GAUTHAM SRINIVASAN
927008557
CSCE 689 HW2

1. You are required to develop a parallel C/C++ implementation of the above algorithm that uses OpenMP directives to parallelize the compute_inverse routine. You should use the task directive for the recursive tasks.

   Compilation:
   icc -qopenmp -o matrix_inverse.exe matrix_inverse.c

   Execution:
   ./matrix_inverse.exe 2 *//where 2 indicates dimension of matrix (2x2)*

   Using Job file:
   bsub < matrix_inverse.job

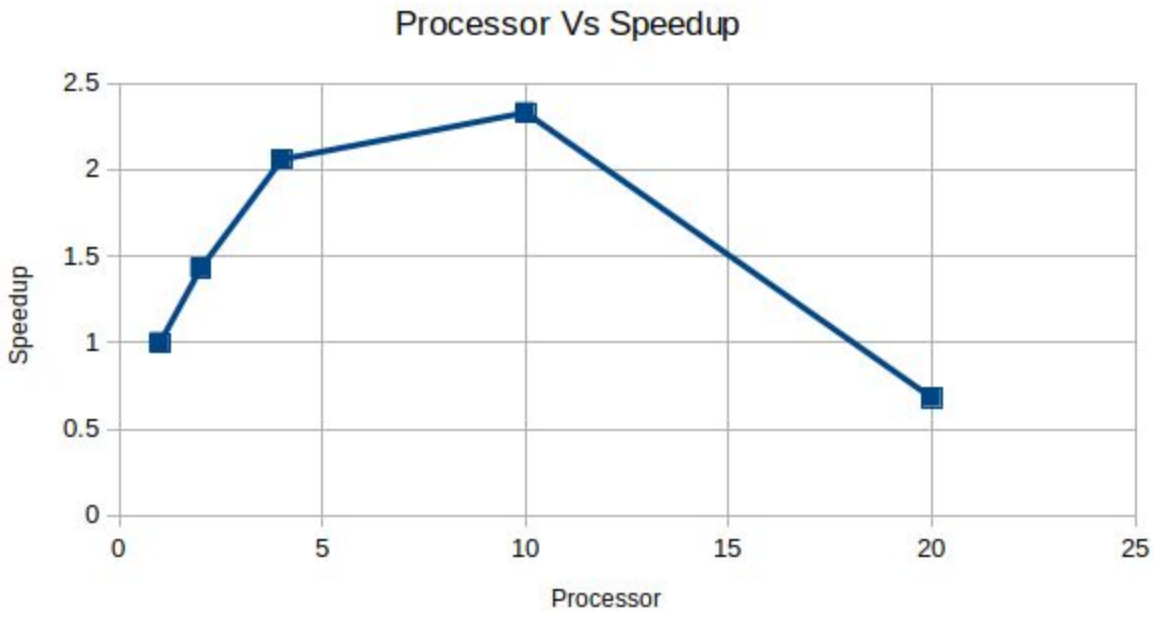
2. Describe your strategy to parallelize the algorithm. Discuss any design choices you made to improve the parallel performance of the code.

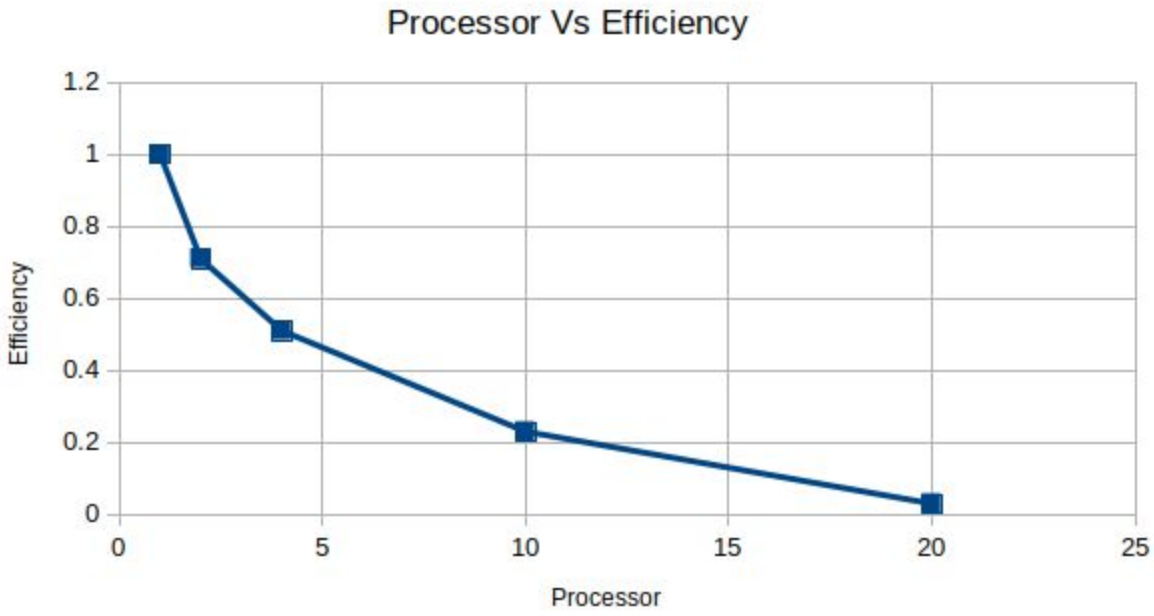
   Just like the implementation in matlab, compute_inverse function divides the matrix upper left and bottom right and calculates its inverse recursively. This can be done by task region where each thread executes its own task using its own data environment and the data variables are shared for synchronization. After computing the inverse of upper left and bottom right, upper right matrix is computed parallely and finally these partial matrices are combined.

3. Determine the speedup and efficiency obtained by your routine on 1, 2, 4, 10, and 20 Processors.

   Following observations made for the matrix of the size 512,

Processors	**Time (sec)**	**Speedup**	**Efficiency**
1	4.00710e-02	1	1
2	2.80149e-02	1.43	0.71
4	1.93539e-02	2.06	0.51
10	1.71759e-02	2.33	0.23
20	5.80049e-02	0.68	0.03


Processor Vs Speedup
Speedup
2.5
2
1.5
1
0.5
0
0
5
10
15
20
25
Processor


Processor Vs Efficiency
Efficiency
1.2
1
0.8
0.6
0.4
0.2
0
0
5
10
15
20
25
Processor