.101	12.606	17.392	24.949	28.765	35.717
256	0.026	0.104	0.397	1.831	6.731	3.853	12.479	17.863	21.95	30.696	36.029
512	0.026	0.099	0.429	1.515	6.28	3.807	10.994	16.032	25.206	30.297	33.413
1024	0.026	0.103	0.383	1.589	7.799	4.001	9.195	13.934	25.471	30.333	34.11

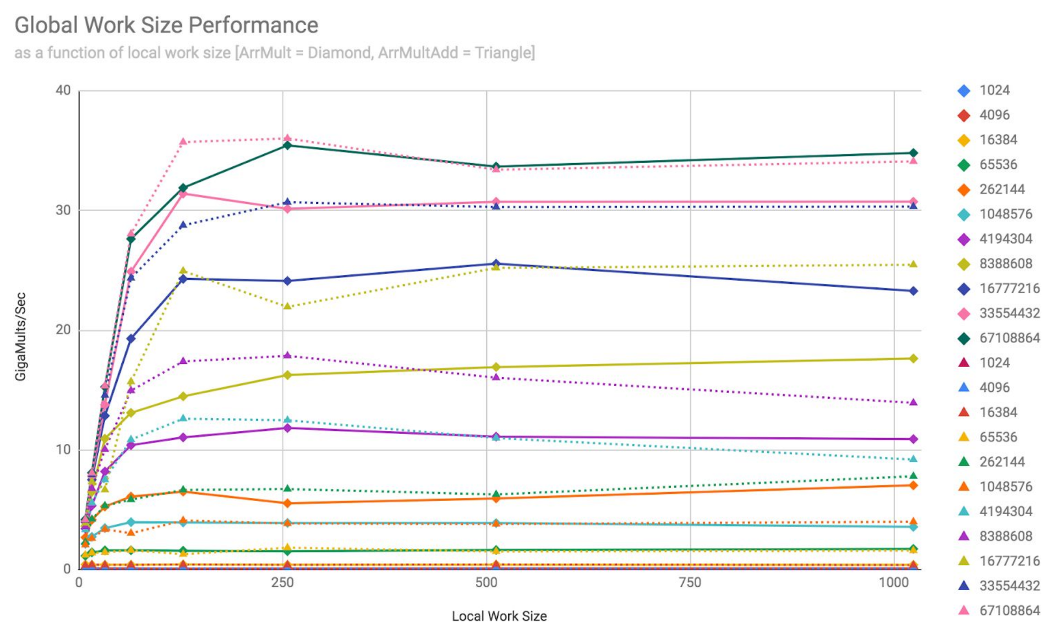
Graph 1 (Local Work Size Performance as a function of global work size)



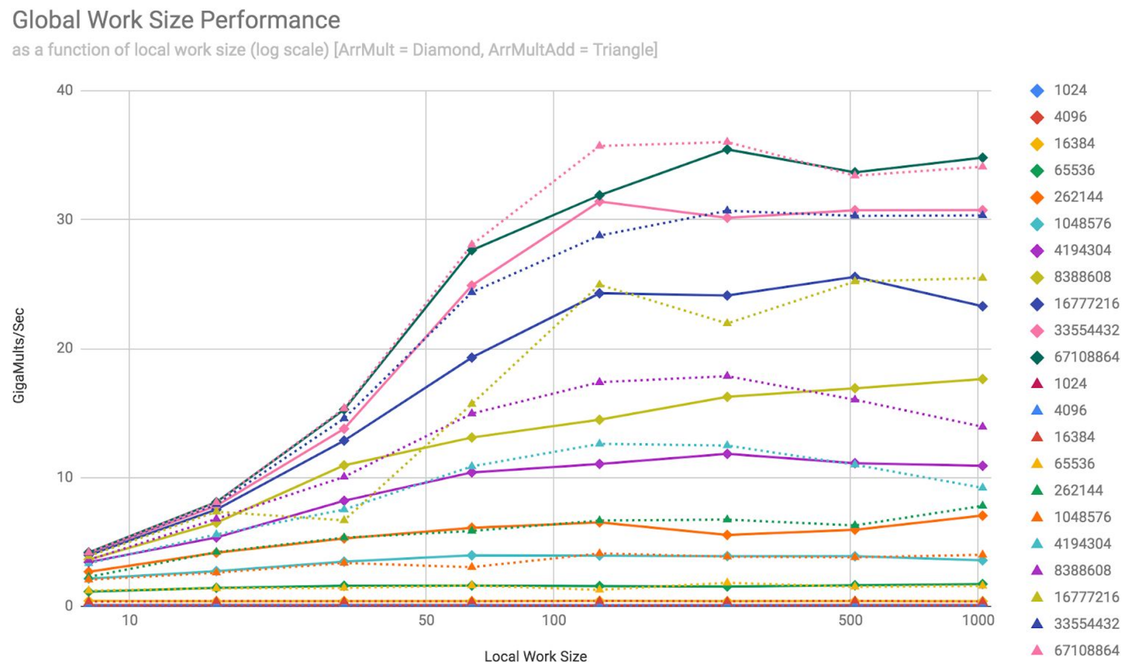
Graph 2 (Local Work Size Performance as a function of global work size log scale)



Graph 3 (Global Work Size Performance as a function of local work size)



Graph 4 (Global Work Size Performance as a function of local work size log scale)



Commentary

1. What patterns are you seeing in the performance curves?
    a. **For the Local Work Size as a function of Global Work Size graphs, after some initial noise, you see a steady performance increase with an increase of global work size. It looks as though the trend would continue if allowed, so it would be interesting to continue with larger global work sizes. Unfortunately, 67108864 was the limit for global work size for this GPU.**
    b. **For the Group Work Size as a function of Local Work Size graphs, you see a steady increase in performance and then a leveling off at a local work size of ~128 or 256. There also tends to be better performance for larger global work sizes.**

2. Why do you think the patterns look this way? **The general performance increase is most likely due to the GPU being utilized more as the workload increases. This effect has diminishing returns as seen in the local work group performance. For this particular GPU, after the local group size reaches ~128 or 256, the performance levels out. This is most likely due to the number of processing elements per compute unit on the GPU and the optimal work size to take advantage of all of the processing elements with minimal queuing.**

3. What is the performance difference between doing a Multiply and doing a Multiply-Add? **In almost all cases, the performance difference between Multiply and Multiply-Add operations is minimal, with performance of multiply-add operations actually being higher in some cases. This is most likely due to the fused multiply add assembly instruction. This instruction takes advantage of extra performance increase from converting multiply and add operations into a single hardware instruction.**

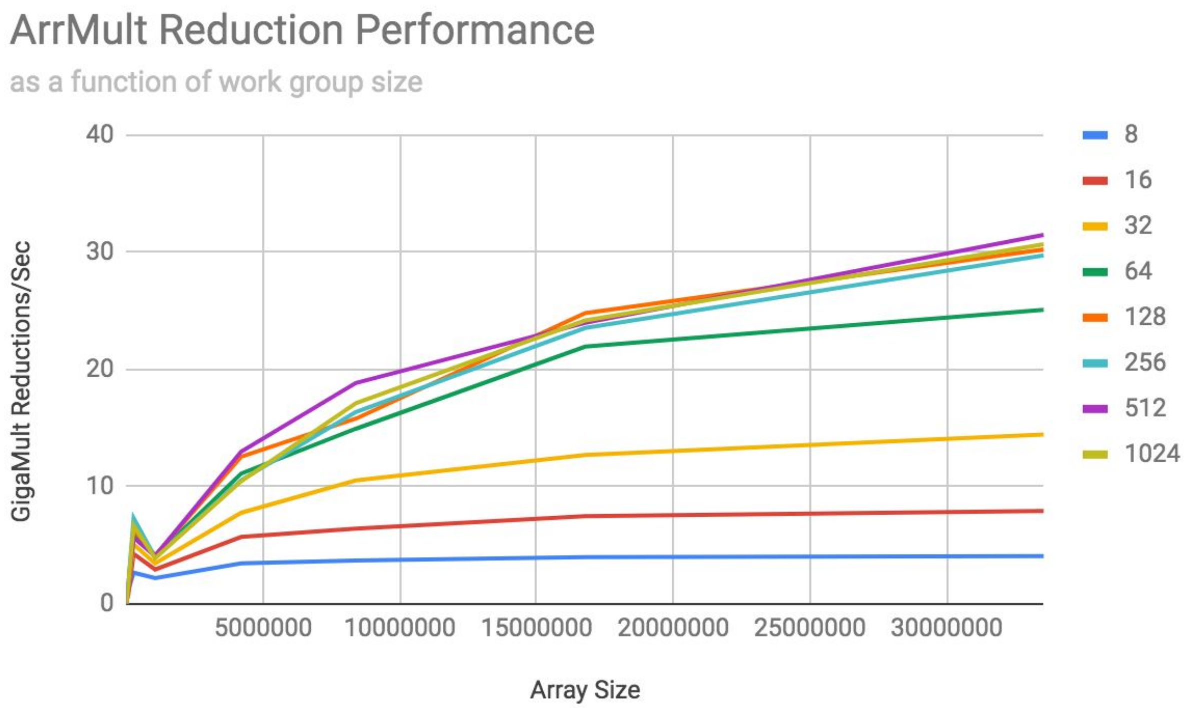
4. What does that mean for the proper use of GPU parallel computing? **It shows that the proper use of GPU parallel computing is for very data intensive applications where you need to perform a large amount of calculations in parallel. It also means that to effectively utilize GPU parallel computing, one needs to have a large enough global data size as well as the ability to divide the global data into large enough local work group sizes to take full advantage of the available processing power.**

Additional Array Multiply Reduction Problem

Table 1 (Performance as a function of Local Work Size vs Global Work size)

	1024	4096	16384	65536	262144	1048576	4194304	8388608	16777216	33554432
8	0.024	0.102	0.379	1.143	2.673	2.2	3.464	3.705	3.998	4.076
16	0.025	0.115	0.411	0.744	4.281	2.937	5.719	6.423	7.495	7.925
32	0.028	0.078	0.344	1.534	5.071	3.466	7.781	10.536	12.71	14.448
64	0.025	0.102	0.34	1.617	5.914	4.039	11.126	14.945	21.954	25.104
128	0.027	0.116	0.416	1.585	6.371	4.142	12.558	15.809	24.815	30.243
256	0.026	0.105	0.373	1.621	7.318	3.974	10.555	16.381	23.546	29.742
512	0.026	0.106	0.397	1.601	5.65	4.102	12.998	18.849	23.98	31.488
1024	0.026	0.086	0.42	1.565	6.684	3.913	10.476	17.122	24.187	30.697

Graph 1 (Performance as a function of work group size)



Commentary

1. What pattern are you seeing in this performance curve? **In this performance curve you see a general increase in performance as the array size increases with the larger work group sizes being clustered toward the top of the graph.**
2. Why do you think the pattern looks this way? **The pattern looks this way because as array size increases, the GPU is able to be utilized more. Also, as the work group size increases each compute unit is being utilized more effectively. This explains the cluster of larger local work group plots near the top of the graph.**
3. What does that mean for the proper use of GPU parallel computing? **It means that in order to properly utilize the power of GPU parallel computing you should have a large enough global group size and have the ability to divide that global size into large enough local group sizes to take full advantage of the available processing power.**