

Sistemas Distribuidos y Paralelos

Ingeniería en Computación



Conceptos básicos

Universidad Nacional de La Plata



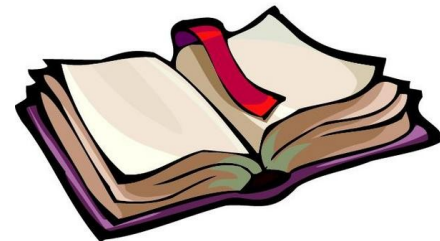
Facultad de Informática



Agenda

2

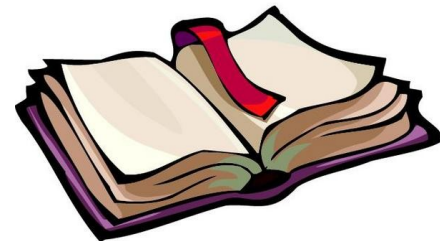
- I. Introducción a los Sistemas Paralelos
- II. Paralelismo implícito
- III. Paralelismo explícito
- IV. Características del desarrollo de aplicaciones paralelas
- V. HPC



Agenda

3

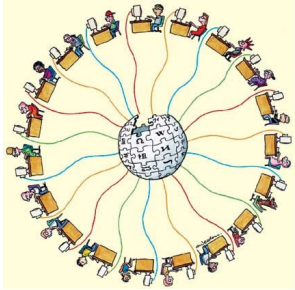
- I. Introducción a los Sistemas Paralelos
- II. Paralelismo implícito
- III. Paralelismo explícito
- IV. Características del desarrollo de aplicaciones paralelas
- V. HPC



Concurrencia y Paralelismo

Breve Repaso

Concurrencia



Simultaneidad en la ejecución de múltiples actividades o tareas.

Permite a distintos objetos actuar en un mismo período de tiempo.

Un programa concurrente especifica dos o más “programas secuenciales” que pueden ejecutarse concurrentemente en el tiempo como hilos o procesos.

Los **procesos** pueden ser **independientes, competir o cooperar**.

Se deben especificar los procesos concurrentes, su comunicación y sincronización.

Es un concepto de software.



Paralelismo



Concurrencia que requiere de hardware adicional.

Su **objetivo** es **mejorar el rendimiento** de una aplicación (Tiempo/Energía)

Un programa paralelo es un programa concurrente donde los hilos o **procesos cooperan** para mejorar el rendimiento de la aplicación.

Es un concepto de hardware.

Concurrencia

Paralelismo

Sistema Paralelo

5

Un **Sistema Paralelo** es la combinación de un algoritmo paralelo y la arquitectura paralela sobre la que se implementa.¹



- Un **Sistema Paralelo** está compuesto por:
 - **Software Paralelo (Algoritmo Paralelo):** varios **procesos o hilos** que **cooperan** para la resolución de un problema, con el objetivo de lograr **mayor rendimiento** que un algoritmo secuencial.
 - **Hardware Paralelo (Arquitectura Paralela):** arquitectura con **varias unidades de procesamiento** que permiten el procesamiento paralelo del software paralelo.

¹Libro: *An Introduction to Parallel Computing* - 2003

Aplicaciones de Sistemas Paralelos

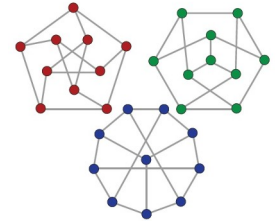
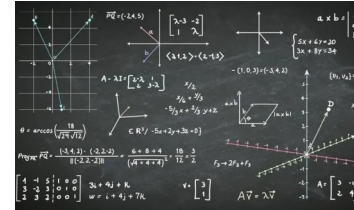
6

- Mayormente, los sistemas paralelos se utilizan en:
 - Aplicaciones que tratan con grandes volúmenes de datos
 - Algoritmos de complejidad exponencial o polinómica de grado elevado
 - Aplicaciones que requieren respuestas en tiempo real

- Estas aplicaciones abarcan distintas áreas:

- Simulación y Modelos de Predicción:

- Clima
- Incendios
- Inundaciones





- Álgebra Lineal
- Búsquedas, organización y análisis de datos
- Problemas combinatorios de optimización (TSP, Puzzle-15, Modelos econométricos)
- Bioinformática
- Procesamiento de imágenes y reconstrucción 3D
- High Performance Data Analytics (HPDA): HPC + IA



Analogía Vida Cotidiana - Sistemas Paralelos

7

- En la vida cotidiana hay tres formas de mejorar el rendimiento:

- Trabajar más duro 
- Trabajar más inteligentemente 
- Pedir ayuda y sumar voluntades



- Analogía con los Sistemas Paralelos:

- Trabajar más duro → Usar hardware más rápido
- Trabajar más inteligentemente → Usar algoritmos optimizados
- Pedir ayuda y sumar voluntades → Usar más hardware

Ley de Moore

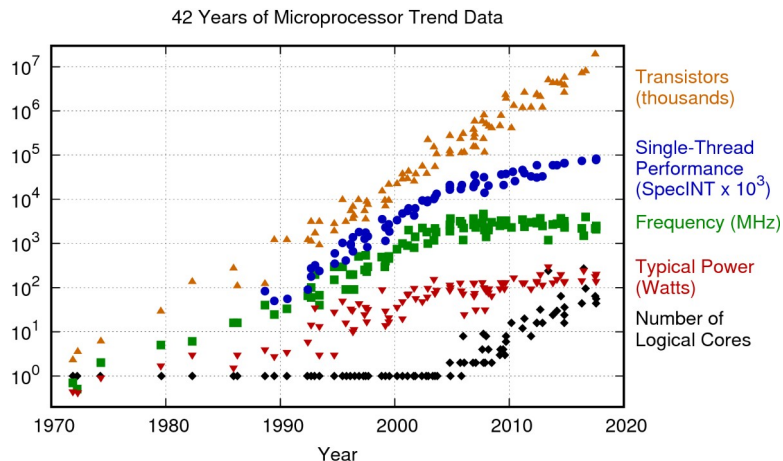
8

Sistemas Paralelos

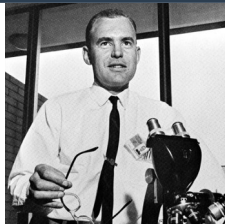
Usar hardware más rápido

Usar algoritmos optimizados

Usar más hardware



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp



Gordon E. Moore: co fundador de Intel (1968) junto a Robert Noyce.

- **Ley de Moore (19 de Abril de 1965)¹:** El número de transistores que pueden integrarse en un dispositivo con un coste determinado se dobla cada 18 meses.
- El desafío está en cómo trasladar el incremento en los transistores en mayor número de operaciones por segundo.
 - Incremento en la frecuencia del procesador.
- Se espera que deje de cumplirse por límites físicos.

¹<http://cva.stanford.edu/classes/cs99s/papers/moore-crammingmorecomponents.pdf>

Ley de Moore

Límite de los monoprocesadores

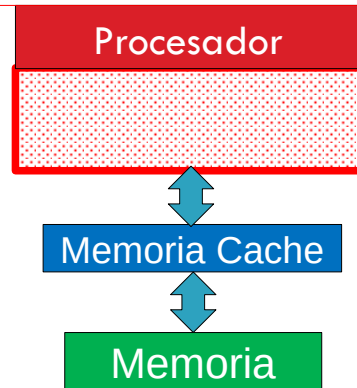
9

- **Siglo XXI:** Límite en monoprocesadores.

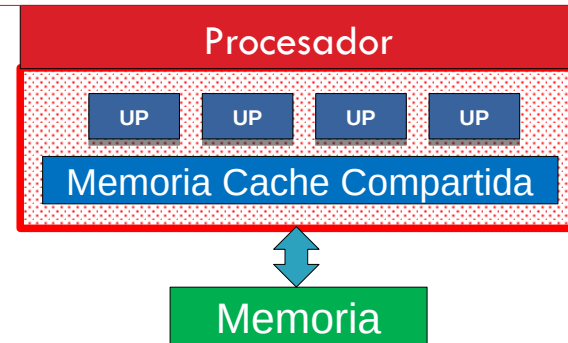
Mayor frecuencia → Mayor Consumo → Mayor Temperatura

- Surgimiento de arquitecturas multicore:

Siglo XX: un procesador es una única unidad de procesamiento.



Siglo XXI: un procesador está compuesto por varias unidades de procesamiento (cores) independientes.



- Ley de Gilder (Ley de la Banda Ancha): “La capacidad de las comunicaciones y que poseemos como individuos, pero también como empresas o instituciones, se triplica cada doce meses.”
- Ley de Metcalfe: “El valor de una red de telecomunicación aumenta proporcionalmente al cuadrado del número de usuarios del sistema.”
 - Un solo dispositivo es inútil, pero su valor se incrementa con el número total de dispositivos de la red, debido a que aumenta el número de dispositivos con los que se puede comunicar.
- Ley de Reed: “La utilidad de redes grandes, en particular redes sociales, escala exponencialmente con el tamaño de la red.”
 - El número posible de subgrupos de participantes de la red es $2^N - N - 1$ siendo N el número de participantes o posibles pares de conexiones.

The three Walls

11

- Si bien el hardware continuó siguiendo la ley de Moore, el crecimiento exponencial de la potencia informática "efectiva" choca contra tres "muros":

Memory Wall	Power Wall	ILP Wall
<ul style="list-style-type: none">• La velocidad de reloj del procesador aumentó más rápido que la velocidad de reloj de la memoria.• Las memorias caché ayudan a aliviar el problema, pero no lo resuelven.• La latencia en el acceso a la memoria suele ser el principal problema de rendimiento en aplicaciones modernas.	<ul style="list-style-type: none">• Procesadores más rápidos consumen más energía .• No lineal:<ul style="list-style-type: none">• Aumentar 73% en la potencia solo mejora el 13% en el rendimiento.• Reducir la velocidad de un procesador en un 13% proporciona la mitad del consumo de energía.• Los centros de computo están limitados por la potencia eléctrica total instalada y la potencia de refrigeración/extracción	<ul style="list-style-type: none">• Las dependencias de datos entre instrucciones y los branches en tiempo de ejecución limitan la cantidad de paralelismo a nivel de instrucción (ILP) alcanzable debido a la complejidad de los Pipelines (la predicción de saltos, ejecución especulativa, ejecución fuera de orden, etc.)

Sistemas Paralelos

Usar hardware más rápido

Usar algoritmos optimizados →

Usar más hardware

• Técnicas para mejorar el rendimiento:

- Reducir la cantidad de fallos de caché
- Minimizar el espacio de almacenamiento
- Minimizar las comunicaciones
- Minimizar la sincronización
- Optimización cómputo/comunicación

Arquitecturas paralelas


13

Sistemas Paralelos

Usar hardware más rápido

Usar algoritmos optimizados

Usar más hardware → • Sumar capacidad de cómputo: **arquitecturas paralelas**.

Arquitecturas de memoria compartida	Arquitecturas de memoria distribuida
Multiprocesadores Multicores Manycores (GPUs, Xeon PHI)	Clusters Grid
Arquitectura Híbrida	
	Combinación de arquitecturas de memoria compartida y memoria distribuida (Ejemplo: Cluster de Procesadores Multicore y GPUs)

Paralelismo implícito y explícito

14

- **Paralelismo implícito:** El control y el procesamiento es transparente al usuario.
 - Transparencia: ocultar al programador los detalles de funcionalidad.
- **Paralelismo explícito:** El programador es responsable de implementar la lógica del programa paralelo.

La mayor parte de este curso esta orientado a Paralelismo Explícito

Agenda

15

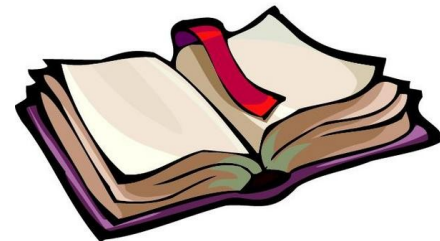
I. Introducción a los Sistemas Paralelos

II. Paralelismo implícito

III. Paralelismo explícito

IV. Características del desarrollo de aplicaciones paralelas

V. HPC



Paralelismo implícito

16

- Las formas más habituales de paralelismo implícito son:
 - **Segmentación (o pipelines):** se da en el paralelismo a nivel de instrucción (**ILP**).
 - Divide una función a realizar en varias etapas que pueden ejecutarse en forma independiente.
 - **División funcional:** utilizando varias unidades funcionales.
 - Multiplicar unidades aritmético-lógicas y de punto flotante.
 - Ejecución simultánea de varias instrucciones.
- De esto se encarga el hardware de **bajo nivel** y las optimizaciones del compilador.
- Si el paralelismo implícito se aplica en alto nivel, la transparencia facilita el proceso de producción y administración de software paralelo pero disminuye su eficiencia debido al overhead de gestión introducido.

Paralelismo implícito - ILP

17

- Las ejecución de una instrucción en un procesador se realiza en varias etapas. Suponiendo un procesador de 5 etapas:

IF	Instruction Fetch	Búsqueda de la instrucción en memoria
ID	Instruction Decode	Decodificación de la instrucción y lectura de registros
E	Execution	Ejecución de la instrucción o cálculo de direcciones
MEM	Memory	Acceso a un operando en memoria de datos
WB	Write back	Escritura de resultado en un registro
NA	No Action	Sin acción

La ejecución de estas dos instrucciones se ejecutaría...



$a = b + c$
 $e = a - d$

LW R1, &10F0
LW R2, &1008
Add R3, R2, R1
LW R4, &1000
Sub R5, R3, R4
SW R5, &2000

IF	ID	E	MEM	WB
IF	ID	E	MEM	WB
IF	ID	E	MEM	WB
IF	ID	E	MEM	WB
IF	ID	E	MEM	WB
IF	ID	E	MEM	WB

Pero... no es una forma eficiente de aprovechar el hardware. Suponiendo que se ejecuta una etapa por ciclo, cada instrucción requiere 5 ciclos.



Paralelismo implícito – ILP

Arquitectura Escalar

18

- Cada etapa usa hardware exclusivo. Se puede ejecutar cada etapa de manera que no interfiera en la ejecución del resto.



	Ciclo 0	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 4	Ciclo 5	Ciclo 6	Ciclo 7	Ciclo 8	Ciclo 9
LW R1, &10F0	IF	ID	E	MEM	WB					
LW R2, &1008		IF	ID	E	MEM	WB				
Add R3, R2, R1			IF	ID	E	MEM	WB			
LW R4, &1000				IF	ID	E	MEM	WB		
Sub R5, R3, R4					IF	ID	E	MEM	WB	
SW R5, &2000						IF	ID	E	MEM	WB

- El máximo paralelismo se alcanza cuando todas las etapas están activas, a partir de ese momento se ejecuta una instrucción en cada ciclo.
- **Arquitectura escalar:** arquitectura que ejecuta una instrucción por ciclo.

Paralelismo implícito – ILP

Arquitectura Superescalar – Operaciones multiciclo

- **Arquitectura superescalar:** procesador que puede ejecutar más de una instrucción por ciclo.

	Ciclo 0	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 4	Ciclo 5	Ciclo 6	Ciclo 7	Ciclo 8	Ciclo 9
LW R1, &10F0	IF	ID	E	MEM	WB					
LW R2, &1008	IF	ID	E	MEM	WB					
Add R3, R2, R1		IF	ID	E	MEM	WB				
LW R4, &1000			IF	ID	E	MEM	WB			
Sub R5, R3, R4				IF	ID	E	MEM	WB		
SW R5, &2000					IF	ID	E	MEM	WB	

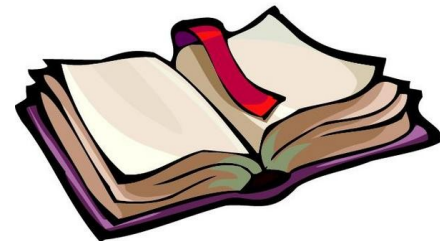
- **Operación multiciclo:** operación cuya etapa de ejecución demora más de un ciclo.
 - Operaciones de punto flotante.

	Ciclo 0	Ciclo 1	Ciclo 2	Ciclo 3	Ciclo 4	Ciclo 5	Ciclo 6	Ciclo 7	Ciclo 8	Ciclo 9
MUL R1, R2, R3	IF	ID	E	E	E	E	E	MEM	WB	
ADD R4, R1, R5		IF	ID	NA	NA	NA	NA	E	MEM	WB

Agenda

20

- I. Introducción a los Sistemas Paralelos
- II. Paralelismo implícito
- III. Paralelismo explícito
- IV. Características del desarrollo de aplicaciones paralelas
- V. HPC



Paralelismo explícito

21

- **Paralelismo explícito:** el programador es responsable de implementar la lógica del programa paralelo.
- Punto de partida:
 - Paralelizar un algoritmo secuencial.
 - Diseñar un algoritmo paralelo diferente al secuencial. La mejor solución puede diferir totalmente de la sugerida por los algoritmos secuenciales existentes:
 - Algoritmos que no tienen versión secuencial (Ej: **P**arallel **S**orting by **R**egular **S**ampling)
- Se deben usar herramientas que ayuden al desarrollo de un programa paralelo y que permitan abarcar los distintos aspectos.

Paralelismo explícito

Etapas de diseño

22

- Diseñar algoritmos paralelos no es fácil y conlleva mucho esfuerzo.
- No existe una única estrategia. Ian Foster¹ propuso una serie de pasos:
 - **Descomposición o Particionado:** Identificar el trabajo que puede hacerse en paralelo dividiéndolo en tareas
 - **Comunicación:** Determinar los patrones de comunicación entre las tareas identificadas en el paso previo
 - **Aglomeración:** Combinar las tareas y las comunicaciones identificadas en los pasos anteriores en tareas de mayor tamaño para adaptarlas a nuestro hardware paralelo
 - **Maapeo:** Asignar cada tarea a un procesador para maximizar la utilización del mismo y reducir la comunicación

¹*Libro: Designing and Building Parallel Programs - 1995*

Paralelismo explícito

Comportamiento de las aplicaciones

23

- Se debe considerar el comportamiento de las aplicaciones a paralelizar:
 - **Intensivas en cómputo** (CPU Bound): llevan al límite a la CPU.
 - La mayoría de las aplicaciones de cómputo científico.
 - **Intensivas en memoria** (Memory Bound): llevan al límite el sistema de memoria.
 - Ordenación
 - Búsquedas
 - **Intensivas en Entrada-Salida** (E/S Bound): llevan al límite el sistema de E/S.
 - Simulación
 - Big data
 - **Híbridos**: La aplicación presenta distintas fases intensivas.

Paralelismo explícito

Comportamiento de las aplicaciones

24

- Distintos eventos relacionados al cómputo requieren de un tiempo específico.

Las magnitudes suelen ser despreciables y no permiten observar las diferencias reales.

¿Qué ocurre si escalamos ese tiempo?



Evento	Tiempo Real	Tiempo Escalado
1 ciclo de CPU	0.3 ns	1 s
Acceso a cache de Nivel 1	0.9 ns	3 s
Acceso a cache de Nivel 2	2.8 ns	9 s
Acceso a cache de Nivel 3	12.9 ns	43 s
Acceso a RAM	120 ns	6 min
E/S a SSD	50-150 μ s	2-6 días
E/S a HDD	1-10 ms	1-12 meses
Internet: San Francisco a Nueva York	40 ms	4 años
Internet: San Francisco a UK	81 ms	8 años
Internet: San Francisco a Australia	183 ms	19 años
Reboot de SO de VM	4 s	423 años
Reboot de SO físico	5 m	32 milenios



Paralelismo explícito

Sistemas operativos

25

- Un sistema operativo que ejecute programas paralelos debería ser dedicado.
- No existen sistemas operativos desarrollados de "cero" para cómputo paralelo.
- Una característica deseable de un sistema operativo es que realice una adecuada asignación (proceso/unidad de procesamiento) para mantener la carga balanceada.
- Alternativas:
 - Distribuciones de sistemas operativos conocidas y adaptarlas.
 - Distribuciones de sistemas operativos conocidas ya adaptadas con software paralelo.
 - Distribuciones de sistemas operativos más Robustas/Dedicadas que permiten desplegar un sistema paralelo que provee aplicaciones para el usuario final.

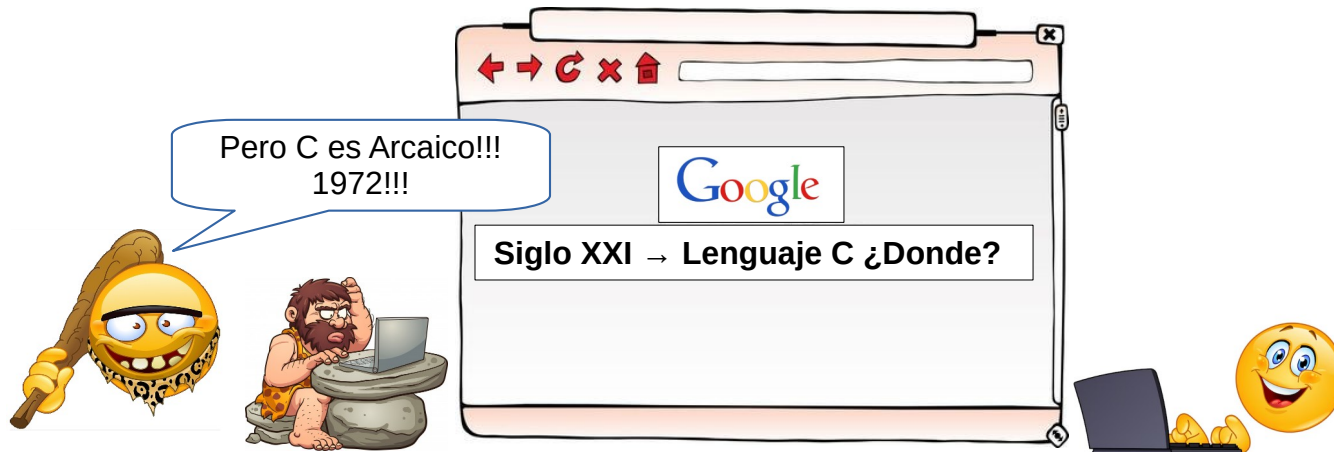
SO Conocidos	SO Adaptados	SO Dedicados
<ul style="list-style-type: none">• Linux: (Mayoritariamente)• FreeBSD/Solaris (Poco usadas)• Windows: NO!!! (no pensado para paralelismo)	<ul style="list-style-type: none">• CentOS: basado en RedHat• Scientific Linux: basado en RedHat. Mantenido por el CERN.	<ul style="list-style-type: none">• Rocks: basada en CentOS• FAI: basada en Debian• PelicanHPC: basado en Debian

Paralelismo explícito

Lenguajes de programación

26

- Es posible desarrollar aplicaciones paralelas con cualquier lenguaje que soporte la creación de varios procesos o hilos.
- Los lenguajes de muy alto nivel, en especial los interpretados, suelen introducir overhead que impacta en el rendimiento.
- La mayoría de las aplicaciones paralelas están escritas en C/C++ o Fortran.
 - Herramientas para otros lenguajes suelen ser "wrappers" que traducen a código C.




Kernel SO: Linux → Android Windows Etc.	Servicios: WebServers <i>Apache</i> MailServers AppServers <i>Tomcat</i> DBServers <i>MySQL</i> <i>Postgress</i> <i>MongoDB</i> Firewalls Buscadores Etc,etc y etc...
Interpretes: Java Python PHP Etc.	
Aplicaciones: Office Navegadores	

Paralelismo explícito

Modelos de programación paralela

27

- El desarrollo de aplicaciones paralelas se basa en dos modelos de programación:

Modelo de memoria compartida	Modelo de memoria distribuida
Procesos/Hilos se comunican mediante la memoria. Herramientas de desarrollo: Pthreads OpenMP Otros (Cuda, OpenCL, Cilk, Sycl...)	Procesos/Hilos se comunican mediante mensajes. Herramientas de desarrollo: PVM (en desuso) MPI Otros (RMI, Sockets, RPC...)
Modelo híbrido	
 Combinación de herramientas de desarrollo de ambos modelos (Ejemplo: MPI + OpenMP + CUDA)	

Paralelismo explícito

Modelos de programación paralela – Modelos de arquitectura paralela

28

- Los modelos de Programación Paralela (**Software**) y los modelos de Arquitectura Paralela (**Hardware**) son complementarios. Sin embargo, el modelo de programación paralela elegido depende del modelo de arquitectura paralela disponible.



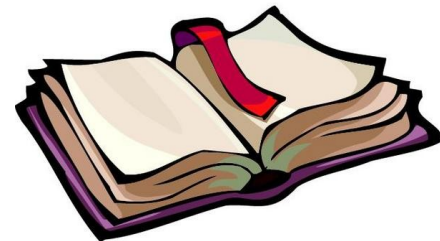
		Software		
		Memoria Compartida (Ej: OpenMP)	Memoria Distribuida (Ej: MPI)	Híbrido (EJ: MPI + OpenMP)
Hardware	Memoria Compartida (Ej: multicore)	Trivial (Ej: OpenMP sobre multicore)	Posible (Ej: MPI sobre multicore)	Posible (extraño) (Ej: MPI + OpenMP sobre multicore)
	Memoria Distribuida (Ej: cluster moncore ¹)	(NO ADECUADO) Single System Image(SSI) Overhead	Trivial (Ej: MPI sobre cluster moncore)	Posible (muy extraño) (Ej: MPI + OpenMP sobre cluster moncore)
	Híbrido (Ej: cluster multicore)	(NO ADECUADO) Single System Image(SSI) Overhead	Posible (Ej: MPI sobre cluster multicore)	Trivial (Ej: MPI + OpenMP sobre cluster multicore)

¹Actualmente, los clusters moncore cayeron en desuso pero se mencionan para ejemplificar.

Agenda

29

- I. Introducción a los Sistemas Paralelos
- II. Paralelismo implícito
- III. Paralelismo explícito
- IV. Características del desarrollo de aplicaciones paralelas
- V. HPC



Desarrollo de aplicaciones paralelas

30

- Características del desarrollo de aplicaciones paralelas:
 - No determinismo
 - Sincronización
 - Corrección y Depuración
 - Fallas
 - Comunicaciones
 - Balance de carga
 - Heterogeneidad
 - Rendimiento y Escalabilidad
 - Portabilidad
 - Seguridad
 - Asincronismo

Desarrollo de aplicaciones paralelas

31

- Características del desarrollo de aplicaciones paralelas:

- **No determinismo**
- Sincronización
- Corrección y Depuración
- Fallas
- Comunicaciones
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- Asincronismo

Un programa paralelo es un programa concurrente y como tal, su ejecución puede arrojar más de un resultado (no-determinismo). Pero no todos estos resultados podrían ser válidos.

Desarrollo de aplicaciones paralelas

32

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- **Sincronización**
- Corrección y Depuración
- Fallas
- Comunicaciones
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- Asincronismo

Como consecuencia del no determinismo pueden ocurrir resultados no válidos.

El buen uso de la sincronización (por exclusión mutua y/o por condición) ayuda a descartar los resultados no válidos y quedarnos con aquellos resultados que nos interesan.

El uso adecuado de la sincronización permite descartar deadlocks, livelocks, idling, race conditions y otros efectos no deseados.

Desarrollo de aplicaciones paralelas

33

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- **Corrección y Depuración**
- Fallas
- Comunicaciones
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- Asincronismo

La corrección y depuración de un programa paralelo es muy compleja.

Esto debido a la cantidad de "interleavings" posibles de un programa paralelo y el no determinismo.

Programar un algoritmo paralelo requiere un desarrollo cuidadoso y de una depuración del programa paso a paso.

Desarrollo de aplicaciones paralelas

34

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- **Fallas**
- Comunicaciones
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- Asincronismo

A mayor cantidad de hardware utilizado mayor posibilidad de falla.

Se incrementa la complejidad de detectar el punto de falla.

Programas paralelos de gran duración no deberían volver a ejecutarse de cero.

Hay que tener en cuenta aspectos como la tolerancia y recuperación de fallas (checkpoints).

Desarrollo de aplicaciones paralelas

35

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- Fallas
- **Comunicaciones**
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- Asincronismo

Al usar de redes de datos, los costos de comunicación son generalmente altos con respecto al tiempo de cómputo.

Tienen mayor impacto si las redes no son locales.

Por esta razón, es necesario minimizar el costo de las comunicaciones.

En ocasiones es posible ocultar la latencia intercalando comunicaciones con cómputo.

Desarrollo de aplicaciones paralelas

36

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- Fallas
- Comunicaciones
- **Balance de carga**
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- Asincronismo

Lograr distribuir equilibradamente el trabajo entre las unidades de procesamiento del sistema paralelo (mapeo), de modo de lograr que los tiempos de fin de trabajo sean similares.

Evitar que haya procesadores ociosos, o bien que haya una asignación de tareas que haga que un procesador tenga más trabajo que otros.(granularidad óptima)

Existen programas cuyo dinamismo provoca desbalances y necesitan balancear la carga en tiempo de ejecución.

Desarrollo de aplicaciones paralelas

37

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- Fallas
- Comunicaciones
- Balance de carga
- **Heterogeneidad**
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- Asincronismo

Puede existir heterogeneidad en varios niveles:

Hardware

Sistemas Operativos

Redes

Herramientas

Esto genera distintos problemas en relación a:

Compilación

Balance de carga

Desarrollo de aplicaciones paralelas

38

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- Fallas
- Comunicaciones
- Balance de carga
- Heterogeneidad
- **Rendimiento y Escalabilidad**
- Portabilidad
- Seguridad
- Asincronismo

El objetivo del algoritmo paralelo es obtener el máximo rendimiento posible respecto al algoritmo secuencial. Se pueden emplear métricas para cuantificar ese rendimiento (Speedup, Eficiencia)

Es deseable que la solución paralela sea escalable, es decir que permita incrementar el número de unidades de procesamiento y la carga de trabajo del problema sin perder rendimiento.

Desarrollo de aplicaciones paralelas

39

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- Fallas
- Comunicaciones
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- **Portabilidad**
- Seguridad
- Asincronismo

Es fácil lograr **portabilidad del código**, es decir establecer estándares que permitan ejecutar nuestro código en distintas arquitecturas.

Sin embargo, es difícil lograr **portabilidad de rendimiento**, es decir lograr que el programa alcance en diferentes arquitecturas un rendimiento similar.

La diferencia entre arquitecturas puede ser tan significativa que es imposible ejecutar el mismo código y el programador se ve obligado a rediseñarlo.

Ejemplo: portar un algoritmo paralelo desarrollado para multicores hacia una GPU.

Desarrollo de aplicaciones paralelas

40

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- Fallas
- Comunicaciones
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- **Seguridad**
- Asincronismo

Si el hardware paralelo tiene acceso desde y hacia el exterior es necesario incluir políticas de seguridad necesarias para evitar ataques.

Tener en cuenta que podría usarse toda la potencia de cómputo de un sistema paralelo para atacar otros sitios o realizar otras actividades “ilícitas”.

Desarrollo de aplicaciones paralelas

41

- Características del desarrollo de aplicaciones paralelas:

- No determinismo
- Sincronización
- Corrección y Depuración
- Fallas
- Comunicaciones
- Balance de carga
- Heterogeneidad
- Rendimiento y Escalabilidad
- Portabilidad
- Seguridad
- **Asincronismo**

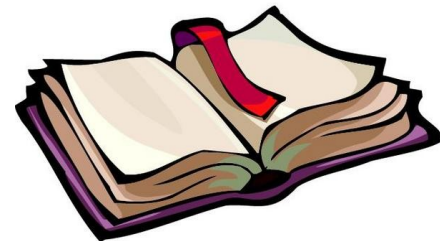
El concepto de tiempo único-universal cambia.
Aparecen distintos relojes en juego.

La ocurrencia de un evento puede ser conocido por dos procesos en instantes distintos.

Agenda

42

- I. Introducción a los Sistemas Paralelos
- II. Paralelismo implícito
- III. Paralelismo explícito
- IV. Características del desarrollo de aplicaciones paralelas
- V. HPC



Concepto de High Performance Computing (HPC)

43

- Los sistemas paralelos son parte de un concepto mas amplio:

High Performance Computing

(Cómputo de altas prestaciones o computación de alto rendimiento)

HPC: práctica de sumar potencia computacional con el fin de alcanzar mayor rendimiento en la ejecución de programas que resuelven problemas de gran tamaño de la ciencia, ingeniería o negocios.

- **HPC** abarca el estudio de distintas áreas:

