High Performance Computing Report

Author 1 (Kunal Jani)
Author 2 (Pratik Ghosh)

Dhirubhai Ambani Institute of Information and Communication Technology
201601444@daiict.ac.in
201721010@daiict.ac.in

Question 1:

Implementation Details

Brief and clear description about the Serial implementation

An image can be represented in the form of a 2 dimensional array of pixels in memory. Each and every pixel in the image stores the value from 0 to 255 of the red, green and the blue components. If the red, green and blue components have a value 0, then the lightest shade of blue, green or red is displayed. If the red, green and blue components have a value of 255 then the darkest shade of red, green and blue are displayed. All possible colours that are displayed on the screen are a mixture of red, green and blue.

In the image warping method, each and every pixel in the image is rotated by a particular angle to give out a twisted image or a warped image.

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

Using the 'pragma omp parallel for' command in the outer block, it is possible to parallelize the outermost for loop for different number of threads and figure out the speedup for different number of threads.

Complexity and Analysis Related

Complexity of serial code

No of operations = n^2
Time taken = n^2
Time complexity = O(n^2)

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

No of operations = n^2
Time taken = n^2

Time complexity = O(n^2)

Cost of Parallel Algorithm

Cost of the algorithm = O(n^2)

Theoretical Speedup (using asymptotic analysis, etc.)

Serial execution time=p
Parallel execution time=1
$Speedup = \frac{Serialtime}{Paralleltime} = p$

Number of memory accesses

Serial algorithm:

No of memory accesses outside loop=18
No of memory accesses inside loop=$146n^2 + 4n$
Total memory accesses=$146n^2 + 4n + 18$

Parallel algorithm:

No of memory accesses outside loop=18
No of memory accesses inside loop=$146n^2 + 4n$
Total memory accesses=$146n^2 + 4n + 18$

Number of computations

Serial algorithm:

No of computations outside loop=5
No of computations inside loop=$2n + 133n^2$
Total no of computations=$133n^2 + 2n + 5$

Parallel algorithm:

No of computations outside loop=5
No of computations inside loop=$2n + 133n^2$
Total no of computations=$133n^2 + 2n + 5$

Curve Based Analysis

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

The speedup will increase for a fixed problem size as the number of processors will increase as a large amount of work will be equally distributed among all the processors.

Time Curve related analysis (as problem size increases, also for serial)

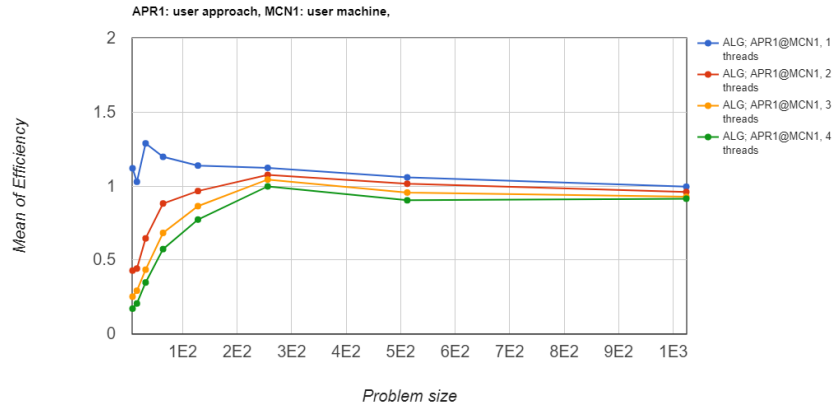
When the image which is to be warped is of a smaller size, the execution time is almost the same because for smaller values, the overhead cost in creating and finishing threads is more dominant while for larger problem sizes, the division of work among threads will be the more dominant factor.

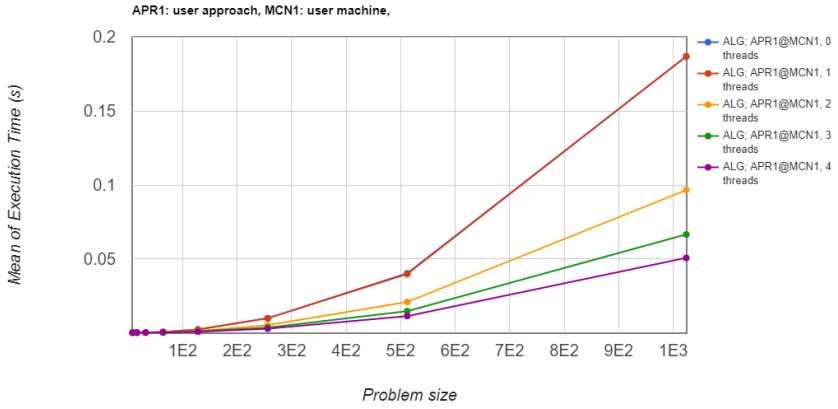
Speedup Curve related analysis (as problem size and no. of processors increase)

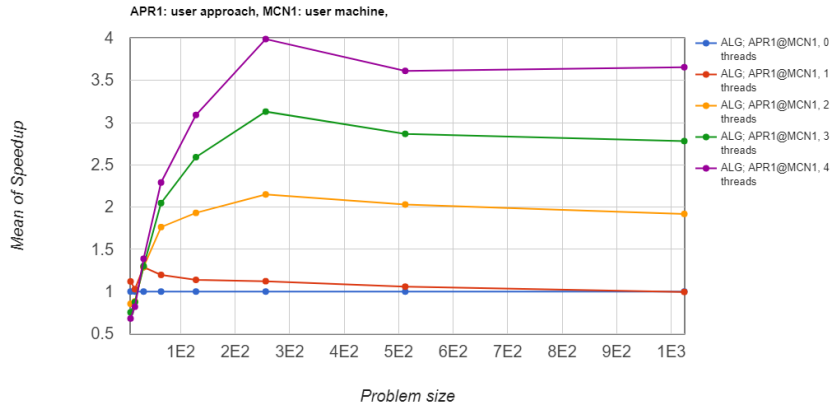
For a low problem size, the speedup will be lower than 1 because the overhead cost in the parallel algorithm dominates over the time reduces in the division of work among threads while in the case of larger problem sizes, the parallel algorithm will give a better speedup.

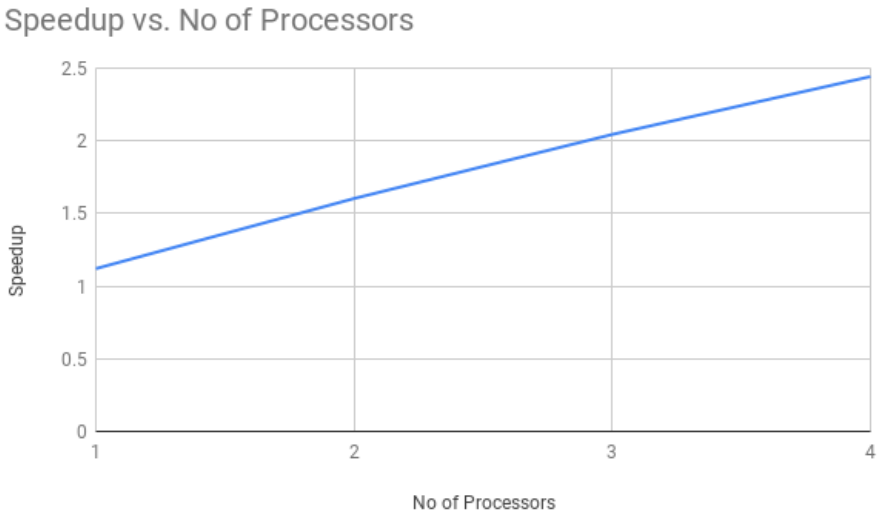
Efficiency Curve related analysis

For a lower problem size, the efficiency is much greater for the serial algorithm compared to the parallel algorithm because the speedup is much greater for a lower number of threads and the speedup to number of processors ratio is much greater in this case. For a higher problem size, efficiency will increase for the parallel algorithm for all the number of threads due to division of work among threads while it will decrease for the serial algorithm because there is no division of work.


APR1: user approach, MCN1: user machine,
Mean of Efficiency
2
1.5
1
0.5
0
1E2
2E2
3E2
4E2
5E2
6E2
7E2
8E2
9E2
1E3
Problem size
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads


APR1: user approach, MCN1: user machine,
Mean of Execution Time (s)
0.2
0.15
0.1
0.05
1E2
2E2
3E2
4E2
5E2
6E2
7E2
8E2
9E2
1E3
Problem size
ALG; APR1@MCN1, 0 threads
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads





Question 2:

Implementation Details

Brief and clear description about the Serial implementation

In the serial implementation of the greyscaling of an image, an image is read in the form of a 2 dimensional array. For each pixel, the average of the red, green and the blue components are taken and the calculated value is stored as the greyscale value which represents a shade of grey from 0 to 255, where 0 represents a completely dark pixel and 255 represents a completely white pixel.

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

Using the 'pragma omp parallel for' command in the outermost for loop, it is possible to divide the work between threads for the calculation of the average values of the pixel.

Complexity and Analysis Related

Complexity of serial code

No of operations = n^2
Time taken = n^2
Time complexity = $O(n^2)$

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

No of operations = n^2
Time taken = n^2
Time complexity = $O(n^2)$

Cost of Parallel Algorithm

The cost of the algorithm is O(n^2).

Theoretical Speedup (using asymptotic analysis, etc.)

Speedup=p/1=p

Number of memory accesses

Serial algorithm:
No of memory accesses outside loop=16
No of memory accesses inside loop=$9n^2 + 4n$
Total memory accesses=$9n^2 + 4n + 16$
Parallel algorithm:
No of memory accesses outside loop=16
No of memory accesses inside loop=$9n^2 + 4n$
Total memory accesses=$9n^2 + 4n + 16$

Number of computations

Serial algorithm:
No of computations outside loop=0

No of computations outside loop=$5n^2 + 2n$
Total no of computations=$5n^2 + 2n$
Parallel algorithm:
No of memory accesses outside loop=0
No of memory accesses inside loop=$5n^2 + 2n$
Total memory accesses=$5n^2 + 2n$

Curve Based Analysis

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

The speedup will increase for a fixed problem size as the number of processors will increase as a large amount of work will be equally distributed among all the processors.

Time Curve related analysis (as problem size increases, also for serial)

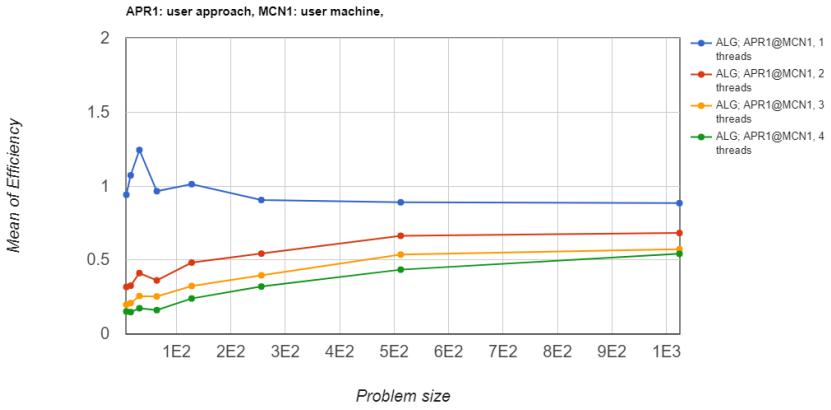
For lower number of threads, the rate of increase of the execution time will be much greater compared to the execution time for higher number of threads because for a greater number of threads, work is distributed on a lager number of processors and thus the execution will be faster.

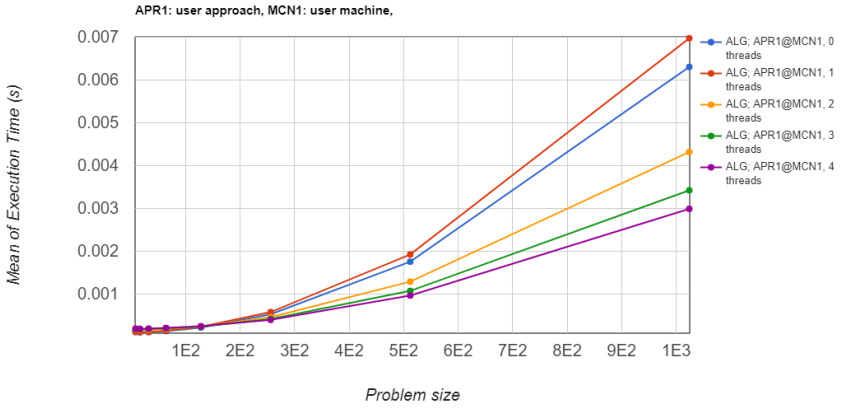
Speedup Curve related analysis (as problem size and no. of processors increase)

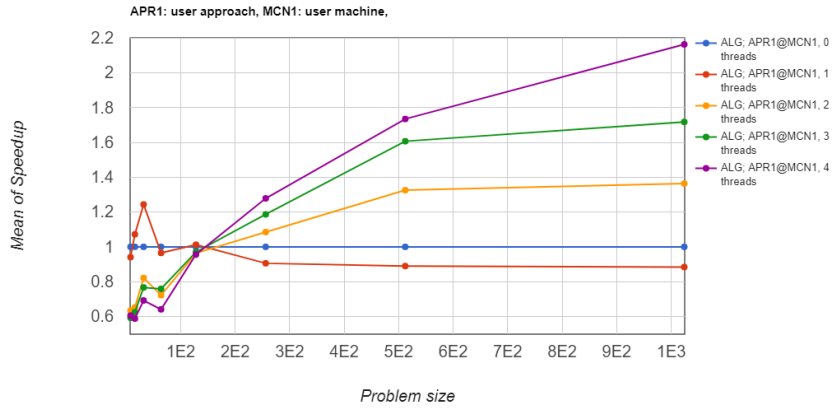
For a smaller process size, the speedup is greater for a smaller number of threads because the number of computations required in a serial algorithm is much greater compared to the number of computations required in a parallel algorithm, but for a larger problem size, the speedup is larger for a parallel algorithm as the large amount of work is divided equally among a larger number of processors.

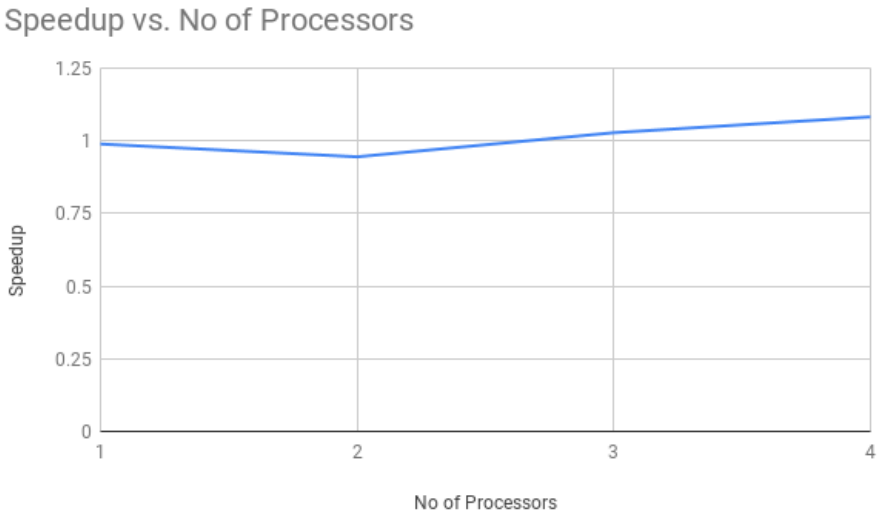
Efficiency Curve related analysis

For small values of the problem size, the efficiency is the highest for the serial algorithm as the speedup due to lower number of operations to be performed is good and the number of processors is only one. For greater values of the problem size, efficiency will be greater for a larger number of processors as the speedup is much greater in this case.


APR1: user approach, MCN1: user machine,
Mean of Efficiency
2
1.5
1
0.5
0
1E2
2E2
3E2
4E2
5E2
6E2
7E2
8E2
9E2
1E3
Problem size
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads


APR1: user approach, MCN1: user machine,
Mean of Execution Time (s)
0.007
0.006
0.005
0.004
0.003
0.002
0.001
1E2
2E2
3E2
4E2
5E2
6E2
7E2
8E2
9E2
1E3
Problem size
ALG; APR1@MCN1, 0 threads
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads





Question 3:

Implementation Details

Brief and clear description about the Serial implementation

the case of blurring the image, all the pixel values have to be sorted in an ascending order to calculate the median of the value of pixel. The median value of the pixel is copied to produce a blurred image.

Brief and clear description about the implementation of the approach

(Parallelization Strategy, Mapping of computation to threads)

The 'pragma omp parallel for' command which is to be used for the purpose of parallel programming is used only to parallelize blurring process.

Complexity and Analysis Related

Complexity of serial code

No of operations = $n^2 + 25$

Time taken = $n^2 + 25$

Time complexity = O(n^2)

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

No of operations = $n^2 + 25$
Time taken = $n^2 + 25$
Time complexity = O(n^2)

Cost of Parallel Algorithm

The cost of the algorithm is O(n^2).
Theoretical Speedup (using asymptotic analysis, etc.) Speedup=p/1=p

Number of memory accesses

Serial algorithm:

No of computations outside loop=16
No of computations outside loop=$9n^2 + 4n$
Total no of computations=$9n^2 + 4n + 16$

Parallel algorithm:

No of memory accesses outside loop=16
No of memory accesses inside loop=$9n^2 + 4n$
Total memory accesses=$9n^2 + 4n + 16$

Number of computations

Serial algorithm:

No of computations outside loop=0
No of computations outside loop=$5n^2 + 2n + 25$
Total no of computations=$5n^2 + 2n + 25$

Parallel algorithm:

No of memory accesses outside loop=0
No of memory accesses inside loop=$5n^2 + 2n + 25$
Total memory accesses=$5n^2 + 2n + 25$

Curve Based Analysis

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

The speedup will decrease and then remain constant for a fixed problem size because with the increase in the execution time, the communication cost will also increase due to the division of work between threads.

Time Curve related analysis (as problem size increases, also for serial)

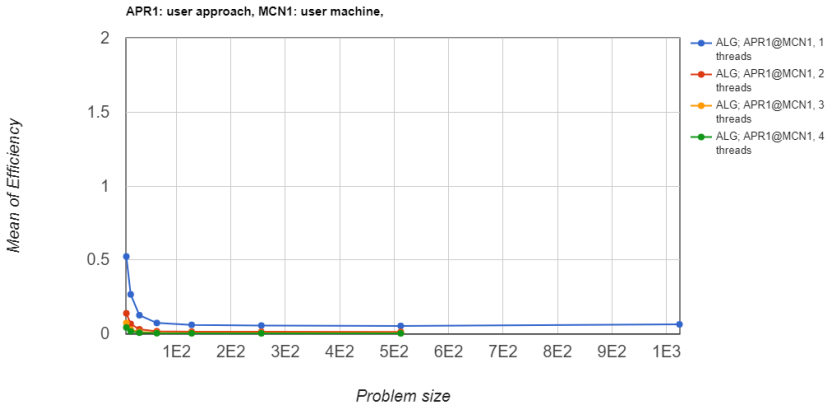
With an increase in the problem size, the execution time will initially not change much because both the effects of higher number of computations due to the implementation of the parallel algorithm and the time reduced due to division of work among processors will cancel out each other, but with a greater problem size, the execution time will increase as a lot of work will have to be done.

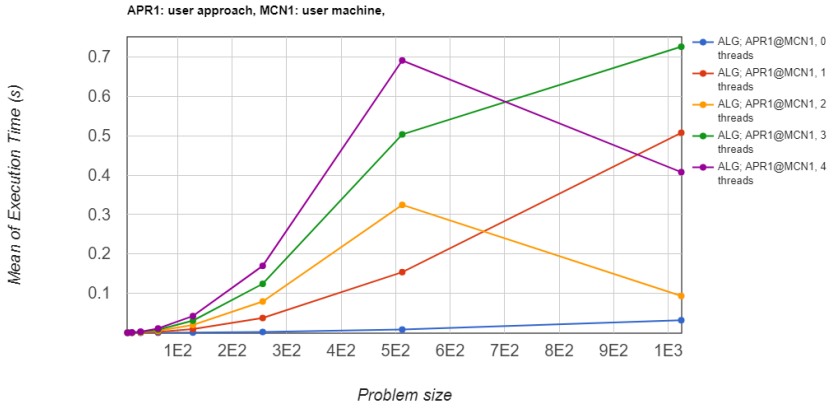
Speedup Curve related analysis (as problem size and no. of processors increase)

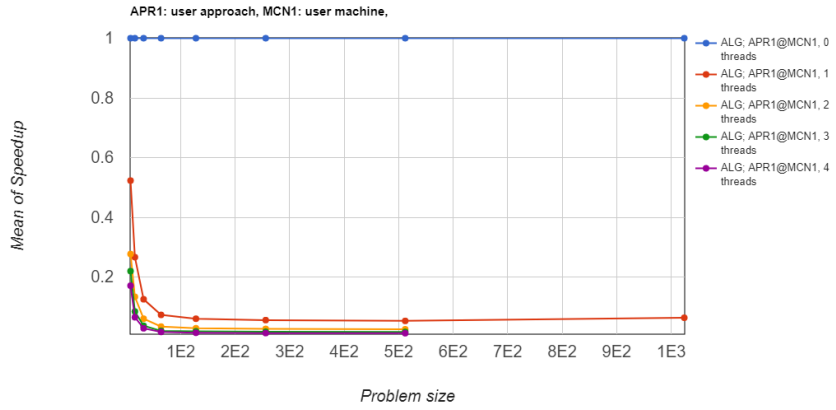
The speedup in this case will not be desirable in the case of the parallel algorithm because the execution time will be greater as the parallelization has only been implemented int the blurring the greyscale image and not in the converting the image from a coloured image to a greyscale image.

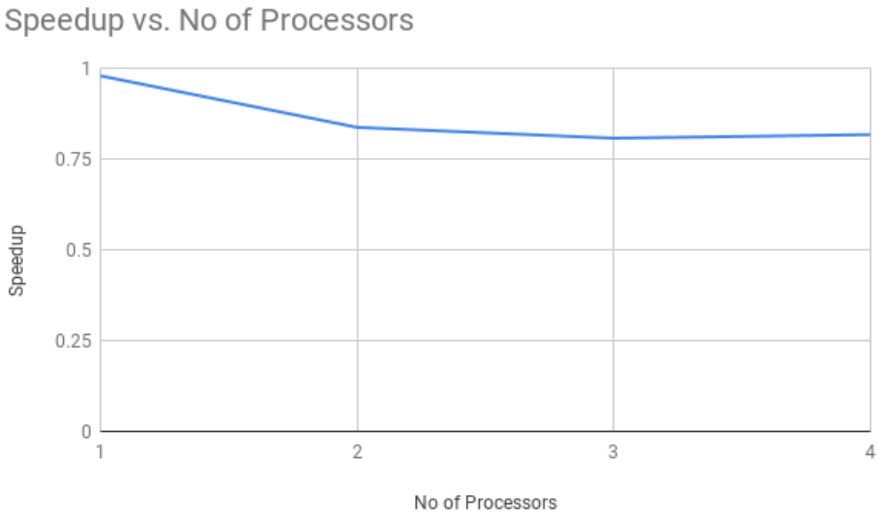
Efficiency Curve related analysis

The efficiency of the parallel algorithm will not be so desirable because of the lower value of the speedup which has been observed, so the speedup to no of processor ratio will be less compared to the serial case.


APR1: user approach, MCN1: user machine,
Mean of Efficiency
2
1.5
1
0.5
0
1E2
2E2
3E2
4E2
5E2
6E2
7E2
8E2
9E2
1E3
Problem size
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads


APR1: user approach, MCN1: user machine,
Mean of Execution Time (s)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1E2
2E2
3E2
4E2
5E2
6E2
7E2
8E2
9E2
1E3
Problem size
ALG; APR1@MCN1, 0 threads
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads





Question 4:

Implementation Details

Brief and clear description about the Serial implementation

For the monte carlo simulation which is used to calculate the value of π, uniformly distributed random values of x and y are taken and the total value of x^2+y^2 is calculated. If the value of x^2+y^2 is greater than 1, the point is outside the circle. IF the value is less than , then the point is inside the circle. $\pi/4$ will be obtained by dividing the total number of points inside the circle by the total

number of simulations.

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

The 'pragma omp parallel for' command is to be used for the purpose of parallel programming. In each and every thread the randomizing function is given a value which is to be seeded so that there is no problem in running all the threads in a parallelized fashion.

Complexity and Analysis Related

Complexity of serial code

No of operations = $n^2 + 25$

Time taken = $n^2 + 25$

Time complexity = O(n^2)

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

No of operations = n
Time taken = n
Time complexity = O(n)

Cost of Parallel Algorithm

The cost of the algorithm is O(n).
Theoretical Speedup (using asymptotic analysis, etc.) Speedup=p/1=p

Number of memory accesses

Serial algorithm:

No of computations outside loop=3
No of computations outside loop=$11n$
Total no of computations=$11n + 3$

Parallel algorithm:

No of memory accesses outside loop=3p
No of memory accesses inside loop=$11n$

Total memory accesses=$11n + 3p$

Number of computations

Serial algorithm:

No of computations outside loop=0
No of computations outside loop=$8n$
Total no of computations=$8n$

Parallel algorithm:

No of memory accesses outside loop=0
No of memory accesses inside loop=$8np$
Total memory accesses=$8np$

Curve Based Analysis

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

As the problem size will increase, the amount of time which will be taken to execute the program will increase for all the threads as a greater amount of work has to be done for a larger problem size.

Time Curve related analysis (as problem size increases, also for serial)

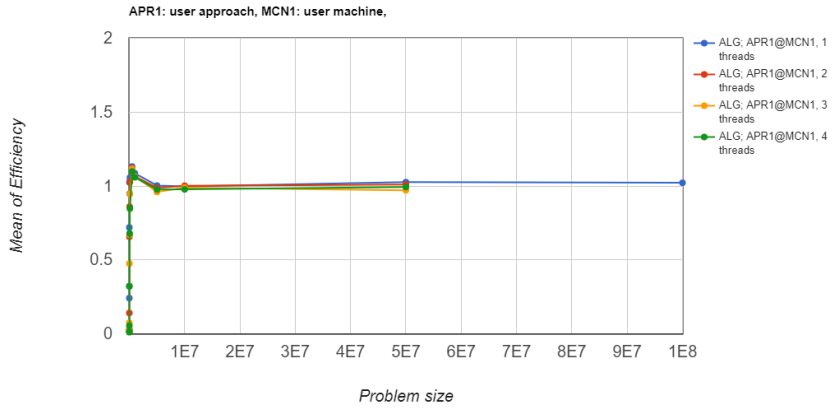
With an increase in the problem size, the execution time will initially not change much because both the effects of higher number of computations due to the implementation of the parallel algorithm and the time reduced due to division of work among processors will cancel out each other, but with a greater problem size, the execution time will increase as a lot of work will have to be done.

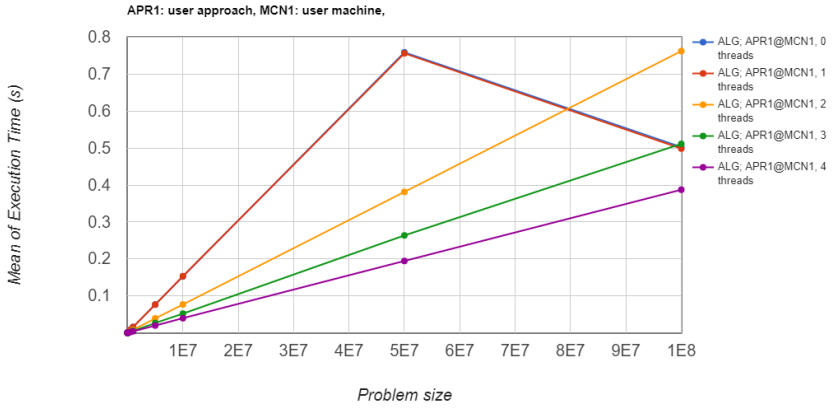
Speedup Curve related analysis (as problem size and no. of processors increase)

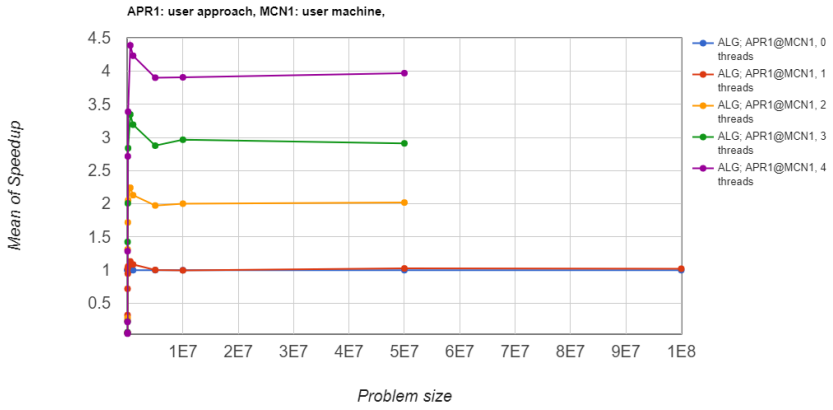
The speedup will be much greater for a greater number of threads because the division of work is done effectively among threads.

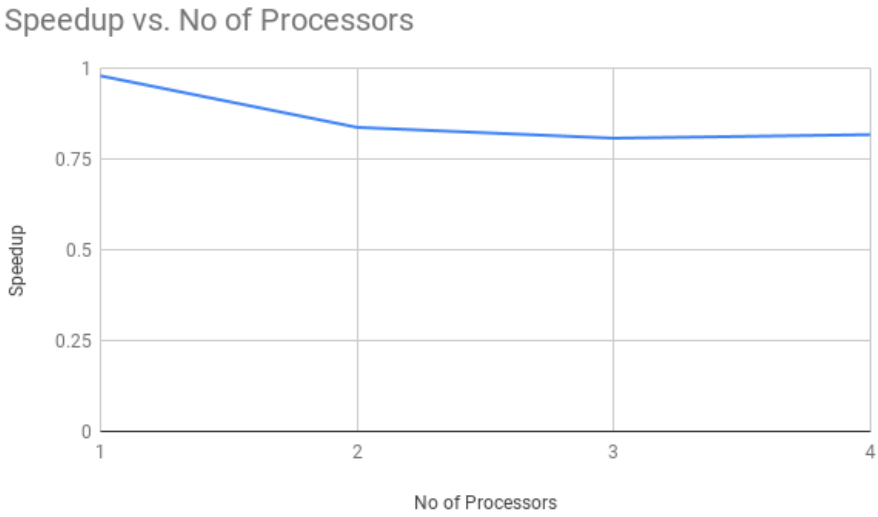
Efficiency Curve related analysis

The efficiency will be almost the same for the division of work among the processors because the speedup will increase with the same factor as the number of processors decrease.


APR1: user approach, MCN1: user machine,
Mean of Efficiency
2
1.5
1
0.5
0
1E7
2E7
3E7
4E7
5E7
6E7
7E7
8E7
9E7
1E8
Problem size
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads


APR1: user approach, MCN1: user machine,
Mean of Execution Time (s)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1E7
2E7
3E7
4E7
5E7
6E7
7E7
8E7
9E7
1E8
Problem size
ALG; APR1@MCN1, 0 threads
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads


APR1: user approach, MCN1: user machine,
Mean of Speedup
4.5
4
3.5
3
2.5
2
1.5
1
0.5
1E7
2E7
3E7
4E7
5E7
6E7
7E7
8E7
9E7
1E8
Problem size
ALG; APR1@MCN1, 0 threads
ALG; APR1@MCN1, 1 threads
ALG; APR1@MCN1, 2 threads
ALG; APR1@MCN1, 3 threads
ALG; APR1@MCN1, 4 threads


Speedup vs. No of Processors
Speedup
1
0.75
0.5
0.25
0
1
2
3
4
No of Processors