Yen-Chun Chen
chenyenc@oregonstate.edu

Project 06

1. Tell what machine you ran this on.

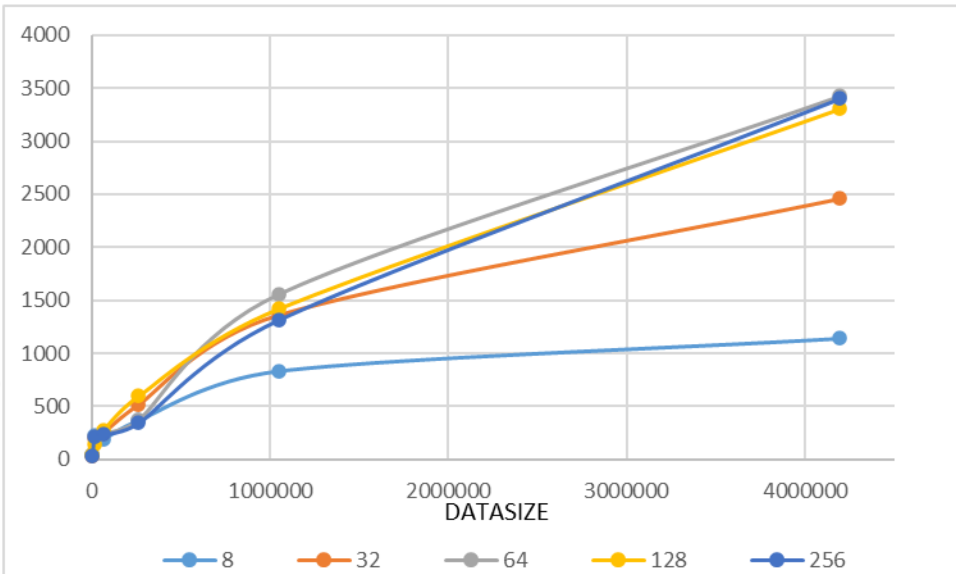
```
rabbit ~ 1001$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
Stepping:              2
CPU MHz:               1282.177
CPU max MHz:           3200.0000
CPU min MHz:           1200.0000
BogoMIPS:              4800.04
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 mtrr pge
 mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall
 nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtop
ology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl v
mx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic
movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm epb i
nvpcid_single ssbd rsb_ctxsw ibrs ibpb stibp tpr_shadow vnmi flexpriority e
pt vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm xsaveopt c
qm_llc cqm_occup_llc dtherm ida arat pln pts md_clear spec_ctrl intel_stibp
 flush_l1d
```

Yen-Chun Chen
chenyenc@oregonstate.edu

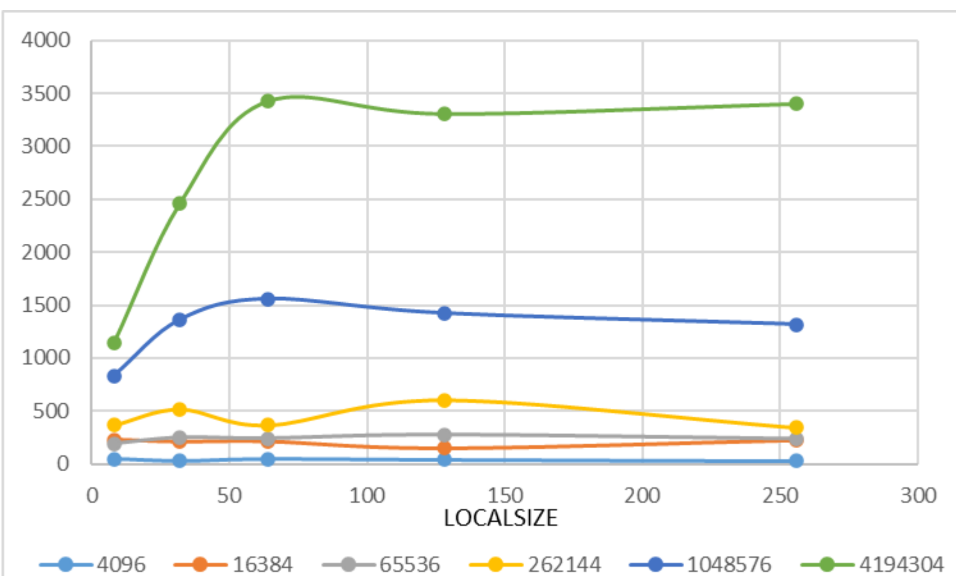
2. Show the table and graphs

DATASIZE	LOCALSIZE 8	LOCALSIZE 32	LOCALSIZE 64	LOCALSIZE 128	LOCALSIZE 256
4096	49.8	35.28	47.99	41.81	32.62
16384	227.77	208.93	210.41	148.25	221.13
65536	191.66	248.98	241.6	275.31	236.37
262144	367.64	514.87	369.51	598.52	345.68
1048576	834.36	1361.35	1557.35	1422.72	1318.08
4194304	1143.42	2461.02	3428.21	3307.79	3402.59

Performance vs. DATASIZE with colored lines of LOCALSIZE



Performance vs. LOCALSIZE with colored lines of DATASIZE



Yen-Chun Chen
chenyenc@oregonstate.edu

3. What patterns are you seeing in the performance curves?

   The first graph shows the relationship between performance and the size of the data. We can tell that larger DATASIZE contributes to better performance. When DATAISZE is the largest which is 4,194,304, it has the best performance.

   The second graph shows the relationship between performance and the number of work-group. It identifies a specific range of the LOCALSIZE and DATASIZE where you can expect the best performance. When operating in this range, we can optimize the performance of the parallel computing tasks. However, beyond this range, the benefit of adding more LOCALSIZE decreases.

4. What difference does the size of the data make?

   The size of the data significantly affects the performance. With small data size such as 4096, the GPU is not fully and effectively utilized, and this leads to lower performance. As the data size increases to higher values such as 65,536 to 262,144, the performance improves because of the better GPU utilization. Lastly, with large data sizes greater than 1,048,576, the GPU almost achieves optimal utilization and gains a significant performance.

5. What difference does the size of each work-group make?

   For the size of each work-group, larger work-group sizes do not necessarily imply better performance, it affects the performance in different ways. When the LOCALSIZE is less than 32, the GPU is not fully utilized which leads to poor performance. As it increases to 32 (a warp), the performance gets better because the GPU can handle more tasks at once. However, with 32, we are probably keeping it close to as busy as it can be because that is a very simple kernel. Therefore, when the work-group sizes are greater than 32, the performances are not so different.

6. Why do you think the patterns look this way?

   The patterns look like this because of how GPU handles different sizes of data and work-group. With small data sizes, GPU is not utilized very well, so the performance is low. As the data size gets larger, the GPU can process more data in parallel, thus the performance improves. However, work-group matter as well. Smaller work-group do not leverage GPU's capability, while the size of work-group above 32 (a warp) utilized it better and improved performance. But if it is a more complicated kernel, the greater size of the work-group should get better performance. All things considered, the best performance happens when both the data size and work-group size are just right for the GPU to work efficiently without too much management effort.