CS575 Project 6

Chiu-Chun, Chen

Email: chenchiu@oregonstate.edu

May 31, 2022 (Used two bonus days)

Array Multiply and the Array Multiply-Add portions

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

```
rabbit ~/cs575 263$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
Stepping:              2
CPU MHz:               1200.000
CPU max MHz:           3200.0000
CPU min MHz:           1200.0000
BogoMIPS:              4800.33
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              20480K
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
NUMA node1 CPU(s):     1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep
all nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_goo
_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1
abm epb invpcid_single ssbd ibrs ibpb stibp tpr_shadow vnmi flexpri
 cqm_llc cqm_occup_llc dtherm ida arat pln pts md_clear spec_ctrl i
```

2. Show the tables and graphs

mult.cl	Global Dataset Size	Local Work Size	Work Goups	Performance	Column1
1	1024	8	128	19.554	MegaMults/Sec
1	1024	16	64	17.588	MegaMults/Sec
1	1024	32	32	10.236	MegaMults/Sec
1	1024	64	16	16.291	MegaMults/Sec
1	1024	128	8	11.155	MegaMults/Sec
1	1024	256	4	15.754	MegaMults/Sec
1	1024	512	2	12.002	MegaMults/Sec
10	10240	8	1280	140.245	MegaMults/Sec
10	10240	16	640	158.462	MegaMults/Sec
10	10240	32	320	151.753	MegaMults/Sec
10	10240	64	160	159.83	MegaMults/Sec
10	10240	128	80	121.258	MegaMults/Sec
10	10240	256	40	171.2	MegaMults/Sec
10	10240	512	20	165.962	MegaMults/Sec
100	102400	8	12800	942.164	MegaMults/Sec
100	102400	16	6400	1330.768	MegaMults/Sec
100	102400	32	3200	1621.228	MegaMults/Sec
100	102400	64	1600	953.88	MegaMults/Sec
100	102400	128	800	1491.624	MegaMults/Sec
100	102400	256	400	1557.106	MegaMults/Sec
100	102400	512	200	1585.999	MegaMults/Sec
1000	1024000	8	128000	2489.407	MegaMults/Sec
1000	1024000	16	64000	4424.912	MegaMults/Sec
1000	1024000	32	32000	6686.56	MegaMults/Sec
1000	1024000	64	16000	7581.479	MegaMults/Sec
1000	1024000	128	8000	8838.638	MegaMults/Sec
1000	1024000	256	4000	9063.872	MegaMults/Sec
1000	1024000	512	2000	8393.03	MegaMults/Sec
2000	2048000	8	256000	2824.781	MegaMults/Sec
2000	2048000	16	128000	4741.432	MegaMults/Sec
2000	2048000	32	64000	8341.581	MegaMults/Sec
2000	2048000	64	32000	11527.375	MegaMults/Sec
2000	2048000	128	16000	12678.285	MegaMults/Sec
2000	2048000	256	8000	14013.575	MegaMults/Sec
2000	2048000	512	4000	11507.043	MegaMults/Sec
4000	4096000	8	512000	2934.768	MegaMults/Sec
4000	4096000	16	256000	5474.691	MegaMults/Sec
4000	4096000	32	128000	9354.139	MegaMults/Sec
4000	4096000	64	64000	14558.172	MegaMults/Sec
4000	4096000	128	32000	17498.364	MegaMults/Sec
4000	4096000	256	16000	16943.753	MegaMults/Sec
4000	4096000	512	8000	16326.923	MegaMults/Sec
8000	8192000	8	1024000	3026.967	MegaMults/Sec
8000	8192000	16	512000	5774.248	MegaMults/Sec
8000	8192000	32	256000	9856.556	MegaMults/Sec
8000	8192000	64	128000	16062.713	MegaMults/Sec
8000	8192000	128	64000	19748.231	MegaMults/Sec
8000	8192000	256	32000	18107.668	MegaMults/Sec
8000	8192000	512	16000	18424.887	MegaMults/Sec

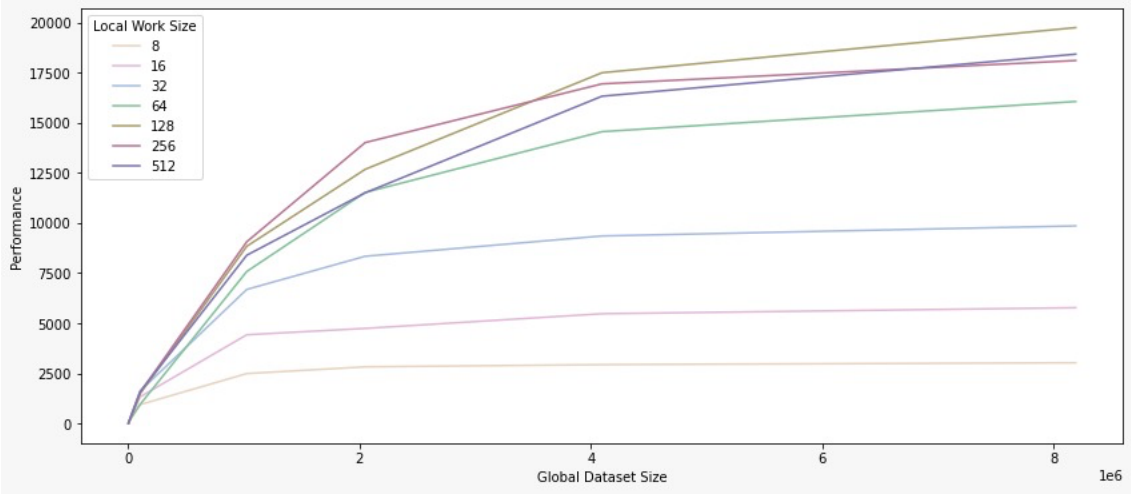

Figure 1. Multiply and multiply-add performance versus global dataset size, with a series of colored constant local work size curves

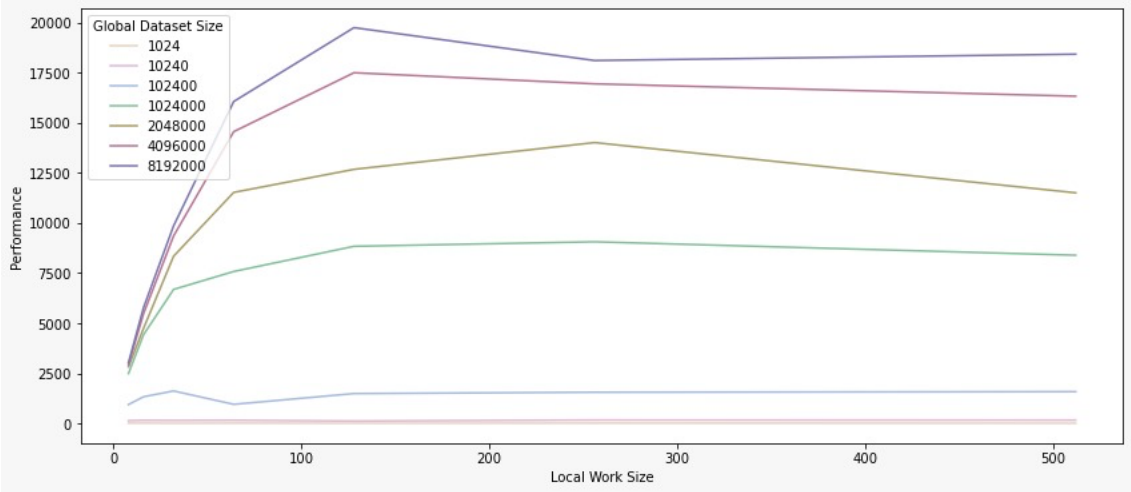

Figure 2. Multiply and multiply-add performance versus local work size, with a series of colored constant global dataset size curves

3. **What patterns are you seeing in the performance curves?**
   In figure 1, we can tell that larger global data set size contributes to better performance. When the local work size is 128, it has the best performance whereas size 256 and 512 have similar performance at the second place.
   We may also conclude from Figure 2 that a greater global dataset size contributes to improved performance. Larger local work sizes, on the other hand, may not automatically imply superior performance. When the local task size is less than 128, the performance of curves with global dataset sizes more than 1M improves, but once they approach 128, they hit a bottleneck and can no longer grow.

4. **Why do you think the patterns look this way?**
   With additional processing units and task groups, each processing unit will have less data to process, which may potentially improve performance.

5. **What is the performance difference between doing a Multiply and doing a Multiply-Add?**
   In general, they both perform similarly.

6. **What does that mean for the proper use of GPU parallel computing?**
   It is vital to evaluate the sizes of work items before beginning our tasks since different task sizes have varying effects on the performance. Our calculation may be done by a large number of threads on the GPU, because GPU processors are designed to handle streaming data. The performance of both array calculations could indicate how effective GPU computing is.

Array Multiply-Reduction portions

1. **Show this table and graph**

NMB	Global Dataset Size	Local Work Size	Work Goups	Performance	Column1
1	1024	32	32	9.617	MegaMults/Sec
1	1024	64	16	16.939	MegaMults/Sec
1	1024	128	8	20.132	MegaMults/Sec
1	1024	256	4	14.995	MegaMults/Sec
10	10240	32	320	144.848	MegaMults/Sec
10	10240	64	160	157.222	MegaMults/Sec
10	10240	128	80	190.949	MegaMults/Sec
10	10240	256	40	131.42	MegaMults/Sec
100	102400	32	3200	1202.245	MegaMults/Sec
100	102400	64	1600	1255.486	MegaMults/Sec
100	102400	128	800	974.821	MegaMults/Sec
100	102400	256	400	1384.701	MegaMults/Sec
1000	1024000	32	32000	1979.727	MegaMults/Sec
1000	1024000	64	16000	2989.825	MegaMults/Sec
1000	1024000	128	8000	2960.958	MegaMults/Sec
1000	1024000	256	4000	2479.263	MegaMults/Sec
2000	2048000	32	64000	2532.911	MegaMults/Sec
2000	2048000	64	32000	3819.962	MegaMults/Sec
2000	2048000	128	16000	4943.576	MegaMults/Sec
2000	2048000	256	8000	3631.772	MegaMults/Sec
4000	4096000	32	128000	3049.9	MegaMults/Sec
4000	4096000	64	64000	4809.741	MegaMults/Sec
4000	4096000	128	32000	5550.918	MegaMults/Sec
4000	4096000	256	16000	5053.677	MegaMults/Sec
8000	8192000	32	256000	3642.759	MegaMults/Sec
8000	8192000	64	128000	5276.44	MegaMults/Sec
8000	8192000	128	64000	7781.717	MegaMults/Sec
8000	8192000	256	32000	6437.296	MegaMults/Sec

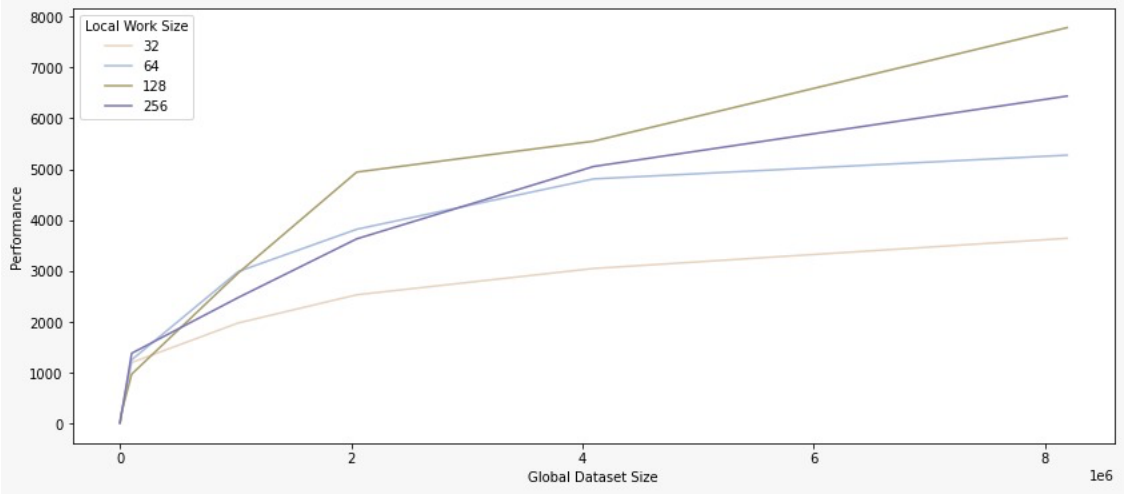

Figure 3. Multiply-reduction performance versus input array size

2. **What pattern are you seeing in this performance curve?**
   In figure 2, we can say that larger global data set size contributes to better performance. Same as figure 1, when the local work size is 128, it has the best performance.

3. **Why do you think the pattern looks this way?**
   Because of the GPU's design, it's possible that utilizing a local work size between 100 to 200 is more appropriate.

4. **What does that mean for the proper use of GPU parallel computing?**
   Same as the aforementioned question, before starting a new task, we must assess the sizes of the work pieces because different job sizes may have distinct effects on performance.