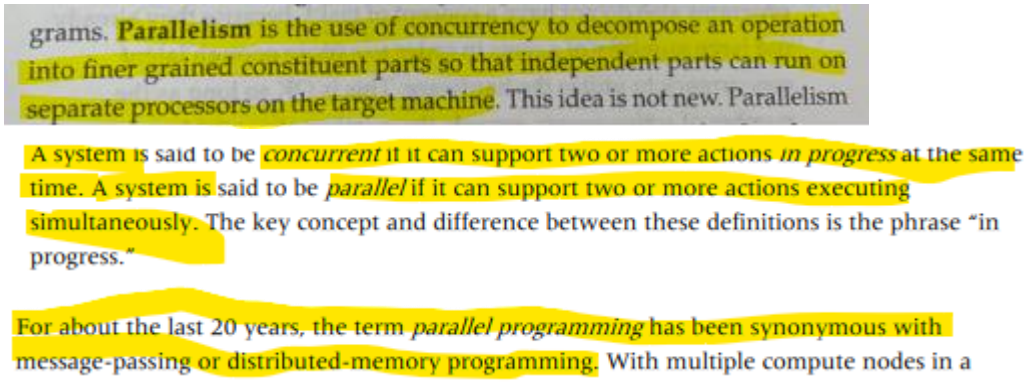
UNIVERSIDAD POLITÉCNICA DE PÉNJAMO

Datos	
Alumnos	Francisco Noé Arriaga Hernándes Diego Iván Morales Camarena Gabrielle Zuñiga Serrano Eder Javier Sosa Rivera
Matrícula	21020200004
Materia	Programación concurrente
Trabajo	Tabla comparativa
Fecha de entrega	11/10/2023
Maestro	Carlos Miguel Mozqueda Rodríguez

Tabla comparativa 1

	Secuencial	Paralela	Concurrente
Ventajas	Es más simple ya que se ejecuta una operación después de otra y su depuración de problemas se facilita ya que lleva un orden lógico	Se le dice paralela si soporta dos o más acciones ejecutándose simultáneamente.	Soporta dos o más acciones en progreso al mismo tiempo.
Desventajas			Debe utilizarse de forma responsable para no afectar negativamente a la robustez y fiabilidad del software (Página 11 Concurrent Programming on Windows)
Patologías			
Recomendaciones			
Características		Es el uso de concurrencia para descomponer una operación en partes constituyentes más finas, de modo que las partes independientes pueden ejecutarse en procesadores separados en la máquina de destino.	
Antecedentes		Durante aproximadamente 20 años, el término programación paralela ha sido sinónimo de paso de mensajes o de memoria distribuida.	



grams. **Parallelism** is the use of concurrency to decompose an operation into finer grained constituent parts so that independent parts can run on separate processors on the target machine. This idea is not new. Parallelism

A system is said to be *concurrent* if it can support two or more actions *in progress* at the same time. A system is said to be *parallel* if it can support two or more actions executing simultaneously. The key concept and difference between these definitions is the phrase "in progress."

For about the last 20 years, the term *parallel programming* has been synonymous with message-passing or distributed-memory programming. With multiple compute nodes in a

Tabla comparativa 2

	Secuencial	Paralela	Concurrente
Ventajas	Mayor facilidad para el entendimiento de los programas, se caracteriza por su orientación a optimizar la claridad, el tiempo y la calidad (KeepCoding Team, 2023).	Resolver problemas con gran cantidad de datos. Reducción del tiempo de solución. Dividir en subproblemas. Sincronización de Tareas. Evaluación de la eficiencia y costos. Maximizar el uso de los recursos de los procesadores paralelos (Tabarez Paz, Hernández Gress, & González Mendoza.	Velocidad de ejecución, solución de problemas de naturaleza concurrente y mejor aprovechamiento de la CPU
Desventajas	Inflexibilidad, no permite un retorno a una secuencia de repetición(Santos, 2013).	La programación paralela es diferente para cada arquitectura	Cuando varios procesos se ejecutan concurrentemente puede haber procesos que colaboren para un determinado fin, mientras que puede haber otros que compitan por los recursos del sistema. Incluso

			aquellos procesos que colaboran deberán competir a la hora de tomar tiempo del procesador.
Patologías	Sincronización en la ejecución de tareas, Actualizaciones concurrentes de variables compartidas (Drake, Diciembre).	Condiciones de carrera (Race Conditions): Ocurren cuando dos o más hilos o procesos acceden y modifican simultáneamente los mismos datos compartidos, lo que puede llevar a resultados inesperados y errores.	Para que un programa concurrente sea correcto debe satisfacer: propiedades de seguridad y propiedades de viveza.
Recomendaciones	Es correcto si cumple con su objetivo y realiza las tareas especificadas siguiendo una secuencia lineal de instrucciones.	Antes de paralelizar un programa, asegúrate de que exista un nivel adecuado de paralelismo en tu aplicación. No todas las tareas se pueden o deben paralelizar.	Un programa concurrente será correcto si, además de contemplar sus especificaciones funcionales, es capaz de evitar que se produzcan situaciones de interbloqueos y de inanición de procesos.
Características	Orden de ejecución de las instrucciones (orden total en la ejecución de las líneas de código).	Se refiere a la técnica de dividir una tarea en partes más pequeñas que se pueden ejecutar simultáneamente en múltiples núcleos de procesamiento o sistemas distribuidos.	Orden de ejecución de las instrucciones (hay un orden parcial) y el indeterminismo (pueden arrojar diferentes resultados cuando se ejecutan repetidamente sobre el mismo conjunto de datos).
Antecedentes	A finales de los años 1970 surgió una nueva forma de	La idea de utilizar múltiples procesadores o	En 1972 del lenguaje de alto nivel Concurrent

	programar que permitía desarrollar programas escritos de manera que se facilitaba su comprensión en fases de mejora posteriores (colaboradores de Wikipedia, 2023).	unidades de cálculo para resolver problemas de manera más eficiente se remonta a la década de 1950. Pioneros como John von Neumann y Alan Turing discutieron y exploraron conceptos relacionados con la paralelización.	Pascal [Brinch-Hansen, 1975], se encargó de romper este mito y abrir la puerta a otros lenguajes de alto nivel que incorporan concurrencia.

https://books.google.com.mx/books?id=LGMZodtyKXMC&pg=PA101&hl=es&source=gbs_selected_pages&cad=1#v=onepage&q&f=false

código escrito directamente en ensamblador. La aparición en 1972 del lenguaje de alto nivel Concurrent Pascal [Brinch-Hansen, 1975], desarrollado por Brinch Hansen, se encargó de romper este mito y abrir la puerta a otros lenguajes de alto nivel que incorporaban concurrencia.

Cuando varios procesos se ejecutan concurrentemente puede haber procesos que colaboren para un determinado fin, mientras que puede haber otros que compitan por los recursos del sistema. Incluso aquellos procesos que colaboran deberán competir a la hora de obtener tiempo de procesador. Por ejemplo, en la Figura 1.2, p1.1 y p1.2

Existen diversos motivos por los que la programación concurrente es útil. Destacaremos aquí dos de ellos: *velocidad de ejecución* y *solución de problemas de naturaleza concurrente*. Otros beneficios adicionales como el *mejor aprovechamiento de la CPU* saldrán a lo largo del capítulo.

1.6. Características de los sistemas concurrentes

La ejecución de sistemas concurrentes tiene algunas características que los diferencian claramente de los sistemas secuenciales. Destacamos dos: el orden de ejecución de las instrucciones y el indeterminismo.

1.6.1. Orden de ejecución de las instrucciones

En los programas secuenciales hay un **orden total** en la ejecución de las líneas de código. Ante un conjunto de datos de entrada se sabe siempre por dónde va a ir el programa (su flujo de ejecución). En la Figura 1.9, por muchas veces que se ejecute el

En los programas concurrentes, sin embargo, hay un **orden parcial**. Ante el mismo conjunto de datos de entrada no se puede saber cuál va a ser el flujo de ejecución. En cada ejecución del programa el flujo puede ir por distinto sitio. En la Figura 1.10, donde se supone que todas las instrucciones pueden ejecutarse concurrentemente, podemos ver cómo en dos ejecuciones distintas el orden en el que se ejecutan las instrucciones puede variar.

Este orden parcial lleva a que los programas concurrentes puedan tener un comportamiento indeterminista, es decir, puedan arrojar diferentes resultados cuando se ejecutan repetidamente sobre el mismo conjunto de datos de entrada. Esto suele llevar a muchas sorpresas cuando uno se inicia en la programación concurrente.

ma secuencial. Para que un programa concurrente sea correcto, además de cumplir las especificaciones funcionales que deba cumplir, debe satisfacer una serie de propiedades inherentes a la concurrencia. Podemos agrupar esas propiedades en:

- **Propiedades de seguridad:** son aquellas que aseguran que nada malo va a pasar durante la ejecución del programa.
- **Propiedades de viveza:** son aquellas que aseguran que algo bueno pasará eventualmente durante la ejecución del programa.

Un programa concurrente será correcto si, además de contemplar sus especificaciones funcionales donde irán implícitas condiciones de exclusión mutua y de sincronización, es capaz de evitar que se produzcan situaciones de interbloqueo y de inanición de procesos.

La estructura secuencial es la más simple de las estructuras de programación, en ella se ejecutan una serie de pasos siguiendo el orden de una secuencia, es decir, primero se ejecuta una operación, después otra, después la que sigue, y así sucesivamente. En general, los programas más simples tienen la estructura secuencial que se muestra en la Figura II-1.

https://www.rcs.cic.ipn.mx/2014_72/Aceleracion%20de%20la%20velocidad%20de%20procesamiento%20a%20traves%20de%20la%20paralelizacion%20de%20algoritmos.pdf
Ventajas, desventajas - Programación paralela

El paralelismo es una forma en la cual pueden realizarse varios cálculos simultáneamente. Está basado el en principio de dividir problemas grandes para obtener problemas pequeños los cuales posteriormente son solucionados en paralelo.

La programación paralela permite:

- Resolver problemas con gran cantidad de datos.
- Reducción del tiempo de solución.

Pereira [1], describe el paralelismo sobre conjuntos de datos métricos y un algoritmo de búsqueda por similitud basado en una estructura de indexación, utilizando el modelo de filtros. Dicho algoritmo mostró un desempeñño eficiente en la praáctica respecto del número de procesadores disponibles, en redes de tamano moderado.

Aracena [3], presenta una optimización del método de detección de puntos SIFT (Scale Invariant Feature Transform), mediante su paralelización empleando una GPU (Unidad de Procesamiento Gráfico). El objetivo es acelerar el tiempo de cómputo, para la detección de puntos característicos. Asimismo, el autor indica que se basa en dos premisas: el balance carga y la distribución de cálculo. Respecto a las pruebas se realizaron en con procesador Core 2 Duo 2.2Ghz,

Aplicaciones de la programación paralela En el diseño de los algoritmos paralelos es importante el enfoque para solucionar problemas grandes para campos de aplicación. Por esta razón se determina que las etapas de diseño de algoritmos paralelos son:

- Particionamiento: Dividir en subproblemas.
- Comunicación: Sincronización de Tareas.
- Agrupación: Avaluación de la eficiencia y costos.
- Asignación de Tareas. Maximizar el uso de los recursos de los procesadores paralelos.

Fig. 3. Etapas de diseño de algoritmos paralelos.

En la figura (3) se ilustra cada etapa del diseño para la paralelización del algoritmo. Por otro lado, existe una gran cantidad de aplicaciones de paralelización de algoritmos como clasificación de grandes bases de datos a través de redes neuronales artificiales (RNA), lo cual nos permite reducir en gran medida la velocidad de procesamiento en un algoritmo que por naturaleza es secuencial y que consume gran cantidad de tiempo de procesamiento, pero con grandes oportunidades de paralelización.

Otras aplicaciones son la predicción a través de redes neuronales artificiales; implementación de algoritmos para tratamiento de imágenes; circuitos de sistemas de control con microcontroladores y procesadores digitales de señal; asimismo en la simulación de imágenes tridimensionales para diseño en ingeniería y simulación de fenómenos físicos, económicos, financieros y sociales, pero sobre todo de videojuegos, en donde se obtiene una gran capacidad de resolución de imágenes y precisión de la dinámica del movimiento, aunado a la gran velocidad de procesamiento. Esto nos permite tener una poderosa herramienta para el desarrollo de la investigación científica para el clculo de arreglos numéricos con dimensiones muy grandes.

Finalmente hay una gran cantidad de aplicaciones sin embargo, la programación paralela es diferente para cada arquitectura. En este caso analizaremos un caso de estudio aplicado a clasificación de bases de datos a través de redes neuronales artificiales de tercera generacin como son las Máquinas de Soporte Vectorial (SVM).

Las fases de *compilación* y *ejecución* traducen y ejecutan el programa. En las fases de *verificación* y *depuración* el programador busca errores de las etapas anteriores y los elimina. Comprobará que mientras más tiempo se gaste en la fase de análisis y diseño, menos se gastará en la depuración del programa. Por último, se debe realizar la *documentación del programa*.

Bibliografía

KeepCoding Team. (16 de mayo de 2023). ¿Qué es la estructura secuencial en programación? Obtenido de keepcoding: https://keepcoding.io/blog/que-es-estructura-secuencial-programacion/#:~:text=Con%20el%20uso%20de%20la,de%20la%20l%C3%B3gica%20interna%20implementada.

colaboradores de Wikipedia. (20 de Enero de 2023). Programación estructurada. Obtenido de Wikipedia: https://es.wikipedia.org/w/index.php?title=Programaci%C3%B3n_estructurada&oldid=148735842

Drake, J. (8 de 10 de Diciembre). Programación concurrente. Obtenido de unican: https://www.ctr.unican.es/asignaturas/mc_procon/Doc/ProCon_I_03.pdf

Santos, C. G. (17 de Septiembre de 2013). Algoritmos Secuenciales. Obtenido de UNIVERSIDAD VERACRUZANA: https://www.uv.mx/personal/clgarcia/files/2012/10/ANTOLOGIA_pag_19_33.pdf

Tabarez Paz, I., Hernández Gress, N., & González Mendoza, M. (2014). Aceleraci´on de la velocidad de procesamiento. Research in Computing Science 72.

Laura Vázquez. (2 de Agosto de 2014). *Adobe Acrobat*. Obtenido de PDF: https://www.ecorfan.org/handbooks/Ciencias%20de%20la%20Ingenieria%20y%20Tecnologia%20T-VII/ARTICULO%207.pdf

Climent, J.-J. (24 de Noviembre de 2003). *RUA*. Obtenido de RUA: https://rua.ua.es/dspace/handle/10045/25282

Moya, R. (23 de Mayo de 2014). *Jarroba*. Obtenido de Jarroba: https://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/