I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honor violations by other students if I become aware of it.

Name:Sai Varshittha Ponnam - CS18BTECH11035

Date: 22-01-2020

Signature: Varshittha.

Time calculation: I used the gettimeofday function defined in " sys/time.h" header file to calculate the time taken by each process.

Interactive mode and non-interactive mode:

In the interactive mode, the global variable "interactive" is set to 1.We use this global variable to call input_matrix() function for interactive mode and init_matrix() function for non-interactive mode using suitable if-else statements.

Single-threaded matrix multiplication:

The function single_thread_mm() performs single-thread matrix multiplication.

About the function: - single_thread_mm() function doesn't take any arguments and returns the time taken in microseconds by single thread matrix multiplication as unsigned long long int type.This function prints a resultant matrix for interactive mode.

Multi-threaded matrix multiplication:

The function multi_thread_mm() performs multi-thread matrix multiplication.

About the function: The function multi_thread_mm() doesn't take any arguments and returns the time taken in microseconds by the multi-thread matrix multiplication as unsigned long long int type.

This function calls a function multi_mul() for each thread and the actual computation for matrix multiplication is done in this function. In this function, the thread number is sent as an argument as a void* which is then typecast to int and evaluation of matrix multiplication is done.

If there are ‘T’ threads, I divided the number of rows to ‘T’ threads such that each thread ‘i’ performs matrix multiplication for the rows starting from row index x= (i*arows)/T till the index x+(arows/T).

To ensure that x and x+(arows/T) are less than arows, I included the same condition in the for loop so that we will not get a segmentation fault and all the cells of the resultant matrix are evaluated.

Multi-process matrix multiplication:

The function multi_process_mm() performs multi-process matrix multiplication.

About the function: The function multi_process_mm() doesn’t take any arguments and returns the time

taken in microseconds by the multi-process matrix multiplication as unsigned long long int type.

In this function, I used shared memory to allocate the matrices A, B, C, and 2 processes to do matrix multiplication such that each process does half of the entire process. The child process will do matrix multiplication for the first half number of rows and the parent process does for the remaining half.

About speedup:

Muli-thread :

For 3 threads , The speedup is less for small values of rows and cols and increases if number of rows and cols are increased.

For 4 threads, the average speed up is greater than that of 3 threads.

For 5 threads,the average speedup is little less that that with 4 threads.

My system has 4 cores and two threads per core, so the speed up is comparitively high for 8 threads.

Multi-process:

With two processes, for higher values of rows and columns, the speedup is 1.5x on an average. However, the speedup insignificant for very small values of rows and columns (less than 10) but is significant and the speed up increases for higher values of rows and columns.

Some observations :

- The speed up is different for same values of rows and columns run many times.However, they are close to eachother
- The multi process speed up first increases with increase in number of rows and columns and then almost remains be in the range of 1.6-1.9.

- With very high values of rows and columns, even multi-process takes more time.

Range of values of rows and columns (all are comparable i.e., the difference is not too high)	**Average speedup**
0-15	< 0.1
30-40	0.5 -1
42-60	1-1.5
80 and above	1.5 -2
~ 1000	1.67

Sample output for non interactive for arows=acols=brows=bcols=30

Time taken for single threaded: 283 us

Time taken for multi process: 484 us

Time taken for multi threaded: 1110 us

Speedup for multi process : 0.58 x

Speedup for multi threaded : 0.25 x

Sample output for non interactive for arows=acols=brows=bcols=75

Time taken for single threaded: 3952 us

Time taken for multi process: 1447 us

Time taken for multi threaded: 1076 us

Speedup for multi process : 2.73 x

Speedup for multi threaded : 3.67 x