**Programacion Paralela**

**Introducción**

La computación paralela es el uso de múltiples recursos computacionales para resolver un problema. Se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente.

El paralelismo clásico, o puesto de otra manera, el clásico uso del paralelismo, es el de diseño de programas eficientes en el ámbito científico. La simulación de problemas científicos es un área de gran importancia, los cuales requieren de una gran capacidad de procesamiento y de espacio de memoria, debido a las complejas operaciones que se deben realizar.

Otro uso clásico es el de las gráficas generadas por computadora. La generación de fotogramas requiere de una gran cantidad de cálculos matemáticos. Esto supone una tarea muy compleja para un solo procesador, luego es necesario que haya algún tipo de paralelismo, para distribuir la tarea para que esta sea realizada eficiente y eficazmente.

## 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)

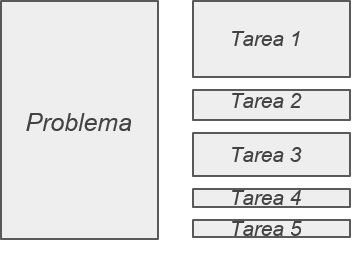
**Desventajas**

* 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

## Conceptos Clave

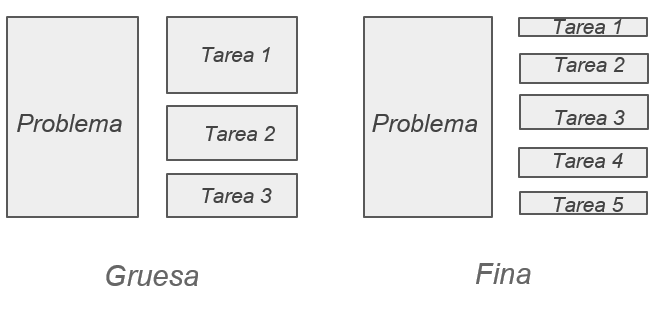
### Conceptos acerca de Tareas

#### Tarea:



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.

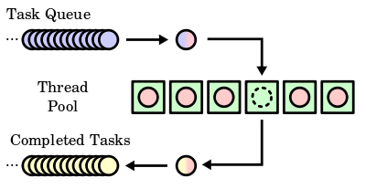
#### Granularidad:



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.

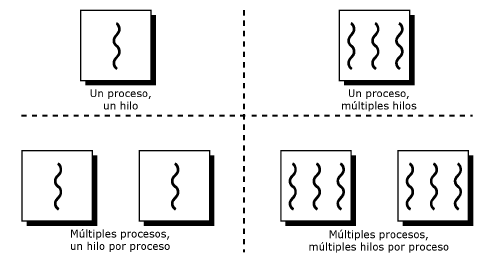
#### Scheduling:



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.

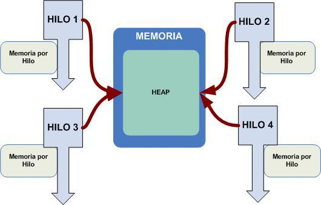
### Conceptos acerca de Hilos

#### Hilo:



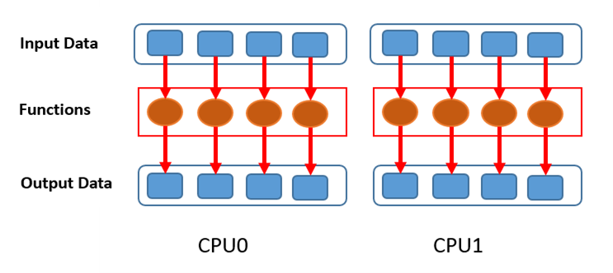
Un proceso pesado padre puede convertirse en varios **procesos livianos hijos**, ejecutados de manera concurrente. Cada uno de estos procesos livianos se conoce como hilo. Estos se comunican entre ellos a través de la memoria global.

#### Sincronizacion:



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.

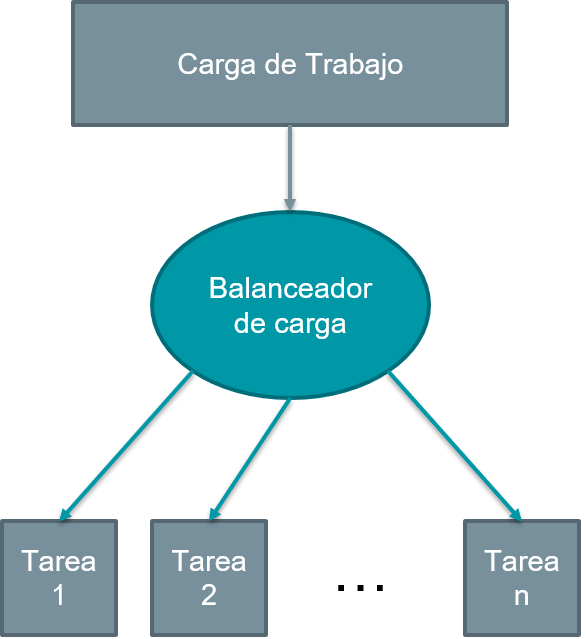
#### Mapping:



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.

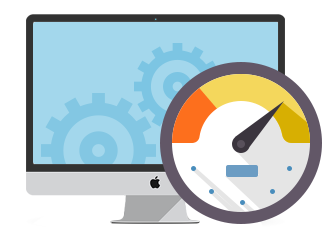
### Otros conceptos

#### Balanceo de carga



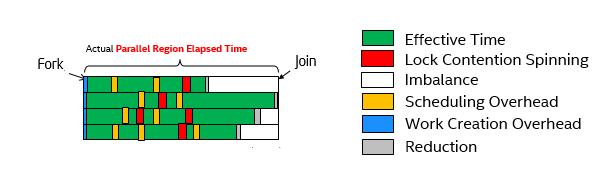
Se refiere a la práctica de distribuir **cantidades equitativas de trabajo** entre las tareas, de modo que todas las tareas se mantengan ocupadas todo el tiempo.

#### Speedup:



Es un proceso para **aumentar el rendimiento** entre dos sistemas procesando el mismo problema. Es la mejora en la velocidad de ejecución de una tarea ejecutada en **dos arquitecturas similares** con diferentes recursos.

#### 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.