High Performance Computing Report

Roshani (201601059)
Nishi Doshi (201601408)

Dhirubhai Ambani Institute of Information and Communication technology
201601059@daiict.ac.in
201601408@daiict.ac.in

Question 1 : Calculate the value of π using the Trapezoidal Method(Critical)

1 Implementation Details

1(a) Brief and clear description about the Serial implementation

The serial code is written for calculating value of π by using trapezoidal method serially by running the loop for entire problem size.

1(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

In parallel approach we have data decomposed the problem size as per the number of processors, value of π is calculated. And finally the calculations on each processor are critically added to the global variable π.

2 Complexity and Analysis Related

2(a) Complexity of serial code

The loop runs n times and hence O(n) is the time complexity for serial code where n is problem size.

2(b) Complexity of parallel code (split as needed into work, step, etc.)

The loop runs $\frac{n}{p}$ times on each processor and hence O($\frac{n}{p}$) where n is problem size and p is number of processors.

2(c) Cost of Parallel Algorithm

Theoretically, speedup is calculated as ratio of time taken to execute the code serially to time taken to execute code parallely.

2(d) Theoretical Speedup (using asymptotic analysis, etc.)

Hence theoretical speed ups are as follows :
For 1 processor use $\frac{n}{(n/1)} = 1$ is theoretical speedup.
For 2 processor use $\frac{n}{(n/2)} = 2$ is theoretical speedup.
For 3 processor use $\frac{n}{(n/3)} = 3$ is theoretical speedup.
For 4 processor use $\frac{n}{(n/4)} = 4$ is theoretical speedup.

3 Curve Based Analysis

3(a) Time Curve related analysis (as no. of processor increases)

It is observed that as the number of processors on which the code runs increases the time taken for executing or calculating the result reduces. Therefore, the time for executing the code is inversely proportional to the number of processors on which the code is working.

3(b) Speedup Curve related analysis (as problem size and no. of processors increase)

The speed up increases as the number of processes increases on which the code runs. Also it is noted that the code works faster in case of parallelism as compared to the serial code. This is observed as the parallel code runs parallely on the four cores instead of the single core in case of a serial code. As the problem size increases initially the parallel threads have a speedup less than one due to decreased latency i.e. the amount of time it takes to load the data but however as N increases the throughput = $\frac{N}{N+m-1}$ where N = Total independent operations and m = Depth of the pipeline so in the large case where $N >> m$ Throughput approaches 1 in the parallel case so the speed increases .
It is observed that the speedup initially remains below 1. However, in the end the speedup of 4 threads is the highest and it first increases and then decreases to around 3.5 as some time may be spent on memory accesses for such a large problem size. When 3 processors are used we observe that speedup obtained is around 2.5 and for 2 processors speedup is a around 2. The deviation in these values from the theoretical values depicts that for large problem size the memory access time also comes into consideration. It is also noted that for some time the speedup of 1 thread is greater than speedup of 2 threads. This may be because we need to consider the memory accesses as well.

3(c) Efficiency Curve related analysis

It is also observed that with greater number of processors working together the efficiency also decreases. The greater number of threads lesser will be the efficiency. Thus, this trend is obtained.

3(d) Number of cores and Speedup relation

As is observed from the graph that as the number of cores increases the speedup increases as work is distributed among the processes and work is achieved efficiently.

4 Graphs

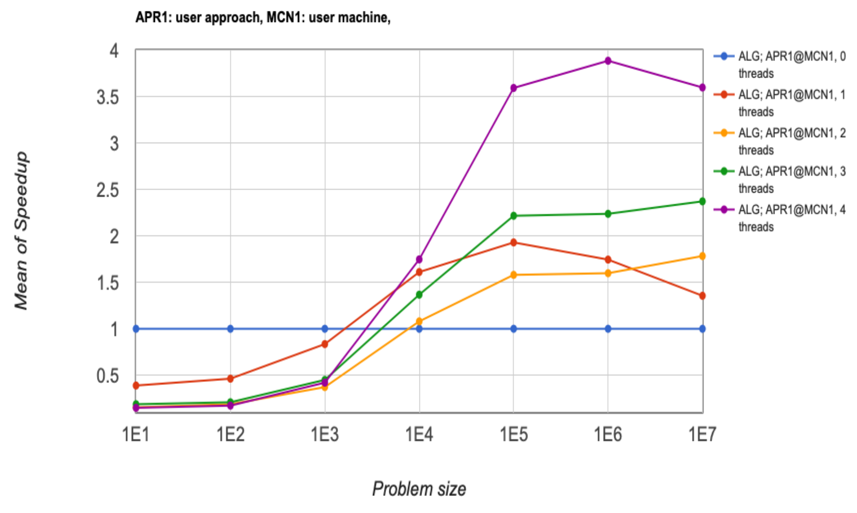

Fig. 1. Mean of Speedup

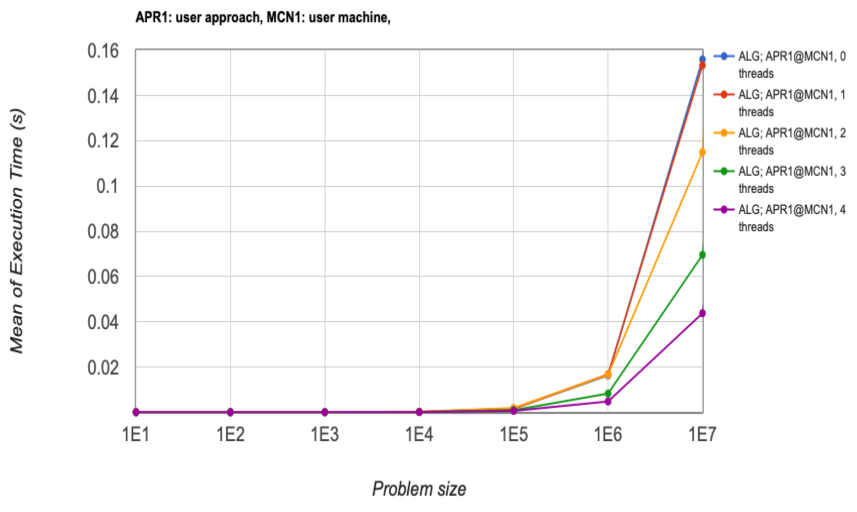

Fig. 2. Mean of Execution Time

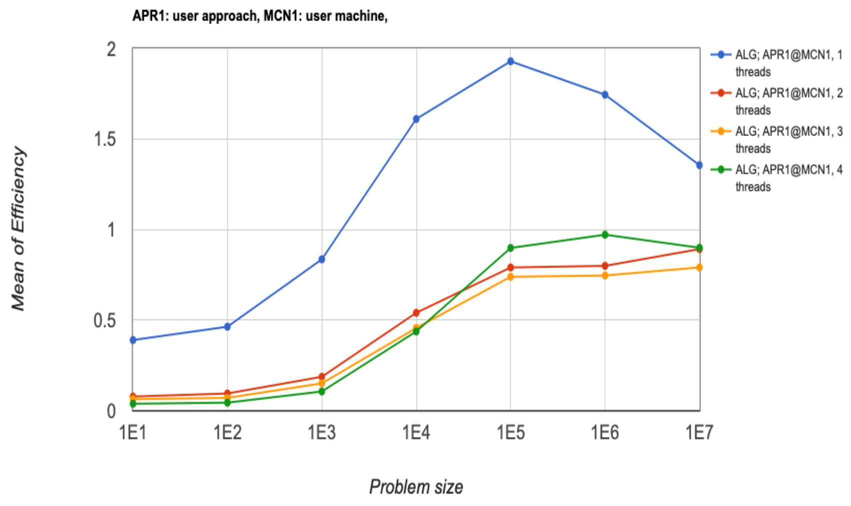

Fig. 3. Mean of Efficiency

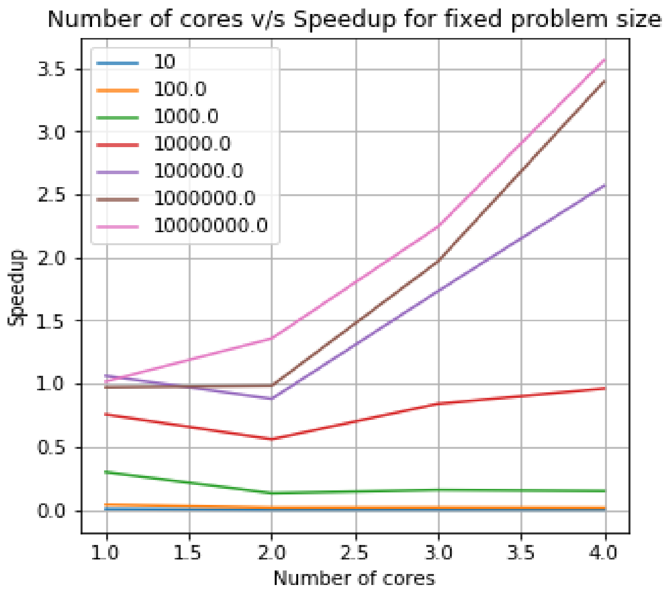

Fig. 4. Number of cores v/s speedup

Calculate the value of π using the Trapezoidal Method(Private)

5 Implementation Details

5(a) Brief and clear description about the Serial implementation

The serial code is written for calculating value of π by using trapezoidal method serially by running the loop for entire problem size.

5(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

In parallel approach we have data decomposed the problem size as per the number of processors, value of π is calculated. A global array is maintained which stores the values calculated on each processor and finally these values in global array are summed to find the actual value of π.

6 Complexity and Analysis Related

6(a) Complexity of serial code

The loop runs n times and hence O(n) is the time complexity for serial code where n is problem size.

6(b) Complexity of parallel code (split as needed into work, step, etc.)

Here, The global array is initialized to zero hence time taken for that is equivalent to size of array that is number of processors that is O(p). The loop runs $\frac{n}{p}$ time on each processor and hence has complexity O($\frac{n}{p}$). The final summation of pi is calculated by adding the global array values which taked additional O(p) time Time taken to O(p + $\frac{n}{p}$ + p) where n is problem size and p is number of processors. As $p <<< n$ we can neglect it and time complexity of problem can be given by O($\frac{n}{p}$).

6(c) Theoretical Speedup (using asymptotic analysis, etc.)

Hence theoretical speed ups are as follows :
For 1 processor use $\frac{n}{(n/1)}$ = 1 is theoretical speedup.
For 2 processor use $\frac{n}{(n/2)}$ = 2 is theoretical speedup.
For 3 processor use $\frac{n}{(n/3)}$ = 3 is theoretical speedup.
For 4 processor use $\frac{n}{(n/4)}$ = 4 is theoretical speedup.

7 Curve Based Analysis

7(a) Time Curve related analysis (as no. of processor increases)

The execution time graph shows that as the processors increased the time taken to execute the code for larger problem size (greater than 1e5) decreased due to division of computations on more processors as opposed to single processor.

7(b) Speedup Curve related analysis (as problem size and no. of processors increase)

It is observed that the speedup initially remains below 1. However, in the end the speedup of 4 threads is the highest and it remains around 3.5 as some time may be spent on memory accesses for such a large problem size. When 3 processors are used we observe that speedup obtained is around 3 and for 2 processors speedup is a around 2. The deviation in these values from the theoretical values depicts that for large problem size the memory access time also comes into consideration.

8 Graphs

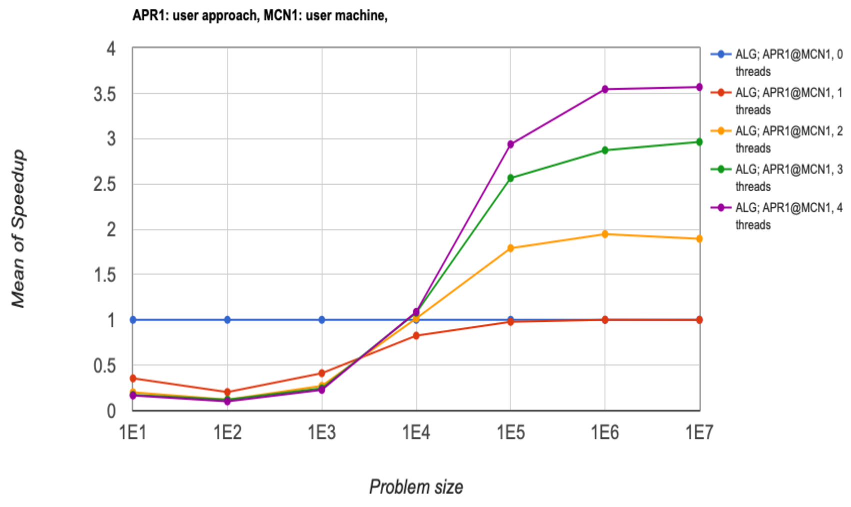

Fig. 5. Mean of Speedup

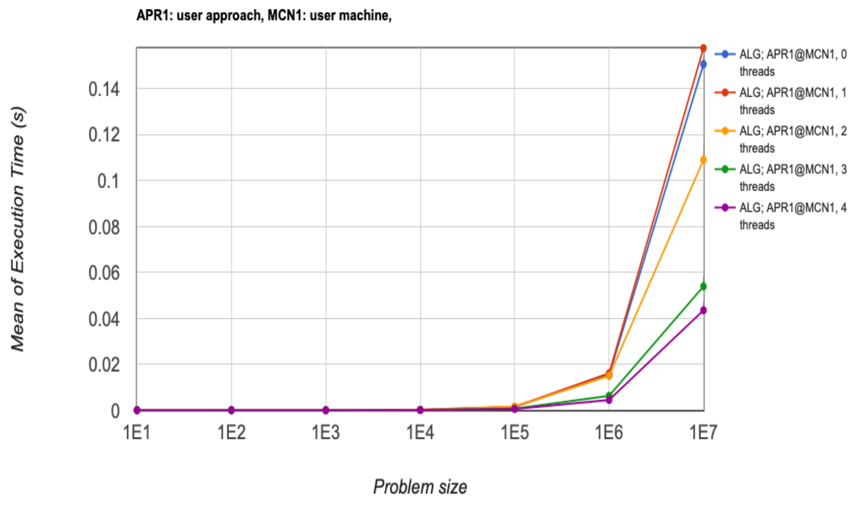

Fig. 6. Mean of Execution Time

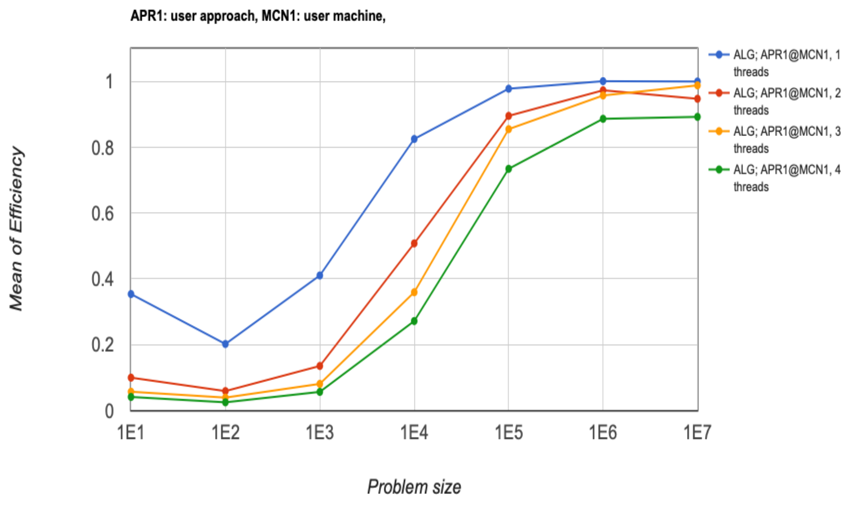

Fig. 7. Mean of Efficiency

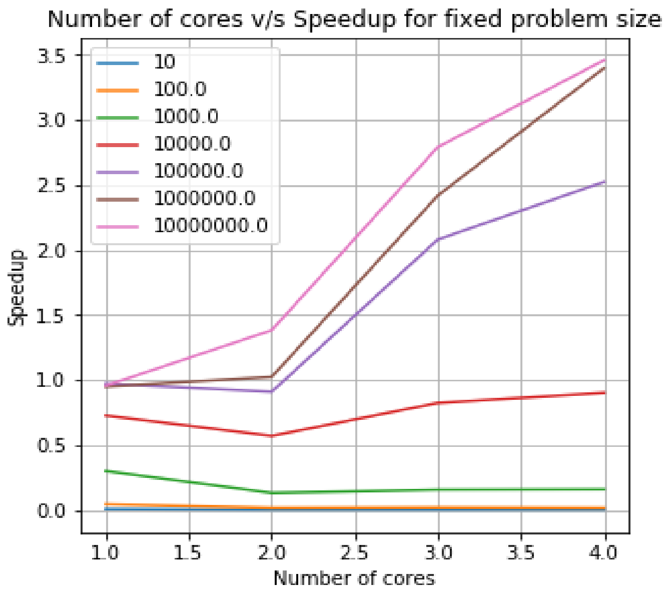

Fig. 8. Number of cores v/s speedup

Question 2 : Calculate the value of π using Series

9 Implementation Details

9(a) Brief and clear description about the Serial implementation

In the serial implementation we simply integrated the function over the interval of 0 to 1.

9(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

The series was divided as per the number of processors and sum was calculated for the respective ranges on different processors. And finally the calculations on each processor are critically added to the global variable π.

10 Complexity and Analysis Related

10(a) Complexity of serial code

The loop runs n times and hence O(n) is the time complexity for serial code where n is problem size.

10(b) Complexity of parallel code (split as needed into work, step, etc.)

The loop runs n/p times on each processor and hence O(n/p) where n is problem size and p is number of processors.

10(c) Theoretical Speedup (using asymptotic analysis, etc.)

Hence theoretical speed ups are as follows :
For 1 processor use $\frac{n}{(n/1)} = 1$ is theoretical speedup.
For 2 processor use $\frac{n}{(n/2)} = 2$ is theoretical speedup.
For 3 processor use $\frac{n}{(n/3)} = 3$ is theoretical speedup.
For 4 processor use $\frac{n}{(n/4)} = 4$ is theoretical speedup.

11 Curve Based Analysis

11(a) Time Curve related analysis (as no. of processor increases)

The execution time graph shows that as the processors increased the time taken to execute the code for larger problem size (greater than 1e5) decreased due to division of computations on more processors as opposed to single processor.

11(b) Speedup Curve related analysis (as problem size and no. of processors increase)

It is observed that the speedup initially remains below 1. However, in the end the speedup of 4 threads is the highest and it remains below 3.5 as some time may be spent on memory accesses for such a large problem size. When 3 processors are used we observe that speedup obtained is around 2 and for 2 processors speedup is a little greater than 1.5. The deviation in these values from the theoretical values depicts that for large problem size the memory access time also comes into consideration.

11(c) Efficiency Curve related analysis

It is also observed that with greater number of processors working together the efficiency also decreases. The greater number of threads lesser will be the efficiency. Thus, this trend is obtained.

11(d) Number of cores and Speedup relation

As is observed from the graph that as the number of cores increases the speedup increases as work is distributed among the processes and work is achieved efficiently.

12 Graphs

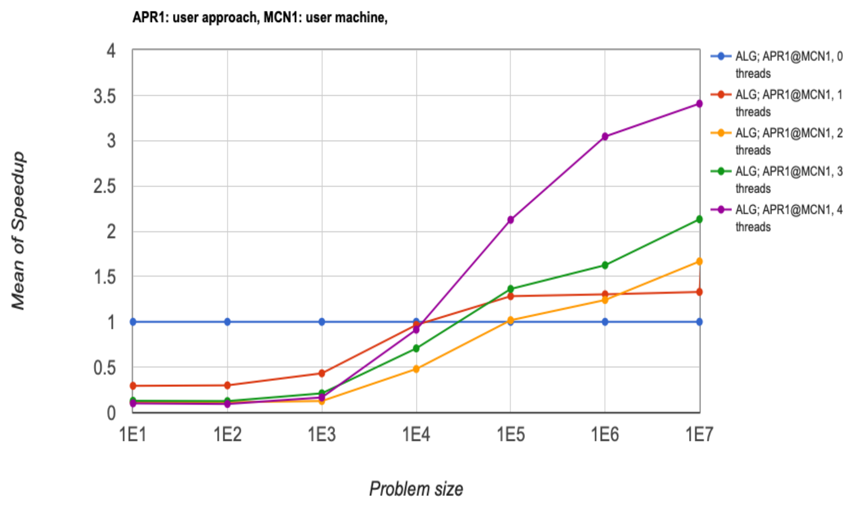

Fig. 9. Mean of Speedup

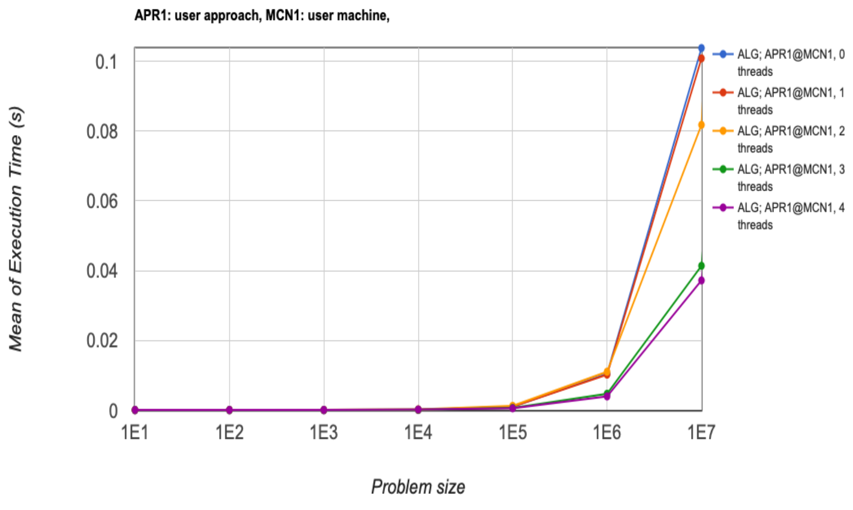

Fig. 10. Mean of Execution Time

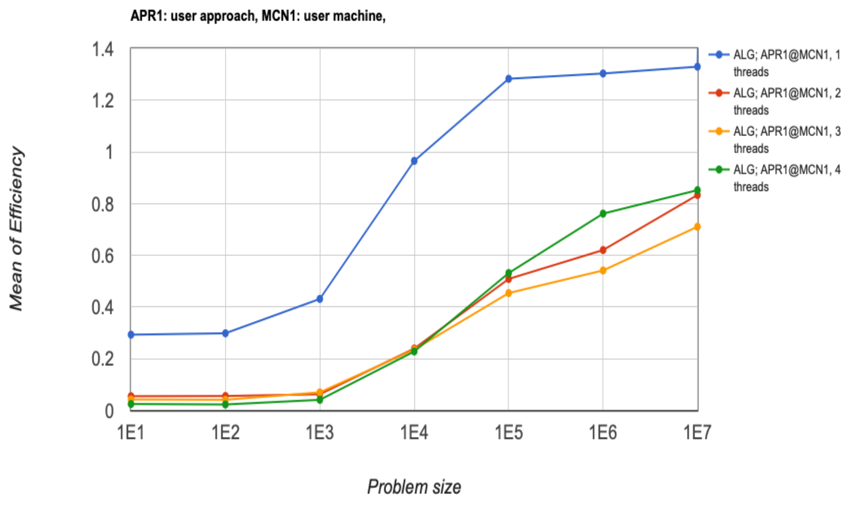

Fig. 11. Mean of Efficiency

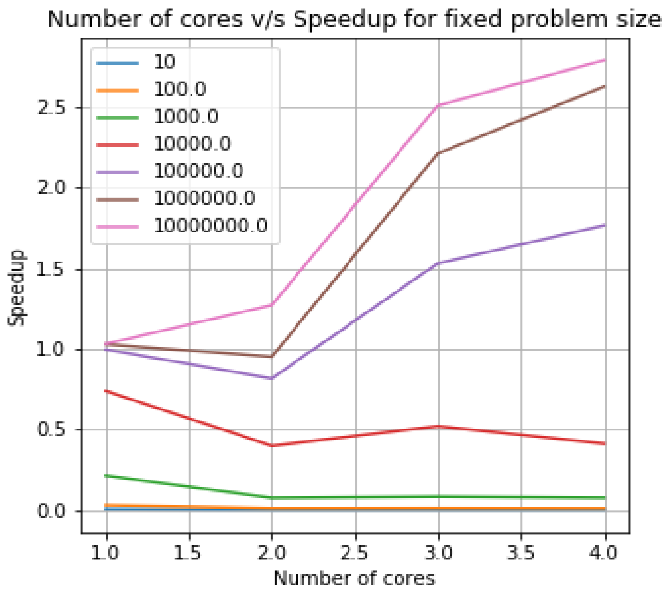

Fig. 12. Number of cores v/s speedup

Question 3 : Vector addition

13 Implementation Details

13(a) Brief and clear description about the Serial implementation

Initialization of both input arrays is done with 1. In case of serial implementation of vector addition we have just run a loop for the number of vectors present and summed over the components.

13(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

Initialization of both input arrays is done with 1. Data decomposition of original array is done and addition of elements as per division is done on individual processor and finally they are stored in corresponding place in the given array.

14 Complexity and Analysis Related

14(a) Complexity of serial code

The loop runs n times and hence O(n) is the time complexity for serial code where n is problem size.

14(b) Complexity of parallel code (split as needed into work, step, etc.)

The loop runs n/p times on each processor and hence O(n/p) where n is problem size and p is number of processors.

14(c) Theoretical Speedup (using asymptotic analysis, etc.)

Hence theoretical speed ups are as follows :
For 1 processor use $\frac{n}{(n/1)} = 1$ is theoretical speedup.
For 2 processor use $\frac{n}{(n/2)} = 2$ is theoretical speedup.
For 3 processor use $\frac{n}{(n/3)} = 3$ is theoretical speedup.
For 4 processor use $\frac{n}{(n/4)} = 4$ is theoretical speedup.

15 Curve Based Analysis

15(a) Time Curve related analysis (as no. of processor increases)

The execution time graph shows that as the processors increased the time taken to execute the code for larger problem size (greater than $1e5$) decreased due to division of computations on more processors as opposed to single processor.

15(b) Speedup Curve related analysis (as problem size and no. of processors increase)

It is observed that the speedup initially remains below 1. However, in the end the speedup of 4 threads is the highest and it remains below 4 as some time may be spent on memory accesses. There is an unusual trend observed in use of 1 processor where initially speed up remains quite high compared to use of 2,3,4 processors.

15(c) Efficiency Curve related analysis

It is also observed that with greater number of processors working together the efficiency also decreases. The greater number of threads lesser will be the efficiency. Thus, this trend is obtained.

15(d) Number of cores and Speedup relation

As is observed from the graph that as the number of cores increases the speedup increases as work is distributed among the processes and work is achieved efficiently.

16 Graphs

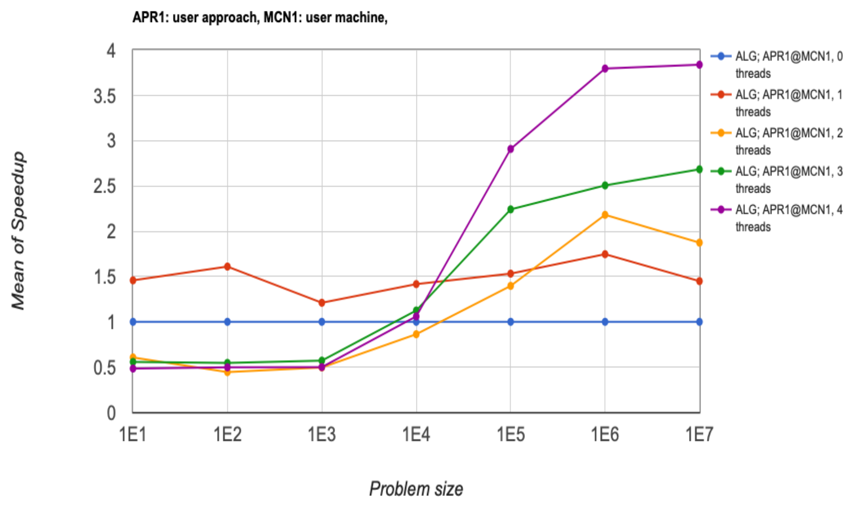

Fig. 13. Mean of Speedup

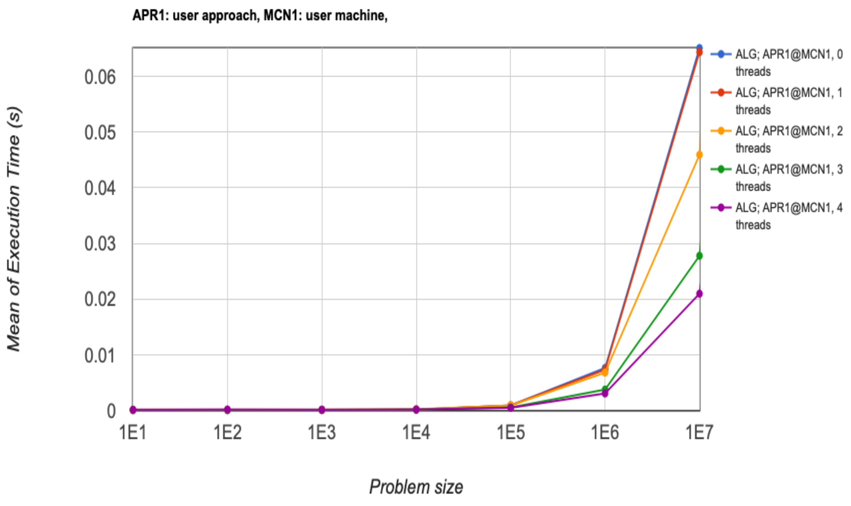

Fig. 14. Mean of Execution Time

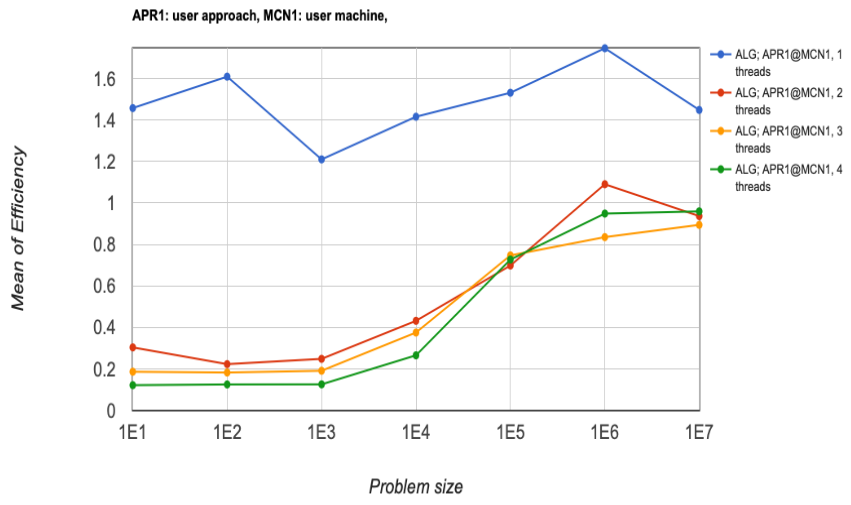

Fig. 15. Mean of Efficiency

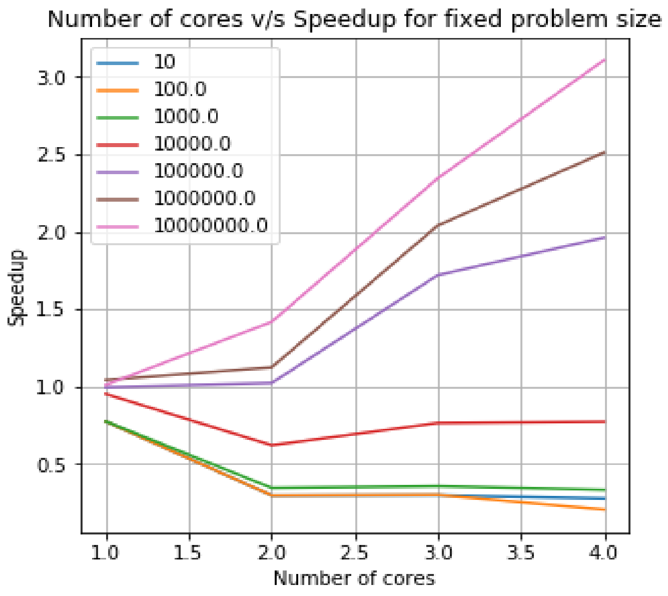

Fig. 16. Number of cores v/s speedup

Question 4 : Vector Product

17 Implementation Details

17(a) Brief and clear description about the Serial implementation

Initialization of both input arrays is done with 1. A variable sum is maintained which is accessed to sum the values of A[i]*B[i].

17(b) Brief and clear description about the implementation of the approach (Parallelization Strategy, Mapping of computation to threads)

Initialization of both input arrays is done with 1 just before the computation is done. A global array of size of number of processors was intialized and computations (multiplication of ith element in one vector with ith element in another vector) were decomposed on input data as per the use of processors. For every processor the sum was calculated and stored on local variable and then that local variable is copied to global array and summation of global array is done.

18 Complexity and Analysis Related

18(a) Complexity of serial code

The loop runs n times and hence O(n) is the time complexity for serial code where n is problem size.

18(b) Complexity of parallel code (split as needed into work, step, etc.)

The loop runs n/p times on each processor and global array values are added to the final sum for which the loop runs p times. Hence overall, time taken for computation is n/p + p. But as we know p¡¡n we can neglect p and hence O(n/p) where n is problem size and p is number of processors.

18(c) Theoretical Speedup (using asymptotic analysis, etc.)

Hence theoretical speed ups are as follows :
For 1 processor use $\frac{n}{(n/1)} = 1$ is theoretical speedup.
For 2 processor use $\frac{n}{(n/2)} = 2$ is theoretical speedup.
For 3 processor use $\frac{n}{(n/3)} = 3$ is theoretical speedup.
For 4 processor use $\frac{n}{(n/4)} = 4$ is theoretical speedup.

19 Curve Based Analysis

19(a) Time Curve related analysis (as no. of processor increases)

The execution time graph shows that as the processors increased the time taken to execute the code for larger problem size (greater than 1e5) decreased due to division of computations on more processors as opposed to single processor which showed the use of maximum time to execute the code.

19(b) Speedup Curve related analysis (as problem size and no. of processors increase)

It is observed that the speedup initially remains below 1. However, in the end the speedup of 4 threads is the highest and it remains below 3.4 as some time may be spent on memory accesses for such a large problem size. When 3 processors are used we observe that speedup obtained is around 2.5 and for 2 processors speedup is around 1.5. The deviation in these values from the theoretical values depicts that for large problem size the memory access time also comes into consideration.

20 Graphs

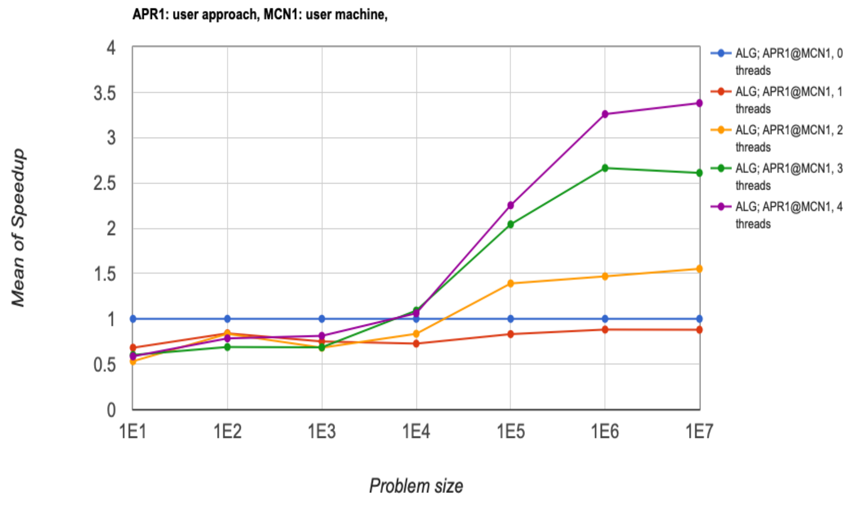

Fig. 17. Mean of Speedup

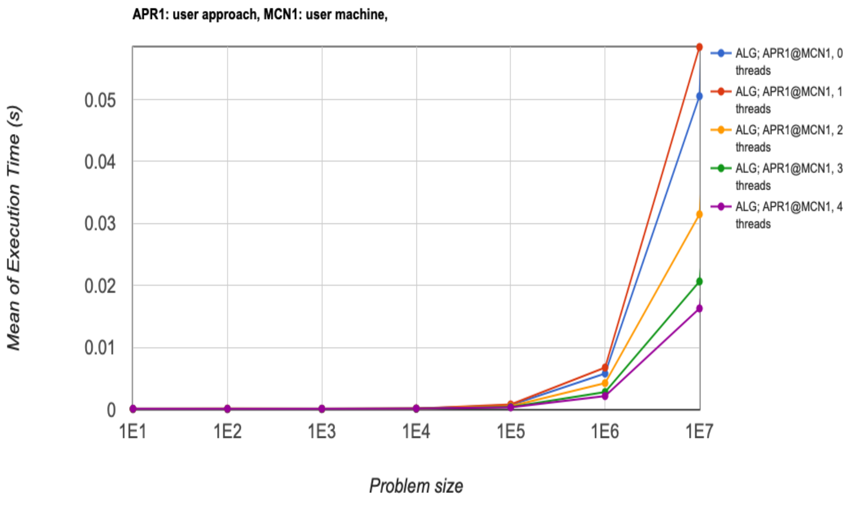

Fig. 18. Mean of Execution Time

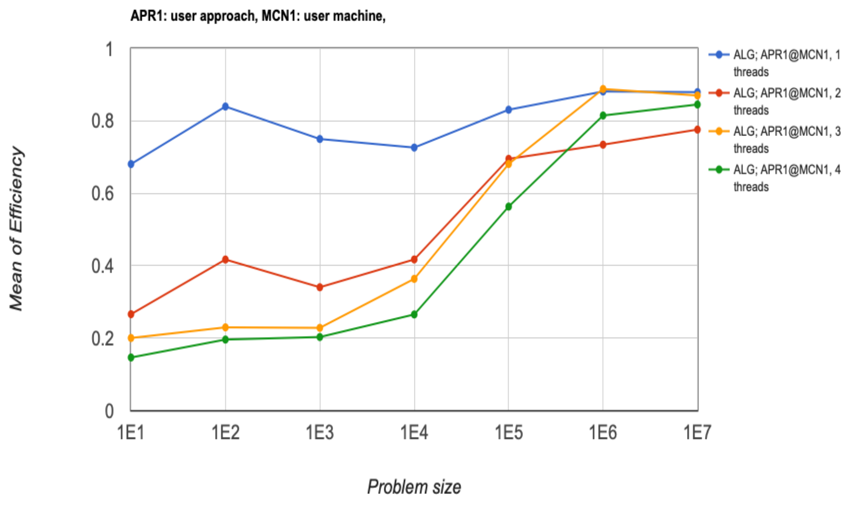

Fig. 19. Mean of Efficiency

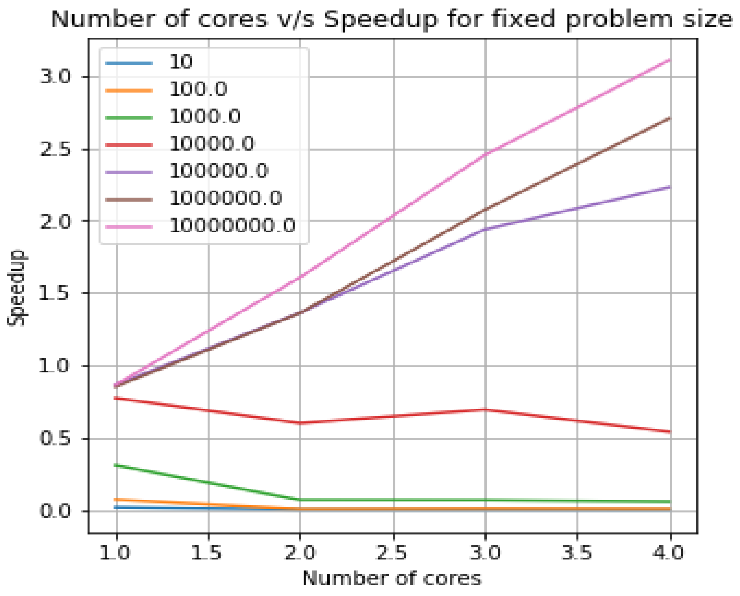

Fig. 20. Number of cores v/s speedup