CO21BTECH11002

Aayush Kumar

OS1 Assignment 2

The checkPoints( ) function:

I have taken the circle to be for doing all the calculations.

We first initialize the arrays pointsX, and pointsY for storing the points.

This function takes the struct values as input and generates random numbers equal to the number of points the thread needs to check using Standard mersenne\_twister\_engine.

For each point (x, y), we calculate its distance from the origin (i.e., ). If the distance is smaller than the radius of circle (i.e., 1), we conclude that the point lies inside the circle, else, it lies outside the circle.

If a point is found to be inside the circle, we put at the position st and increment st by 1 and increase count by 1. Similarly, if the point is found to be outside the circle, we place it at the position en and decrement en by 1.

In this way all the points inside the circle will occupy front of the array and points outside the circle will occupy the back of the array.

The main( ) function:

We start by taking reading the value of n and k from the input file.

ct variable holds the number of points each thread will work on.

The program initializes the number of points each thread needs to check.

Then it creates and runs all the threads and waits for them to finish. After the threads have executed, it joins all the threads.

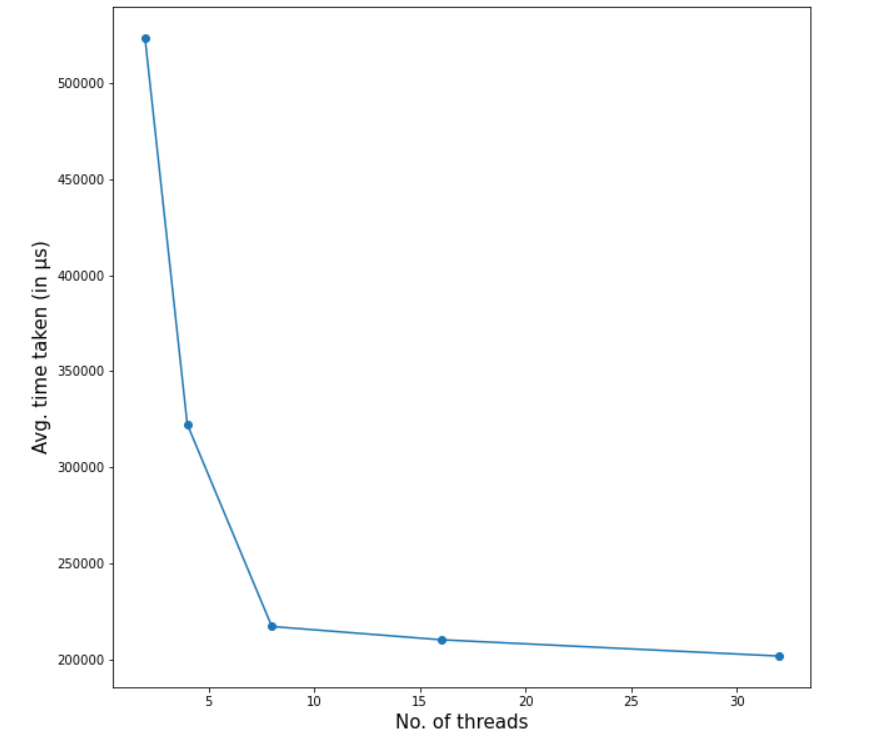
Then it creates the Output file and writes it using the values found by running the threads, stored in threadValues array.

Results:

After running the program 5 times and taking the average value of time taken (in microseconds), we get the following results:

Time taken vs no. of threads (keeping no. of points = 6000000):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Threads | Exec 1 | Exec 2 | Exec 3 | Exec 4 | Exec 5 | Average |
| 2 | 533183 | 514716 | 521572 | 532014 | 532014 | 516068.6 |
| 4 | 320099 | 328037 | 340128 | 318545 | 304758 | 322313.4 |
| 8 | 212696 | 239114 | 214009 | 208763 | 210784 | 217073.2 |
| 16 | 213268 | 215722 | 205352 | 200224 | 216299 | 210173.0 |
| 32 | 217148 | 193805 | 201499 | 196844 | 199139 | 201687.0 |



We can see that after the no. of threads becomes more than no. of cores in the system (8 in my case), there is no significant increase in performance. This is because context switching between threads is much more expensive than context switching between processes. Also creating and exiting threads has an extra overhead which becomes significant when large no. of threads are doing smaller operations.

Time taken vs no. of points (keeping no. of threads = 32):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Points | Exec 1 | Exec 2 | Exec 3 | Exec 4 | Exec 5 | Average |
| 1000000 | 62859 | 56864 | 65733 | 56409 | 69325 | 62238.0 |
| 2000000 | 123617 | 97805 | 106219 | 96939 | 90911 | 103098.2 |
| 3000000 | 115083 | 118380 | 112546 | 142723 | 114983 | 120743.0 |
| 4000000 | 142028 | 151060 | 143778 | 137430 | 169239 | 148707.0 |
| 5000000 | 172341 | 192583 | 186115 | 167714 | 200741 | 183898.8 |

