Algoritmos Paralelos

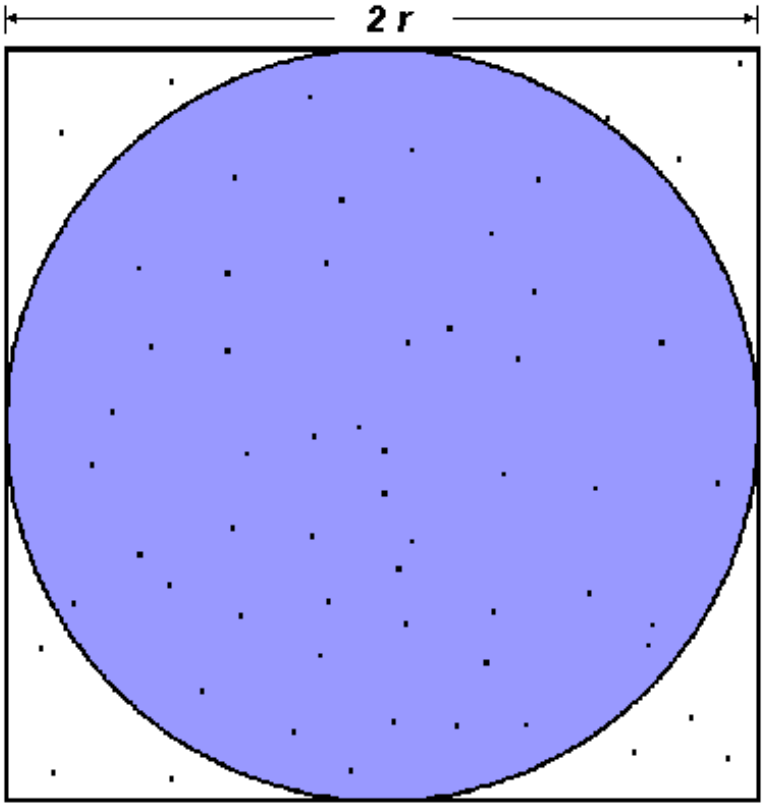
(18.09.15)

Prof. J.Fiestas

Ejercicio 10:

Método de Monte Carlo para calcular PI

- Genera N puntos random en el cuadrado exterior
- Determina el número de puntos n dentro del círculo



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

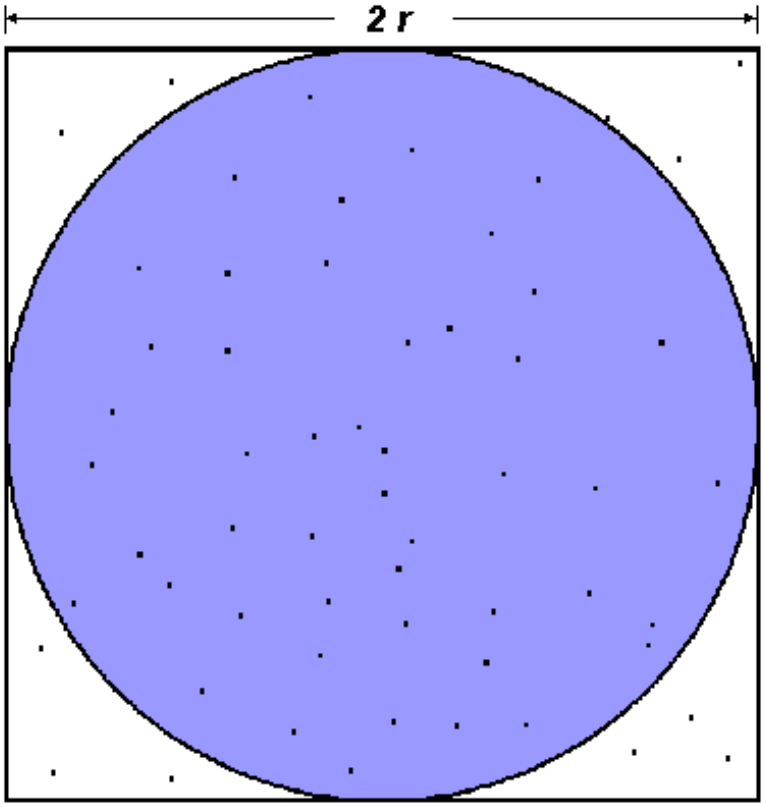
$$A_C = \pi r^2$$

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

Ejercicio 10 :

Método de Monte Carlo para calcular PI

- Siendo el área del círculo pi*r^2, y el área del cuadrado 4*r^2
- Entonces n/N = pi/4, o pi=4*n/N
- Generando un número suficiente de puntos se logra un resultado mas exacto



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

$$A_C = \pi r^2$$

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

Ejercicio 10:

Método de Monte Carlo para calcular PI

```
int main ()
{
   long i;  long Ncirc = 0;
   double pi, x, y, test;
   double r = 1.0;   // radius of circle. Side of squrare is 2*r
   time_t t;

   srand(time(&t));

   for(i=0;i<num_trials; i++)
   {
      x  = (double)rand() / RAND_MAX;
      y  = (double)rand() / RAND_MAX;
      test = x*x + y*y;
      if (test <= r*r) Ncirc++;
    }

    pi = 4.0 * ((double)Ncirc/(double)num_trials);
    printf("\n %ld trials, pi is %lf \n",num_trials, pi);
    return 0;
}
```

Ejercicio 10:

Método de Monte Carlo para calcular PI

1. Programar el código en serie para *num_trials* de 10^2 hasta 10^8 y comparar resultados

Determinar el error relativo en el calculo de PI con respecto a

PI = 3.14159265358979323846264 3

i.e. $(PI_{exp}-PI_{teo})/PI_{teo}$

Ejercicio 10:

Método de Monte Carlo para calcular PI

Números aleatorios

rand() en C utiliza un generador random de congruencia lineal basado en la relación

$$I_{j+1} = aI_j + c \pmod{m}$$

Ejercicio 10:

Método de Monte Carlo para calcular PI

2. Programar un generador pseudo-random alternativo de congruencia lineal (Numerical Recipees in C, Cap.7), segun:

```
double drandom()
{
    long random_next;
    double ret_val;

// compute an integer random number from zero to mod
    random_next = (MULTIPLIER  * random_last + ADDEND)% PMOD;
    random_last = random_next;

// shift into preset range
    ret_val = ((double)random_next/(double)PMOD)*(random_hi-random_low)+random_low;
    return ret_val;
}
```

Ejercicio 10:

Método de Monte Carlo para calcular PI

Donde:

```
static long MULTIPLIER  = 1366;
static long ADDEND      = 150889;
static long PMOD        = 714025;
```

Y el seed:

```
void seed(double low_in, double hi_in)
{
   if(low_in < hi_in)
   {
      random_low = low_in;
      random_hi  = hi_in;
   }
   else
   {
      random_low = hi_in;
      random_hi  = low_in;
   }
   random_last = PMOD/ADDEND;  // just pick something

}
```

Ejercicio 10:

Método de Monte Carlo para calcular PI

Note que estamos usando el último random (*random_last*) como siguiente seed

3. Repetir el ejercicio y comparar resultados. Con cual método se obtienen mejores resultados y por que?

Ejercicio 10:

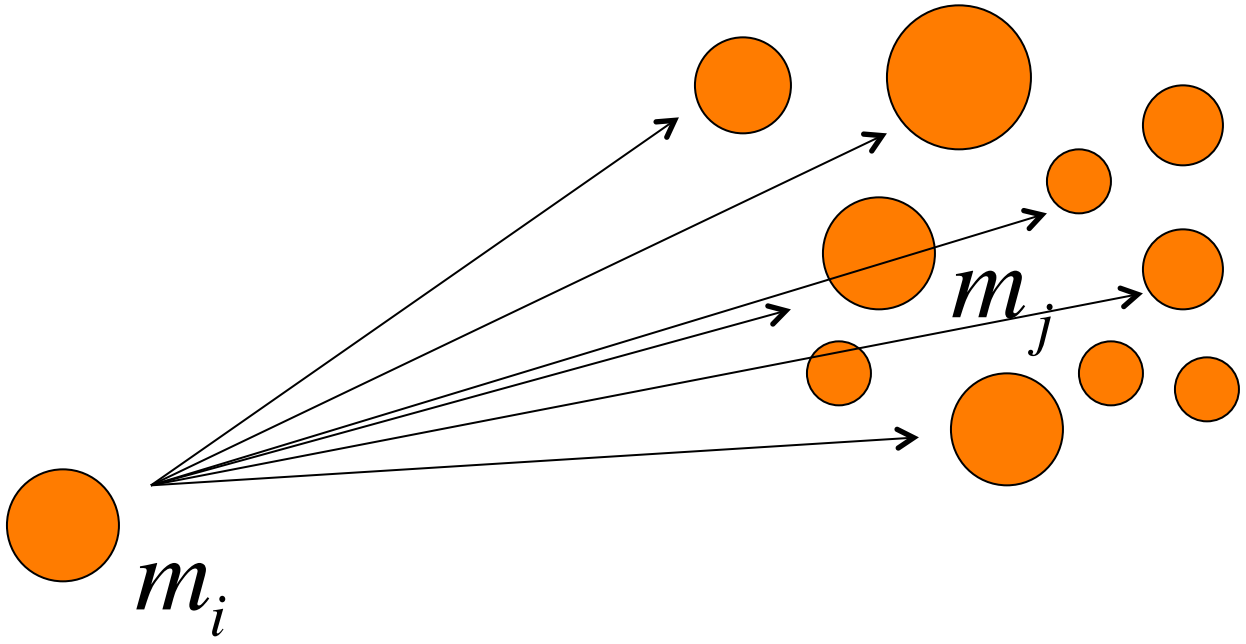
Método de Monte Carlo para calcular PI

4. Paralelizar el método de cálculo de PI utilizando directivas **MPI_Reduce**

Para ello los procesos se repartiran el número de puntos a evaluar en el círculo y el contador será sumado luego en el nodo master.

5. Medir tiempos totales de ejecución para np=2,4,8,16 y demostrar que el tiempo disminuye con np en un gráfico. Uilize *num_trials* = 10^8 para esta prueba.

Ejercicio 11: Problema de N-cuerpos



$$F_i = -Gm_i \sum_{\substack{1 \le j \le N \\ j \neq i}} \frac{m_j r_{ij}}{\left\| r_{ij} \right\|^3}$$

Ejercicio 11: Problema de N-cuerpos

Gravitación en N-cuerpos

Se trata de calcular las fuerzas gravitatorias de cuerpos en el espacio (planetas, estrellas, galaxias), con lo que se obtendrán posiciones y velocidades de los cuerpos a través del tiempo.

1. Cree una estructura o clase `CuerpoCeleste`, con miembros para posición, velocidad y masa

```
Class CuerpoCeleste
{
public:
float pos[3][N];
float vel[3][N];
float m[N]
}
```

Ejercicio 11: secuencial

2. Genere un código en serie que calcule la fuerza entre N cuerpos, segun:

```
int main (int arg, char
**argv)
{
…
// define class galaxy
Nbody galaxy;
…
// initialize properties
galaxy.init();
…
// integrate forces
galaxy.integr();
return 0;
}
```

```
void integr ()
{
…
// measure CPUtime
start=clock();

force(n, pos, vel, m, dt)

// measure CPUtime
end = clock();
cpuTime= difftime(end,start)/
(CLOCKS_PER_SEC)
…
}
```

Ejercicio 11: secuencial

Función de cálculo de fuerzas:

Se realiza en dos bucles de 0 a n-1 y se acumulan los valores de la aceleración. Recuerde que $f_i = m_i \times a$

```
void force(int n, float pos[][..],….,float
vel[][..], float m[..], float dt)
{
 // sume over i
  for (int i=0; i<n; i++)
  {
    float my_r_x = pos[0][i];
    ....
 // sume over j
    for (int j=0; j<n; j++)
    {
        if(j!=i)   // avoid i=j
        {
// compute accelerations
    float d = pos[0][j]-my_r_x; // 1 flop

    a_x += G*m[j]/(d*d);  // 4 flops
        …
        }
    }
```

Ejercicio 11: secuencial

3. Incluya la función `galaxy.integr()` en un bucle de tiempo *for(t=0;t<t_f;t=t+dt)*, donde t_f=1 es el tiempo final (dt=0.001)

Note que para que una simulación sea lo mas exacta posible se debe aproximar el movimiento en intervalos Δt pequeños.

Ejercicio 11: secuencial

4. Cálculo de velocidad y posición:

Utilizando la aceleración *a_x, a_y, a_z* luego de cada iteración en el tiempo, se puede calcular la nueva velocidad *vel* y la posición *pos*

```
//update velocities
vel[0][i] += a_x*dt; // 2 flops
// update positions
pos[0][i] += vel[0][i]*dt; // 2 flops
  }
}
```

Ejercicio 11: secuencial

4. Inicialize valores random para m, pos y vel entre 0 y 1, y ejecute el código en serie para N=1000

5. Paralelize el código repartiendo el número total de cuerpos en tamaños **N/p**, donde p es el número total de procesos. Para ello defina los límites adecuados en los argumentos de la función `force()`, o bien modifique los límites en el bucle `for()` dentro de la función. Utilize **MPI_Reduce** para sumar los resultados de las aceleraciones parciales para cada cuerpo.

Ejercicio 11: secuencial

6. Ejecute el código en paralelo para np=2,4,8,16 y N=1024, 2048,4096,8192,16384 midiendo tiempos de ejecución para cada combinación. Grafique una curva para cada N en un gráfico de tiempo (en segundos) vs. np. Que tipo de relación tienen T(np) ?

7. Opcional: calcule el número total de Flops por iteración y el rendimiento según $P(N,T)=Flops*N^2/T$ Grafique P vs. np para cada N, y determine el rendimiento por CPU segun P(np)=P(N,T)/np