Design of the program

The program is designed to use multithreading using the pthread library in c++. Each thread generates random points, stores the points in a global variable and flags whether the point lies inside the circle or not. The flags are also stored in a global variable. The main thread calculates pi using the formula given in the textbook. To find out the number of points inside the circle, the main thread counts the number of set flags.

The points are stored in a single global double array. The flags are stored in a single global integer array. Therefore each thread needs to know which indices it needs to store its points and flags in. The space in the global arrays is partitioned and assigned to each thread based on the thread number. Thus each thread is passed the thread number as an argument. In this design, the threads don't need to return anything.

The formula used is

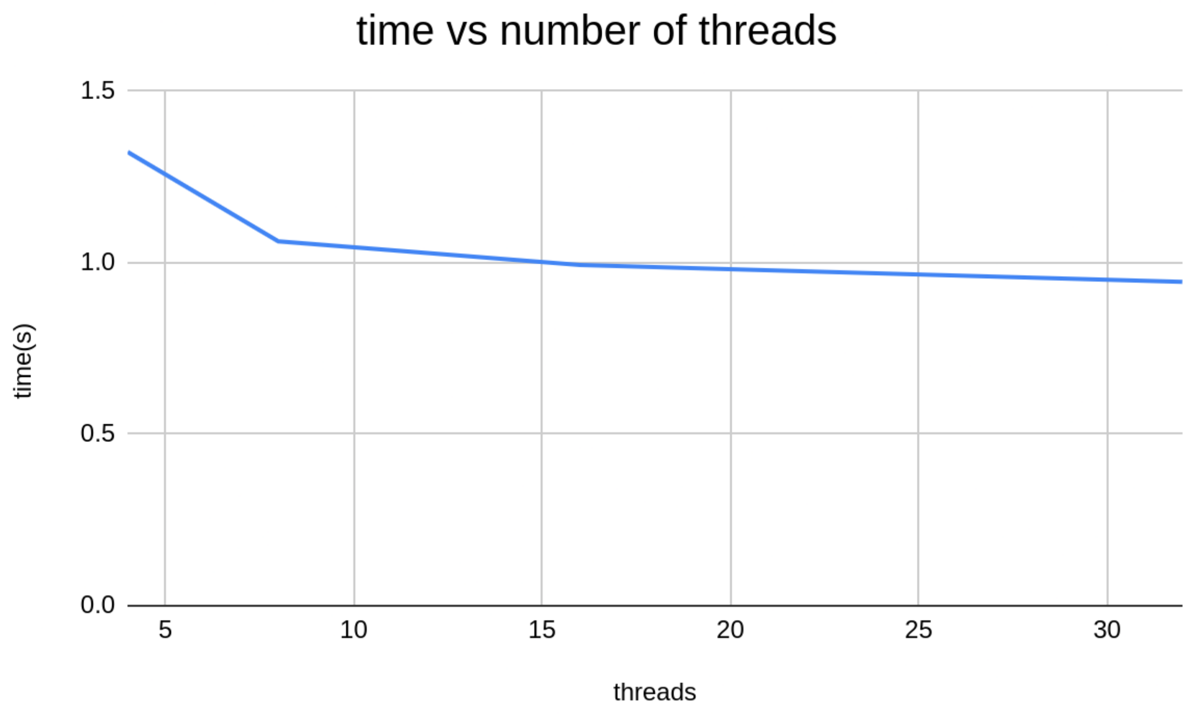
$$\pi = 4\times \textit{(number of points in circle) / (total number of points)}$$

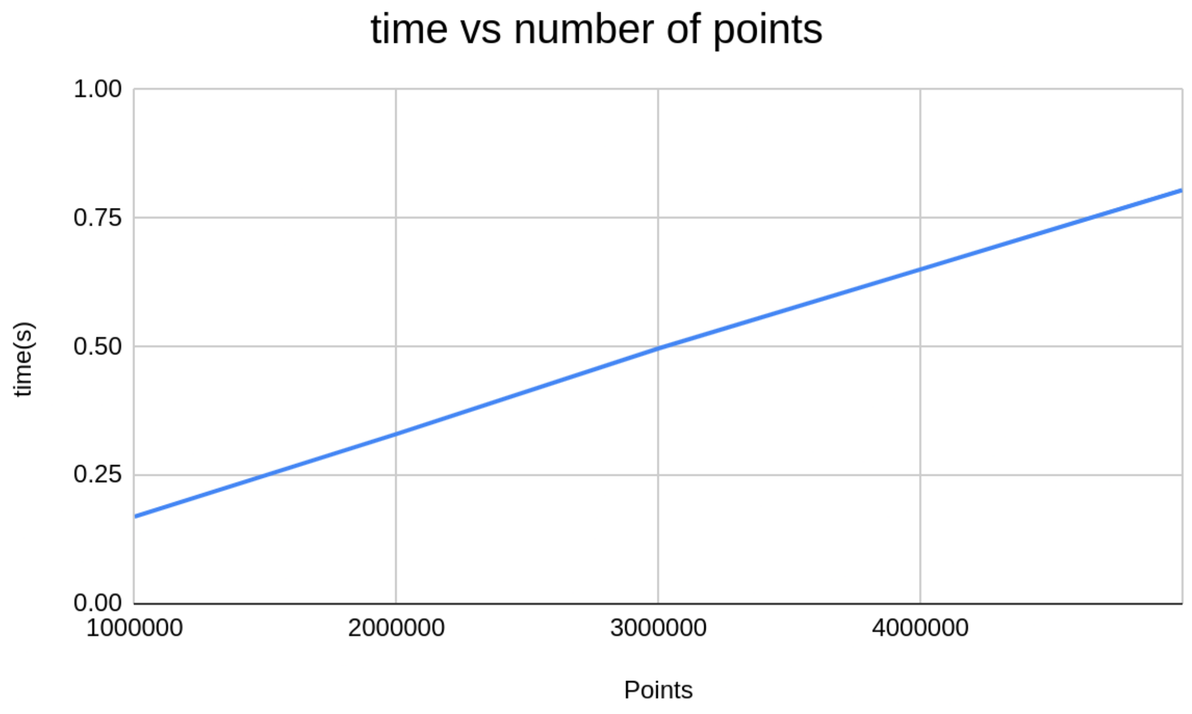
Since a single output file needs to be created, only the main thread writes to the output file. The time is calculated in the main thread using the chrono package. Main thread uses the global arrays to write the logs. Multithreading is not used while writing to the output file to avoid possible conflicts. Therefore creating the output file is the bottleneck in the code in terms of the time taken. However, only the time taken to calculate pi is considered in the results.

For more details on the implementation, please refer to the comments written in the source code.

Analysis of the results and plots

The obtained plots after averaging over 5 observations are:





As expected, the time taken when increasing the number of points linearly increases linearly as observed in the second plot.

In the first plot, we observe that the time keeps decreasing as the number of threads increases. However, we can observe that on doubling the number of threads, the time is not halved. This is because of Amdahl's law

$$speedup \leq \frac{1}{S + \frac{(1-S)}{N}}$$

Where S is the fraction of sequential processing, and N is the number of threads.