Parallel Programming

CS575

Chao Zhang

Project #2

1. Source listing

```
run_cd.sh
#!/bin/bash
#number of threads:
bodies=400
steps=200

for threads in 1 2 4 6 8
do
echo "called with theads: $threads..."
                g++ -DNUMT=$threads -DNUMBODIES=$bodies -DNUMSTEPS=$steps project2_c_d.cpp -o prog -lm -fopenmp
                ./prog
done
```

This is the run_cd.sh file, it is used to run the program file which is the project2_c_d.cpp.I have four files to run the coarse with static, coarse with dynamic, fine with static and fine with dynamic. With those .sh files, I can get all the numbers I need at one time. The threads are 1, 2, 4, 6, and 8. And I give the bodies 400 and the steps 200.

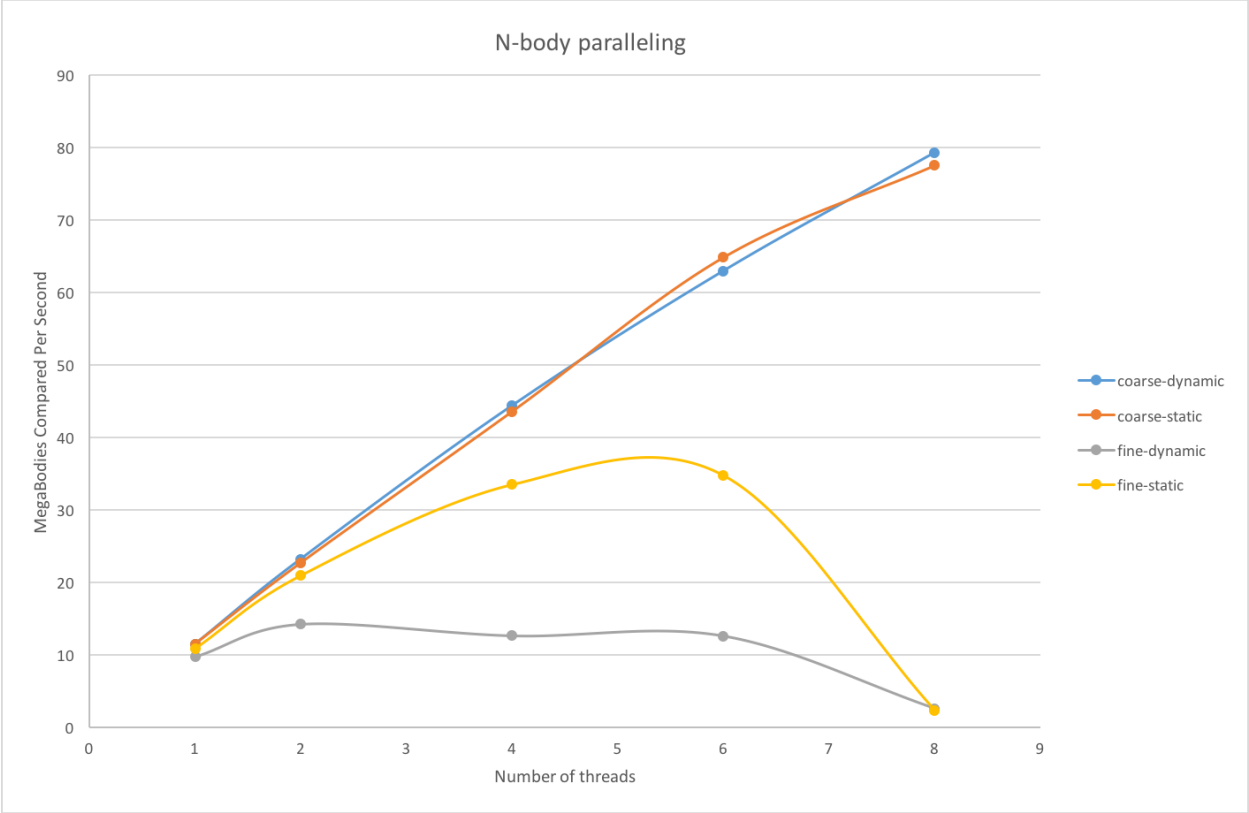
```
68
69      for( int t = 0; t < NUMSTEPS; t++ )
70      {
71          #pragma omp parallel for default(none), shared(Bodies), schedule(dynamic)
72          for( int i = 0; i < NUMBODIES; i++ )
73          {
74              float fx = 0.;
75              float fy = 0.;
76              float fz = 0.;
77              Body *bi = &Bodies[i];
78              for( int j = 0; j < NUMBODIES; j++ )
79              {
80                  if( j == i )    continue;
81
82                  Body *bj = &Bodies[j];
83
84                  float rsqd = GetDistanceSquared( bi, bj );
85                  if( rsqd > 0. )
86                  {
87                      float f = G * bi->mass * bj->mass / rsqd;
```

The #pragma omp parallel for default(none), shared(Bodies), schedule(dynamic) for the coarse is before the I for loop and the fine is before the j for loop. The dynamic or the static is set with the schedule(). The fine-grained parallelism is also has reduction(+:fx, fy, fz) part. I use the MegaBodies Compared Per Second as the out put.

2. Result and analysis

N-body paralleling				
	coarse-dynamic	coarse-static	fine-dynamic	fine-static
1	11.49007	11.49271	9.68475	10.80286
2	23.21708	22.68512	14.2027	20.90993
4	44.38625	43.58333	12.60958	33.50897
6	62.9725	64.87762	12.58939	34.79477
8	79.31071	77.58257	2.55915	2.30283

N-body paralleling Table



N-body paralleling Graphs

By compare the performance of the coarse-grained and the fine-grained, I got the result that the coarse is more effective than the fine. This is because the coarse will put more resources into the paralleling. So the parallel fraction is bigger. Also, the performance of the coarse goes better with the growth of the number of the threads, more threads make the ability of the calculate stronger.

For the fine-grained, the performance goes up in the beginning and goes down after the threads higher than 5. I think this is because the threads used in the fine-grained will spend times to wait the parallel process. When the threads under 5, the waiting time is lower than the increased calculate ability, but after the threads number higher than 5, the waiting time growth bigger and bigger. That is the reason why the fine-grained lines looks like this.