**PROGRAMACION PARALELA**

En el sentido más simple, la computación paralela es el uso simultáneo de múltiples recursos computacionales para resolver un problema computacional:

* Un problema se divide en partes discretas que se pueden resolver simultáneamente
* Cada parte se descompone en una serie de instrucciones
* Las instrucciones de cada parte se ejecutan simultáneamente en diferentes procesadores
* Se emplea un mecanismo global de control/coordinación.

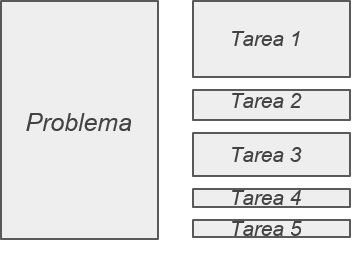
**Ventajas**

* Resuelve problemas que no se podrían realizar en una sola CPU
* Resuelve problemas que no se pueden resolver en un tiempo razonable
* Permite ejecutar problemas de un orden y complejidad mayor
* Permite ejecutar código de manera más rápida (aceleración)
* Permite ejecutar en general más problemas
* Obtención de resultados en menos tiempo
* Permite la ejecución de varias instrucciones en simultáneo
* Permite dividir una tarea en partes independientes

**Desventajas**

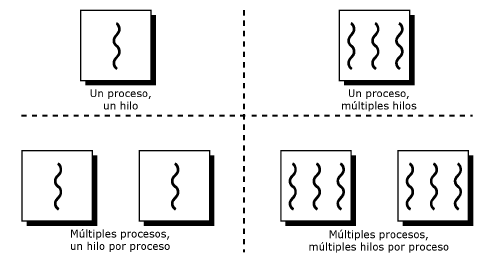
* Mayor consumo de energía
* Mayor dificultad a la hora de escribir programas
* Dificultad para lograr una buena sincronización y comunicación entre las tareas
* Retardos ocasionados por comunicación ente tareas
* Número de componentes usados es directamente proporcional a los fallos potenciales
* Condiciones de carrera
  + Múltiples procesos se encuentran en condición de carrera si el resultado de los mismos depende del orden de su llegada
  + Si los procesos que están en condición de carrera no son correctamente sincronizados, puede producirse una corrupción de datos

**Tareas**



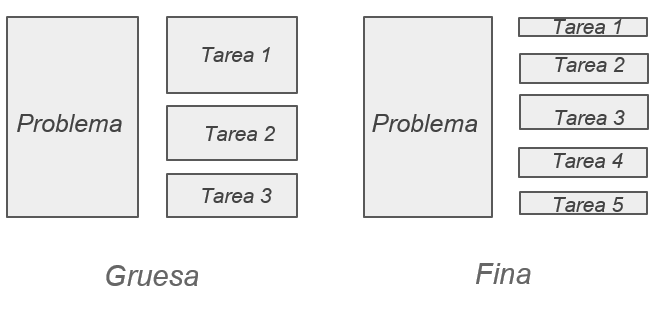
Un problema complejo se subdivide en una **cantidad discreta** de tareas que representan trabajo computacional. Una tarea esta compuesta de un **conjunto de instrucciones** que seran ejecutadas por un procesador.

**Hilo**



Un proceso pesado padre puede convertirse en varios **procesos livianos hijos**, ejecutados de manera concurrente. Cada uno de estos procesos livianos se conoce como hilo. Estos se comunican entre ellos a través de la memoria global.

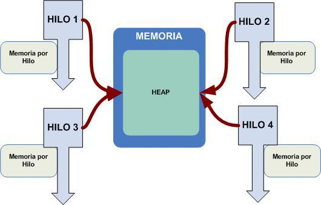
#### Granularidad:



Se refiere al tamaño de cada tarea y a la independiencia de las demás tareas, se dividen en dos categorías.

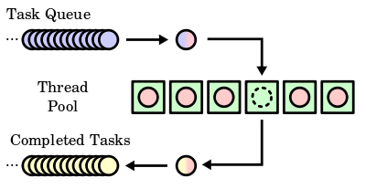
* **Gruesa:** Cantidad relativamente grande de trabajo, alta independencia entre tareas y poca necesidad de sincronización.
* **Fina:** Cantidades pequeñas de trabajo, poca independencia entre tareas, y una alta demanda de sincronización.

#### Sincronización:



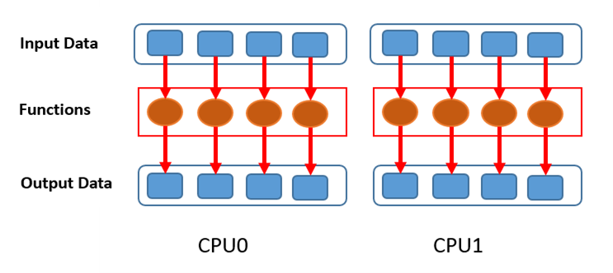
Los programas en paralelo necesitan la **coordinación de procesos e hilos, para que haya una ejecución correcta**. Los métodos de coordinación y sincronización en la programación paralela están fuertemente asociados a la manera en que los procesos o hilos intercambian información, y esto depende de cómo está organizada la memoria en el hardware.

#### Scheduling:



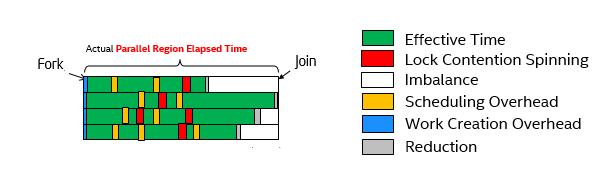
Scheduling es el proceso en el que **las tareas son asignadas a los procesos o hilos**, y se les da un orden de ejecución. Este puede ser especificado en el código, en tiempo de compilación o dinámicamente en tiempo de ejecución. El proceso de scheduling debe tener en cuenta la dependencia entre tareas, ya que, aunque muchas pueden ser independientes, otras pueden requerir los datos producidos por otras tareas.

#### Mapping:



Mapping en el proceso de **asignación de procesos e hilos a unidades de procesamiento**, procesadores o núcleos. Usualmente el mapping se hace por el sistema en tiempo de ejecución, aunque en ocasiones puede ser influenciado por el programador.

#### Overhead



Es la cantidad de **tiempo requerido para coordinar tareas paralelas**, en lugar de hacer un trabajo útil. Incluye factores como:

* Tiempo de inicio de la tarea
* Sincronización
* Comunicaciones de datos
* Sobrecarga de software impuesta por lenguajes paralelos, bibliotecas, sistema operativo, etc.
* Tiempo de terminación de la tarea

#### Sección crítica

Un proceso tiene un segmento de código llamado sección crítica cuando este puede modificar o leer información de memoria compartida con otros procesos. Dos procesos no pueden ejecutar su seccion critica al mismo tiempo.

## Concurrencia vs Paralelismo

### Concurrencia

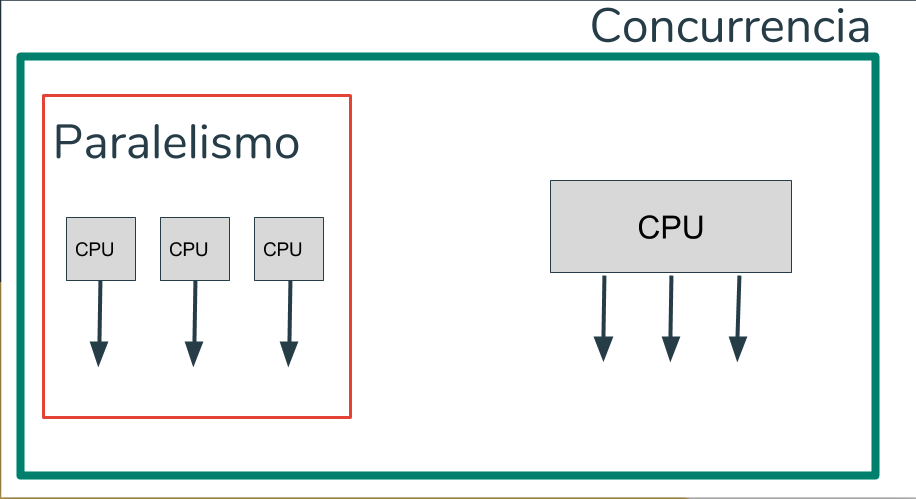
Capacidad de operar actividades al mismo tiempo. Es decir se pueden tener varios procesos corriendo cada uno en un procesador o puede haber varios proceso que corran solo en un procesador

### Paralelismo

Son muchas actividades teniendo lugar al mismo tiempo, “la cualidad o el estado de ser paralelo”. El hecho de ser paralelo implica que solo se pueden tener varios procesos corriendo cada uno en un procesador.

### Fork-Join

Fork-Join es un patrón que nos permite partir el flujo de un programa en múltiples flujos paralelos para reunirlos luego.



Ejemplo “hello world”:

