Tema 1: Evolución de la arquitectura de los sistemas paralelos y de sus modelos de programación

Introducción a la computación paralela

Asignatura: Arquitectura de Sistemas Paralelos

Profesor: Francisco Javier Gómez Arribas
Departamento de Tecnología Electrónica y de las Comunicaciones

UNIVERSIDAD AUTONOMA DE MADRID

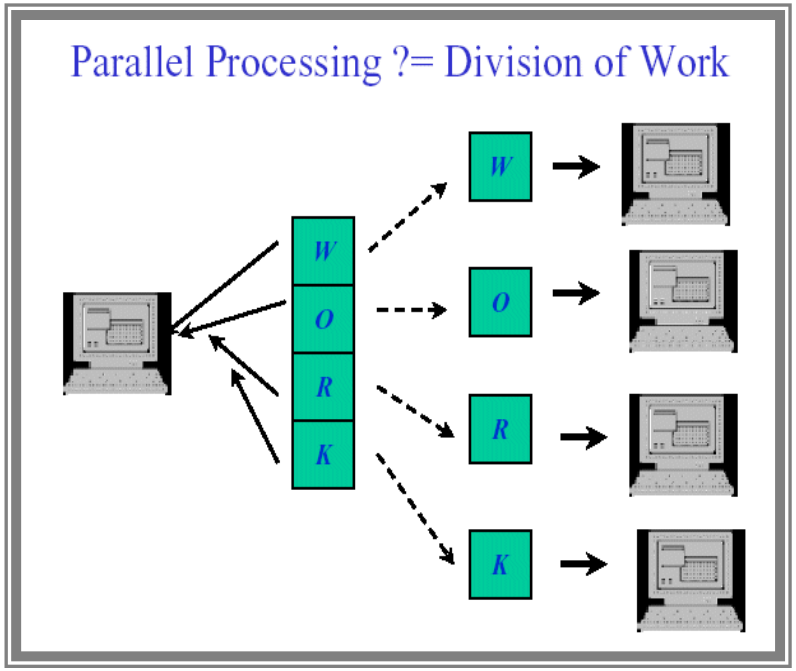
Escuela Politécnica Superior

Tema 1: Contenidos

- **Introducción a la computación paralela**
  - Motivación y Objetivos
  - Métricas de rendimiento. Leyes de Amdahl y Gustafson
  - Aplicaciones de la computación paralela
  - Arquitecturas para procesamiento en paralelo: Clasificación
- **Evolución de sistemas de computación paralela**
  - Integración, Ley de Moore, Limitaciones Tecnológicas, Mejoras Arquitectura de procesador, Mejora de Interconexión
  - Clusters
  - Top 500. Ejemplos de SuperOrdenadores
- **Modelos de Programación**

Computación Paralela

Parallel computer: *"A collection of processing elements that communicate and cooperate to solve large problems"*(ALMASE and GOTTLIEB, 1989)



Metodología de programación:

- **Dividir el trabajo en tareas independientes.**
- **Asignar las tareas a diferentes procesadores que trabajan simultáneamente.**

Objetivo:

- **Incrementar la rapidez o la precisión al solucionar problemas computacionales grandes.**

Dificultades:

- **Coordinación, comunicación, control y monitorización del proceso….**

y Debug

Computación paralela vs distribuida

La computación distribuida se caracteriza por:

- **Dividir un problema en partes diferenciadas, que se ejecutan en sistemas especializados para esa tarea.**
- **Las tareas no se ejecutan necesariamente en paralelo, aunque si no hay dependencias se intentan ejecutar en paralelo.**
- **Se busca fiabilidad con redundancia y se optimiza la utilización de recursos.**

La computación paralela se caracteriza por:

- **Dividir un problema en partes independientes, que admitan ejecución en paralelo.**
- **Las tareas se deben ejecutar en paralelo, con la limitación de las dependencias que existan entre ellas.**
- **Se mejora el rendimiento y/o la cantidad de trabajo realizado**

Computación paralela: Objetivos

- **Mejorar el rendimiento (Throughput): Calcular P problemas simultáneamente en el menor tiempo.**
  - **Ejecutar P instancias de un programa secuencial con diferentes conjuntos de datos ("embarrassing parallelism"); SETI@home,**
  - **Dificultades: limitado por los recursos en cada nodo (unidad de proceso)**
- **Disminuir el tiempo de ejecución: Calcular en menos tiempo (1/n)**
  - **Ejecutar (con n procesos) un programa paralelo que obtiene la solución juntando los resultados de los procesos; por ejemplo, encontrar la posición de un patrón en una base de datos.**
  - **Dificultades: Escribir el programa en paralelo; comunicación. Ley de Amdahl**
- **Aumentar el tamaño del problema: Calcular un problema con un número P veces mayor de datos**
  - **Ejecutar un instancia (con P=n procesos) de un programa paralelo utilizando la suma de todas la memorias locales para incrementar el tamaño del problema**
  - **Solucionar con mayor precisión simulaciones de Montecarlo.**
  - **Ley de Gustafson.**
  - **Dificultades: Escribir el programa en paralelo; comunicaciones**

Análisis de prestaciones al trabajar en paralelo

$$\text{Aceleración} \leq \frac{\text{Tiempo Trabajo Secuencial}}{\textit{Max}\,(\text{Tiempo Trabajo} + \text{Tiempo Sincr.} + \text{Tiempo Comunic.} + \text{Extra})}$$

Solucionar comunicación y el balaceo de carga es un problema de complejidad NP en el caso general

Afortunadamente en la práctica, funcionan heurísticas simples de planificación

Existe un compromiso (tradeoff) entre :

- **Balanceo de carga.**
- **Minimizar sincronización.**
- **Minimizar comunicaciones.**
- **Trabajo extra**

Modelos de trabajo en Computadores Paralelos

Si la carga de trabajo *W* o el tamaño del problema es fijo entonces:

- **La eficiencia *E = Aceleración /n*, decrece rápidamente conforme le tamaño de la máquina *n* se incrementa**

La posibilidad de aplicar un computador paralelo escalable para solucionar un problema escalable se justifica cuando:

- **Se mantiene un nivel de eficiencia a medida que se incrementa proporcionalmente el tamaño de la máquina**
- **Idealmente W debe ser una función lineal con *n (Linear scalability in problem size).***

Los modelos de carga de trabajo (Workload) para computadores paralelos son:

- **Modelo de Carga Fija (WC): Corresponde con workload constante.**
- **Modelo de Tiempo Fijo (TC): Tiempo de ejecución constante.**
- **Modelo de Memoria Fija (MC): Limitada por la capacidad de memoria.**

Medidas de Ganancia de Velocidad (Speedup)

Speedup = Aceleración= Rendimiento (n) / Rendimiento (1) =

[Trabajo (n) / Tiempo (n)] / [Trabajo (1) / Tiempo (n))]

Speedup(WC) = Tiempo (1) / Tiempo (n)

Trabajo Fijo (Work Constrained)

Speedup(TC) = Trabajo (n) / Trabajo (1)

Tiempo Fijo (Time Constrained)

Speedup(MC) = (Trabajo (n) / Trabajo (1)) / (Tiempo(n) / Tiempo (1))

Recursos por Procesador (memoria) Fija

Dos visiones (Leyes)

Ley de Amdahl

- Planteada en 1967 por Gene Amdahl
- El Speedup está limitador por la parte serie
- Asume carga de trabajo fijo y tamaño de problema fijo

Ley de Gustafson

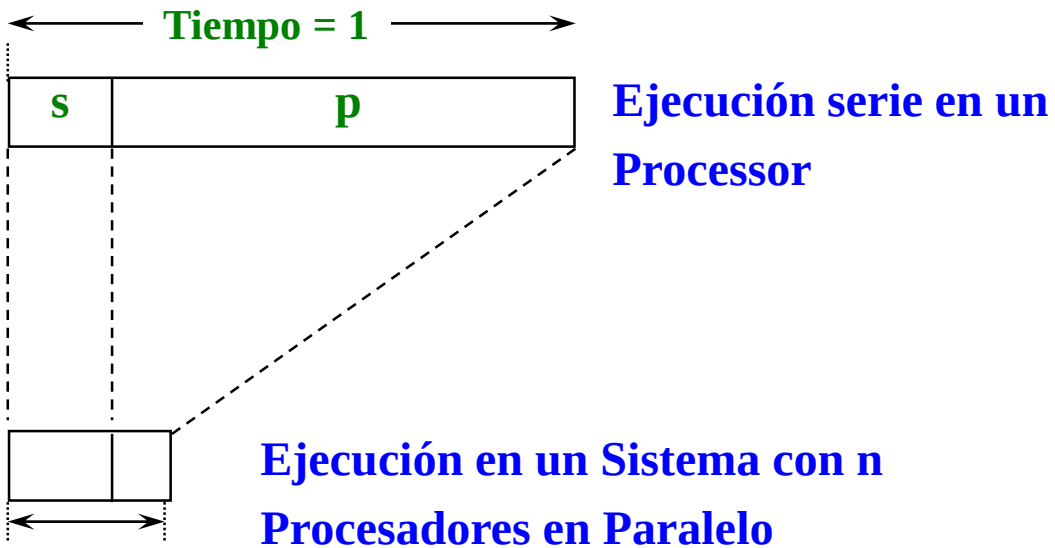
- Observación de John Gustafson en 1988
- Rescata el procesamiento paralelo de la ley de Amdahl
- Propone fijar el tiempo e incrementar la carga de trabajo. La parte serie es un límite pero su efecto es poco significativo.

Ley de Amdahl

Ley de Amdahl: "Cuando la fracción de un trabajo serie de un problema es pequeña, y la denominamos s, la máxima aceleración o Speedup alcanzable (incluso para un número infinito de procesadores) es sólo 1/s."

$$\text{Speedup} = \frac{1}{s + p/n} = s + p$$

by Amdahl, G: "Validity of the single-processor approach to achieve large-scale computer capabilities" AFIPS Conf. Proceedings 30, 1967, pp 483-485



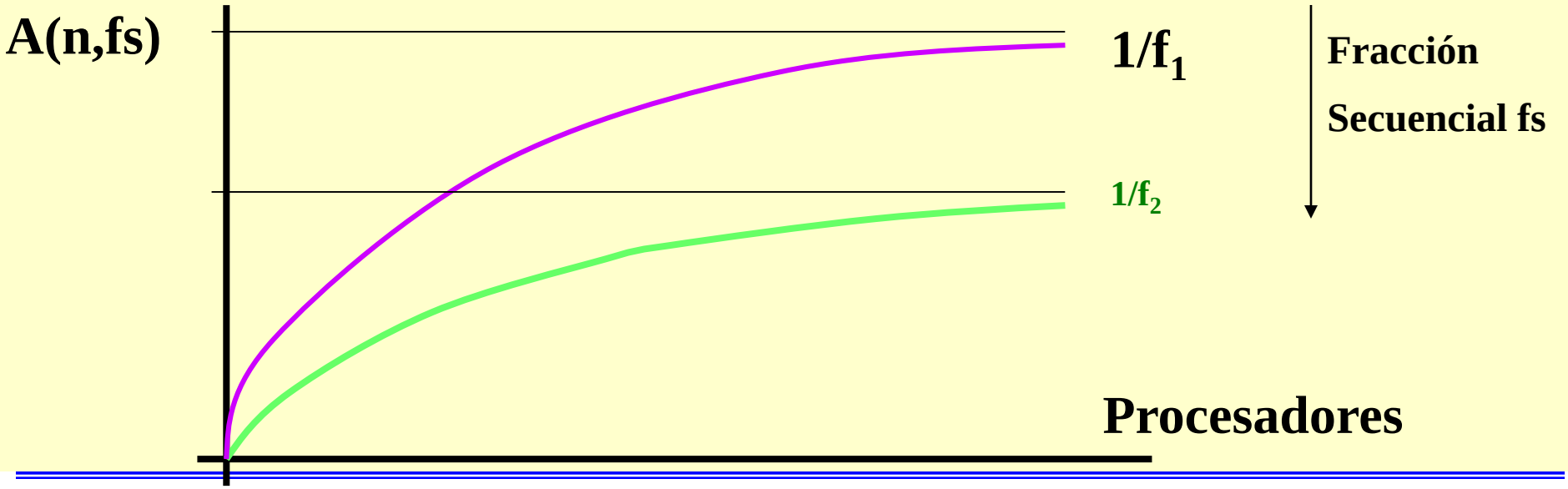
Limites del paralelismo

En todo problema hay una parte secuencial, fs, donde no se puede aprovechar el trabajo cooperativo de los procesadores, limitando de forma importante la ganancia de velocidad alcanzable con n procesadores

$$fs + fp = 1$$

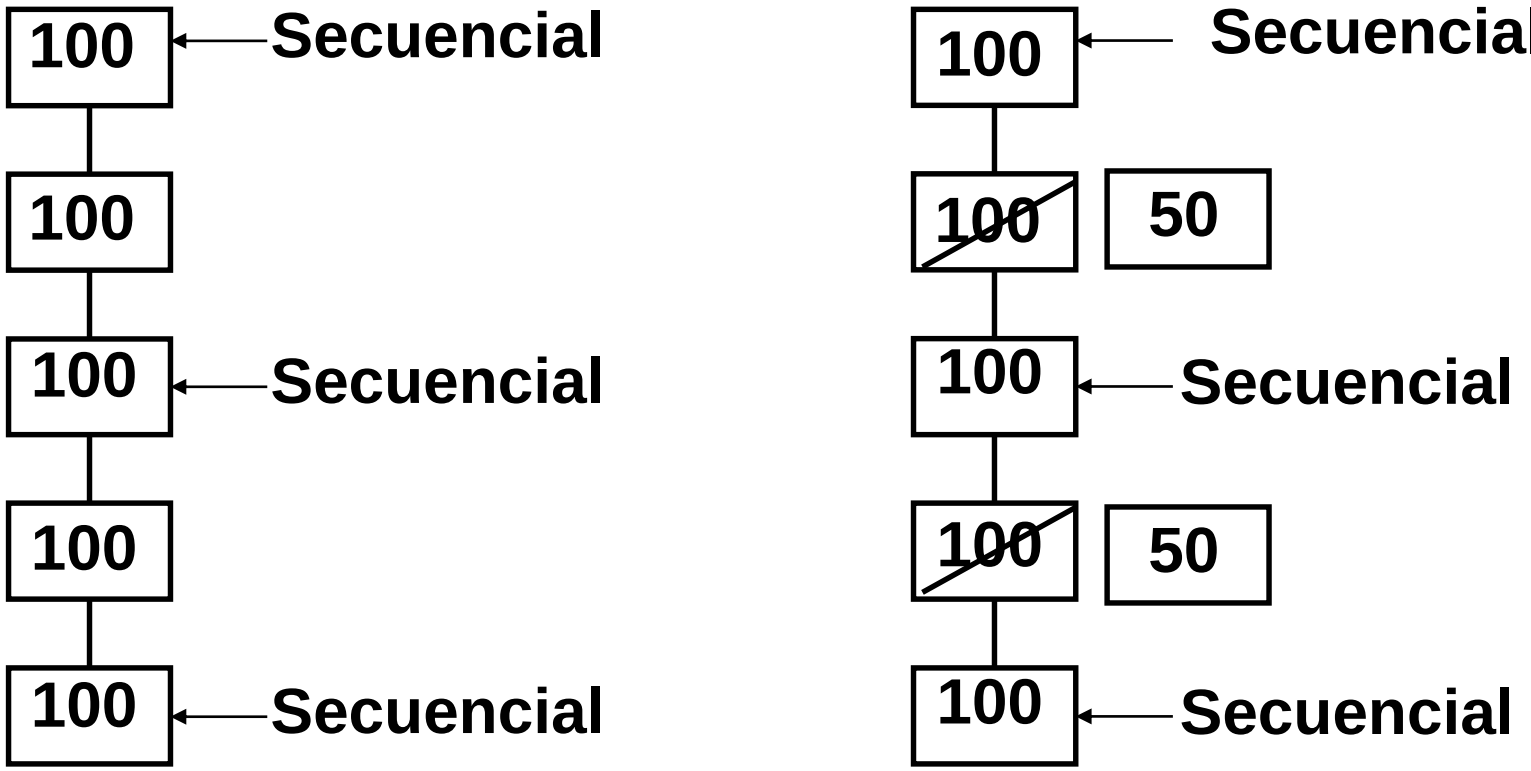
$$A=1/(fs + fp/n)$$

$$A(n,fs) < n / (1 + fs*(n-1))$$



Ley de Amdahl

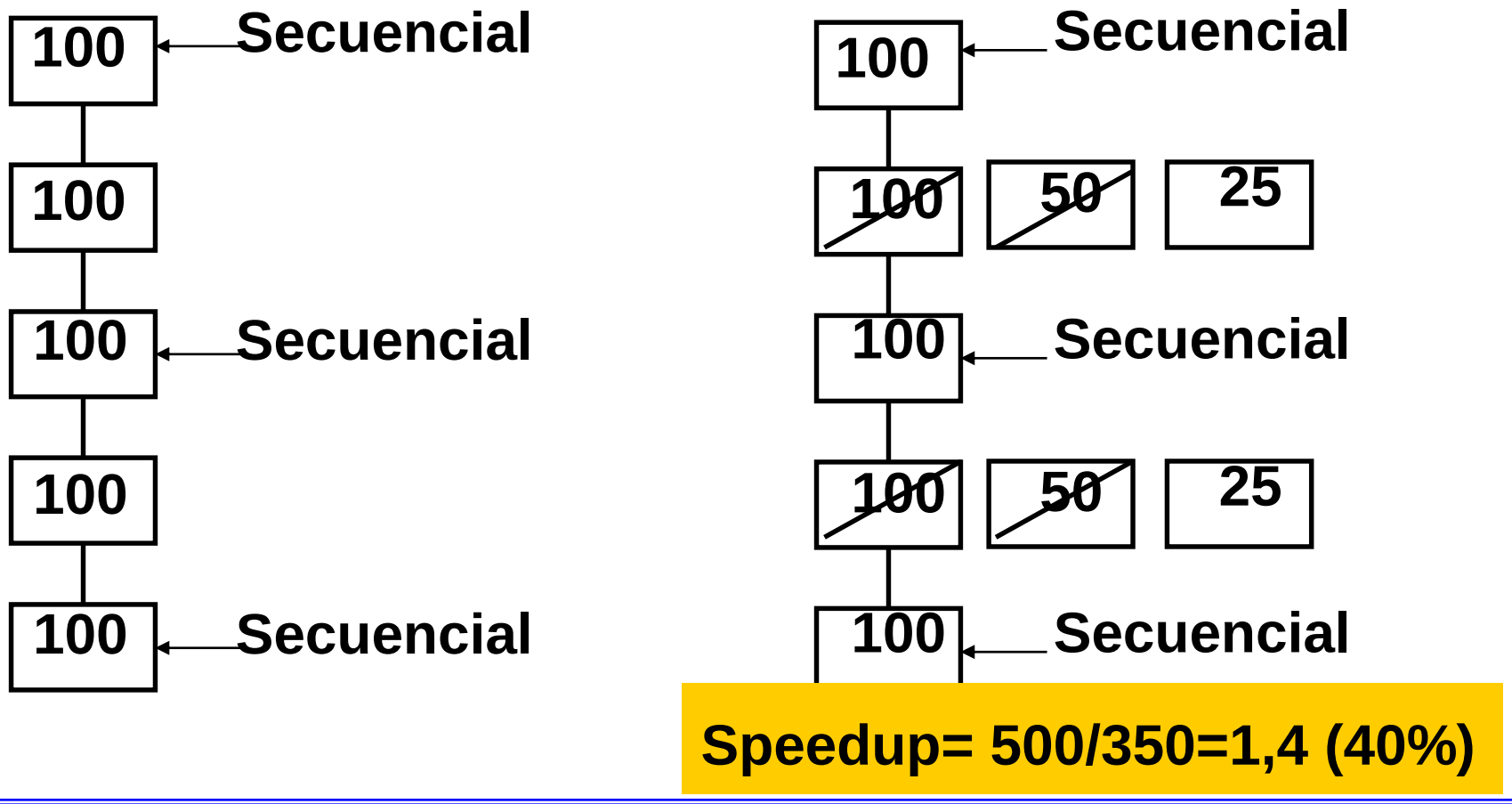
Se paraleliza las partes 2 y 4
Con 2 procesadores



Speedup= 500/400= 1,25 (25%)

Ley de Amdahl (4 proc)

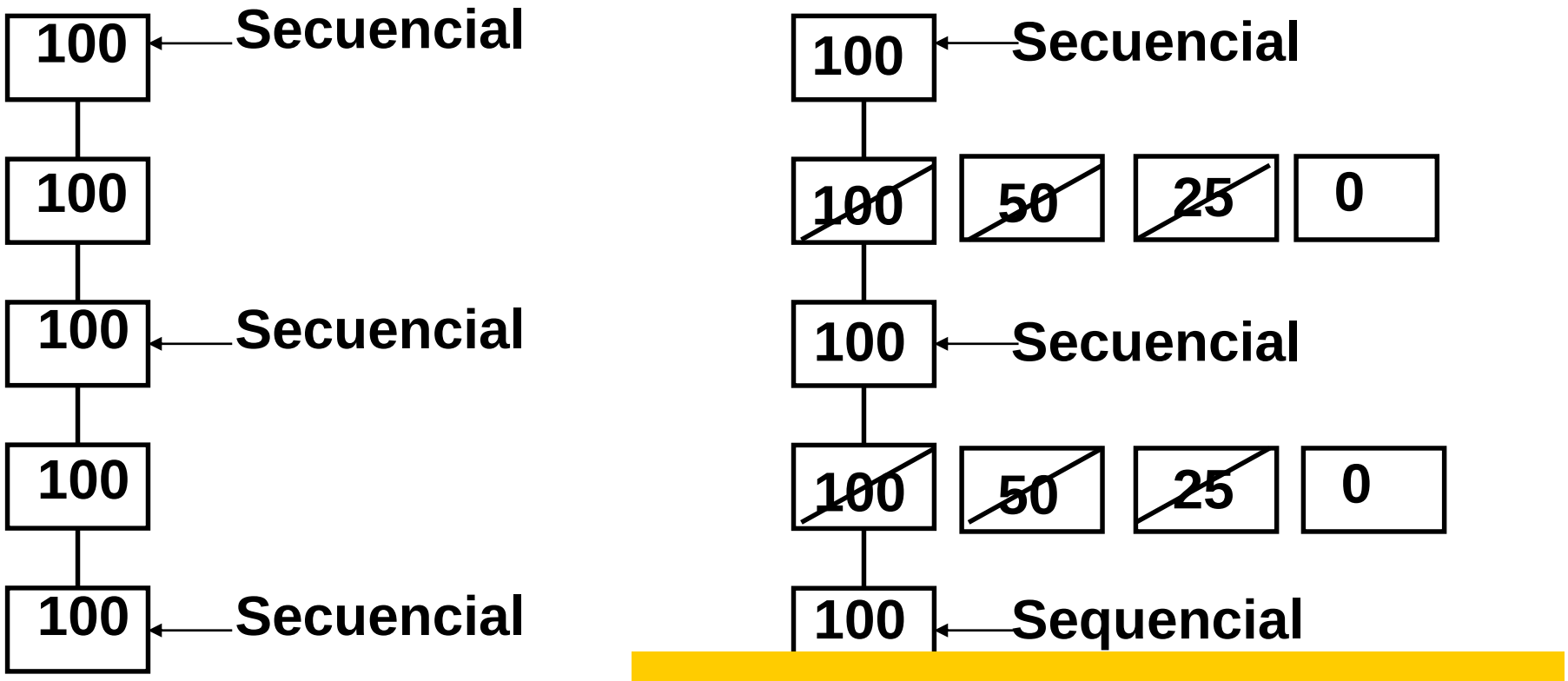
Se paraleliza las partes 2 y 4
Con 4 procesadores



Speedup= 500/350=1,4 (40%)

Ley de Amdahl (infinitos proc)

Se paraleliza las partes 2 y 4
Con INFINITOS procesadores

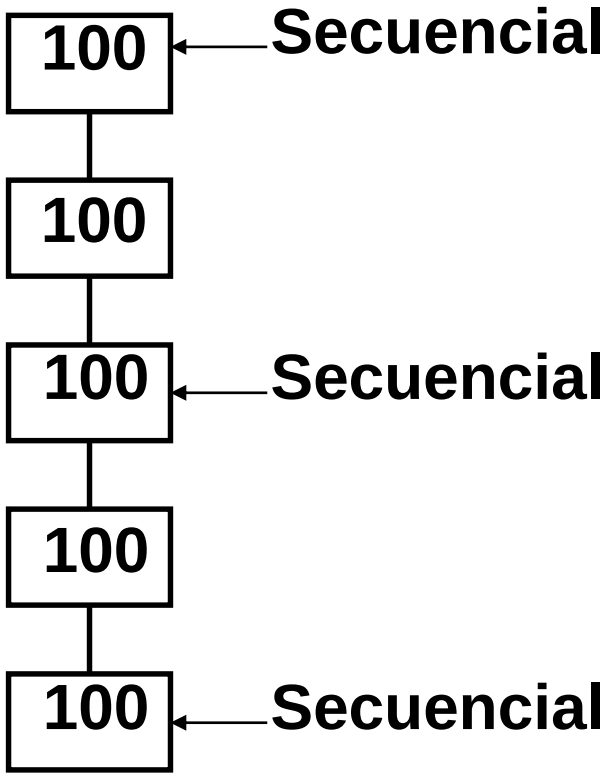


Speedup= 500/300=1,7 (solo 70%)

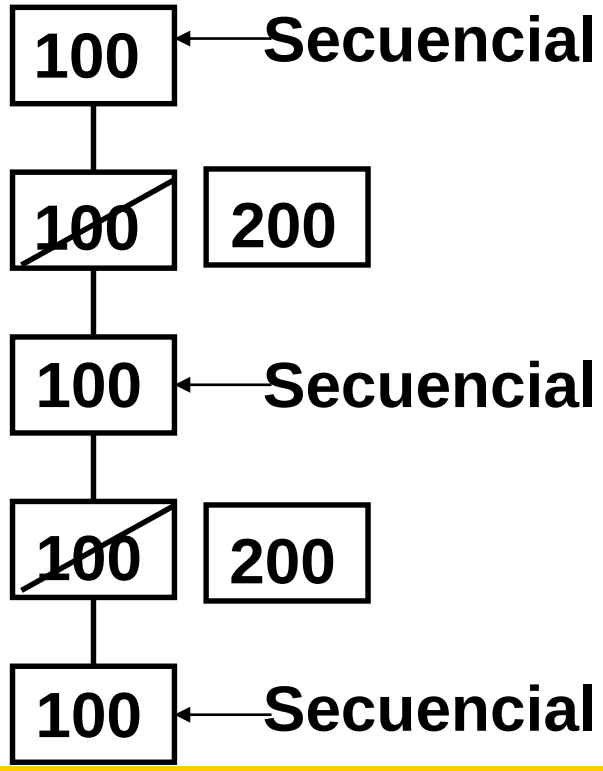
¡¡ Los sistemas Multicore no parece una solución!!

Ley de Gustafson

Los recuadros son unidades de trabajo!



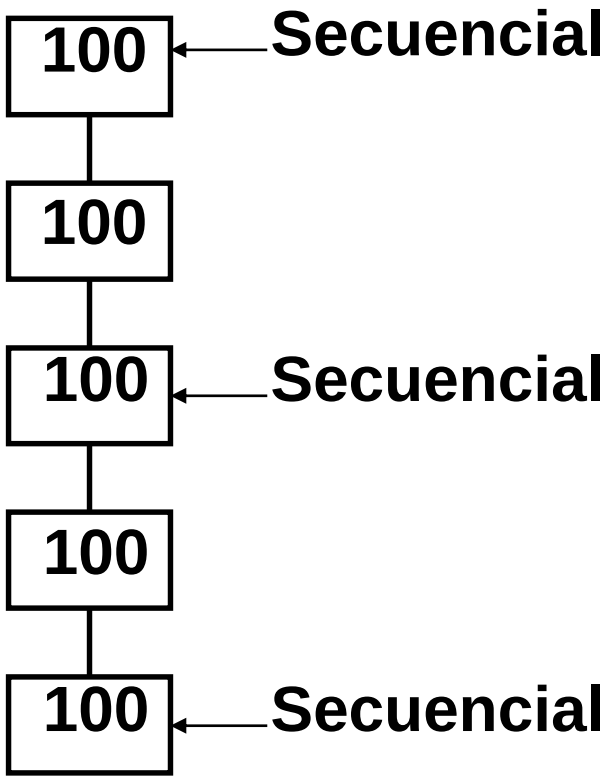
500 unidades tiempo, pero 700 unidades de trabajo



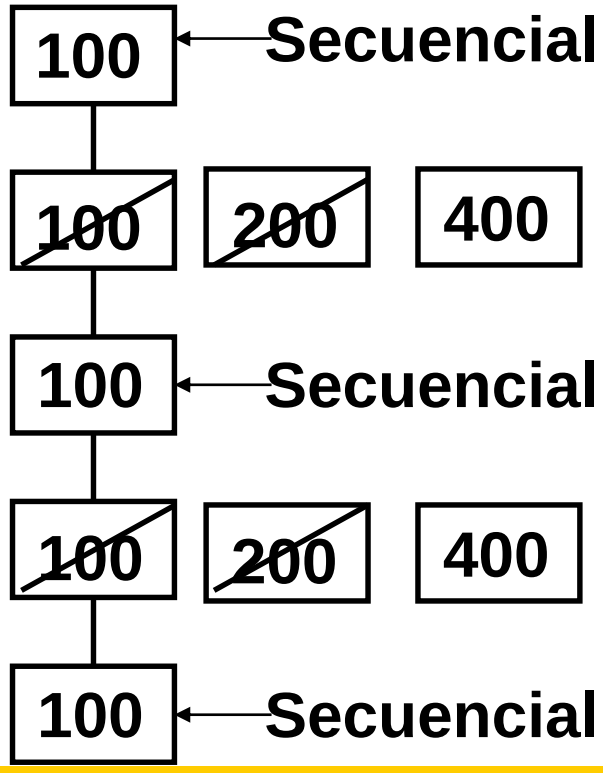
Speedup: 700/500= 1,4 (40%)

Ley de Gustafson

Los recuadros son unidades de trabajo!



500 unidades tiempo, pero 1100 unidades de trabajo



Speedup= 1100/500=2,2 (120%)

Ley Gustafson vs Ley de Amdahl

Las observaciones de Gustafson

- Conforme aumenta el numero de procesadores, el usuario incrementa (escala) el tamaño del problema
- El cuello de botella que representa la parte serie no aumenta al escalar el problema.

El incremento de procesadores consigue un crecimiento lineal del speedup

- 20 procesadores son aproximadamente dos veces mas rápidos que 10

Se justifica porque son necesarios los supercomputadores

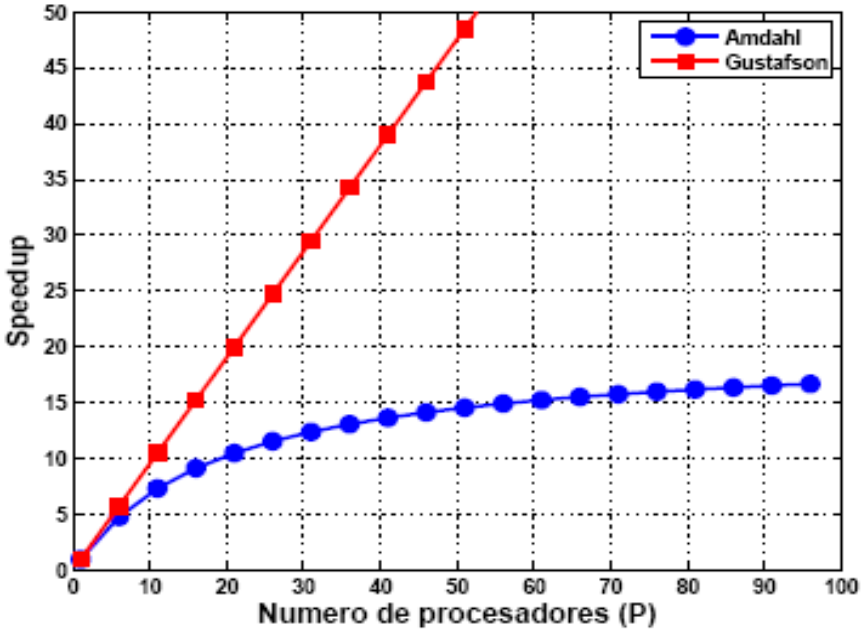
- Más procesadores permite incrementar el tamaño del problema

Ref: http://www.scl.ameslab.gov/Publications/Gus/AmdahlsLaw/Amdahls.html

Ley Gustafson vs Ley de Amdahl

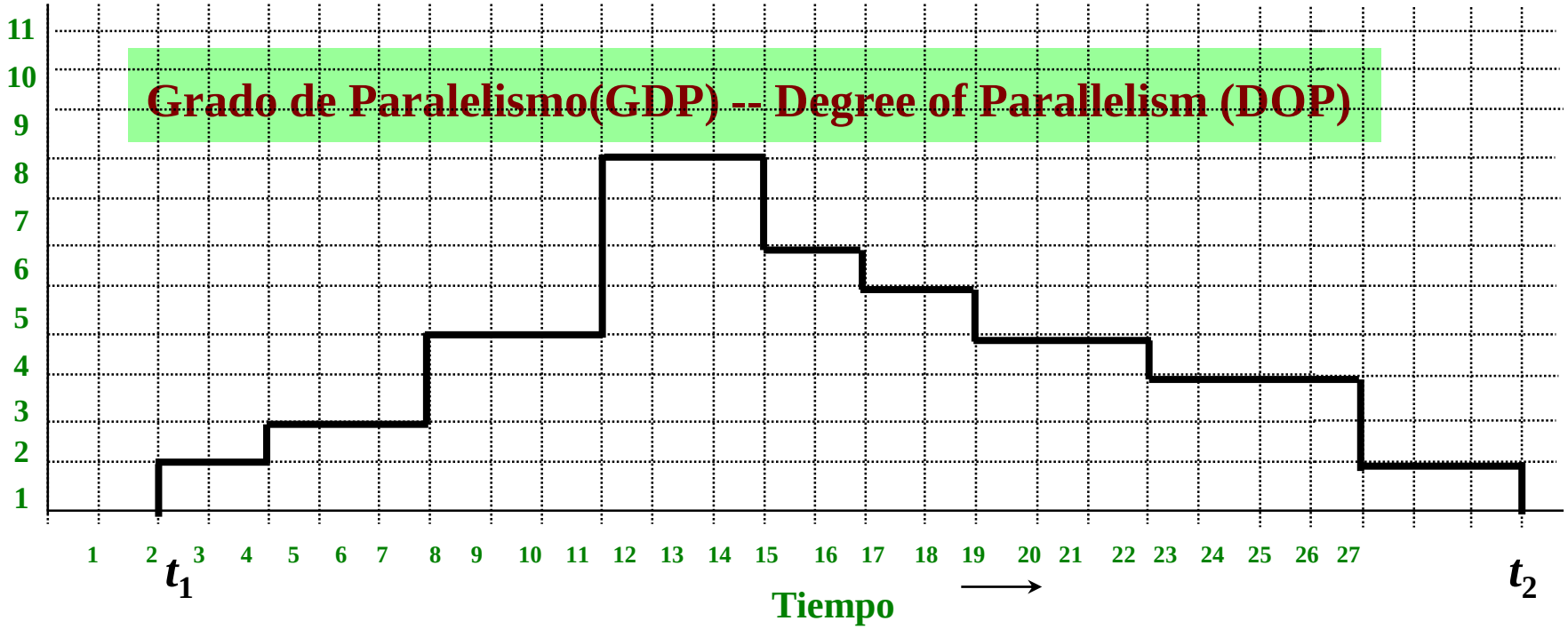
Cuando la fracción paralelizable es escalable con n, manteniendo la fracción serie intacta, obtenemos un speedup escalable

$$S(n) = \frac{T_s + nT_p}{T_s + T_p}$$



En la línea roja, el tiempo monoprocesador crece con el número de procesadores, y el tiempo paralelo permanece constante

Grado de paralelismo



- Refleja la coincidencia entre el paralelismo software y el hardware.
- Mide en cada periodo de tiempo el número de procesadores utilizados(#).
- El perfil del paralelismo es un gráfico del DOP como una función del tiempo.
- Idealmente se dispone de recurso ilimitados (n >> m).

n : procesadores homogéneos

m : máximo paralelismo en un perfil

Δ : capacidad de computo de un procesador (tasa de ejecución sin overhead)

DOP = # procesadores ocupados durante un instante de observación.

Computación paralela: Necesidad y aplicaciones

¿ Se necesita la programación paralela y la computación de altas prestaciones (HPC)?

- **Hay problemas complejos (los denominados "grand challenges") que demandan potencia de computación:**
  - **Simulaciones climáticas y/o geofísica (tsunami, tornados)**
  - **Simulaciones estructurales, flujos,…(crash test, CFD)**
  - **Sistemas avanzados de diseño, realidad Virtual (CAD, efectos)**
  - **Análisis de datos (Large Hadron Collider CERN, Carnivore,..)**
  - **Aplicaciones militares, médicas (crypto analysis,)**

– Incrementar de prestaciones con:

- **Hardware mas rápido, más memoria ("fuerza bruta")**
- **Algoritmos más eficientes, optimización ("inteligencia")**
- **Computación Paralela (Eficiencia = trabajo/unidad proceso)**

Ámbito de Aplicación de la Computación Paralela

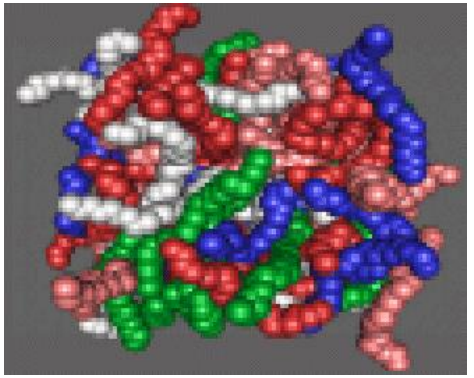
Algunos ejemplos donde se utiliza/necesita la computación paralela

Aplicaciones para solucionar grandes problemas de cálculo científico

- **Modelo climático global: simulación y predicción.**
  - **Superficie de la tierra $5x10^8$ km^2**
  - **1 punto por km^2 y 15 niveles de altura**
  - **Cada minuto 6 datos por punto y 3 coord.**
  - **3 Gigabytes de datos/segundo**

- **Dinámica molecular .**
  - **10^6 partículas interactuando cada segundo**
  - **3 coordenadas, 3 velocidades, 3 fuerzas.**
  - **Para simular 1 segundo se necesita...**

    9000 PetaFlops



- **Secuenciación de Proteinas (sequence matching)**

Ejemplos de Computación Paralela

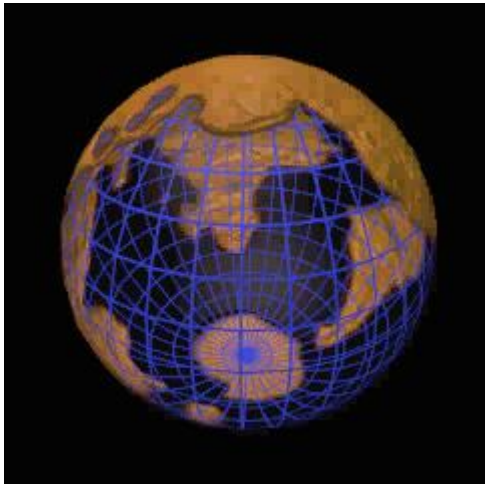
2009 Avatar. (Director James Cameron)

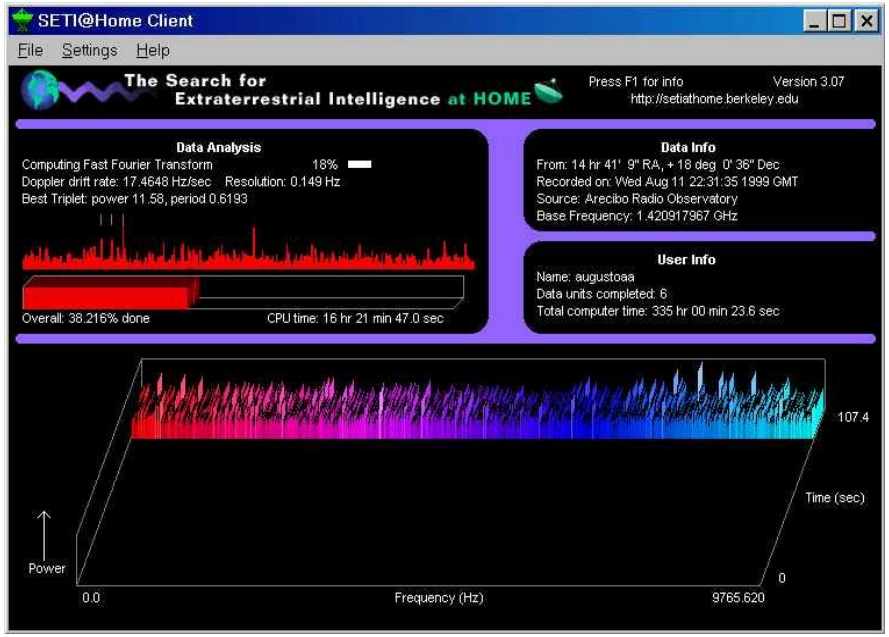
- Wellington, NZ-based Weta Digital served as the primary visual effects vendor on "Avatar," creating the stunningly realistic 3D world of Pandora
  - **Weta spent close to two years on the production.**
  - **At peak production, the BlueArc system served over 34,000 cores on the render farm, handling 16,000 concurrent read and write processes at a sustained load of 8 gigabits per second**
- **Weta used a clustered system of 12 Titan servers to store and manage over 500 terabytes of data feeding thousands of render nodes, as well as an additional 700 terabytes of nearline storage.**

Ejemplos de Computación Paralela

Otras aplicaciones donde la computación paralela procesa masivamente.

- **Aprovechar Internet como fuente de recursos**
- **Millones de ordenadores interconectados en red**

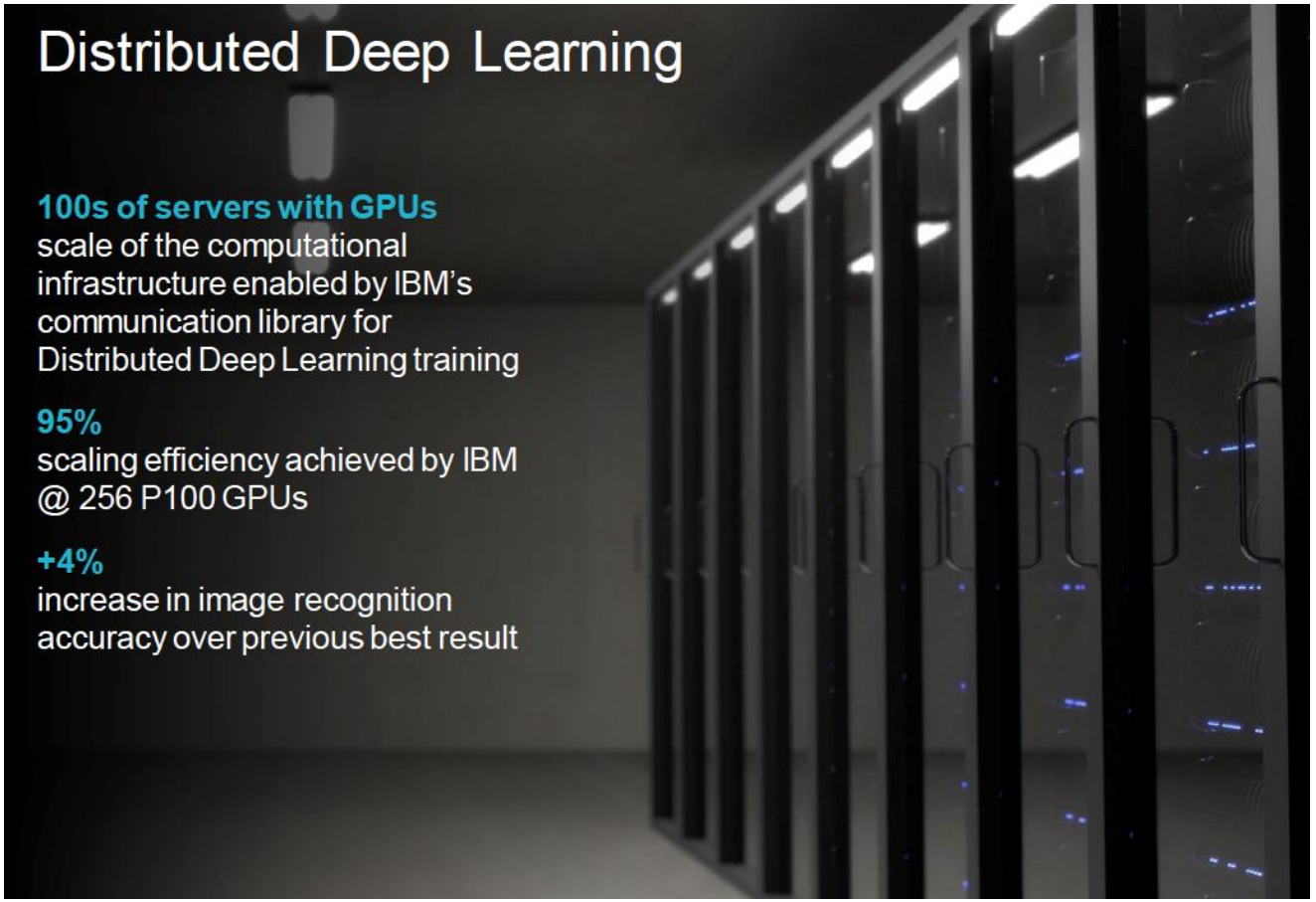




Proyecto SETI@HOME

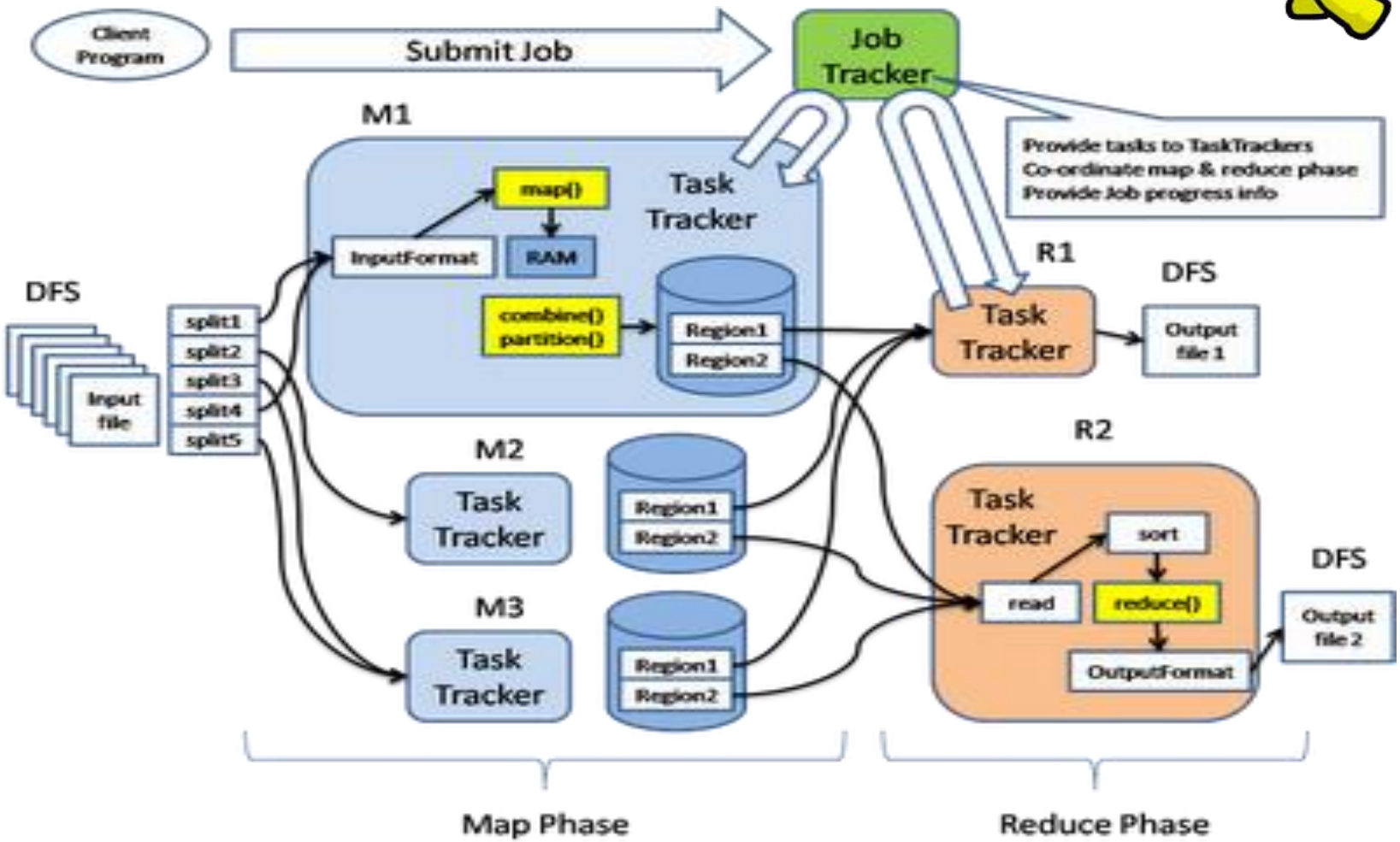
Búsqueda de Inteligencia Extraterrestre

Computación paralela con aceleradores/coprocesadores GPU



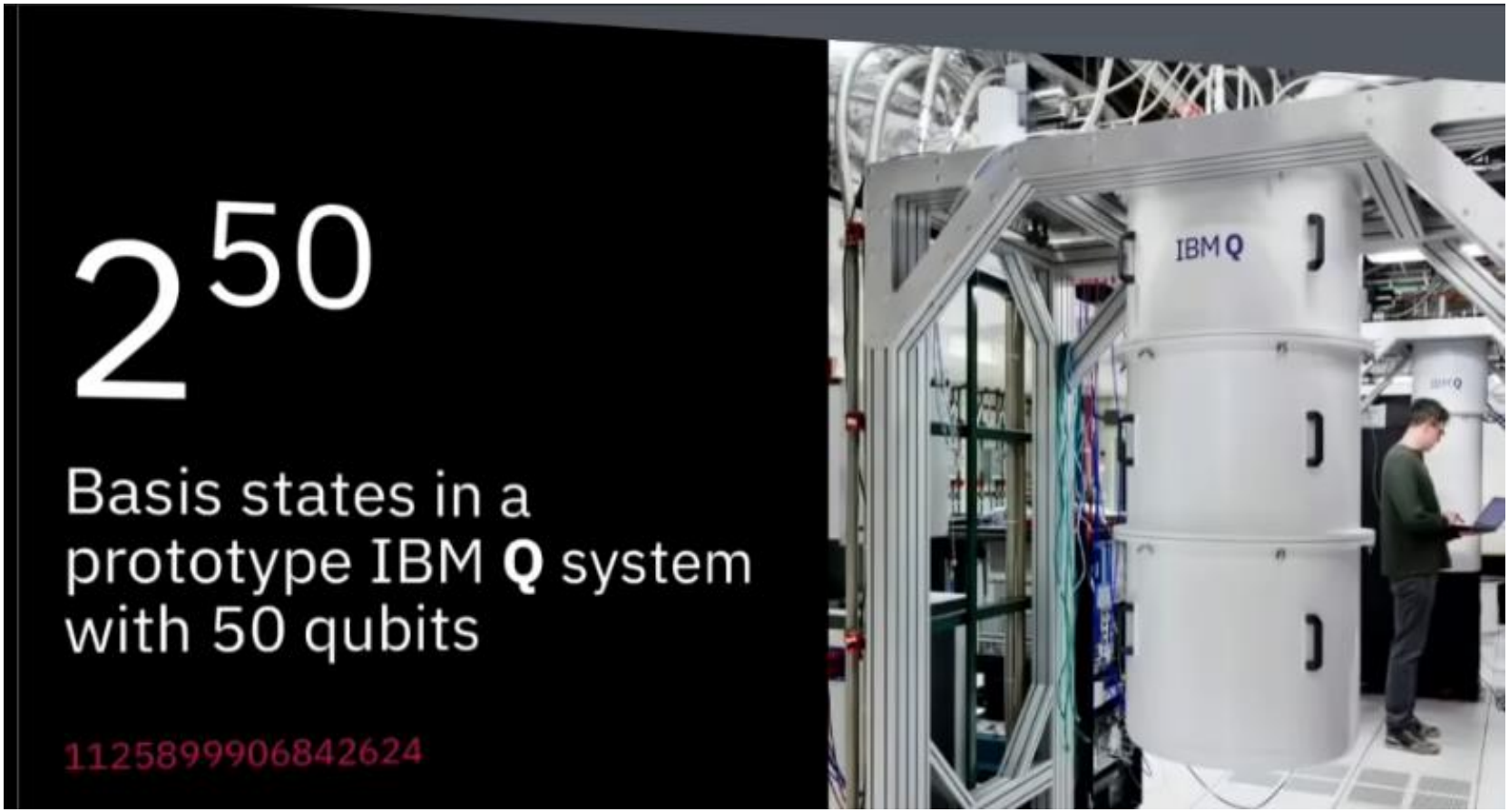
Otros ejemplos de Computación Paralela

Procesos Bigdata con Hadoop: MapReduce



Otros ejemplos de Computación Paralela

Computación Cuántica: Escala exponencial



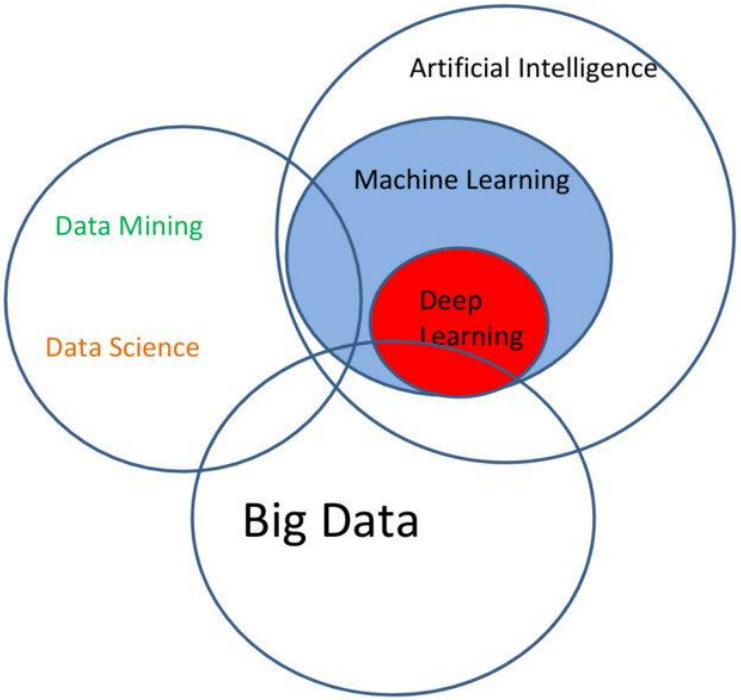
Escalado de potencia en un computador cuántico

Quantum computers, would be based on the strange principles of quantum mechanics, in which the smallest particles of light and matter can be in different places at the same time.

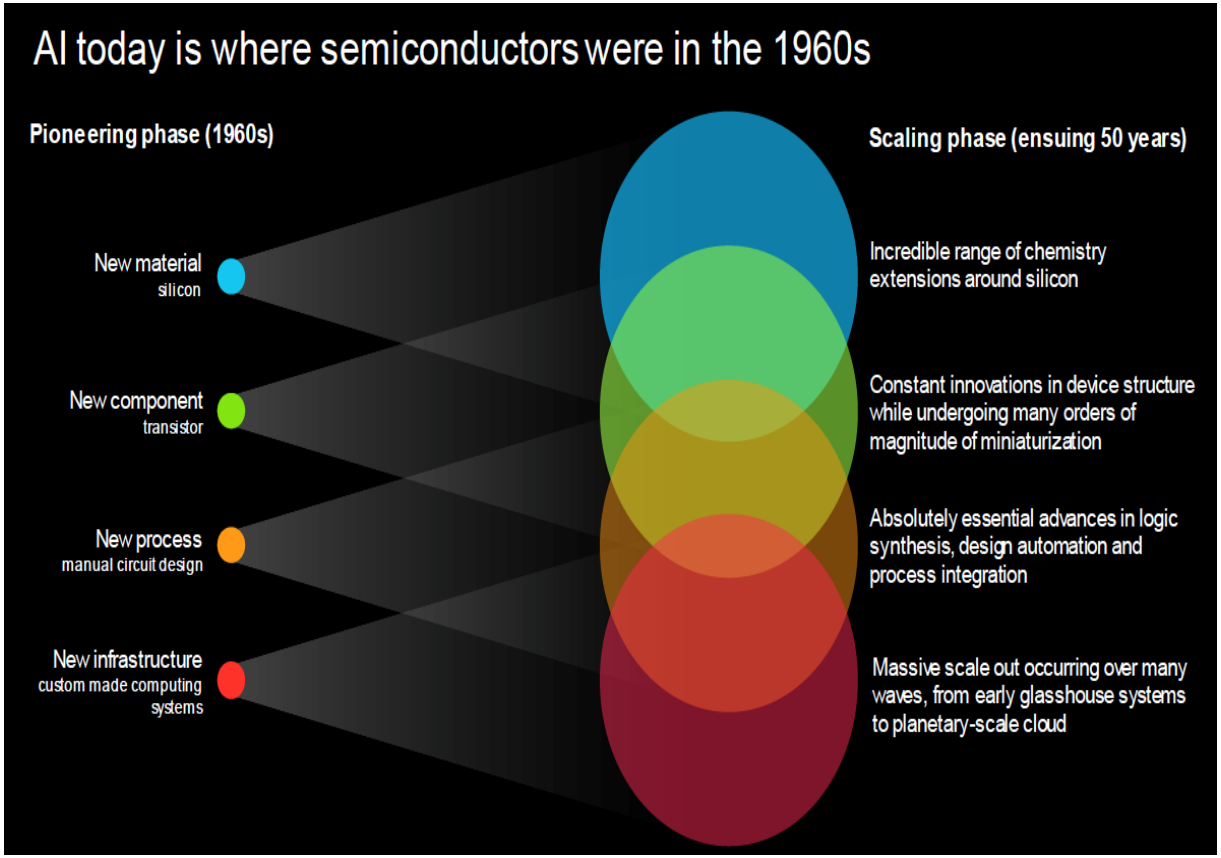
In a quantum computer, one "qubit" - quantum bit - could be both 0 and 1 at the same time. So with three qubits of data, a quantum computer could store all eight combinations of 0 and 1 simultaneously. That means a three-qubit quantum computer could calculate eight times faster than a three-bit digital computer.

Typical personal computers today calculate 64 bits of data at a time. A quantum computer with 64 qubits would be 2 to the 64th power faster, or about 18 billion billion times faster. (Note: billion billion is correct.)

Tecnologías para BigData, Data Science, y ML /DL



Optimización End to End de Software y Hardware



Procesamiento Paralelo

Mas aplicaciones que justifican la computación paralela

- **BigData: almacena y trata grandes cantidades de datos**
  - **Proyecto Genoma humano.**
  - **Minería de datos y extracción de conocimiento.**
  - **Procesamiento de imágenes y de información multimedia.**
  - **Búsqueda y almacenamiento de información de Internet.**

CONCLUSION => Se necesita computación paralela

Aproximaciones para obtener un Sistemas de Computación de alto rendimiento:

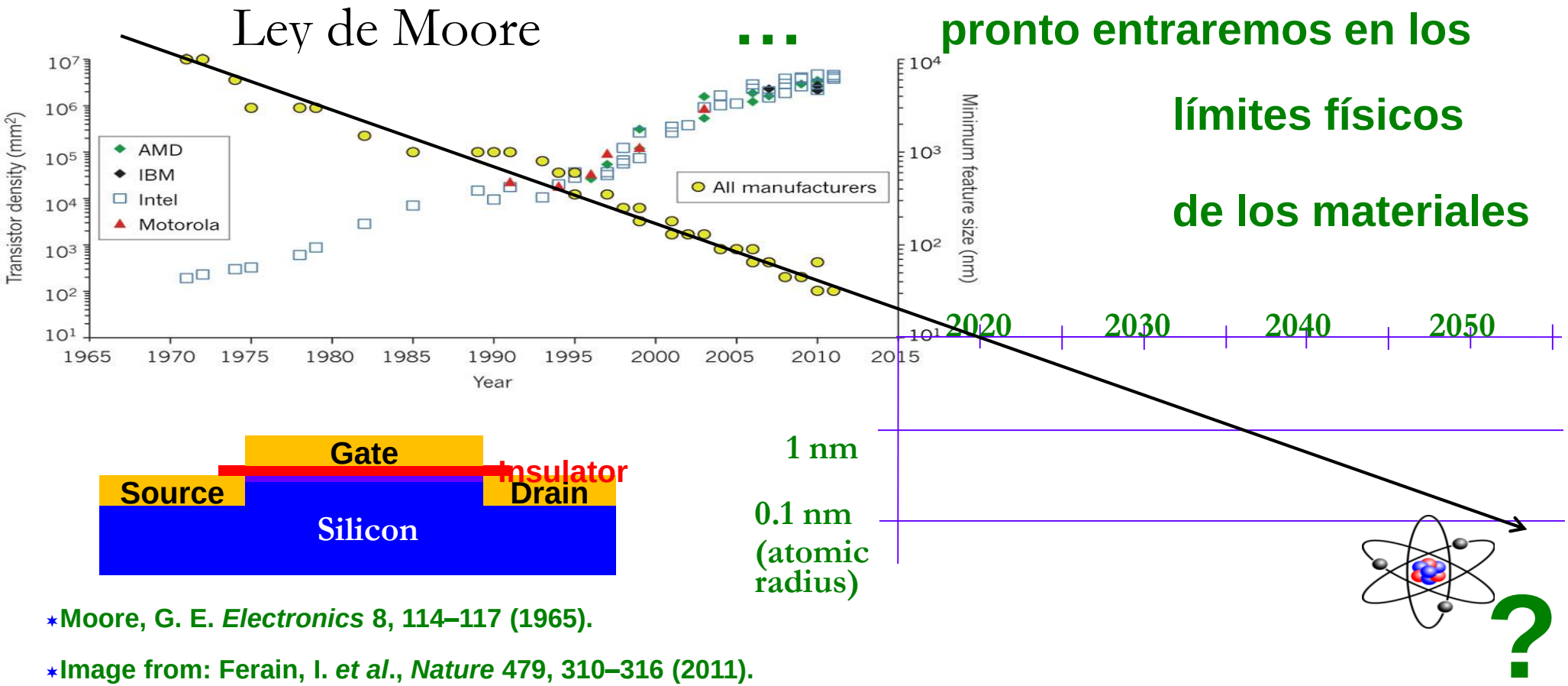
- **Superordenadores: multiprocesadores trabajando en paralelo**
- **Cluster de ordenadores: Paralelismo comunicando ordenadores por red.**
- **Nuevas arquitecturas para escalar el procesamiento.**

Computación paralela: Motivación

¿ Se justifica la inversión en arquitecturas paralelas?

- La computación basada en una arquitectura paralela proporciona hoy unas prestaciones equivalentes a las que se obtendrán dentro de unos años con sistemas de sobremesa.
  - Hay aplicaciones reales en la actualidad que ya necesitan estas prestaciones.
- Proporcionan una alternativa para mejorar las prestaciones sin depender únicamente de factores tecnológicos
  - Por ejemplo rendimiento por aumento de la frecuencia de reloj del procesador.
  - Su uso es inevitable si se aproximan los limites de una tecnología
- Posibilidades de explotar el paralelismo a distintos niveles: ILP, TLP, Paralelismo de datos, Cluster, Grid, Cloud,….

Futuro de los procesadores



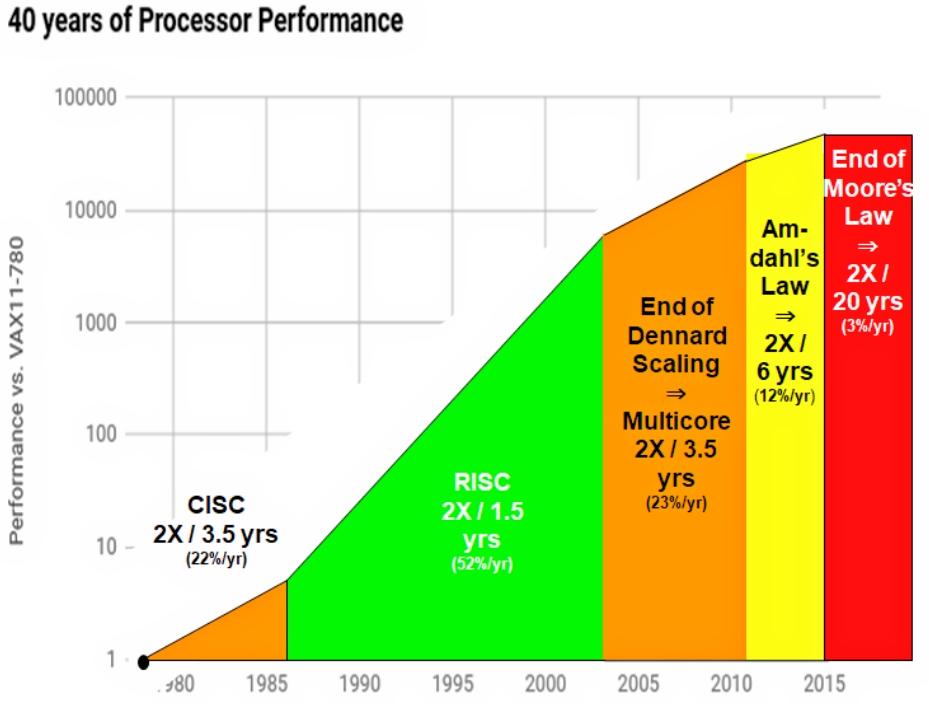
⋆Moore, G. E. *Electronics* 8, 114–117 (1965).

⋆Image from: Ferain, I. *et al*., *Nature* 479, 310–316 (2011).

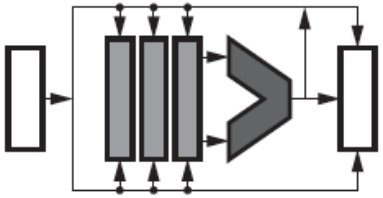
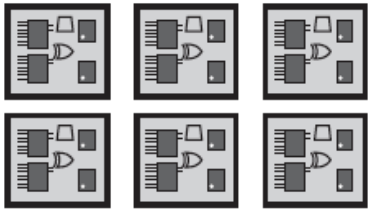
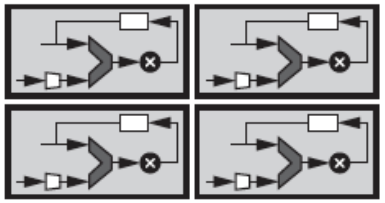
Evolución del rendimiento de los procesadores

El rendimiento de los procesadores de propósito general se está estancando:

- **Se necesita nuevas tendencias para dar soluciones:**
  - **Arquitecturas específicas para cada dominio.**
  - **Computación aproximada.**
  - **Tecnologías más disruptivas: procesadores cuánticos.**



Diferentes necesidades de computación

Scalar Processing	Adaptable Hardware	Vector Processing *(e.g., GPU, DSP)*
		
Complex Algorithms and Decision Making	Processing of Irregular Data Structures *Genomic Sequencing*	Domain-specific Parallelism
	Latency Critical Workloads *Real-Time Control*	Signal Processing *Complex Math, Convolutions*
	Sensor Fusion *Pre-processing, Programmable I/O*	Video and Image Processing

WP505_02_092918

Necesidades actuales de computación

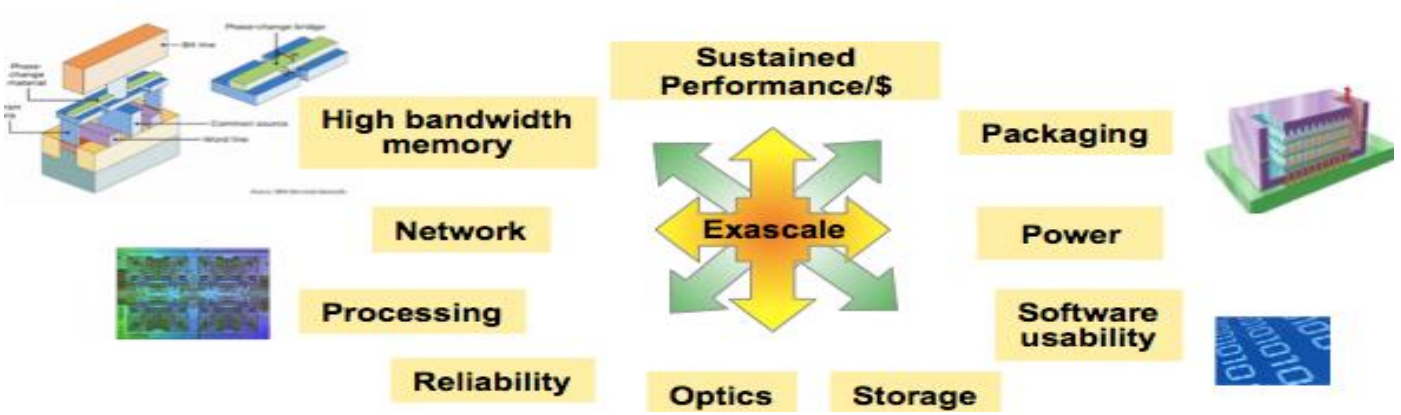
Ámbito de aplicabilidad de los sistemas es muy exigente:

- **Big Data and High Performance Analytics**
- **Data-centric Computing**

¿ Son necesarias nuevas arquitecturas?

Tendencias en computación de altas prestaciones

- **Evolución de los procesadores para la era Exascale**
  - **Evitar dependencia de factores tecnológicos**
  - **Arquitecturas caracterizadas por llevar la computación mas cerca de los datos**
  - **Nuevas tecnologías de memoria y empaquetamiento.**



- **Nodos con arquitectura heterogénea interconectados con redes.**
- **Explotar el paralelismo: ILP, TLP, Paralelismo de datos, Cluster, Grid Computing, Cloud Computing**
- **Eficiencia en coste y consumo de potencia.**

Computador paralelo: una definición

Un computador paralelo es un conjunto de elementos de procesamiento que cooperan para resolver grandes problemas computacionales rápida y eficientemente.

Características que lo describen:

- **Número de elementos de procesamiento y potencia de esos elementos.**
- **Memoria disponible.**
- **Tipo de comunicación entre los procesadores: Red de interconexión**
- **Modelos empleados para expresar la cooperación y mantener coherencia**
- **Rendimiento y Eficiencia.**
- **Grado de escalabilidad de la arquitectura.**
- **Potencia eléctrica consumida.**

Clasificacion de las arquitecturas paralelas

Clasificación Estándar de acuerdo a FLYNN (1972)

– Criterio : Flujos (Streams) globales de instrucciones y datos.

- **Flujo de instrucciones (instruc. stream): secuencia de comando a ejecutar.**
- **Flujo de Datos(data stream): secuencia de datos sujetos al flujo de instruc.**

– Subdivisión en dos dimensiones:

- **Cantidad de instrucciones por unidad de tiempo que puede ejecutar.**
- **Cantidad de datos por unidad de tiempo que puede procesar.**

–FLYNN distingue cuatro clases de arquitecturas

- **SISD: single instruction, single data**
- **SIMD: single instruction, multiple data**
- **MISD: multiple instruction, single data**
- **MIMD: multiple instruction, multiple data**

INCONVENIENTE: Computadores muy diferentes pueden pertenecer a la misma clase.

Clasificación de Flynn (1972)

Combina Flujo de Datos y Flujo de Instrucciones con (Single) Único y Múltiple, dando 4 combinaciones:

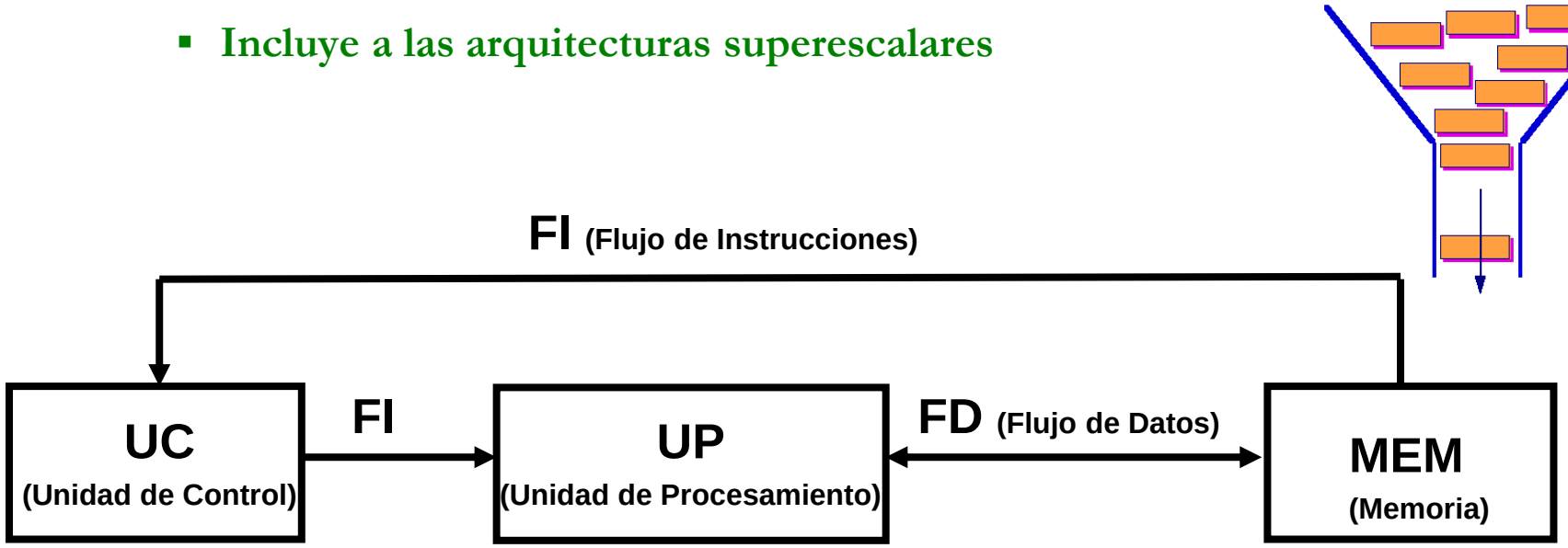
		Flujo de Datos	
		Único	Múltiple
Flujo de Intrucc.	Único	SISD	SIMD
	Múltiple	MISD	MIMD

SISD	Una Instrucción un Dato	Von Neumann
SIMD	Una Instrucción muchos Datos	Vectoriales
MISD	Muchas Instrucciones un Dato	Operación en pipeline (?)
MIMD	Muchas Instrucciones muchos Datos	Multiprocesadores / Multicomputadores

Clasificación de Flynn : Modelo SISD

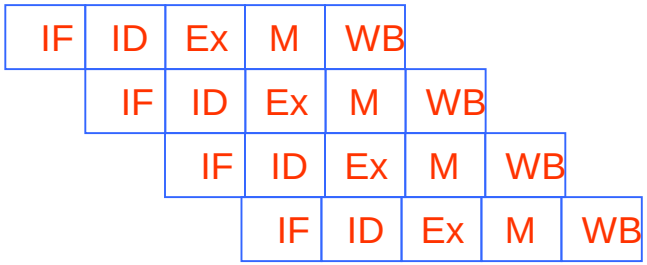
Sistemas SISD (Single Instruction Single Data)

- **Comprende a los sistemas basados en arquitecturas Von Neumann o similares.**
  - **Incluye a las arquitecturas superescalares**

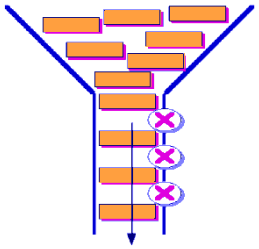


Clasificación de Flynn: SISD Mejorada

Segmentado

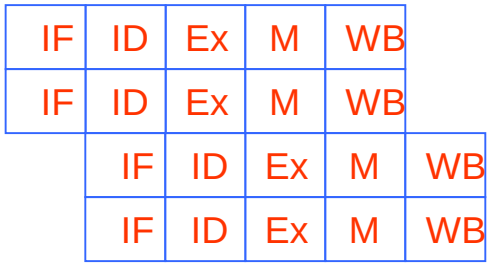


- *Ventajas*
  - Emisión de instrucciones cada ciclo.

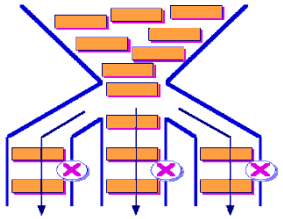


- Limitaciones
  - * Frecuencia de Emisión.
  - * Paradas de FU.
  - * Profundidad de FU.

Superescalar

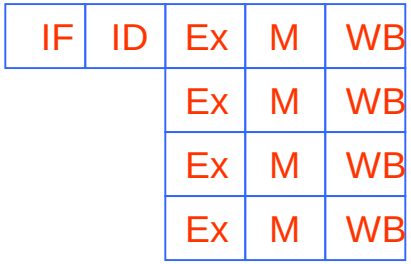


- *Ventajas*
  - * Emisión múltiple de instrucciones.



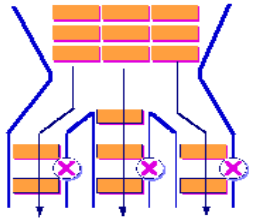
- Limitaciones
  - * Resolución de riesgos

VLIW("EPIC")



- *Ventajas*
  - * El Compilador determina el grado de paralelismo.
  - * Una instrucción implica múltiples operaciones.

- Limitaciones
  - * Empaquetamiento



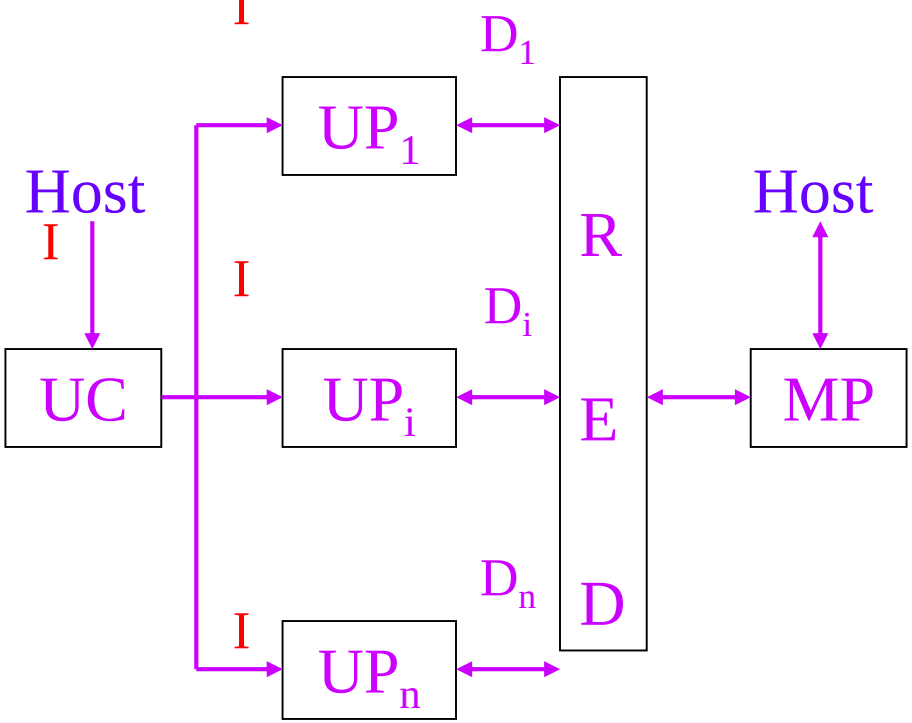
Clasificación de Flynn : Modelo SIMD

SIMD: Una Instrucción muchos Datos (Vectoriales, x86 con SSE)

(Streaming SIMD Extensions)

La Ejecución se caracteriza por necesitar **SINCRONISMO**

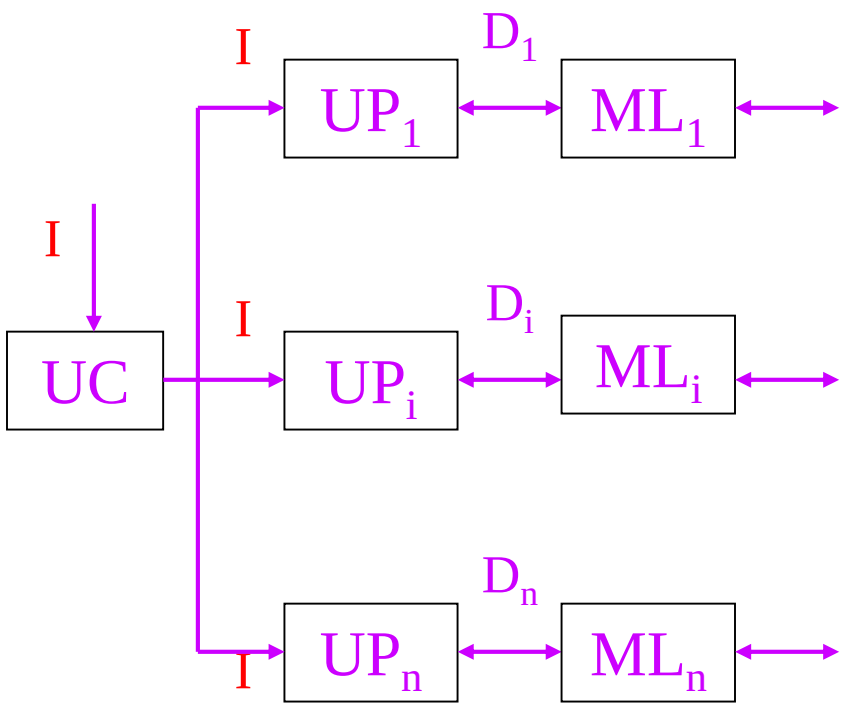
Memoria Común



Acoplamiento fuerte

No escalable

Memoria Distribuida

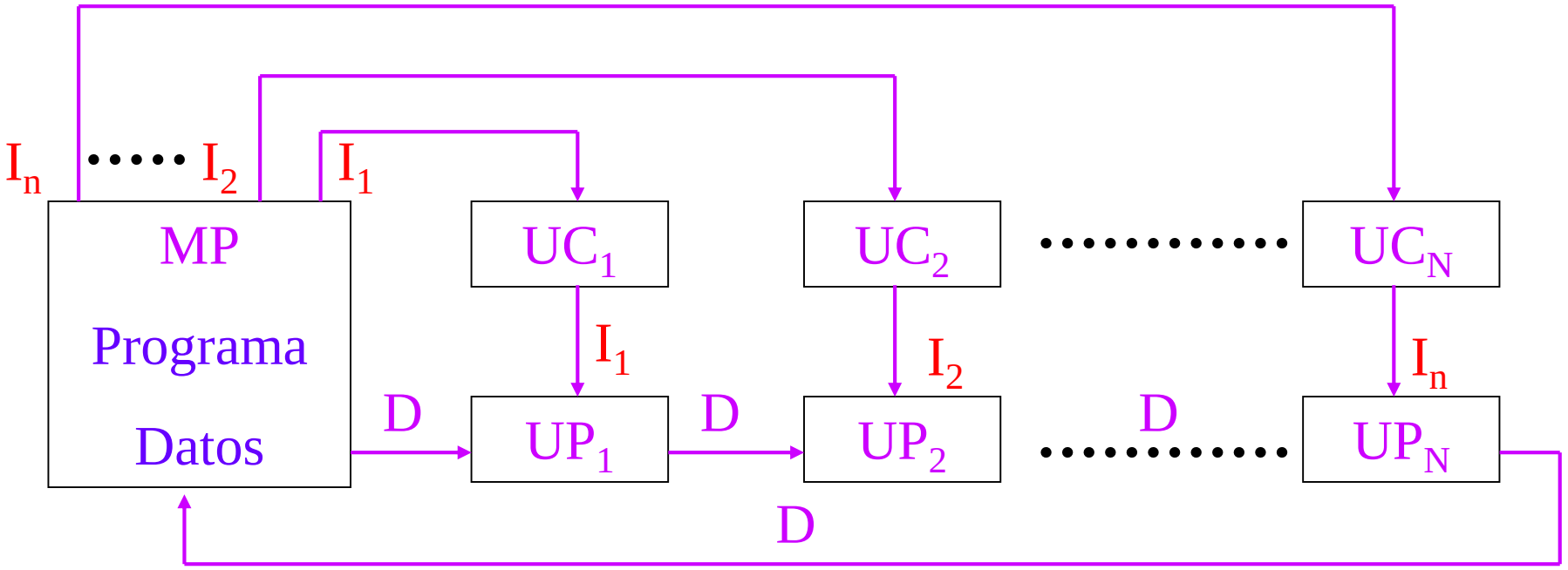


Acoplamiento débil

Clasificación de Flynn : Modelo MISD

MISD: Multiples Instrucciones operando en un Dato

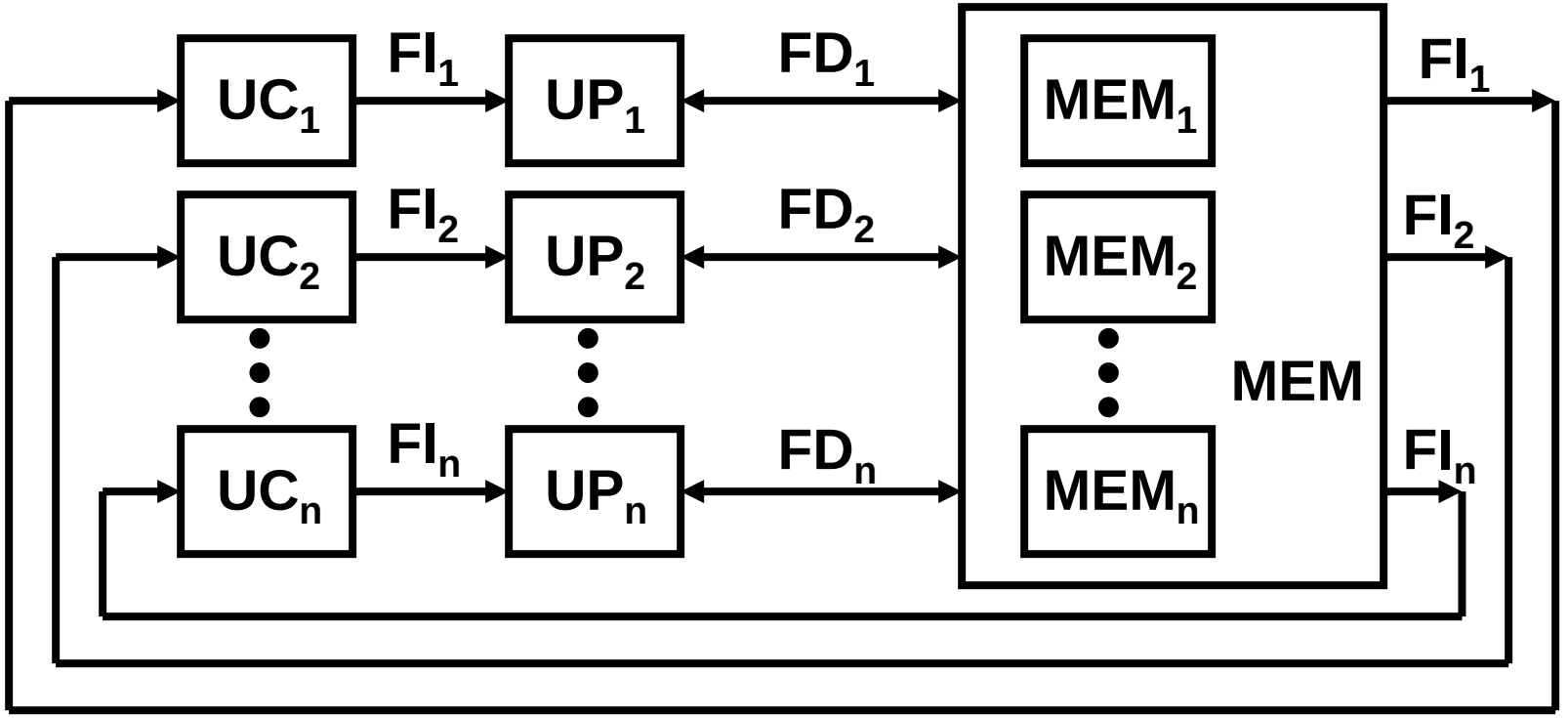
No existen implementaciones ¿es útil?



Clasificación de Flynn : Modelo MIMD

Sistemas MIMD (Multiple Instruction Multiple Data)

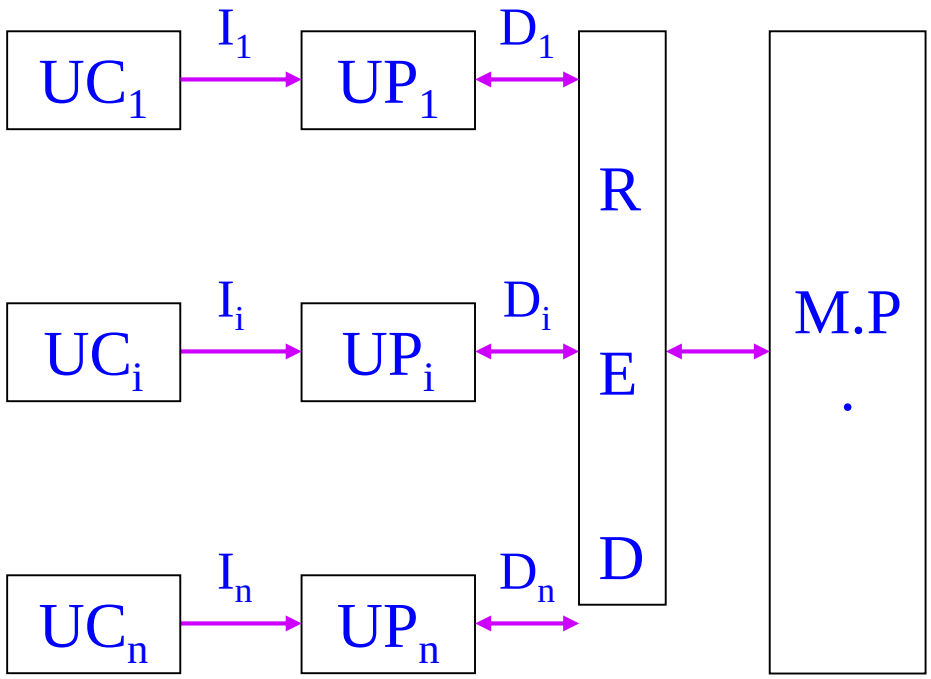
- **Cada instrucción se ejecuta en un procesador distinto y utilizando diferentes datos**
- **Funcionamiento ASINCRONO**



MIMD: Múltiples variantes

Diferencias en el acceso a datos con arquitectura MIMD

M. Común (Multiprocesador)

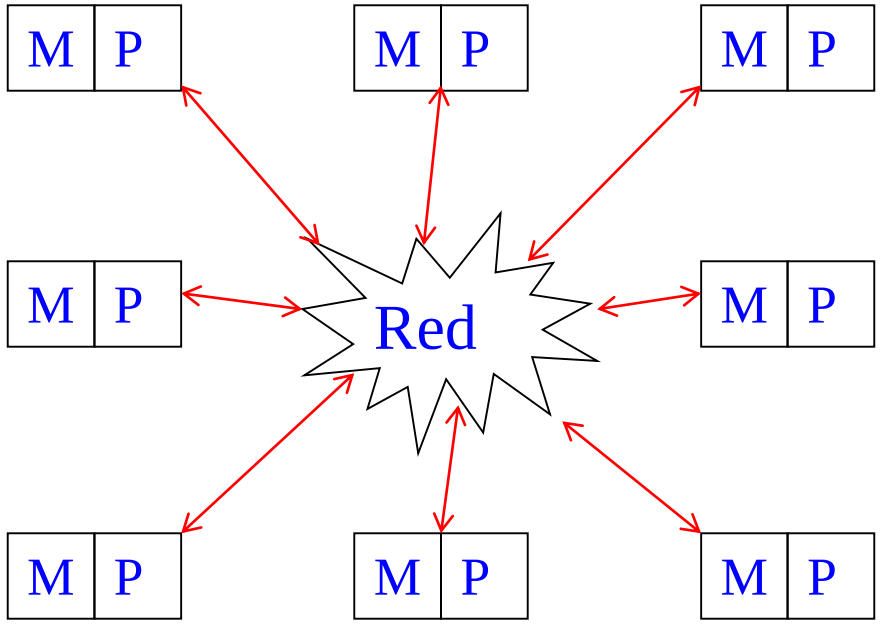


Acoplamiento Fuerte

Memoria común

vs

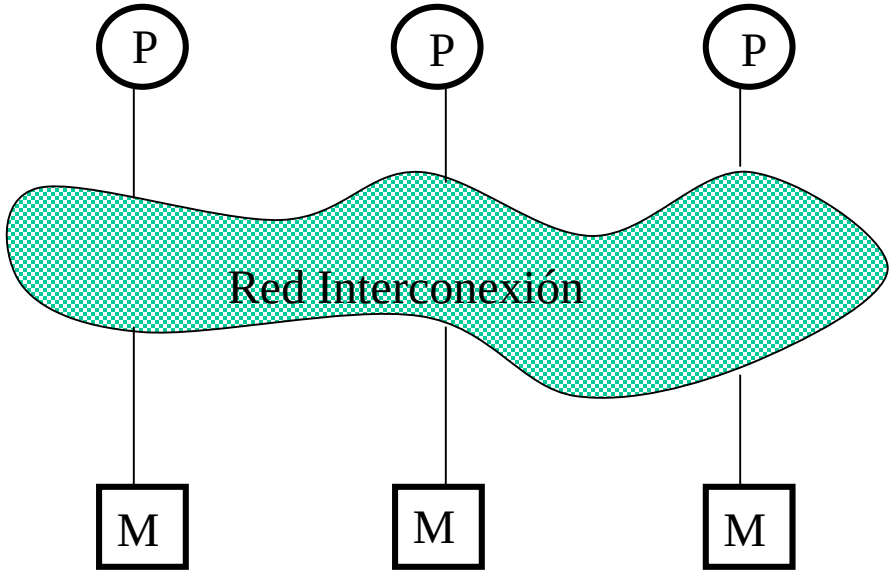
M. Distribuida (Multicomputador)



Acoplamiento Débil

Paso de mensajes

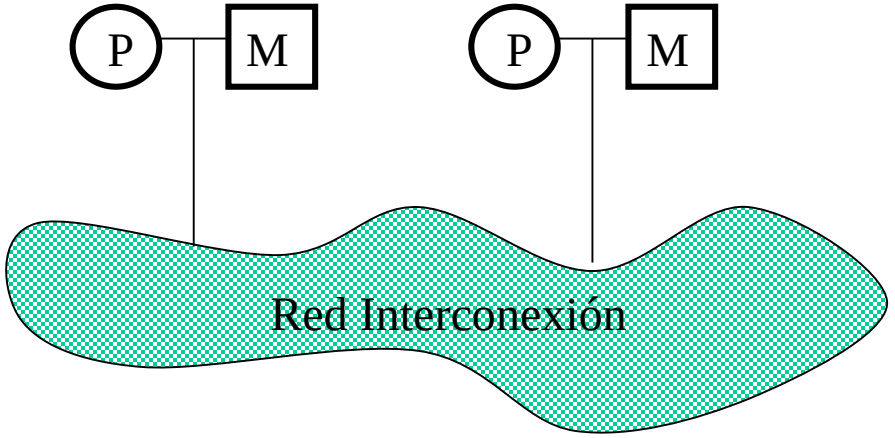
MIMD : acceso a memoria



Basado en la forma en que los procesadores comparten/comunican los programas/datos:

SM:Memoria compartida o memoria global

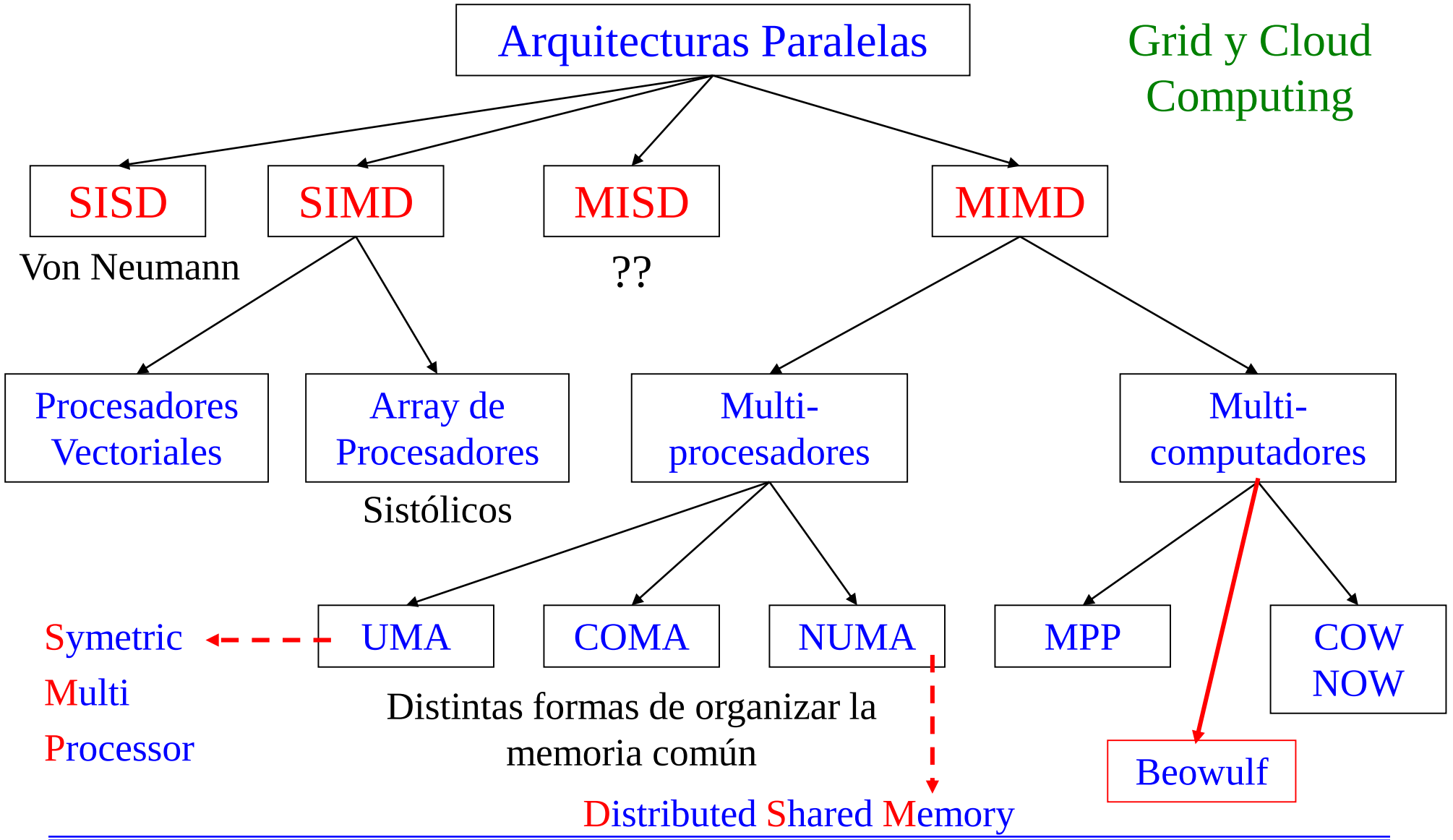
- ✶ Todos los procesadores acceden a los mismos módulos de memoria
- ✶ Red interconexión a medida.



DM: Memoria distribuida o memoria local

- ✶Cada procesador tiene su propia memoria.
- ✶El acceso a los datos en la memoria de otro procesador es mediante el paso de mensajes (o mediante radiación)
- ✶Red de interconexión estándar.

Clasificación de Flynn Ampliada



Arquitecturas Paralelas y sus modelos de programación