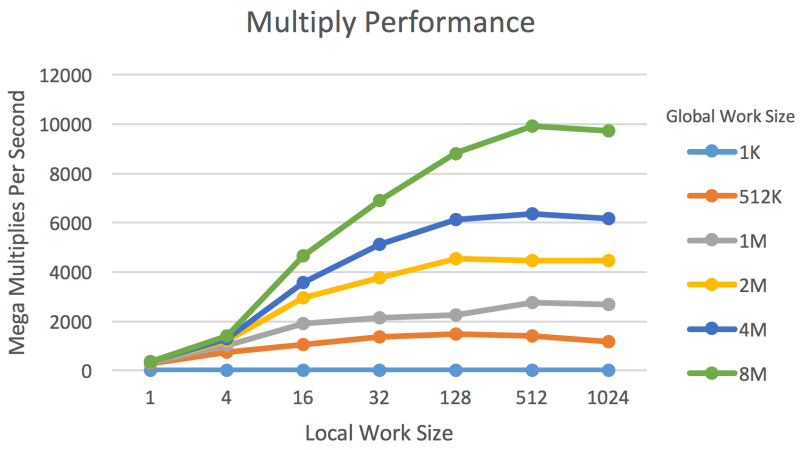
Deqing Qu
qud@oregonstate.edu

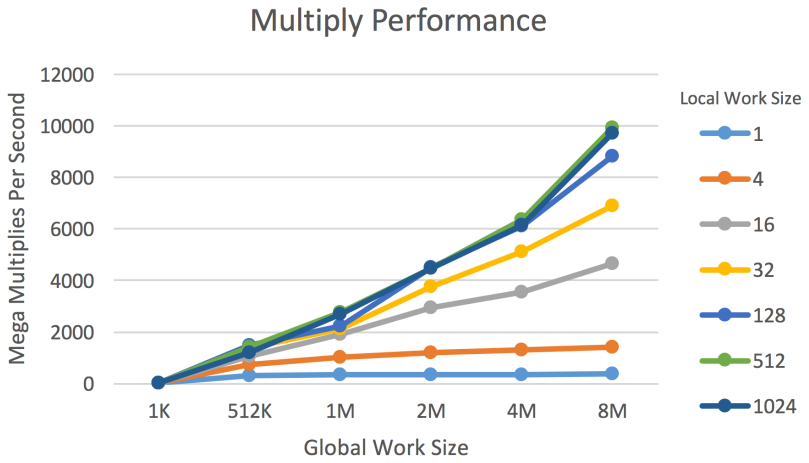
Part 1. Multiply and Multiply-Add

1. What machine you ran this on
   Rabbit
2. Show the tables and graphs

Multiply

Global Work Size / Local Work Size	1K	512K	1M	2M	4M	8M
1	12.634	297.75	335.415	347.764	357.583	368.02
4	12.157	730.267	1028.4	1199.128	1304.054	1405.909
16	9.381	1044.486	1924.526	2953.169	3558.433	4658.192
32	17.012	1382.496	2127.235	3769.023	5131.112	6913.257
128	12.836	1492.512	2248.31	4528.479	6110.645	8814.017
512	15.245	1423.752	2774.568	4477.334	6356.547	9918.18
1024	16.843	1183.366	2697.739	4477.87	6149.519	9732.953

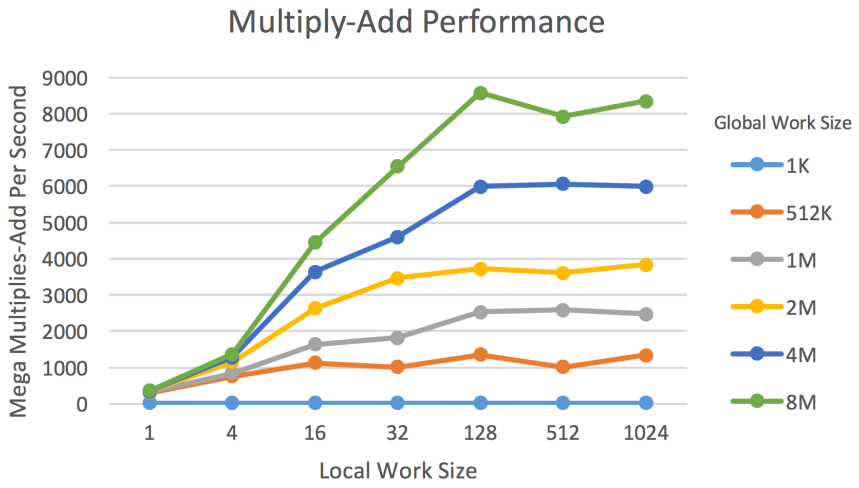


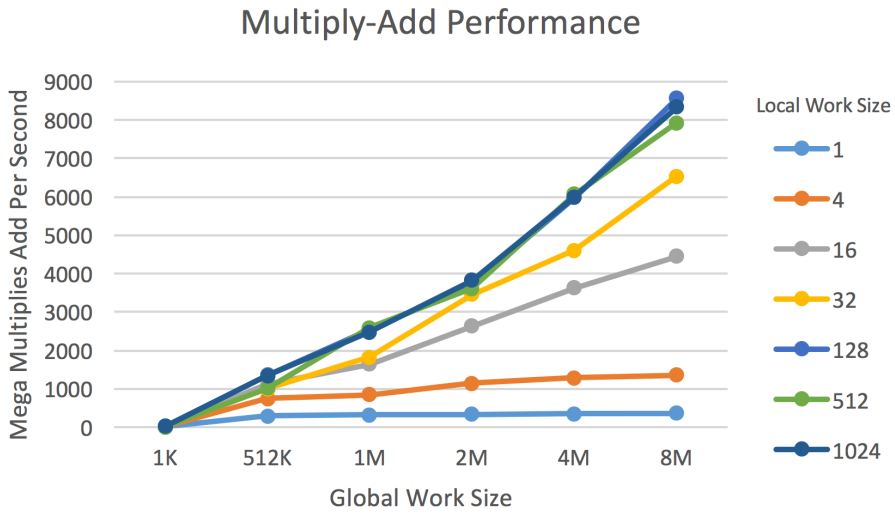


Deqing Qu
qud@oregonstate.edu

Multiply-Add

Global Work Size / Local Work Size	1K	512K	1M	2M	4M	8M
1	12.196	293.998	314.318	338.996	350.037	356.878
4	12.775	749.675	840.435	1145.398	1280.855	1358.077
16	12.507	1121.229	1637.602	2630.931	3629.013	4446.899
32	12.503	1006.628	1820.887	3461.241	4598.781	6537.135
128	17.969	1349.102	2526.458	3716.582	5983.486	8567.375
512	12.858	1015.784	2587.173	3607.215	6065.366	7923.491
1024	16.946	1334.807	2472.643	3828.932	5989.895	8338.278





Deqing Qu
qud@oregonstate.edu

3. What patterns are you seeing in the performance curves?
   For a given global work size, the performance increases when the local work size increases and the performance reaches top when local work size equals to 128.
   For a given local work size, the performance increases when the global work size increases.

4. Why do you think the patterns look this way?
   When the local work size is too small (like 1 or 4), there are more processing elements in the compute units are idle and a lot of compute time are wasted.
   When the global work size is too small, the GPU is not so busy and not enough work done on GPU can't overcome the overhead of setting all up.

5. What is the performance difference between doing a Multiply and doing a Multiply-Add?
   For a given global work size and a given local work size, the performance of doing a Multiply is better than doing a Multiply-Add. This makes sense. The kernel of Multiply-Add is more complicated than Multiply, so the processing time of doing Multiply-Add on GPU is longer.

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

   There is a sweet pot for the local work size and 128 is a good choice, according to the experiments.

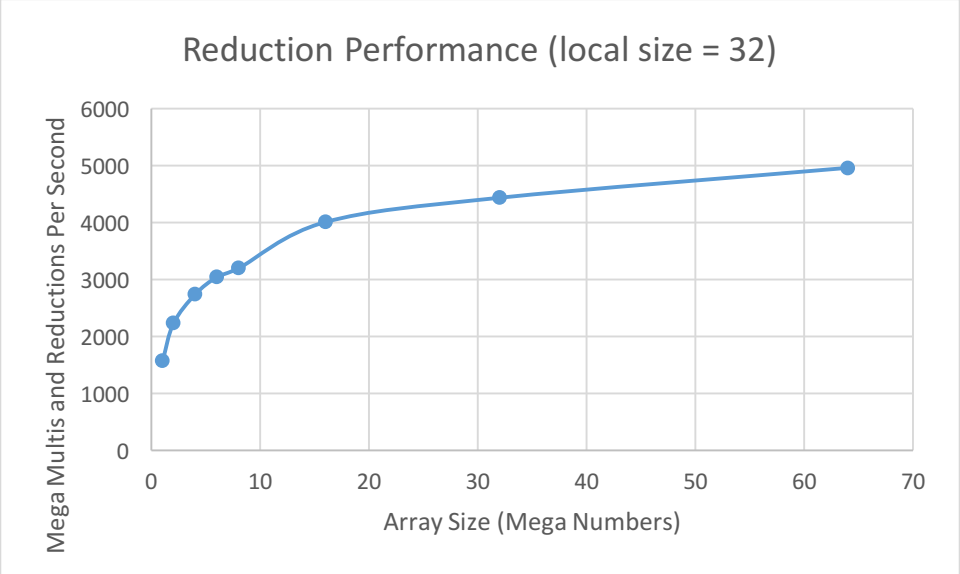
   If the data size is too small, it is not worth to do it on GPU. Only when the data size is big enough, the GPU parallel computing can overcome the overhead of setting up.

Deqing Qu
qud@oregonstate.edu

Part 2. Multiply-Reduction.

1. show the table and graph

Array Size (Mega Numbers)	1	2	4	6	8	16	32	64
Performance (Mega Multi and Reduction Per Second)	1570.919	2231.946	2741.834	3045.247	3200.265	4010.672	4435.883	4960.188



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

   The performance increases as the array size increase and the top performance is close to 5000 Mega Multiplied and Reduced Per Second.

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

   When the array size is less than 60 Mega Numbers, the GPU is not so busy and the overhead may cost too much time.

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

   If the data size is too small, it is not worth to do it on GPU. Only when the data size is big enough, the GPU parallel computing can overcome the overhead of setting up.