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:

$$\pi = \sum_{i=0}^{n-1} \frac{4}{1+((i+0.5)/n)^2} * (\frac{1}{n})$$, 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.

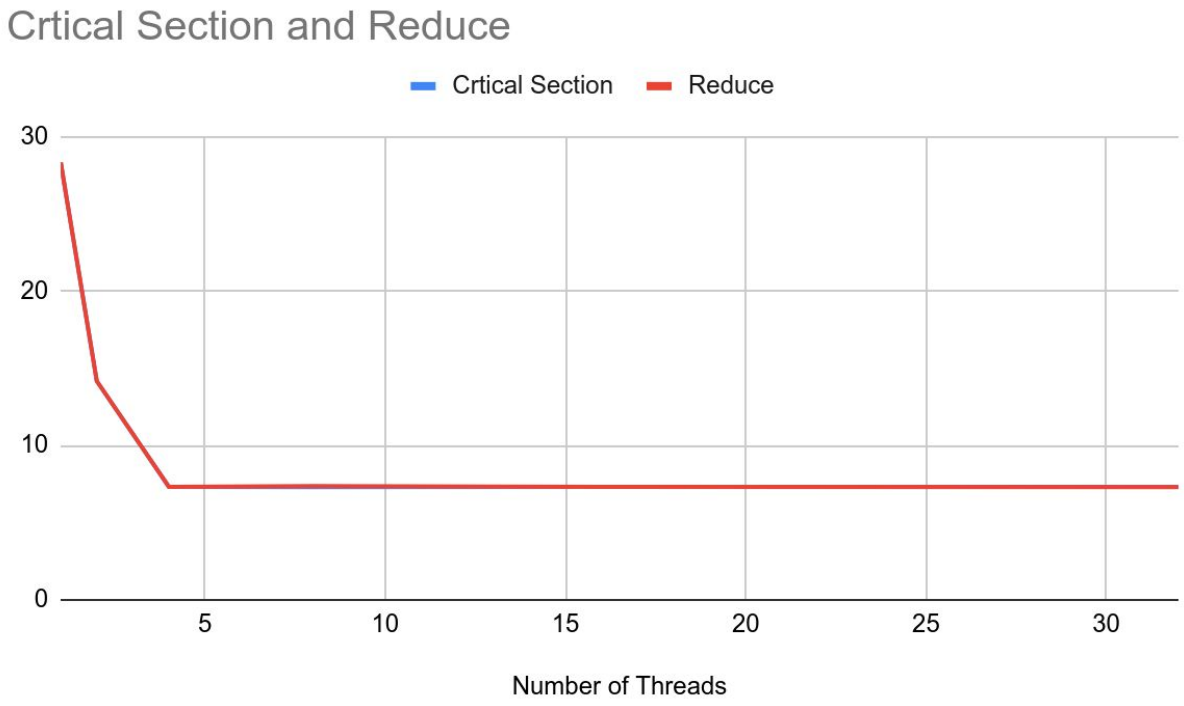
Using Reduce

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:

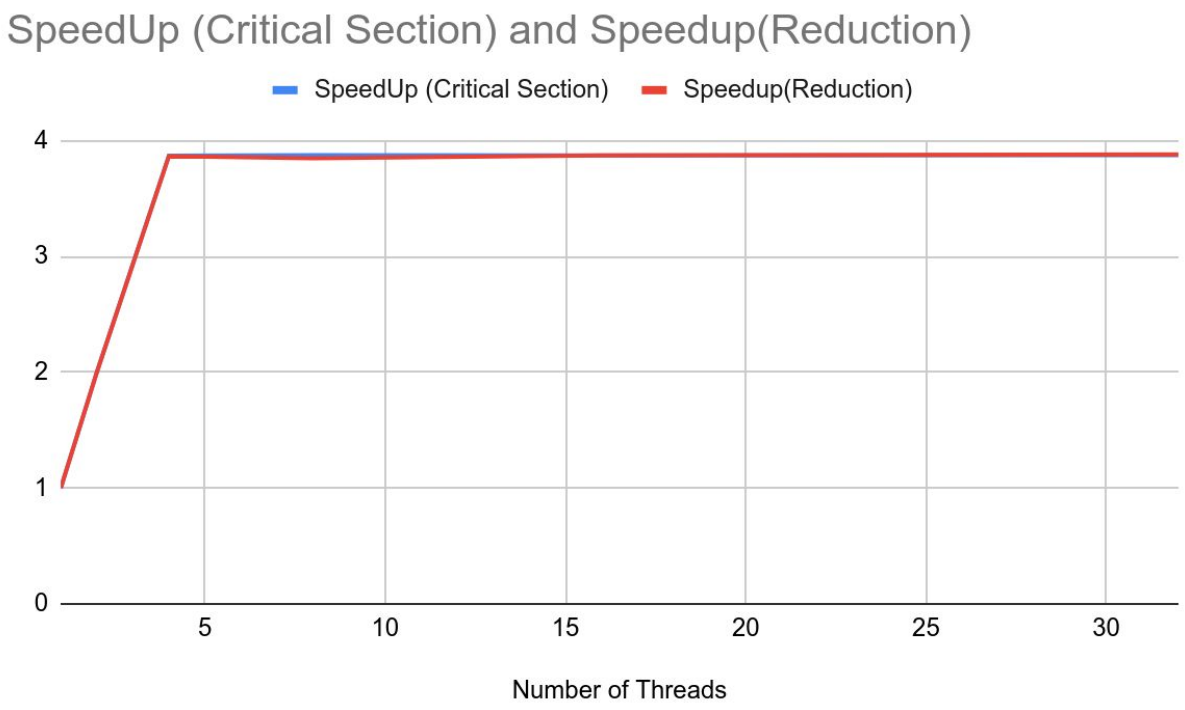
Number of Threads	Critical Section(time taken in sec)	Reduce(time taken in sec)
1	28.3357	28.3302
2	14.1711	14.1684
4	7.32178	7.32635
8	7.31018	7.36204
16	7.31789	7.31345
32	7.31174	7.29848



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:

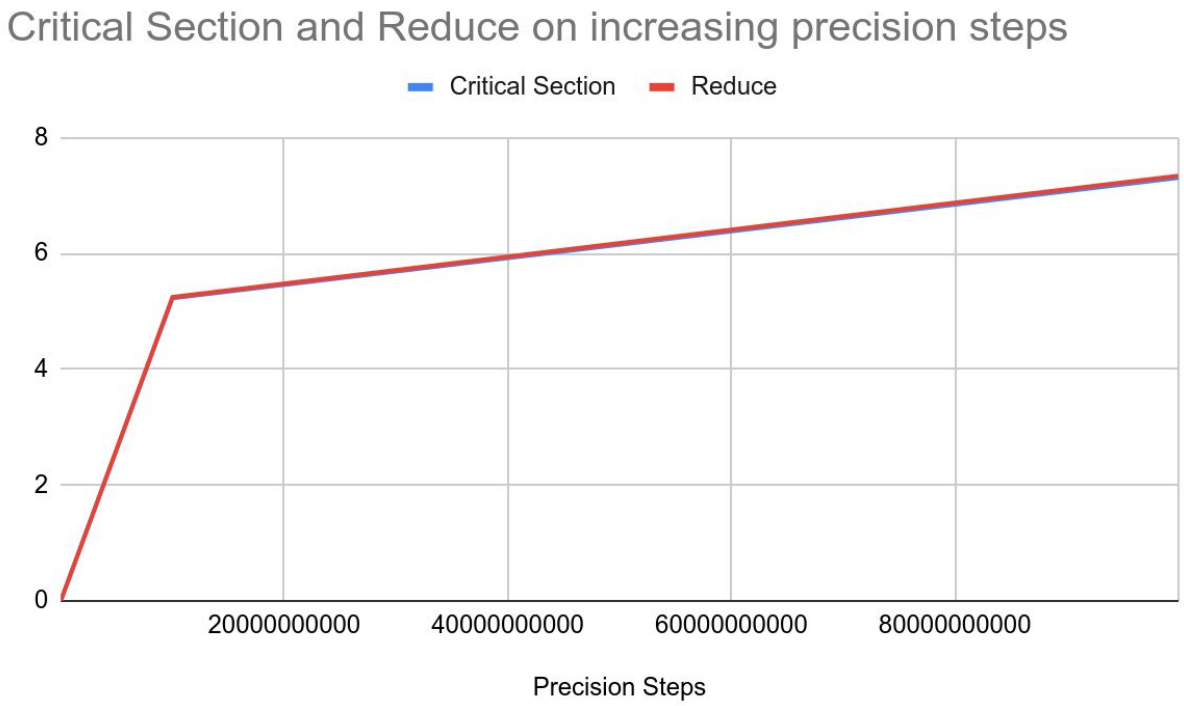
Number of Threads	SpeedUp (Critical Section)	Speedup(Reduction)
1	1	1
2	1.99954132	1.999534175
4	3.870056189	3.866891426
8	3.876197303	3.848145351
16	3.87211341	3.873712133
32	3.875370295	3.881657551



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	0.00015481	0.00021749
1000000	0.00105391	0.00317279
10000000	0.00873687	0.00879048
100000000	0.0518898	0.0519559
1000000000	0.52165	0.519224
10000000000	5.22786	5.24639
100000000000	7.30476	7.34433



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