Kristin Schaefer

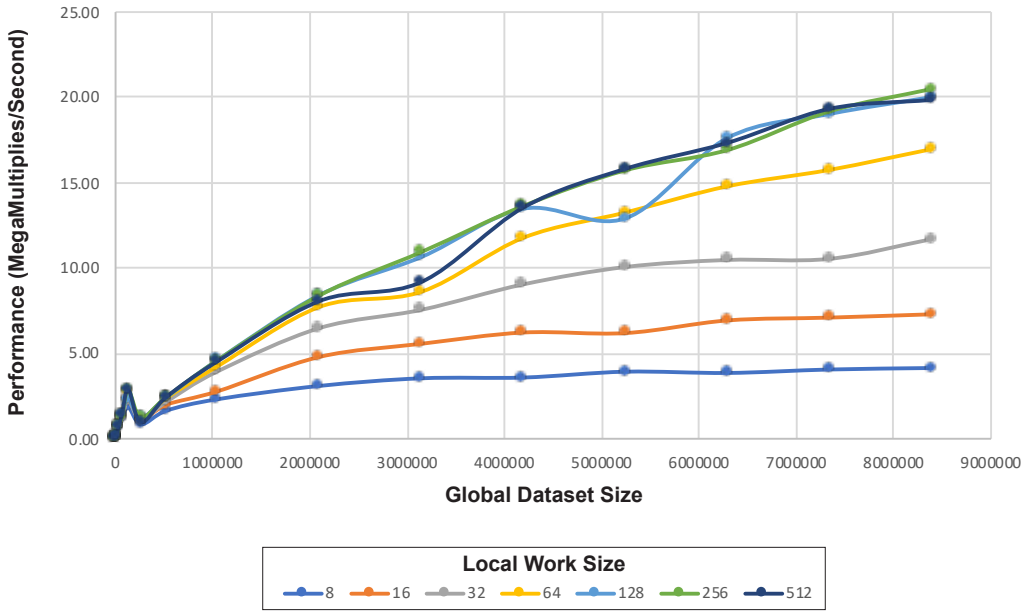
Project #6 : OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

Part I

Performance vs. Global Dataset Size and Local Work Size

Global Dataset Size \ Local Work Size	8	16	32	64	128	256	512
1024	0.02	0.02	0.02	0.02	0.02	0.02	0.02
4096	0.09	0.09	0.09	0.09	0.09	0.09	0.09
8192	0.17	0.18	0.18	0.18	0.18	0.18	0.18
32768	0.65	0.69	0.73	0.72	0.71	0.72	0.74
65536	1.14	1.26	1.20	1.44	1.44	1.31	1.43
131072	1.85	2.30	2.20	2.74	2.36	2.85	2.88
262144	0.85	1.12	1.14	1.15	1.24	1.24	0.96
524288	1.65	2.00	2.17	2.38	2.42	2.46	2.46
1048576	2.32	2.78	3.95	4.25	4.61	4.60	4.54
2097152	3.10	4.80	6.49	7.73	8.43	8.40	8.07
3145728	3.54	5.57	7.56	8.62	10.66	10.95	9.19
4194304	3.59	6.23	9.05	11.78	13.48	13.65	13.56
5242880	3.93	6.19	10.07	13.26	12.92	15.73	15.82
6291456	3.86	6.93	10.48	14.79	17.63	16.92	17.34
7340032	4.06	7.10	10.54	15.75	19.03	19.18	19.33
8388608	4.15	7.29	11.70	16.97	20.02	20.49	19.86

Performance : MegaMultiplies/Second



Max Performance: 20.49 MegaMultiplies/Second with Global Dataset Size of 8,388,608 and Local Work Size of 256

Kristin Schaefer

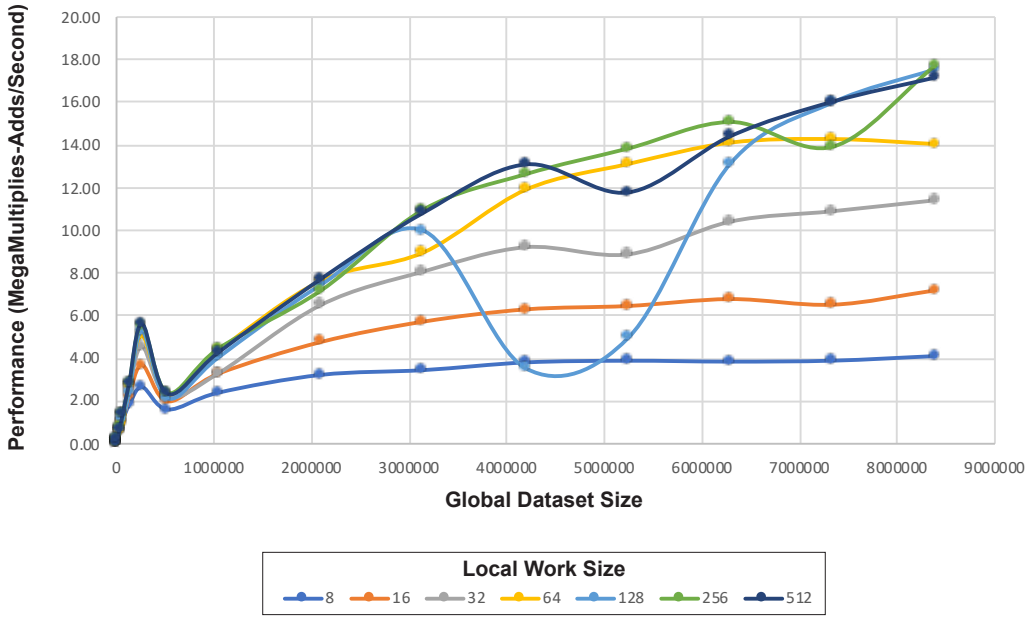
Project #6 : OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

Part II

Performance vs. Global Dataset Size and Local Work Size

Global Dataset Size \ Local Work Size	8	16	32	64	128	256	512
1024	0.02	0.02	0.02	0.02	0.02	0.02	0.02
4096	0.09	0.09	0.09	0.09	0.09	0.09	0.08
8192	0.17	0.14	0.18	0.16	0.17	0.18	0.18
32768	0.65	0.68	0.60	0.60	0.69	0.72	0.71
65536	1.14	1.05	1.38	0.93	1.14	1.41	1.43
131072	1.82	2.24	2.57	2.72	2.42	2.85	2.88
262144	2.66	3.69	4.56	5.08	5.28	5.56	5.62
524288	1.61	1.99	2.20	2.32	2.12	2.38	2.34
1048576	2.40	3.29	3.31	4.38	4.06	4.47	4.30
2097152	3.23	4.77	6.51	7.65	7.46	7.19	7.74
3145728	3.45	5.73	8.08	8.97	9.99	10.98	10.83
4194304	3.81	6.30	9.21	11.91	3.53	12.64	13.10
5242880	3.89	6.46	8.88	13.12	4.96	13.84	11.78
6291456	3.85	6.78	10.41	14.12	13.14	15.08	14.42
7340032	3.90	6.52	10.90	14.28	16.00	13.93	16.04
8388608	4.11	7.19	11.41	14.05	17.56	17.73	17.17

Performance : MegaMultiplies-Adds/Second



Max Performance: 17.73 MegaMultiplies-Adds/Second with Global Dataset Size of 8,388,608 and Local Work Size of 256

Kristin Schaefer

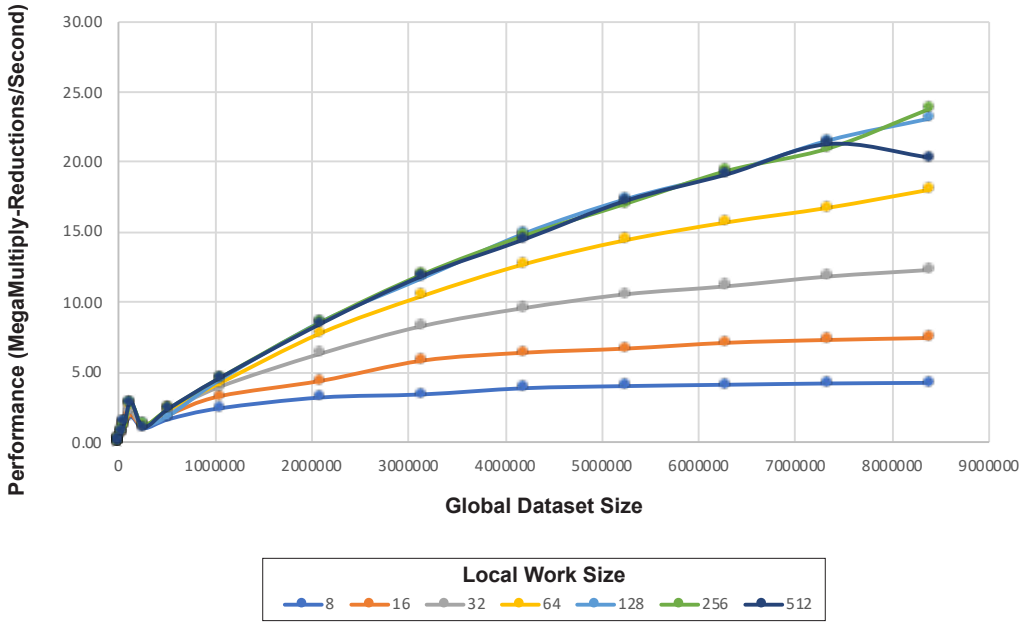
Project #6 : OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

Part III

Performance vs. Global Dataset Size and Local Work Size

Global Dataset Size \ Local Work Size	8	16	32	64	128	256	512
1024	0.02	0.02	0.02	0.02	0.02	0.02	0.02
4096	0.08	0.07	0.09	0.09	0.09	0.09	0.09
8192	0.17	0.18	0.17	0.18	0.18	0.17	0.18
32768	0.64	0.67	0.57	0.73	0.71	0.74	0.73
65536	1.15	1.27	1.37	1.15	1.41	1.20	1.43
131072	1.84	1.95	2.19	2.74	2.75	2.79	2.87
262144	0.99	1.11	1.14	1.21	1.24	1.23	1.08
524288	1.64	1.96	2.19	2.35	1.90	2.41	2.41
1048576	2.41	3.28	3.95	4.21	4.44	4.58	4.58
2097152	3.19	4.38	6.33	7.80	8.60	8.57	8.47
3145728	3.41	5.82	8.29	10.45	11.72	11.99	11.88
4194304	3.85	6.38	9.57	12.71	14.90	14.74	14.48
5242880	4.00	6.68	10.56	14.42	17.35	17.02	17.23
6291456	4.10	7.09	11.13	15.70	19.21	19.37	19.13
7340032	4.19	7.30	11.83	16.74	21.55	20.98	21.29
8388608	4.23	7.44	12.30	18.03	23.11	23.81	20.32

Performance : MegaMultiply-Reductions/Second



Max Performance: 23.81 MegaMultiplies-Reductions/Second with Global Dataset Size of 8,388,608 and Local Work Size of 256

Kristin Schaefer

Project #6 : OpenCL Array Multiply, Multiply-Add, and Multiply-Reduce

Part I & II Anaylsis

Machine:

DGX / NVIDIA V100 graphics card

What patterns are you seeing in the performance curves? Why do you think the patterns look this way?

In both part I and part II there is a consistent increase in performance as the global work size and local work size increase. I think the patterns look this way for two reasons. The first reason being that with GPU data parallelism, as the global work size increases, performance increases. The second reason is that generally the greater the number of threads per workgroup there are, the greater the performance. The performance is significantly worse with less than 32 threads because the number of threads in a warp is 32. If 8 or 16 threads are used, then many of the threads are left idle.

What is the performance difference between Multiply and Multiply-Add? What does this mean for the proper use of GPU Parallel Computing?

The maximum performance for OpenCL Multiply was 20.49 MegaMultiplies/Second with global work size of 8,388,608 and local work size of 256. The maximum performance for OpenCL Multiply-Add was 17.73 MegaMultiplies-Adds/Second with global work size of 8,388,608 and local work size of 256. Thus, there is a difference of 2.76 MegaMultiplies/Second/MegaMultiplies-Adds/Second. The Multiply-Add performance was likely lower because a third array had to be copied from the host to the device and there was an additional add operation. It is worth nothing that for part I, II and III the combination of global work size of 8,388,608 and local work size of 256 always achieved the maximum performance. To make proper use of GPU Parallel Computing I would only consider a global work size of 2,000,000 or greater. I would also run benchmarks with local work sizes of 32 or greater to find local work size which consistently gives the best performance.

Part III Analysis

What pattern are you seeing in the performance curve? Why do you think the pattern looks this way?

As in part I & II, there is a consistent increase in performance as the global work size and local work size increase. I think the patterns look this way for the same reasons stated above: that GPU parallelism generally gives the best performance with large data set sizes and large local work sizes (above 32).

What does this mean for the proper use of GPU Parallel Computing?

As with part I & II, I would only consider using OpenCL and reduction if the dataset size is greater than 2,000,000 and the number of threads is 32 or greater. If applicable, I would use multiply-reduction over multiply, because the performance is greater. I believe this is because reduction takes place in a single workgroup.