Report - ME766 - HW1

Nahit Pawar - 113079008 - EE

March 25, 2014

0.1 System Specification

- Intel Core i7 - 3610QM, 2.3 GHz
- RAM - 8GB
- Operating System - Linux (Ubuntu 12.04)
- HDD - 1TB

0.2 Serial, OpenMP and MPI version of matrix multiplication code

Note: No Optimization flag is used in any version. See makefile for more details about type of flags used during compilation.

0.2.1 Serial Code

Following table shows the time taken to execute serial matrix code for matrix size $N = 100, 500, 1000, 2000$.

	Time(Sec.)
N=100	.01
N=500	.76
N=1000	14.03
N=2000	102.55

0.2.2 OpenMP

Following table shows time taken to execute OpenMP version of code for matrix size $N = 100, 500, 1000, 2000$ for each thread count num threads $= 2, 4, 8, 16$. I used the same serial code with one pragma at the outer loop. omp_get_wtime() is used to profile the code.

	Num Threads $\rightarrow$ Time (sec.) $\downarrow$	2	4	8	16
N=100		.006	.013	.024	.008
N=500		.484	.277	.252	.259
N=1000		5.375	2.698	2.424	2.413
N=2000		46.76	30.803	20.98	21.28

0.2.3 MPI

In MPI code the **B** matrix is transferred to all processes. While **A** matrix is partitioned into slices of rows, where the width of each slice is given by (Num. of rows) $\div$ (Num. of processes). Broadcast is used to transfer **B** matrix to all processes. MPI_Wtime() is used to profile the code.
Following table shows the time taken to execute MPI version of code for matrix size $N = 100, 500, 1000, 2000$ for each process count np $= 2, 4, 8, 16$.

	Num Threads → Time (sec.) ↓	2	4	8	16
N=100		.005	.007	.006	.022
N=500		.422	.290	.387	.504
N=1000		4.86	2.60	2.933	2.840
N=2000		46.98	24.20	28.299	26.909