Parallel Programming

CS575

Chao Zhang

Project #1

1. Source listing

run.sh

```
#!/bin/csh
#number of threads:
foreach t ( 1 4 8 16 32)
        foreach s (512 1024 1536 2048 3072 4096)
                g++ -DNUMS=$s -DNUMT=$t project1.cpp -o prog -lm -fopenmp
                ./prog
        end
end

```

This is the run.sh file, it is used to run the program file which is the project1.cpp.With this .sh file, I can get all the numbers I need at one time. The NUMT are 1, 4, 8, 16 ,32 and the NUMS are 512, 1024, 1536, 2048, 3072 and 4096.

```
57  int main( int argc, char *argv[ ] )
58  {
59
60      float volume = 0.;
61      // the area of a single full-sized tile:
62      float fullTileArea = (  ( (XMAX-XMIN)/(float)(NUMS-1) )  *  ( ( YMAX - YMIN )/(float)(NUMS-1)
63
64      // sum up the weighted heights into the variable "volume"
65      // using an OpenMP for loop and a reduction:
66
67      omp_set_num_threads(NUMT);
68
69      double start = omp_get_wtime();
70
71      #pragma omp parallel for collapse(2) reduction(+:volume)
72      for( int iv = 0; iv < NUMS; iv++ )
73      {
74          for( int iu = 0; iu < NUMS; iu++ )
75          {
76              float h = Height(iu, iv);
77              float area = h * fullTileArea;
78
79              if (iv == 0 || iv == NUMS-1){
80                  if (iu == 0 || iu == NUMS-1){
81                      area /= 4;
82                  }else{
83                      area /= 2;
84                  }
85              }else{
86                  if (iu == 0 || iu == NUMS-1){
87                      area /= 2;
88                  }
89              }
90
91              volume += area;
92
93          }
94      }
95
96      double end = omp_get_wtime();
97          double sum_height_calculated = (double)(NUMS * NUMS)/(end-start)/1000000.;
98
99
100     printf("Number of threads:      %d\n", NUMT);
101     printf("Number of subdivisions: %d\n", NUMS);
102     printf("Total Area:             %f\n", volume);
103     printf("Average Performance:    %.2f MegaHeights Computed/Sec\n", sum_height_calculated);
```

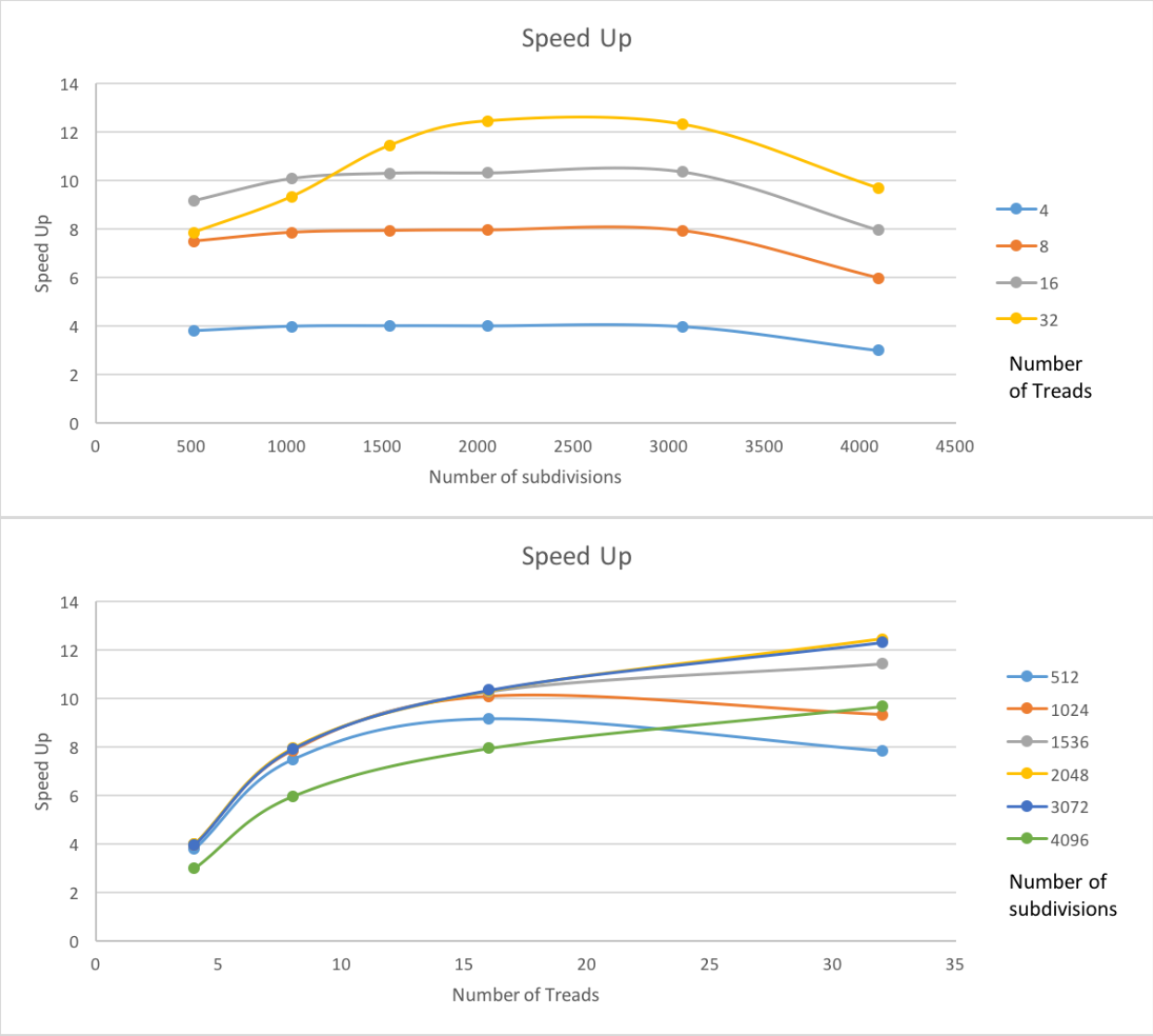
This is the most important part of this project, it uses the OpenMP for loop and the reduction to make sure the results is correct. Inside the for loop, it will calculate the volume use the area and the iu and iv.

2. Result and analysis

I will use the graphs to show the final results. The speed up is the time for 1 core/ time for n core.

Speed Up						
	512	1024	1536	2048	3072	4096
4	3.788	3.969	3.988	3.982	3.949	2.988
8	7.477	7.837	7.914	7.937	7.905	5.956
16	9.157	10.067	10.285	10.298	10.334	7.933
32	7.832	9.313	11.43	12.446	12.303	9.658

Speed Up Table



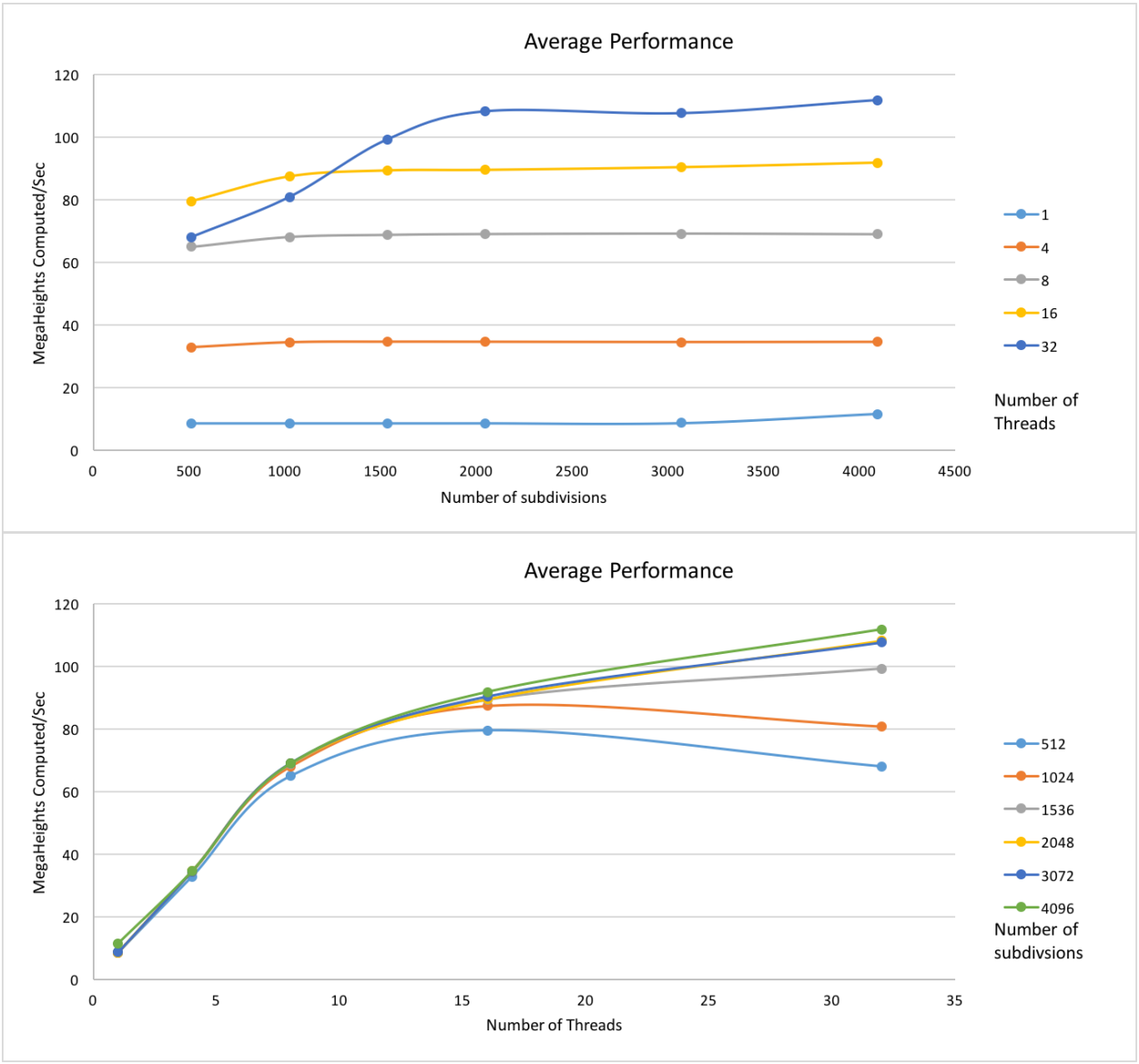
Speed Up Graphs

The average performance is the MegaHeights Computed/Sec of the project.

Average Performance

	512	1024	1536	2048	3072	4096
1	8.69	8.69	8.69	8.697	8.75	11.58
4	32.92	34.49	34.66	34.64	34.56	34.61
8	64.98	68.11	68.77	69.03	69.17	68.98
16	79.58	87.49	89.38	89.56	90.43	91.87
32	68.06	80.93	99.33	108.25	107.65	111.85

The Average Performance Table



The Average Performance Graphs

The reason for the MegaHeights Computed/Sec of the 32 treads with 512 subdivisions is lower than the 16 treads is because the total of subdivisions is lower than the capability of the 32 treads. It finished quickly so the MegaHeights Computed/Sec is lower. The same reason for the speed up.

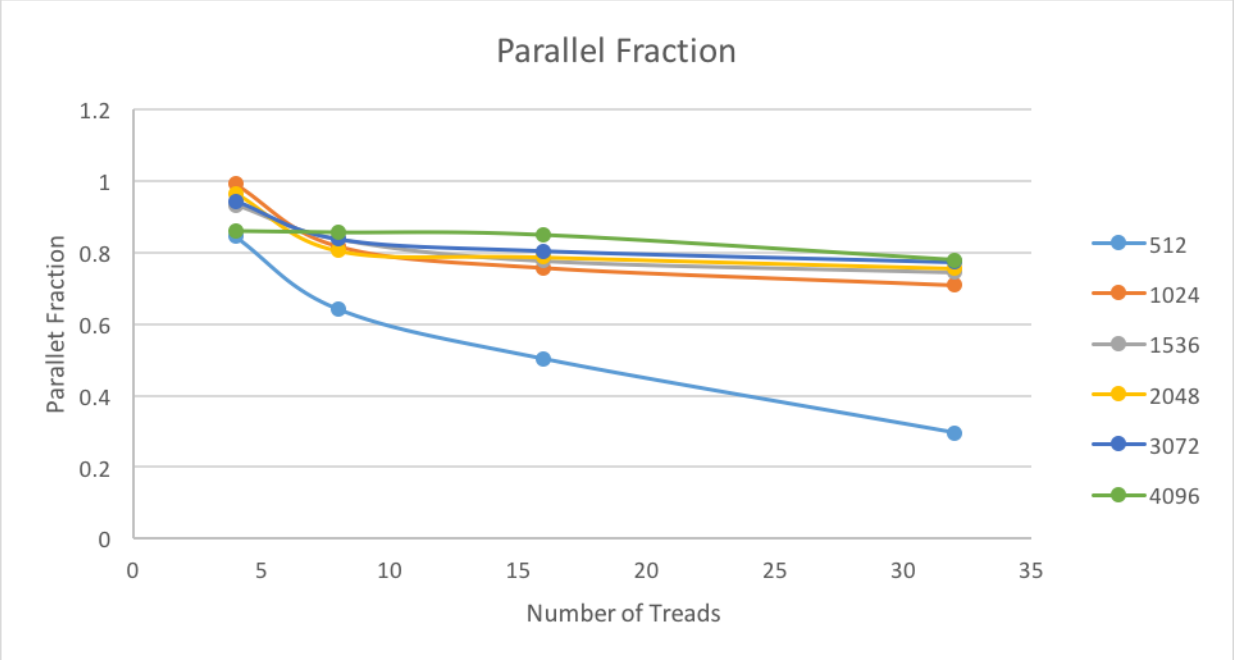
Total Volume						
	512	1024	1536	2048	3072	4096
1	14.0624	14.0623	14.048215	14.067798	14.158243	13.787033
4	14.062441	14.062488	14.063604	14.062373	14.050597	14.087351
8	14.06249	14.062563	14.063004	14.063554	14.059641	14.076975
16	14.06249	14.062487	14.062382	14.06274	14.06315	14.062189
32	14.062488	14.06249	14.062501	14.062503	14.06309	14.062587

With all the data I have, the total area which is the volume are around the 14.065, so I think the actual volume is 14.065.

The Parallel Fraction is calculate with the formula F = NUMT / (NUMT – 1) * (1 – 1 / speedup).

Parallel Fraction						
	512	1024	1536	2048	3072	4096
4	0.845	0.9911	0.933	0.965	0.943	0.8588
8	0.642	0.817	0.837	0.805	0.837	0.855
16	0.503	0.7566	0.776	0.7871	0.804	0.8477
32	0.296	0.708	0.744	0.756	0.773	0.778

Parallel Fraction Table



Parallel Fraction Graphs

The parallel fraction is going down with the increase of the threads, this is because the fraction parallel means the percentage of runtime of the part that can be parallelized in the program. With the growth of threads, the runtime of parallelized part is reduced

The maxSpeedup = 1 / (1- F). The max F is 4 threads in 2048 subdivisions which is 0.965, so the max speedup is 28.57.