Introducción

Metodología de la Programación Paralela

Jesús Sánchez Cuadrado (jesusc@um.es)

Curso 2020/21

Conceptos básicos

- Computación paralela
  - Uso de varios procesadores o nodos de computación para resolver un problema
  - Cada procesador trabaja en una porción del problema de manera independiente
  - Se coordinan a través de una memoria común o intercambiando datos
  - Requiere un computador paralelo

- Computador paralelo
  - Sistema capaz de ejecutar secuencias de instrucciones de manera independiente
  - Los procesadores actuales ya son paralelos (multicore)

Conceptos básicos

- Programación paralela
  - Técnicas y paradigmas de programación para aprovechar los recursos de un computador paralelo
  - Técnicas de programación secuencial pero requiere construcciones especiales para aprovechar el paralelismo

  - ¿Cómo conseguimos que un programa secuencial se ejecute de manera eficiente en un computador paralelo?

Necesidad de la programación paralela

- Ley de Moore

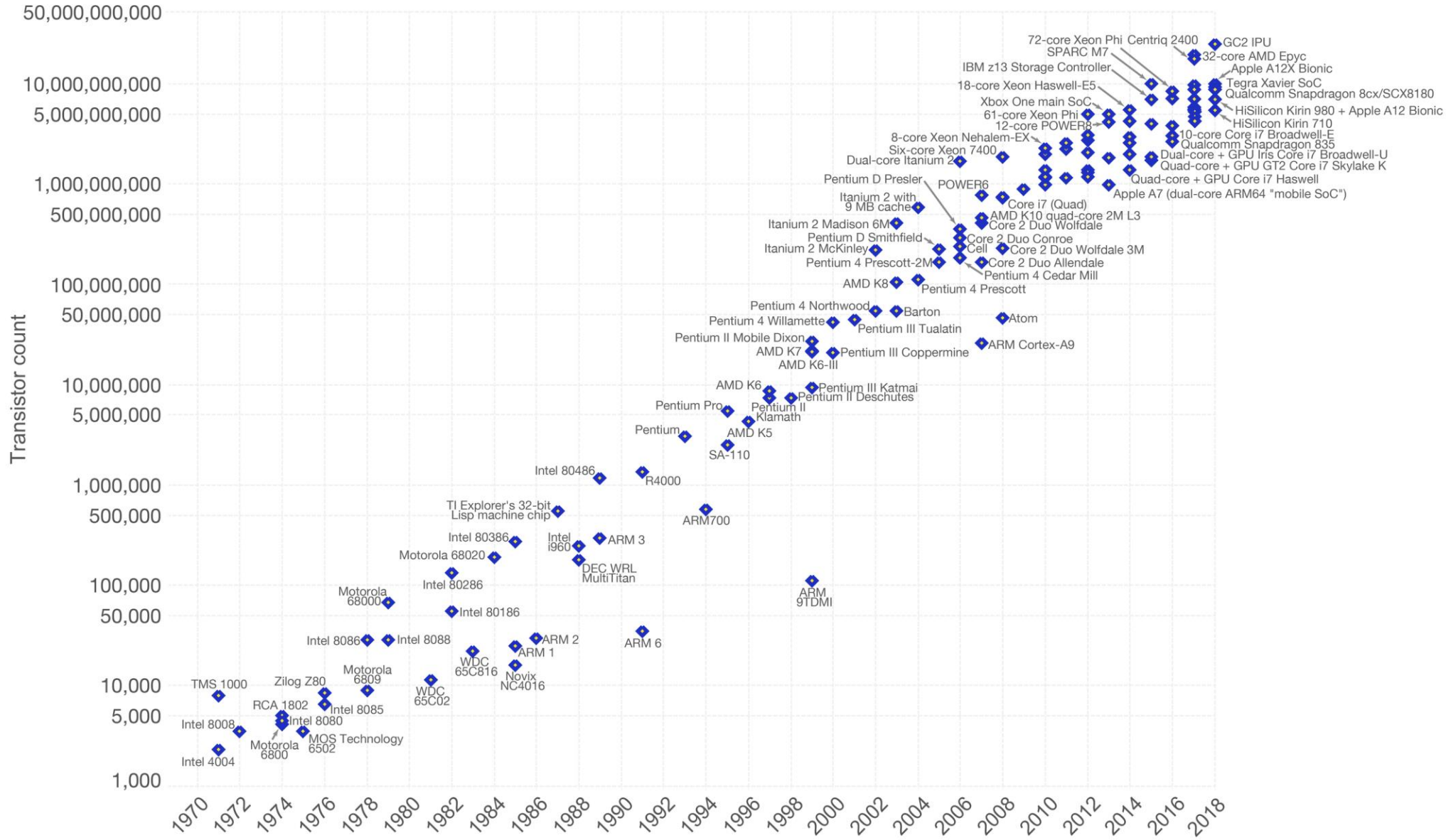
> "Cada dos años se duplica el número de transistores de un procesador"

  - Sin embargo, "the free launch is over"
  - The End of Moore's Law & Faster General Purpose Computing, and a New Golden Age, John Hennessy

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

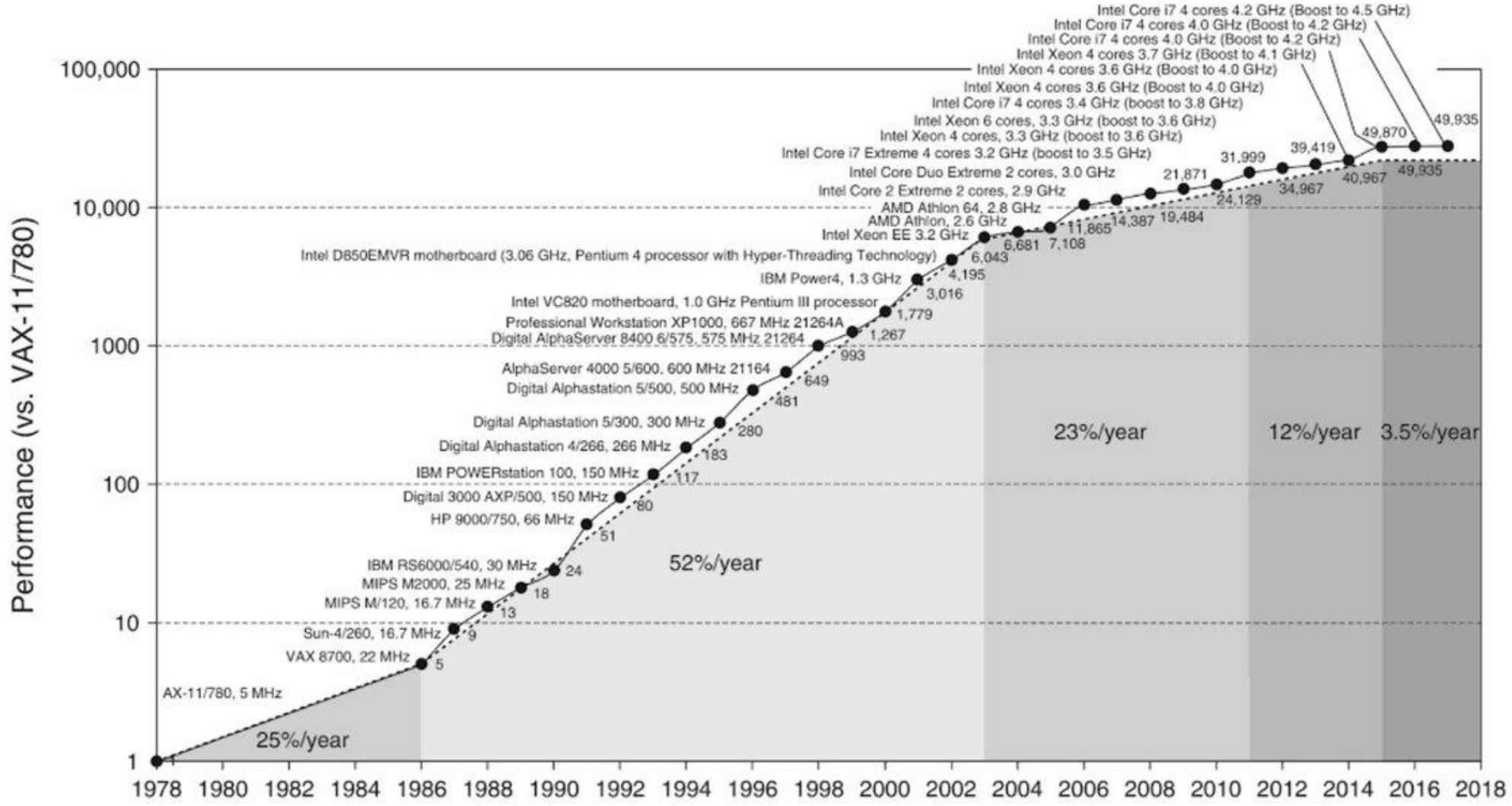
Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)

Necesidad de la programación paralela



Necesidad de la programación paralela

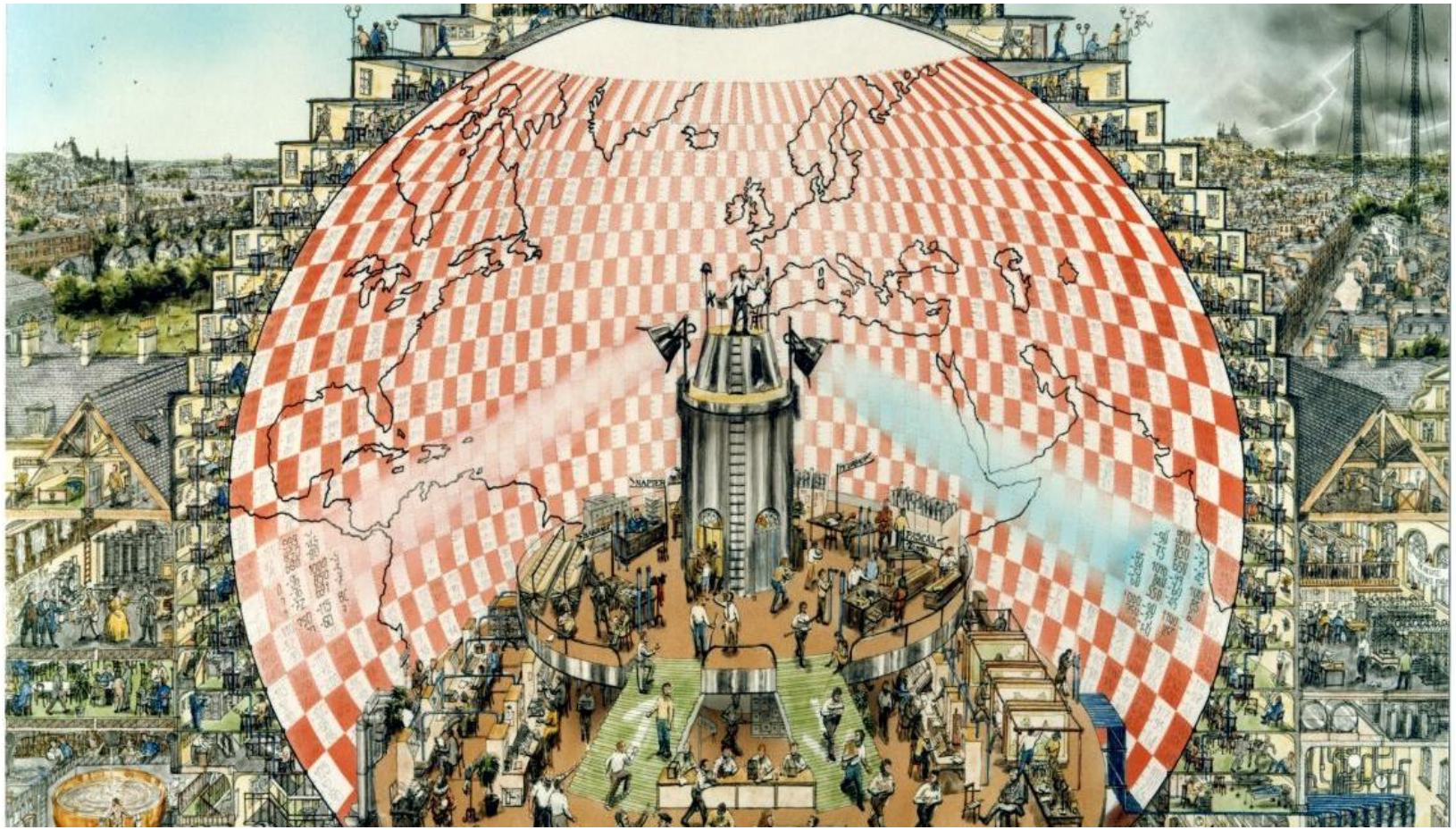
- Límites físicos de los computadores secuenciales
  - Ya no es posible mejorar sustancialmente un procesador
- Abordar problemas muy complejos
  - Paralelismo es la base del HPC
- Tratar grandes cantidades de datos
  - Big Data

Necesidad de la programación paralela

- Abordar problemas complejos
- HPC
  - Modelado del clima
  - Optimización
  - Análisis de datos
    - Colisionador de hadrones del CERN
- Otros ámbitos
  - Bases de datos
  - Compresión de ficheros
  - Recolectores de basura

Ejemplo – Modelado del clima

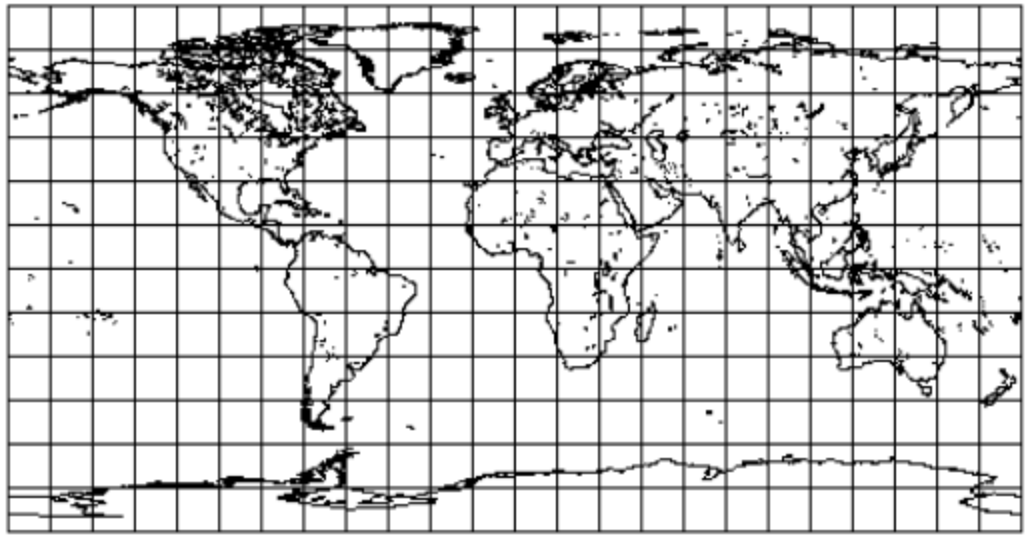
- Lewis Fry Richardson (1881 – 1953)
  - Primer esquema de modelado del clima*



* https://thatsmaths.com/2016/01/07/richardsons-fantastic-forecast-factory/

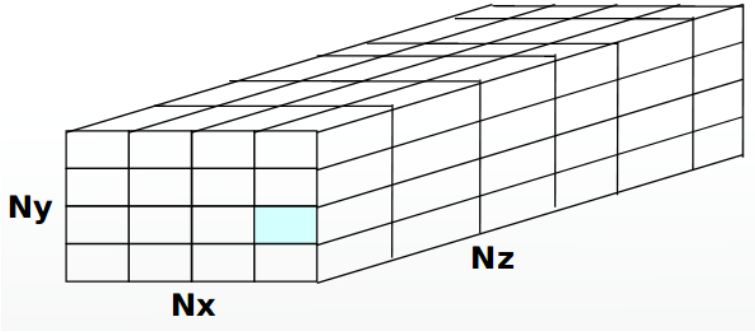
Ejemplo

- Modelado del clima
- Nx=Ny=3.000 km, Nz=11km
- Descomponer en 10^{11} cubos



- Ejecución en 1 core
  - 12 días
- Computador paralelo (256 cores)
  - 4 horas

- Resolución de problemas irresolubles
- Mallado más fino



Introducción

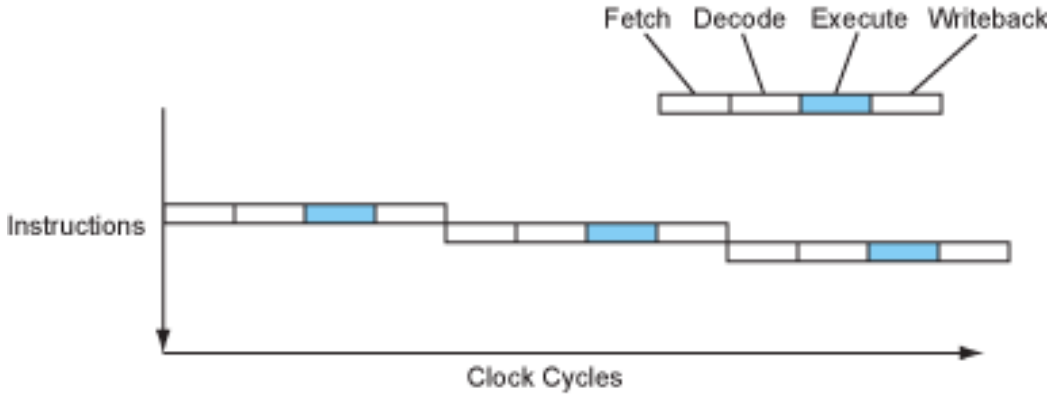
Paralelismo en sistemas secuenciales

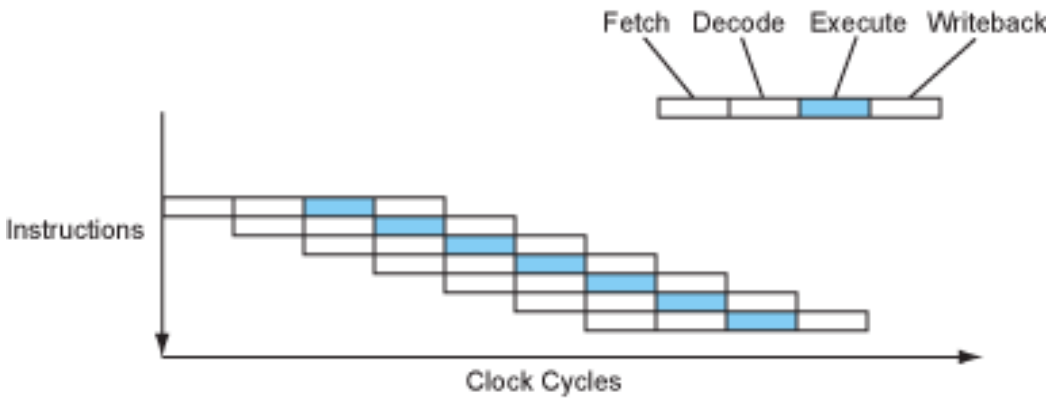
Paralelismo en sistemas secuenciales

- Modern Microprocessors: a 90 minutes guide!
  - http://www.lighterra.com/papers/modernmicroprocessors/

Paralelismo en sistemas secuenciales

- Pipelining y paralelismo a nivel de instrucción
- Secuencial
  - Una instrucción está dividida en etapas
  - Hay que esperar a que acabe la última etapa
  - CPI = 4
- Pipelining
  - Una instrucción puede empezar tan pronto como la anterior ha terminado una etapa
  - CPI = 1
  - Speedup = 4x

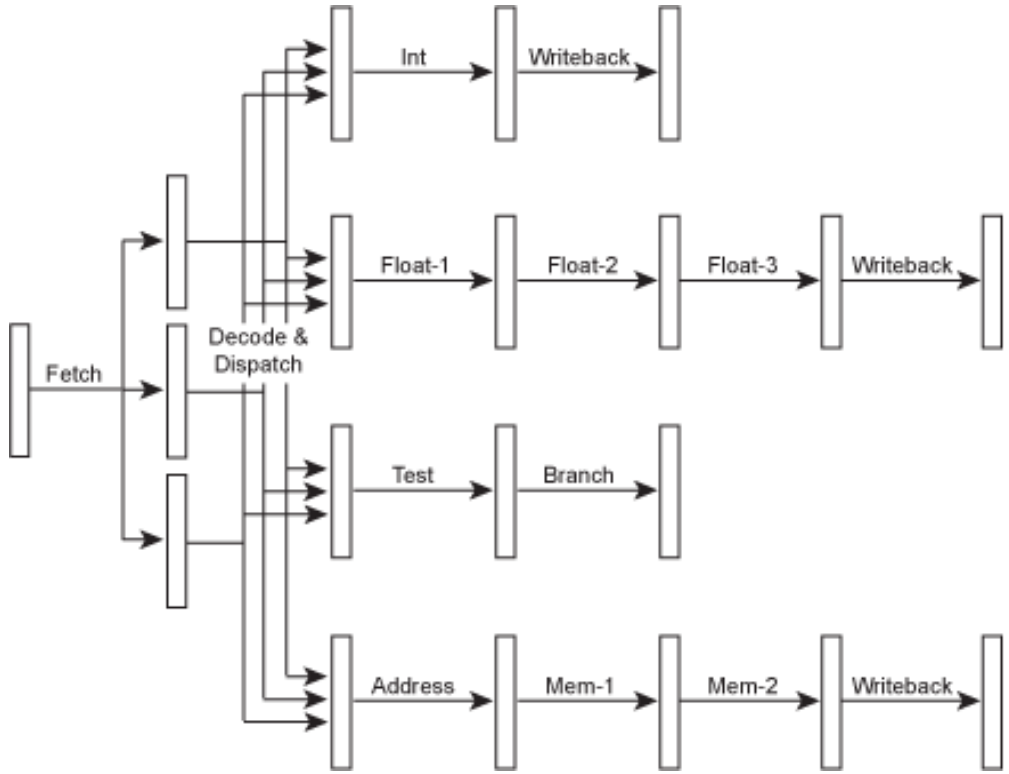


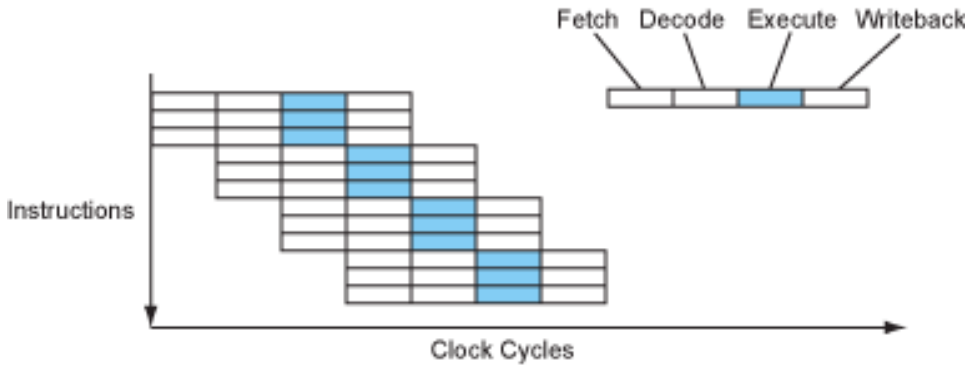


Paralelismo en sistemas secuenciales

- Superescalares – Multiple issue
  - Aprovechar los diferentes tipos de unidades funcionales
  - Decodificador de instrucciones en paralelo
  - Ejecución en paralelo a las unidades libres
  - Replicación de unidades funcionales para más rendimiento
  - Ejemplo, IPC = 3

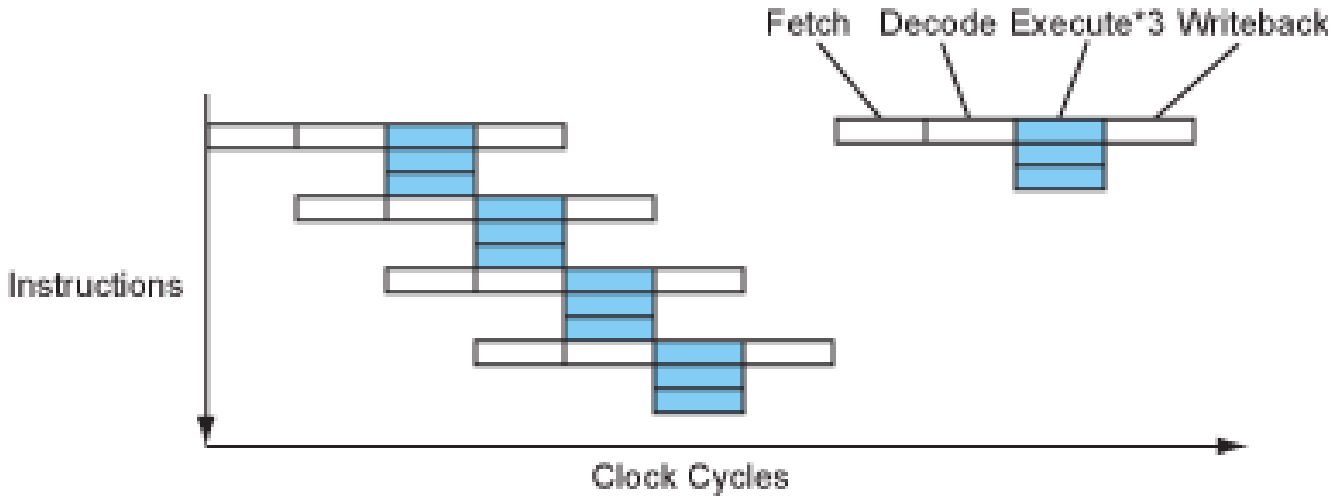
superpipelined + superscalar





Paralelismo en sistemas secuenciales

- Paralelismo explícito – Very Large Instruction Word (VLIW)
  - Diseño del juego de instrucciones para agrupar instrucciones que pueden ejecutarse en paralelo
  - Requiere soporte software especial (compilador)
    - Incluso para insertar no-ops para garantizar las dependencias de datos



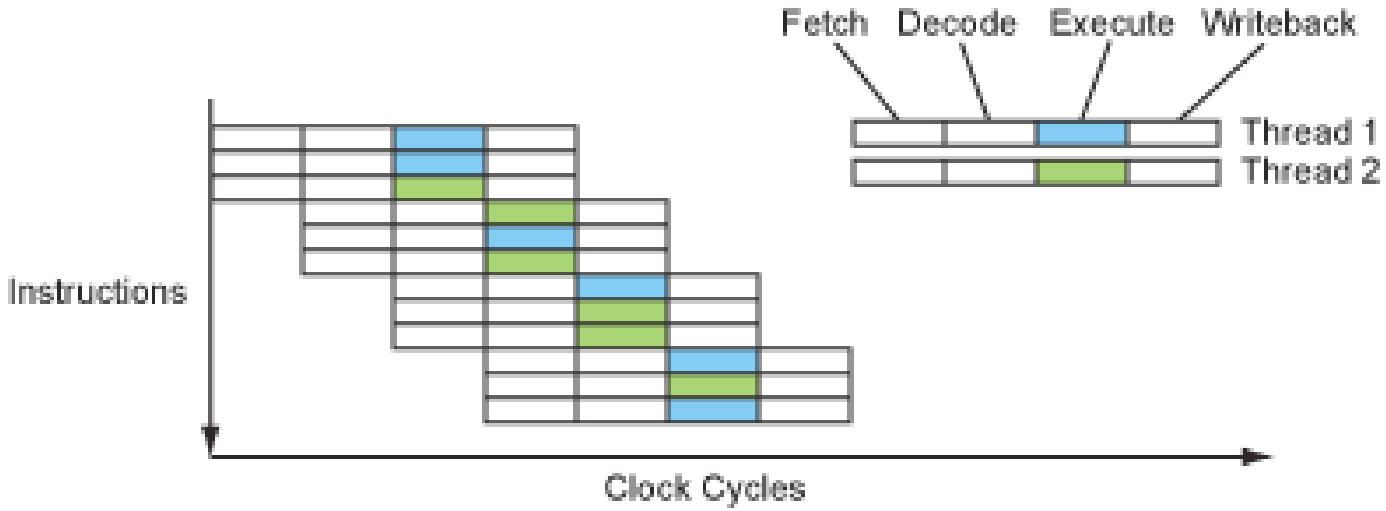
  - Intel Itanium
  - GPUs

Paralelismo en sistemas secuenciales

- Límites
  - Difícil conseguir paralelismo de más de 2 o 3 instrucciones por ciclo
  - No es posible aumentar el tamaño de los pipelines
    - No todas las instrucciones tardan lo mismo
    - Se quieren “by-passes”
  - Disipación de calor

Paralelismo en sistemas secuenciales

- Simultaneous multi-threading (SMT) – HyperThreading
- Busca otra fuente de instrucciones que se ejecuten de manera independiente
  - Otros programas, otros hilos dentro del mismo programa
- Un procesador SMT tiene varios cores lógicos
- Se duplican los elementos que guardan el estado de la ejecución de un hilo



Introducción

Sistemas paralelos

Sistemas paralelos

- **Multicore**
- Los sistemas multicore (multinúcleo) contienen varios cores que tienen acceso a un espacio de memoria común, organizado de forma jerárquica.
- En la actualidad son los sistemas computacionales estándar, y sistemas más complejos se obtienen combinando varios de ellos.
- Se programan comúnmente a través de hilos (threads), que comparten datos en la memoria común.

Sistemas paralelos

- **Tarjetas gráficas - GPU**
- Originariamente para vídeojuegos, en la actualidad también para procesamiento de propósito general (GPGPU).
- Normalmente como coprocesadores con un sistema multicore, que dispone de una o varias GPUs.
- El programa se ejecuta en CPU y manda trabajos a la GPU.
- Programación dependiente del fabricante (CUDA) y también hay software portable (OpenCL).
- Constan de muchos cores de GPU, organizados en bloques, y con organización jerárquica de la memoria.
- Varios tipos de tarjetas con distintas capacidades computacionales (Gforce, Tesla, Kepler...)

Sistemas paralelos

- Otros aceleradores
  - Los aceleradores computacionales o coprocesadores se utilizan normalmente mandándoles trabajo desde la CPU para que se encarguen de realizar cálculos computacionalmente costosos. Además de las GPUs hay algunos tipos más:
- Intel Xeon Phi:
  - Contiene hasta 61 cores, y cada core soporta 4 threads por hardware.
  - Programación más cercana que la de las GPUs a la estándar en paralelo.
- FPGA (Field Programmable Gate Array):
  - Similares a circuitos integrados pero reprogramables.
  - Programación distinta a la paralela estándar.
- DSP (Digital Signal Processor):
  - Para tratamiento de señales y problemas en tiempo real.
  - Normalmente para operaciones matemáticas que se repite continuamente.
- SBC (Single Board Computer):
  - Normalmente para sistemas embebidos, con bajo consumo de potencia.
  - Ejemplo: Raspberry Pi

Sistemas paralelos - Supercomputación

- La que se realiza en los computadores más potentes para resolver los problemas científicos con mayores necesidades computacionales (simulación climática, análisis de ADN...)
- Varía históricamente.
  - Lista Top500 contiene los 500 más rápidos del mundo, se actualiza cada seis meses.
  - https://www.top500.org/
- Problemas de gestión, sistemas de enfriamiento, reducción del consumo de energía (Green computing)...

Supercomputación

- Sunway TaihuLight en el National Super Computer Center en Wuxi, China
- Cluster de Intel Xeon + Xeon Phi
  - 10.649.600 cores
  - potencia 15.371 kw
  - rendimiento máximo 125.435,9 (93.014,6) Tflops/seg

HPC

- La Computación de Alto Rendimiento (o Altas Prestaciones, High Performance Computing) consiste en el aprovechamiento al máximo de las capacidades computacionales del sistema.
- Normalmente en supercomputadores, pero también en clusters, multicores, y para tiempo real.

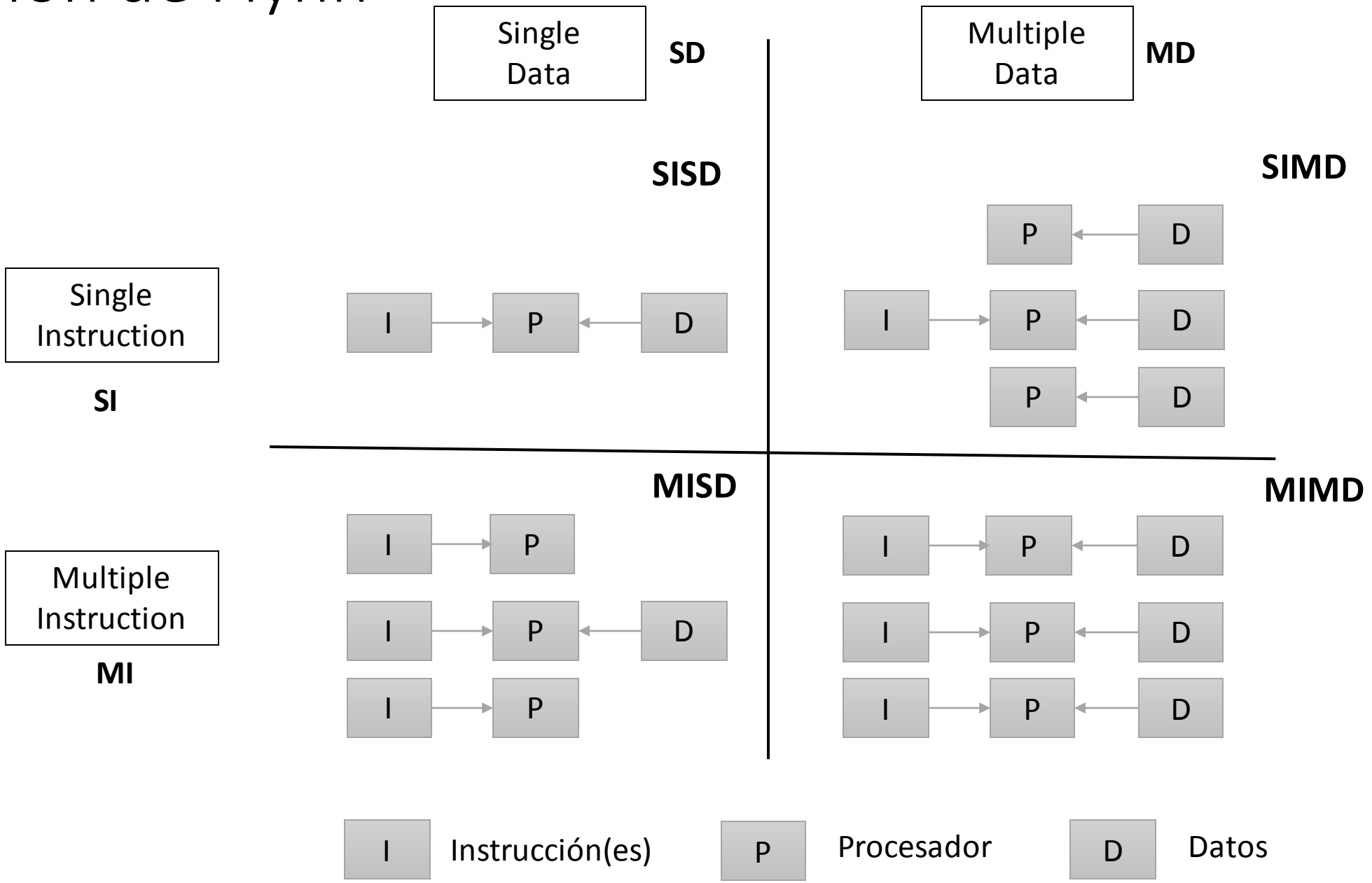
Sistemas paralelos

- **Computación heterogénea**
- Los componentes de un sistema computacional presentan distintas fuentes de heterogeneidad:
  - Memorias primarias y secundarias de distinta capacidad y con distinta organización.
  - Componentes computacionales con distinta velocidad.
  - Componentes computacionales de distinta arquitectura (ejemplo, multicore+GPU).
  - Redes de conexión a distinta velocidad.
- Hay que programarlos de forma especial para aprovechar al máximo la capacidad de todos ellos, quizás con programación paralela estándar pero asignación balanceada de trabajos a los distintos componentes.

Introducción

Arquitecturas paralelas

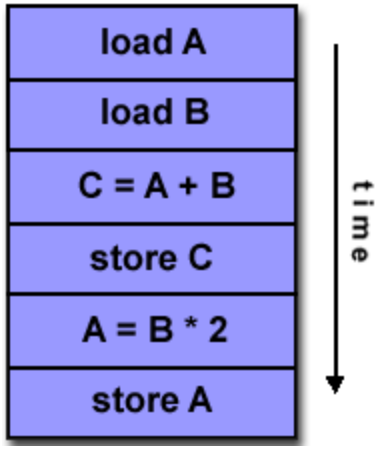
Clasificación de Flynn



Clasificación de Flynn

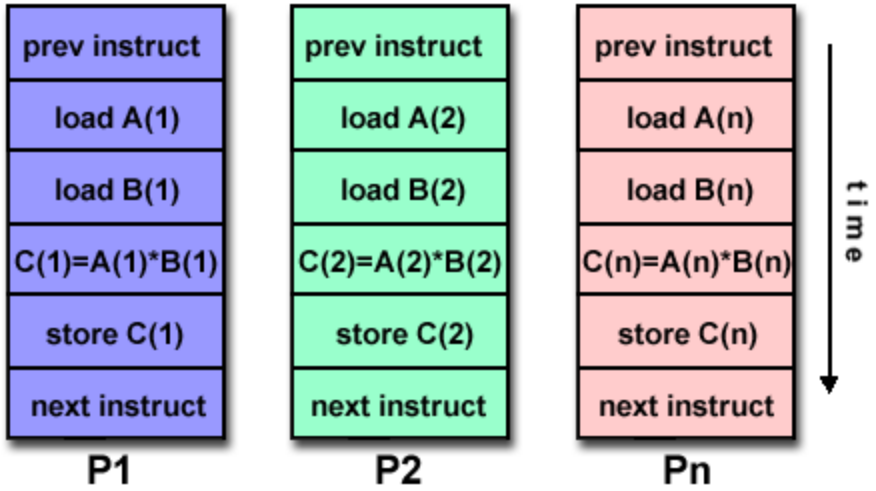
- **SISD: Single Instruction, Single Data**

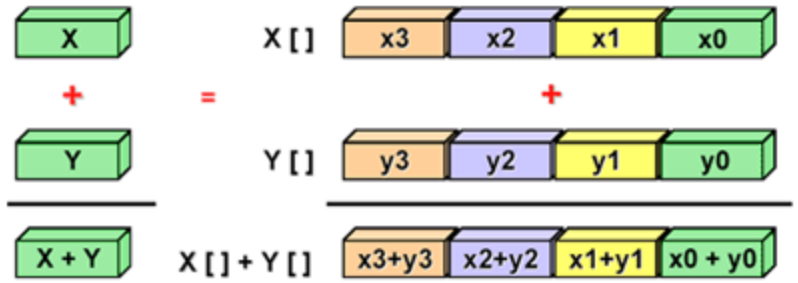
- Esencialmente el modelo Von Neuman
  - Computadores secuenciales
- Se procesa una instrucción por ciclo de reloj
- Por cada instrucción un único flujo de datos
- La ejecución es determinista



Clasificación de Flynn

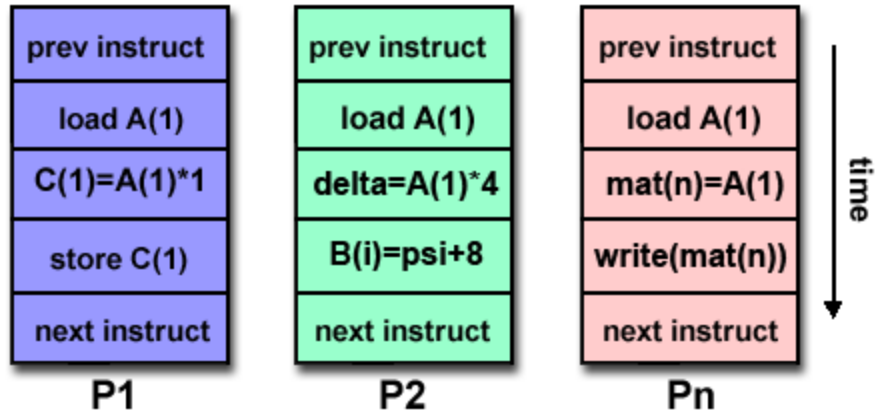
- **SIMD: Single Instruction, Multiple Data**
- **SI:** Todas las unidades ejecutan la misma instrucción por ciclo de reloj
- **MD:** Cada unidad de procesamiento opera sobre un dato diferente
- Programas con gran regularidad (procesamiento de imágenes)





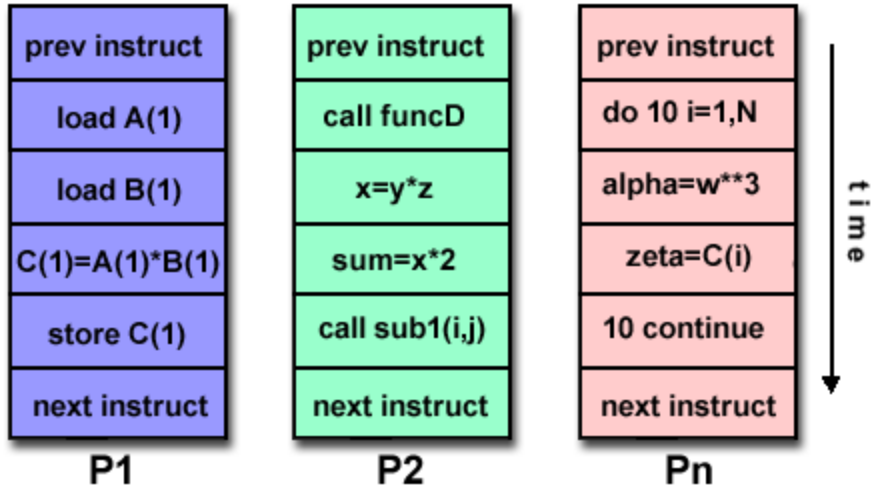
Clasificación de Flynn

- **MISD: Multiple Instruction, Single Data**
- **MI:** Cada unidad de procesamiento tiene como entrada un flujo diferente de procesamiento
- **SI:** El mismo flujo de datos se usa como entrada para todas las unidades de procesamiento
- Pocos ejemplos de esta clase
  - Varios algoritmos criptográficos tratando de descifrar el mismo mensaje



Clasificación de Flynn

- **MIMD: Multiple Instruction, Multiple Data**
- **MI:** Cada procesador puede ejecutar un flujo de instrucciones diferente
- **SI:** Cada procesador puede trabajar con datos diferentes
- Es el tipo más común de computador paralelo
  - Supercomputers (HPC), networked parallel computer "grids" y procesadores multi-core.

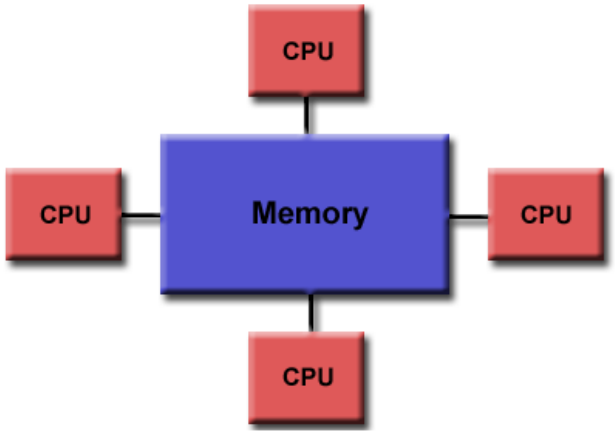


Paradigmas de programación

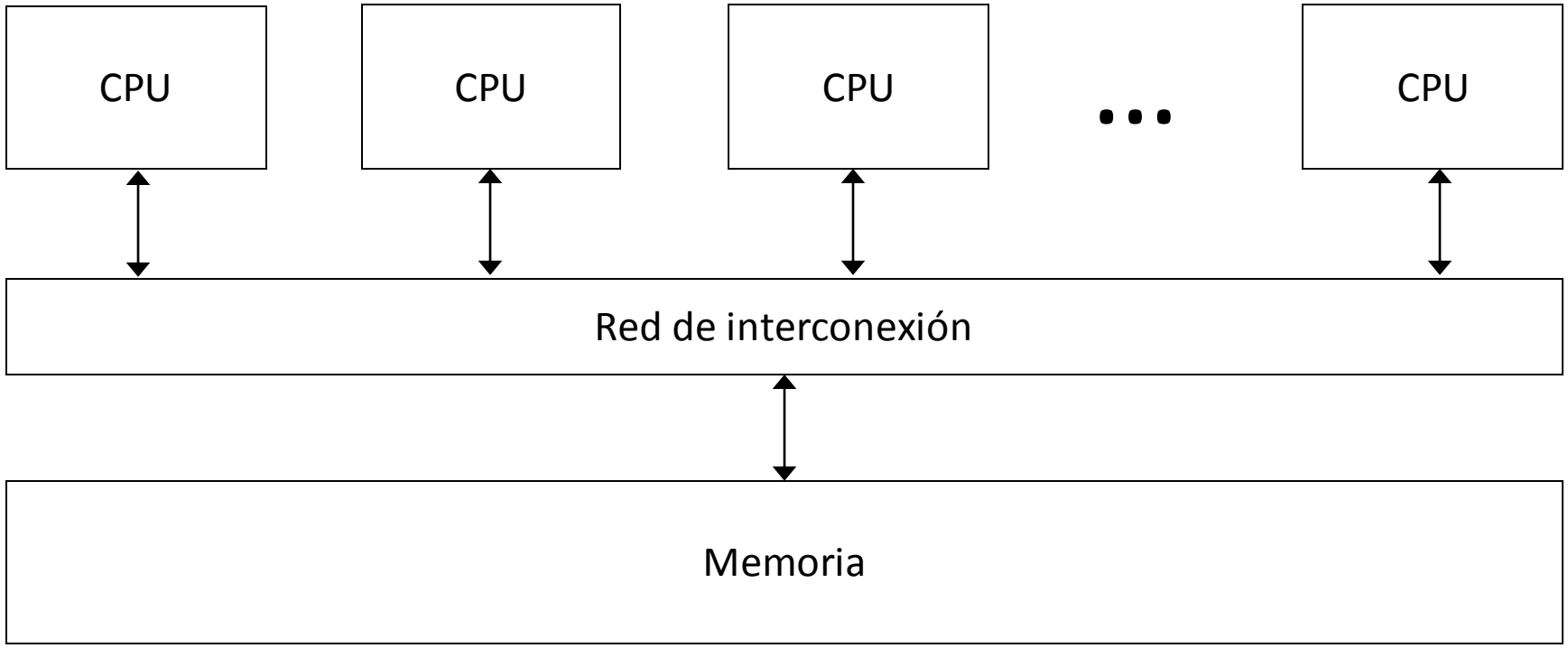
- Programación con memoria compartida 
- Programación con paso de mensajes 
- Programación para procesadores SIMD
  - CUDA
  - Máster NTI

Sistemas con memoria compartida

- Todos los procesadores tienen acceso a una memoria común (espacio de direcciones común)
  - Los procesos o hilos de un programa paralelo “ven” la misma memoria
  - Los cambios son visibles para todos los procesadores
- Comunicación entre los procesos: escribir y leer datos en memoria



Sistemas con memoria compartida

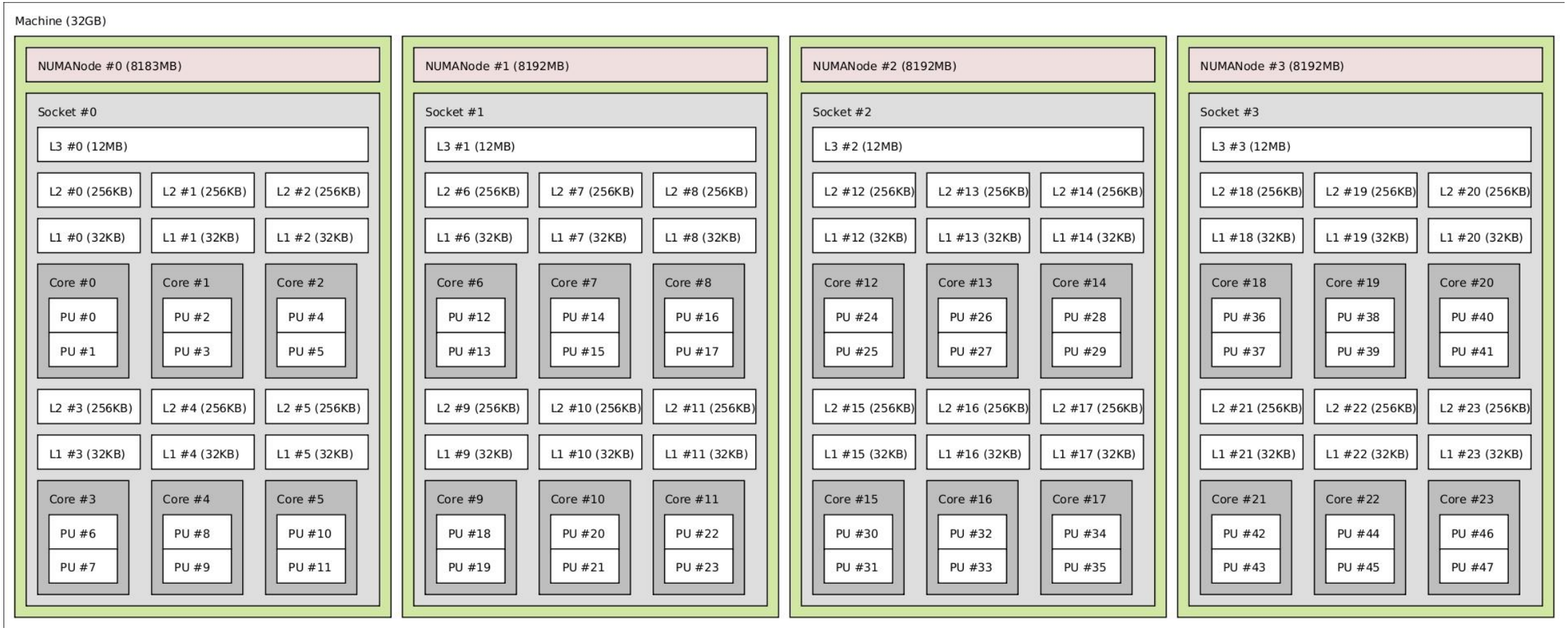


Sistemas con memoria compartida

- Multiprocesadores con memoria compartida
- UMA (Uniform Memory Access)
  - Todos los procesadores mismo tiempo de acceso a memoria
- NUMA (Non-Uniform Memoria Access)
  - Cada procesador tiene su memoria local
  - Un procesador puede acceder a la memoria de otro procesador
  - Tiempos de acceso diferentes
  - Fácil de extender
  - Requiere mecanismos especiales de coherencia (SO)

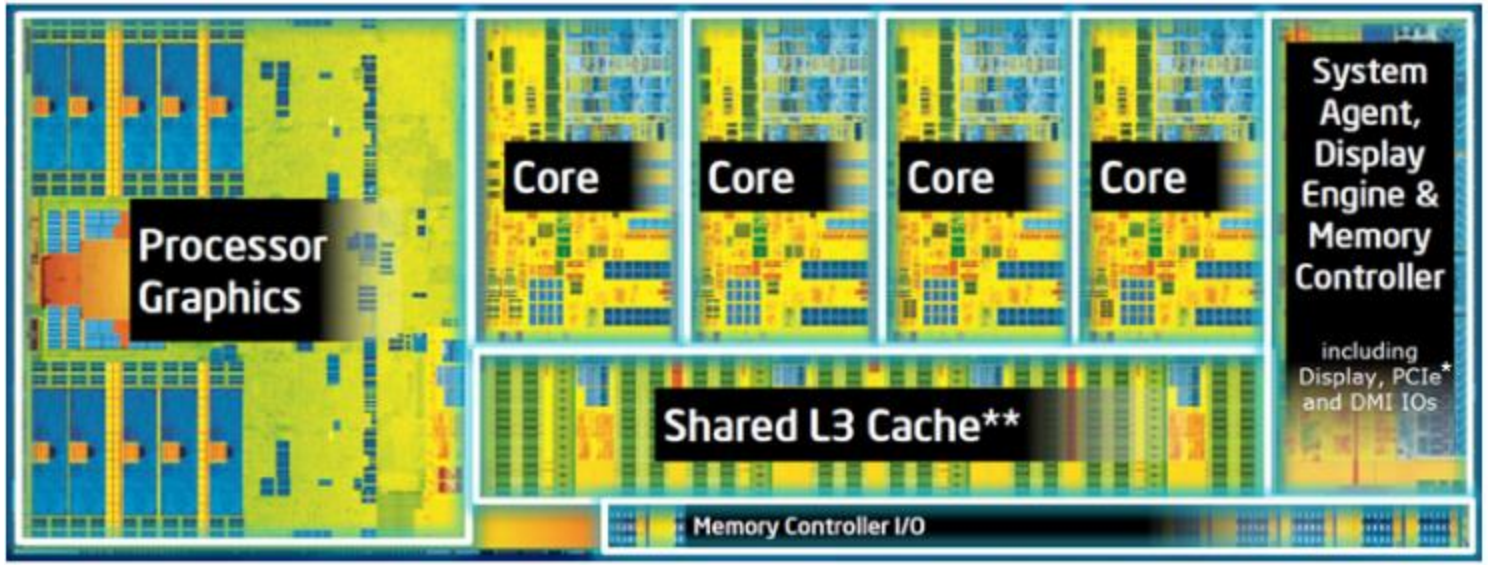
Ejemplo

Imagen de **Saturno** con `hwloc`



Sistemas con memoria compartida

- **Procesadores multinúcleo (multicore)**
- Cada procesador tiene dos o más núcleos
  - Es un multiprocesador en un solo chip
  - El SO percibe cada núcleo como un procesador diferente
- Ejemplos
  - Intel i5/i7

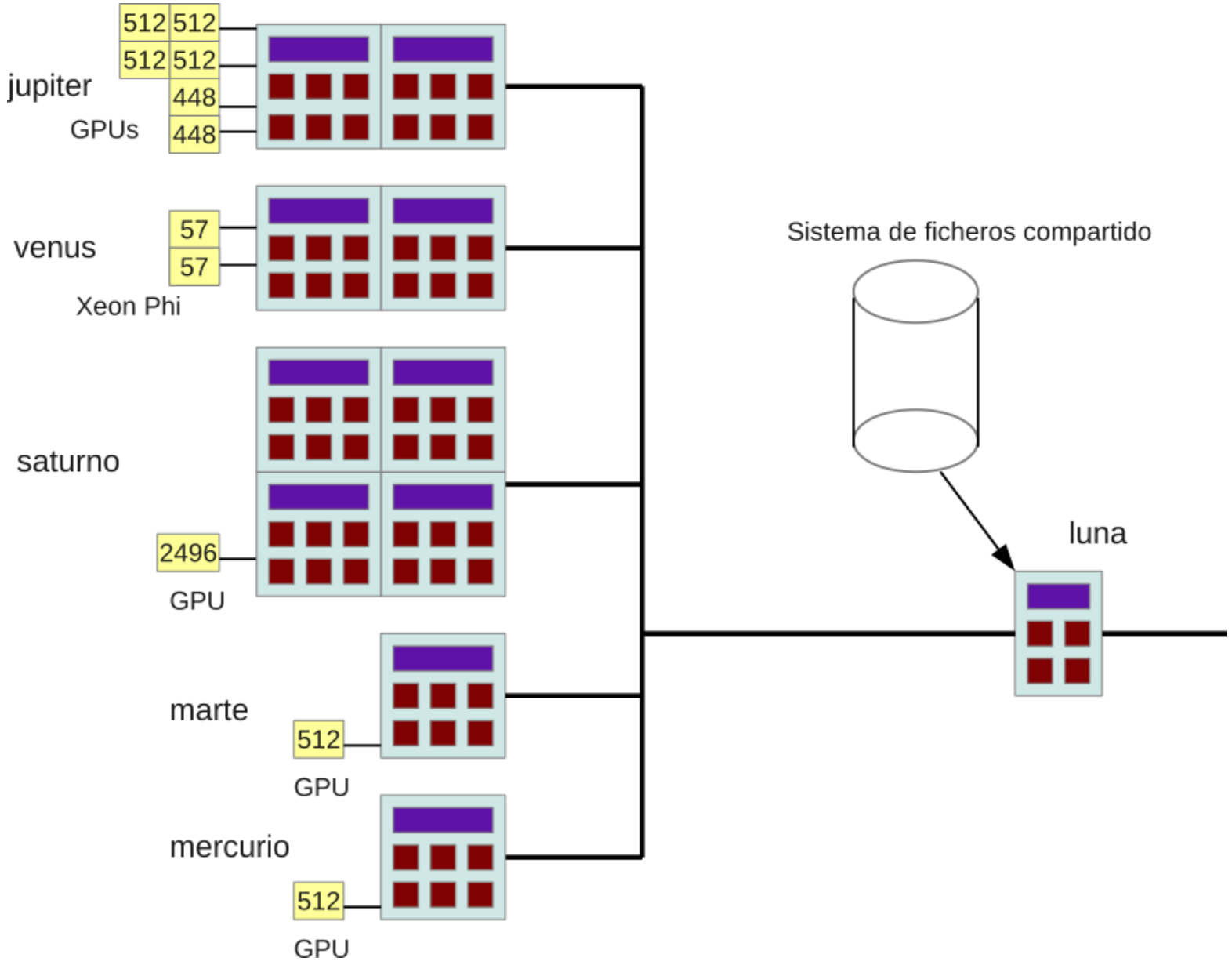


Sistemas de memoria distribuida

- Conexión de ordenadores mediante red de interconexión
  - Cada ordenador su memoria local y espacio de direcciones propio
  - Envío y recepción de mensajes a través de la red

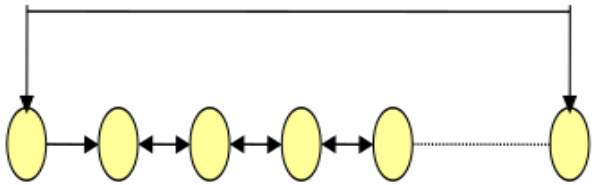
- Programación
  - Paradigma diferente al habitual (paso de mensajes)
    - ¿Es más complicado? No está claro…
  - No hacen falta mecanismos de exclusión mutua
  - Aplicable a redes de ordenadores => mayor adaptación
- Ingeniero de hardware
  - Más escalabilidad que los mc. de memoria compartida

Ejemplo


512 512
512 512
448
448
jupiter
GPUs
57
57
venus
Xeon Phi
saturno
2496
GPU
marte
512
GPU
mercurio
512
GPU
Sistema de ficheros compartido
luna

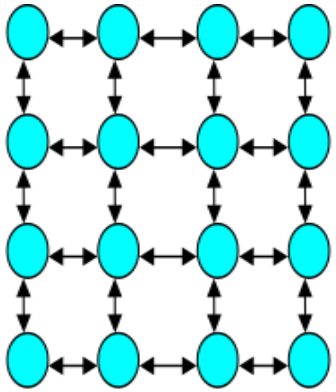
Sistemas de memoria distribuida

- Redes de interconexión



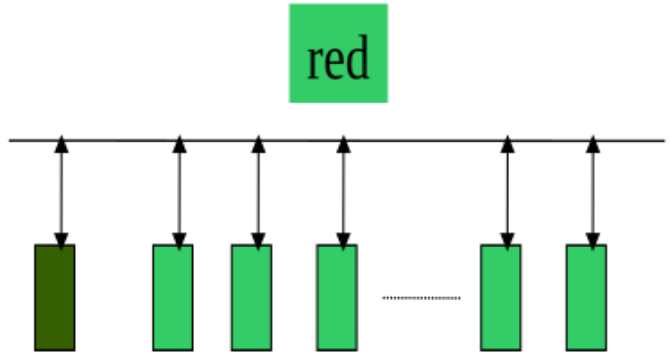
anillo

Diámetro: p/2



Malla

Diámetro: √p



Hipercubo

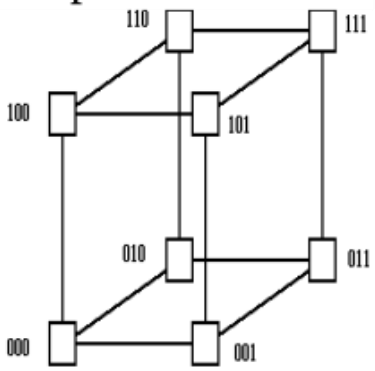


Figure 1.10 Three-dimensional hypercube

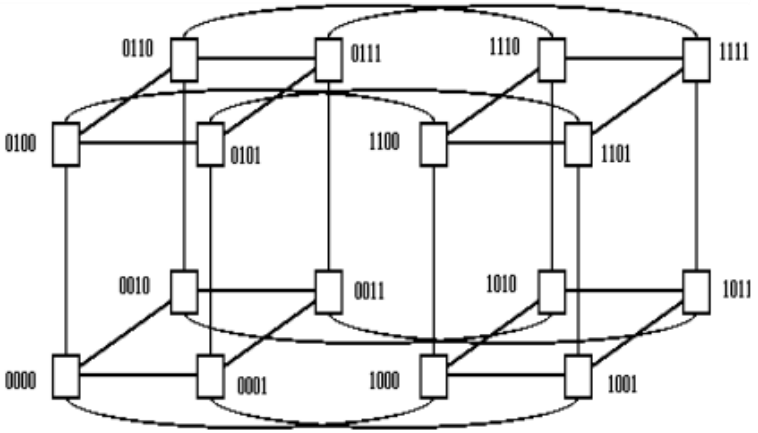


Figure 1.11 Four-dimensional hypercube

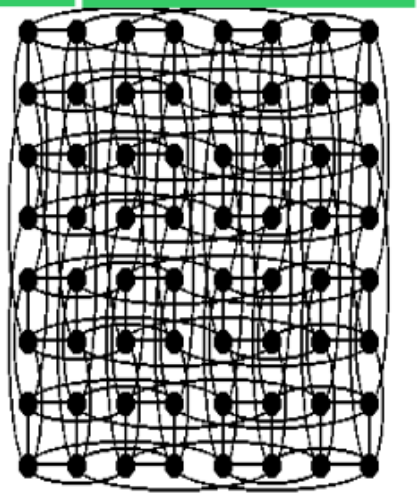


Figure 1.12 Six-dimensional hypercube laid out in one plane

Sistemas de memoria distribuida

- **Clusters de ordenadores**
- Ordenadores convencionales conectados a través de red de interconexión de alta velocidad
- El conjunto se ve como un único ordenador

- Ventajas sobre el multiprocesador
  - Disponibilidad a bajo costo de PCs potentes
  - Fácil incorporación de nuevas CPUs
  - No requiere software especializado de un fabricante

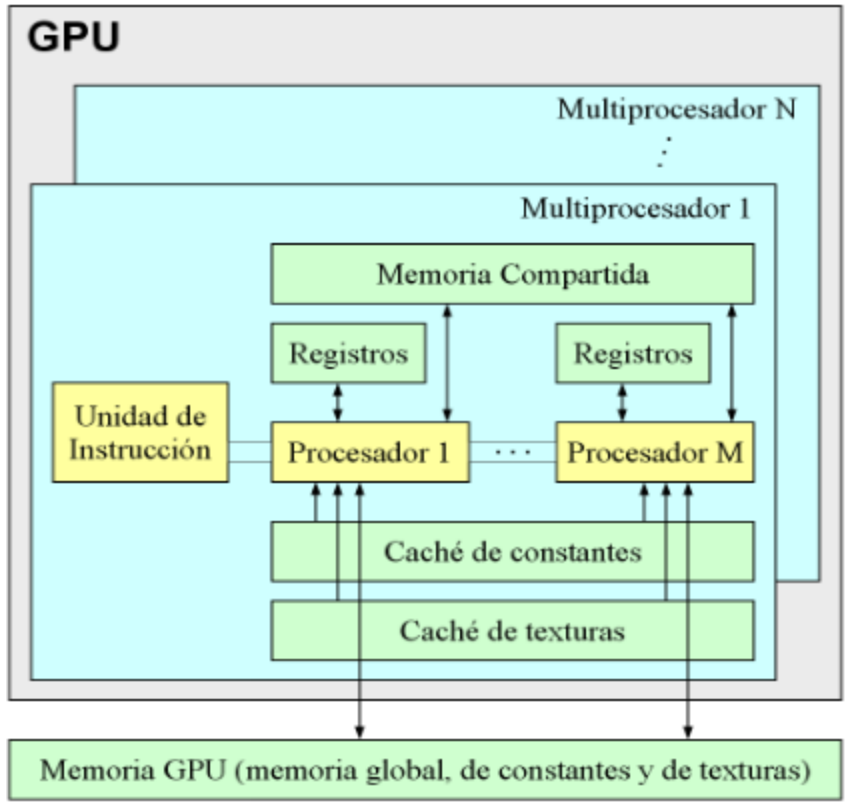
- Inconvenientes
  - Cada vez menos

Memoria Compartida

- **Ventajas**
  - Facilidad de programación con respecto a los accesos a memoria
  - Compartir datos entre tareas es rápido y uniforme
- **Desventajas**
  - Problemas de escalibilidad entre la memoria y las CPUs. Al añadir más CPU se incrementa geométrica el tráfico entre memoria y CPUs
  - Necesidad de usar mecanismos de sincronización para asegurar el correcto acceso a la memoria global (mutex, semáforos, etc.)

Procesadores gráficos – GPUs

- La GPU contiene N multi-procesadores
  - › Cada multi-procesador
    - M procesadores
    - Banco de registros
    - Memoria compartida, rápida y pequeña
    - Cachés de sólo lectura (texturas)
  - › Memoria global 500 veces más lenta que la memoria compartida



Introducción

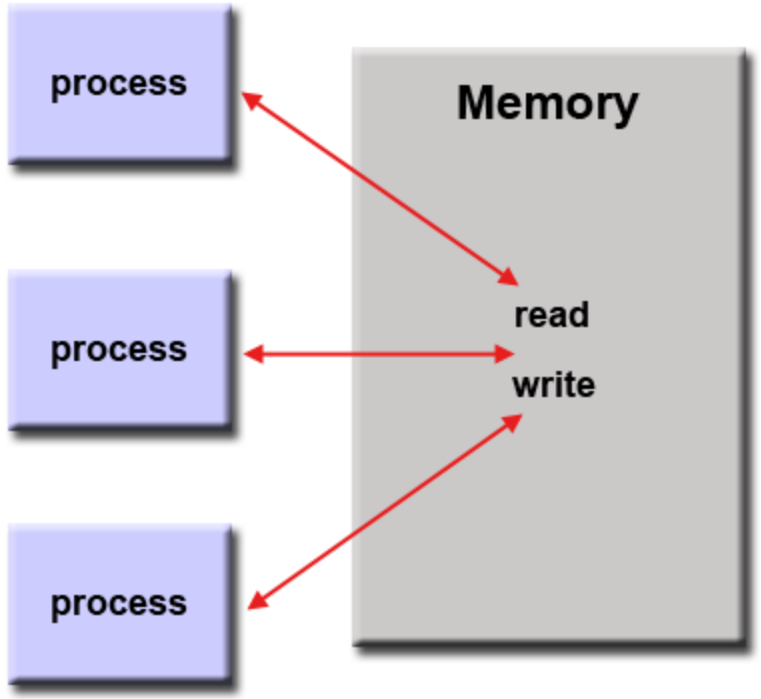
Modelos de programación paralela

Modelos de programación paralela

- Existen varios modelos
  - IPC (Inter-Process Communication)
  - Hilos
  - Paso de mensajes
  - Híbrido
  - Data Parallel – Modelo SIMD
- Un modelo de programación es una abstracción sobre las arquitecturas hardware y de memoria.
  - Programación con hilos sobre un computador basado en paso de mensajes
  - Paso de mensajes en una arquitectura de memoria compartida

IPC

- Procesos comparten la memoria
- Modo habitual antes de la programación multi-hilo
- Ejemplo,
  - Procesos POSIX
  - Funciones para crear procesos (ej., fork)
  - Funciones para reservar segmentos comunes de memoria (ej., shmget)



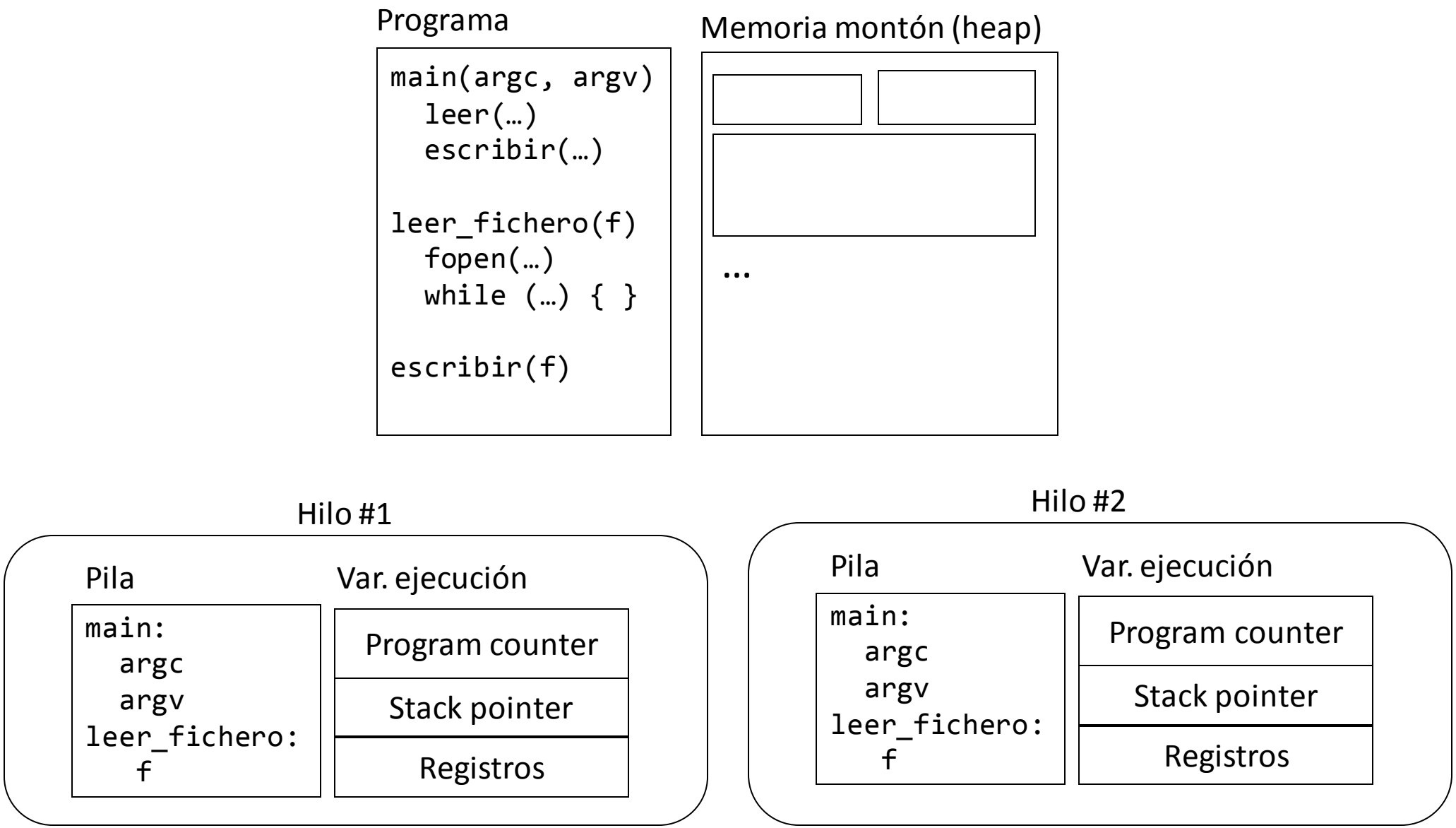
Hilos (Threads)

- Un proceso un hilo al inicio
  - Puede crear más hilos adicionales
  - Crear un hilo es más ligero que un proceso

- Un hilo tiene, su propio:
  - Contador de programa
  - Pila para variables locales
  - Identificador del hilo

- Un hilo comparte:
  - Instrucciones del proceso
  - Memoria
  - Descriptores de ficheros, PID, etc.

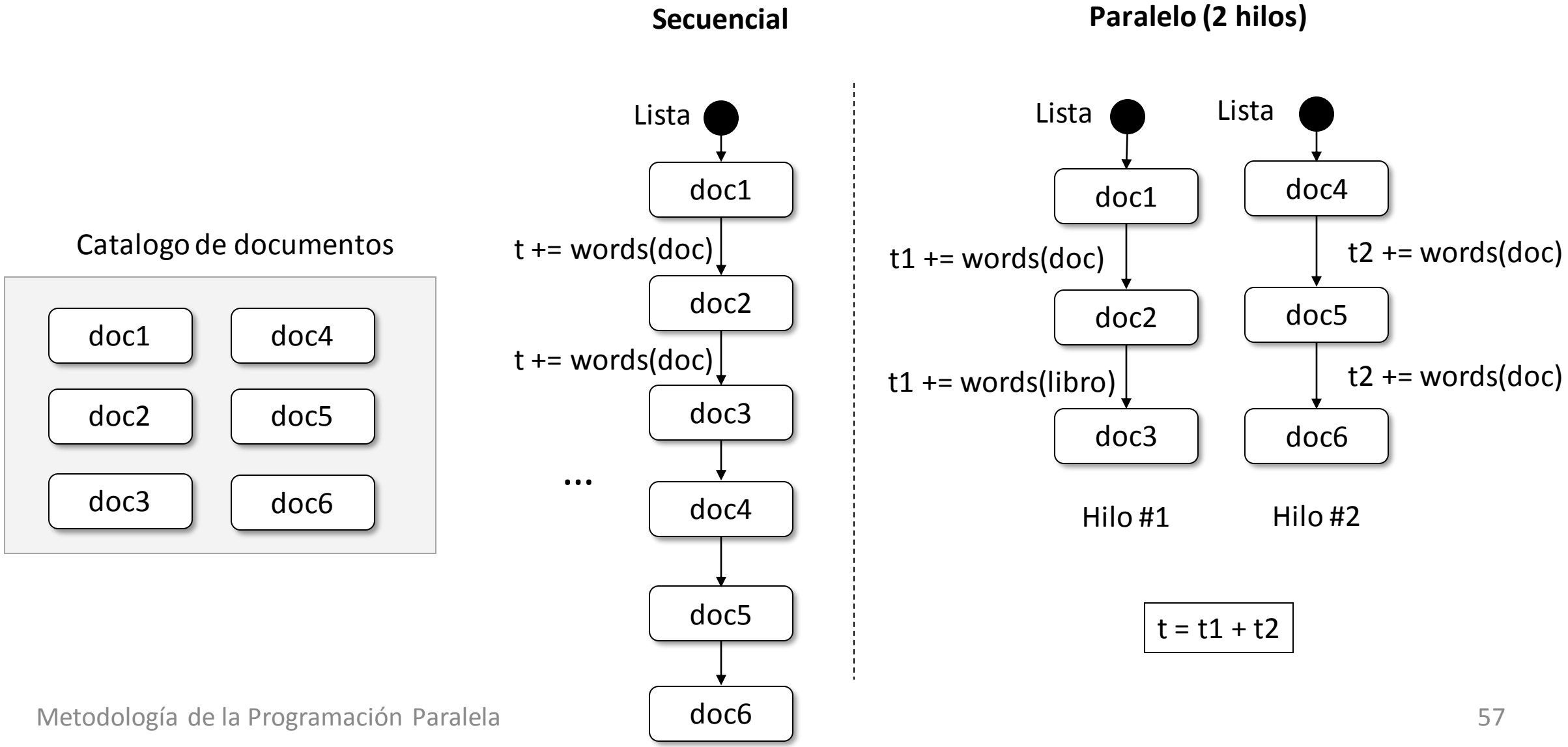
Hilos (Threads)



Hilos (Threads)

- Ejemplo en Java
- Contar palabras de una cantidad grande ficheros

Ejemplo de programación multi-hilo en Java



Hilos (Threads)

- Java
  - synchronized y métodos en la clase Object
  - Librerías de concurrencia (e.g., ReentrantLock)
- Pthreads
  - C/C++
  - Interface de programación (API) para trabajo con hilos.
- OpenMP
  - Especificación de API para programación paralela en Memoria Compartida. Se puede considerar el estándar para computación en memoria compartida.
  - Se encuentra en implementaciones de lenguajes, como gcc

Paso de mensajes

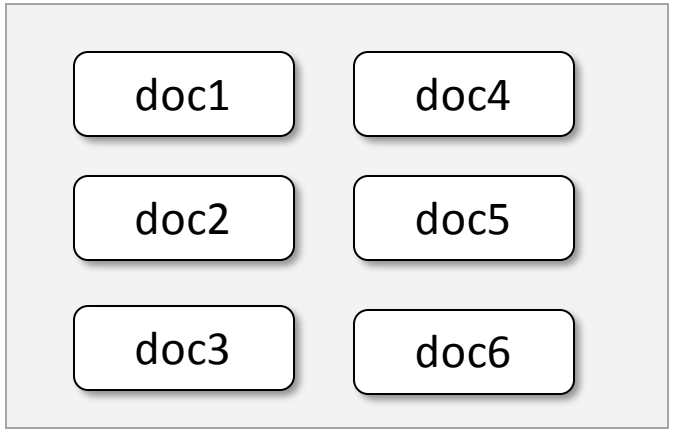
- Tipo de programa
  - Puede haber un único programa y ponerse en marcha varios procesos con el mismo código. Modelo “Simple Programa Múltiple Dato (SPMD).” (en MPI)
  - Aunque sea el mismo programa los códigos que se ejecutan pueden ser distintos si se compila para arquitecturas distintas.
  - Puede haber varios programas y generarse procesos con códigos distintos.
- Generación de procesos:
  - Generación estática: todos los procesos se ponen en marcha al mismo tiempo.
  - Generación dinámica: unos procesos ponen en marcha otros durante la ejecución.

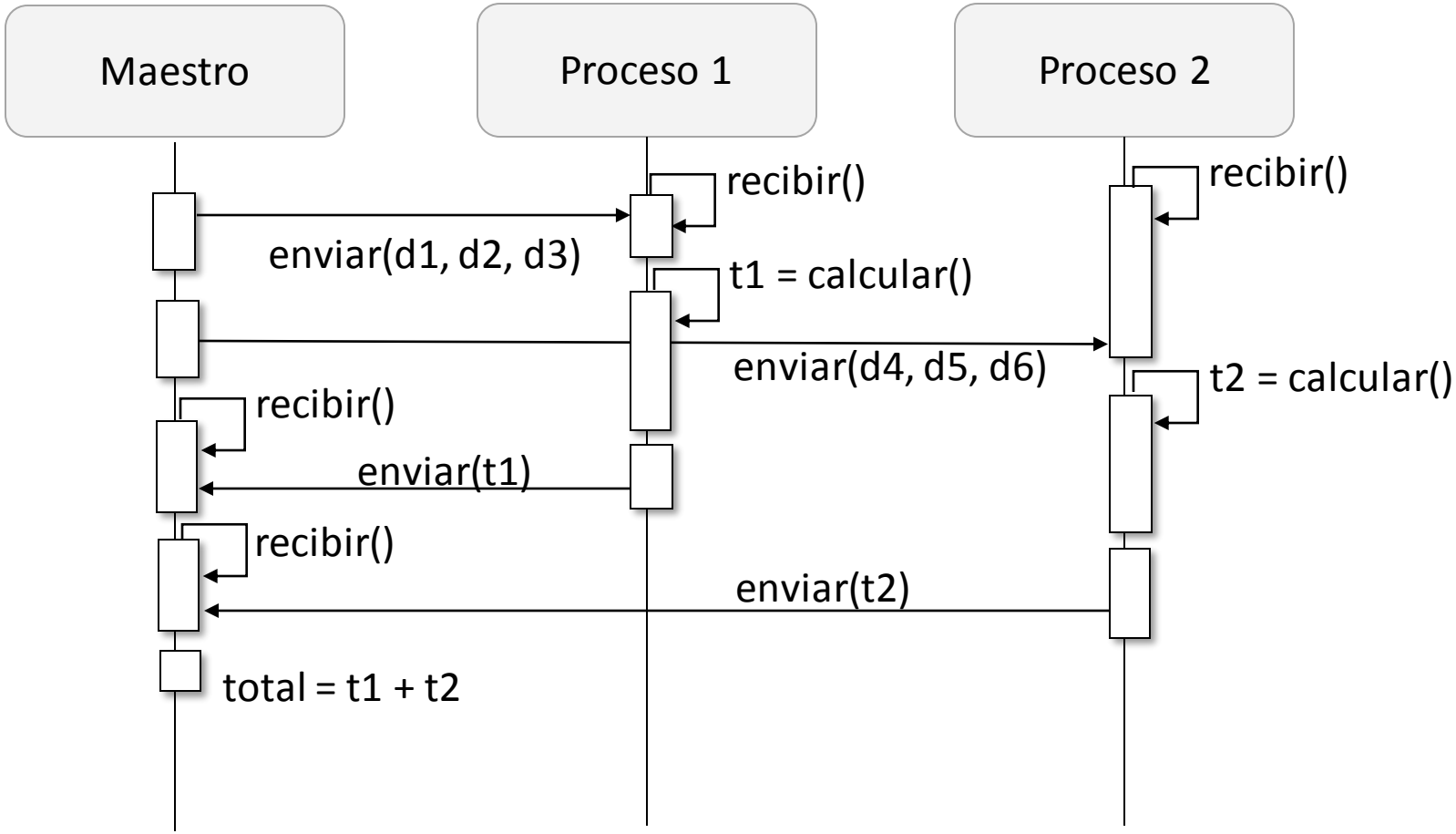
Paso de mensajes

- Los procesos se comunican con mensajes, que pueden ser:
- Según el número de procesos:
  - Punto a punto: un proceso envía y otro recibe.
  - Globales: intervienen varios procesos, posiblemente uno enviando o recibiendo datos de todos los demás.
- Según la sincronización:
  - Síncronos: los procesos que intervienen se bloquean hasta que se realiza la comunicación.
  - Asíncronos: los procesos no se bloquean. El que envía manda los datos y sigue trabajando, el que recibe, si no están disponibles los datos continúa con su trabajo.

Ejemplo

Catalogo de documentos



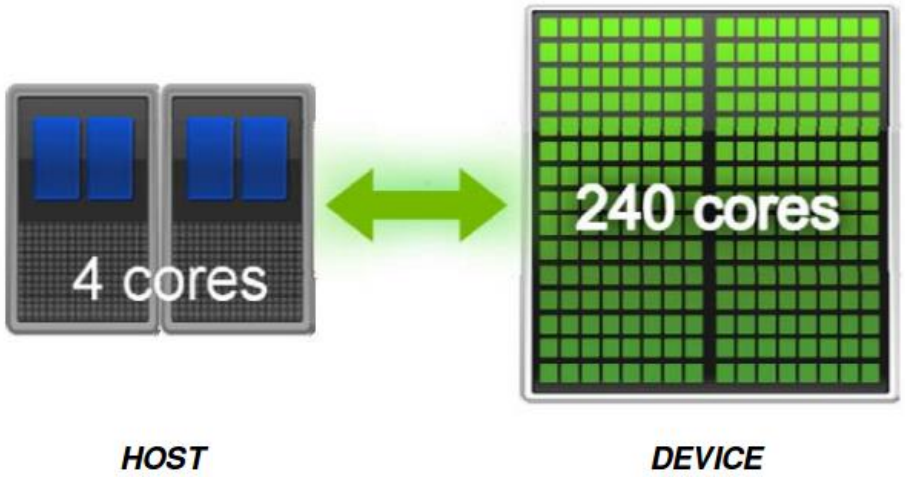


Paso de mensajes

- Java
  - Requiere librerías específicas de paso de mensajes
  - Modelo de actores – Akka
- MPI (Message Passing Interface):
  - Especificación de API para programación con Paso de Mensajes. Se puede considerar el estándar para computación en sistemas distribuidos.
  - API para varios lenguajes: C/C++, Fortran...
  - Varias implementaciones gratuitas: MPICH, LAMMPI, OpenMPI

Modelo SIMD - GPUs

- Paralelismo de datos
  - El mismo programa para procesar muchos datos al mismo tiempo
- Parte secuencial en la GPU
- Se delega el trabajo costoso de cálculo a la GPU
- Jerarquía de memoria
  - Minimizar acceso a memoria principal o solapar con computación



Modelo SIMD - GPUs

- CUDA
  - Arquitectura de cálculo paralelo de propósito general (Nvidia)
  - OpenCL estándar para otros fabricantes
- Modelo de programación
  - Programa secuencial (ej., en C para CUDA)
  - La parte paralela es una función (el *kernel*)
  - El kernel se ejecuta de forma paralela como hilos
  - Descomposición del problema como un grid (1D o 2D)
  - Cada hilo usa su *id* para saber qué trabajo tiene que hacer

Bibliografía

- Diapositivas basadas en:
  - Curso de Domingo Giménez Cánovas
    - http://dis.um.es/~domingo/mpp.html
  - Curso de José Miguel Mantas
    - https://lsi.ugr.es/jmantas/ppr/teoria/teoria.php
  - Modern Microprocessors: a 90 minutes guide!
    - http://www.lighterra.com/papers/modernmicroprocessors/
  - High Performance Computing is More Parallel than Ever
    - https://medium.com/tebs-lab/the-age-of-parallel-computing-b3f4319c97b0
  - Tutorial de programación paralela
    - https://computing.llnl.gov/tutorials/parallel_comp/

Ejercicicios

- Busca y describe aplicaciones de la programación paralela. Por ejemplo, buscando en GitHub palabras clave como OpenMP, threads, parallel, etc.
  - Identifica alguna de ellas.
  - ¿Qué estrategia sigue?
  - ¿Qué aporta la programación paralela?
- Investiga cómo ejecutar programas en paralelo con el Shell
  - Comandos parallel y slepp
  - Modificador &

Bibliografía

- Diapositivas basadas en:
  - Curso de Domingo Giménez Cánovas
    - http://dis.um.es/~domingo/mpp.html
  - Curso de José Miguel Mantas
    - https://lsi.ugr.es/jmantas/ppr/teoria/teoria.php
  - Modern Microprocessors: a 90 minutes guide!
    - http://www.lighterra.com/papers/modernmicroprocessors/
  - High Performance Computing is More Parallel than Ever
    - https://medium.com/tebs-lab/the-age-of-parallel-computing-b3f4319c97b0
  - Tutorial de programación paralela
    - https://computing.llnl.gov/tutorials/parallel_comp/