High Performance Computing
<u>Assignment -1</u>

Jahnavi Suthar (201301414)

Deeksha Koul (201301435)

Date: 25th August, 2015

Description of the problem

Calculate the value of PI by integrating f(x) = $\frac{4.0}{1+x^2}$ over the interval [0, 1] using trapezoidal rule.

$$\int_0^1 \frac{4.0}{1+x^2} = \pi$$

Complexity of the Problem (Serial):

We integrate the function f(x) by dividing the interval [0, 1] into 'n' equal rectangles and summing the area of each rectangle.

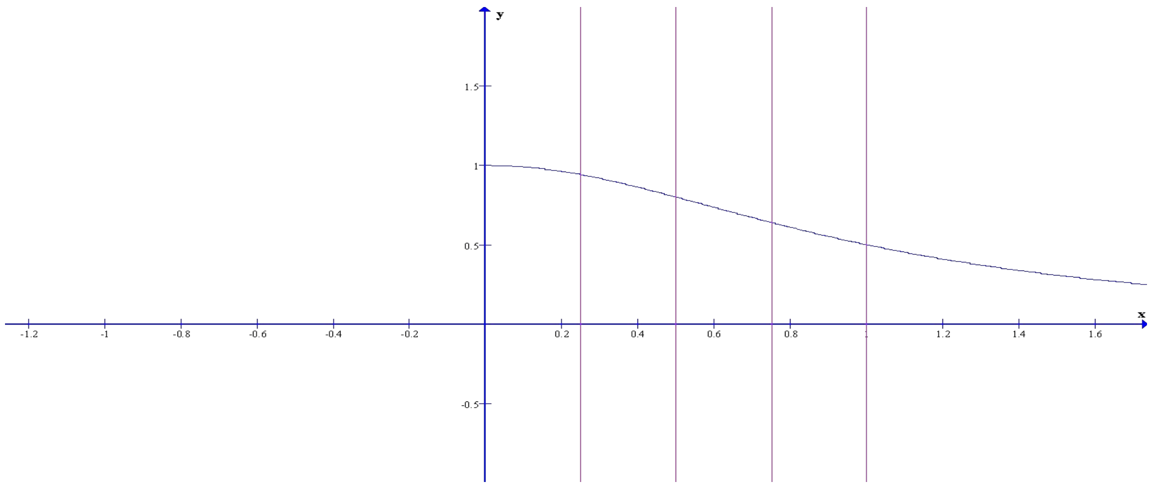
The complexity of the problem is O (n), where n is number of rectangles in which f(x) has been divided.

Possible Speedup (Theoretical)

Speedup = 1 / (S + (P/M))

= M (Code can be fully parallelized. P = 1)

where M is no. of cores, P is fraction of code that can be parallelized and S is serial fraction of code. P+S=1.



f(x) = $\frac{4.0}{1+x^2}$

Optimization strategy

We equally divide the number of steps between threads. Each thread integrates the function f(x) over the given interval using trapezoidal rule. At the end we add up partial sum of each thread to get final integral value.

Problems faced in Parallelization and possible solutions

1. Synchronization problem is faced when using global shared variables (such as sum). To solve this problem we used private variables i.e. for each thread we calculate partial sum, and finally add them together. We can also use mutual exclusion if we want to use a global variable.
2. As there are only four cores, there is no use in exceeding number of threads, because that would not help in improving performance.

Hardware Details

CPU Model: INTEL(R) Core i5-4590

No. of cores: 4

Memory: 7.6 GiB

Compiler: gcc

Optimization flags if used: None

Precision used: single point precision (float)

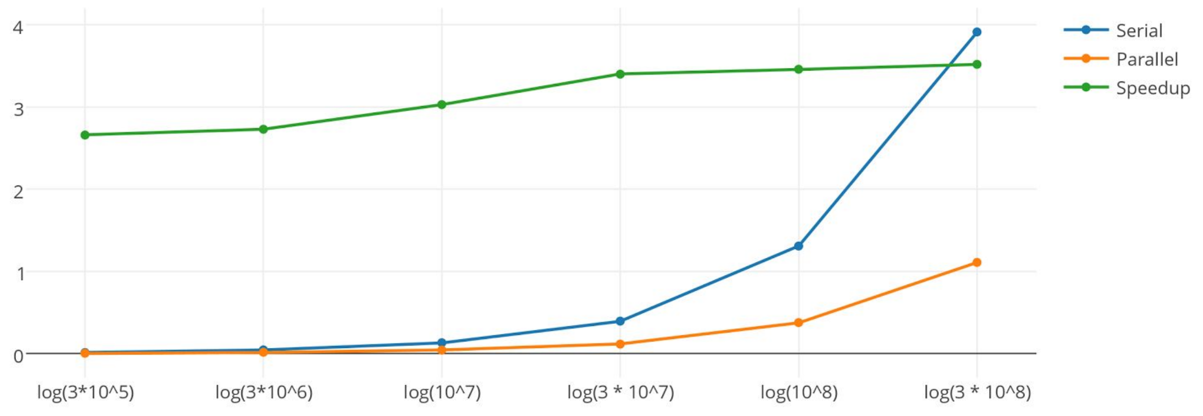
Input Parameters: Number of steps, Number of threads(for parallel code).

Output: Value of PI = 3.141593

Problem size vs time for serial and parallel code and speedup:

Problem size	Serial Time	Parallel Time	Speedup
3×10^5	0.013830	0.005191	2.66
3×10^6	0.046180	0.016882	2.73
10^7	0.133000	0.043800	3.03
3×10^7	0.395616	0.116347	3.40
10^8	1.309391	0.377933	3.46

3×10^8	3.913586	1.110313	3.52

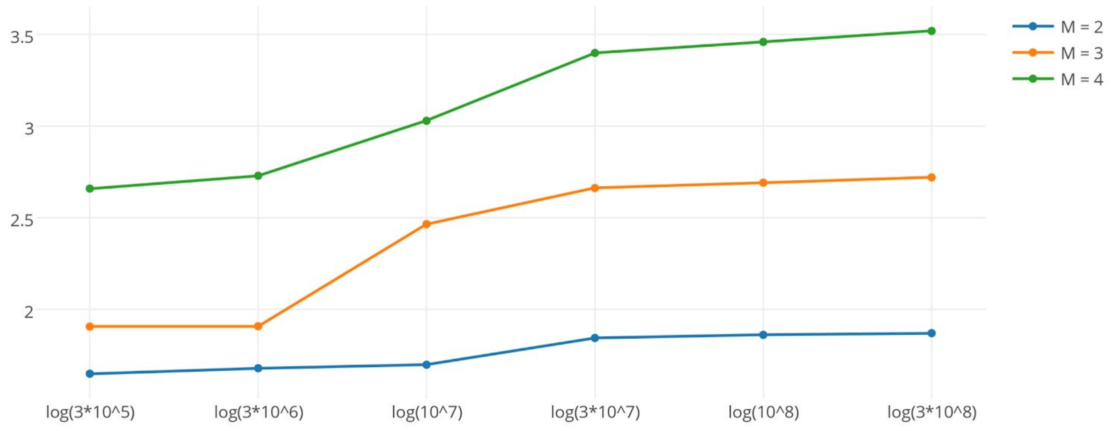


Problem size vs running time for serial and parallel code and speedup for M = 4

As the problem size increases the parallel code takes lesser and lesser time than the serial code. Thus, the speedup increases.because as the ‘n’,number of step keeps on increasing,there is more amount of parallelization(and hence running time in case of parallel code becomes less) and thus code becomes more optimized.

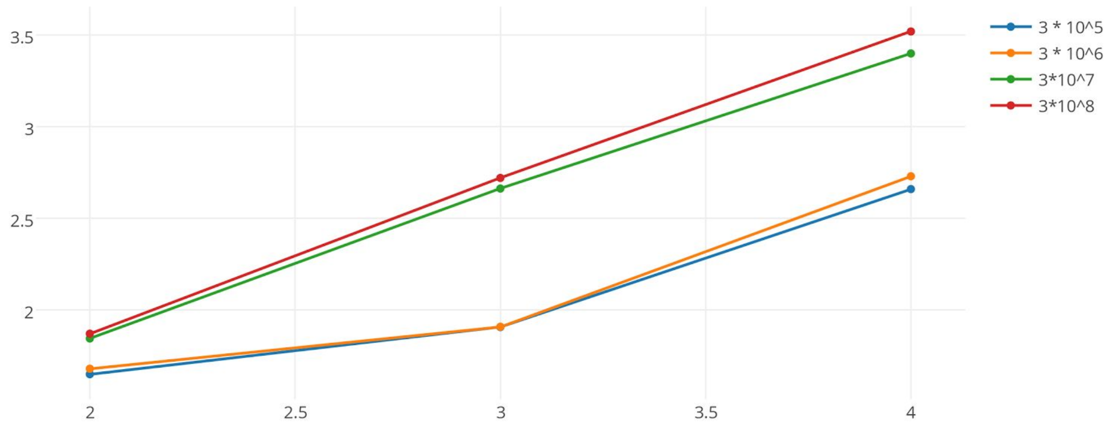
Problem Size vs speedup for different number of cores

Problem size	M = 2	M = 3	M = 4
3×10^5	1.65	1.907	2.66
3×10^6	1.680	1.909	2.73
10^7	1.699	2.466	3.03
3×10^7	1.845	2.663	3.40
10^8	1.8621	2.692	3.46
3×10^8	1.8712	2.721	3.52



Problem size vs speedup for M = 2, 3 and 4

At a constant number of steps, as the number of cores increases, the execution time decreases and hence the speedup increases as more number of cores are performing the same task. When we run code with four processors we get the maximum speedup. But there is not much improvement in performance if we increase the number of threads after 4.



No. of cores vs speedup for problem sizes 3×10^5, 3×10^6, 3×10^7, 3×10^8