CS 575

Project #6

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

Submitted By: Aman Pandita

ONID: panditaa@oregonstate.edu

Part-1 Multiply and Multiply Add

1. **What machine you ran this on**

   Rabbit(rabbit.engr.oregonstate.edu)

2. **Show the tables and graphs**

Multiply two arrays together using OpenCL

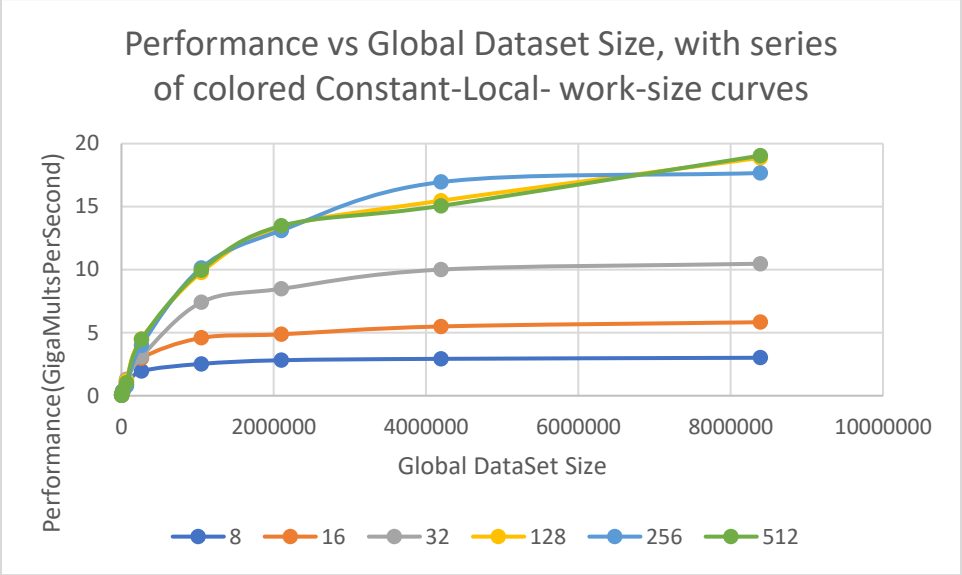
NMB	LOCAL_SIZE	GigaMultsPerSecond	NUM_WORK_GROUPS
1024	8	0.019	128
1024	16	0.013	64
1024	32	0.021	32
1024	128	0.021	8
1024	256	0.014	4
1024	512	0.018	2
4096	8	0.071	512
4096	16	0.083	256
4096	32	0.086	128
4096	128	0.074	32
4096	256	0.066	16
4096	512	0.082	8
16384	8	0.247	2048
16384	16	0.358	1024
16384	32	0.333	512
16384	128	0.312	128
16384	256	0.356	64
16384	512	0.328	32
65536	8	0.858	8192
65536	16	1.065	4096
65536	32	1.296	2048
65536	128	1.161	512
65536	256	0.756	256
65536	512	1.024	128
262144	8	1.947	32768
262144	16	2.972	16384
262144	32	3.101	8192
262144	128	4.504	2048
262144	256	3.99	1024
262144	512	4.46	512
1048576	8	2.523	131072
1048576	16	4.585	65536
1048576	32	7.411	32768
1048576	128	9.785	8192
1048576	256	10.137	4096
1048576	512	9.952	2048

2097152	8	2.814	262144
2097152	16	4.869	131072
2097152	32	8.48	65536
2097152	128	13.365	16384
2097152	256	13.107	8192
2097152	512	13.477	4096
4194304	8	2.925	524288
4194304	16	5.495	262144
4194304	32	10.005	131072
4194304	128	15.467	32768
4194304	256	16.946	16384
4194304	512	15.057	8192
8388608	8	3.011	1048576
8388608	16	5.831	524288
8388608	32	10.47	262144
8388608	128	18.87	65536
8388608	256	17.666	32768
8388608	512	19.047	16384

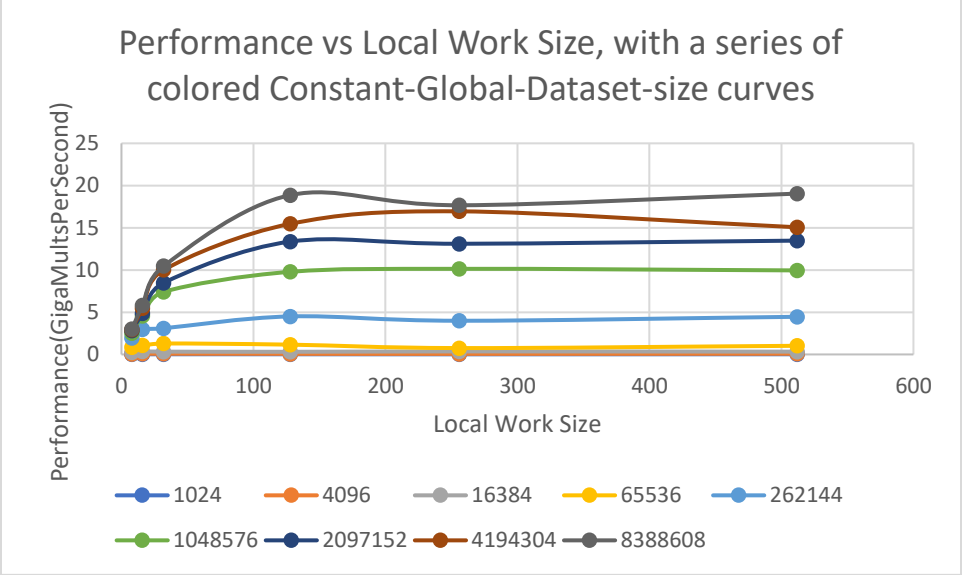
Pivot Table:

	8	16	32	128	256	512
1024	0.019	0.013	0.021	0.021	0.014	0.018
4096	0.071	0.083	0.086	0.074	0.066	0.082
16384	0.247	0.358	0.333	0.312	0.356	0.328
65536	0.858	1.065	1.296	1.161	0.756	1.024
262144	1.947	2.972	3.101	4.504	3.99	4.46
1048576	2.523	4.585	7.411	9.785	10.137	9.952
2097152	2.814	4.869	8.48	13.365	13.107	13.477
4194304	2.925	5.495	10.005	15.467	16.946	15.057
8388608	3.011	5.831	10.47	18.87	17.666	19.047

Performance vs Global Dataset Size, with series of colored Constant-Local- work-size curves



Performance vs Local Work Size, with a series of colored Constant-Global-Dataset-size curves



Multiply two arrays together and add a third using OpenCL

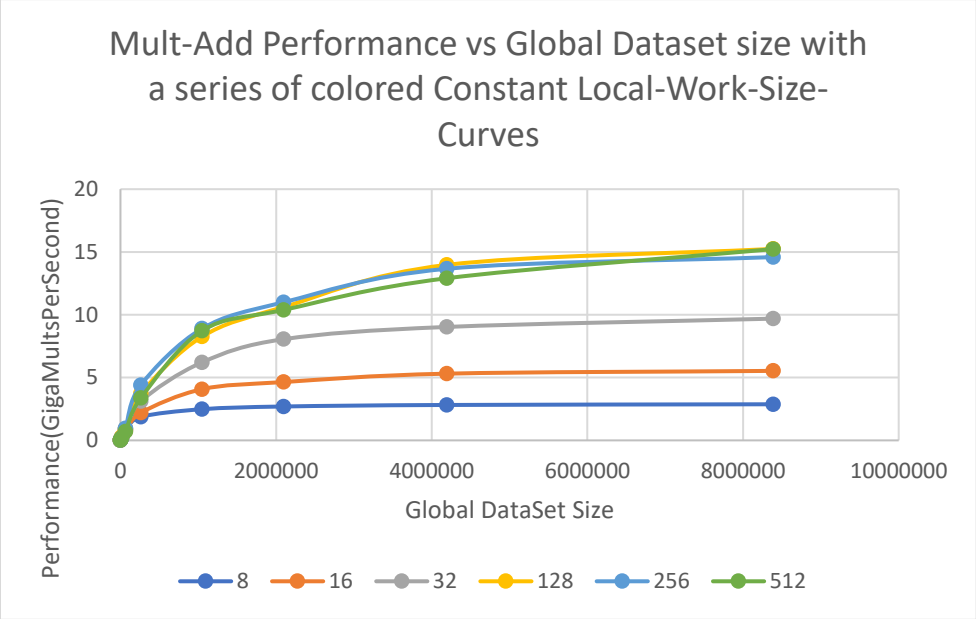
NMB	LOCAL_SIZE	GigaMultsPerSecond	NUM_WORK_GROUPS
1024	8	0.015	128
1024	16	0.013	64
1024	32	0.016	32
1024	128	0.015	8
1024	256	0.013	4
1024	512	0.019	2
4096	8	0.051	512
4096	16	0.054	256
4096	32	0.071	128
4096	128	0.059	32
4096	256	0.063	16
4096	512	0.057	8
16384	8	0.207	2048
16384	16	0.211	1024
16384	32	0.258	512
16384	128	0.278	128
16384	256	0.203	64
16384	512	0.199	32
65536	8	0.9	8192
65536	16	0.744	4096
65536	32	0.863	2048
65536	128	0.901	512
65536	256	0.969	256
65536	512	0.707	128
262144	8	1.878	32768
262144	16	2.202	16384
262144	32	3.125	8192
262144	128	3.73	2048
262144	256	4.407	1024
262144	512	3.38	512
1048576	8	2.481	131072
1048576	16	4.076	65536
1048576	32	6.215	32768
1048576	128	8.262	8192
1048576	256	8.898	4096
1048576	512	8.729	2048
2097152	8	2.698	262144
2097152	16	4.644	131072
2097152	32	8.057	65536
2097152	128	10.616	16384
2097152	256	11	8192
2097152	512	10.388	4096
4194304	8	2.82	524288
4194304	16	5.313	262144
4194304	32	9.032	131072

4194304	128	13.972	32768
4194304	256	13.662	16384
4194304	512	12.91	8192
8388608	8	2.874	1048576
8388608	16	5.532	524288
8388608	32	9.692	262144
8388608	128	15.245	65536
8388608	256	14.592	32768
8388608	512	15.228	16384

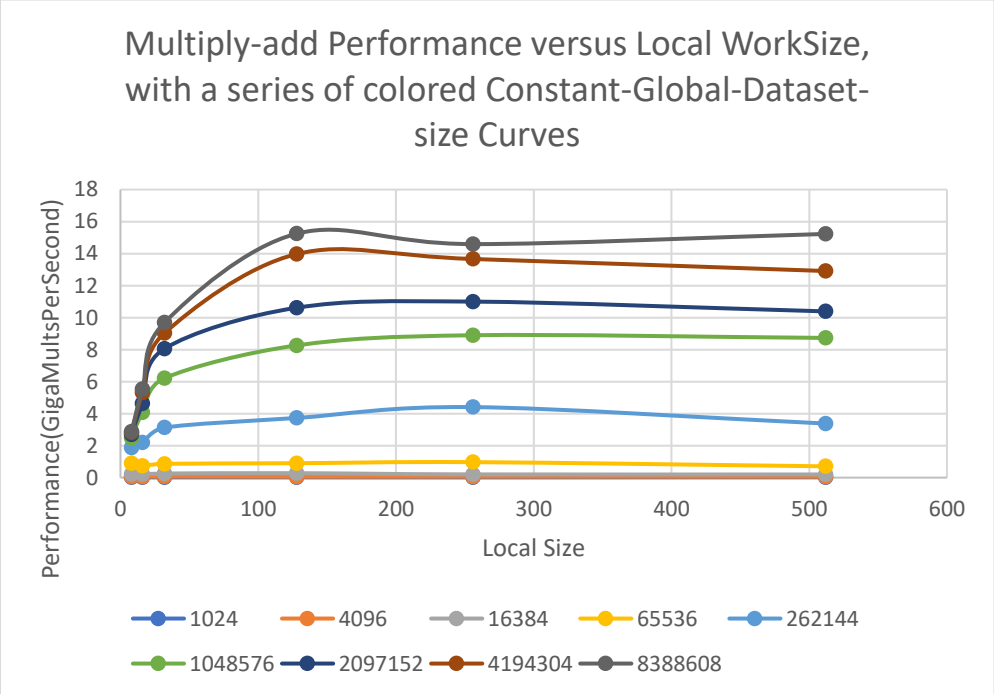
Pivot Table:

	8	16	32	128	256	512
1024	0.015	0.013	0.016	0.015	0.013	0.019
4096	0.051	0.054	0.071	0.059	0.063	0.057
16384	0.207	0.211	0.258	0.278	0.203	0.199
65536	0.9	0.744	0.863	0.901	0.969	0.707
262144	1.878	2.202	3.125	3.73	4.407	3.38
1048576	2.481	4.076	6.215	8.262	8.898	8.729
2097152	2.698	4.644	8.057	10.616	11	10.388
4194304	2.82	5.313	9.032	13.972	13.662	12.91
8388608	2.874	5.532	9.692	15.245	14.592	15.228

Mult-Add Performance vs Global Dataset size with a series of colored Constant Local-Work-Size-Curves



Multiply-add Performance versus Local WorkSize, with a series of colored Constant-Global-Dataset-size Curves



3. **What patterns are you seeing in the performance curves?**
   We can see that the performance increases when the Global Work size increases, and it gets maxed out when the local work size is around 512. Similarly, in case of the Local Work Size, the performance increases as per the increasing Global Work Size.

4. **Why do you think the patterns look this way?**
   A lot of computing time is wasted due to the idle processing units, this is because of the smaller local work size and the same thing is happening with the GPU as well, the smaller Global work Group isn't able to do enough Work Done.

5. **What is the performance difference between doing a Multiply and doing a Multiply-Add?**
   We can see that the performance of Multiply is better than the performance of Multiply Add as more resources and processing time are required for the more complicated multiply add kernel when compared to the Multiply kernel. Same patterns are observed among both Global and Local work Size.

6. **What does that mean for the proper use of GPU parallel computing?**
   GPU parallel computing can be beneficial when the data size is big, for small data sizes, it won't be that beneficial as it could not deal with the overhead of setting up. The local work sizes of 8,16,32 are very small data sizes and as a result not great but whereas the work size of 128 seems to be the perfect size.

Part-2 Multiply Reduction

1. **Show this table and graph**

Perform the same array multiply as in #1, but this time with a reduction

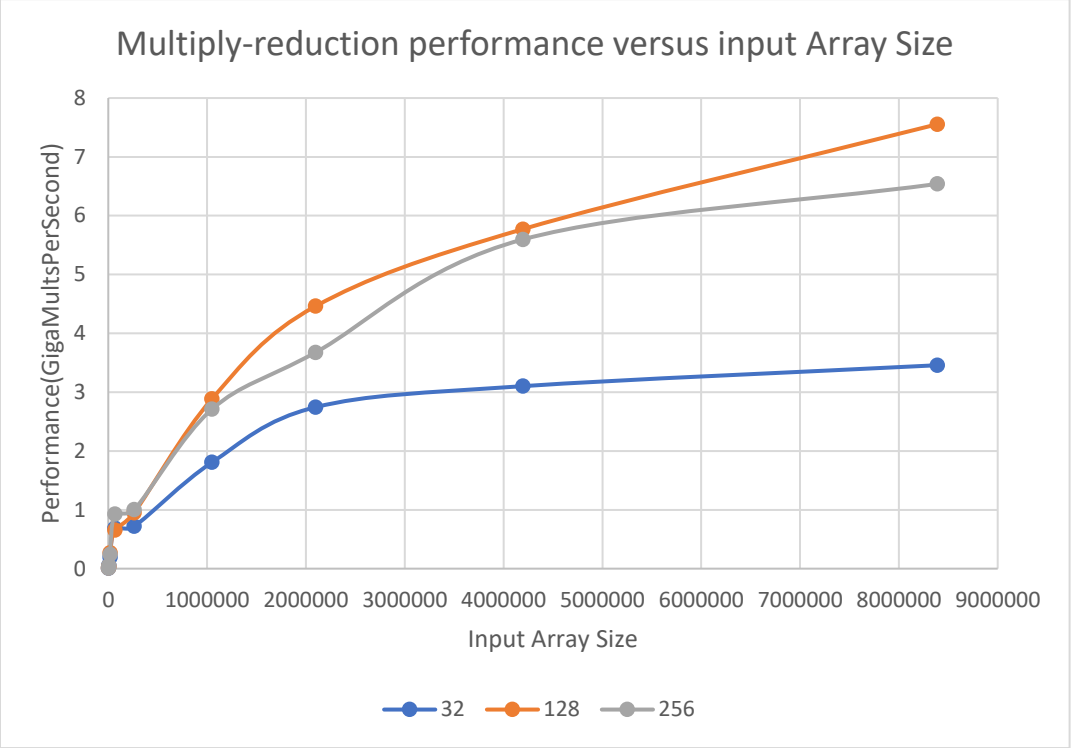
NMB	LOCAL_SIZE	GigaMultsPerSecond	NUM_WORK_GROUPS
1024	8	0.015	128
1024	16	0.012	64
1024	32	0.01	32
1024	128	0.013	8
1024	256	0.009	4
1024	512	0.011	2
4096	8	0.052	512
4096	16	0.064	256
4096	32	0.054	128
4096	128	0.041	32
4096	256	0.05	16
4096	512	0.046	8
16384	8	0.223	2048
16384	16	0.137	1024
16384	32	0.195	512
16384	128	0.268	128
16384	256	0.241	64
16384	512	0.218	32
65536	8	0.511	8192
65536	16	0.726	4096
65536	32	0.689	2048
65536	128	0.654	512
65536	256	0.926	256
65536	512	0.781	128
262144	8	0.545	32768
262144	16	0.751	16384
262144	32	0.72	8192
262144	128	0.951	2048
262144	256	1.002	1024
262144	512	0.762	512
1048576	8	1.001	131072
1048576	16	1.357	65536
1048576	32	1.806	32768
1048576	128	2.885	8192
1048576	256	2.71	4096
1048576	512	2.604	2048
2097152	8	1.106	262144
2097152	16	1.848	131072
2097152	32	2.745	65536
2097152	128	4.461	16384

2097152	256	3.675	8192
2097152	512	3.944	4096
4194304	8	1.2	524288
4194304	16	1.967	262144
4194304	32	3.102	131072
4194304	128	5.771	32768
4194304	256	5.596	16384
4194304	512	4.645	8192
8388608	8	1.26	1048576
8388608	16	2.09	524288
8388608	32	3.458	262144
8388608	128	7.554	65536
8388608	256	6.54	32768
8388608	512	5.517	16384

Pivot Table:

	32	**128**	**256**
1024	0.01	0.013	0.009
4096	0.054	0.041	0.05
16384	0.195	0.268	0.241
65536	0.689	0.654	0.926
262144	0.72	0.951	1.002
1048576	1.806	2.885	2.71
2097152	2.745	4.461	3.675
4194304	3.102	5.771	5.596
8388608	3.458	7.554	6.54

Multiply-reduction performance versus input Array Size



2. **What pattern are you seeing in this performance curve?**
   We can see that the increasing array size also triggers the increase in performance, and the max performance is achieved at 9000000 array sizes.
3. **Why do you think the pattern looks this way?**
   This is because when the array size is small, there is not much load on the GPU, so similar performance is observed in the initial stage whereas when the Array size increases, we can see that the performance also increases in accordance with the Local Work Size
4. **What does that mean for the proper use of GPU parallel computing?**
   GPU parallel computing can be beneficial when the data size is big, for small data sizes, it won't be that beneficial as it could not deal with the overhead of setting up.