Report for HomeWork 2 ME766 Spring 16

Name: Aniruddh Rao K
Roll Number: 133079005

1) Matrix multiplication of A x B of size NxN serially and using OpenMP.

PFA code files named matrix_mul.c, matrix_mul_transpose.c and run_OpenMP.sh scripts for compiling and running matrix multiplication in single thread mode and with mutli thread mode using openmp over values of N = 100, 500, 1000, 5000, 10000
N is taken as argument with code
Machine used:
Core i3 processor with 4 cores, RAM 4GB, OS is UBUNTU 14.04 64 bit.
Number of threads is set to 4 by varying the environment variable OMP_NUM_THREADS=4.

Observations recorded in the following table:

Different methods of execution	N=100	N=500	N=1000	N=5000	N=10000
Time for Serial (Without Transpose)	0.003245s	0.448238s	5.332717s	1049.601336s	6800.393809s
Time for OpenMP (without Transpose)	0.003363s	0.262160s	3.796187s	634.721759s	3475.614530s
Time for Serial (with transpose of B)	0.003180s	0.379256s	3.053564s	380.804536s	3044.448694s
Time for OpenMP (With transpose of B)	0.003349s	0.243592s	1.953752s	245.237610s	1933.732142s

Inference:
a) There is a huge improvement when changing our approach of multiplication by using B transpose instead of B. That is due to bringing down the number of cache misses or increasing cache hits by taking an entire column of B into cache at a time. **Better idea of memory allocation and fetching gives better results.**

b) The matrix multiplication is of order N^3, making it much time taking process as N increases. Multi process and Multi threading can bring down the computation time and make the process faster.

c) By making code run in parallel threads we see that for small values of N, multiple threads may not be advantageous as thread creation and destruction overheads eatup the time. But for large values of N, this gives a improvement in performance by 100% or more with 4 parallel threads.

2) Matrix multiplication of A x B of size NxN using mpi

PFA code files named matrix_mul_mpi.c and run_mpi.sh scripts for compiling and running matrix multiplication using mpi over values of N = 100, 500, 1000, 5000, 10000
N is taken as argument with code
Machine used: Only one machine is used.
Core i5processor with 4 cores, RAM 8GB, OS is UBUNTU 14.04 32 bit.

Observations recorded in the following table:

Number of process	N=100	N=500	N=1000	N=5000	N=10000
2	0.006495s	0.220855s	1.700768s	235.644062s	2024.028873s
4	0.008895s	0.205206s	1.578896s	229.398116s	1855.126280s
8	0.009671s	0.247197s	1.783996s	233.621187s	1881.558487s

Inference:

a) Multi process parallel execution improves the performance very much compared to serial execution. The processing times are relatively similar to multiple threads execution.

b) The overheads of communication can be see in small values of N, but as N increases we see that higher number of process helps bring down the execution time.

c) Since it was a 4 core machine, 8 process execution made the execution a bit slower compared to 4 process execution.