CS6230 Programming Assignment 2 Report

Che Chang, uNID: u1302280

November 5, 2022

1 Problem 1

1.1 1-(a)

1. Doing dependence analysis with this problem, we observe that loop k carries a dependence, so the stride of k would impact the performance of this program.

2. We can increase the stride of k by placing the k-loop outermost.

3. modified code:

```c
#pragma omp parallel private(i, j, k)
{
#pragma omp master
for (k = 0; k < n; k++)
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            // ...
}
```

4. Performance report (on kingspeak):

```
Matrix Size = 401; NTrials=5
Reference sequential code performance in GFLOPS Min: 0.24; Max
    : 0.24
Max Threads (from omp_get_max_threads) = 32
Performance (Best & Worst) of parallelized version: GFLOPS on
    1/2/4/8/10/12/14/15/31 threads
Best Performance (GFLOPS): 3.26 3.25 3.25 3.25 3.25 3.25 3.25
    3.21 3.21
Worst Performance (GFLOPS): 3.25 3.24 3.24 3.24 3.24 3.21 3.05
     3.18 3.16
```

1.2 1-(b)

1. Performance can be boosted with loop work-sharing on the i loop, and static scheduling (since each iteration takes roughly the same amout of time).

2. Performance report (on kingspeak):

```
Matrix Size = 401; NTrials=5
Reference sequential code performance in GFLOPS Min: 0.24; Max: 0.24
Max Threads (from omp_get_max_threads) = 32
Performance (Best & Worst) of parallelized version: GFLOPS on 1/2/4/8/10/12/14/15/31 threads
Best Performance (GFLOPS): 3.22 6.32 13.01 29.21 33.55 40.28 40.78 40.44 38.04
Worst Performance (GFLOPS): 3.21 6.25 12.83 27.15 30.50 2.14 1.79 37.54 1.40
```

1.3 1-(c)

1. I tried unrolling on each loop, but they all don't seem to enhance performance.

2 Problem 2

1. I analyzed the dependences first, and realized that the k loop again carries the dependence, so I made the k loop outmost.

2. I then calculated the bounds so it outputs the correct result.

3. And lastly I parallelized the i loop

4. Performance report (on kingspeak):

```
Matrix Size = 801; NTrials=5
Reference sequential code performance in GFLOPS Min: 0.15; Max: 0.15
Max Threads (from omp_get_max_threads) = 32
Performance (Best & Worst) of parallelized version: GFLOPS on 1/2/4/8/10/12/14/15/31 threads
Best Performance (GFLOPS): 3.11 6.13 12.18 23.76 23.32 24.59 19.64 19.77 17.86
Worst Performance (GFLOPS): 3.11 5.92 10.79 6.24 2.67 2.38 2.09 2.05 0.91
```

3 Problem 3

1. Since S1, S2 has no dependence, for the compiler to better execute vectorization, I separated S1 and S2 and place them in separate nested for loops.

2. Then I performed loop permutation, and parallelization.

3. Performance report (on kingspeak):

```
Reference sequential code performance in GFLOPS Min: 0.54; Max: 0.54
Max Threads (from omp_get_max_threads) = 32
Performance (Best & Worst) of parallelized version: GFLOPS on 1/2/4/8/10/12/14/15/31 threads
Best Performance (GFLOPS): 1.88 3.44 5.76 7.61 5.83 6.28 4.34 4.15 0.34
Worst Performance (GFLOPS): 1.84 3.40 5.59 2.60 0.91 0.96 0.75 0.72 0.22
```