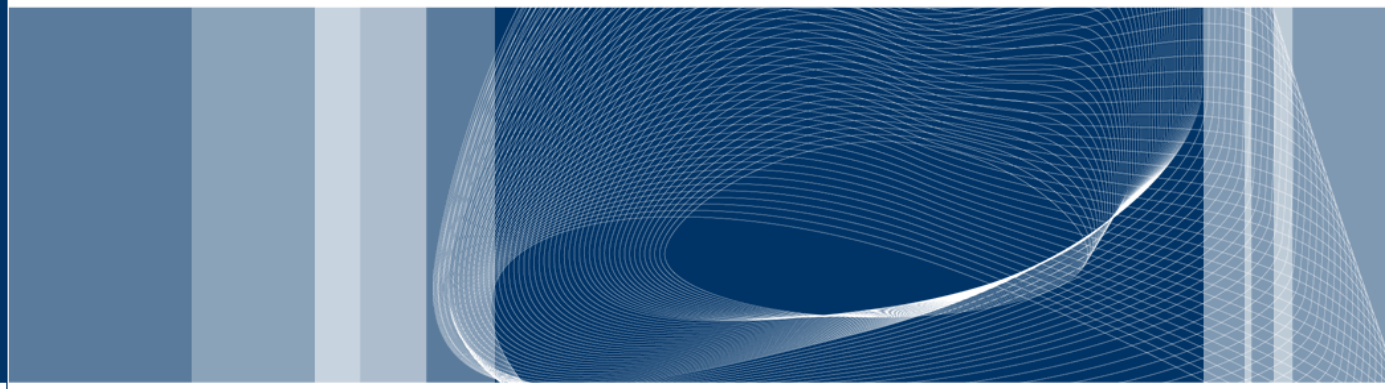
POLITECNICO DI MILANO



Esercizi MPI

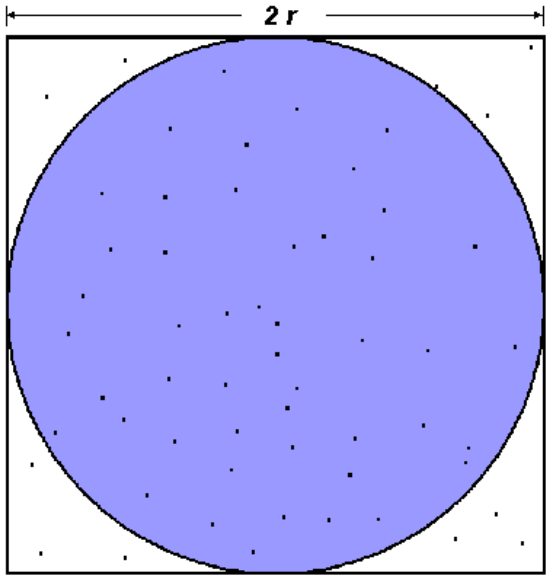
Algoritmi e Calcolo Parallelo

Daniele Loiacono

Esercizio 1

- Implementare in MPI una soluzione parallela del seguente algoritmo per approssimare PI

```
cin >> npoints; count = 0;
for(j=0, j<npoints; j++) {
      x = random();
      y = random();
      if (inCircle(x, y))
            count++;
}
PI = 4.0*count/npoints
```



$A_S = (2r)^2 = 4r^2$

$A_C = \pi r^2$

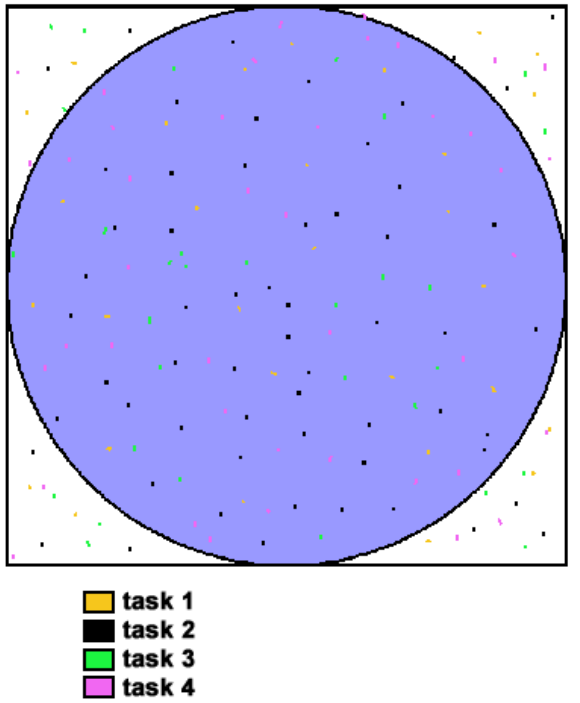
$\pi = 4 \times \frac{A_C}{A_S}$

- Non è necessario implementare la funzione `inCircle()`
- Si implementi
  - una soluzione senza utilizzare le funzioni di comunicazione collettive
  - una soluzione in cui vengono utilizzate le funzioni di comunicazione collettive

Esercizio 1: soluzione


```
cin >> npoints; count = 0;
p = number of tasks; N = npoints/p;

for(j=1, j<N; j++) {
    x = random();
    y = random();
    if (inCircle(x, y))
         count++;
}
PI = 4.0*count/N
find out if I am MASTER or WORKER
if I am MASTER
    receive from WORKERS their PI
    compute PI (use MASTER and WORKER calculations)
else if I am WORKER {
   send to MASTER PI
}
```



Esercizio 2

- Parallelizzare in MPI il seguente algoritmo che
  - conta i numeri primi inferiori a LIMIT
  - ritorna il numero primo più grande trovato

```
pc=4;   /* 2,3,5,7 are counted here */
for (n=11; n<=LIMIT; n=n+2) {
   if (isprime(n)) {
      pc++;
      foundone = n;
   }
}
cout << "found= " << pc << " largest= " << foundone << endl;
```