Assignment 6

Inderpreet Singh Chera - 160101035

Calculation of PI:

π can be calculated using the following integral:

$$\pi = \int_0^1 \frac{4}{1+x^2} * dx$$

Now using trapezoidal rule, we have:

$$\pi = \sum_{i=0}^{n-1} \frac{4}{1+((i+0.5)/n)^2} * \left(\frac{1}{n}\right)$$, where n is number of steps

The precision of the algorithm is determined by the number of steps taken. And, experiment is performed by taking 1, 2, 4, 8, 16 and 32 threads.

System Settings

All tests were done on **Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz** processor. This computer is **dual core** where each core has **2 threads**. Also during each experiment it was insured that **no other applications** were running in the background, so that we don't have any biased readings.

Implementations

Pi calculations were done with 2 different methods. And comparisons between two were made to which is better.

Using Critical Section

Here, each thread locally calculates the area under the graph that is allocated to it, and then in the critical section adds that value to the shared variable sum. And then pi is computed by multiplying it with the width of one step.

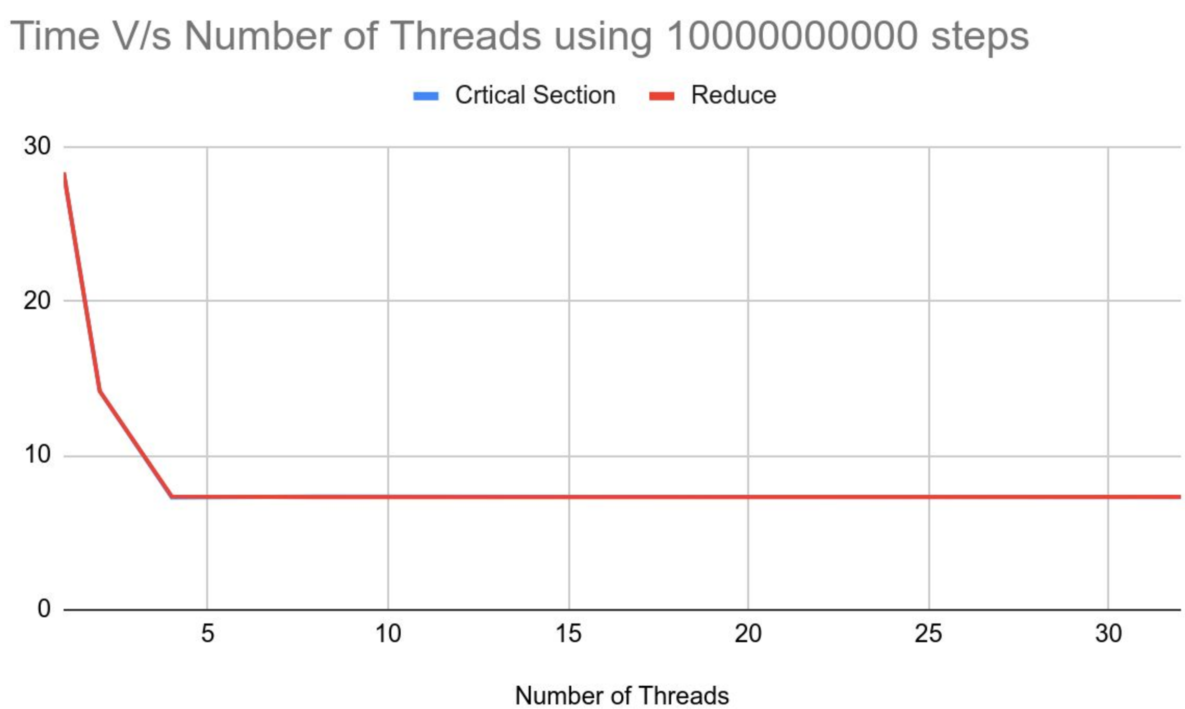
Using Reduce

Here, inbuilt reduce is used to add the values in common shared variable sum and then the value of pi is computed by multiplying it with the width of one step.

Observations

Comparison between the two codes using 10000000000 of steps:

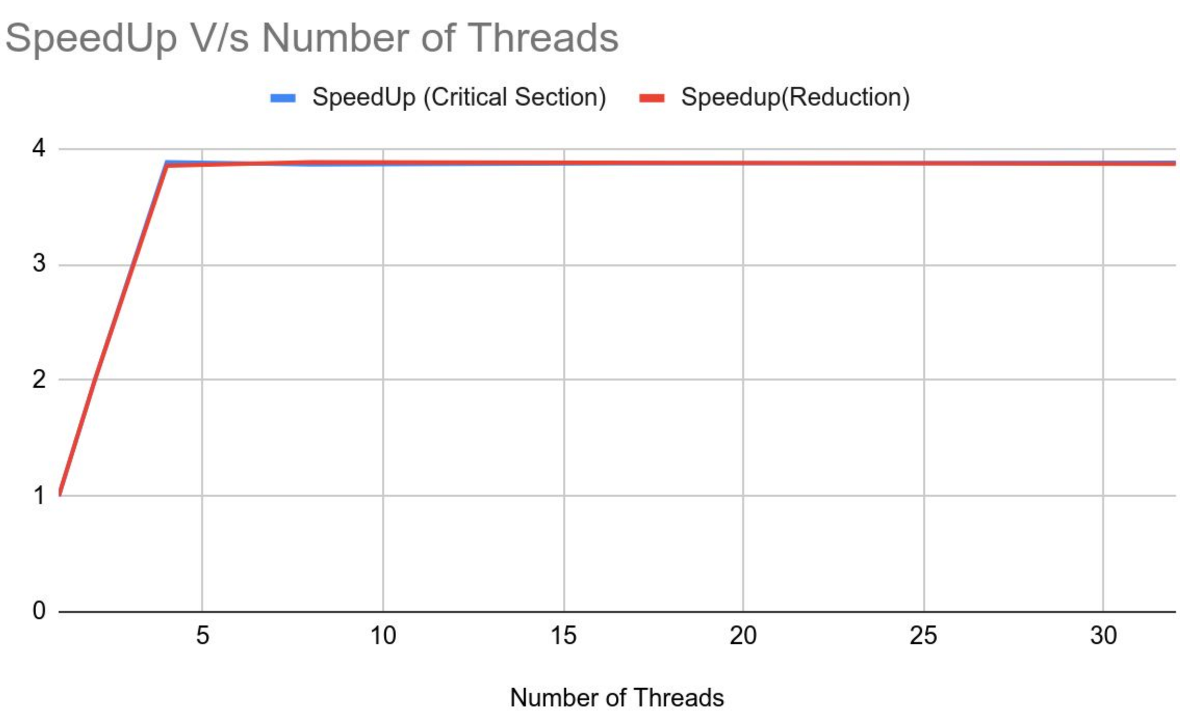
Number of Threads	Critical Section	Reduce
1	28.3277	28.3331
2	14.1718	14.1712
4	7.29166	7.35309
8	7.3293	7.29527
16	7.31392	7.3036
32	7.30319	7.32444



We observe that the two graphs almost coincide. Time taken by each algo first decreases till using 4 numbers of threads, and becomes nearly constant.

Comparison of speedup between two codes:

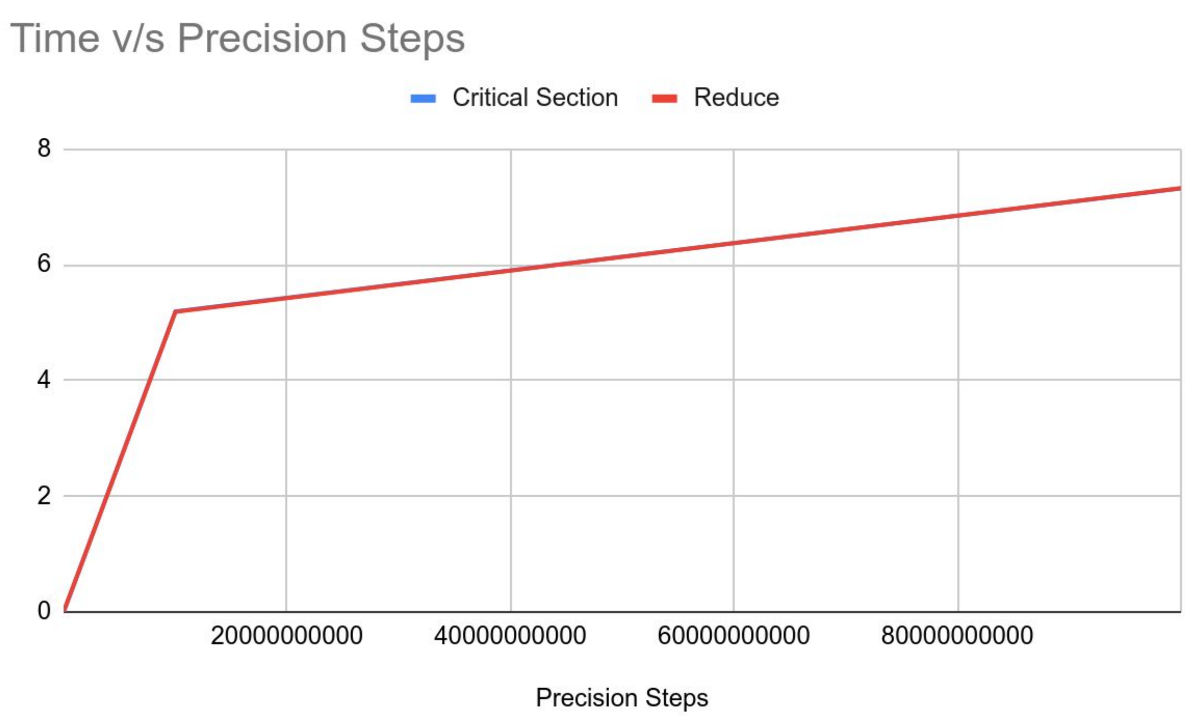
Number of Threads	SpeedUp (Critical Section)	Speedup(Reduction)
1	1	1
2	1.998878054	1.999343739
4	3.884945266	3.853223611
8	3.864993928	3.883763041
16	3.87312139	3.879333479
32	3.878811862	3.868295733



Speed up becomes constant for both types of algo after 4 processes. Before that it is linear. It is probably my pc handles at max 4 threads in parallel.

Comparison among two codes using different number of steps with 4 processes:

Precision Steps	Critical Section	Reduce
100000	0.000156169	0.000158252
1000000	0.000618395	0.00284051
10000000	0.00598753	0.00530832
100000000	0.0528604	0.0604328
1000000000	0.518192	0.517816
10000000000	5.19481	5.18782
100000000000	7.31982	7.32348



Time taken increases on an increasing number of steps, i.e, precision, which should happen.