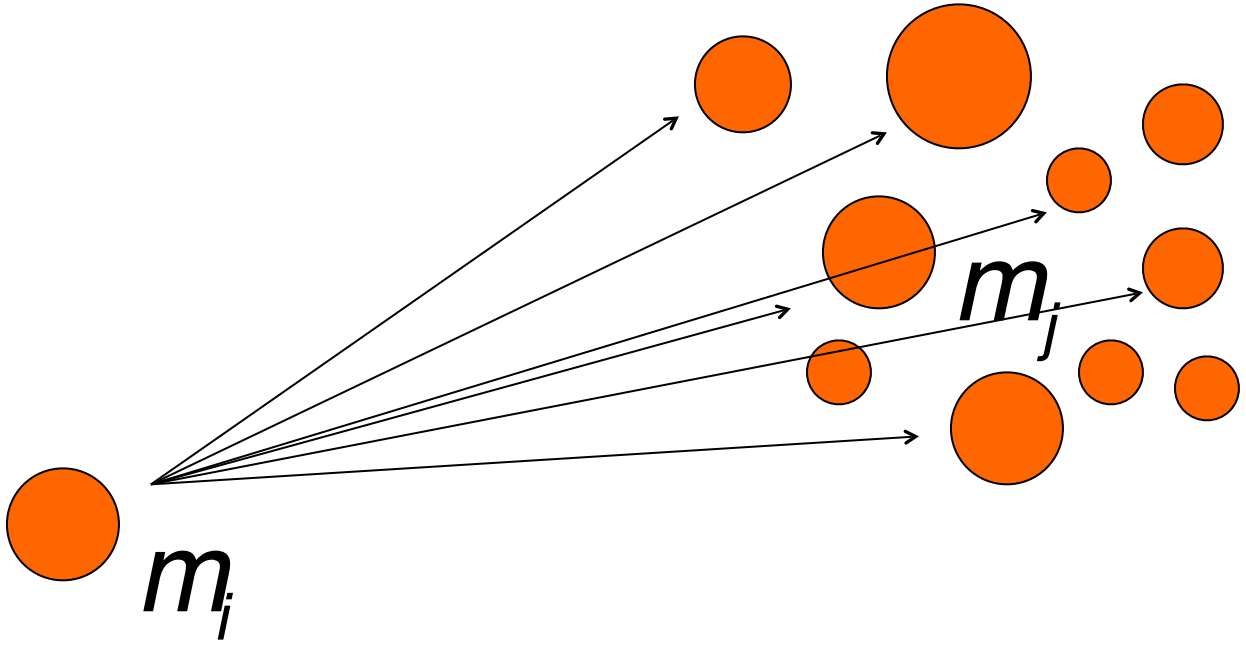
Programación Paralela

MPI vs OPENMP

Prof. J.Fiestas

Ejercicio : 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:

Se muestra la estructura de un código secuencial de 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.

El código secuencial consiste de 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 :

El main() tiene la siguiente estructura básica

```
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 :

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 $f_i=m_i$ x 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 :

La función `galaxy.integr()` está en un bucle de tiempo `for(t=0;t<t_f;t=t+dt)`, e.g. 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 : secuencial

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`

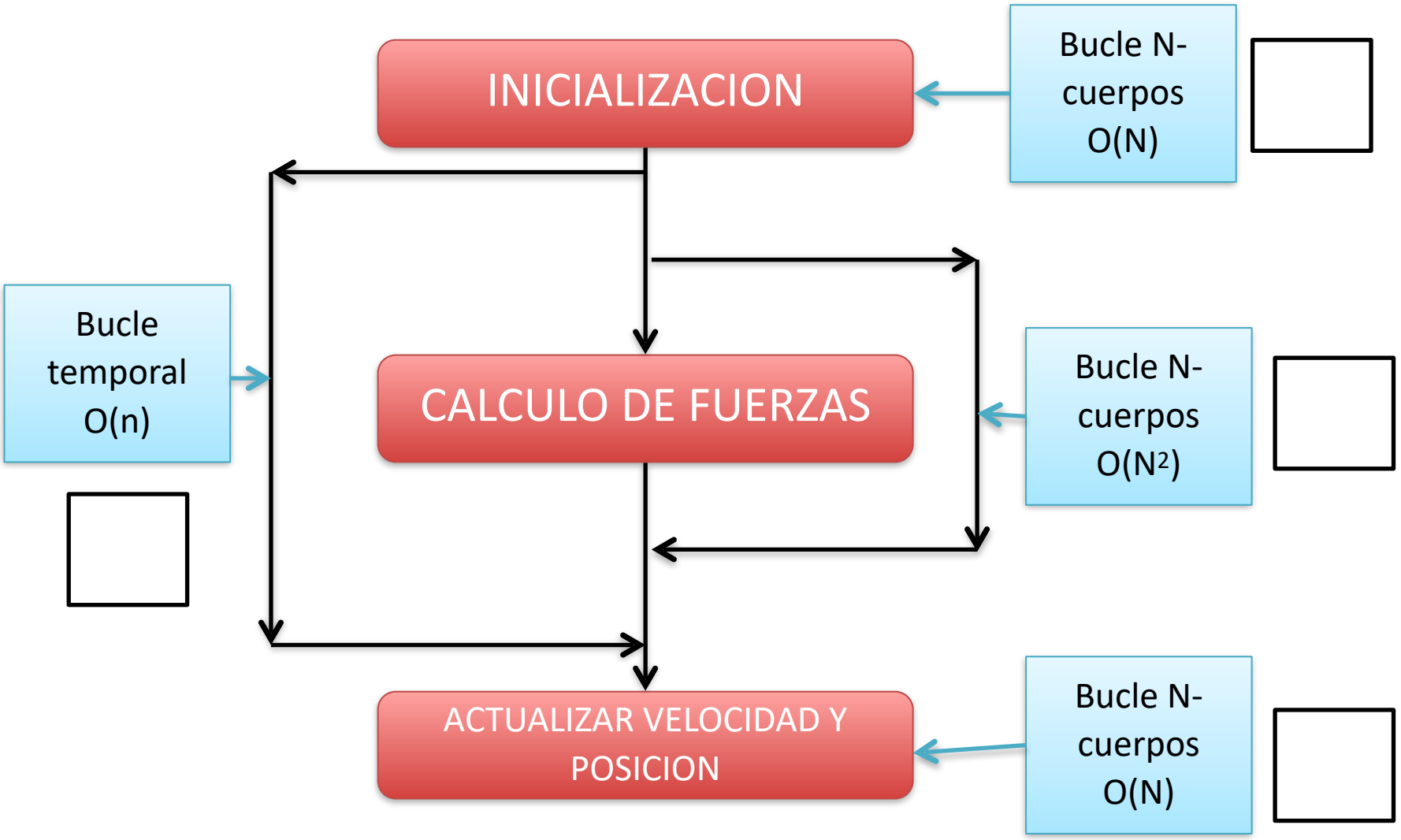
```
//update velocities
vel[0][i] += a_x*dt; // 2 flops
// update positions
pos[0][i] += vel[0][i]*dt; // 2 flops
  }
}
```

Ejercicio:

1. Evalue la mejor técnica de resolver el problema de N-cuerpos mostrado utilizando tecnicas en paralelo compartidas y/o distribuidas (MPI, OMP, hibrida)

- Llene los casilleros en blanco del siguiente diagrama de flujo con los acrónimos MPI / OMP
- Argumente las ventajas de cada una de ellas y decida CUALITATIVAMENTE sobre la mejor técnica a utilizar
-

Algoritmo N-cuerpos :



Ejercicio :

2. Basado en los métodos elegidos, diseñe un pseudocódigo en paralelo 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 :

3. Defina una fórmula para calcular el número total de Flops.
Estime la complejidad, speedup y eficiencia del algoritmo y comente acerca de su escalabilidad