Abstract

Reproducing the serial code by creating random arrays of different sizes and observing the duration.

1 Introduction

Reproducing the serial code by creating random arrays of sizes 1000, 10000, 100000, and 10000000. Then observing the value of Pi on each array size, and checking how long it took.

2 Implementation

At each iteration we're generating a random number from 0 to 1.

Listing 1: C Example

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

int main()
{
        int             i;//number of iterations
        double  duration;
        double  pi;
        clock_t start;
        clock_t end;
        int     counter;

        i               = 10000000;
        start   = clock();
        counter = 0;

        srand(time(NULL));

        for (int index = 0; index < i; index++)
    {
        double a = (rand() % 543) / (double)(542);
        double b = (rand() % 543) / (double)(542);
        if (sqrt((a * a) + (b * b)) <= 1)
        {
            counter++;
        }
    }

        end             = clock();
        duration= ((double)(end - start) / CLOCKS_PER_SEC) * 1000;
        pi              = 4.0 * ((double)counter / i);
        printf("Serial of pi is %f and it takes %fms\n", pi, duration);

        return 0;
}
```

3 Experimental Platform

Windows 10, Sublime text editor and a GCC compiler

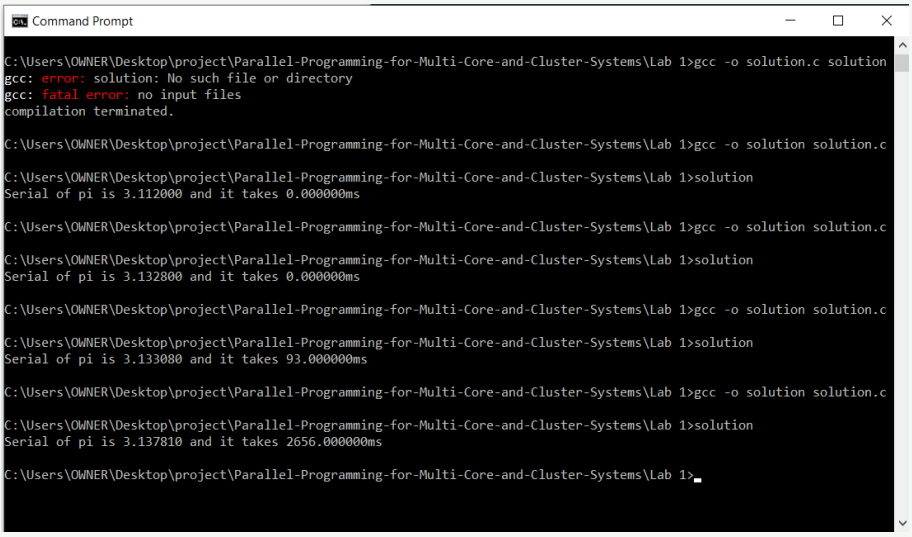

Figure 1: A screenshot of the terminal

4 Results

Serial Performance		
Size	Pi	Time (ms)
1000	3,112000	0.0000
10000	3.132800	0.0000
100000	3.133080	93.0000
100000000	3.137810	2656.0000

5 Discussion

With the increase of the number of iterations, the value of pi becomes more accurate, and the run-time of the program increases. Thus, the more random numbers we generate to increase the accuracy of pi, the more the duration will increase (bad performance)

6 Conclusion

The more random numbers we generate to increase the accuracy of pi, the more the duration will increase (bad performance)