CUADRO COMPARATIVO (TEÓRIA)

# Computación Paralela

## ¿Qué es?

La computación paralela es una forma de procesamiento que permite **dividir un problema en varias partes** que se pueden ejecutar de forma simultánea. En vez de que una sola tarea se resuelva paso a paso, múltiples tareas trabajan al mismo tiempo en diferentes partes del problema, utilizando varios procesadores o núcleos. Esto permite **acelerar considerablemente el tiempo de procesamiento**, especialmente en tareas pesadas como simulaciones científicas, cálculos matemáticos complejos o procesamiento gráfico.

## Niveles

La computación paralela se puede aplicar en distintos niveles:

* **Nivel de bit o dato**, usando instrucciones vectorizadas que procesan varios datos a la vez.
* **Nivel de instrucción**, donde se aprovecha el pipeline del procesador.
* **Nivel de hilo y proceso**, usando múltiples hilos o procesos ejecutándose en paralelo.
* **Multiprocesamiento**, donde varios procesadores físicos colaboran.
* **Nivel distribuido**, donde muchas computadoras trabajan como una sola.

## ¿Para qué sirve?

Su propósito principal es **reducir el tiempo de ejecución** de programas exigentes. Se utiliza ampliamente en áreas que requieren un alto poder de cómputo, como:

* Inteligencia artificial y aprendizaje automático.
* Renderizado de gráficos 3D.
* Simulación científica y análisis de datos.
* Criptografía y cálculos numéricos intensivos.

## Historia/Origen:

La computación paralela tiene sus inicios en la década de 1940 con computadoras como la ENIAC. En los años 70 y 80, con el desarrollo de supercomputadoras, se empezó a aprovechar mejor la ejecución paralela. El crecimiento de las GPUs y los entornos de computación distribuida, como los clústeres, impulsaron su uso en las décadas siguientes. Hoy es común en ciencia, tecnología e incluso en tareas cotidianas como ver videos o usar redes sociales.

## Ventajas:

* **Velocidad**: Reduce drásticamente los tiempos de ejecución.
* **Escalabilidad**: Se puede aplicar desde pequeños sistemas hasta supercomputadoras.
* **Aprovecha hardware moderno**, como GPUs.
* **Ideal para tareas repetitivas o masivas.**

## Desventajas:

* **Complejidad**: Es difícil de programar y depurar.
* **Sincronización**: Las tareas deben coordinarse correctamente.
* **No todo se puede paralelizar**, hay límites teóricos.
* **Puede requerir hardware costoso o especializado.**

## Ejemplos:

* Renderizado de imágenes o animaciones en cine.
* Modelos de predicción meteorológica.
* Minería de criptomonedas.
* Juegos de alto rendimiento gráfico.
* Análisis de grandes volúmenes de datos en ciencia o negocios.

## Conclusión Computación Paralela:

* Es una herramienta esencial en la computación de alto rendimiento y permite resolver problemas que de otra manera tomarían horas o días.
* Se basa en dividir el trabajo y distribuirlo, lo que requiere no solo recursos, sino una lógica adecuada para evitar cuellos de botella.
* Aunque es más compleja de aplicar, es la base de tecnologías avanzadas como inteligencia artificial, big data o simulaciones científicas.

# Computación Concurrente

## ¿Qué es?

La computación concurrente es una técnica donde **varias tareas se ejecutan “al mismo tiempo” de manera lógica, aunque no necesariamente en paralelo físico**. El sistema interrumpe y reanuda tareas para que parezca que todas avanzan juntas, compartiendo el tiempo del procesador. Es ideal para sistemas interactivos, donde muchas cosas deben pasar al mismo tiempo (por ejemplo, una app que recibe datos, reproduce audio y responde al usuario).

## Niveles

* **A nivel de hilos**: varios hilos dentro de un mismo programa.
* **A nivel de procesos**: varios procesos independientes compartiendo el procesador.
* **A nivel del sistema operativo**, que administra la ejecución.

## ¿Para qué sirve?

Sirve para mejorar la **eficiencia y fluidez de programas** que hacen muchas cosas a la vez, como:

* Interfaces gráficas de usuario.
* Aplicaciones móviles o web.
* Sistemas embebidos que deben responder a muchos eventos (sensores, botones, etc).
* Servidores que atienden múltiples conexiones.

## Historia/Origen:

Comenzó a desarrollarse en los años 60 con la aparición de sistemas operativos multitarea como Unix. A medida que las computadoras se hicieron más interactivas, surgió la necesidad de gestionar múltiples tareas simultáneamente. Más adelante, con la evolución de lenguajes como Java, Go o C#, se incorporaron estructuras para manejar la concurrencia más fácilmente.

## Ventajas:

* **Multitarea fluida**, sin bloqueos evidentes.
* **Buen uso del tiempo del procesador** en tareas que esperan recursos (como disco o red).
* **Eficiencia en sistemas interactivos y servidores.**

## Desventajas:

* **Condiciones de carrera**, si no se controlan los accesos compartidos.
* **Bloqueos (deadlocks)** por mal diseño.
* **Más difícil de depurar** que la ejecución secuencial.
* **Puede parecer paralelismo, pero no siempre lo es.**

## Ejemplos:

* Navegadores web cargando múltiples pestañas.
* Apps que descargan archivos y permiten interacción al mismo tiempo.
* Juegos que manejan múltiples eventos de usuario.
* Chats y redes sociales.
* Servidores web como Apache o Nginx.

## Conclusión Computación Concurrente:

* La concurrencia no significa que las tareas se ejecuten físicamente al mismo tiempo, sino que se estructuran de forma que parecen hacerlo.
* Es fundamental para sistemas interactivos y software moderno, donde múltiples eventos deben ser gestionados sin demoras.
* Su correcta implementación mejora la experiencia del usuario, pero requiere un diseño cuidadoso para evitar errores complejos.

# Computación Multinúcleo

## ¿Qué es?

La computación multinúcleo consiste en usar **varios núcleos de procesamiento** dentro de un solo procesador físico. Cada núcleo puede ejecutar tareas de manera independiente, lo que permite que el sistema **realmente ejecute tareas en paralelo** si el software lo permite. Es la base física que hace posible aprovechar tanto la concurrencia como el paralelismo.

## Niveles

* **Núcleo:** unidad física de procesamiento.
* **Proceso/hilo:** cada núcleo puede manejar uno o más procesos/hilos.
* **Chip multicore:** donde residen todos los núcleos.

## ¿Para qué sirve?

Sirve para **mejorar el rendimiento del sistema** al poder ejecutar múltiples tareas simultáneamente. Esto se nota en:

* Mayor fluidez en el uso de muchas aplicaciones.
* Ejecución más rápida de software diseñado para múltiples núcleos.
* Mejor uso de recursos en juegos, edición de video, desarrollo de software, etc.

## Historia/Origen:

Surge a inicios de los años 2000 cuando los fabricantes como Intel y AMD llegaron al límite de aumentar la velocidad del reloj del procesador. En lugar de hacer CPUs más rápidas, optaron por incluir más núcleos en un solo chip. Esto revolucionó el diseño de software y hardware.

## Ventajas:

* **Mayor rendimiento en tareas simultáneas.**
* **Mejor multitarea**, incluso en tareas sencillas.
* Menor consumo energético comparado con CPUs más rápidas.
* Es compatible con la mayoría de los sistemas actuales.

## Desventajas:

* No todo el software aprovecha varios núcleos.
* Puede haber núcleos inactivos si no se usa bien.
* La programación paralela sigue siendo necesaria para explotarlo.

## Ejemplos:

* Procesadores modernos de PC o portátiles.
* Smartphones con 4, 6 u 8 núcleos.
* Consolas como PlayStation o Xbox.
* Aplicaciones modernas como navegadores, IDEs, editores de video.

## Conclusión Computación Multinúcleo:

* Representa el soporte físico que permite aplicar paralelismo real.
* Hoy es imposible imaginar una computadora moderna sin múltiples núcleos, desde smartphones hasta supercomputadoras.
* Aunque está ampliamente distribuido, muchas aplicaciones comunes aún no aprovechan todo su potencial, lo que representa un desafío para los desarrolladores.

# Conclusiones Comparativas y Relacionales:

* Aunque son conceptos distintos, **están profundamente conectados**: la computación **concurrente organiza**, la **paralela ejecuta simultáneamente** y la **multinúcleo lo hace físicamente posible**.
* En muchos sistemas modernos, **los tres trabajan juntos**: por ejemplo, una aplicación web (concurrente) puede distribuir cargas entre núcleos (multinúcleo) y usar librerías que paralelizan tareas pesadas (paralela).
* Comprender cómo se relacionan permite a los desarrolladores crear software más eficiente, optimizado y preparado para las exigencias actuales.
* En el futuro, a medida que el hardware avance y se demanden más tareas simultáneas, esta **combinación será aún más relevante** en inteligencia artificial, realidad virtual, procesamiento de datos y más.

# Opinión Personal:

* “La computación moderna ya no se trata de qué tan rápido es un solo procesador, sino de cuán bien aprovechamos los recursos disponibles. La concurrencia, el paralelismo y los procesadores multinúcleo son las herramientas clave para enfrentar los desafíos del presente y del futuro.”