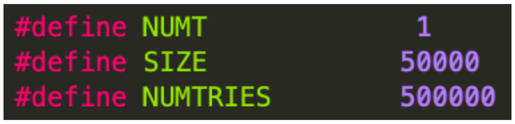
Xiaoru Chen
chenxia2@oregonstate.edu

Project #0
CS 475/575
2021-04-05

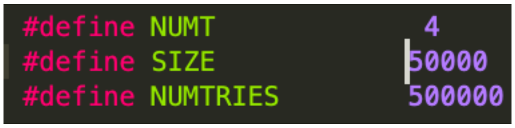
Machine I ran this on:

I used my MacBook pro to ssh to OSU ENGR server.

Key snippets of code:

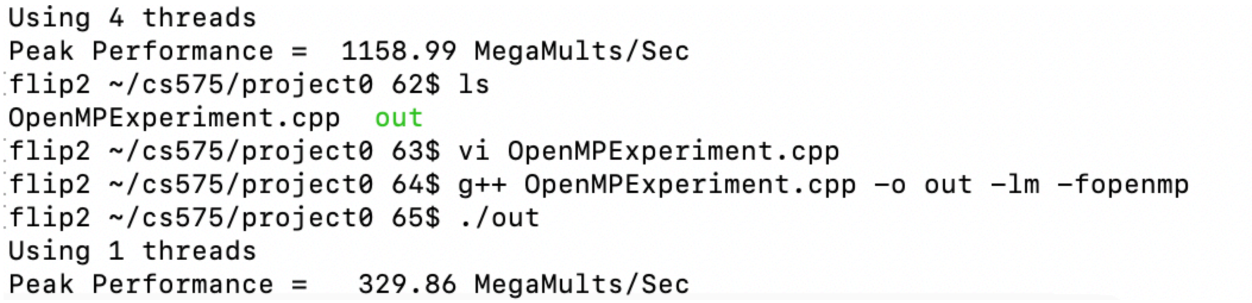


```
#define NUMT            1
#define SIZE        50000
#define NUMTRIES   500000
```



```
#define NUMT            4
#define SIZE        50000
#define NUMTRIES   500000
```

Performance results:



```
Using 4 threads
Peak Performance =  1158.99 MegaMults/Sec
flip2 ~/cs575/project0 62$ ls
OpenMPExperiment.cpp  out
flip2 ~/cs575/project0 63$ vi OpenMPExperiment.cpp
flip2 ~/cs575/project0 64$ g++ OpenMPExperiment.cpp -o out -lm -fopenmp
flip2 ~/cs575/project0 65$ ./out
Using 1 threads
Peak Performance =   329.86 MegaMults/Sec
```

Speedup:

$$S = \frac{Performance\ with\ four\ threads}{Performance\ with\ one\ thread} = \frac{\left(1158.99\left(\frac{MegaMults}{Sec}\right)\right)}{\left(329.86\left(\frac{MegaMults}{Sec}\right)\right)} = 3.514$$

Parallel fraction:

$$Fp = \left(\frac{4}{3}\right) * \left(1 - \left(\frac{1}{S}\right)\right) = \left(\frac{4}{3}\right) * \left(1 - \left(\frac{1}{3.514}\right)\right) = 0.954$$

Analysis:

The speedup is 3.514, which is less than 4.0.

The four threads program finishes faster than the one thread program because it can proceed to work simultaneously. However, a multithreaded program needs to do extra work coordinating multiple threads when calculating. So the speedup cannot reach 4 times.