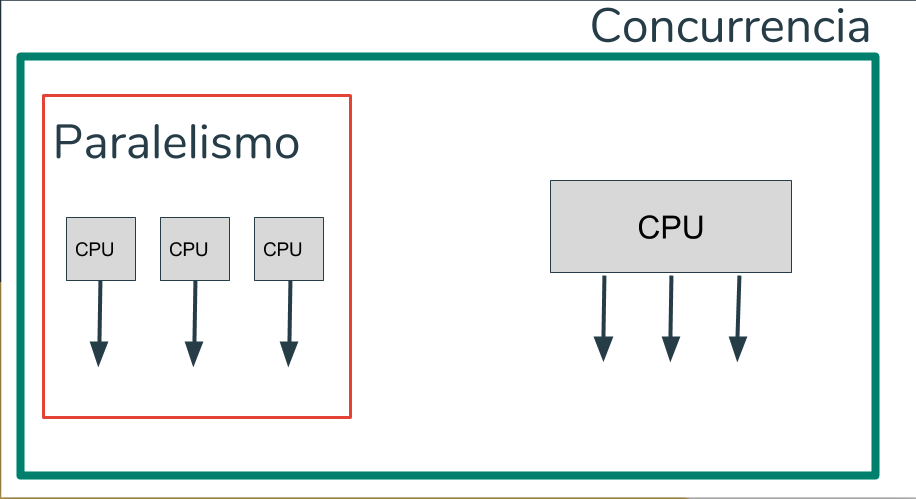
Programación paralela

# ¿Qué es la programación paralela?

La programación paralela es aquella que permite computar 2 o más tareas de forma simultánea permitido que el código generado no utilice demasiados recursos computacionales y optimizando en mayor medida el calculo de cantidades enormes de información.

Muchas veces el concepto de paralelismo se confunde con el de concurrencia, pero esta mal referenciado, mientras que el paralelismo indica que se puedan ejecutar 2 o mas tareas de forma simultánea, la concurrencia es la ejecución de 2 o mas tareas en periodos de tiempo solapados (superponer de forma parcial).



# ¿Qué es computación 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

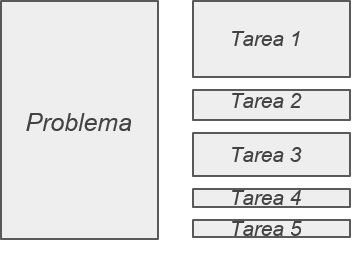
* Resuelve problemas que no se podrían realizar en una sola CPU
* Resuelve problemas que no se pueden resolver en un tiempo razonable
* Permite ejecutar problemas de un orden y complejidad mayor
* Permite ejecutar código de manera más rápida (aceleración)
* Permite ejecutar en general más problemas
* 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 estos 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

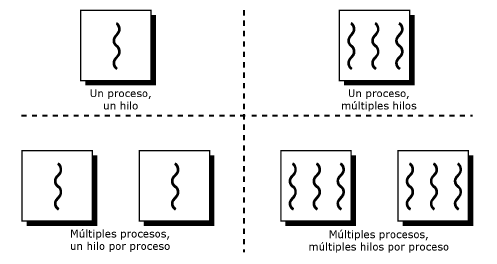
# Conceptos clave

## Tarea



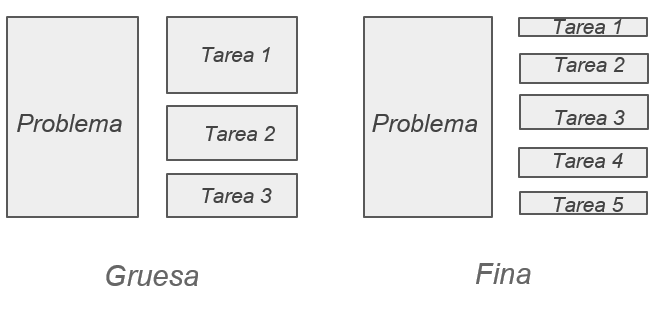
Un problema complejo se subdivide en una cantidad discreta de tareas que representan trabajo computacional. Una tarea está compuesta de un conjunto de instrucciones que serán ejecutadas por un procesador.

## 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.

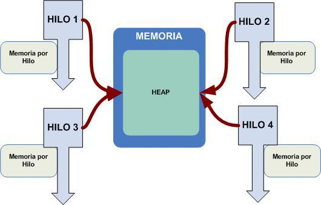
## Granularidad



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

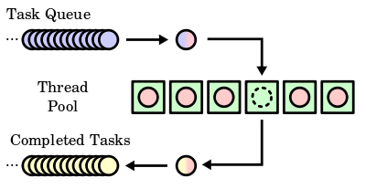
* Gruesa: Cantidad relativamente grande de trabajo, alta independencia entre tareas y poca necesidad de sincronización.
* Fina: Cantidades pequeñas de trabajo, poca independencia entre tareas, y una alta demanda de sincronización.

## Sincronización



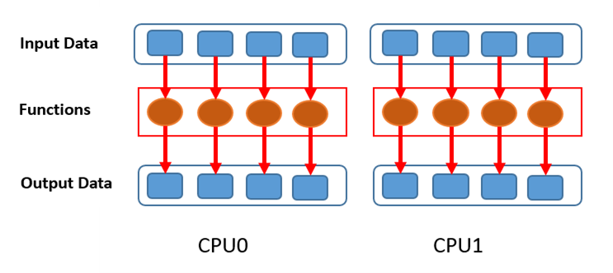
Los programas en paralelo necesitan la coordinación de procesos e hilos, para que haya una ejecución correcta. Los métodos de coordinación y sincronización en la programación paralela están fuertemente asociados a la manera en que los procesos o hilos intercambian información, y esto depende de cómo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

## Scheduling



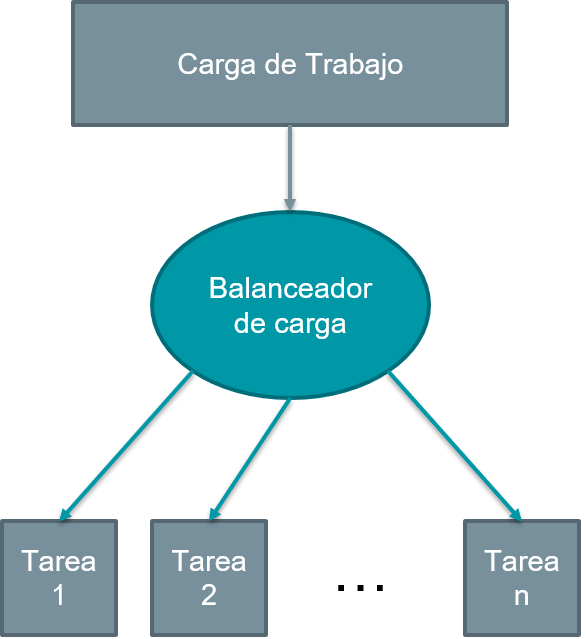
Scheduling es el proceso en el que las tareas son asignadas a los procesos o hilos, y se les da un orden de ejecución. Este puede ser especificado en el código, en tiempo de compilación o dinámicamente en tiempo de ejecución. El proceso de Scheduling debe tener en cuenta la dependencia entre tareas,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

## Mapping



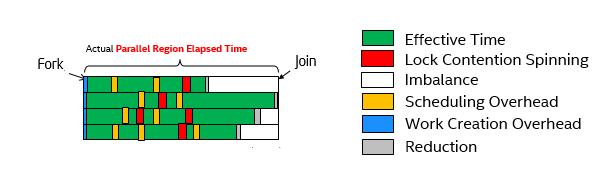
Mapping en el proceso de asignación de procesos e hilos a unidades de procesamiento, procesadores o núcleos. Usualmente el mapping se hace por el sistema en tiempo de ejecución, aunque en ocasiones puede ser influenciado por el programador.

## 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.

## 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

# Ejemplo en java de programación paralela

## Supermercado

Se inicializan los dos hilos, luego se imprime en pantalla cada vez que se pasen los productos en cada cajero.

Se puede observar cómo se realiza de forma paralela la compra en cada cajero.

1. **public** **class** MainThread {
3. **public** **static** **void** main(String[] args) {
5. Cliente cliente1 = **new** Cliente("Cliente 1", **new** **int**[] { 2, 2, 1, 5, 2, 3 });
6. Cliente cliente2 = **new** Cliente("Cliente 2", **new** **int**[] { 1, 3, 5, 1, 1 });
8. // Tiempo inicial de referencia
9. **long** initialTime = System.currentTimeMillis();
10. CajeraThread cajera1 = **new** CajeraThread("Cajera 1", cliente1, initialTime);
11. CajeraThread cajera2 = **new** CajeraThread("Cajera 2", cliente2, initialTime);
13. cajera1.start();
14. cajera2.start();
15. }
16. }

Resultados obtenidos al ejecutar el programa para los casos de ejemplo descritos en el main haciendo uso de dos hilos de trabajo.

La cajera Cajera 1 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIEMPO: 0seg

La cajera Cajera 2 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIEMPO: 0seg

Procesado el producto 1 del cliente Cliente 2->Tiempo: 1seg

Procesado el producto 1 del cliente Cliente 1->Tiempo: 2seg

Procesado el producto 2 del cliente Cliente 2->Tiempo: 4seg

Procesado el producto 2 del cliente Cliente 1->Tiempo: 4seg

Procesado el producto 3 del cliente Cliente 1->Tiempo: 5seg

Procesado el producto 3 del cliente Cliente 2->Tiempo: 9seg

Procesado el producto 4 del cliente Cliente 2->Tiempo: 10seg

Procesado el producto 4 del cliente Cliente 1->Tiempo: 10seg

Procesado el producto 5 del cliente Cliente 2->Tiempo: 11seg

La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 11seg

Procesado el producto 5 del cliente Cliente 1->Tiempo: 12seg

Procesado el producto 6 del cliente Cliente 1->Tiempo: 15seg

La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg