Yen-Chun Chen
chenyenc@oregonstate.edu

Project 05

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

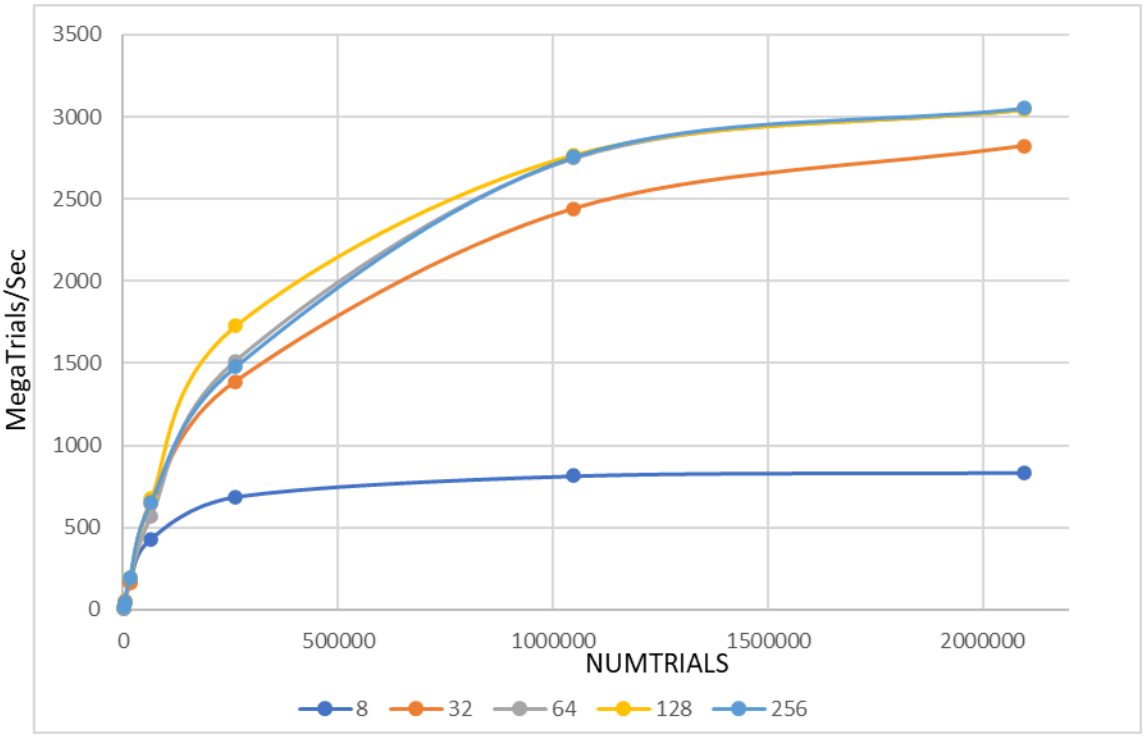
2. What do you think this new probability is?
   This time, the probability around 83.7%, is higher than the probability of project01, around 57%. I think the main reason might be the given data BeforeY, AfterY, DistX, and the hole's RADIUS is bigger.

3. Show the rectangular table and the two graphs.

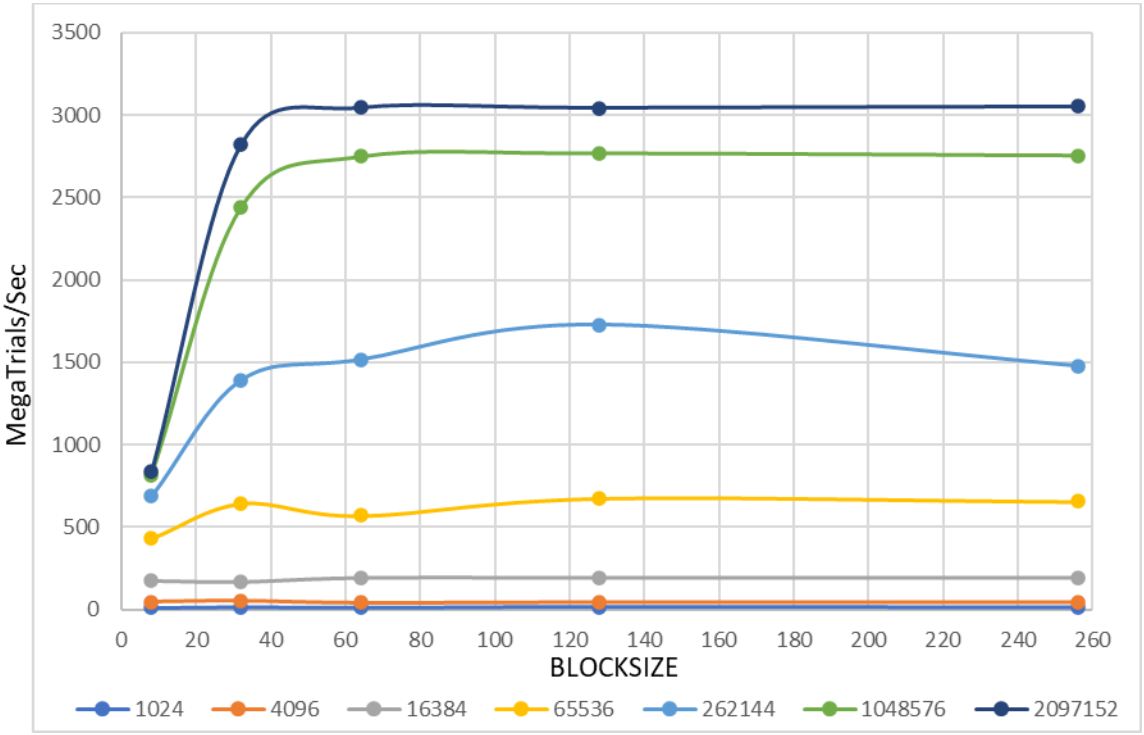
NUMTRIALS	BLOCKSIZE	megaTrialsPerSecond	probability
1024	8	9.9626	82.13
4096	8	46.5455	84.25
16384	8	171.6969	84.01
65536	8	430.7045	83.76
262144	8	686.4421	83.78
1048576	8	813.9703	83.78
2097152	8	834.8642	83.8
1024	32	11.8959	83.79
4096	32	50.7735	84.13
16384	32	164.8953	83.44
65536	32	643.0141	83.81
262144	32	1390.5958	83.72
1048576	32	2440.6375	83.78
2097152	32	2821.4223	83.8
1024	64	10.7419	82.42
4096	64	40.2263	83.23
16384	64	191.1161	84.07
65536	64	568.7309	83.63
262144	64	1515.3533	83.78
1048576	64	2748.3016	83.8
2097152	64	3044.08	83.77
1024	128	12.6783	83.79
4096	128	42.5815	83.79
16384	128	192.6985	84.64
65536	128	674.1277	83.91
262144	128	1727.1769	83.85
1048576	128	2767.8014	83.77
2097152	128	3043.5147	83.82
1024	256	11.8519	85.16
4096	256	42.8667	83.5
16384	256	191.7603	83.51
65536	256	654.3131	83.75
262144	256	1477.9	83.88
1048576	256	2755.6976	83.86
2097152	256	3051.7345	83.79

Yen-Chun Chen
chenyenc@oregonstate.edu

Performance vs. NUMTRIALS with colored lines of BLOCKSIZE



Performance vs. BLOCKSIZE with colored lines of NUMTRIALS



Yen-Chun Chen
chenyenc@oregonstate.edu

4. What patterns are you seeing in the performance curves?
   In the performance curves, the increase in the number of trials leads to an increase in performance, then stabilizes after reaching a specific point. Larger block sizes tend to result in higher performance.

5. Why do you think the patterns look this way?
   Because they are GPUs, which are designed for processing large amounts of data and computations, so once the block size is appropriate for the data amount, more data usually implies higher efficiency. In addition, for parallel computing, many data sets can be collected at the same time and stored in different blocks.

6. Why is a BLOCKSIZE of 8 so much worse than the others?
   Because there are only 8 threads per block, It is totally rational that the fewer threads get the poorer performance. Here, the smallest number of BLOCKSIZE is 8, therefore, it got the worst performance than others.

7. How do these performance results compare with what you got in Project#1? Why?
   Using GPU for calculation completely outperformed using CPU in these assignments since these assignments are toward doing calculations and computations. GPUs are specialized in utilizing a larger number of threads to do the calculation, while CPUs are specialized in handling streaming data.

8. What does this mean for what you can do with GPU parallel computing?
   As I know, GPUs are famous and popular in Machine Learning, Deep Learning, Image and Signal Processing, Cryptographic computations, and so on. The reason is that GPUs are designed with a huge number of smaller and simpler cores that can handle many tasks at the same time. Therefore, GPUs are ideal for parallel processing tasks such as matrix calculation, vector calculation, big data processing, and so on.