Parallel and Distributed Computing
HW03
M10702130 戴健宇

1. What is your achieved speedup and parallel efficiency (with 2, 4, ... 32 threads)?

$ srun -N 1 -w gpu0 ./best.exe Site_56_college_division.pts 18 9 100 100 Site.ply

Thread Numbers	t0	t1	t2	t3
2	2.05009	2.07891	0.0014014	5.39971
4	2.05951	1.07838	0.00164839	5.36217
8	2.0795	0.552569	0.00112621	5.42827
16	2.05964	0.300672	0.00170551	5.40979
32	2.03546	0.182324	0.00162596	5.44619

2. Analyze the characteristic of the Hough Transform function. Is it a compute-bound or a memory-bound problem?

```
kendai  ~/NTUST_Course/Parallel_and_Computing/Homework/HW03  master ● perf stat -e cache-misses,cache-references,instructions,cycles,task-clock ./best.exe Site.pts 18 9 100 100 set1.ply

3.11256, 1.25792, 0.00177131, 8.73792,

 Performance counter stats for './best.exe Site.pts 18 9 100 100 set1.ply':

      200,360,560      cache-misses              #   64.360 % of all cache refs
      311,310,390      cache-references          #   23.753 M/sec
   62,327,428,798      instructions              #    2.03  insn per cycle
   30,772,698,704      cycles                    #    2.348 GHz
     13105.904390      task-clock (msec)         #    0.999 CPUs utilized

     13.116554263 seconds time elapsed
```

使用 perf profiler 可知，cache-misses 所佔的比率太高，代表很少的 data reuse，因此為 memory-bound problem。

3. For mydata struct for storing point-cloud points, which is the better data layout for performance? SoA or AoS?

$ srun -N 1 -w gpu0 ./soa.exe Site_56_college_division.pts 18 9 100 100 Site.ply

$ srun -N 1 -w gpu0 ./aos.exe Site_56_college_division.pts 18 9 100 100 Site.ply

	t0	t1	t2	t3	total
SoA	2.05106	4.82402	0.000948048	5.33002	12.206048048
AoS	2.04464	4.82194	0.000945513	5.37475	12.644035513

就結果而言，兩種 data layouts 的方式並無明顯的差別，但是 SoA 比 AoS 略微快些，故在之後 best 的選擇上採用 SoA 的 data layout 方式。

4. For accumlator struct for storing votes in the parametric space, which is the best/worst data layout out six possible permutations?

$ srun -N 1 -w gpu0 ./p1.exe Site_56_college_division.pts 18 9 100 100 Site.ply

$ srun -N 1 -w gpu0 ./p2.exe Site_56_college_division.pts 18 9 100 100 Site.ply

$ srun -N 1 -w gpu0 ./p3.exe Site_56_college_division.pts 18 9 100 100 Site.ply

$ srun -N 1 -w gpu0 ./p4.exe Site_56_college_division.pts 18 9 100 100 Site.ply

$ srun -N 1 -w gpu0 ./p5.exe Site_56_college_division.pts 18 9 100 100 Site.ply

$ srun -N 1 -w gpu0 ./p6.exe Site_56_college_division.pts 18 9 100 100 Site.ply

	t0	t1	t2	t3	total
p1	2.03009	4.82744	0.000950542	5.44262	12.301100542
p2	2.06809	4.83031	0.00094907	5.41822	12.31756907
p3	2.10771	4.50939	0.000946314	5.44672	12.064766314
p4	2.01702	4.10263	0.000934022	5.4739	11.594484022
p5	2.02182	4.51106	0.000945383	5.49117	12.024995383
p6	2.0622	4.10593	0.000935895	5.35677	11.525835895

從上述可知，p6 (*votes*$[\phi][\theta][\rho]$) 為最佳的 data layout 方式，而 p2 (v*otes*$[\rho][\phi][\theta]$) 為最差的 data layout 方式。

5. How does scheduling strategies (static, dynamic, ...) and their chunk size affects performance?

$ srun -N 1 -w gpu0 ./best.exe Site_56_college_division.pts 18 9 100 100 Site.ply

	Chunk Size	t0	t1	t2	t3
static	1	2.1103	1.56928	0.0016294	5.38531
	512	2.01892	0.238828	0.00165545	5.29987
	1024	2.09695	0.238731	0.00169157	5.31898
dynamic	1	2.09477	13.9791	0.00127934	5.50863
	512	2.04262	1.12525	0.00128029	5.34015
	1024	2.08101	1.05936	0.00129229	5.50682
guided	1	2.0455	0.172994	0.00130987	5.45523
	512	2.08121	0.172049	0.00128197	5.41181
	1024	2.09601	0.177739	0.00122578	5.4385
auto		2.01833	0.177363	0.00198889	5.43543

由上面表格可知，當 chuck size 為 1 時，因被分配的工作數量太少，故一直在分配工作給 thread 做，導致分配工作的 overhead 會太高

6. Other worthy discussions that you discover during this assignment.

無。