MA513
Parallel Computing
Assignment-6 Report

Paranjay Bagga
160101050

Problem Statement :-

Calculate the value of PI (using Trapezoidal rule) using a shared memory model while observing the speedup for different values of PEs as well as the precision involved.

System Specifications :-

All the experiments have been performed on **Dell Inspiron 5559** laptop with **Intel i5-6200U dual core processor** and **8 GB RAM**. Each core has **2 threads.** Also, it was made insured that no other applications were running in the background during each experiment which could have incurred biased readings.

Theory :-

Value of **π** can be calculated from the following equation :-

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

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 the no. of steps

The precision of the algorithm is determined by (N) the number of steps taken.

Optimizations :-

Two versions of programs are considered. One using **#pragma omp critical** and the other with **#pragma omp parallel for reduction(+:sum)**.

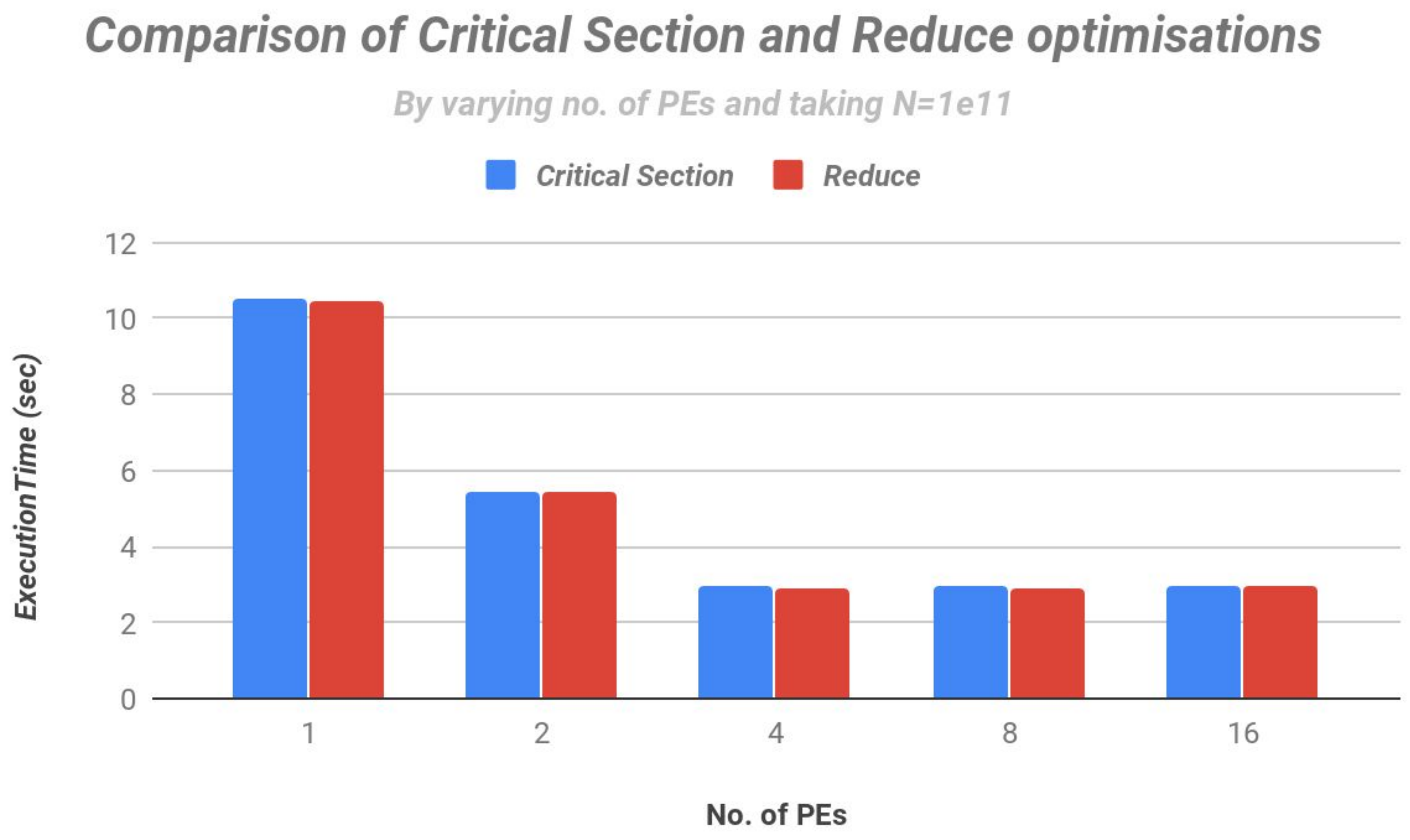
- **Using Critical Section :**
  In this, each PE locally calculates the area under the graph that is allocated to it, and then only once, finally, in the critical section, adds that value to the shared variable sum. Pi is then computed by multiplying the sum with the width of one step (step_val).
- **Using Reduce :**
  Here, the inbuilt Reduce function 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.

Experiments :-

The value of *N* is varied from **10^5(100000)** upto **10^11(100000000000)**.
The results are shown in the tables and graphs below. The values in the table are average values of 8 (or 4 for larger N) executions to avoid any unexpected behaviour introduced as a result of thread switching.

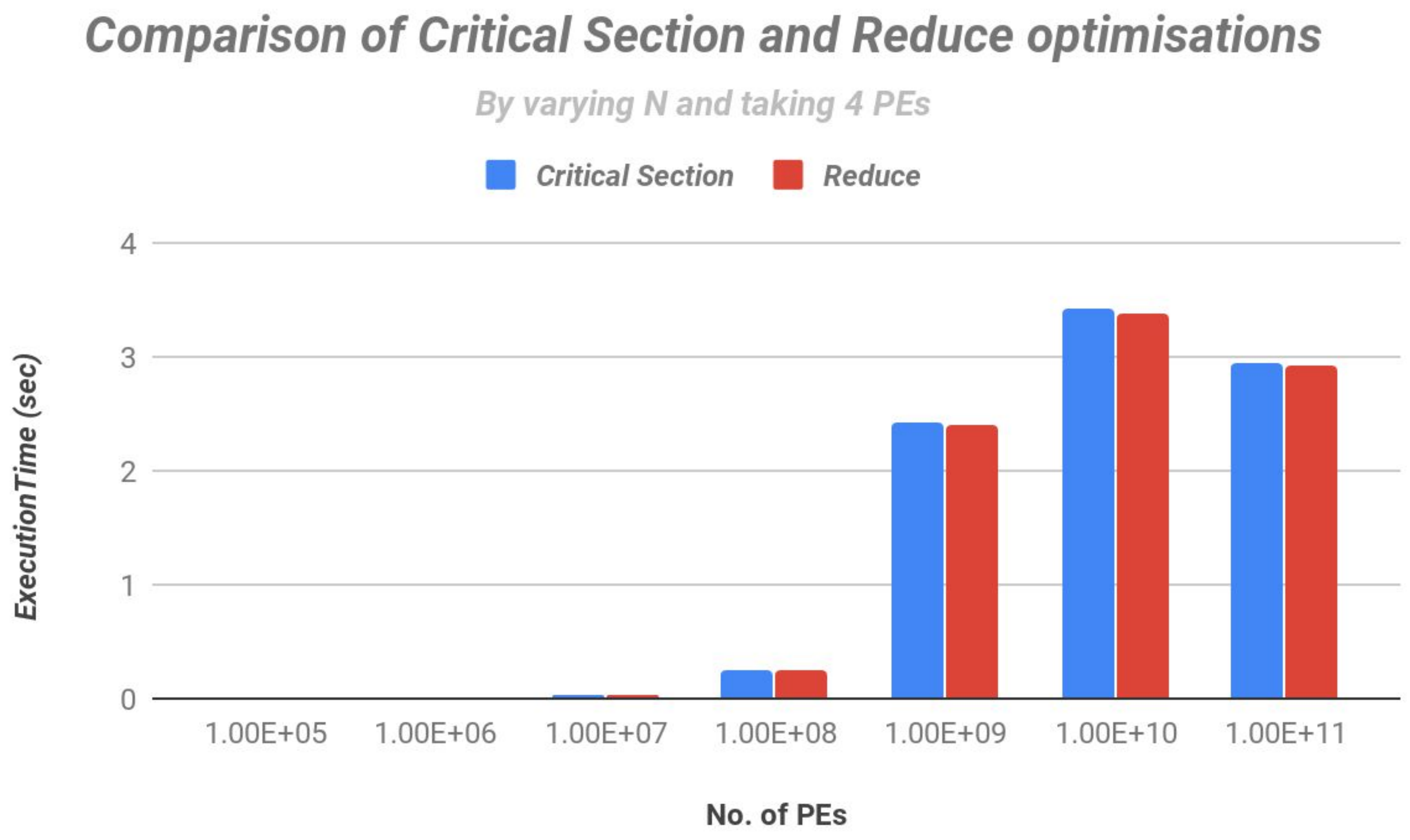
1. Varying No. of PEs and taking N=1e11 steps :

No. of PEs	Critical Section	Reduce
1	10.49665	10.478625
2	5.42793	5.4265875
4	2.9464825	2.9222175
8	2.939075	2.9223675
16	2.9423625	2.9330475



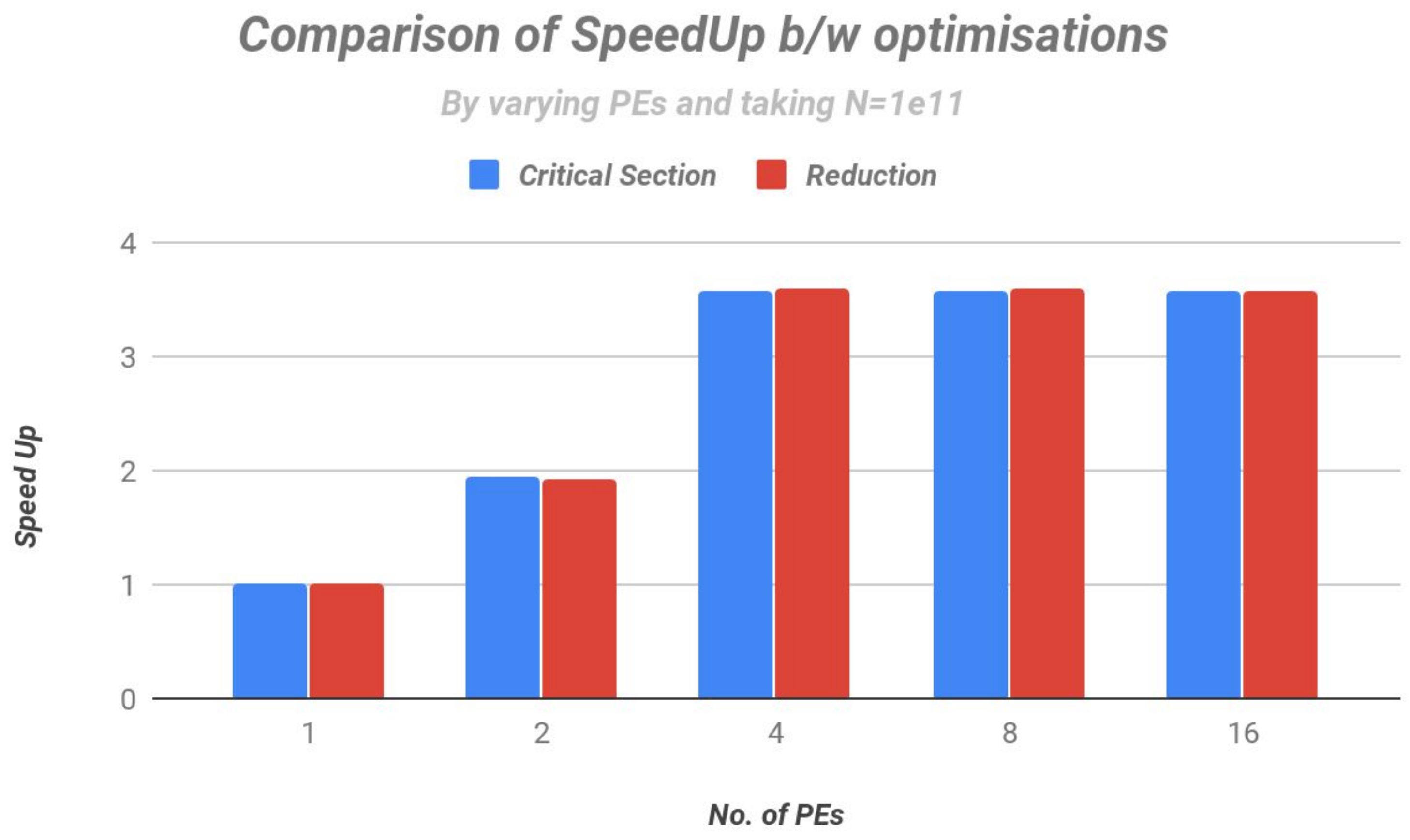
2. Varying N and taking 4 PEs :

Precision Steps	Critical Section	Reduce
100000	0.00321241875	0.002342675125
1000000	0.00406597375	0.00425912375
10000000	0.0290803625	0.0286489
100000000	0.2483815	0.242964
1000000000	2.429215	2.4005925
10000000000	3.417515	3.38556
100000000000	2.9464825	2.9222175



3. Comparing SpeedUp by varying PEs and taking N=1e11 steps :

No. of PEs	Critical Section (SpeedUp)	Reduction (SpeedUp)
1	1	1
2	1.93382191737919	1.93097872281613
4	3.56243419059845	3.58584704937261
8	3.57141277442733	3.58566299413061
16	3.56742243690232	3.57260664888652



Observations :-

- In all the above cases, it is observed that both the optimisations take similar execution time whether N is changed or no. of PEs. However, the Reduce optimisation mostly performs ‘slightly’ better than the Critical Section one.
- From Graph-1, it is observed that, for a particular value of N, the execution time decreases upto 4 PEs, and then becomes nearly constant even on increasing PEs.
- Graph-2 points out that the execution time taken increases on an increasing number of steps, i.e, precision, which should happen naturally. But, some exception was observed with N=1e11 since the value got reduced.
- From Graph-3, it is clear that the Speed Up increases upto 4 PEs and then it becomes nearly constant for both the optimisations.
- The reason behind the changing point occurring at 4 PEs in the graphs may be due to the system being dual core, and thus having 4 threads in total.