**CSCE 659 Fall 2017**

**HW 3: Dense Matrix Computations with OpenMP**

**Haiping Xue**

**UIN:122006286**

**Problem 1**

For my code, use following command to compile the code for omp code and code without parallel.

icc -qopenmp -o Rinverse Rinverse.c

icc -qopenmp -o ori ori.c

Execute the code use following command to execute the code where 2048 specifies the size of a n\*n matrix.

./Rinverse 2048

./ori 2048

job file for the code:

OMP\_NESTED=TRUE

module load intel/2017A

export OMP\_STACKSIZE=512M

export OMP\_NUM\_THREADS=20

./Rinverse 6144

export OMP\_NUM\_THREADS=10

./Rinverse 6144

export OMP\_NUM\_THREADS=4

./Rinverse 6144

export OMP\_NUM\_THREADS=2

./Rinverse 6144

export OMP\_NUM\_THREADS=1

./Rinverse 6144

#single core

./ori 6144

problem 2

For me, I first parallelized the recursive calls for computerInverse routine with #pragma omp task. Then I parallelized all copy operations in computerInverse routines including for loops and copyM, assginM. After this, I found multiply routine with nested for loops is one place I can boost my performance. Then I parallelized it using omp parallel for. For my inverseUp routine used for inverse an upper triangular matrix, I boost the speed using back substitution method. This improve the performance of matrix inverse. I choose not parallel it since I found the can solve a matrix of size 100 X 100 in very short time. Therefore, I think parallelize it will only degrade the performance. Other than that, all the routines with for loops and recursions in computerInverse are parallelized.

Problem 3

The speedup and efficiency of computerInverse routine for matrix of 6144. The best performance is achieved when the cores equals to 20. As we can see the performance of the algorithm increases while we are adding more cores. However, the efficiency drops.

|  |  |  |  |
| --- | --- | --- | --- |
| cores | time | speedup | eifficiency |
| 1 | 4.73E+01 | 1.00E+00 | 1.00E+00 |
| 2 | 3.06E+01 | 1.54E+00 | 7.71E-01 |
| 4 | 1.56E+01 | 3.04E+00 | 7.60E-01 |
| 10 | 6.47E+00 | 7.31E+00 | 7.31E-01 |
| 20 | 4.02E+00 | 1.18E+01 | 5.89E-01 |