

Esquemas paralelos

Metodología de la Programación Paralela

Jesús Sánchez Cuadrado (jesusc@um.es)

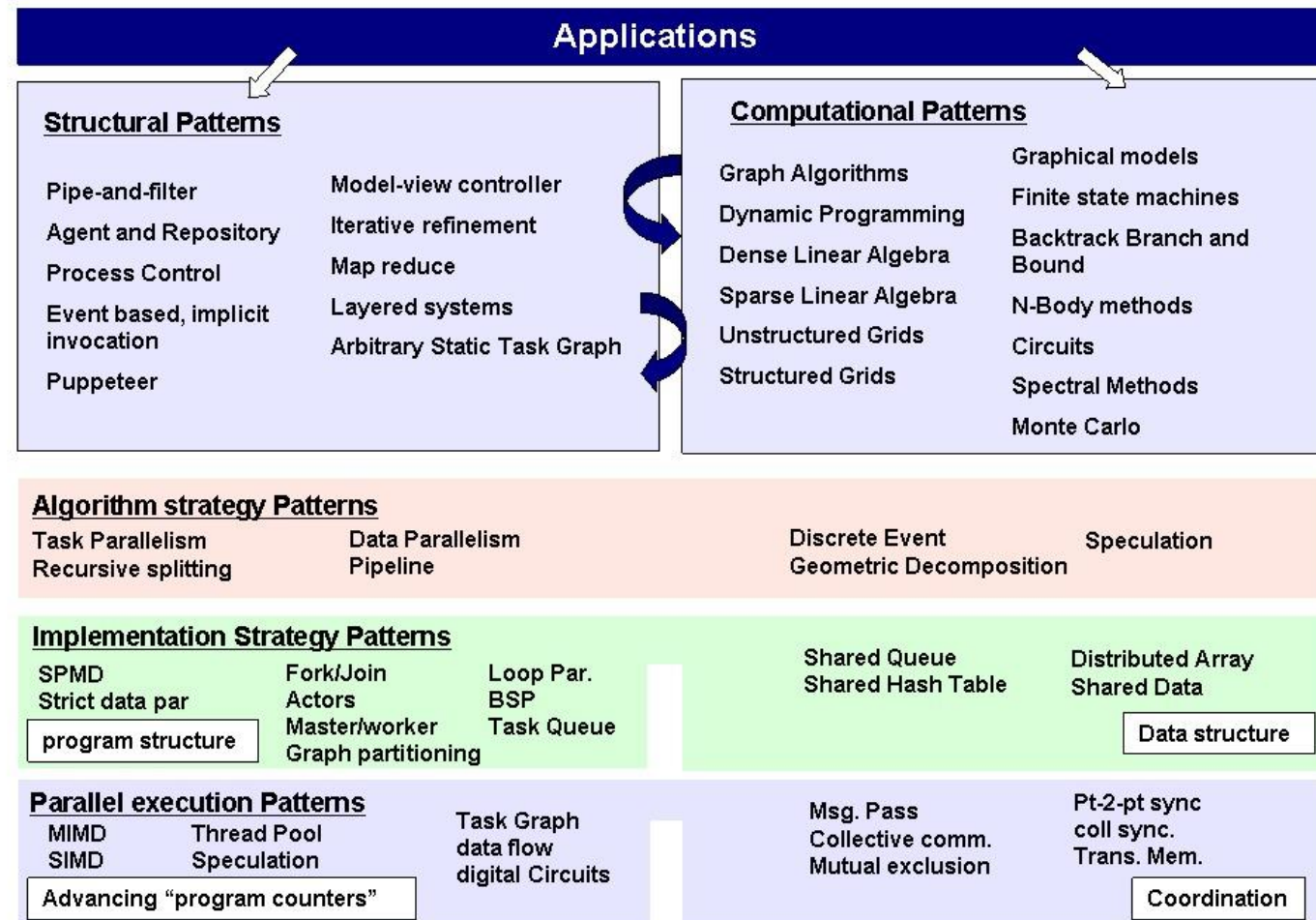
Curso 2020/21

Tipos de esquemas

- Podemos considerar esquemas de distintos tipos:
- Descomposición del trabajo:
 - Paralelismo de datos.
 - Particionado de datos.
 - Algoritmos relajados.
- De paralelismo basado en dependencias de datos:
 - Paralelismo síncrono.
 - Dependencias en árbol o grafo.
 - Pipeline.
- De paralelización de esquemas secuenciales:
 - Divide y Vencerás.
 - Programación Dinámica.
- Recorridos de árboles: Backtracking y Branch and Bound.
- De múltiples tareas o trabajadores:
 - Bolsa de tareas.
 - Granja de procesos.
 - Maestro-Esclavo.

Otras fuentes de patrones

- Our pattern language: <https://patterns.eecs.berkeley.edu/>



Particionado y paralelismo de datos

Paralelismo y particionado de datos

- Aplicable cuando todos los datos se tratan de igual manera
 - Aplicable a GPUs
- Ejemplo:
 - Algoritmos numéricos donde los datos están en vectores o matrices
 - Posible procesamiento vectorial
 - Paralelismo asignando partes distintas del array a distintos elementos de proceso (directivas OpenMP)
- Memoria Compartida (Paralelismo de datos):
 - Distribución del trabajo entre los hilos
 - Paralelización automática o implícita
- Memoria Distribuida (Particionado de datos):
 - Distribución de los datos a los procesos
 - Paralelización explícita

Ejemplo – Suma de N números

- Suma de N números

```
int s = 0;
for(int i = 0; < n; i++) {
    s = s + a[i];
}
```

- Posible vectorización con simd.
- Paralelización automática: con opción de compilación si no hay dependencia de datos o el compilador detecta que se puede resolver

Ejemplo – Suma de N números

- Paralelismo implícito
 - El programador especifica cuáles son los datos
 - El sistema de decide cómo se reparten
- En OpenMP
 - Clausula `schedule` permite indicar cómo distribuir el trabajo: bloques contiguos, asignación cíclica, dinámica

```
int s = 0;
#pragma omp parallel for private(i) reduction(+:s)
for(int i = 0; < n; i++) {
    s = s + a[i];
}
```

Ejemplo – Suma de N números

- Paralelismo explícito

```
void sumaparcial(double *a, int n, int p) {  
    int s=0;  
    for(j=0; j<n/p; j++)  
        s=s+a[j];  
    a[0]=s;
```

```
void sumatotal(double *a, int n, int p) {  
    double s=0;  
    for(j=0; j<p; j+=n/p)  
        s = s + a[j];  
    return s;  
}
```

$$t(n, p) = \frac{n}{p} + p$$

```
#pragma omp parallel for private(i,s)  
for(i=0; i<p; i++)  
    sumaparcial(&a[(i*n)/p], n, p);  
S = sumatotal(a, n, p);
```


Ejemplo – Ordenación por rango

- Para cada elemento, contar cuántos hay menores que el
 - El rango nos dice la posición que ocupa el elemento ordenado
 - Coste $O(n^2)$
 - Fácilmente paralelizable

```
for(i=0; i<n; i++)  
    for(j=0; j<n; j++)  
        if (a[i]>a[j] || ((a[i] == a[j]) && (i > j)))  
            r[i]+=1;  
  
for(i = 0; i < n; i++)  
    b[r[i]] = a[i];
```

Ejemplo – Ordenación por rango

- Paralelismo implícito
 - Se divide el trabajo de cada bucle

```
#pragma omp parallel for private(i,j)
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
        if (a[i]>a[j] || ((a[i] == a[j]) && (i > j)))
            r[i] += 1;
```

```
#pragma omp parallel for private(i)
for(i = 0; i < n; i++)
    b[r[i]] = a[i];
```

$$t(n, p) = \frac{n^2}{p} + \frac{n}{p}$$

Ejemplo – Ordenación por rango

- Paralelismo implícito
 - Se divide el trabajo de cada bucle

```
#pragma omp parallel for private(i,j)
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
        if (a[i]>a[j] || ((a[i] == a[j]) && (i > j)))
            r[i] += 1;
```

```
#pragma omp parallel for private(i)
for(i = 0; i < n; i++)
    b[r[i]] = a[i];
```

¿Es posible colapsar los dos bucles?

Ejemplo – Ordenación por rango

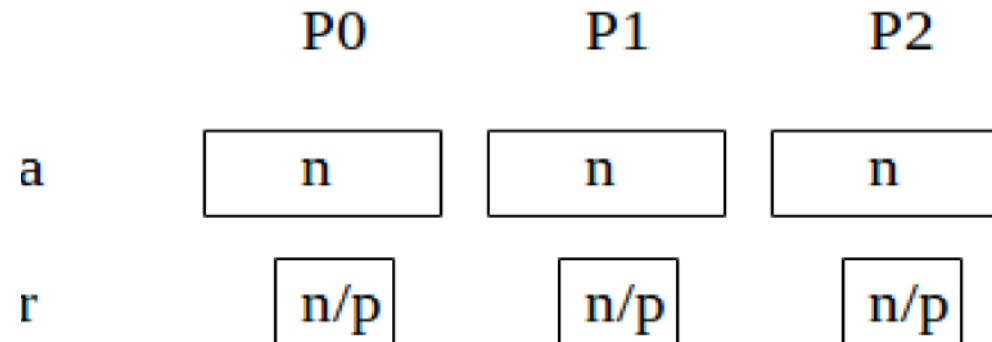
- Paralelismo implícito
 - Se divide el trabajo de cada bucle

```
#pragma omp parallel for private(i,j)
for(i=0; i<n; i++)
    for(j=0; j<n; j++)
        if (a[i]>a[j] || ((a[i] == a[j]) && (i > j)))
            r[i] += 1;

// En este punto la posición del elemento i
// ya ha sido calculada
b[r[i]] = a[i];
}
```

Ejemplo – Ordenación por rango

- Paralelismo explícito
 - Asignación manual del trabajo a los hilos



Ejemplo – Ordenación por rango

- Paralelismo explícito

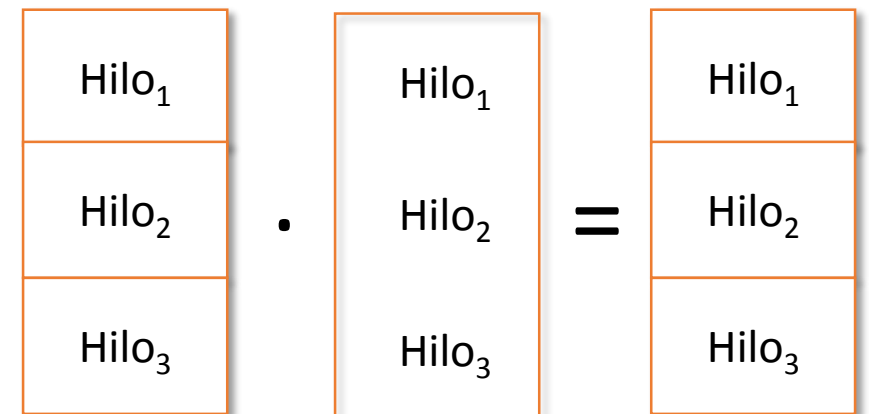
```
void rangoparcial(int *a, int *r, int *b,
                 int hilo, int n, int p) {
    for(int i = hilo * n / p; i < (hilo + 1) * n / p; i++) {
        for(j = 0; j < n; j++)
            if (a[i]>a[j] || ((a[i] == a[j]) && (i > j)))
                r[i] += 1;
        b[r[i]] = a[i];
    }
}
```

```
#pragma omp parallel for private(i)
for(i = 0; i < THREADS; i++)
    rangoparcial(a, r, b, n, i, THREADS)
```

Ejemplo – Multiplicación de matrices

- Con paralelismo implícito

```
#pragma parallel for private(i,j,k)
for(i=0; i<n; i++)
  for(j=0; j<n; j++) {
    c[i,j] = 0;
    for(k=0; k<n; k++)
      c[i,j]=c[i,j]+a[i,k]*b[k,j];
  }
```



Ejemplo – Multiplicación de matrices

- Con paralelismo explícito

```
#pragma omp parallel for private(i)
for(i=0;i<p;i++)
    multiplicar(c,a,b,i)

void multiplicar(c,a,b,i) {
    for(j=(i*n)/p; j < ((i+1)*n)/p; j++)
        for(k=0;k<n;k++) {
            c[j,k]=0;
            for(l=0;l<n;l++)
                c[j,k]=c[j,k]+a[j,l]*b[l,k];
        }
}
```


Otros ejemplos de paralelismo de datos

- Producto escalar de dos vectores x e y de tamaño n :
 - Similar a la suma de n datos.
 - Asignación de bloques de tamaño n/p de x e y a los p hilos.
 - Suma (secuencial o paralela) de los productos parciales.
- Producto matriz-vector:
 - Dos bucles, paralelización del más externo.
 - Posible trabajo por bloques.

Particionado de datos

- Idea similar al paralelismo de datos para Memoria Distribuida
 - Además de dividir el trabajo hay que distribuir los datos
 - Implementación típica usando Paso de Mensajes
- El espacio de datos se divide en regiones adyacentes:
 - Se asignan a procesos distintos
 - Posible intercambio de datos entre regiones adyacentes
 - Más semejante al Paralelismo de Datos explícito
- Para obtener buenas prestaciones hay que intentar que el volumen de computación entre comunicaciones sea grande.
 - Paralelismo de grano grueso, efecto volumen-superficie.

Particionado de datos

- Ejemplo – Multiplicación de matrices
 - Volumen de computación n^3
 - Volumen de distribución de datos n^2
 - Cada hilo realiza el mismo cálculo que en la versión OpenMP

Ordenación por rango

```
void ordenrango(int *a, int n, int *r, int pid, int numprocs) {  
    int i, j, k, t1 = n / numprocs; *b = NULL;  
  
    MPI_Bcast(a, n, MPI_INT, 0, MPI_COMM_WORLD);  
  
    for(k = 0, i = hilo * n / p; i < (hilo + 1) * n / p; i++, k++) {  
        for(j = 0; j < n; j++)  
            if (a[i]>a[j] || ((a[i] == a[j]) && (i > j)))  
                r[k] += 1;  
    }  
  
    ...  
}
```

Ordenación por rango

```
void ordenrango(int *a, int n, int *r, int pid, int numprocs) {  
    ...  
    if (pid == 0) {  
        for(i = 1; i < numprocs; i++) {  
            MPI_Recv(&r[t1 * i], t1, MPI_INT, i, 10, ...);  
            b = (int *) malloc(sizeof(int) * n);  
            for(i = 0; i < n; i++)  
                b[r[i]] = a[i];  
            for(i = 0; i < n; i++)  
                a[i] = b[i];  
            free(b);  
        }  
    } else {  
        MPI_Send(r, t1, MPI_INT, 0, 10, MPI_COMM_WORLD);  
    }  
}
```

Ordenación por rango

- Coste
 - Volumen de computación n^2
 - Volumen de distribución de datos n
 - Todos los procesos necesitan el array que se va a ordenar completo

$$\underbrace{(p-1)(ts + tw \cdot n)}_{\text{Distribución inicial}} + \underbrace{tc \frac{n^2}{p}}_{\text{Cómputo}} + \underbrace{(p-1) \cdot \left(ts + \frac{tw \cdot n}{p} \right)}_{\text{Distribución final}}$$

Algoritmos relajados

Algoritmos relajados

- Cada elemento de proceso trabaja de manera independiente.
- No hay sincronización ni comunicación, salvo las de distribuir datos y recoger resultados.
- Buenas prestaciones en Memoria Compartida y Paso de Mensajes.
- A veces a costa de no utilizar el mejor algoritmo paralelo.
- Fáciles de programar.
- Difícil encontrar algoritmos que se adecuen estrictamente al esquema.

Ejemplo – Fractales: el conjunto de Mandelbrot

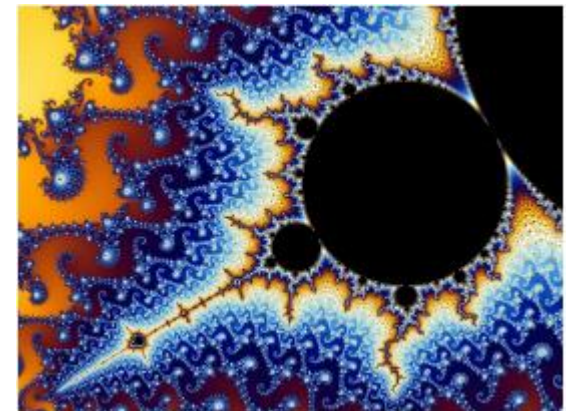
- Algoritmo “embarrassingly parallel”
 - Se crear un número N de tareas, totalmente independientes
 - Particionado estático
- Para cada punto c en el plano complejo
 - Calcular la función $\text{color}(c)$

$$\begin{cases} z_0 = 0 \in \mathbb{C} & (\text{término inicial}) \\ z_{n+1} = z_n^2 + c & (\text{sucesión recursiva}) \end{cases}$$

Los puntos cuya distancia al origen es superior a 2 ($x^2 + y^2 > 2$) no pertenecen al conjunto

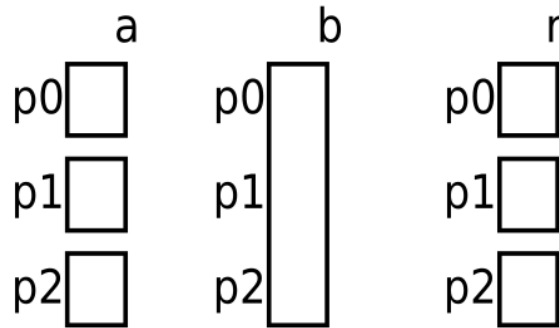
* https://es.wikipedia.org/wiki/Conjunto_de_Mandelbrot

* <https://www.dais.unive.it/~calpar/>



Ordenación por rango

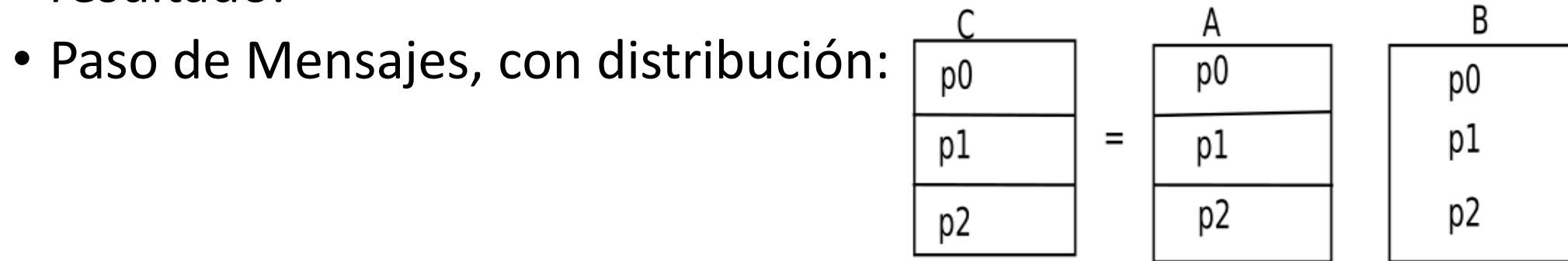
- Memoria Compartida: cada hilo calcula el rango de una parte de los elementos.
- Paso de Mensajes. Primero distribuir en la forma siguiente, y cada proceso calcula el rango de una parte de los elementos.



- Hay duplicación de datos pero se simplifica la programación y se obtienen buenas prestaciones.

Multiplicación de matrices

- Memoria Compartida: cada hilo calcula un bloque de filas de la matriz resultado.



- cada procesador calcula las filas de C correspondientes a las filas de A que contiene.
- No es necesaria sincronización ni comunicación
 - salvo la distribución y acumulación (¿qué funciones se utilizarían?),
 - pero es más costoso el envío inicial al repetirse B en cada proceso.

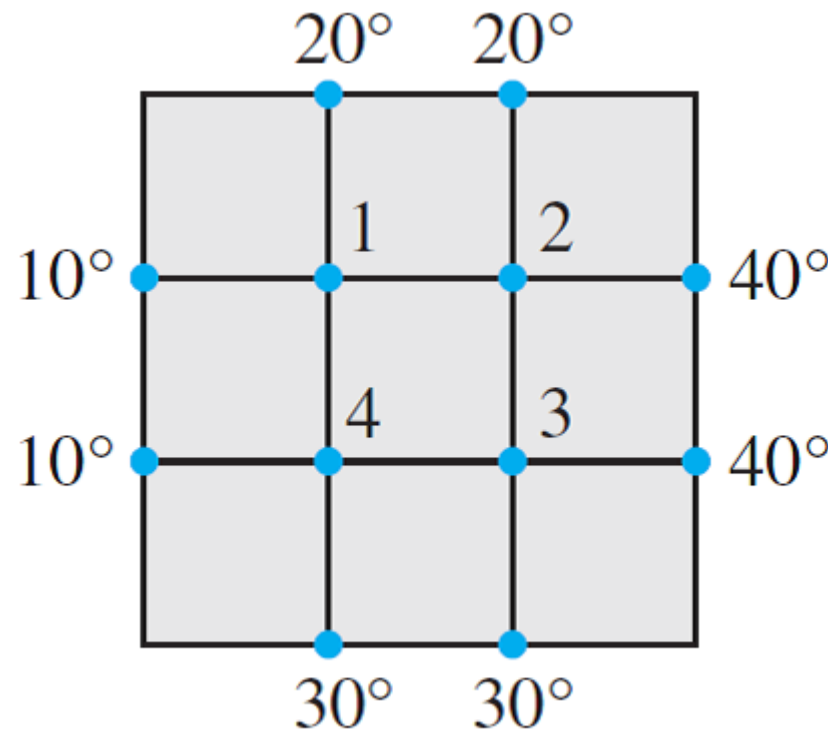
Descomposición geométrica/ Paralelismo síncrono

Características

- Se realizan iteraciones sucesivas:
 - Cada elemento de proceso realiza el mismo trabajo sobre una porción distinta de los datos.
 - Datos de una iteración se utilizan en la siguiente.
 - Tras cada iteración hay sincronización (local o global).
 - La iteración finaliza cuando se cumple algún criterio de convergencia o cuando se alcanza un número fijo de iteración (o tiempo límite).
- Las prestaciones están afectadas por la sincronización:
 - En Memoria Compartida buenas prestaciones.
 - En Paso de Mensajes bajan prestaciones por el coste de las comunicaciones.

Ejemplo – Iteración de Jacobi

- Por ejemplo, $f(x, y)$ el calor de un punto de una placa de metal
 - Dada una placa de metal para la que conocemos la temperatura de los bordes, calcular cuál es la temperatura en los puntos del interior.



Ejemplo – Iteración de Jacobi

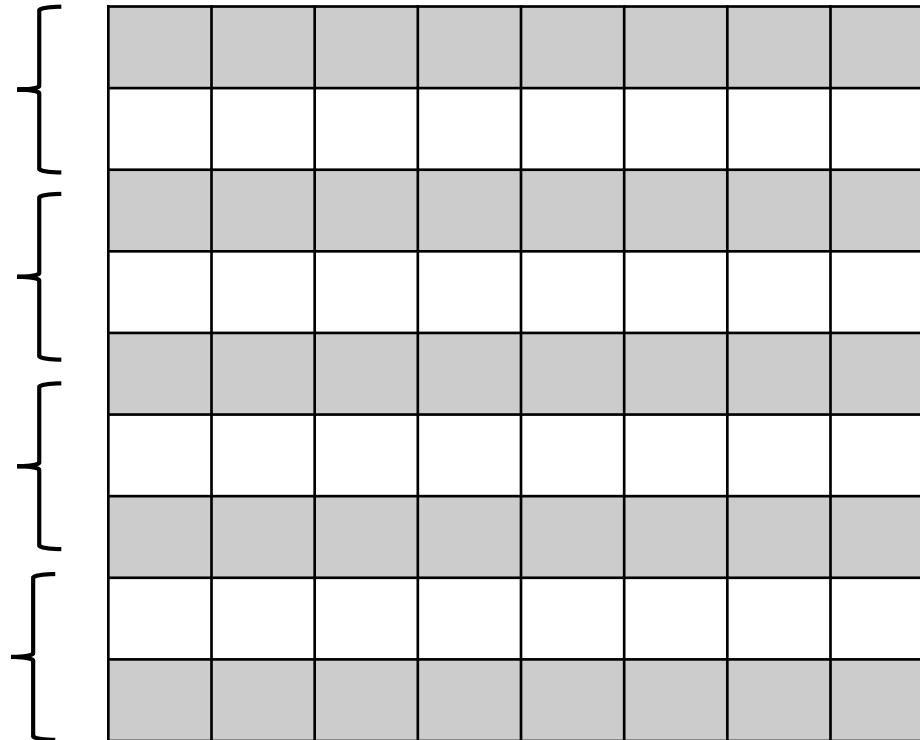
- Se calcula la ecuación de diferencias:

$$V^t(i, j) = \frac{V^{t-1}(i-1, j) + V^{t-1}(i+1, j) + V^{t-1}(i, j-1) + V^{t-1}(i, j+1)}{4}$$

- Converge gradualmente a una solución cada vez más precisa.
- Para obtener una solución más precisa aumentar el número de puntos.
- Una iteración tras otra secuencialmente, pero dentro de cada iteración paralelismo.
- Entre iteraciones se necesita comunicación local.
- Para comprobar condición de fin comunicación global.

Paralelismo implícito

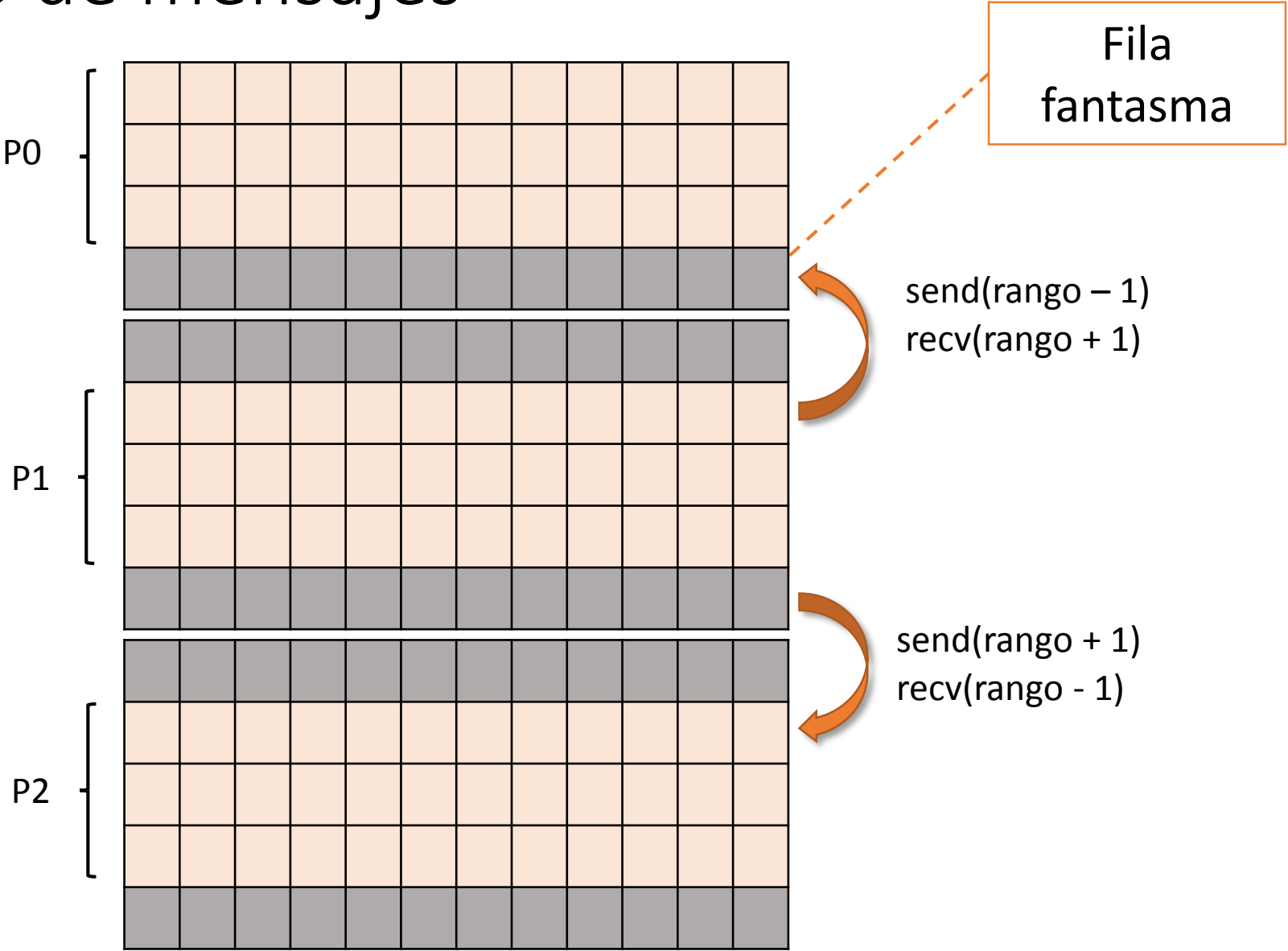
- Entre cada bucle paralelo, la sincronización es implícita
- Hay paralelismo de datos en los bucles
- División del trabajo



Paso de mensajes

- El array de datos n filas
- Cada proceso tiene un array con $tl = n/p + 2$ filas
- Las dos filas extras son para almacenar “los bordes”
 - Filas fantasma
- Los intercambios consisten en comunicarse los resultados del borde

Paso de mensajes



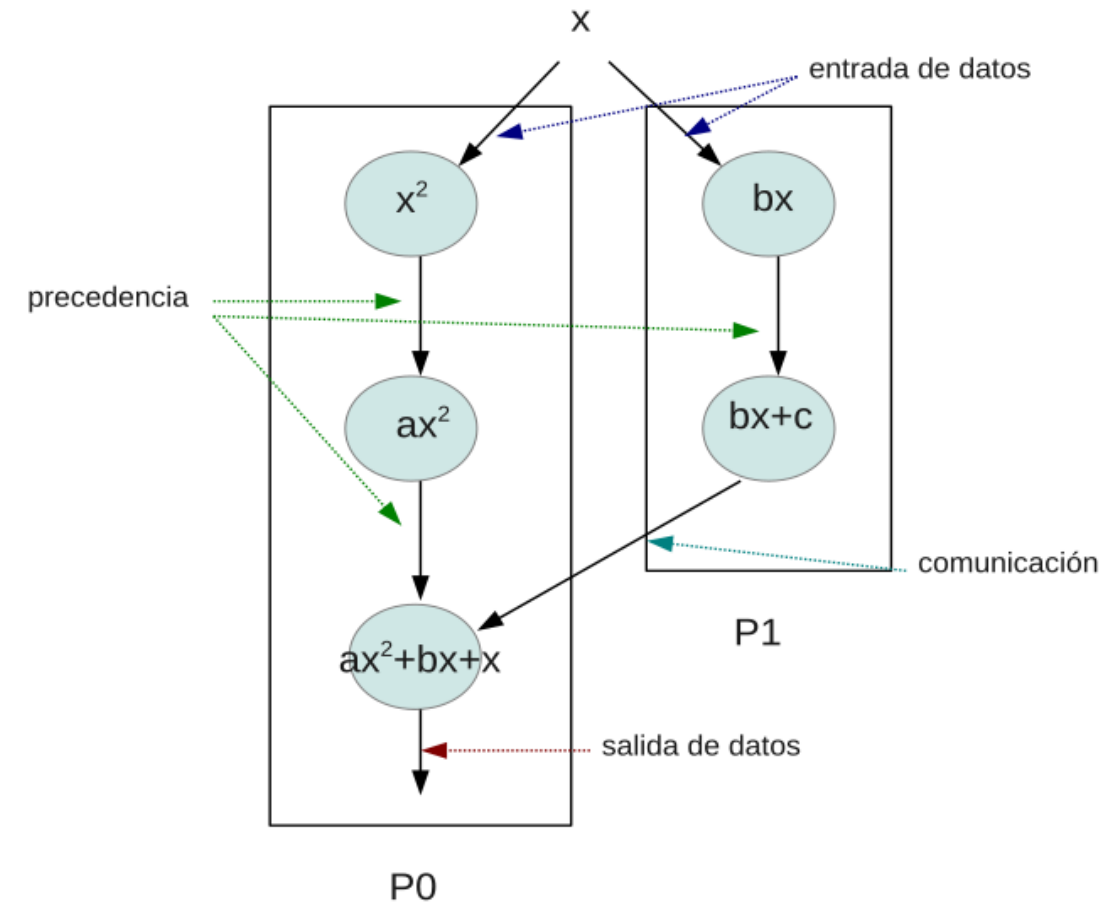
Paso de mensajes

- ¿Cómo se puede optimizar introduciendo comunicaciones asíncronas?

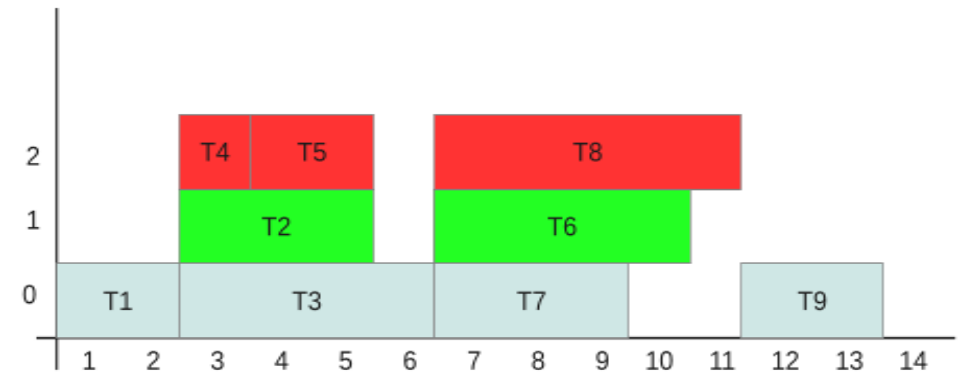
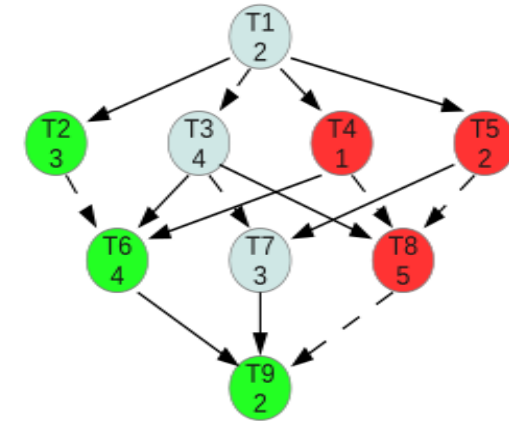
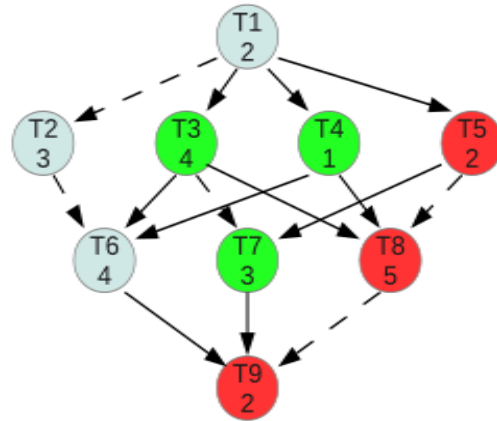
Dependencias en árbol o grafo

Grafos de dependencias

- Un Grafo de Dependencias es un grafo dirigido acíclico
- Nodos representan tareas
- Una arista de un origen a un destino representa que para poder realizarse la tarea destino tiene que haberse ejecutado la origen (posiblemente por una dependencia de datos).
- Los nodos pueden etiquetarse con un valor que representa el coste de la tarea. Las aristas pueden etiquetarse con valor que representa el coste de la comunicación



Asignación de tareas



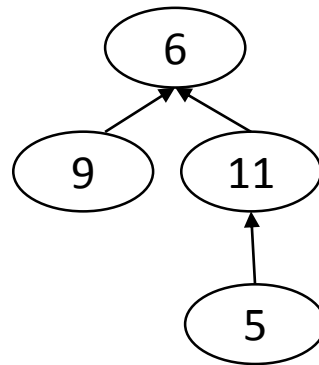
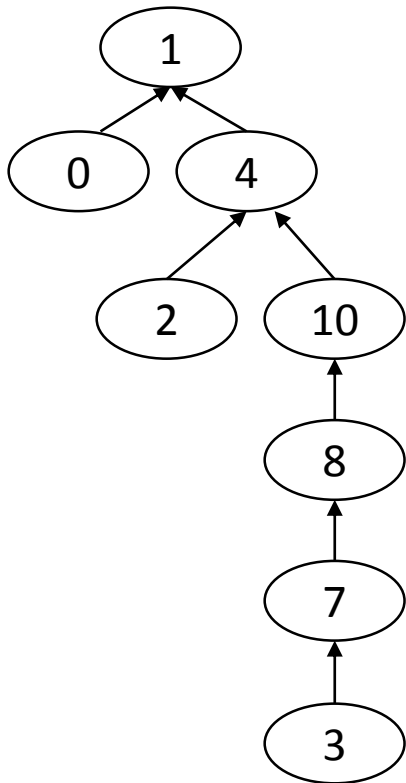
- Hay que asignar las tareas a los elementos de proceso.
- Distintas asignaciones pueden originar tiempos distintos.
- El problema de la asignación óptima es NP.

Asignación de tareas

- Dentro de un elemento de proceso hay que elegir el orden de ejecución de las tareas.
- Desbalanceo de la carga: balancear cálculo y comunicaciones.
- Tiempos de espera: por dependencias entre tareas.
- Si granularidad fina: muchas tareas, lo que posibilita el balanceo pero genera más comunicaciones.
- Si granularidad gruesa menos comunicaciones pero más difícil el balanceo.

Clases de equivalencia

- Relación de equivalencia expresada como árboles
 - Elementos de un conjunto que comparten cierta propiedad



Problema: encontrar el representante de la clase a la que pertenece cada nodo

Nodo	0	1	2	3	4	5	6	7	8	9	10	11
Padre	1	1	4	7	1	11	6	8	10	6	4	6

Clases de equivalencia

- Algoritmo secuencial
 - Sustituir en cada iteración el padre de un nodo por el padre de su padre

```
int cambio = 1;
while (cambio) {
    cambio = 0;
    for(int i = 0; i < n; i++) {
        if (array[i] != array[array[i]]) {
            array[i] = array[array[i]];
            cambio = 1;
        }
    }
}
```

Clases de equivalencia

- Algoritmo paralelo

```
int cambio = 1;
while (cambio) {
    cambio = 0;
    memcpy(tmp, array, n);

    #pragma omp parallel for private(i)
    for(int i = 0; i < n; i++) {
        if (array[i] != array[array[i]]) {
            tmp[i] = array[array[i]];
            cambio = 1;
        }
    }
}
```

Ejemplo

- Procesador de consultas (SQL) paralelo

```
SELECT nombre, apellidos, teléfono, direccion  
FROM persona P JOIN direcciones D ON P.dni = D.dni_persona  
WHERE edad < 18
```

