REPORT
ASSIGNMENT 1

PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that 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 honour violations by other students if I become aware of it.

Name: P.Chaitanya Janakie
Date: 22-01-2020
Signature: P.Chaitanya Janakie

MATRIX MULTIPLICATION USING SINGLE THREAD:

- The function ‘single_thread_mm()’ computes the matrix multiplication and returns the time taken for the process.
- I Implemented this normally by using the loops, similar to normal matrix multiplication.

MATRIX MULTIPLICATION USING MULTIPLE THREADS:

- The function ‘multi_thread_mm()’ creates the threads and calls the function ‘matmul()’ to compute the resultant matrix and also waits for all threads to terminate and returns the time taken for the task.
- I Implemented this using 4 threads, to compute the elements of C in the fastest manner (optimizing).
- Here each thread computes ¼ th part of resultant array C for which I wrote a function ‘matmul()’.

MATRIX MULTIPLICATION USING MULTIPLE PROCESSES:

- The function ‘multi_process_mm()’ creates the child processes using fork(); and waits for all processes to finish their work and terminate, and also returns the time taken for the task.
- I Implemented this by creating ‘n’ processes, where n = no.of rows in C (crows) by using fork(); to compute the elements of C in the fastest manner (for optimization).
- Here each process computes each row of the resultant matrix C.

SAMPLE OUTPUT (INTERACTIVE):

Enter A:
2 3 4
4 3 4
5 3 7
Enter B:
1 0 0
0 1 0
0 0 1
Result:
2 3 4
4 3 4
5 3 7
Enter A:
2 3 4
4 3 4
5 3 7
Enter B:
1 0 0
0 1 0
0 0 1
Result:
2 3 4
4 3 4
5 3 7
Enter A:
2 3 4
4 3 4
5 3 7
Enter B:
1 0 0
0 1 0
0 0 1

Result:
2 3 4
4 3 4
5 3 7
Time taken for single threaded: 3 us
Time taken for multi process: 1300 us
Time taken for multi threaded: 2706 us
Speedup for multi process : 0.00 x
Speedup for multi threaded : 0.00 x

SAMPLE OUTPUT (NON INTERACTIVE):

Time taken for single threaded: 1 us
Time taken for multi process: 339 us
Time taken for multi threaded: 316 us
Speedup for multi process : 0.00 x
Speedup for multi threaded : 0.00 x

OBSERVATIONS:

- For small inputs i.e; matrix sizes of order (50x50), time taken for multi process and multi threaded are relatively more than single threaded process.
- For large inputs i.e; matrix sizes of order (100x100) and above , the time taken foe multi process and multi threaded are lesser than single threaded process.
- Therefore for large inputs speed up is >0 for both multi process and multi threaded.
- When compared between multi threaded and multi process , time taken for multi threaded is lower than time taken foe multi process.
- Therefor speedup for multi threaded is more than multi process.

SPEEDUP’s FOR SMALLER INPUTS:

Speedup for multi process : 0.02 x
Speedup for multi threaded : 0.31 x

SPEEDUP’s FOR LARGER INPUTS:

Speedup for multi process : 1.43 x
Speedup for multi threaded : 1.94 x