PARALLEL COMPUTING ASSIGNMENT 2

BHUVNESH JAIN (2014A7PS028P)
CHIRAG AGARWAL (2014A7PS033P)

Problem : Traveling Salesman Problem

a. OpenMP Version

The below results were obtained on “Intel Core i7” quad-core processor using 8 threads. The programs were written in “C++” language and OpenMP was used for writing the parallel code. 2 different datasets were considered. Each dataset was run for 5 times and the average execution time was recorded below. The datasets were as follows:

Number of nodes = 10, Cost of any edge <= 1000

Number of Threads	Execution Time	Speedup
1	2.572041	-
2	2.036779	1.262
4	1.564739	1.643
8	1.122410	2.101

Number of nodes = 12, Cost of any edge <= 1000

Number of Threads	Execution Time	Speedup
1	25.256320	-
2	20.843581	1.212
4	15.943161	1.584
8	13.871254	1.820

Conclusion:
The scaling was almost linear till 4 threads but sub-linear afterwards due to hyper-threading playing a major role.

b. MPI Version

The below results were obtained on a cluster of 2 nodes each with “Intel Core i7” quad-core processor. The programs were written in “C++” language and MPI was used for writing the parallel code. 2 different datasets were considered. Each dataset was run for 5 times and the average execution time was recorded below. The datasets were as follows:

Number of nodes = 12, Cost of any edge <= 1000

(Sequential code on single processor took 5.386 seconds)

Number of Processors	Execution Time	Speedup
2	5.519699	0.976
4	4.698604	1.15
8	4.315705	1.248
16	5.163952	1.043

Number of nodes = 17, Cost of any edge <= 1000

(Sequential code on single processor took 19.281 seconds)

Number of Processors	Execution Time	Speedup
2	19.650990	0.981
4	16.296858	1.183
8	14.256837	1.352
16	18.349241	1.050

Conclusion:
The time taken by 2 processors was slower than the single one as communication cost added overheads. Afterwards the scaling was almost linear till 4 processors, then sub-linear till 8 processors as the speed of interconnect would be the determining factor in this case. Since, a total of 8 processors were available on these 2 computers, the time taken by calling 16 processors increased as almost half of the processors remained idle for most of the time. Also, the master node never did the work of solving the task as just did the work of communication, hence the scale up was also less.