CS 475/575 -- Spring 2022

Project 6

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

Vishwas Somashekara Reddy

ID- 934402783

reddyv@oregonstate.edu

1. What machine you ran this on

   I ran it on the rabbit server, which contains a Nvidia driver for GPU with the specs as follows.

   Name = 'NVIDIA CUDA'

   Vendor = 'NVIDIA Corporation'

   Version = 'OpenCL 3.0 CUDA 11.4.158'

   Profile = 'FULL_PROFILE'

   Number of Devices = 1

   Device #0:

   Type = 0x0004 = CL_DEVICE_TYPE_GPU

   Device Vendor ID = 0x10de (NVIDIA)

   Device Maximum Compute Units = 15

   Device Maximum Work Item Dimensions = 3

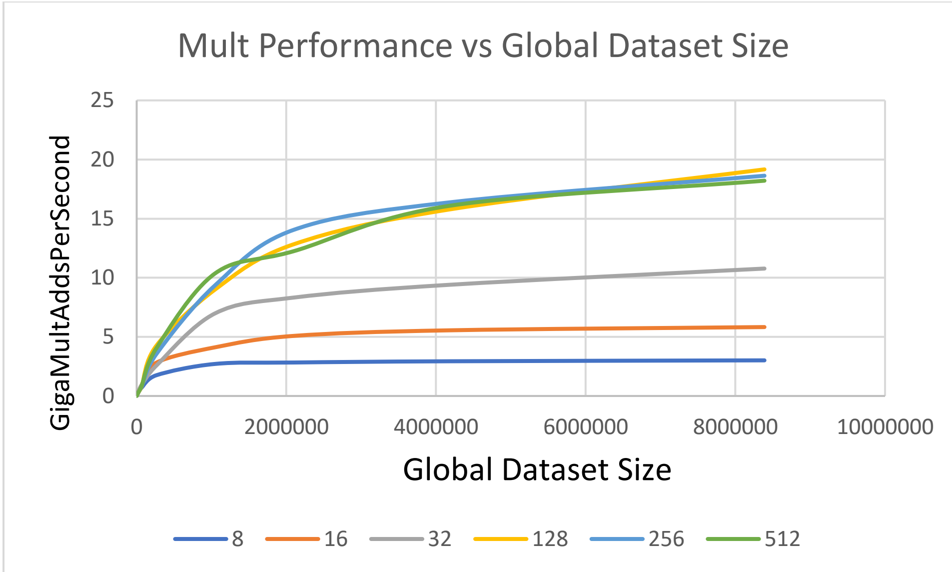
   Device Maximum Work Item Sizes = 1024 x 1024 x 64

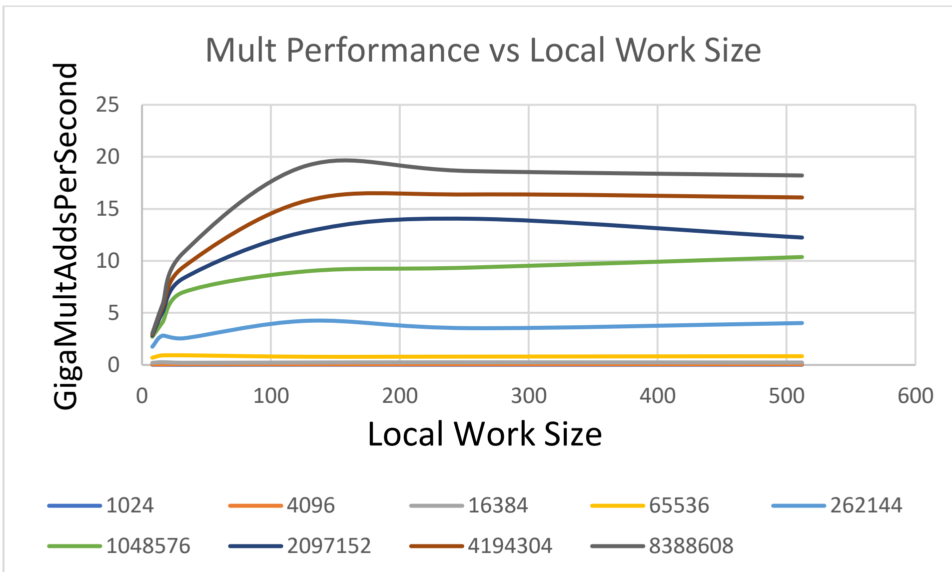
   Device Maximum Work Group Size = 1024

   Device Maximum Clock Frequency = 1071 MHz

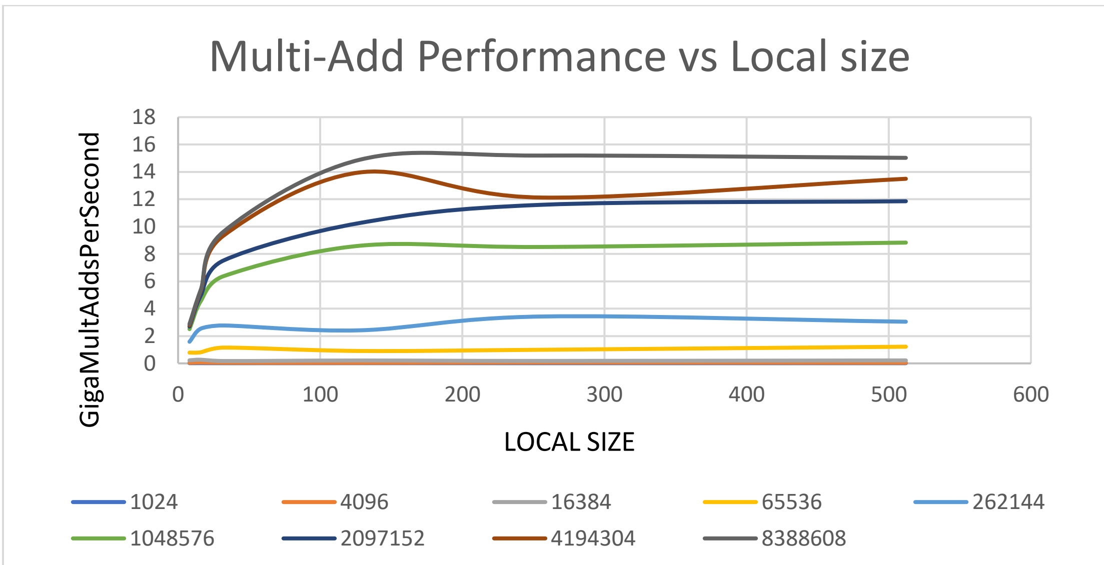
2. Show the tables and graphs

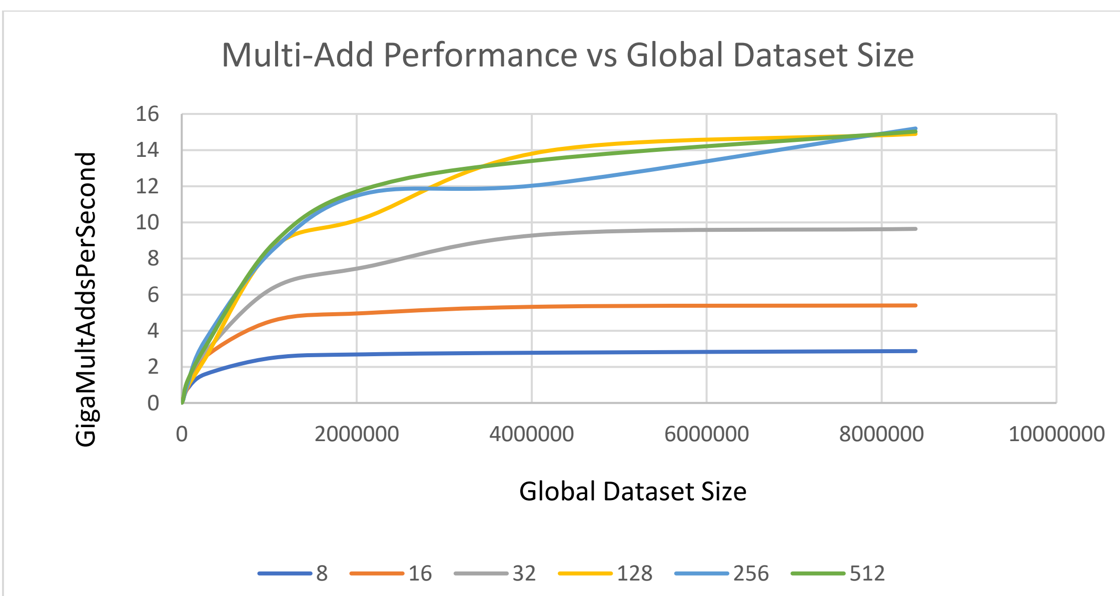
Dataset Size \ Local Size	8	16	32	128	256	512
1024	0.015	0.019	0.015	0.021	0.017	0.018
4096	0.051	0.051	0.069	0.06	0.065	0.063
16384	0.202	0.266	0.211	0.236	0.254	0.235
65536	0.697	0.92	0.928	0.789	0.801	0.841
262144	1.752	2.815	2.563	4.242	3.535	4.027
1048576	2.711	4.118	6.99	9.007	9.364	10.37
2097152	2.83	5.081	8.322	12.828	14.05	12.245
4194304	2.935	5.555	9.41	15.793	16.386	16.1
8388608	3.012	5.828	10.777	19.164	18.629	18.209


Mult Performance vs Global Dataset Size
GigaMultAddsPerSecond
25
20
15
10
5
0
0
2000000
4000000
6000000
8000000
10000000
Global Dataset Size
8
16
32
128
256
512


Mult Performance vs Local Work Size
GigaMultAddsPerSecond
25
20
15
10
5
0
0
100
200
300
400
500
600
Local Work Size
1024
4096
16384
65536
262144
1048576
2097152
4194304
8388608

Dataset Size\Local size	8	16	32	128	256	512
1024	0.015	0.018	0.014	0.016	0.014	0.013
4096	0.046	0.067	0.049	0.043	0.06	0.048
16384	0.224	0.266	0.171	0.21	0.18	0.216
65536	0.785	0.814	1.153	0.907	0.995	1.218
262144	1.582	2.541	2.77	2.418	3.427	3.046
1048576	2.504	4.566	6.378	8.601	8.511	8.83
2097152	2.691	4.975	7.523	10.267	11.595	11.853
4194304	2.784	5.335	9.343	13.963	12.123	13.495
8388608	2.871	5.397	9.636	14.893	15.194	15.025





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

   The performance lines in the preceding performance versus Data Set graphs climb till 128 and then remain static on top of them.

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

   As the size of the data set increases, the number of GPU memory and threads waiting to process it will also increase, leading to faster computation as the situation becomes more complex. Now, if the local block size is smaller, it means there is more data to process and a lot of waiting to complete each block Thus, the graph appears in this way - each like is stacked on top of each other.

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

   During the multiplying arrays, the process of multiplying and adding numbers can take a bit longer than usual because there is no wait after the multiplication.

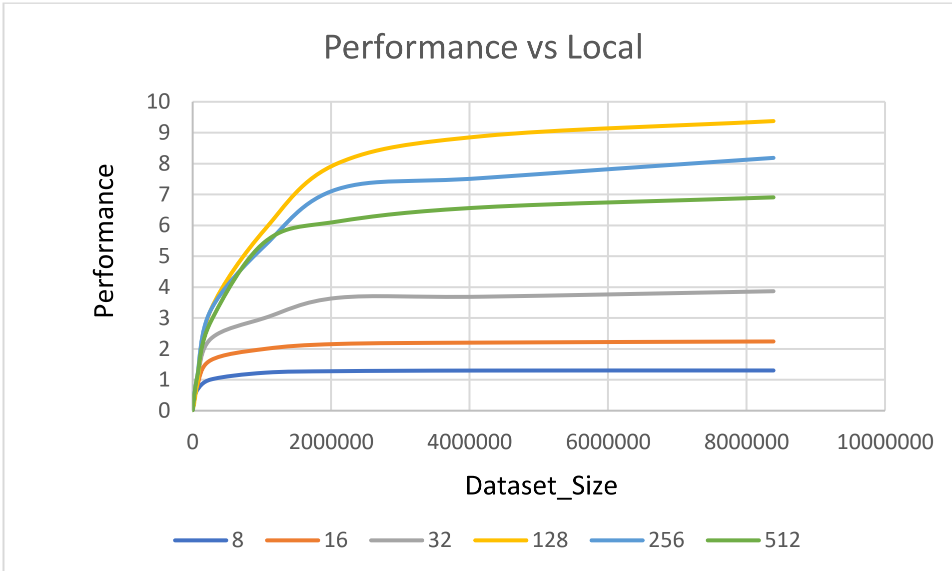
6. What does that mean for the proper use of GPU parallel computing?

   Based on the graphs, the 128-block size option is the best option. Even if we set the block size to a value greater than 128, the network still operates at the same speed. However, if we set the block size to a lower value, performance decreases.

Part 2:

1. Show this table and graph

Dataset Size\Local size	8	16	32	128	256	512
1024	0.021	0.019	0.021	0.02	0.02	0.016
4096	0.081	0.061	0.074	0.076	0.079	0.094
16384	0.19	0.293	0.205	0.16	0.388	0.331
65536	0.665	0.807	0.985	0.856	1.109	1.095
262144	1.009	1.635	2.327	3.261	3.257	2.883
1048576	1.23	2.001	3.007	5.882	5.357	5.474
2097152	1.279	2.16	3.655	8.009	7.175	6.125
4194304	1.299	2.204	3.69	8.889	7.533	6.584
8388608	1.302	2.241	3.867	9.375	8.184	6.905



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

   The maximum performance that can be achieved when using a block size of 128 is significantly higher than when using a block size of any other size. The 256-block size was marginally worse in performance than the 512-block size, but the latter had comparable performance to the 64-block size. Therefore, if the workload is so large that the threads are affected by the lack of resources, it will damage the performance.

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

   Small arrays perform poorly, up to the size of 4M, array performance rises swiftly, but it develops considerably more slowly beyond that. The expensive cost of graphics processing units (GPUs) and the quantity of data they require to be transferred across threads are the primary reasons for their lack of performance. Larger array sizes can benefit from this more, resulting in improved performance.

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

   It's advisable to choose a block size which was neither too big nor too little for GPU computation to work well. For this system, 128 appears to be the best option. Also, to make the most out it, choose a local work size that is a multiple of 32, but not less than 32.