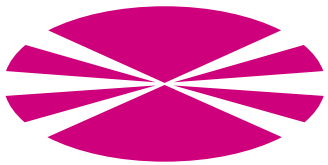
Concurrency and Parallelism. Block II Parallelism

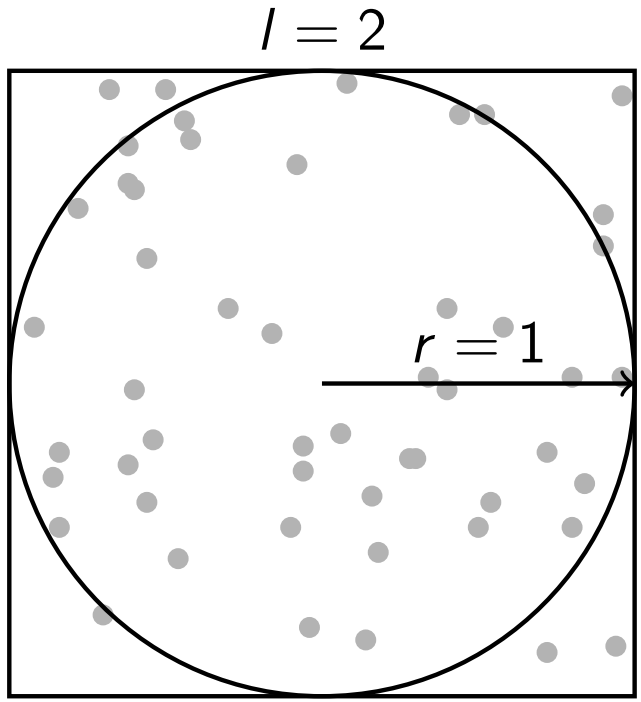
Assignment 1: estimation of PI by the Montecarlo method

Spring 2022



Estimation of PI by the Montecarlo method

- Geometric aproximation of PI.
- Generate N random points within a square S with side length equal to 2, and centered in $(0, 0)$.
- Assume a circle D with radius 1 also centered in $(0, 0)$.
- The amount of points that fall inside the circle is proportional to pi:
- $Q = \frac{A(D)}{A(S)} = \frac{\pi \cdot r^2}{l^2} = \frac{\pi}{4}$
- The greater N, the more accurate is the PI aproximation.



Estimation of PI by the Montecarlo method

Sequential code

```
int main(int argc, char *argv[]){
  int i, done = 0, n, count;
  double PI25DT = 3.14159265358979323846264643;
  double pi, x, y, z;

  while (!done) {
    printf("Enter the number of points: (0 quits) \n");
    scanf("%d",&n);
    if (n == 0) break;
    count = 0;
    for (i = 1; i <= n; i++) {
      x = ((double) rand()) / ((double) RAND_MAX);
      y = ((double) rand()) / ((double) RAND_MAX);
      z = sqrt((x*x)+(y*y));
      if(z <= 1.0)
        count++;
    }
    pi = ((double) count/(double) n)*4.0;
    printf("pi is approx. %.16f, Error is %.16f\n", pi, fabs(pi - PI25DT));
  }
}
```

Estimation of PI by the Montecarlo method

Parallelization

- SPMD implementation
- I/O (scanf/printf) is made by process 0
- Distribute n to all the processes (with Send/Recv)
- Divide the workload of the for loop with "step" i+=numprocs instead of i++
- Gather the estimation of PI in each process (with Send/Recv, is the data required to be received sorted?)

Conditions of the assignment

- Assigned points: 0.25
- It must be done in couples
- Defended in the laboratory lectures: April 19th to 25th