SALIH KILICLI – STAT 624 HOMEWORK 4

—— SETUP ——

module load intel/2019

export OMP_NUM_THREADS=8

icc -qopenmp -o matmul matmul.c //repeated this part and below after editing the matmul.c file

chmod u+x matmul.c

./matmul

I have edited the matmul.c file that calculates a number of matrix multiplication. I have added **#pragma omp parallel for** right above the part that programs runs 3 nested for loops (which takes the most amount time).

Problem 1) Here is the edited part of the matmul.c code, and their runtimes, whereas I shared a picture of outputs of original file and edited file, respectively.

```
/* Do the matrix product */
start_time = omp_get_wtime();
   #pragma omp parallel for
   for (i=0; i<Ndim; i++){
                  for (j=0; j<Mdim; j++){
                        tmp = 0.0;
                        for(k=0;k<Pdim;k++){
                              /* C(i,j) = sum(over k) A(i,k) * B(k,j) */
                              tmp += *(A+(i*Ndim+k)) *  *(B+(k*Pdim+j));
                        }
                        *(C+(i*Ndim+j)) = tmp; // C[i][j] = tmp
                  }
   }
/* Check the answer */
```

Runtime for Original File: Order 2500 multiplication in 29.096020 seconds

Order 2500 multiplication at 1074.030057 MFLOPS

Runtime for Edited File: Order 2500 multiplication in 4.451982 seconds

Order 2500 multiplication at 7019.345507 MFLOPS

SALIH KILICLI – STAT 624 HOMEWORK 4

export OMP_NUM_THREADS=8

icc -qopenmp -o mandel mandel.c //repeated this part and below after editing the mandel.c file

./mandel

I have edited the mandel.c file that calculates the area of mandel sets. I have added **#pragma omp parallel for default(shared) private(c, j) firstprivate(eps)** right above double for loops and **#pragma omp critical** line right before the definitions of c.i and c.r. I also shared an output in the final remarks.

Problem 2) Here is the edited part of the mandel.c code, and their runtimes, whereas I shared a picture of outputs of original file and edited file, respectively.

```
// Loop over grid of points in the complex plane which contains the Mandelbrot set,
// testing each point to see whether it is inside or outside the set.

  #pragma omp parallel for default(shared) private(c, j) firstprivate(eps)
  for (i=0; i<NPOINTS; i++) {
   for (j=0; j<NPOINTS; j++) {
   #pragma omp critical
    c.r = -2.0+2.5*(double)(i)/(double)(NPOINTS)+eps;
    c.i = 1.125*(double)(j)/(double)(NPOINTS)+eps;
    testpoint();
   }
  }
// Calculate area of set and error estimate and output the results
```

Runtime for Original File: real 0m7.973s Approximates to = 1.51084062 +/- 0.00050361
user 0m7.968s
sys 0m0.006s

Runtime for Edited File: real 0m4.296s Approximates to = 1.52622063 +/- 0.00050874
user 0m29.435s
sys 0m0.016s

Comparing the results we see that parallelized code work around 2x times faster.

—— FINAL REMARKS ——

Problem 1: Output 1 – for running the original and edited version of matmul.c file and their runtimes.

```
[math3mantic@terra1 hw4]$ export OMP_NUM_THREADS=8
[math3mantic@terra1 hw4]$ icc -qopenmp -o matmul matmul.c
[math3mantic@terra1 hw4]$ chmod u+x matmul.c
[math3mantic@terra1 hw4]$ ./matmul
 Order 2500 multiplication in 29.096020 seconds
 Order 2500 multiplication at 1074.030057 mflops

 Hey, it worked
 all done
[math3mantic@terra1 hw4]$ export OMP_NUM_THREADS=8
[math3mantic@terra1 hw4]$ icc -qopenmp -o matmul matmul.c
[math3mantic@terra1 hw4]$ ./matmul
 Order 2500 multiplication in 4.451982 seconds
 Order 2500 multiplication at 7019.345507 mflops

 Hey, it worked
 all done
```

Apparently, it runs around 7 times faster the original file after parallelizing the code using OpenMP.

Problem 2: Output 2 – for running the original and edited version of mandel.c file and their runtimes.

```
[math3mantic@terra1 hw4]$ export OMP_NUM_THREADS=8
[math3mantic@terra1 hw4]$ icc -qopenmp -o mandel mandel.c
[math3mantic@terra1 hw4]$ time ./mandel
Area of Mandlebrot set =   1.51084062 +/-   0.00050361
Correct answer should be around 1.510659

real    0m7.973s
user    0m7.968s
sys     0m0.006s
[math3mantic@terra1 hw4]$ export OMP_NUM_THREADS=8
[math3mantic@terra1 hw4]$ icc -qopenmp -o mandel mandel.c
[math3mantic@terra1 hw4]$ time ./mandel
Area of Mandlebrot set =   1.52622063 +/-   0.00050874
Correct answer should be around 1.510659

real    0m4.296s
user    0m29.435s
sys     0m0.016s
[math3mantic@terra1 hw4]$
```

So, the edited version runs almost x2 faster than the original mandel.c file and give pretty good approximation.