

# Tema 1. Introducción a la Programación Distribuida y Paralela



***Programación Distribuida y Paralela***  
***Depto. de Lenguajes y Sistemas Informáticos***  
***Universidad de Granada***



# **Contenidos**

**1. Introducción**

**2. Modelos de Sistemas Paralelos**

**3. Modelos de Programación  
Paralela**

**4. Análisis del rendimiento**

# 1. Introducción

## Nociones básicas de programación paralela

- **Computador paralelo:** Capaz de ejecutar varias instrucciones simultáneamente.
- **Computación Paralela:** Uso de varios procesadores trabajando juntos para resolver una tarea común:
  - Cada procesador trabaja en una porción del problema
  - Los procesos pueden intercambiar datos, a través de la direcciones de memoria compartidas o mediante una red de interconexión
- **Programación Paralela:** Considera aspectos conceptuales y las particularidades físicas de la computación paralela.
  - **Objetivo:** Mejorar las prestaciones mediante un buen aprovechamiento de la ejecución simultánea.

# 1. Introducción

## Ejemplo: Ordenar un conjunto de libros

- Libros organizados en **estantes**.
- Estantes agrupados en **estanterías**.
- **Una persona** → Velocidad limitada.
- **Varias personas** → **Enfoques**
  - a) Repartir libros entre trabajadores + ordenac. simultánea.  
Trabajadores deben desplazarse.
  - b) Repartir libros y estanterías  
Si trabajador encuentra un libro suyo lo almacena.  
Si no, lo pasa al responsable.
- **Aspectos de la solución paralela:**
  - **Descomposición**: Tarea es dividida en subtareas.
  - **Asignación**: tareas son asignadas a trabajadores.
  - **Comunicación**: Trabajadores deben cooperar.
  - **Diferentes soluciones paralelas** de un problema



# 1. Introducción

## Necesidad de la computación paralela

### Limitaciones físicas de la computación secuencial

- › Límite de la velocidad de la luz: Estancamiento en los incrementos de la frecuencia de reloj.
- › Límite de Integración: Cerca del límite máximo.
  - › Más frecuencia → Más consumo + Temperaturas muy elevadas

### Problemas con complejidad elevada

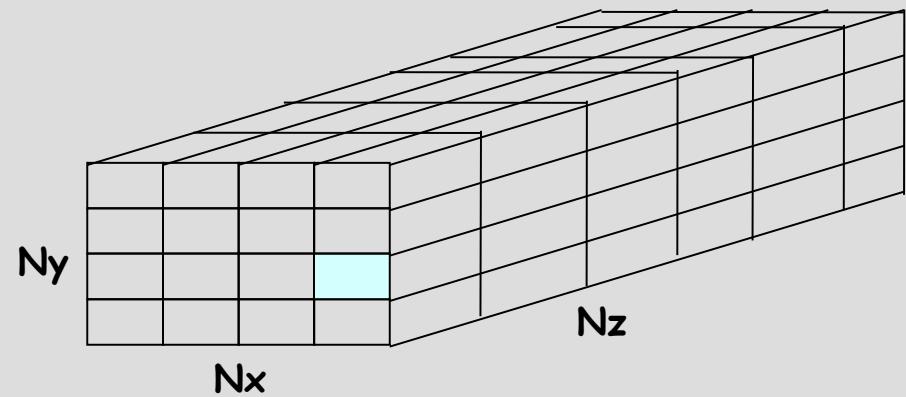
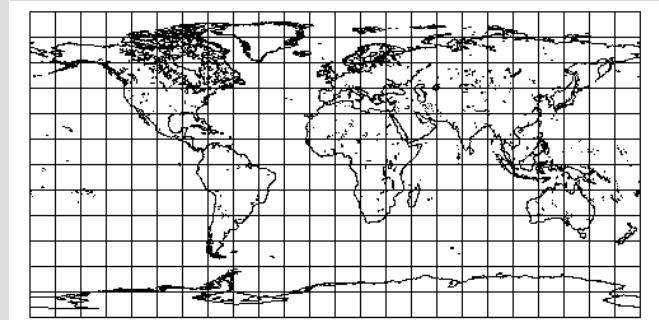
- Dentro de los problemas tratables (tiempo polinomial) existen:
  - **Problemas de gran dimensión:** coste polinomial de grado alto o aplicabilidad a grandes probs.
  - **Problemas de tiempo real**
  - **Problemas de gran desafío:** gran importancia social. Estudio del genoma humano, predicción meteorológica mundial, modelado fenómenos sísmicos, ...

# 1. Introducción

## Necesidad de la Computación paralela. Ejemplo

**Modelar el clima:**  $N_x = N_y = 3000$  Kms,  $N_z = 11$  kms.

- Dominio descompuesto en segmentos cúbicos  $0.1 \times 0.1 \times 0.1 \rightarrow \approx 10^{11}$  segmentos.
- Evolución temporal (2 días): recálculo segmento en paso de tiempo (30 min.).  $\rightarrow 100$  ops.
- Modificación dominio  $\rightarrow 10^{11} \times 100 = 10^{13}$  ops. 2 días  $\rightarrow \approx 100$  pasos de tiempo  $\rightarrow \approx 10^{15}$  ops.
- Ordenador secuencial  $10^9$  inst./seg  $\rightarrow \approx 12$  días !
- Ordenador paralelo con 1000 procesadores ( $10^8$  inst./seg).
  - Dominio descompuesto ( $10^{11}$  segmentos)  $\rightarrow 10^8$  segmentos/proc.
  - Utilización eficiente del poder de cálculo  $\rightarrow$  resolución en **menos de 3 horas**.
- Ventajas Paralelismo:
  - Resolución problemas antes “irresolubles”
  - Mallado más fino para incrementar la precisión

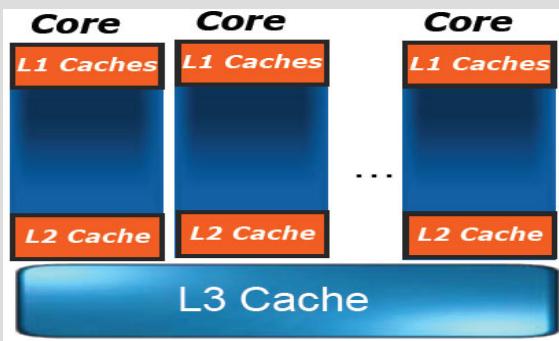


# 1. Introducción

## Computador paralelo. Ventajas

- **Computador paralelo.**

- Multiprocesadores, procesadores multicore, GPUs, etc.
- Varios ordenadores (paralelos o no) interconectados
  - clusters de ordenadores



- **Ventajas**

- Más velocidad de ejecución y precisión cálculos
- Buena relación costo/prestaciones
- Permite atacar problemas considerados irresolubles

# 1. Introducción

## Algunos aspectos de la Programación Paralela

- Diseño de computadores paralelos
  - Escalabilidad hardware
  - Velocidad alta en comunicación.
- Diseño de algoritmos eficientes
  - Minimicen sobrecargas, escalables, independientes arquitectura, etc.
- Lenguajes e interfaces para programación paralela: Flexibilidad
  - Implementación eficiente de algoritmos sobre varias arquitecturas paralelas
  - Programabilidad, aislando programador de detalles hardware.
- Herramientas de programación paralela:
  - Entornos de programación, Depuración, simulación y visualización.
- Métodos de evaluación de algoritmos

# 1. Introducción

## Alcance la computación paralela

### Impacto sobre gran variedad de áreas

- Desde simulaciones para ciencia e ingeniería a aplicaciones comerciales en minería de datos y procesamiento de transacciones.

### Argumento de peso: requisitos rendimiento de aplicaciones

#### • **Aplicaciones en Ingeniería y Diseño**

- Diseño de aviones, Circuitos de alta velocidad, estructuras, etc.....
- Diseño de sistemas nanoelectromecánicos: Múltiples escalas espaciales y temporales, varios fenómenos físicos acoplados, etc.
  - Modelos matemáticos, modelos geométricos, desarrollo de algoritmos, etc.
- Optimización discreta/continua: Optimización lineal, Branch-and-Bound, Progr. Genética.

#### • **Aplicaciones científicas**

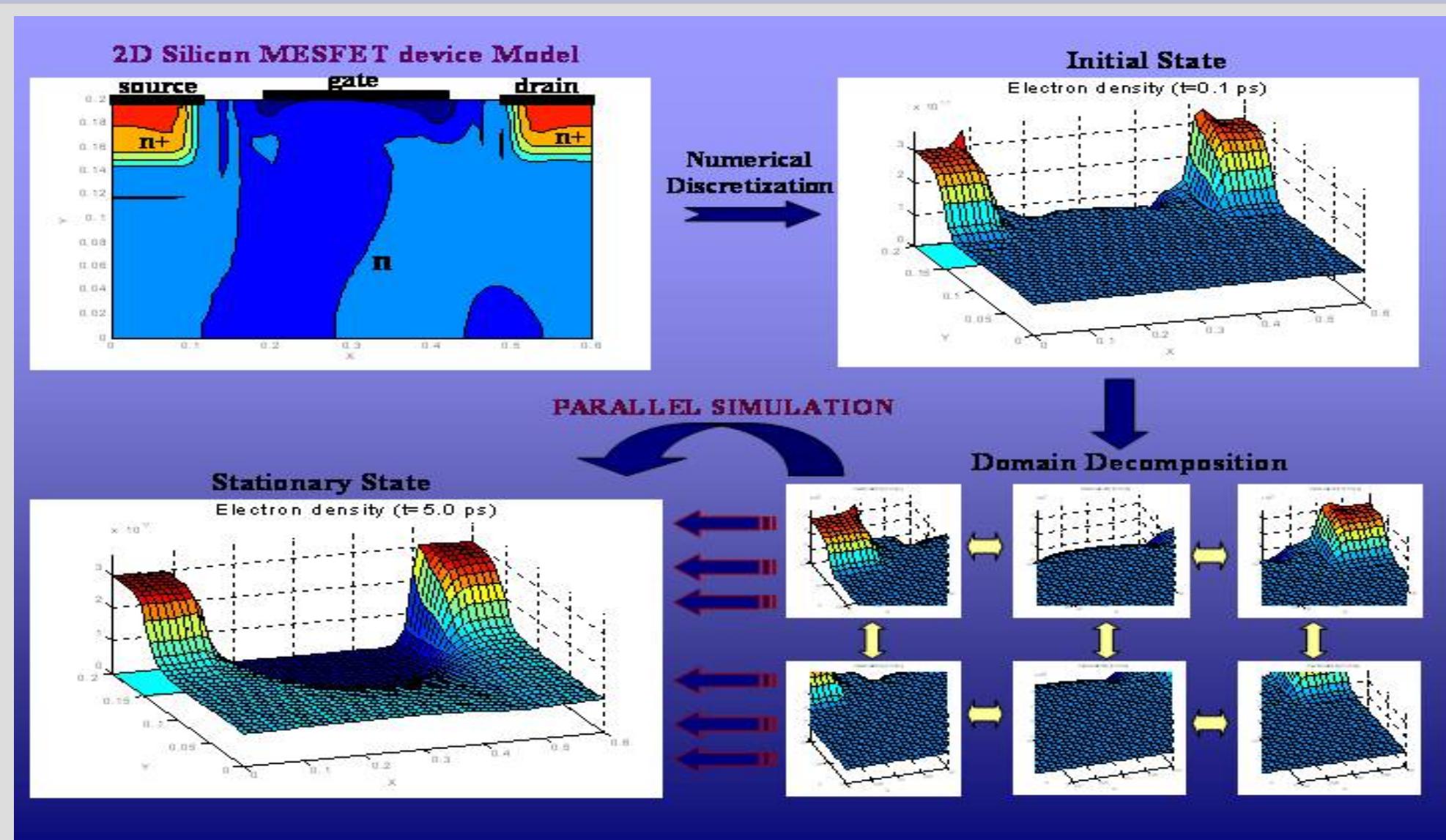
- Bioinformática: Análisis de secuencias biológicas → nuevas medicinas, curar enfermedades
- Física computacional: predicción meteorológica, Astrofísica, predicción inundaciones, etc.

#### • **Aplicaciones Comerciales**

- Grandes servidores de bases de datos: Acelerar búsquedas ...
- Análisis para optimizar negocios y decisiones de mercado, minería de datos, etc.

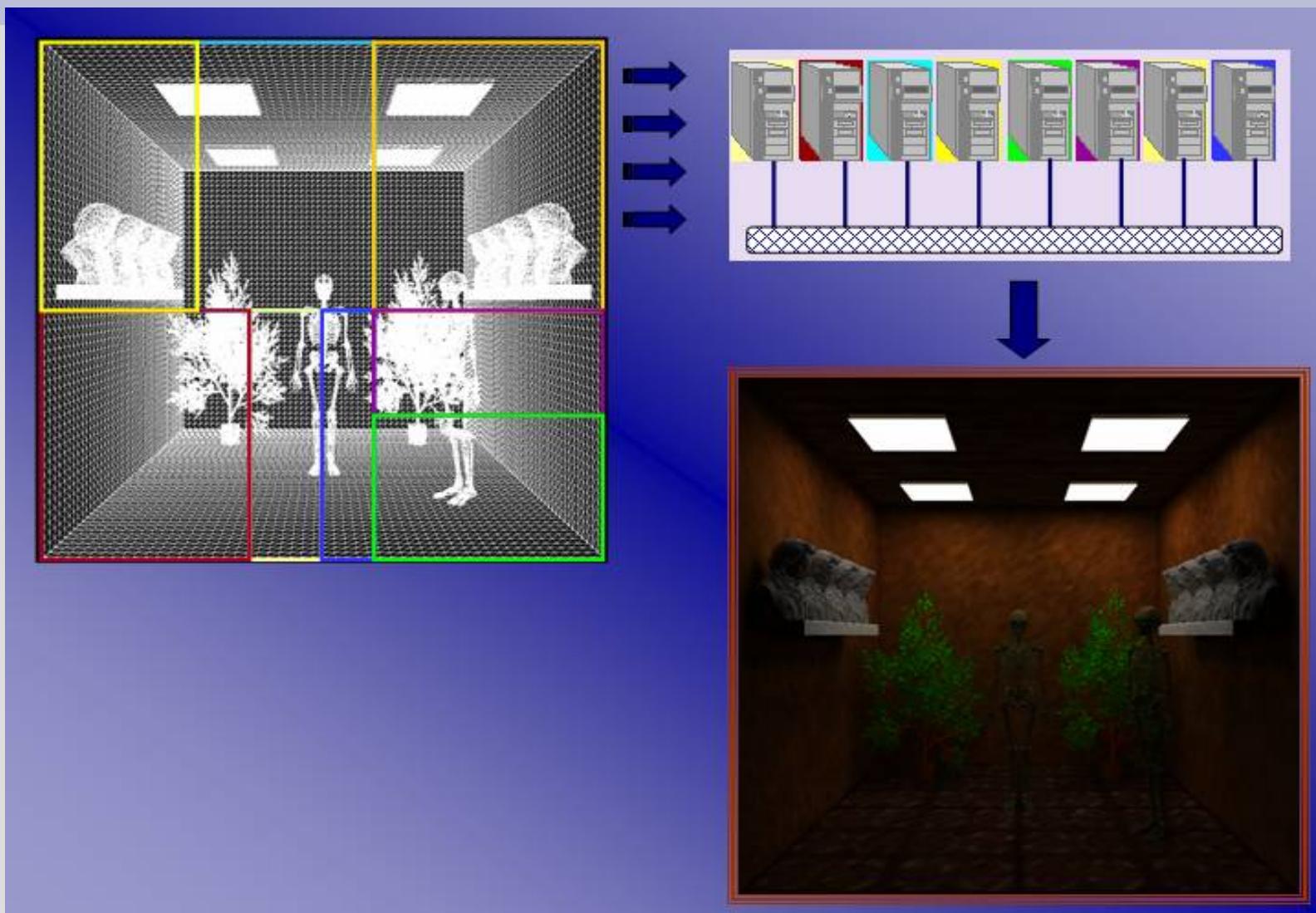
# 1. Introducción

## Ejemplo 1: Simulación de semiconductores 2D



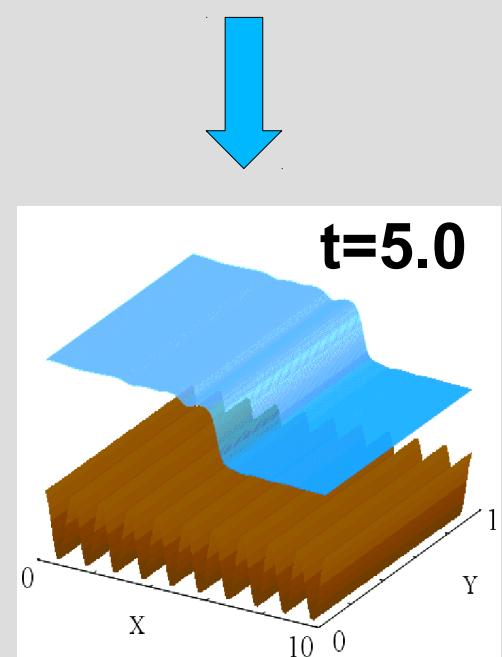
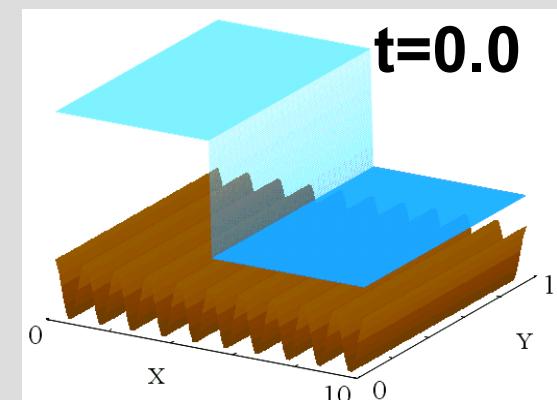
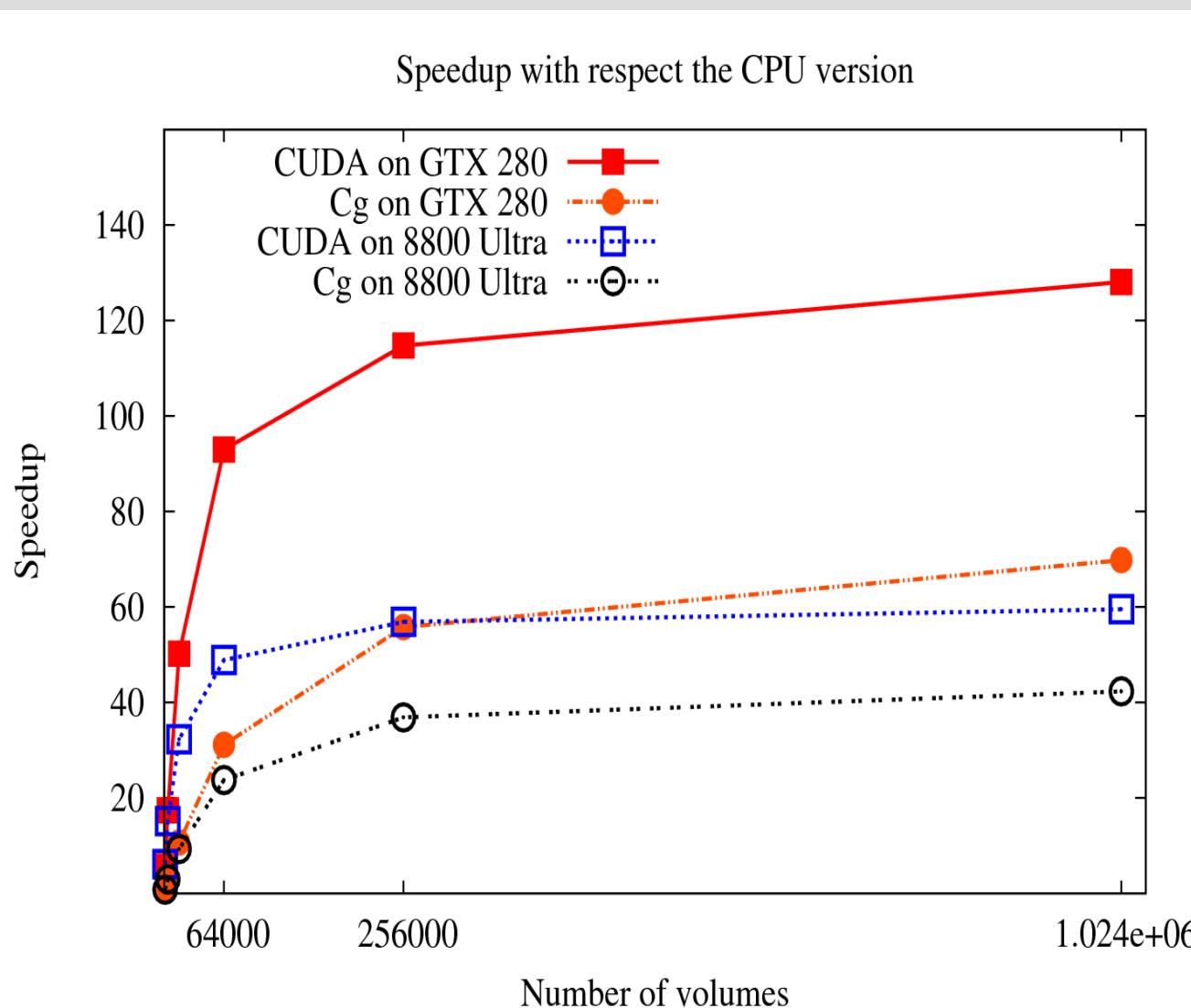
# 1. Introducción

## Ejemplo 2: Síntesis de imágenes Fotorealistas



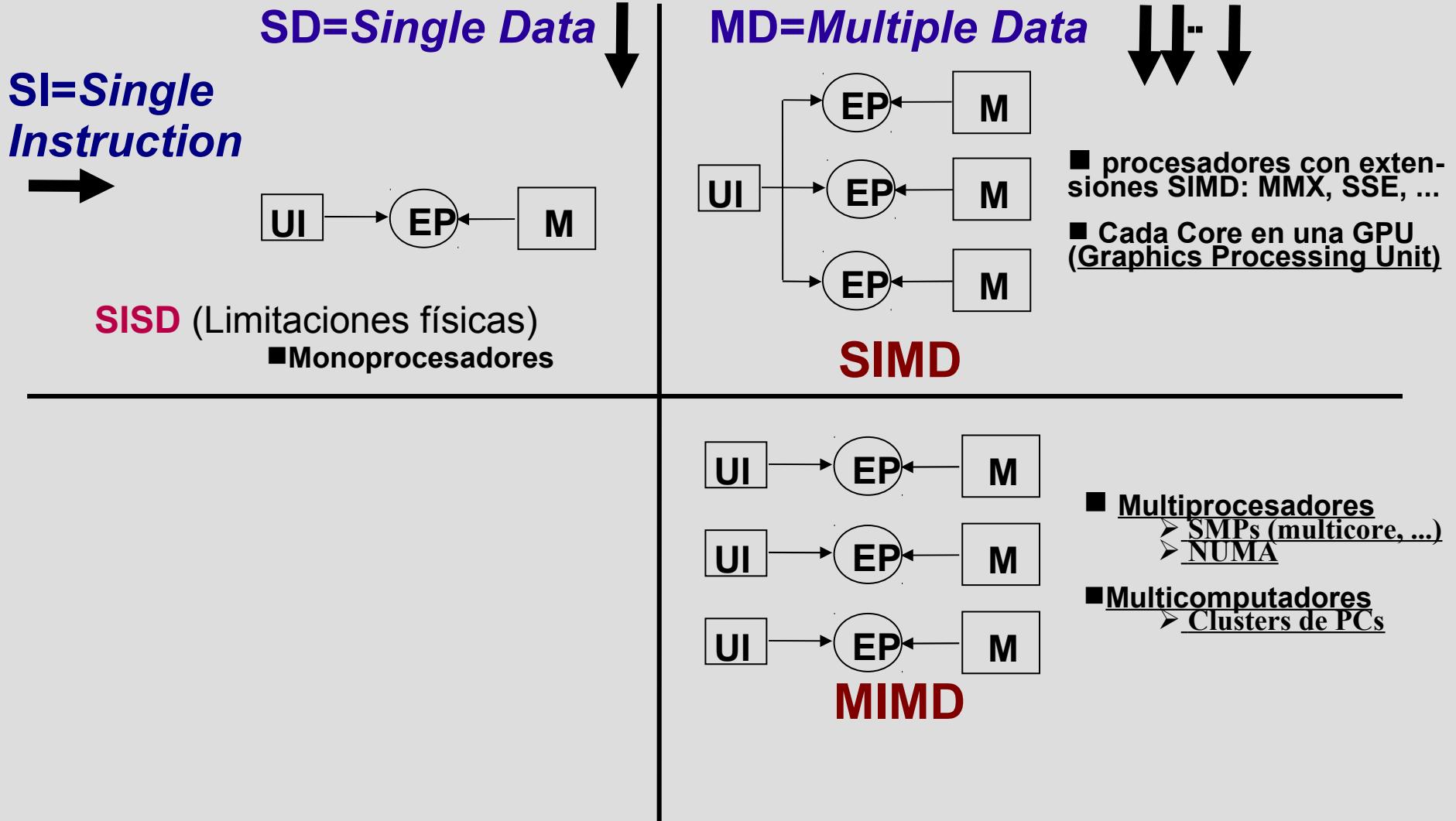
# 1. Introducción

## Ejemplo 3: Simulación de aguas poco profundas en GPUs



# 2. Modelos de Sistemas Paralelos

## Clasificación de Flynn



# 2. Modelos de Sistemas Paralelos

## Arquitecturas Multiprocesador MIMD

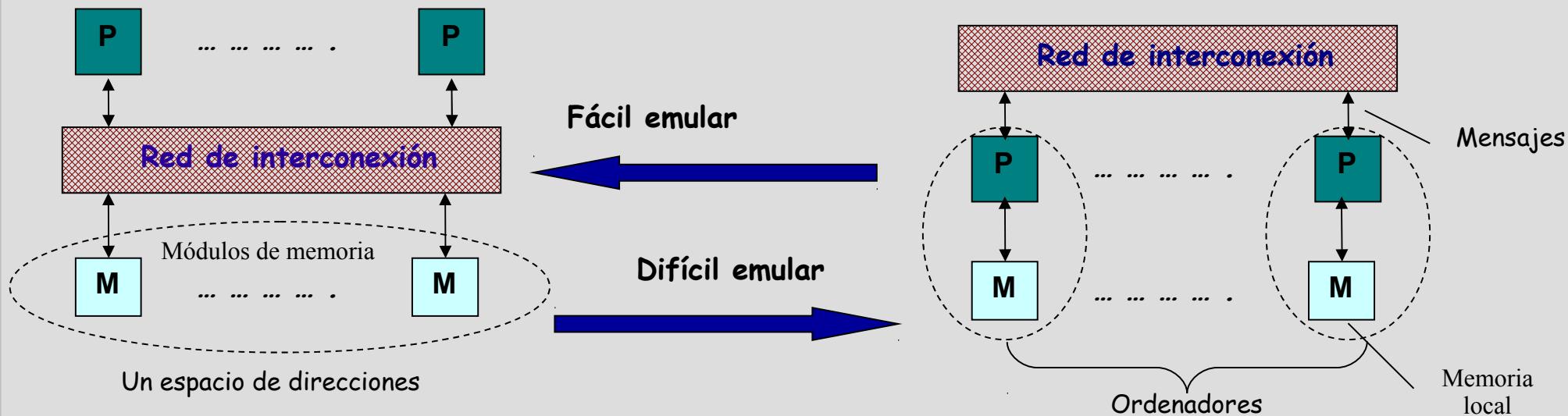
- **Alternativa exitosa:** Utilizar múltiples CPUs y unidades de mem. Interconectadas
  - Ordenador Paralelo MIMD (Multiple Instruction, Multiple Data)
- Criterio: Forma de interacción entre procesadores.

### Multiproc. de mem. Compartida

Procesadores interactúan modificando datos en espacio de direcciones compartido.

### Multiproc. de paso de mensajes

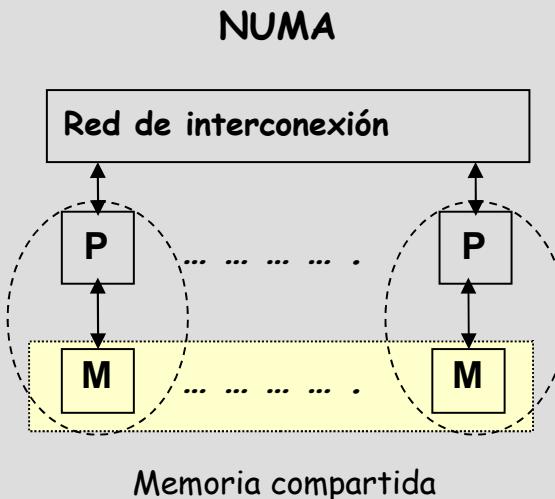
- Memoria local a cada procesador.
- Red de interconexión.
- Interacción via Paso de mensajes.



# 2. Modelos de Sistemas Paralelos

## Multiprocesadores de memoria compartida

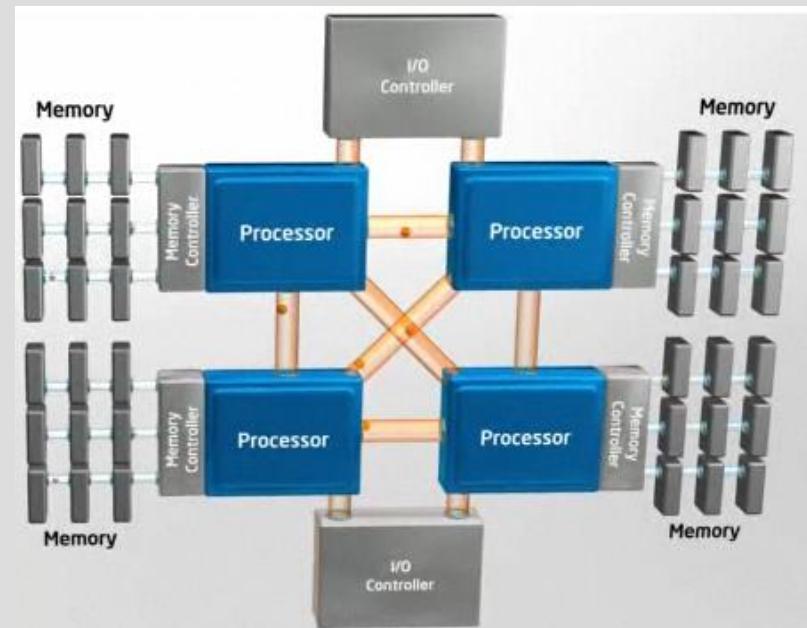
- Espacio de direcciones único
  - Soporte hardware para acceso de lectura y escritura a espacio de direcciones compartido por parte de todos los procesadores.
- 
- Punto de vista del programador:
  - fácil programar con datos compartidos
- Punto de vista del ingeniero de hardware:
  - Complicado implementar acceso rápido a una memoria compartida.
    - Red con elevado ancho de banda: UMA (Uniform Memory Access).
    - Difícil aumentar el nº de procesadores.
  - Implementación práctica:
    - NUMA (NonUniform Memory Access)
    - Estructura jerárquica de memoria.



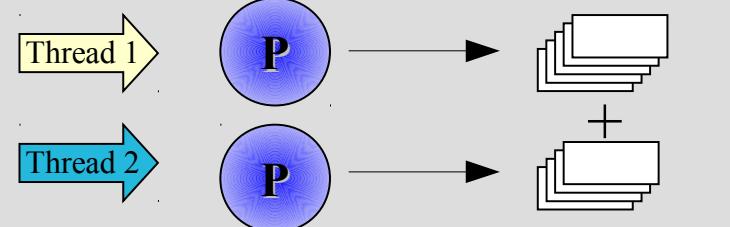
# 2. Modelos de Sistemas Paralelos

## Procesadores multinúcleo (multicore)

- **Arquitectura multicore:** cada procesador contiene dos o más núcleos que pueden ejecutar instrucciones de forma simultánea.
- **Multiprocesador en un solo chip:** el sistema operativo percibe cada núcleo como procesador independiente.
  - Teóricamente, paralelismo perfecto.
- **Ventajas:**
  - Menor consumo y mejor disipación calor.
  - Mejor aprovechamiento multithreading.
- **Ejemplos:** Core i5 /i7 (4/6), IBM Cell (8+1), ...
- **Futuro inmediato:** Ordenadores basados en esta tecnología y adaptación software al nuevo hardware.



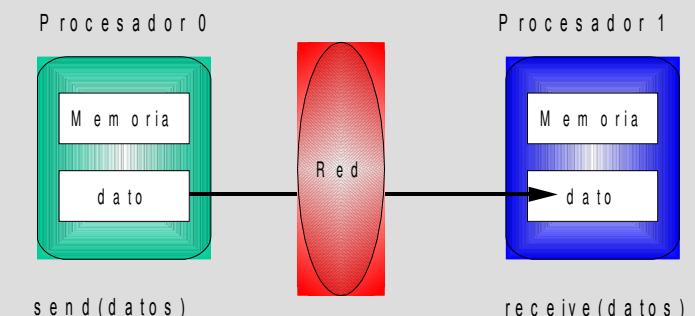
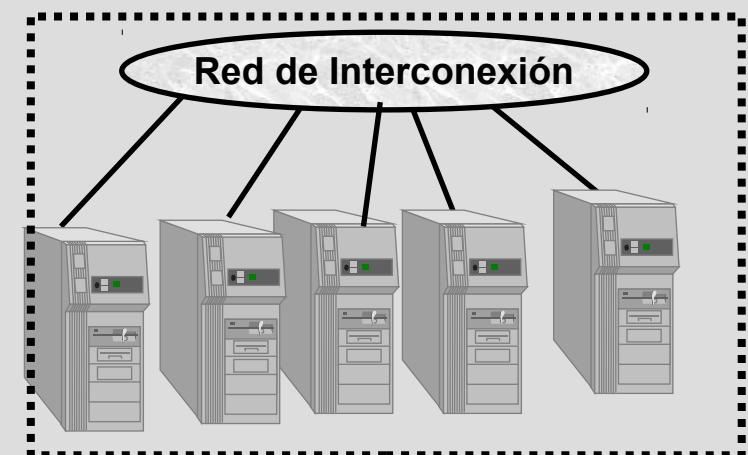
Procesadores  
visibles al Sist.  
Operativo



# 2. Modelos de Sistemas Paralelos

## Multiprocesadores de memoria distribuida

- Conexión de ordenadores mediante red de interconexión. Multicomputadores
  - Cada ordenador con su memoria local y su propio espacio de direcciones
  - Red de interconexión:
    - envío/recepción de mensajes.
- Punto de vista del programador:
  - Difícil programar con paso de mensajes.
  - No se necesitan mecanismos de Exclusión mútua
  - Aplicable a redes de ordenadores
    - Mayor flexibilidad y adaptación a los avances tecnológicos
- Punto de vista del ingeniero de hardware:
  - Mejor escalabilidad que multiprocs de memoria compartida



# 2. Modelos de Sistemas Paralelos

## Clusters de ordenadores

Conexión de ordenadores de uso convencional (monoprocesador o multiprocesador) mediante red de interconexión de alta velocidad donde **el conjunto se ve como un único ordenador**.

- **Ventajas sobre multiprocesador normal**
  - Disponibilidad a bajo costo de potentes estaciones de trabajo y PCs
  - Fácil incorporación de últimos avances en CPUs y redes
  - Usa software ya existente.

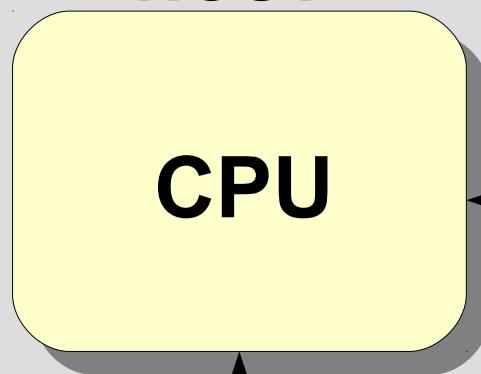
- **Inconvenientes:** cada vez menos
  - Se tiende a mejorar con avances software/hardware de redes



## 2. Modelos de Sistemas Paralelos

### Procesadores Gráficos (GPUs)

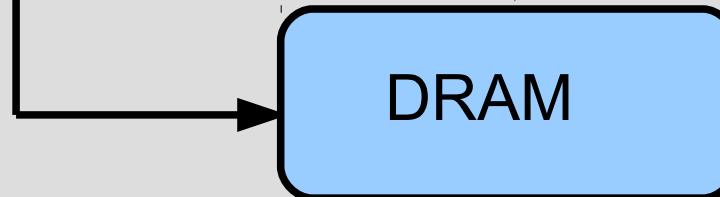
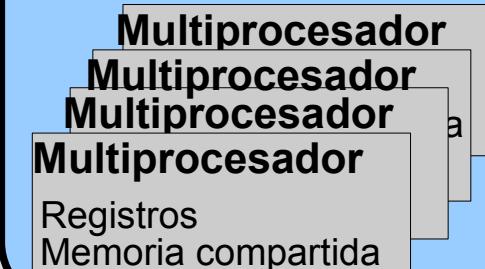
HOST



DEVICE



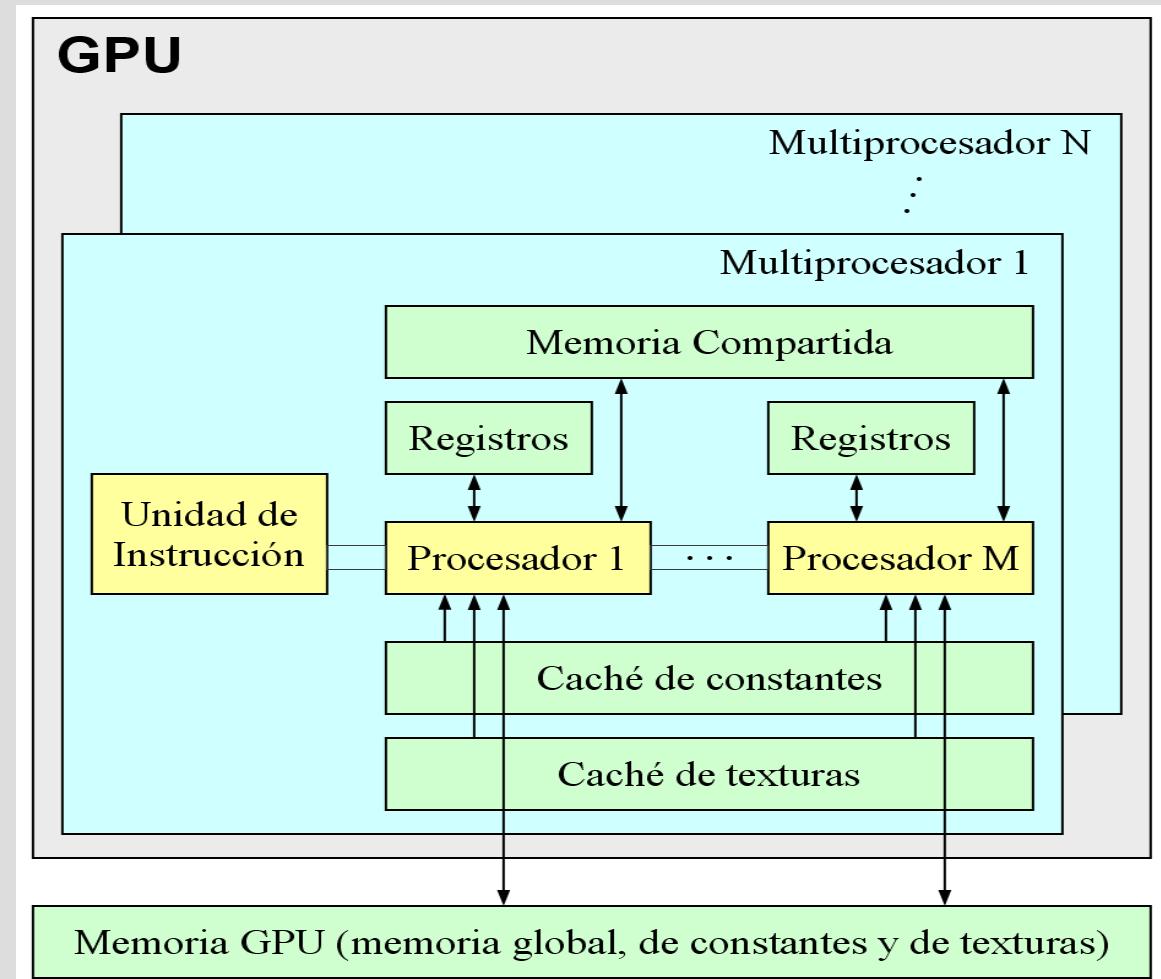
GPU



# 2. Modelos de Sistemas Paralelos

## Procesadores Gráficos (GPUs)

- LA GPU contiene **N multiprocesadores**.
- Cada multiprocesador incluye:
  - **M procesadores**
  - Banco de **registros**
  - **Memoria compartida**: muy rápida, pequeña.
  - **Cachés** de ctes y de texturas (sólo lectura)
- La **memoria global** es 500 veces más lenta que la memoria compartida

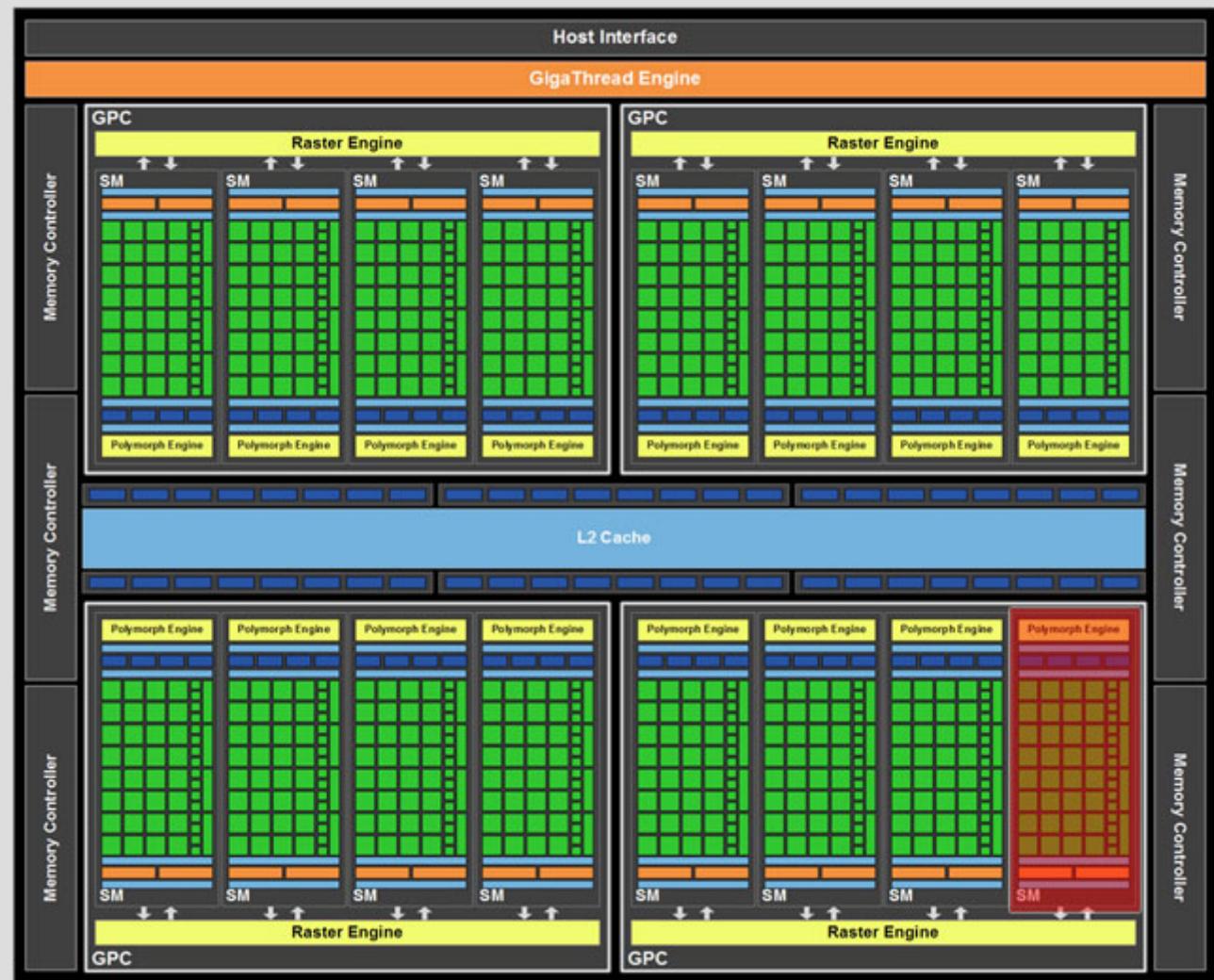


# 2. Modelos de Sistemas Paralelos

## Procesadores Gráficos (GPUs)

### Fermi (GT400)

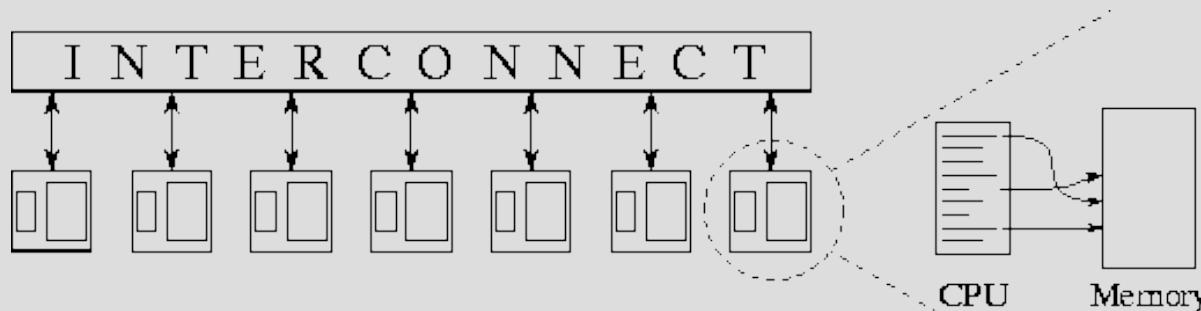
- 512 cores
- 16 multiprocs.
- Cada multiprocs. con 32 cores y 16 Unidades de doble precisión.
- 64 KB. de SRAM a repartir entre memoria compartida y Cache L1.



# 2. Modelos de Sistemas Paralelas

## Modelo abstracto. Multicomputador ideal

- Modelo simple y realista
- Nodos secuenciales (CPU + memoria) enlazados con *red de interconexión*.
- Cada nodo ejecuta su propio programa: podría acceder a memoria local o enviar/recibir mensajes (comunicación remota).
- Costo de comunicación es independiente de localización del nodo y tráfico en la red, pero depende de la longitud del mensaje.
- Costo (acceso a memoria local) < Costo (acceso remoto).
- **Localidad:** indica proporción costos accesos locales y remotos.



# 3. Modelos de Programación Paralela

## Definición y Propiedades deseables

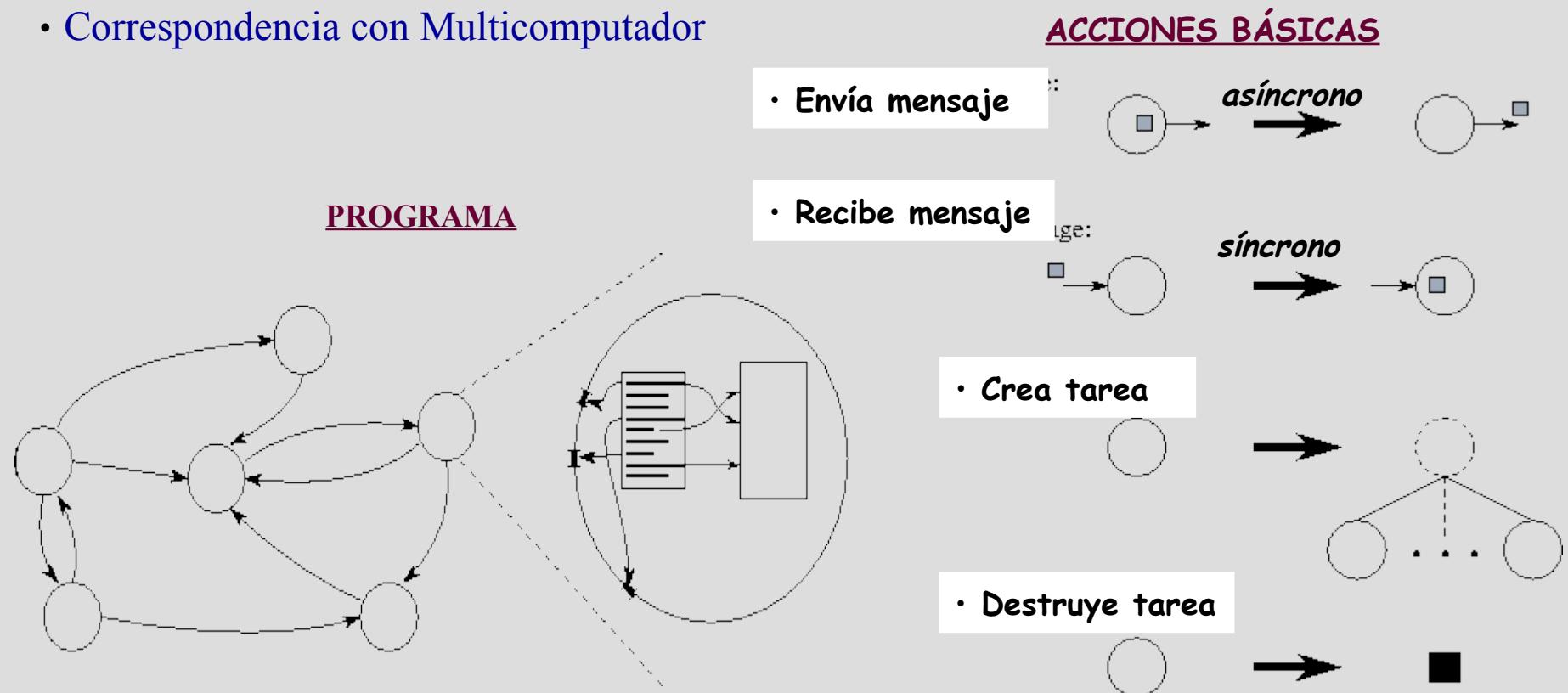
**Modelo de Progr. Paralela** → Mecanismo para el desarrollo de programas paralelos eficientes

- Debe permitir describir **conurrencia**, ocultar complejidad máquina paralela (**abstracción**), obtener programas que den buen rendimiento sobre diferentes máquinas (**portabilidad**) y usando técnicas de **diseño modular**.
- **Escalabilidad:** Los programas desarrollados deben poder ejecutarse de forma eficiente sobre un número creciente de procesadores.
- **Independencia de la asignación :** Diseño e implementación de las tareas es indep. del número de procesadores físicos sobre los que se ejecutarán.
- **Modelo de costo:** estimaciones fiables del costo de un programa en etapas tempranas del desarrollo.

# 3. Modelos de Programación Paralela

## Paso de Mensajes

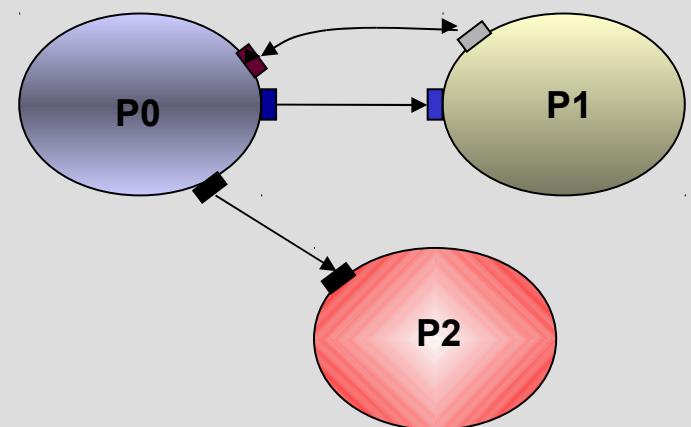
- Computación paralela = una o más tareas concurrentes. Número de tareas variable.
- Tarea = programa secuencial + memoria local + identificación dentro de contexto de comunicación
- Correspondencia con Multicomputador



# 3. Modelos de Programación Paralela

## Paso de Mensajes (2)

- Envío/Rec. a tareas nombradas
  - SEND (valor, P0)
  - RECEIVE (buffer, P1)
  - Mecanismos para diferenciar mensajes
- Práctica: Modelo estático SPMD (*Single Program Multiple Data*)
  - Cada tarea ejecuta mismo programa sobre diferentes datos
  - Amplio rango de aplicaciones pero limita.
- Superar limitaciones SPMD
  - Mecanismos para ejecución de programas SPMD por grupos de procesadores
    - MPMD (*Multiple Program Multiple Data*)
- Mayor difusión
  - Válido para un amplio rango de arquitecturas (MIMD).



# 3. Modelos de Programación Paralela

## Paso de Mensajes (3)

- Enfoque más común para Programación paralela con paso de mensajes
  - Lenguaje secuencial de alto nivel (C, C++) + Biblioteca de paso de mensajes (MPI):
    - La biblioteca ofrece procedimientos externos para envío/recepción de mensajes entre procesos.
- Procedimientos básicos
  - Creación de procesos separados para ejecución sobre diferentes máquinas
  - Envío y recepción de mensajes

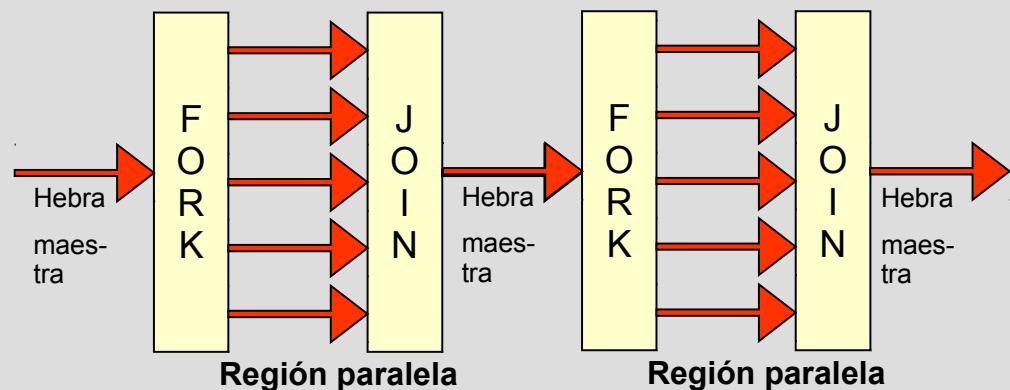
# 3. Modelos de Programación Paralela

## Memoria compartida

- Tareas/Hebras comparten un espacio de memoria común donde leen y escriben asíncronamente
- Sincronización y Comunicac. mediante variables. compartidas
- Mecanismos control acceso: cerros, semáforos, monitores, etc.
- Simplifica el desarrollo de programas
  - No es necesario especificar explícitamente la comunicación de datos
- Eficiente sólo en arquitecturas de memoria compartida, difícil razonar sobre la localidad y escribir programas deterministas.



**OpenMP:** Notación basada en memoria compartida



# 4. Análisis del rendimiento

## Factores que influyen en el rendimiento paralelo

- Modelos de rendimiento: Modelos matemáticos
  - Evaluación costes de alternativas de diseño antes implementación
    - Explicar observaciones
    - Predecir circunstancias futuras
    - Abstracción de los detalles del hardware
- Factores: Requisitos de memoria, Tiempo de ejecución, Precisión, Escalabilidad, Costes de diseño e implementación, etc.
- Importancia relativa: Depende del problema
  - Sistema paralelo de previsión Meteorológica
    - Tiempo de ejecución + precisión + escalabilidad.
  - Búsqueda paralela en una base de datos de ingeniería
    - Costes de implementación y mantenimiento.
- Más Problemáticos: tiempo de ejecución y escalabilidad



# 4. Análisis del rendimiento

## Enfoques tradicionales. Ley de Amdahl

- **Ley de Amdahl:** Cada alg. tiene una parte intrínsecamente secuencial (no paralelizable) que eventualmente limitará la ganancia de velocidad alcanzable en un ordenador paralelo.
  - Coste componente secuencial de alg.=  $1/s$  del total
  - Tiempo mínimo de ejecución paralela =  $T/s \rightarrow$  Máxima ganancia= $s$
- **Válidez del enfoque:** Paralelización incremental de algoritmos secuenciales.
  - Las partes más costosas se identifican y se paralelizan.
    - Costes partes no paralelizadas dan cota inferior.
- Ejemplo: Si el 20% del algoritmo es no paralelizable, entonces la máxima ganancia es 5.
- No tiene en cuenta los cambios en el tamaño del problema!
  - Válido para tamaño de problema fijado. Al escalar el problema “la proporción cambia”.
  - Los costes (partes secuenciales y paralelizables) dependen de parámetros de tamaño.
  - **Paralelización interesa cuando el orden del coste de la componente secuencial sea menor que el orden del coste paralelizable.**

# 4. Análisis del rendimiento

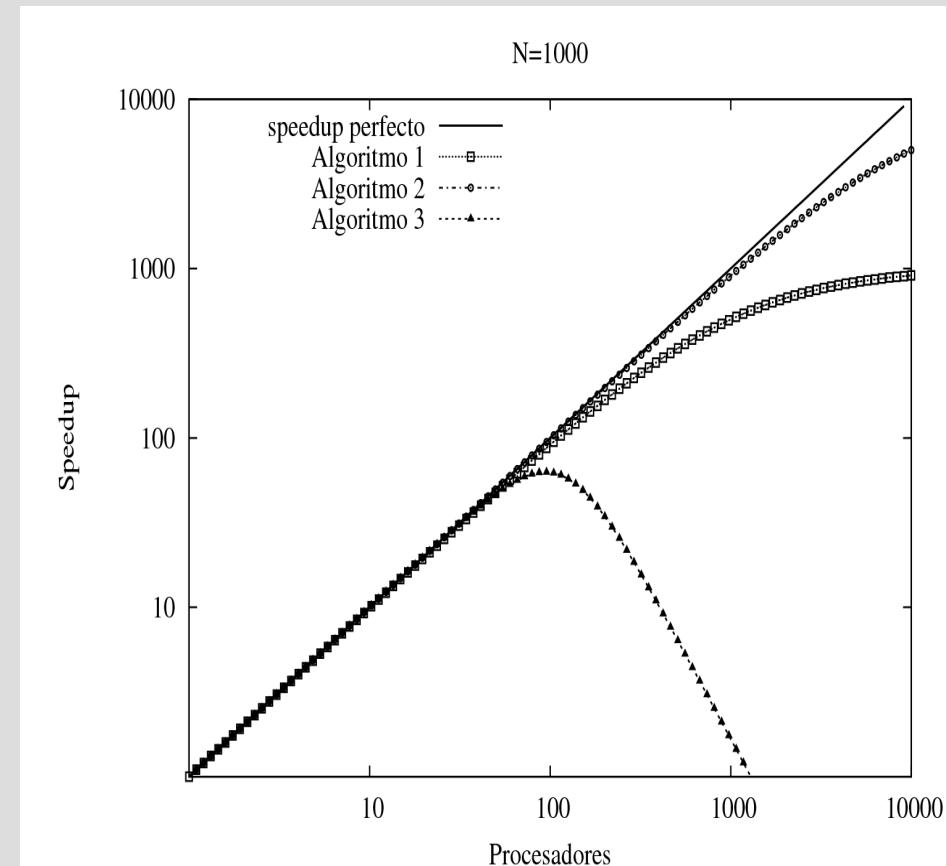
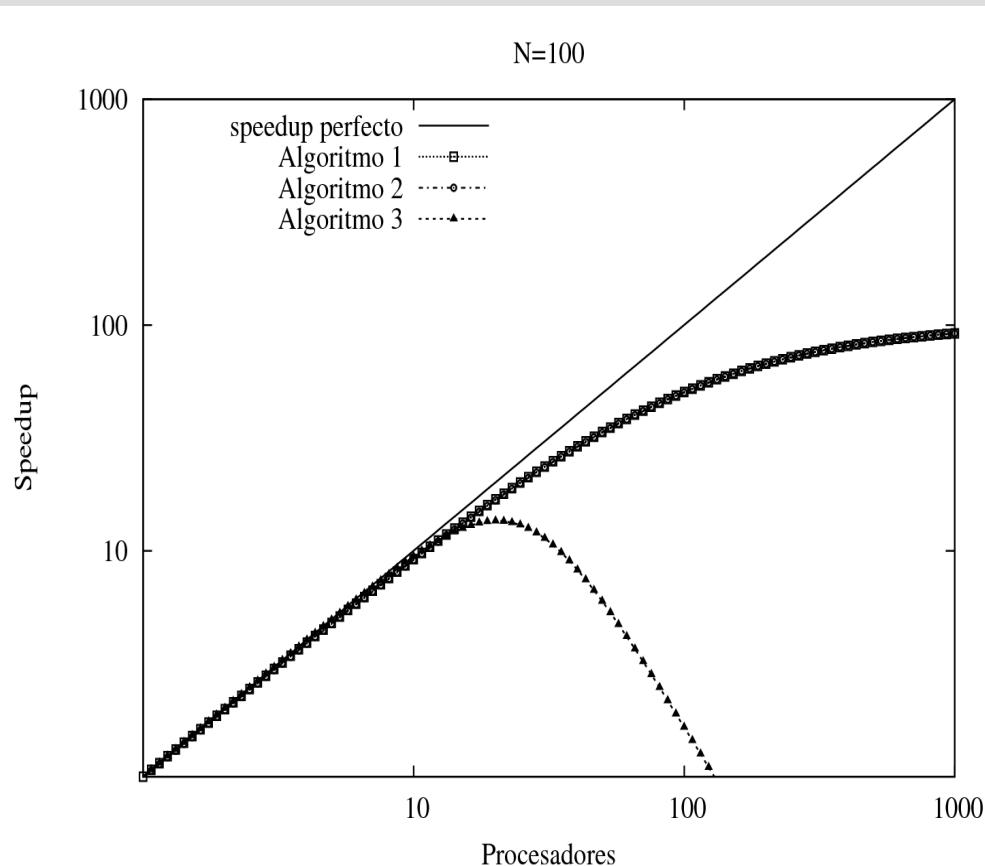
## Enfoques tradicionales. Extrapolación de Observaciones

- Una implementación de un algoritmo sobre computador X alcanza una ganancia de 10.8 con 12 procesadores con tamaño de prob.  $N = 100$ .

$$\text{Alg. 1: } T=N+N^2/P$$

$$\text{Alg. 2: } T=(N+N^2)/P + 100$$

$$\text{Alg 3: } T=.(N+N^2)/P + 0.6P^2$$



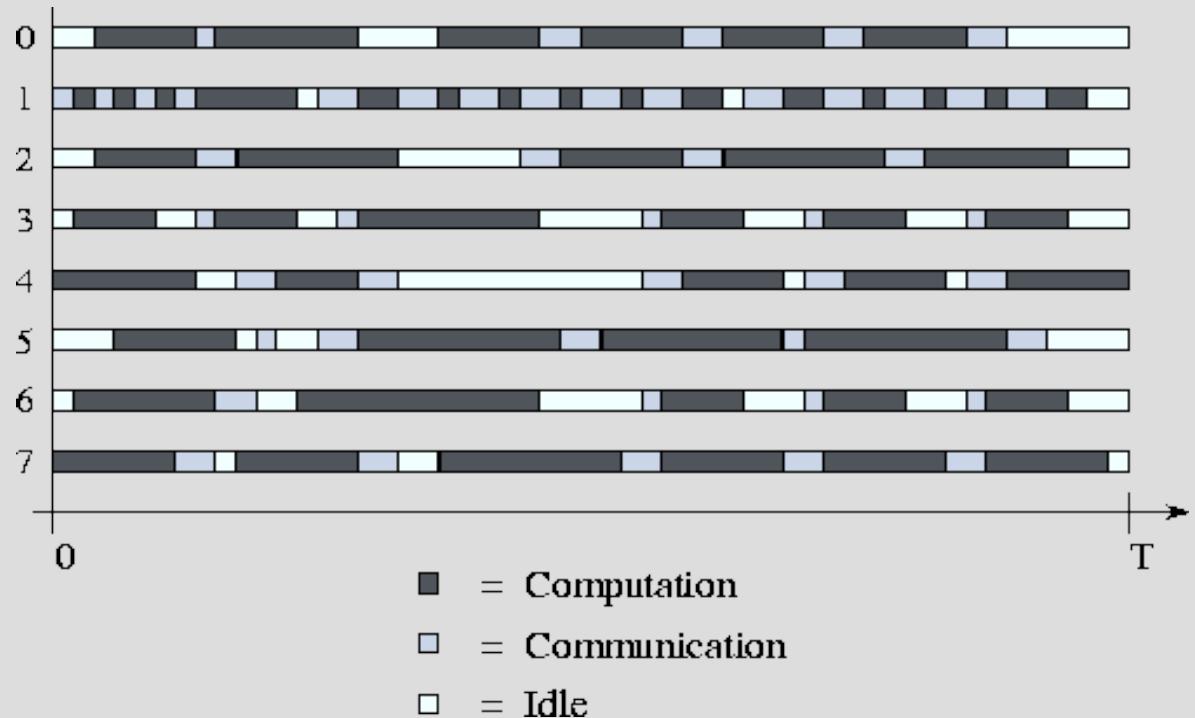
# 4. Análisis del rendimiento

## Tiempo de ejecución paralelo

- Nivel intermedio de detalle : Válido para Arquitecturas multicomputador
- Tiempo de ejecución paralelo: Tiempo transcurrido desde que el primer procesador empieza a ejecutar el programa hasta que el último termina su ejecución.

$$T_p = T_{comp} + T_{com} + T_{ocio}$$

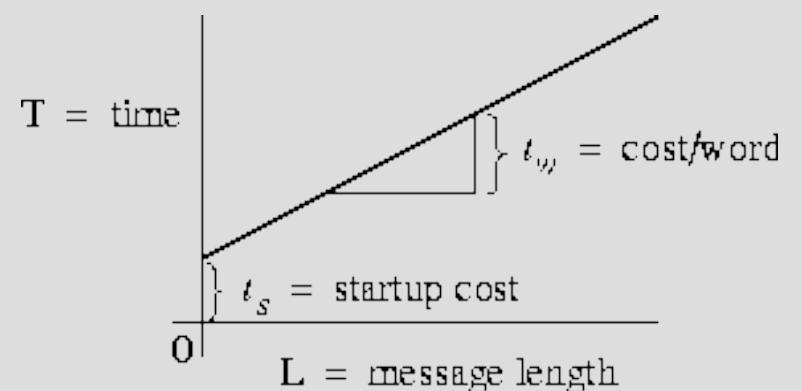
$$T_p = f(N, P, \dots)$$



# 4. Análisis del rendimiento

## Tiempos de computación y comunicación

- **Tiempo de computación:** Tiempo gastado en cómputo con datos locales
  - Depende del tamaño del problema y del nº de procesadores
  - Las características de los procesadores influyen
    - Tiempo gastado en operaciones aritméticas
    - Complicado en sistemas heterogéneos
- **Tiempo de comunicación:** Tiempo gastado en envío/recepción entre tareas.
  - Modelo de coste de comunicación para un envío/recepcion
    - $T_{msg}(L) = ts + tw L$
    - $ts$  =tiempo de inicialización de mensaje.
    - $tw$  =tiempo de transferencia por palabra.



# 4. Análisis del rendimiento

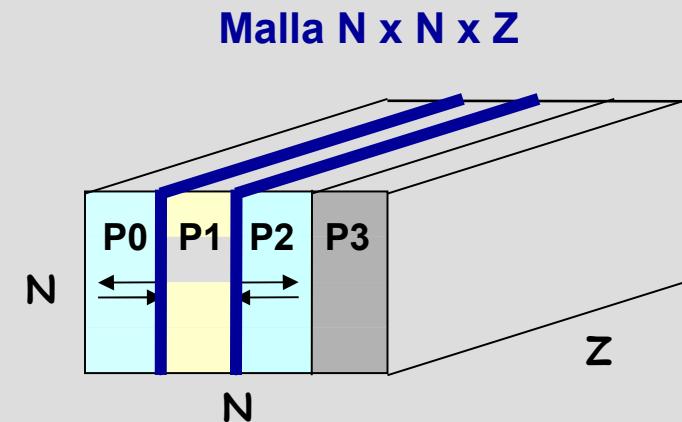
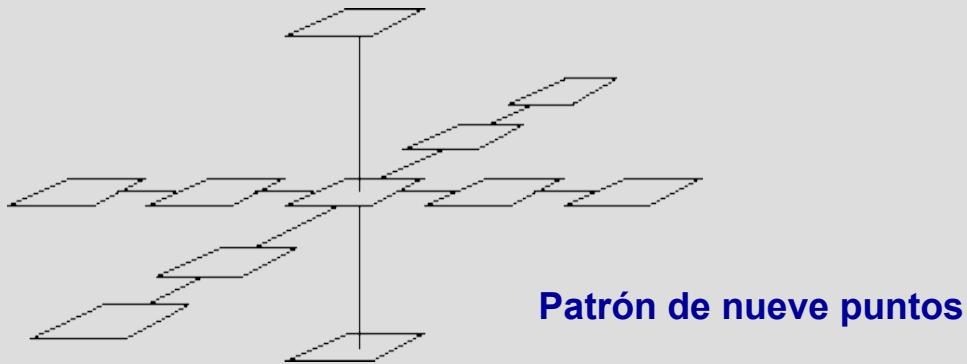
## Tiempo de ocio

- Tiempo de ocio
  - Tiempo gastado sin hacer nada útil
  - Difícil de determinar:
    - depende del orden en el que se realizan las operaciones
  - Lo ideal es reducirlo al máximo en el diseño.
- Motivos y posibles soluciones
  - Falta de computación → Técnicas de equilibrado de carga
  - Falta de datos → Solapando comunicación y computación
    - Creando varias tareas por procesador (programación multihebrada)
    - Estructurando tarea para que las peticiones remotas se entrelacen explícitamente con la computación.



# 4. Análisis del rendimiento

## Ejemplo: Diferencias finitas



- Descomponemos dimensión horizontal.
  - Tarea se asocia a submalla  $N \times (N/P) \times Z$ .
- Cada tarea realiza la misma computación sobre cada punto en cada paso.
  - $T_{comp} = tc \frac{N^2}{P} Z$ ;  $tc$  = tiempo medio de comp. por punto.
- Cada tarea intercambia  $2NZ$  puntos con dos vecinas:
  - Dos mensajes y  $2NZ$  puntos/mens. ( $N=2kP$ ,  $k$  natural)
  - $T_{comm} = 2(t_s + t_w 2 NZ)$
- Computación por punto cte  $\rightarrow T_{ocio} \approx 0$ .

$$T_p = T_{comp} + T_{comm} = tc \frac{N^2}{P} Z + 2t_s + 4t_w NZ$$

# 4. Análisis del rendimiento

## Ganancia en velocidad. Speedup

- Cuantificar beneficio de resolver un problema en paralelo.
- **Ganancia en velocidad:** razón entre tiempo de ejecución monoprocesador y el tiempo paralelo sobre P procesadores idénticos.  
$$S = T_1/T_P \quad T_1 = \text{tiempo del mejor alg. secuencial}$$
- Teóricamente,  $S \leq P$ .
- A veces  $S > P$ . → Speedup superlineal
  - Alg. secuencial no óptimo
  - Características hardware que ponen alg. secuencial en desventaja.
    - Ej.: Los datos agotan capacidades de almacenamiento del sistema monoprocesador.
- Coste de un sistema Paralelo:  $C = P \cdot T_p$ 
  - Tiempo total consumido por todos los procs.

# 4. Análisis del rendimiento

## Eficiencia

- Tiempo de ejecución no siempre es la métrica más conveniente para evaluar el rendimiento de un algoritmo.
  - Tiende a variar con el nº de procs.
  - Tiempos de ejec. deben ser normalizados cuando se compara el rendimiento del alg. sobre diferentes números de procs.
- **Eficiencia** → Fracción de tiempo procs. realizan trabajo útil.
$$E = S/P = T_1/(PT_P)$$
- **Sistema ideal** →  $S = P \rightarrow E=1$ . Generalmente:  $0 < E < 1$ .
- $E$  tiende a aumentar conforme aumenta el tamaño del problema y tiende a disminuir conforme aumenta nº procs ( $P$ ).

# 4. Análisis del rendimiento

## Eficiencia del algoritmo de diferencias finitas

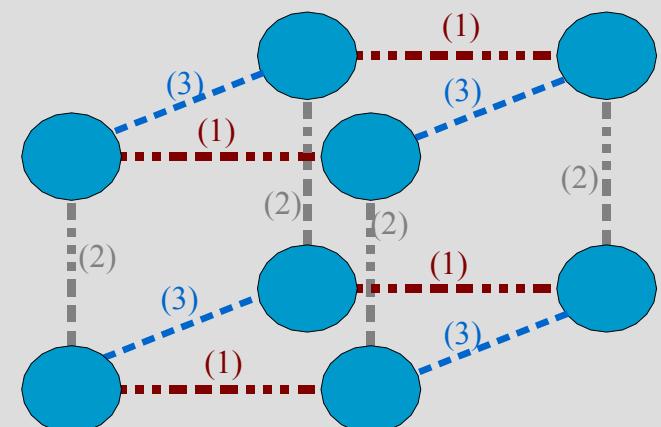
- $T_1 = t_c N^2 Z$
  - $T_p = (T_{\text{comp}} + T_{\text{comm}})/P = t_c N^2 Z / P + 2t_s + 4t_w N Z$
- $$S = \frac{t_c N^2 Z}{\frac{t_c N^2 Z}{P} + 2t_s + 4N Z t_w} \Rightarrow E = \frac{t_c N^2 Z}{t_c N^2 Z + 2Pt_s + 4PNZt_w}$$
- Conclusiones
    - E **baja** al incrementar P, ts **y** tw.
    - E **sube** al incrementar N, Z **y** tc.
    - $T_p$  **baja** al incrementar P, pero está acotado inferiormente por el coste de intercambiar dos trozos de array.
    - $T_p$  **se incrementa** al incrementar N, Z, tc, ts **y** tw.

# 4. Análisis del rendimiento

## Suma de N números en hipercubo con P procesos

1. Asignamos  $N/P$  elementos a cada proc ( $N = k_1 * P$ ,  $P = 2^k$ )
2. Cada proc. obtiene la suma del grupo de elementos asignados
3. Para  $i=1, \dots, \log(P)$ , procs. conectados en  $i$ -ésima dimensión:
  - Intercambian resultados locales.
  - Actualizan resultado con valor obtenido.

- 1º suma de  $N/P$  elem.
- 
- Después 1 suma para cada dim.  
 $T_{comp} = tc(N/P - 1) + tc \log P$
- 1 envío/recepción para cada dim.  
 $T_{comm} = \log P (ts + tw)$



$$T_p = T_{comp} + T_{comm} = tc(N/P - 1) + \log P (tc + ts + tw)$$

# 4. Análisis del rendimiento

## Suma de N números en hipercubo con P procesos (2)

- Suma sec.:  $T_1 = tc(N-1)$ ,  $tc=t_{\text{suma}}$
- $T_p = T_{\text{comp}} + T_{\text{comm}} = tc(N/P - 1) + \log P (tc + ts + tw)$
- $S = tc(N-1)/(tc(N/P - 1) + \log P (tc + ts + tw))$
- $E=S/P \approx tcN/(tcN + P \log P (tc + ts + tw))$

# 4. Análisis del rendimiento

## Análisis de Escalabilidad

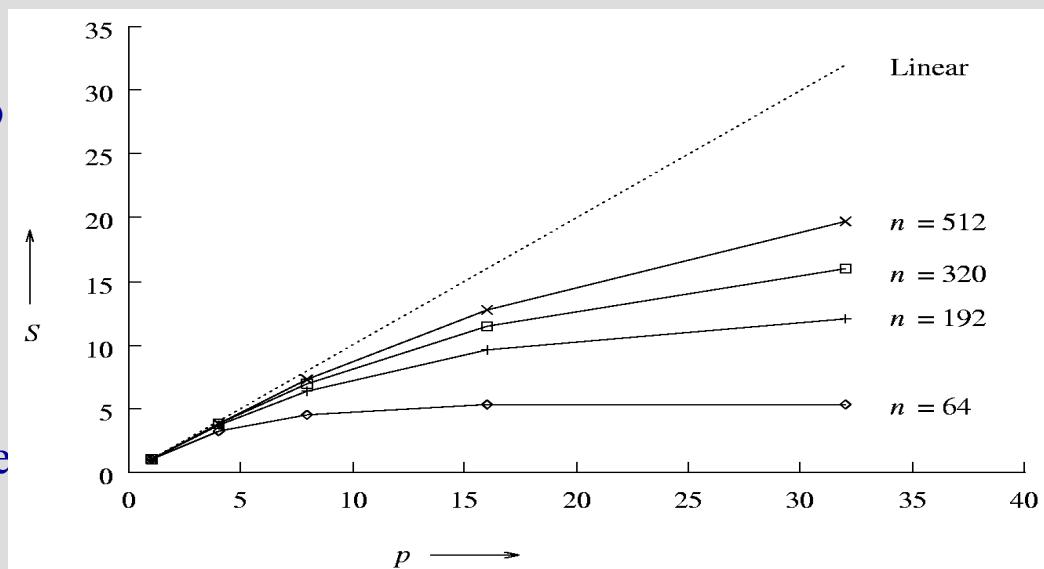
### Suma de N nums. en hipercubo de P procs.

- Suposición:  $tc = (ts + tw) = 1$
- $T_1 \approx N$
- $T_p = N/P - 1 + 2 \log P \approx N/P + 2 \log P$
- $S = N/(N/P + 2 \log P)$
- $E = N/(N + 2P \log P)$

$n$	$p = 1$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
64	1.0	.80	.57	.33	.17
192	1.0	.92	.80	.60	.38
320	1.0	.95	.87	.71	.50
512	1.0	.97	.91	.80	.62

### Observaciones

- $S$  no se incrementa linealmente con  $P$  sino que tiende a saturarse.
- Aumentar  $N$  produce aumento de  $S$  y  $E$  para  $P$  fijo
- Debería ser posible incrementar tanto  $P$  como  $N$  manteniendo  $E$  cte.
- En Tabla,  $E=0.8$  para diferentes valores de  $N$  y  $P$ .



# 4. Análisis del rendimiento

## Definición de Escalabilidad. Sistemas Escalable

- **Sistema Escalable:**
  - Es posible mantener E cte incrementando simultáneamente el tamaño del problema (W) y P.
- **Escalabilidad de un sist. Paralelo**

Capacidad de incrementar S en proporción a P.

  - Refleja capacidad de utilizar más recursos de manera efectiva.
- Ejemplo: Suma en hipercubo, N=8 P log P → E=0.8

<b><i>n</i></b>	<b><i>p = 1</i></b>	<b><i>p = 4</i></b>	<b><i>p = 8</i></b>	<b><i>p = 16</i></b>	<b><i>p = 32</i></b>
64	1.0	.80	.57	.33	.17
192	1.0	.92	.80	.60	.38
320	1.0	.95	.87	.71	.50
512	1.0	.97	.91	.80	.62

# 4. Análisis del rendimiento

## Isoeficiencia

- Ritmo incremento W en función de P para mantener E fija
  - Determina grado de escalabilidad sistema.
- **Tamaño de problema:** No basta parámetro del tamaño de la entrada → las interpretaciones dependen del problema.
  - $W = \text{nº de pasos de comp. básicos mejor alg. secuencial} = f(\text{tamaño entrada})$ .
- **Función de Sobrecarga:** Parte del coste no incurrida por mejor alg. secuencial.

$$T_o(W, P) = PT_P - W$$

Eficiencia

Tiempo de Ejecución  $\rightarrow$  Ganancia  $\rightarrow$   $E = \frac{W}{W + T_o(W, P)} = \frac{1}{1 + \frac{T_o(W, P)}{W}}$

$$T_P = \frac{W + T_o(W, P)}{P}$$
$$S = \frac{W}{T_P} = \frac{WP}{W + T_o(W, P)}$$

Para un valor de E fijo:

$$\frac{W}{T_0(W, P)} = \frac{E}{1 - E} K$$

Función de Isoeficiencia

$$W = K \cdot T_0(W, P)$$

# 4. Análisis del rendimiento

## Isoeficiencia(2)

- **Función de isoeficiencia**
  - Indica ritmo crecimiento de  $W$  para mantener  $E$  fija según  $P$  crece.
  - Determina facilidad con la que un sistema puede mantener  $E$  cte y obtener ganancias crecientes en proporción a  $P \rightarrow$ **Escalabilidad**.
- Isoeficiencia baja  $\rightarrow$  incrementos pequeños en  $W$  permiten utilización eficiente + procs. $\rightarrow$  Sist. altamente escalable.
  - Ej.:  $W$  crece linealmente con  $P$
- Isoeficiencia alta  $\rightarrow$  Sistema poco escalable.
  - Ej.:  $W =$  función exponencial de  $P$
- Función de isoeficiencia NO existe para sistemas no escalables

# 4. Análisis del rendimiento

## Isoeficiencia de la suma en hipercubo

$$T_P \approx N/P + 2\log P$$

$\left. \begin{array}{c} \\ W \approx N \end{array} \right\} \longrightarrow T_0(W, P) = PT_P - W \approx 2P\log P$

Función de isoeficiencia:  $W = 2K \cdot P \cdot \log P$

# 4. Análisis del rendimiento

## Estudios Experimentales

- Utilidad Estudios Experimentales
  - Etapas iniciales
    - Determinar parámetros de los modelos de rendimiento: tc, ts, tw, etc.
  - Después de iniciar implementación
    - Comparar rendimiento modelado y observado
- Diseño de experimentos
  - Identificar datos a obtener: tiempo de comunicación, tiempo total programa paralelo, etc.
  - **Determinar rango de datos:** Amplio rango de tamaños de problema y valores de P para minimizar el impacto de los errores individuales
  - **Objetivo: Resultados exactos y “reproducibles”.** Pueden existir variaciones en las medidas.
    - ✓ Normalizar tiempos de ejecución y realizar gran nº de medidas en algoritmos no deterministas
    - ✓ Usar temporizador adecuado y omitir costes de inicialización-terminación
    - ✓ Evitar interferencias con otros procesos: Compiten por los recursos (red de interconexión, CPU, memoria, ...).

# 4. Análisis del rendimiento

## Ajuste de datos experimentales a modelo

- **Propósito:** Determinar parámetros de modelo.
- Se ajustan resultados a función interés  $f$ 
  - Ej.:  $T_{msg}(L) = ts + tw L \rightarrow ts$  y  $tw$
- Ajuste de mínimos cuadrados simple:
  - Minimizar  $\sum_i (obs(i) - f(i))^2$
  - Menos peso a tiempos menores
- Ajuste de mínimos cuadrados escalado:
  - Min  $\sum_i ((obs(i) - f(i))/obs(i))^2$
- Procedimiento
  - 1. Realizar varias observaciones y obtener media, descartando extremos
  - 2. Ajuste mínimos cuadrados → Calcular params.

N	Observed	Performance Model	
		Simple	Scaled
2	0.476	0.480	0.448
4	1.74	1.82	1.78
8	6.64	7.68	7.16
16	26.7	30.7	28.7
32	112	123	115
64	450	491	450
128	1931	1966	1835
256	7806	7864	7340

# 4. Análisis del rendimiento

## Evaluación de implementaciones mediante modelo

- **Discrepancias entre rendimiento observado y modelado.** Causas
  - A) Modelo incorrecto: Utilizar resultados para determinar fallos modelo
  - B) Implementación inadecuada: Usar modelo para identificar mejoras
- **Omisión de sobrecargas en modelo:**  $T_{obs.} > T_{modelo}$ . Ejemplos:
  - Desequilibrios de carga
  - Herramientas ineficientes: Ej: Implementación ineficiente paralelismo virtual
- **Anomalías de la Ganancia:**  $T_{obs.} < T_{modelo}$ . Ganancia superlineal
  - Efectos por memoria caché: + procesadores → + memoria caché
    - $T_{comp}$  decremente debido a + aciertos de página de caché
  - Anomalías de búsqueda: Búsqueda paralela B&B → Descomposición del rango de búsqueda + estrategias de búsqueda
    - Proceso “afortunado” encuentra resultado temprano → reducción rango