Tema 2
Modelos de programación paralela adaptados a la arquitectura

Nicolás Calvo Cruz
Dpto. de Arquitectura y Tecnología de los Computadores
@ncalvocruz
ncalvocruz@ugr.es

Objetivos

- Aprender a **encontrar** puntos para paralelizar un algoritmo
- Aprender a **descomponer** un algoritmo en tareas
- Aprender a identificar y respetar **dependencias** entre tareas
- **Agrupar** las tareas que se pueden realizar en paralelo
- Aplicar revisiones de **diseño**

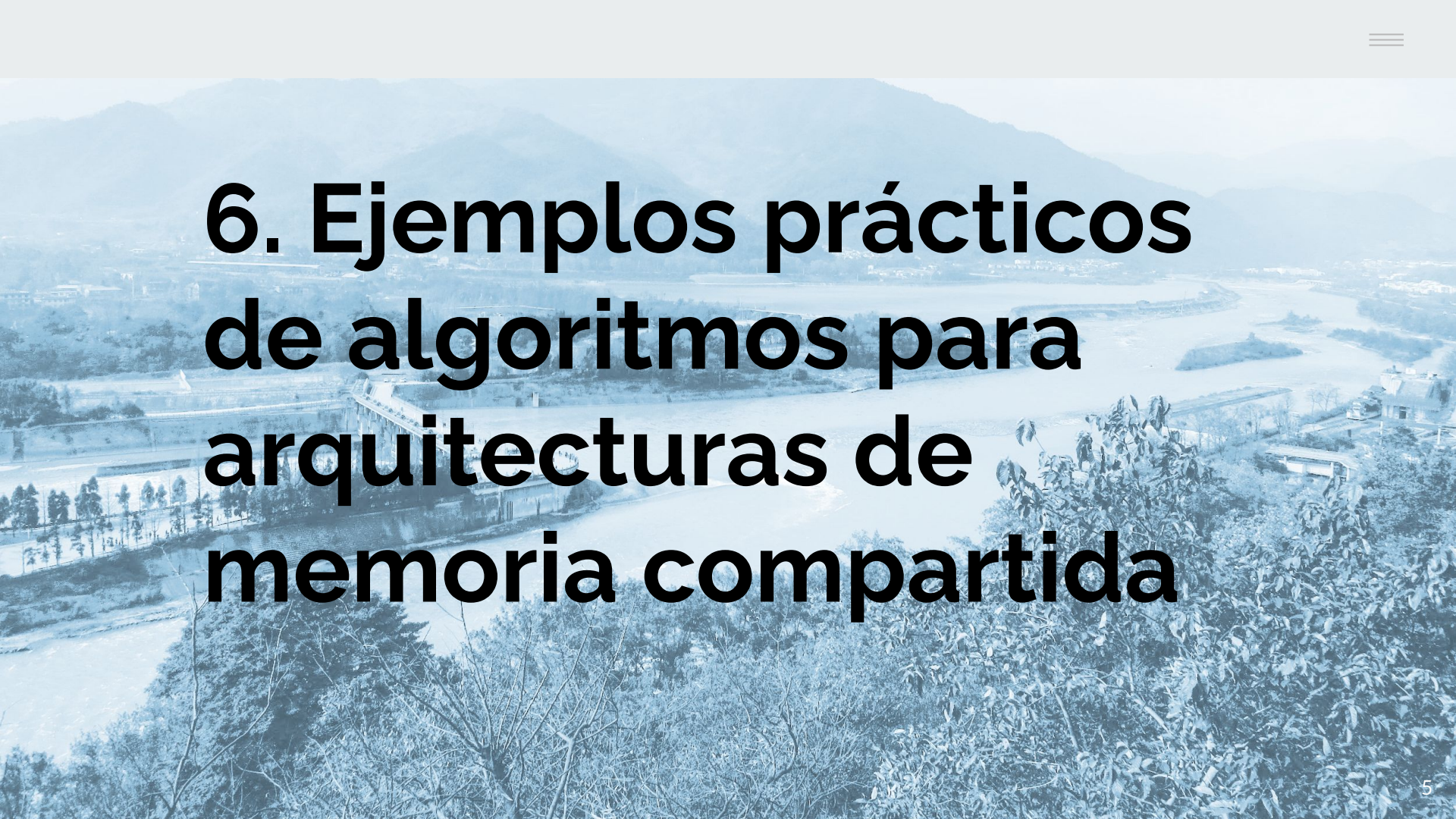
Motivación

1. Abordar **problemas** cada vez más **grandes**.

2. Obtener (mejores) soluciones en un **tiempo razonable**.

3. Ilustrar diferentes **enfoques** de la bibliografía para **paralelizar** algoritmos tradicionales.

Índice



6. Ejemplos prácticos de algoritmos para arquitecturas de memoria compartida

4. Estructuras de algoritmos más comunes (Recordatorio)

	Task Parallelism	Divide & Conquer	Descomposición Geométrica	Recursive Data	Pipeline	Event-Based Coordination
SPMD	****	***	****	**	***	**
Loop Parallelism	****	**	***			
Master/Worker	****	**	*	*	*	*
Fork/Join	**	****	**		****	****

4. Estructuras de algoritmos más comunes (Recordatorio)

	OpenMP	MPI	Java	CUDA (OpenCL)
SPMD	***	****	**	***
Loop Parallelism	****	*	***	*
Master/Worker	**	***	***	*
Fork/Join	***		****	**

6. Ejemplos prácticos: SPMD - Map-Reduction

Map: fase en la que el **trabajo** se divide **en partes** y se gestiona **independientemente**.

Reduction: fase en la que se **agrupan los datos en un solo**. La **combinación** se hace con una operación que tiene elemento neutro y cumple las propiedades asociativa y conmutativa, como la suma, la multiplicación…

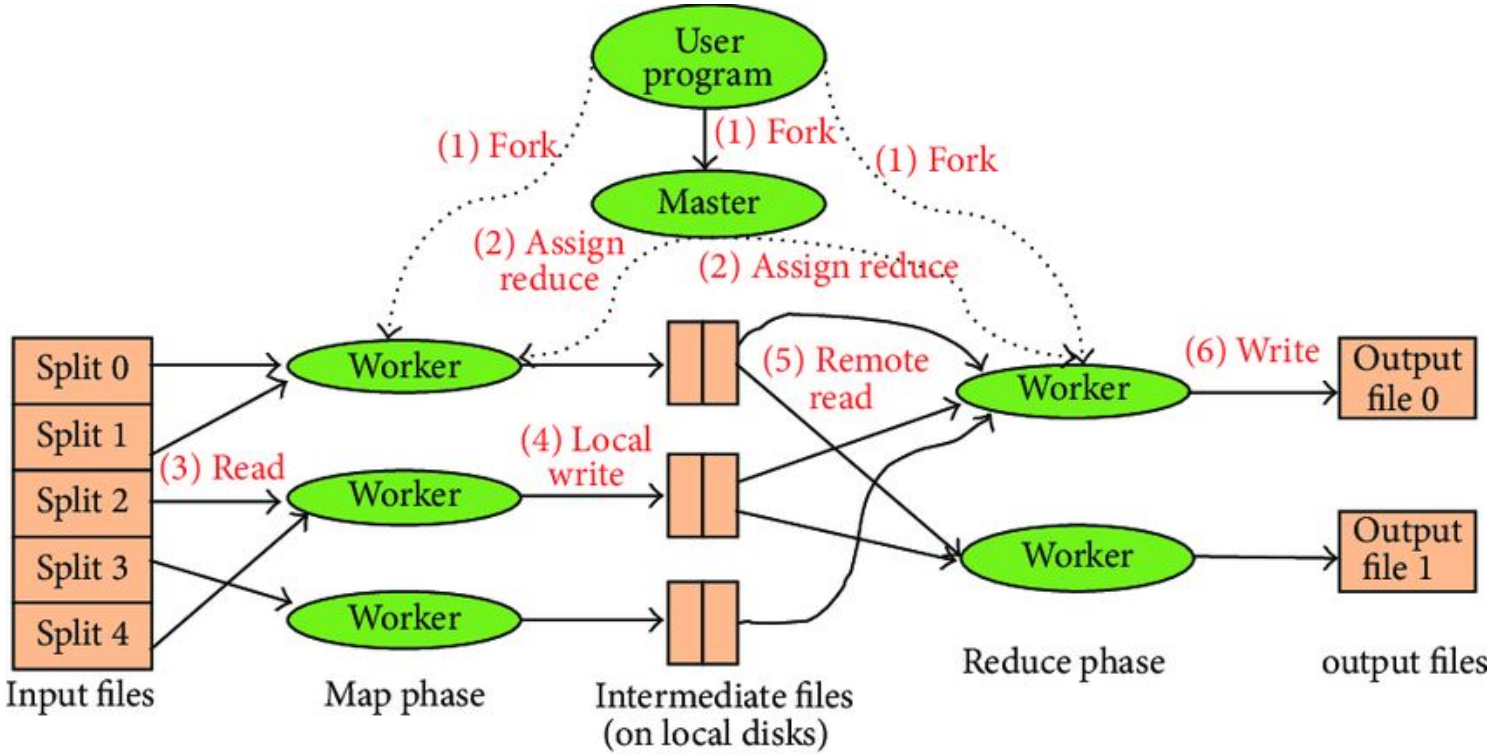
¿Donde se usa?

- Map-Reduce es el ejemplo más famoso, primero se mapea la información y luego se reduce. (**Contar palabras en Google**)
- Calcular frecuencias (**Histograma**)
- En el **mundo real**, es como funciona cualquier competición deportiva (**bracket de competición**)

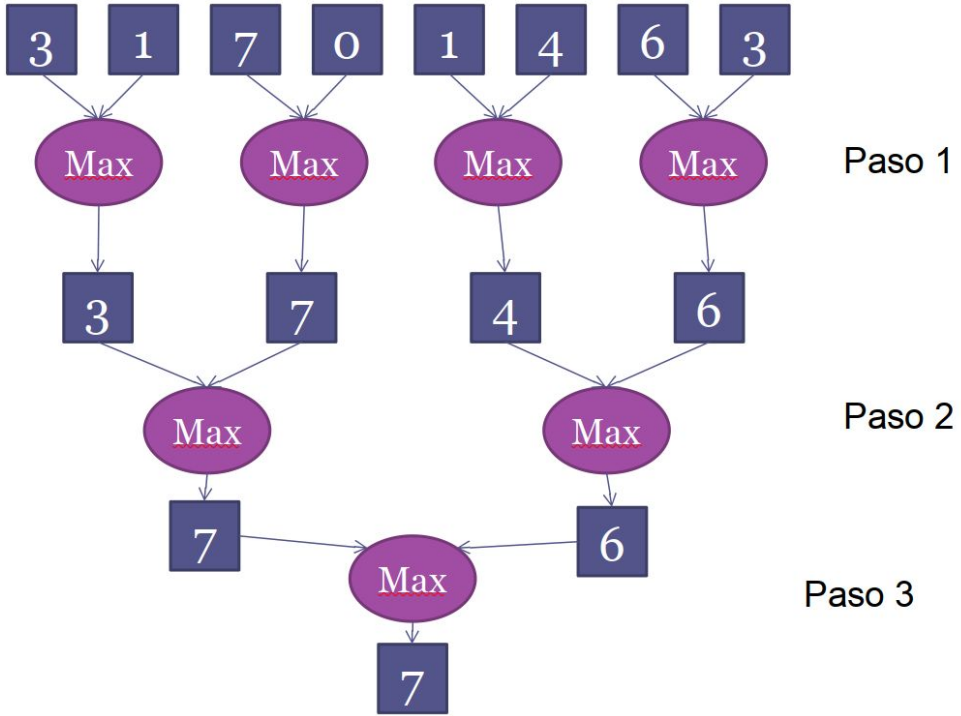
¿Problemas?

- Los **volúmenes de datos**.
- Las **operaciones atómicas** que hay que llevar a cabo.

6. Ejemplos prácticos: SPMD - Reduction



6. Ejemplos prácticos: Reduction



6. Ejemplos prácticos: SPMD - Reduction

Imaginemos que hay distintos grupos (bloques) de unidades de ejecución (hilos), con la **memoria compartida a nivel de bloque** (como procesos MPI que lanzan hilos… o **como en una GPU**).

Bloque 0:	Hilo 0	Hilo 1	Hilo 2	Hilo 3
Bloque 1:	Hilo 0	Hilo 1	Hilo 2	Hilo 3
Bloque 2:	Hilo 0	Hilo 1	Hilo 2	Hilo 3
Bloque 3:	Hilo 0	Hilo 1	Hilo 2	Hilo 3

6. Ejemplos prácticos: SPMD - Reduction

Imaginemos que hay distintos grupos (bloques) de unidades de ejecución (hilos), con la **memoria compartida a nivel de bloque** (como procesos MPI que lanzan hilos… o **como en una GPU**).

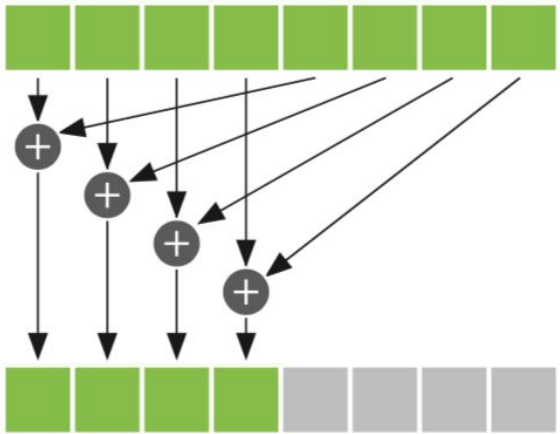
Tenemos que tener en cuenta:

- Dónde almacenar los **resultados parciales**:
  - En la primera **mitad del vector de datos**
  - En otra **estructura de datos**
  - Es importante **adecuar la estructura** a los conjuntos de hilos que tengamos.
- Controlar los hilo que ya no tienen datos a los que acceder.

6. Ejemplos prácticos: SPMD - Reduction

- Pensemos en una suma, donde cada bloque de hilos tiene un buffer con una posición para cada hilo:

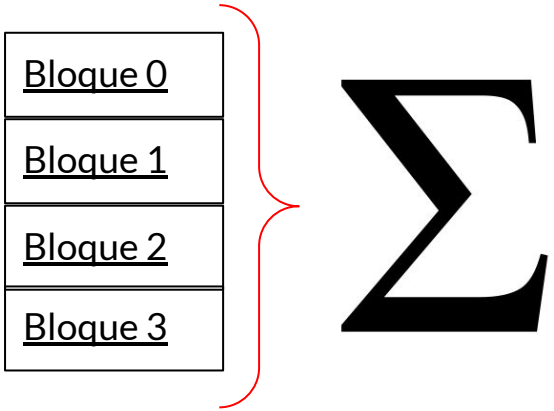
Bloque i:



```
calcularMiElemento();
barrier();
int focus = myThreadID;
int i = blockSize/2;
while (i != 0) {
        if (focus < i)
                buffer[focus] += buffer[i + focus];
        barrier();
        i /= 2;
}
```

6. Ejemplos prácticos: SPMD - Reduction

- Pensemos en una suma, donde cada bloque de hilos tiene un buffer con una posición para cada hilo:




```
barrierBloque();
int resultadoFinal = 0;
for (i in bloques) {
        resultadoFinal += bloques[i].buffer[0]
}
```

6. Ejemplos prácticos: SPMD - Reduction

- Si esta reducción o suma acumulada la hiciéramos **en secuencial**: El **tiempo** sería **proporcional a la longitud del vector**.
- Aprovechando los múltiples hilos… el tiempo será proporcional al logaritmo de la longitud del vector:
  - Primero, cada hilo combina dos registros en uno, necesitando la mitad de pasos que elementos.
  - Se repite con la **mitad** que queda, tardando entonces la **mitad de la mitad**…
- Tardaremos entonces $\log_2(hilosEnBloque)$... Por ejemplo, con **256 hilos** (y elementos), quedan **sumados tras 8 iteraciones**.

6. Ejemplos prácticos: SPMD - Reduction

- Así se plasma esta idea en CUDA:

```
__global__ void productoEscalar( float *a, float *b, float *c ){
        __shared__ float cache[threadsPerBlock];
       int tid = threadIdx.x + blockIdx.x * blockDim.x;
        int cacheIndex = threadIdx.x;
       float temp = 0;
        while (tid < N) {
                temp += a[tid] * b[tid];
                tid += blockDim.x * gridDim.x;
        }
        cache[cacheIndex] = temp;
        // Sincronizar hilos de este bloque
        __syncthreads();
        // threadsPerBlock debe ser potencia de 2
        int i = blockDim.x/2;
        while (i != 0) {
                if (cacheIndex < i)
                        cache[cacheIndex] += cache[cacheIndex + i];
                __syncthreads();
                i /= 2;
        }
        if (cacheIndex == 0)
                c[blockIdx.x] = cache[0];
}
```

Finalmente, en la CPU:

```
(...)
cudaMemcpy( partial_c, dev_partial_c, sizeof(float)*blocksPerGrid, cudaMemcpyDeviceToHost );

c = 0;
for (int i=0; i<blocksPerGrid; i++) {
        c += partial_c[i];
}
(...)
```

6. Ejemplos prácticos: SPMD-scan

- El algoritmo **Prefix Sum** (**Scan**) es usado en asignación de tareas en procesadores masivamente paralelos y en asignación de recursos para hilos.

- Se utiliza también para pasar de código secuencial a paralelo, eliminando la limitación de la escalabilidad por las secuencias de código paralelo.

```
for(j=1;j<n;j++)
out[j]=out[j-1]+f(j);
```

```
forall(j){
        temp[j]=f[j] // se hace en paralelo
}
scan(out, temp) // se puede hacer en paralelo
```

- Es uno de los patrones de computación paralela más usado y da pistas para resolver otros problemas similares.

6. Ejemplos prácticos: SPMD-scan

Formalmente, el problema se puede definir así, en su variante inclusiva:

- Sea un vector X de tamaño N: $[X_0, X_1, ..., X_{N-1}]$

Se desea **calcular la secuencia** que relaciona todos los X_i con una función f:

$X_0, f(X_0, X_1), f(X_0, X_1, X_2), ... f(X_0 ... X_{N-1})$

Por ejemplo, si f es la operación suma y X = [3, 1, 7, 0, 4, 1, 6]

La solución será: [3, 3+1, 3+1+7, 3+1+7+0, 3+1+7+0+4, 3+1+7+0+4+1, 3+1+7+0+4+1+6]

= [3, 4, 11, 11, 15, 16, 22]

(Nota: En definición exclusiva, X_i no se coge para $f(X_i)$)

6. Ejemplos prácticos: SPMD-scan

- Supongamos que vamos a repartir **100 trozos de tarta entre 10 personas**, y cada una pide la siguiente cantidad: [3,5,2,7,28,4,3,0,8,1].
- Uno repartiría tarta de forma constante y pararía cuando cada persona tuviera lo pedido, es decir, cortar 1 porción y darla, otra porción y darla y así sucesivamente… ¿Pero **cuánta dejar** por si viene alguien más?
- Una **opción (secuencial)** sería cortar primero 3, luego los 5… y así sucesivamente.
- Otra: calculamos **scan inclusivo**, obteniendo [3, 8, 10, 17, 45, 49, 52, 52, 60, 61] y **sabiendo así por donde cortar la tarta dejando 100-61 porciones y** cómo **marcar las** grandes **zonas** de tarta **rápidamente**.

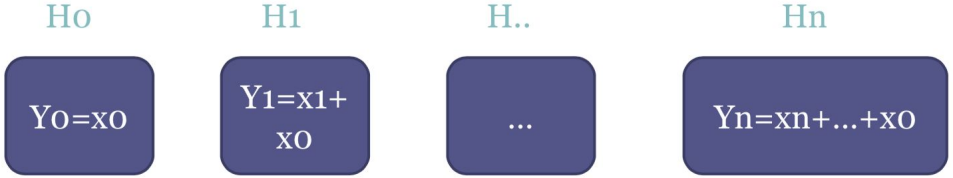
6. Ejemplos prácticos: SPMD-scan

- En secuencial: Hacer $Y_0 = X_0$; $Y_1 = X_0+X_1$; $Y_2 = X_0+X_1+X_2$...:

```
y[0]=x[0];
for(i=1;i<n;i++) y[i]=y[i-1]+x[i];
```

Eficiencia de orden N, O(N)

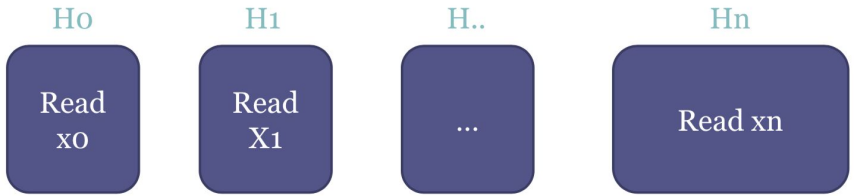
- Una primera (y descuidada) versión paralela crearía un hilo para calcular cada elemento de salida...:



- Pero claro, las **iteraciones finales** son **más costosas**... La latencia vendrá dada por el último hilo.

6. Ejemplos prácticos: SPMD-scan

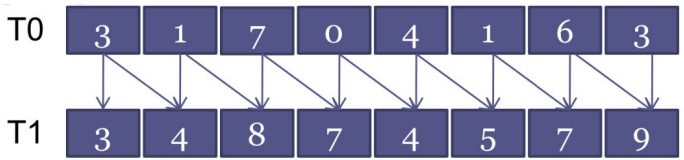
- Una segunda versión paralela, conceptualmente mucho más afinada, haría que el **1er hilo** calculara los elementos **0 y N-1**, que el **2º** se ocupara del **1** y el **N-2**, que el **3º** hiciera el **2** y **N-3**... Esto nos balancea perfectamente las operaciones... pero no tiene en cuenta los accesos a **memoria** y su escalabilidad llega **hasta N/2**.
- Otra alternativa (GPU): La estructura de datos es de lectura y escritura -> cada hilo debe acceder a una parte de la estructura. El primer paso será:



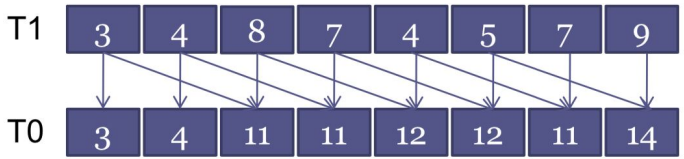
- Tras esto, todos los elementos estarán en la memoria compartida entre hilos.

6. Ejemplos prácticos: SPMD-scan

- Seguidamente: Iterar log(n) y sumar elementos que distan un determinado paso y que aumentará al doble en la siguiente iteración y alternando T0 y T1.



Distancia de suma = 1

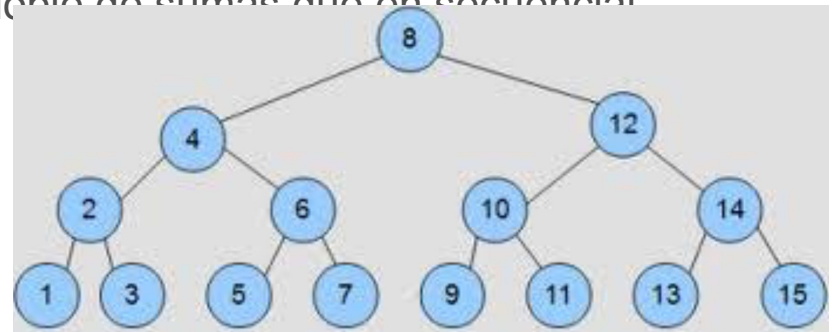


Distancia de suma = 2

(...)

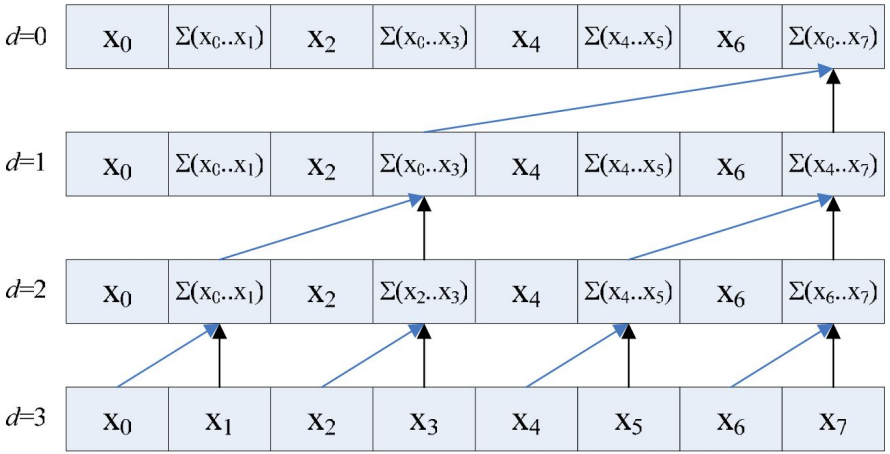
6. Ejemplos prácticos: SPMD-scan

- Según el tamaño de los bloques (hilos), tienen que **sincronizarse** por bloque en cada iteración. Además, se hacen **O(n*log(n)) sumas**. Por ejemplo para 256 elementos, la secuencial hace 255 sumas y la paralela (256*log(256)-(255)=1793 sumas.
- Solución: **Árboles** balanceados => Transformar el vector de números en un árbol B y aprovecharnos de las propiedades del árbol para implementar el SCAN. Así sólo se hace el doble de sumas que en secuencial

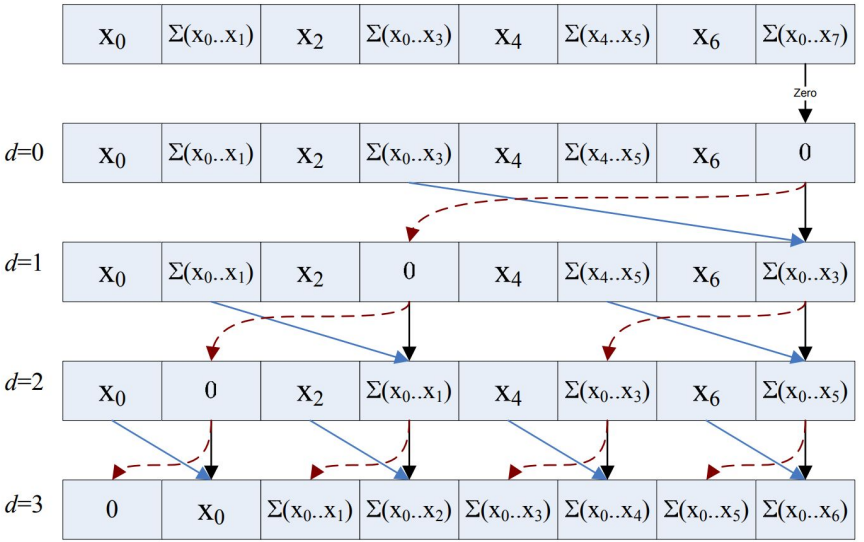


6. Ejemplos Prácticos: SPMD-scan

Up-Sweep



Down-Sweep



Este enfoque tiene conflictos por los accesos a memoria en los arrays compartidos por cada bloque de hebras, porque en algunos casos las hebras acceden a posiciones contiguas de memoria

6. Ejemplos Prácticos: SPMD-scan

Enfocamos el problema como dos fases:

Fase de reducción (desde hojas hasta raíz) calculando algunas sumas parciales que son necesarias para la fase 2 (Fase Up-Sweep)

Fase de barrido de subida, (Down-Sweep) donde se ponen algunos elementos del array a 0 inicializando con el último elemento del array que se pone a 0 desde el mismo programa

```
for d := 0 to log2n - 1 do
    for k from 0 to n – 1 by 2^(d+1) in parallel do
        x[k + 2^(d+1) - 1] := x[k + 2^d - 1] + x [k + 2^(d+1) - 1]
```

```
x[n - 1] := 0
for d := log2n down to 0 do
    for k from 0 to n – 1 by 2^(d+1) in parallel do
        t := x[k + 2^d - 1]
        x[k + 2^d - 1] := x [k + 2^(d+1) - 1]
        x[k + 2^(d+1) - 1] := t + x [k + 2^(d+1) - 1]
```

- Más detalles de este tipo de problema en: https://slideplayer.com/slide/14460027/

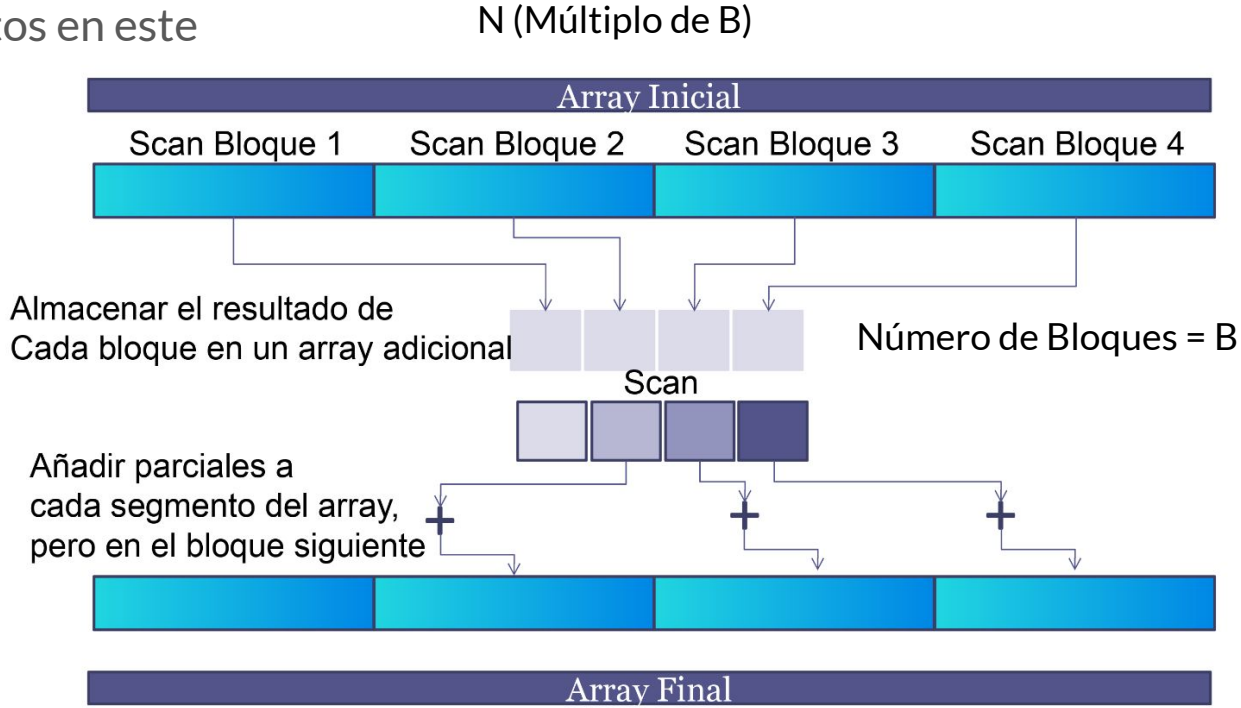
```
y[0]=x[0];
for(i=1;i<n;i++) y[i]=y[i-1]+x[i];
```

6. Ejemplos prácticos: scan

- Y si no caben los datos en este enfoque lineal...

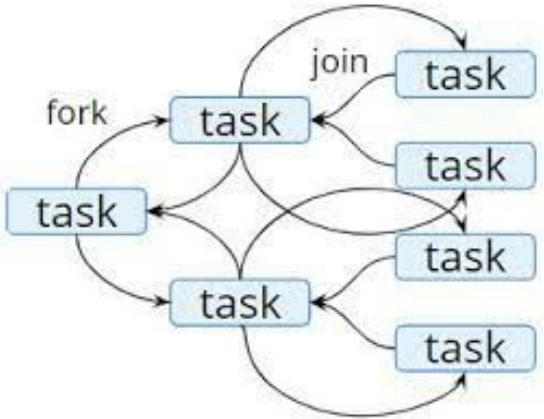
B=Número elementos calculados en un bloque
Bloques = N/B

Hilos = B/2

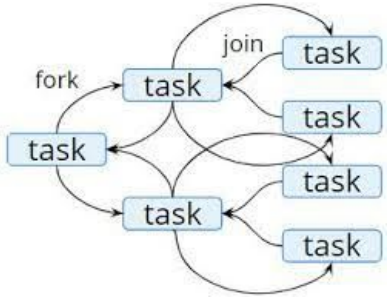


6. Ejemplos prácticos: fork-join

El esquema principal de la estructura Fork-Join es:



6. Ejemplos prácticos: fork-join



Veamos el framework de **Java Fork/Join** dentro del paquete **java.util.concurrent**.

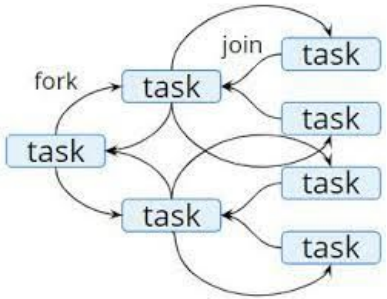
Las dos clases principales son:

1. **ForkJoinTask**: clase que nos permite crear bifurcaciones y uniones de tareas con dos subclases principales:
   - **RecursiveAction** : se utiliza para las tareas que **no devuelven resultados**
   - **RecursiveTask** : se utiliza para las tareas que sí **devuelven resultados**
2. **ForkJoinPool**: que mediante ForkJoinTask la ajustamos para crear una pila de tareas a gestionar.

La forma de gestionar los hilos es mediante una cola de tareas que cada hilo mantiene y que si se queda vacía se podrían recopilar tareas de otras colas para ejecutarlas (*work stealing*).

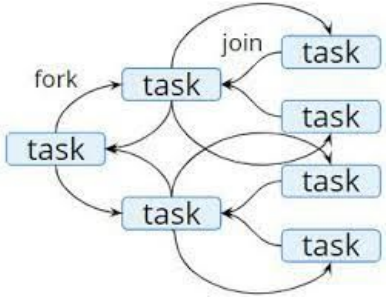
6. Ejemplos prácticos: fork-join

```
public static void main(String[] args) throws Exception {
        // Toma de tiempos e inicialización de la pila de tareas.
        //
        long start=System.currentTimeMillis();
        ForkJoinPool pool=new ForkJoinPool();
        Future<Integer>future=pool.submit(new ForkJoin(1,1000000));
        long end=System.currentTimeMillis();
        System.out.println("----------------------");
        System.out.println("Resultado Paralelo:"+future.get()+" Tiempo"+(end-start));

        // Comparación con la ejecución secuencial
        long start1=System.currentTimeMillis();
        int sum=getadd(1,1000000);
        long end1=System.currentTimeMillis();
        System.out.println("Resultado Secuencial:"+sum+" Tiempo "+(end1-start1));
}
```



```
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.Future;
import java.util.concurrent.RecursiveTask;

//Esquema del marco de ForkJoin (calcule la suma desde comenzar ... terminar, como 1 + 2 + 3 ... + 100)

public class ForkJoin  extends RecursiveTask<Integer>{
    private static final long serialVersionUID = 28980553733573142L;
    private int begin;  // Calcular el punto de partida de la suma acumulativa
    private int end;    // Calcular el punto final de la suma acumulativa
    public ForkJoin(int begin, int end) { super(); this.begin = begin; this.end = end; }

    @Override
    protected Integer compute() {
        int sum=0;
        if(end-begin<=1) { // La tarea se divide si supone un cierto tamaño
            for(int i=begin;i<=end;i++) sum+=i; // Calcula la pequeña suma
        }else { // Split
            ForkJoin d1= new ForkJoin(begin,(begin+end)/2);
            ForkJoin d2= new ForkJoin((begin+end)/2+1,end);
             // Tarea de ejecución dividida
             d1.fork();
             d2.fork();
             Integer a=d1.join();
             Integer b=d2.join();
             sum=a+b;
        }
        return sum;
    }
    // Resultado de ejecución de subproceso único
    public static int getadd(int begin,int end) {
        int sum=0;
        for(int i=begin;i!=end+1;i++) { sum+=i; }
        return sum;
    }
```

6. Ejemplos prácticos: fork-join



- Y otro ejemplo que ya vimos al hablar de recursividad: https://stackoverflow.com/questions/22583012/parallel-algorithm-to-produce-all-possible-sequences-of-a-set

6. Ejemplos prácticos: master/worker

Hay una entidad por encima de las demás y que puede controlar el flujo general del trabajo.

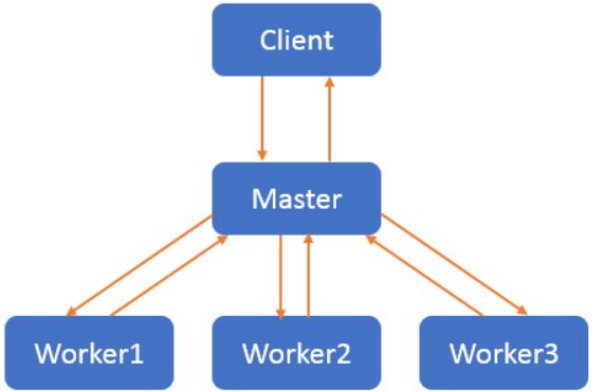
La principal diferencia con fork/Join es que la cantidad de trabajo no tiene por qué estar fijada desde el principio.

Existen distintas variantes de este esquema. Por ejemplo:

- **Cada Worker** tendrá **su** propia **cola de tareas** a las que el **Master le va introduciendo trabajo**.
- Los hilos **Worker** se crean **desde el principio** y permanecen **hasta el final** aunque no existan tareas pendientes en sus colas.
- Es el **Maestro** el que **marca la creación y la destrucción** de los Workers.

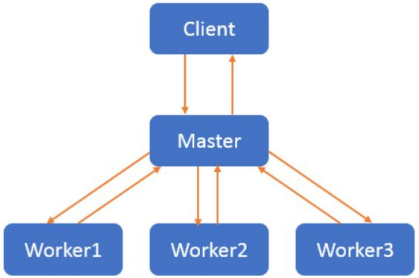
6. Ejemplos prácticos: master/worker

El esquema principal de la estructura Master/Worker es:



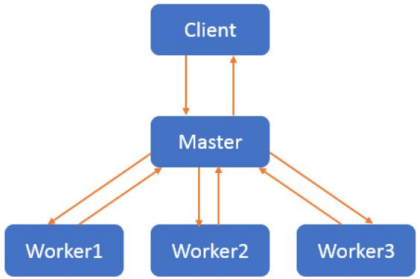
6. Ejemplos prácticos: master/worker

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

#define NUM_WORKERS 4

int works[NUM_WORKERS];
pthread_mutex_t mutex[NUM_WORKERS];
pthread_cond_t cond[NUM_WORKERS];
```



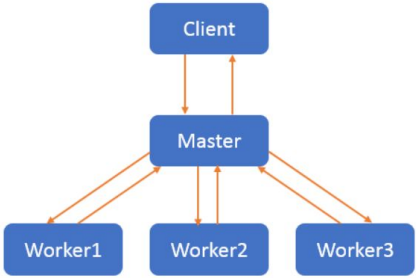
6. Ejemplos prácticos: master/worker

```
void* worker(void* param){
        long int myID = (long int) param;
        int myTask;
        while(1){
                pthread_mutex_lock(&(mutex[myID]));
                while(works[myID]<0){
                        pthread_cond_wait(&(cond[myID]), &(mutex[myID]));
                }
                myTask = works[myID];
                if(myTask!=0){
                        sleep(myTask);
                        printf("Hilo [%ld]: %d^2 = %d\n", myID, myTask, myTask*myTask);
                        works[myID] = -1;
                }else{
                        pthread_mutex_unlock(&(mutex[myID]));
                        break;
                }
                pthread_mutex_unlock(&(mutex[myID]));
        }
        return 0;
}
```



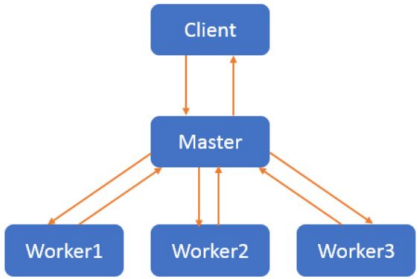
6. Ejemplos prácticos: master/worker

```
int main(void){
        pthread_t workers[NUM_WORKERS];
        for(long int i = 0; i<NUM_WORKERS; i++){
                pthread_mutex_init(&(mutex[i]), 0);
                pthread_cond_init(&(cond[i]), 0);
                works[i] = -1;
        }
        for(long int i = 0; i<NUM_WORKERS; i++){
                pthread_create(&(workers[i]), 0, worker, (void*) i);
        }
        (...)
```



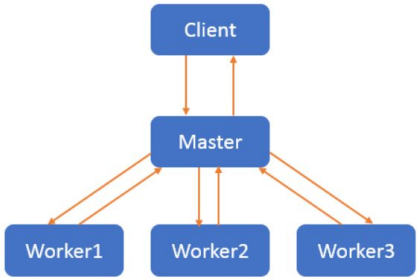
6. Ejemplos prácticos: master/worker

```
(...)
//BUCLE DE GESTION DE TRABAJO <MASTER>:
int newTask = 1;
int focus = 0;
while(newTask){
        printf("Introduce un elemento >0 para elevarlo al cuadrado:\n");
        scanf("%d", &newTask);
        if(newTask){
                while(1){
                        if(!pthread_mutex_trylock(&(mutex[focus]))){
                                works[focus] = newTask;
                                pthread_cond_signal(&(cond[focus]));
                                pthread_mutex_unlock(&(mutex[focus]));
                                focus = (focus + 1) % NUM_WORKERS;
                                break;
                        }else{
                                focus = (focus + 1) % NUM_WORKERS;
                        }
                }
        }else{//Indicamos salida:
                for(long int i = 0; i<NUM_WORKERS; i++){
                        pthread_mutex_lock(&(mutex[i]));
                        works[i] = 0;
                        pthread_cond_signal(&(cond[i]));
                        pthread_mutex_unlock(&(mutex[i]));
                }
        }
}
(...)
```

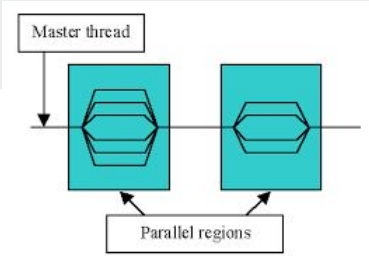


6. Ejemplos prácticos: master/worker

```
        (...)
        for(long int i = 0; i<NUM_WORKERS; i++){
                pthread_join(workers[i], 0);
        }
        for(long int i = 0; i<NUM_WORKERS; i++){
                pthread_mutex_destroy(&(mutex[i]));
                pthread_cond_destroy(&(cond[i]));
        }
        return 0;
}
```




```
MBP-de-Nicolas:Desktop nccruz$ ./mW
Introduce un elemento >0 para elevarlo al cuadrado:
10
Introduce un elemento >0 para elevarlo al cuadrado:
4
Introduce un elemento >0 para elevarlo al cuadrado:
5
Introduce un elemento >0 para elevarlo al cuadrado:
2
Introduce un elemento >0 para elevarlo al cuadrado:
Hilo [3]: 2^2 = 4
Hilo [1]: 4^2 = 16
Hilo [2]: 5^2 = 25
Hilo [0]: 10^2 = 100
```



6. Ejemplos prácticos: Loop Parallelism

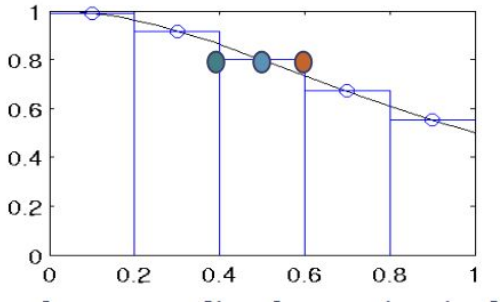
- El ámbito de la paralelización se centra en **repartir iteraciones de bucles entre** distintas **unidades de ejecución**.

- El estándar **OpenMP**, una API para programación en memoria compartida sobre C, C++ y Fortran, es uno de los ejemplos por excelencia orientados (aunque no limitados) a esta estrategia:
  - Define directivas de compilación que se traducen automáticamente en código paralelo para la plataforma.
  - Define también funciones con las que consultar ID's de hilo, total de hilos, variables de estado… y tipos de datos (por ejemplo, *omp_lock_t*).

- Podemos encontrar **implementaciones** de este modelo de paralelismo en otros entornos, como ***parfor* en Matlab** o **Parallel.For en C#**.

6. Ejemplos prácticos: Loop Paralellism

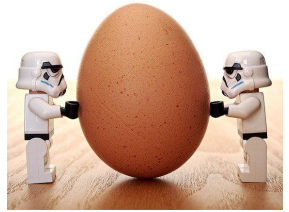
```
double piRectangles(int intervals){
     double width = 1.0/intervals;
     double sum = 0.0, x;
     #pragma omp parallel for reduction(+:sum) private(x)
     for(int i = 0; i<intervals; i++){
      x = (i + 0.5)*width;
      sum += 4.0/(1.0 + x*x);
      }
     return sum*width;
}
```



$$\int_0^1 \frac{1}{1+x^2}$$

```
nicolas@miriam-pc:~/Documentos/Docencia/UGR_ACAP/MiPr1$ gcc -o pi_omp omp_pi.c -fopenmp
nicolas@miriam-pc:~/Documentos/Docencia/UGR_ACAP/MiPr1$ time ./pi_omp 10000000
PI por integración de círculo [10000000 intervalos] =   3.141593

real    0m0,025s
user    0m0,182s
sys     0m0,000s
```

Trabajo para la próxima semana



Cada grupo debe explicar y presentar un mini-ejemplo funcional de las siguientes funciones colectivas de MPI:

- Grupo 1: MPI_Barrier
- Grupo 2: MPI_Reduce
- Grupo 3: MPI_Bcast
- Grupo 4: MPI_Gather
- Grupo 5: MPI_Allgather
- Grupo 6: MPI_Scatter
- Grupo 7: MPI_Scatterv

Gracias.