*CSE 6010 Assignment 7*
*Zheng Cai*

## *Introduction*

This program uses both static and dynamic parallel computing to solve the matrix multiplication problem by openMP. The form of that matrix multiplication is like C=A*B and both matrixes have the same size of columns and rows. For the static part, I assign each thread to calculate particular rows and finally merge them to the integral result. But throughout dynamic parallel computing, the idle thread will be assigned to calculate a row to make the use of all the resources.

## *Accuracy of the computing*

After running that program, I compared the results with the Matlab. The example a and b are two 4*4 matrix:

```
The matrix a is
0.131538 0.458650 0.218959 0.678865
0.934693 0.519416 0.034572 0.529700
0.007698 0.066842 0.686773 0.930436
0.526929 0.653919 0.701191 0.762198
The matrix b is
0.000008 0.755605 0.532767 0.047045
0.679296 0.383502 0.830965 0.053462
0.671149 0.383416 0.417486 0.588977
0.846167 0.091965 0.415999 0.910321
```

The result from this program is:

```
1.032948 0.421668 0.825021 0.777655
0.824263 0.967425 1.164379 0.574300
1.293637 0.380338 0.733424 1.255424
1.559759 0.987872 1.433926 1.166578
```

And the result from matlab is:

```
>> a*b

ans =

    1.0329    0.4217    0.8250    0.7777
    0.8243    0.9674    1.1644    0.5743
    1.2936    0.3803    0.7334    1.2554
    1.5598    0.9879    1.4339    1.1666
```

Which is the same to the former one. Hence, we can conclude the calculation of that program is correct.

### *Running time under different threads and size of matrix*

| Size of matrix | | | | | |
|---|---|---|---|---|---|
| 50*50 | | | 2000*2000 | | |
| Threads | Static | Dynamic | Threads | Static | Dynamic |
| 5 | 0.02009 | 0.019827 | 5 | 16.40631 | 10.511122 |
| 20 | 0.035831 | 0.021153 | 20 | 14.12339 | 10.405142 |
| 35 | 0.027931 | 0.000512 | 35 | 14.056002 | 10.387928 |
| 50 | 0.011378 | 0.000667 | 50 | 13.968066 | 10.45053 |
| 65 | 0.012052 | 0.000443 | 65 | 14.14223 | 10.414822 |
| 80 | 0.017342 | 0.000452 | 80 | 13.902101 | 10.36864 |
| 95 | 0.019809 | 0.000384 | 95 | 13.9724 | 10.428028 |

Obviously, dynamic parallel computing is more efficient than static one from the results of running time when they process the same size of the matrix. If the number of threads is fewer than the row size of the matrix, adding threads can improve the speed of the static parallel computing but doesn't work when threads exceed the row size (which means a lot of threads are idle.) In that case, dynamic parallel is the better way to use. For the dynamic parallel computing, the general rule from the table above is that more threads make the program faster although some of them don't follow it. And we can see under the tremendous size of the matrix, increasing the limited number of threads only speeds up the calculation a little.