Xiaoru Chen
chenxia2@oregonstate.edu
Project #6
CS 475/575
2021-05-30

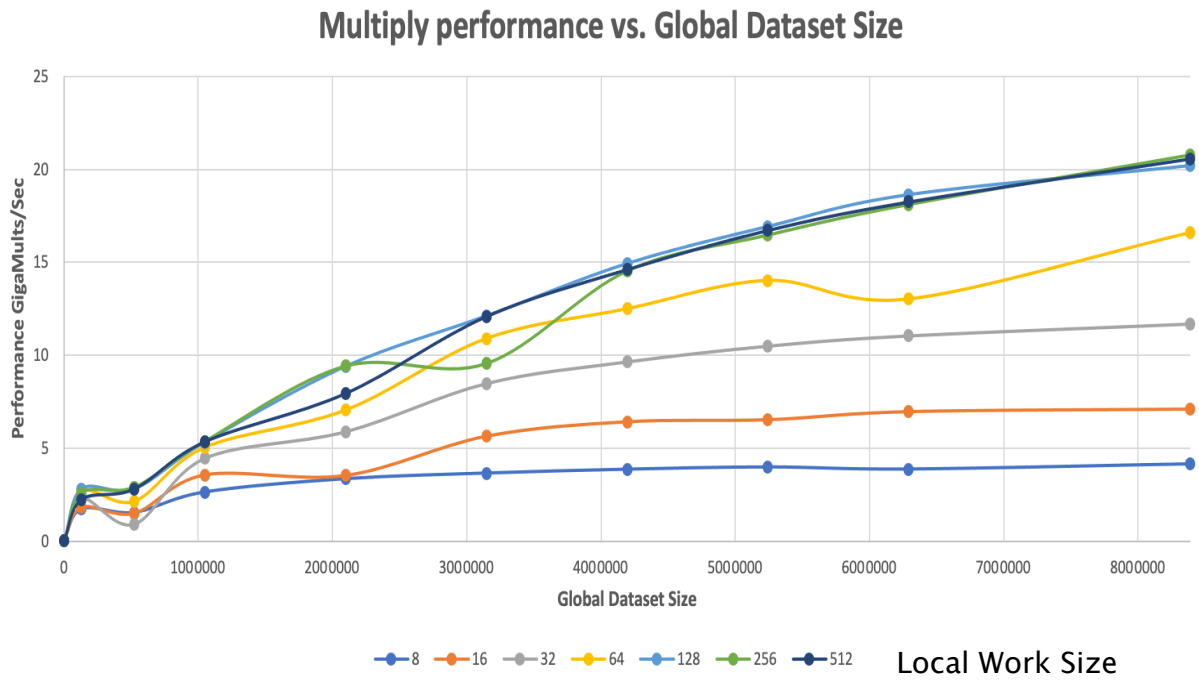
Machine I ran this on:
I used my MacBook pro to ssh to OSU ENGR server (DGX).

Table for Multiply Performance:

	8	16	32	64	128	256	512
1024	0.015	0.014	0.014	0.013	0.014	0.021	0.018
131072	1.733	1.813	2.299	2.686	2.807	2.59	2.242
524288	1.536	1.49	0.893	2.137	2.809	2.9	2.797
1048576	2.635	3.555	4.446	5.041	5.319	5.362	5.339
2097152	3.349	3.523	5.881	7.047	9.399	9.418	7.941
3145728	3.652	5.64	8.48	10.9	12.115	9.552	12.084
4194304	3.86	6.418	9.648	12.505	14.935	14.537	14.616
5242880	3.98	6.528	10.498	14.03	16.928	16.47	16.724
6291456	3.87	6.962	11.05	13.034	18.64	18.114	18.257
8388608	4.146	7.101	11.692	16.605	20.207	20.785	20.58

Table.1

Graph for Multiply Performance vs Global Dataset Size:



Graph.1

Graph for Multiply Performance vs Local Work Size:

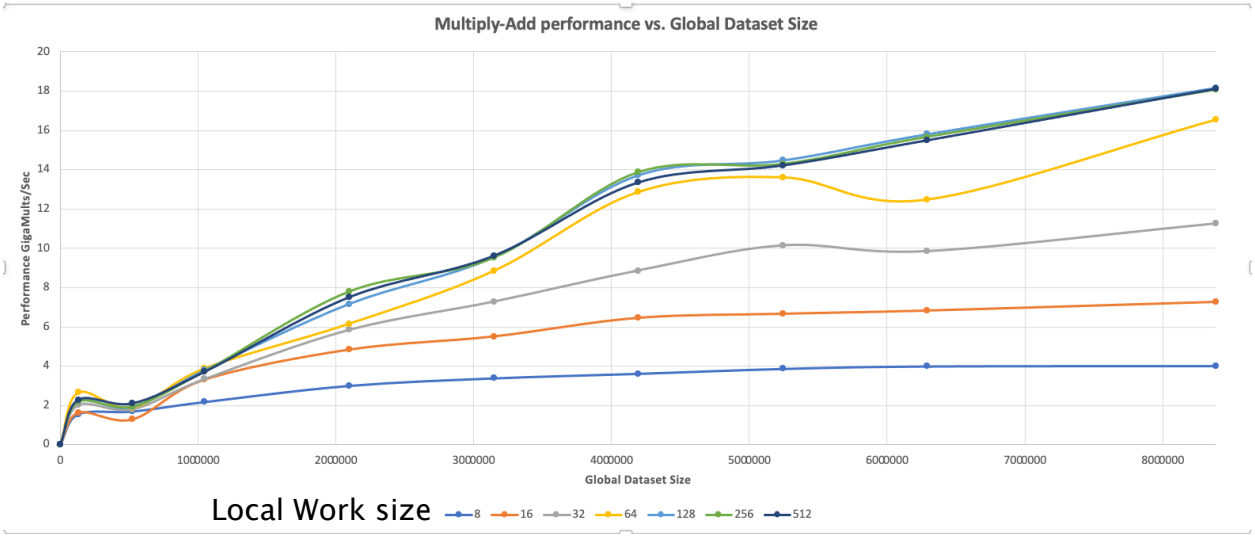
Graph.2

Table for Multiply-Add Performance:

	8	16	32	64	128	256	512
1024	0.017	0.015	0.02	0.017	0.017	0.017	0.018
131072	1.555	1.627	2.018	2.659	2.222	2.19	2.293
524288	1.699	1.291	1.809	1.88	1.918	1.955	2.11
1048576	2.181	3.314	3.345	3.89	3.798	3.742	3.717
2097152	3.008	4.84	5.844	6.166	7.173	7.801	7.512
3145728	3.391	5.51	7.281	8.851	9.551	9.521	9.615
4194304	3.623	6.456	8.862	12.87	13.72	13.874	13.349
5242880	3.873	6.662	10.139	13.619	14.479	14.297	14.217
6291456	4	6.828	9.846	12.486	15.81	15.676	15.497
8388608	4.022	7.267	11.251	16.556	18.175	18.083	18.122

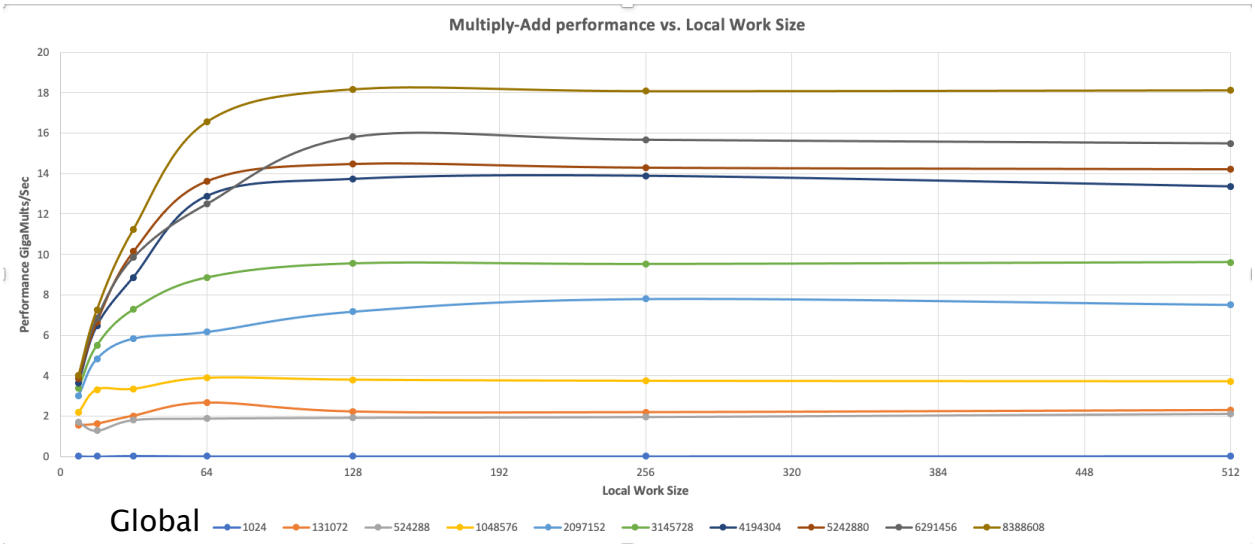
Table.2

Graph for Multiply-Add Performance vs Global Dataset Size:



Graph.3

Graph for Multiply-Add Performance vs Local Work Size:



Graph.4

What patterns am I seeing in the performance curves?

In both Multiply and Multiply-Add performance curves, when the Global Data Size increases, performance increases accordingly.

Before the local work size is less than 128, as the local work size increases, performance increases accordingly. When the local work size is greater than 128, the performance is almost flat and remains unchanged.

Why do I think the patterns look this way?

Because the GPU's workload is divided into a Grid of Work-Groups, and each Work-Group runs on a Compute Unit and is organized as a grid of Work-items. Each Compute Unit is organized as a grid of Processing Elements, and Work-item runs on a Processing Element. One thread is assigned to each Work-item. The more Work-Groups we have the more threads we have, and the higher performance it performs.

When the local work size reaches 128, there are no more compute units, so the performance stays flat.

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

The performance of Multiply added is slightly lower than that of Multiply performance, since it performs an extra add operation. When the length/size of the array is getting very large, the speed gap is becoming slightly more noticeable, but it's not significant.

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

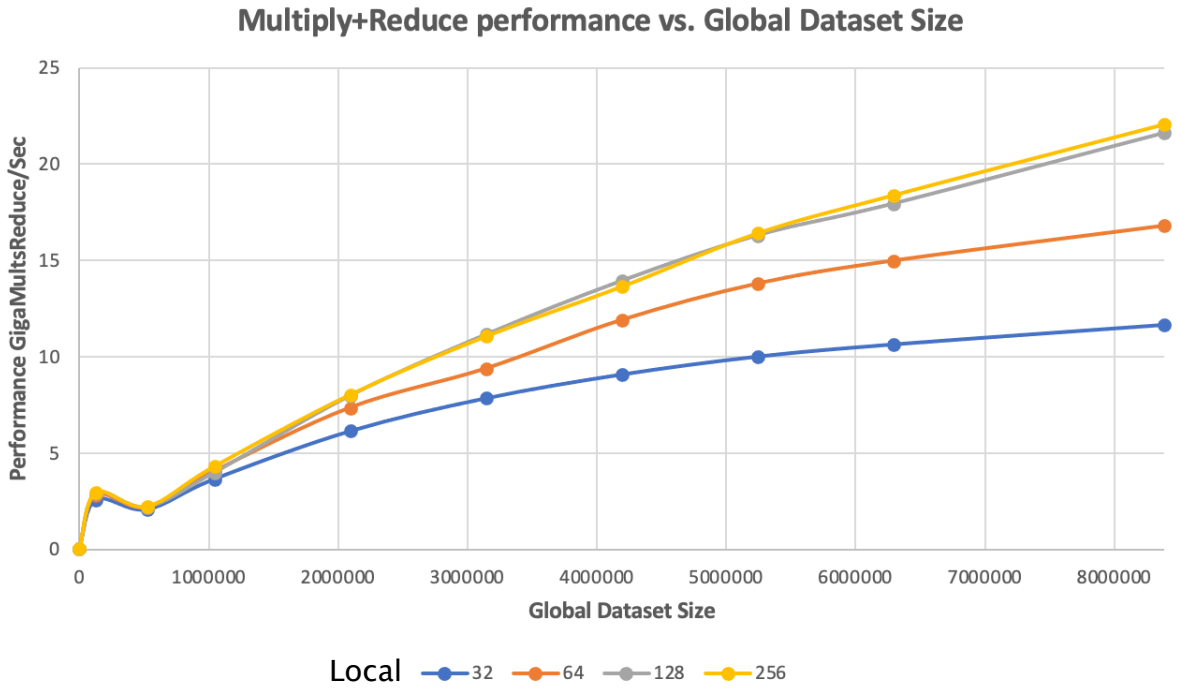
The performance difference is not significant, because of GPU's massive parallel processing power. So the proper use of GPU parallel computing can give us a great performance boost, as it has a large number of cores.

Table for Multiply+Reduce performance:

	32	64	128	256
1024	0.023	0.023	0.022	0.023
131072	2.578	2.822	2.918	2.946
524288	2.072	2.18	2.205	2.218
1048576	3.658	4.07	4	4.309
2097152	6.147	7.359	7.985	8.027
3145728	7.854	9.385	11.153	11.055
4194304	9.09	11.901	13.94	13.629
5242880	10.021	13.783	16.301	16.4
6291456	10.654	14.973	17.944	18.374
8388608	11.675	16.79	21.619	22.066

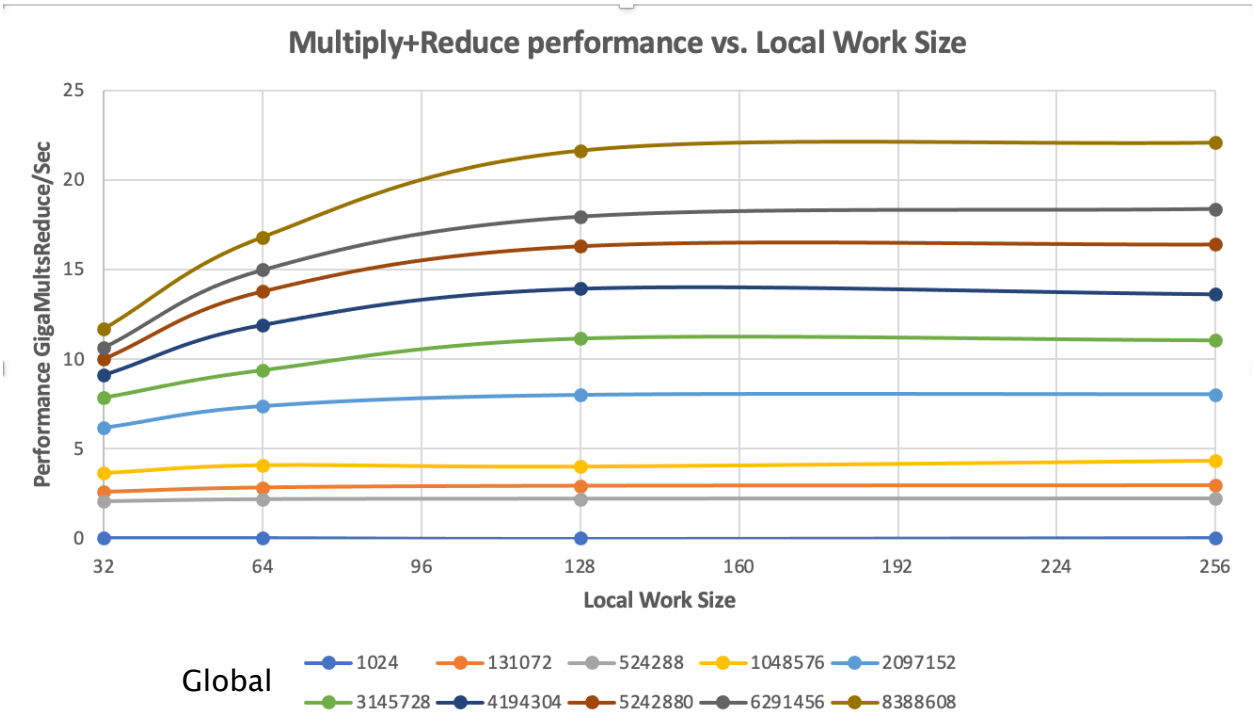
Table.3

Graph for Multiply+Reduce performance vs. Local Work Size:



Graph.5

Graph for Multiply+Reduce performance vs. Local Work Size:



Graph.6

What pattern am I seeing in this performance curve?

In the Multiply Reduce curves, when the Global Data Size increases, performance increases accordingly.

Before the local work size is less than 128, as the local work size increases, performance increases accordingly. When the local work size is greater than 128, the performance is almost flat and remains unchanged.

Why do I think the pattern looks this way?

Because the GPU's workload is divided into a Grid of Work-Groups, and each Work-Group runs on a Compute Unit and is organized as a grid of Work-items. Each Compute Unit is organized as a grid of Processing Elements, and Work-item runs on a Processing Element. One thread is assigned to each Work-item. The more Work-Groups we have the more threads we have, and the higher performance it performs.

When the local work size reaches 128, there are no more compute units, so the performance stays flat.

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

The reason why Multiply reduction is faster than Multiply and Multiply-Add is that when the array multiplications done, we do not put the products into a large global device array, but into a prods[] array that is shared within its work-group.

Therefore, when using GPU parallel computing, taking this into account can improve the performance of computing.