

Computación Paralela y Distribuida

2022-II

José Fiestas

02/09/22

Universidad de Ingeniería y Tecnología
jfiestas@utec.edu.pe

Práctica Grupal Dirigida 02:
Unidad 2: PRAM
(PD02: 2 pts, Tarea 2: 4 pts.)

Ejercicio 1: N-cuerpos (1 pt)

Considere el siguiente código de N-cuerpos, y el diagrama en la figura (abajo)

```
class Nbody{
public:
float pos[3][n];
float vel[3][n];
float m[n];
}
int main(int arg, char**argv){
// clase galaxy
Nbody galaxy;
...
// inicializacion
galaxy.init();
// calculo de fuerzas

galaxy.integr();
}

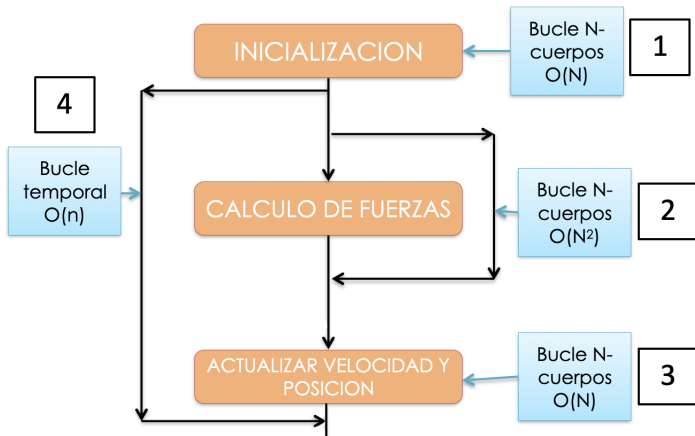
void integr() {
...
// medir CPU time
start=clock();
force(n,pos,vel,m,dt);
// medir CPU time
end=clock();
cpuTime=difftime(end,start)/(CLOCKS_PER_
...
}
```

Ejercicio 1 (cont.)

```
void force(int n, float pos[][3], float vel[][3] , float m[], float dt) {
    ...
    // suma en i
    for (int i=0;i<n;i++)
    {
        float my_r_x=pos[i][0]; // considerar coordenadas y, z
        // suma en j
        for (int j=0;j<n;j++)
        {
            if(j!=i) // evitar i=j
            {
                //calcular aceleracion
                float d=pos[j][0]-my_r_x; // 1 FLOP
                a_x+= G*m[j]/(d*d); // 4 FLOP
                ...
            }
        }
    }
    // actualizar velocidades (considerar coordenadas y, z)
    for (int i=0;i<n;i++) {
        vel[i][0] += a_x*dt; // 2 FLOPS
        // actualizar posiciones (considerar coordenadas y, z)
        pos[i][0] += vel[i][0] *dt; // 2 FLOPS
    }
```

Ejercicio 1 (cont.)

Algoritmo N-cuerpos :



Ejercicio 1 (cont.)

- a) Desarrolle un algoritmo en paralelo bajo el paradigma de memoria compartida usando el formalismo PRAM (0.5 pts)
- b) Desarrolle un algoritmo en paralelo bajo el paradigma de memoria distribuida usando el formalismo PRAM. Utilice operaciones de comunicación entre procesos como **Broadcast** (copia del maestro a todos los procesos), **Reduction** (calcula el resultado final en el maestro desde resultados parciales de cada proceso) y/o **Send/Receive** (envío de información de un proceso a otro), donde lo considere necesario (0.5 pts)
- c) Estime $T(n)$ y $W(n)$ en cada uno de ellos y decida cuál es el mejor algoritmo para este problema. Utilice los criterios discutidos en clase de eficiencia y escalabilidad (0.5 pts)

Ejercicio 2: Prim (1 pt)

Considere la complejidad del algoritmo de Prim en paralelo

$$O\left(\frac{n^2}{p} + n \log(p)\right)$$

- Calcule la cantidad óptima de procesos que minimiza el tiempo de ejecución para este modelo. ¿Cuál es la complejidad mínima de Prim en paralelo para éste caso?
- Compare este resultado con el caso en que se utilicen $p = O(n/\log n)$ procesos.

Ejercicio 2: Prim (1 pt)

- Discuta cuál algoritmo tendría un menor tiempo de ejecución entre ámbos.
- Analice speedup y eficiencia para ambos casos

Ejercicio 3: DAG (1 pt)

Dado el siguiente procedimiento

```
double sistema()
{
    double A[n][n], B[n][n], a, b, x, y, z;
    ini(A, B);
    a = tarea_1(A);
    b = tarea_2(B);
    s1 = suma_prefijos1(B, a);
    s2 = suma_prefijos2(B, a, b);
    z = s1 + s2;
    return z;
}
```

La función `ini()` recibe dos matrices y las inicializa con valores generados internamente, con un costo de $2n^2$ flops. Los parámetros del resto de funciones son sólo de entrada (no se modifican). Las tareas 1 y 2 tienen un costo $O(n^2)$, la suma de prefijos tiene un costo $O(n)$.

Ejercicio 3: DAG

- 3.1) Genere el DAG y señale su grado máximo de concurrencia (cantidad máxima de tareas simultáneas)
- 3.2) Genere un PRAM que resuelva el problema utilizando el paradigma de memoria compartida
- 3.3) Determine la complejidad secuencial, en paralelo, y el speed-up en función a n y p . Calcule la eficiencia del código, para una cantidad mínima de hilos.

Ejercicio 4: subsecuencia máxima (1 pt)

Relevante en bioinformática para estudios de secuencias genómicas es determinar la subsecuencia máxima de elementos en un array. Dado el siguiente algoritmo secuencial (divide y vencerás) de solución del problema:






Input: array A de tamaño r

Output: subarray maximo

```
MaxSubarray(A,p,r)
if p==r return A[1]
q=(p+r)/2
L=MaxSubarray(A,p,q-1) // array izquierda
R=MaxSubarray(A,q+1,r) // array derecha
C=MaxSubarrayCentro(A,p,q,r) // array que pasa por el centro
return MAX(L,R,C)
```

Determine la complejidad secuencial de cada paso

Elabore un PRAM para el mismo problema y determine $T(n)$, $S(n)$ y $E(n)$. Discuta la eficiencia del algoritmo paralelo con estos resultados

-  David B. Kirk and Wen-mei W. Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013. isbn: 978-0-12-415992-1.
-  Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014.
-  Peter S. Pacheco. *An Introduction to Parallel Programming*. 1st. Morgan Kaufmann, 2011. isbn: 978-0-12-374260- 5.
-  Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. 1st. McGraw-Hill Education Group, 2003. isbn: 0071232656.
-  Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Program- ming*. 1st. Addison-Wesley Professional, 2010. isbn: 0131387685, 9780131387683.