

---

# Tema 1

# Arquitecturas MIMD

Nicolás Calvo Cruz

Dpto. de Arquitectura y Tecnología de los Computadores

Telegram: [@nkalvocruz](https://t.me/nkalvocruz)

email: [nkalvocruz@ugr.es](mailto:nkalvocruz@ugr.es)

# Índice

1. Objetivos
2. Motivación
3. Qué es el High Performance Computing (HPC)
4. Un poco de historia
5. Aplicaciones
6. Clasificación de Arquitecturas y Criterios de Clasificación
7. Enfoques
8. Evaluación de Prestaciones
9. Limitación de prestaciones
10. Ley de Amdahl, Ley de Gustafson
11. Mejorando prestaciones

---

# Objetivos

- Conocer arquitecturas paralelas
- Entender los criterios de clasificación
- Repasar enfoques de diseño de arquitecturas paralelas



# Motivación

1 Cada día hay más **necesidades de recursos para computación**, simulación numérica, gráficos, sistemas de predicción y modelado.

2 El conocimiento de las arquitecturas existentes **es una ventaja** a la hora de ganarle la partida a un algoritmo que deseamos acelerar, permitiendo que el software se adapte al hardware de la mejor forma posible.

3 La simulación de grandes sistemas donde los datos son el flujo fundamental para su funcionamiento deben ir **acoplados a la arquitectura**, sacándole así el máximo partido.

4 La simulación numérica permite:

- Disminuir los costes de producción
- Disminuir los costes de construcción de sistemas reales
- Aumentar la productividad disminuyendo el tiempo de desarrollo
- Aumentar la seguridad

---

### 3. Qué es el High Performance Computing (HPC)

Es la rama de la **Informática** que estudia la resolución de **problemas computacionalmente costosos** mediante el **máximo aprovechamiento** de los recursos hardware disponibles.

Permite que la comunidad científica e ingenieril solucionen complejos problemas empresariales, industriales y científicos con la ayuda de aplicaciones que requieren un **gran ancho de banda, una red de baja latencia y altas prestaciones** en el sistema computacional donde se resuelven.



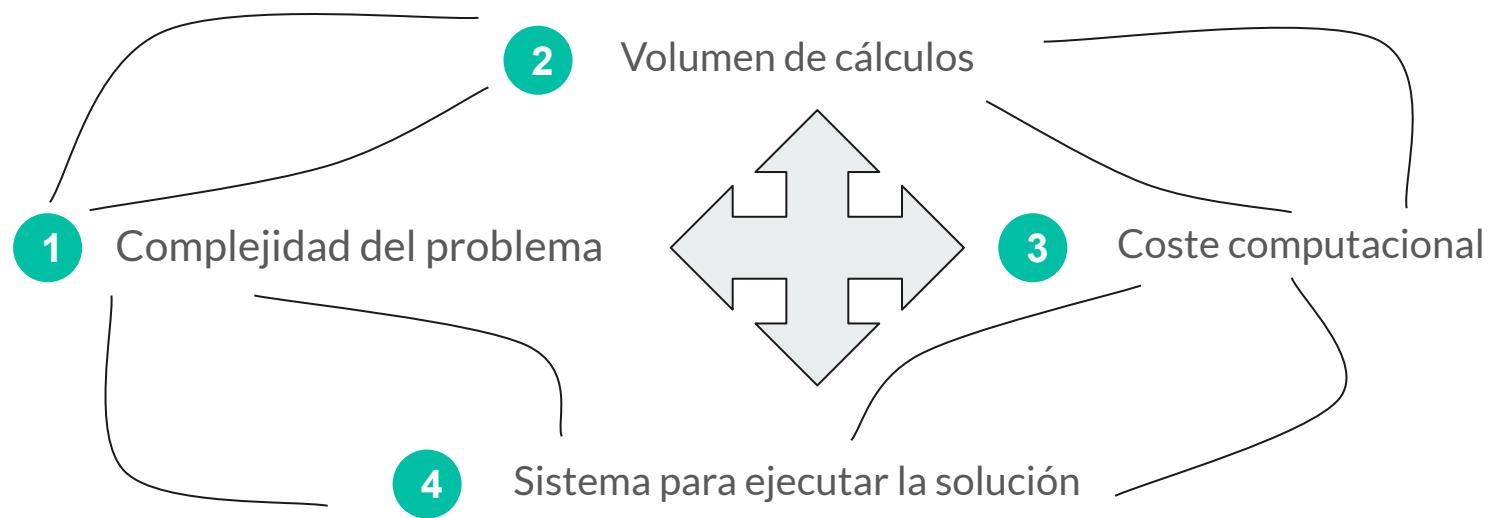
---

## 3.1 Factores que deben estar presentes

1. Problema a resolver de forma computacional
2. Gran volumen de cálculos a realizar
3. Gran coste computacional (“**tiempo**”, ¿**energía**?)
4. Sistema con capacidad de paralelización de tareas y capacidad de cómputo suficiente para resolver el problema



## 3.2 Influencia de los factores



### 3.3 Cosas a tener en cuenta

HPC va más de **FLOPS** que de **tiempos**, pero no siempre incluye operaciones en punto flotante.

HPC habla de procesos, de hebras/hilos, o de ambos simultáneamente, para **aprovechar toda la máquina**.

HPC para las compañías es más bien qué máquina te venden y qué características tiene que se adapten a lo que quieras hacer.

HPC va más de **optimización y adaptación**. La optimización es hacer algo lo mejor posible, y para eso hay que tener una **comparación**, para **saber si has mejorado o no**.

## 4. Un poco de Historia

1. Durante los últimos 50 años, hemos aumentado la **frecuencia** de los relojes y **disminuido el coste**
2. Pero esto **ya no es posible**
3. Crecemos en capacidad a diferentes niveles
  - a. Aumentando más cores on-chip (Ley de Moore) (SMP)
  - b. Aumentando el ancho de las unidades funcionales (SIMD)
  - c. Combinando nodos individuales (MIMD)
  - d. Uniendo otras arquitecturas (GPGPU)



## 4. Un poco de Historia



1. Seymour Cray creó el primer supercomputador, o la primera máquina así llamada, a finales de los 60.
2. El primer supercomputador fue el CDC 6600, que utilizaba una sola CPU con un repertorio de instrucciones RISC y un pipeline capaz de ejecutar una instrucción en cada ciclo.
3. Tenía más prototipos, pero dejó CDC en 1972 y fundó su propia compañía para abandonar los multiprocesadores y dedicarse a los procesadores vectoriales
4. Lideró el mercado durante 5 años, de 1985 a 1990

## 4.1 Necesidades

1. **Compiladores adaptados a las unidades funcionales que tenga la máquina (32, 64... )**
2. **Uso de instrucciones que aprovechen la arquitectura, como las vectoriales**
3. **Eliminación de dependencias de forma automática**
4. **Distribuir de forma semiautomática los accesos a memoria y minimizarlos** (por eso está indicado para aplicaciones con mucho cálculo)
5. **Evitar comunicaciones innecesarias y optimizarlas** porque la latencia de red perjudica al rendimiento



---

## 4.2 Tres niveles ... tres tipos de paralelismo

1. Paralelismo a nivel de comunicación entre los nodos del sistema masivamente paralelo (**Procesos**)
2. Paralelismo a nivel de unidades de ejecución dentro del mismo nodo (**Hebras**)
3. Paralelismo a nivel de datos, utilizando unidades funcionales paralelas para las operaciones que lo permitan (**SIMD**)

## 4.3 La importancia de fijar el esfuerzo

- Si se desea escalar un programa hasta poder ejecutarlo en unos 10.000 procesadores simultáneamente, y estamos en un punto en el que solo se ejecuta en 500, habrá que optimizar el proceso de escalado a nivel de nodo, ya que las comunicaciones entre nodos afectarán mucho, reduciendo la sincronización, el desbalanceo de carga.
- Si la aplicación escala bien, es decir, pasa de ejecutarse en 5 procesadores a ejecutarse en 100 procesadores sin incrementar en exceso el tiempo de comunicación, entonces nuestro esfuerzo debe centrarse en optimizar el paralelismo dentro de los nodos, reduciendo accesos a memoria, quitando dependencias, etc.



## 5. Aplicaciones

- Las **aplicaciones** deben estar especialmente **diseñadas para sacar partido de la naturaleza paralela de los sistemas** que las ejecutan.
- Además, los algoritmos deben permitir extraer esa ventaja, y **adaptarse a una máquina que no tenga esas características**, sin más que volverlo a compilar o ejecutarlo con otros parámetros.
- La elección de cualquier parámetro puede afectar a la ejecución.
- La elección del **lenguaje**, y los **esquemas de comunicación** son muy importantes (Lo veremos más a fondo en el tema 2).

## 5. Aplicaciones

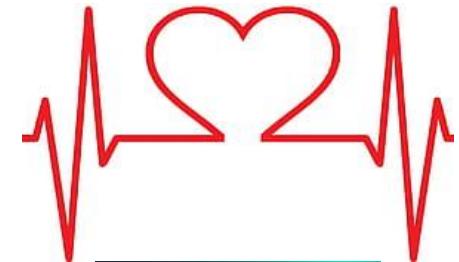
**Salud:** Aunar tecnología y medicina ha permitido, por ejemplo, digitalizar la simulación de procesos y analizar todos sus datos:

1. Doblado y secuenciación de una proteína
2. La secuenciación del material genético humano para detectar posibles enfermedades
3. Simular el efecto de ciertos compuestos
4. ...



### LIVING HEART PROJECT: A CYBER-HEART.

Stanford, California



### TEXAS ADVANCED COMPUTING CENTER: IS CANCER WRITTEN IN OUR DNA?

Austin, Texas

### RADY CHILDREN'S INSTITUTE FOR GENETIC MEDICINE: SWIFT GENETIC TESTING.

San Diego, California

---

**BOEING 787** Seattle, Washintong DC.

## 5. Aplicaciones

**Ingeniería:** La creación de obras de ingeniería es un reto cada vez mayor. Las simulaciones las hacen más seguras, más baratas y acelera su desarrollo.

1. Simulación de aviones
2. Simulaciones de fluidos
3. Simulación de centrales
4. ...



**BICICLETAS TREK.** Waterloo, Wisconsin

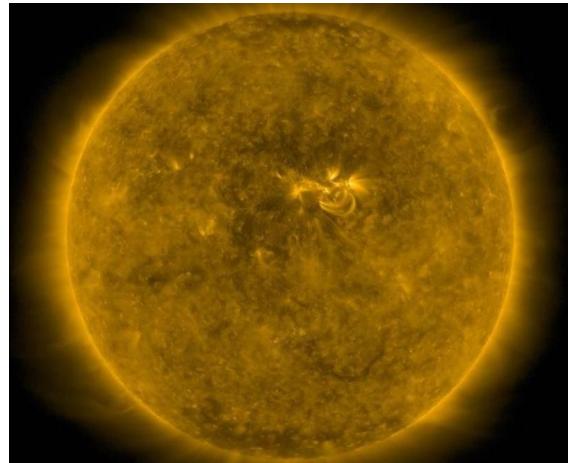


**SUPERTRUCK, OPTIMIZACIÓN DE CONSUMOS EN CAMIONES.**  
EE.UU. Department of Energy.

## 5. Aplicaciones

**Investigación Espacial:** El espacio es algo totalmente desconocido, y se investiga en muchos ámbitos:

1. Vida inteligente fuera de la Tierra (*SETI @ home*)
2. Movimientos de meteoritos
3. De dónde viene el universo
4. ...



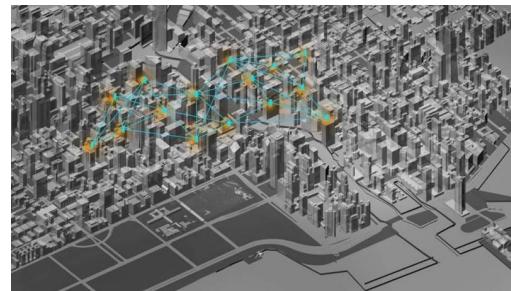
**NASA.** Washington D.C. .

**POLARBEAR, DE DONDE VIENE TODO** Berkeley. California

## 5. Aplicaciones

**Redes de Transporte:** Las Smart Cities son un objetivo que se conseguirá en no muchos años. Una ciudad es una fuente ingente de datos de movilidad, clima, patrones de tráfico, niveles de ruido... lo que permite:

1. Predecir contaminación
2. Coordinación de transportes
3. Diseñar de controles de semáforos
4. ...



**ARRAY OF THINGS**

Chicago, EE.UU



**PREVISIÓN DE HUMO**

Iowa City, EE.UU.



**CONSTRUCCIONES SMART**

Tianjin, China.

## 5. Aplicaciones

### Economía y Negocio

1. Criptomonedas
2. Búsqueda de clientes
3. Predicciones del mercado financiero
4. Procesos de producción más rápidos y eficientes
5. Ubicación de negocios
6. ...

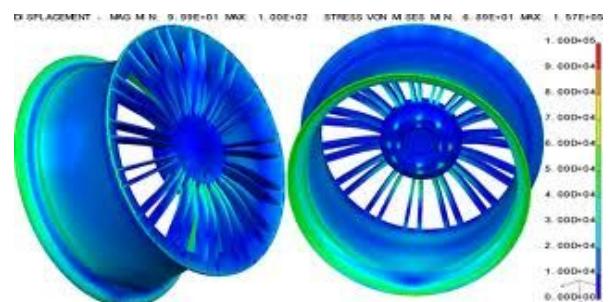
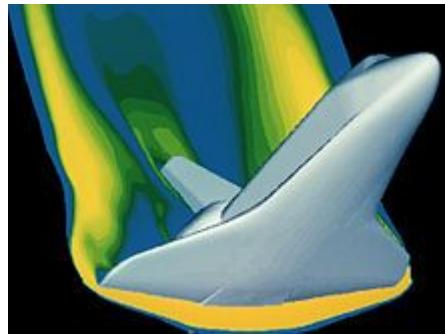
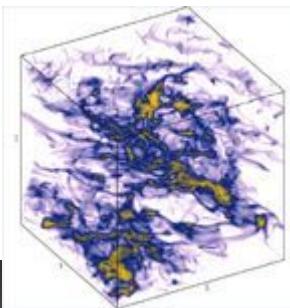
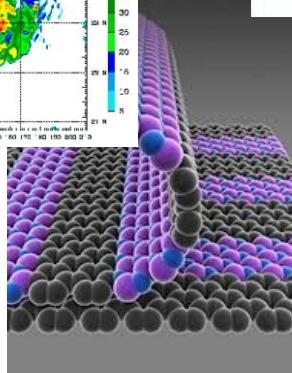
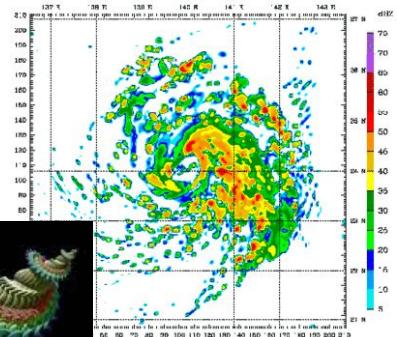
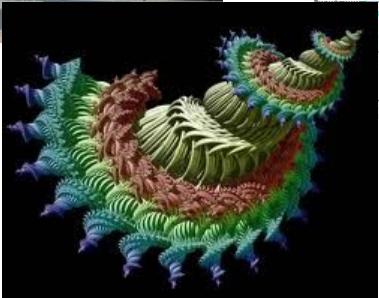


**CRISTALES EFICIENTES Y  
SEGUROS** Tokyo, Japón



**BITCOINS** The Web.

## 5. Aplicaciones



# Índice

1. Objetivos
2. Motivación
3. Qué es el High Performance Computing (HPC)
4. Un poco de historia
5. Aplicaciones
6. Clasificación de Arquitecturas y Criterios de Clasificación
7. Enfoques
8. Evaluación de Prestaciones
9. Limitación de prestaciones
10. Ley de Amdahl, Ley de Gustafson
11. Mejorando prestaciones

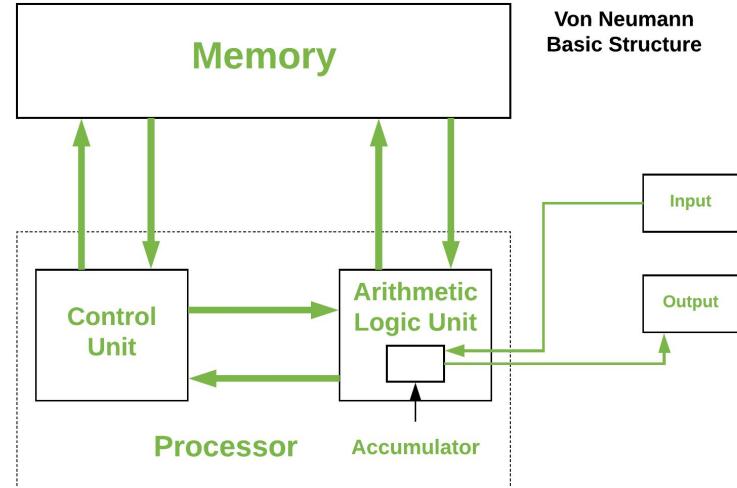
# 6. Clasificación de Arquitecturas



# 6. Clasificación de MIMD

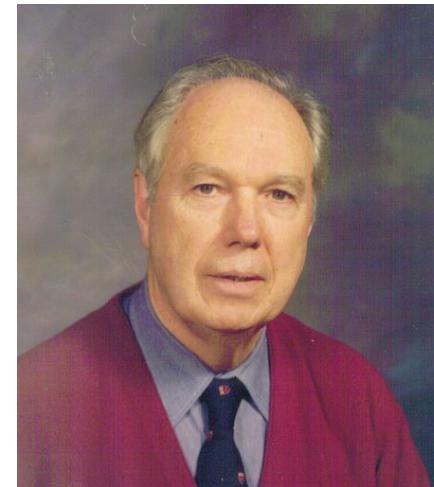
## 6.1 Conceptos previos

- La máquina de von Neumann
- ¿Cómo funciona?
- ¿Cuál es su problema? ¿Cuellos de botella?
- ¿Modos de aumentar su capacidad?
  - Caching, Jerarquía de memoria
  - Memoria Virtual: Para parecer que se hace más de una cosa a la vez sin gastar tiempo en crear procesos completos cada vez
  - Ejecutar más de una instrucción a la vez, pipelining, envío a ejecución de más de una instrucción, multithreading...
  - Aumentar el número de máquinas y coordinarlas



## 6. Clasificación de MIMD

### 6.1 Taxonomía de Flynn

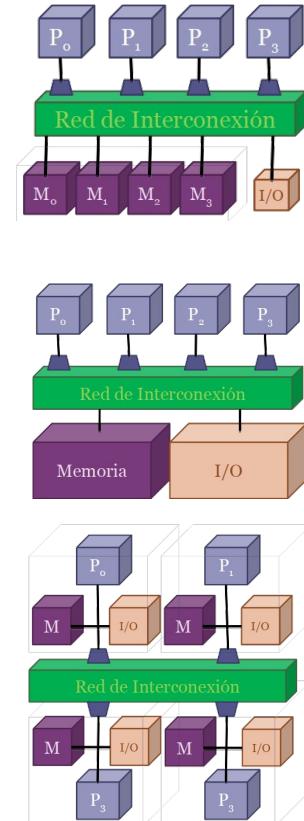
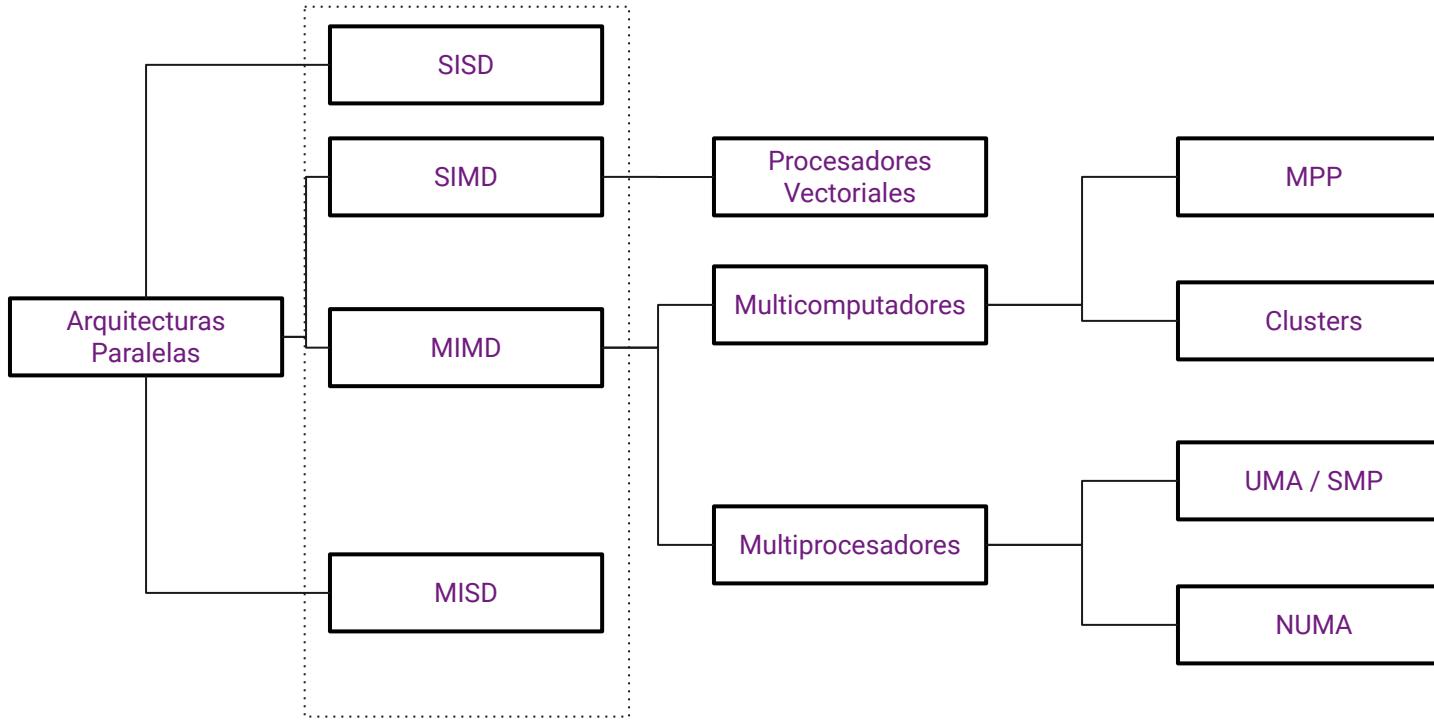


Michael J. Flynn

	Un solo dato usado (SD)	Muchos datos usados (MD)
Una instrucción (SI)	SISD	SIMD
Varias instrucciones (MI)	MISD	MIMD

# 6. Clasificación de MIMD

## 6.2 Según Andrew S. Tanenbaum



## 6. Clasificación de MIMD

### 6.3 MIMD

1. Normalmente son **máquinas asíncronas**: cada unidad de procesamiento tiene su propio reloj
2. No existe una sola arquitectura posible
  - a. Múltiples unidades de ejecución y control y una sola memoria: **multiprocesadores**
  - b. Múltiples unidades de memoria, y múltiples unidades de ejecución y control: **multicomputadores**
  - c. Enfoques Híbridos
3. Actualmente:
  - a. En una máquina para HPC, **se mezclan todos** los enfoques de paralelización de arquitectura. Se tienen multicomputadores donde cada uno de los nodos a su vez es un multiprocesador, y donde la red que une todos los nodos es una red de altas prestaciones.

# 6. Clasificación de MIMD

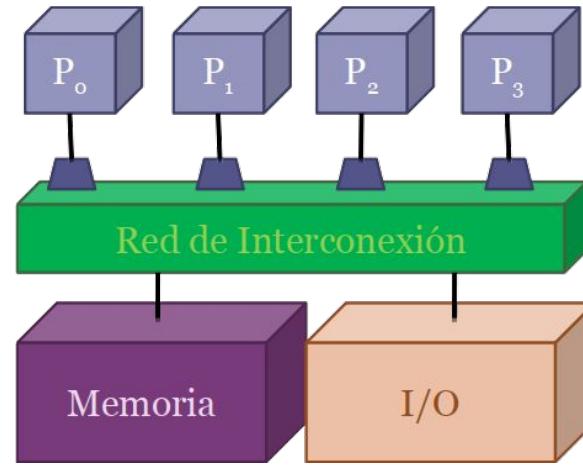
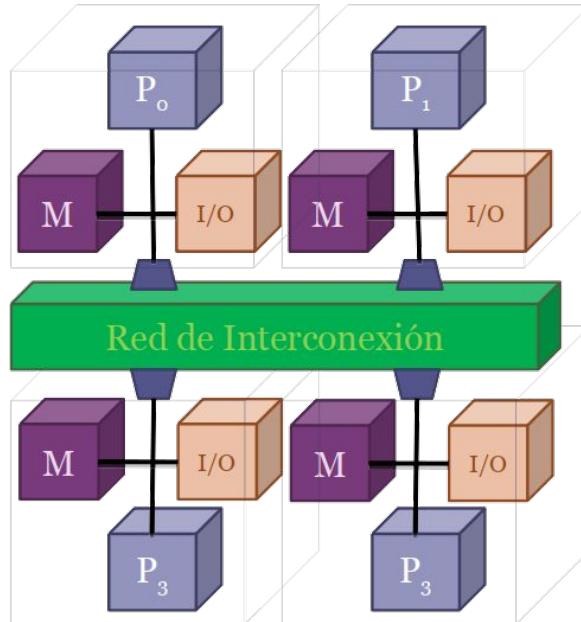
## 6.4 Criterios de clasificación

1. Criterios temporales: máquinas síncronas o asíncronas
2. Criterios de control: máquinas con un control centralizado o distribuido
3. Criterios relacionados con la distribución física de la memoria:
  - a. Memoria centralizada (Multiprocesadores)
  - b. Memoria distribuida (Multicomputadores)
4. Criterios relacionados con el espacio de direcciones de la memoria:
  - a. Único (Shared Memory Systems)
  - b. Múltiple (Distributed Memory Systems)
5. Criterios relacionados con los tiempos de acceso a memoria:
  - a. Tiempo de acceso a memoria no uniforme (NUMA)
  - b. Tiempo de acceso a memoria uniforme (UMA)
6. Criterios según la red de interconexión
  - a. Máquinas con redes estáticas
  - b. Máquinas con redes dinámicas
7. Criterios comerciales, según precio y prestaciones

# 6. Clasificación de MIMD

## 6.4 Criterios de clasificación

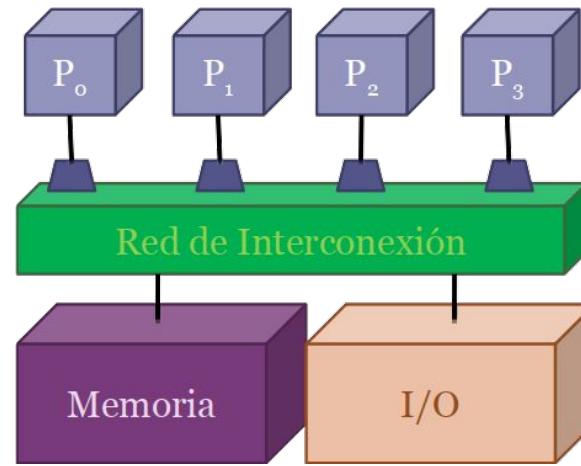
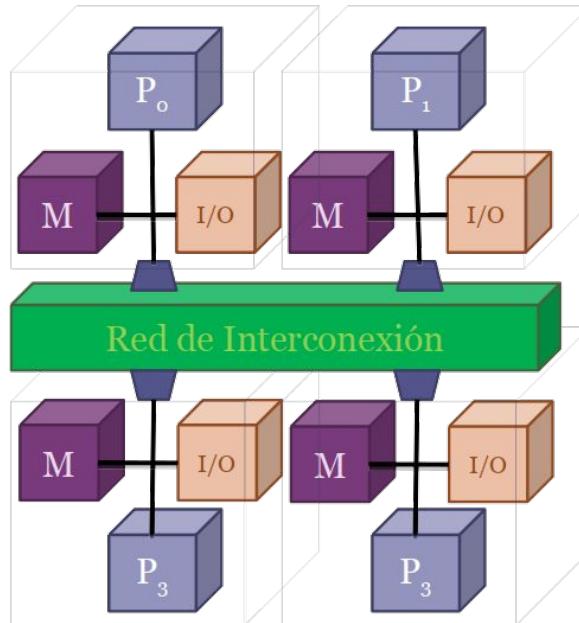
### Distribución de la memoria



## 6. Clasificación de MIMD

### 6.4 Criterios de clasificación

#### Espacio de direcciones, ÚNICO O MÚLTIPLE

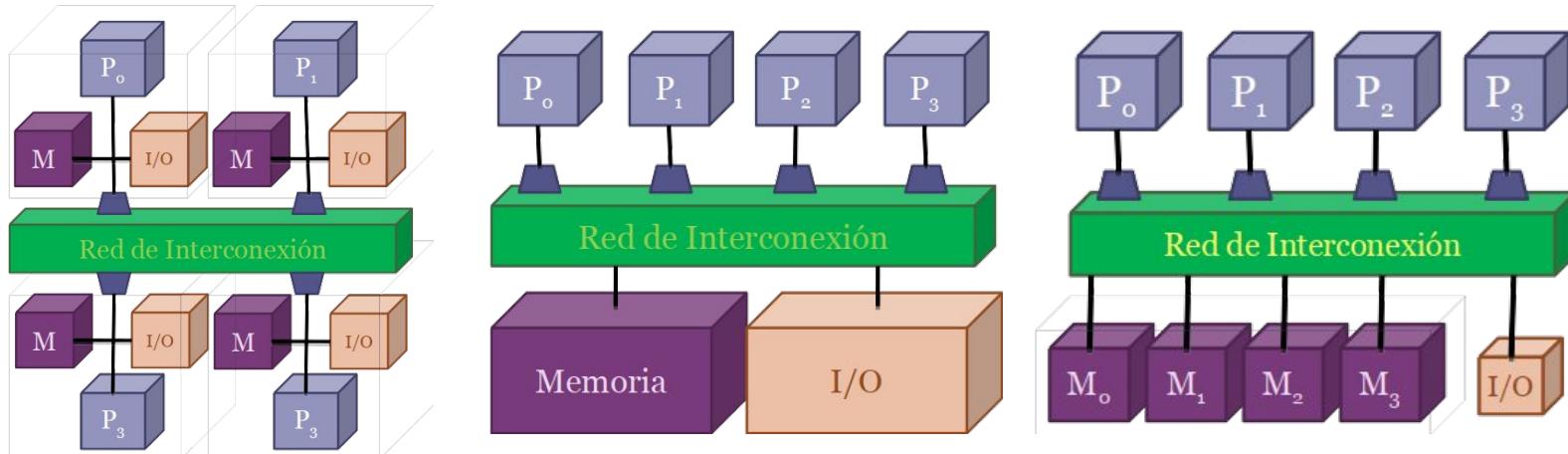


## 6. Clasificación de MIMD

### 6.4 Criterios de clasificación

#### Tiempo de acceso a memoria

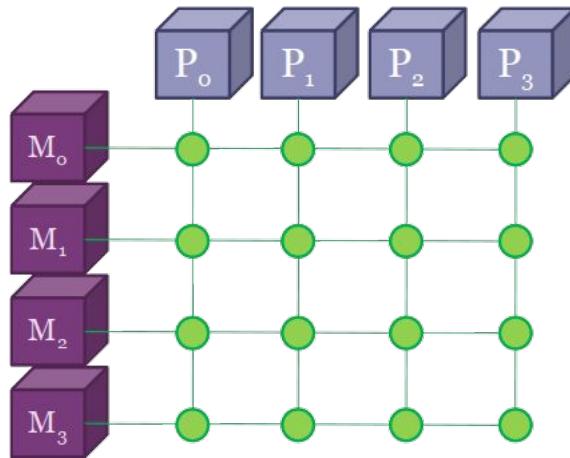
#### NUMA, UMA, NORMA



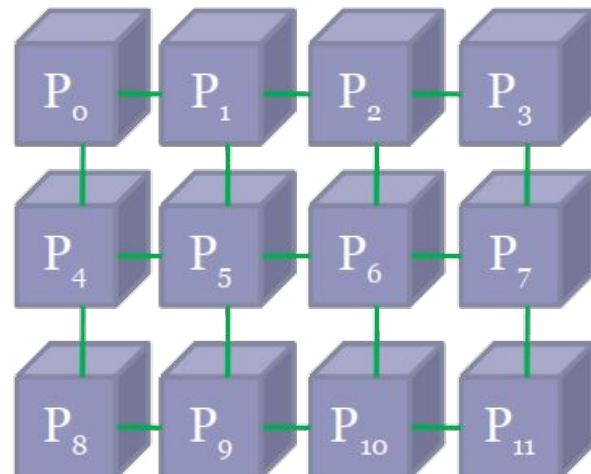
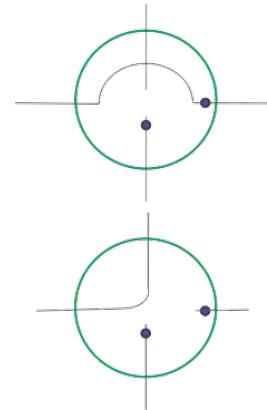
## 6. Clasificación de MIMD

### 6.4 Criterios de clasificación

#### Red de interconexión



Indirecta



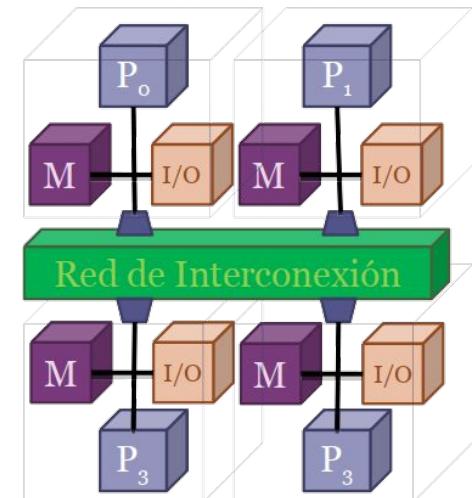
Directa



## 7. Enfoques

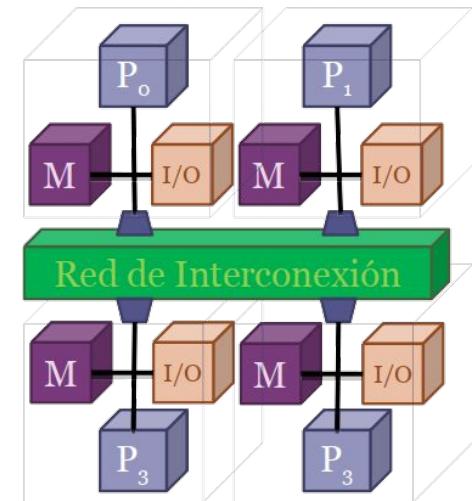
## 7.1 Multicomputadores

- Tiempo de acceso a memoria no uniforme
- Más escalables o ampliables, la limitación es la red de interconexión
- Comunicación y Sincronización por paso de mensajes
- Programación más compleja: el programador debe distribuir el trabajo y los datos además del código
- Normalmente se programan con enfoque de Paso de Mensajes, ya sean síncronos o asíncronos
- Ventajas:
  - Independencia de las partes paralelas, mantenimiento
  - Escalabilidad
  - Ampliación
- Desventajas:
  - Programación
  - Red de interconexión
  - Sincronización



## 7.1 Multicomputadores

- Con un solo sistema operativo y un solo reloj: control centralizado
- Con más de un sistema operativo y varios relojes: control distribuido.
- Hoy en día, casi todos los nodos tienen capacidad de cómputo paralelo también, ya sea por el paralelismo a nivel de instrucción, o a nivel de hebra/funcional.
- Herramientas disponibles:
  - Sistemas operativos específicos
  - Compiladores
  - Lenguajes o librerías que permitan el paso de mensajes



# 7.1 Multicomputadores

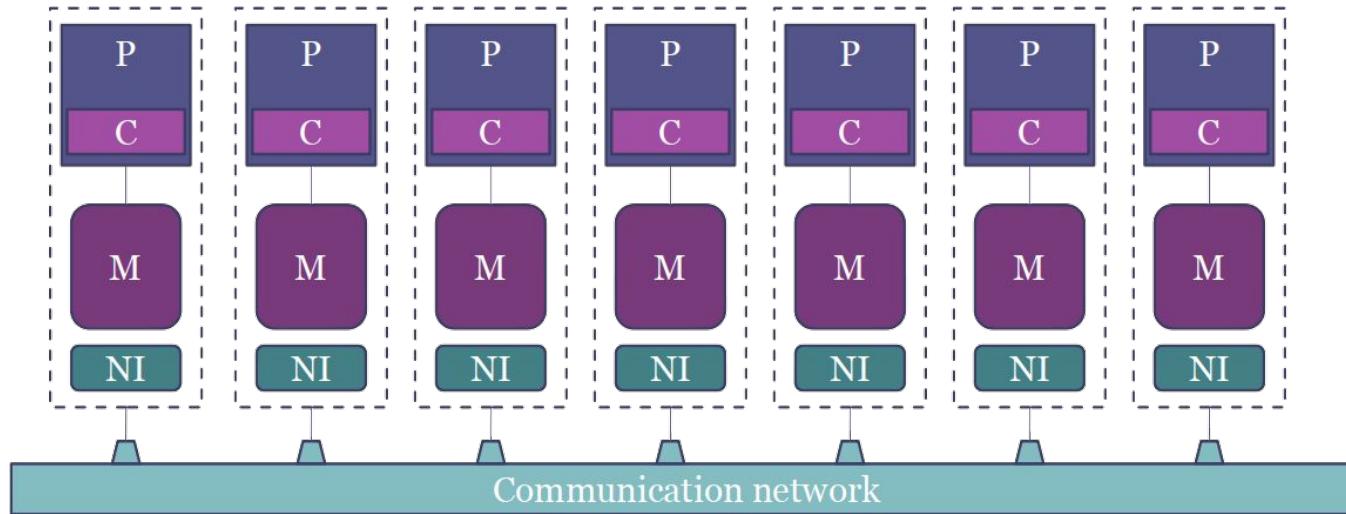
---

## NORMA

- No Remote Memory Access
- Se trata de un multiprocesador con un sistema de direcciones privado para cada unidad de procesamiento
- Los **espacios de memoria no son accesibles entre nodos**
- Su única ventaja es que son **altamente escalables**, más que CC-NUMA o COMA, ya que no tiene que mantener la coherencia de memoria en el sistema completo
- Su **programación es extremadamente complicada**, ya que hay que partir los datos en memorias locales y mantener la consistencia mediante la programación

## 7.1 Multicomputadores

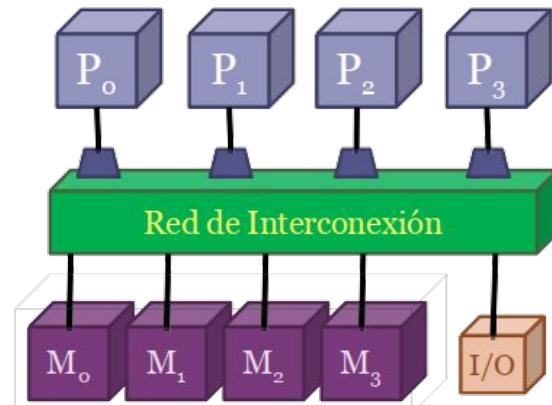
### Estructura simplificada NORMA



Visión simplificada de un sistema de memoria distribuida en la que cada nodo de procesamiento tiene un solo nivel de caché, y un solo módulo de memoria. La comunicación se realiza a través de una interfaz de red y la conexión es directa a la red de comunicaciones. Ningún procesador puede acceder a otro módulo de memoria que no sea el suyo, por eso se denominan No Remote Memory Access System (NORMA)

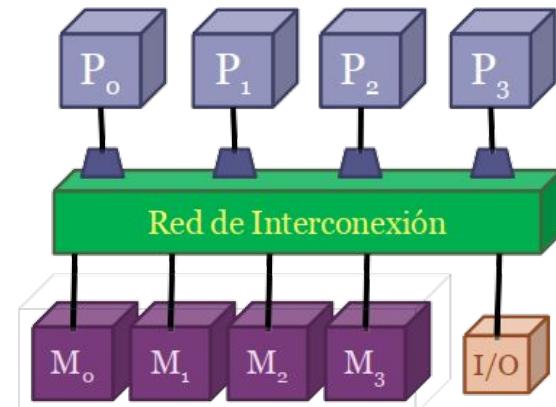
## 7.2 Multiprocesadores

- Espacio de memoria uniforme
- Aumenta la latencia por la red de interconexión que se sitúa entre las unidades de procesamiento y la memoria
- La sincronización es por hardware usando primitivas de sincronización
- Programación más sencilla
- Herramientas disponibles:
  - Compiladores
  - Lenguajes con capacidad de manejo de hebras
  - ...
- No se distribuye ni código ni datos
- Comunicación a través de variables compartidas



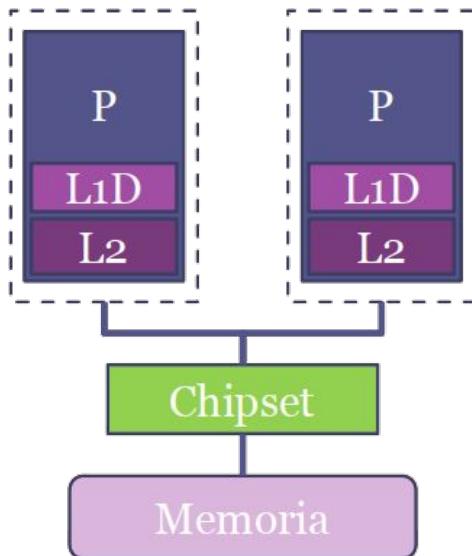
## 7.2 Multiprocesadores

- Clasificación:
  - UMA
    - SMP
    - ¿COMPUTACIÓN HETEROGÉNEA?
  - NUMA
    - CC-NUMA: Cache Coherent- Non Uniform Memory Access
    - Non CC-NUMA: Non Cache Coherent- Non Uniform Memory Access
    - COMA: Cache Only Memory Access

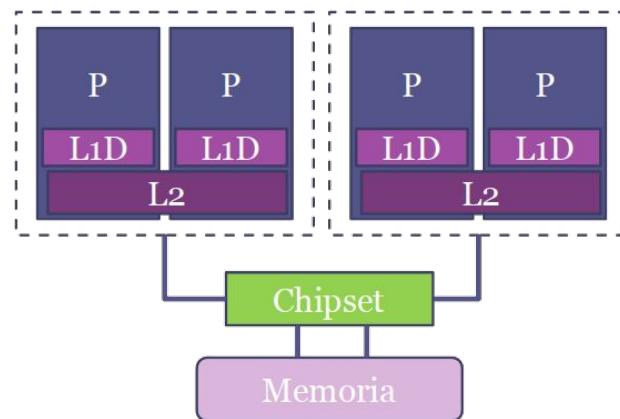


## 7.2 Multiprocesadores

### Esquemas simplificados comunes



Sistema UMA con dos procesadores single-core que comparten un bus (frontside bus -FSB-)

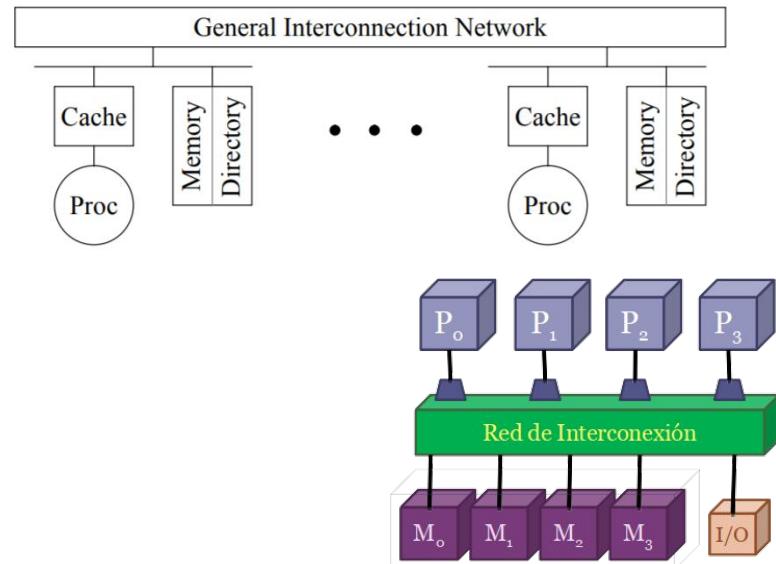


Sistema UMA con dos procesadores dual-core que no comparten la conexión al bus (frontside bus -FSB-)

## 7.2 Multiprocesadores

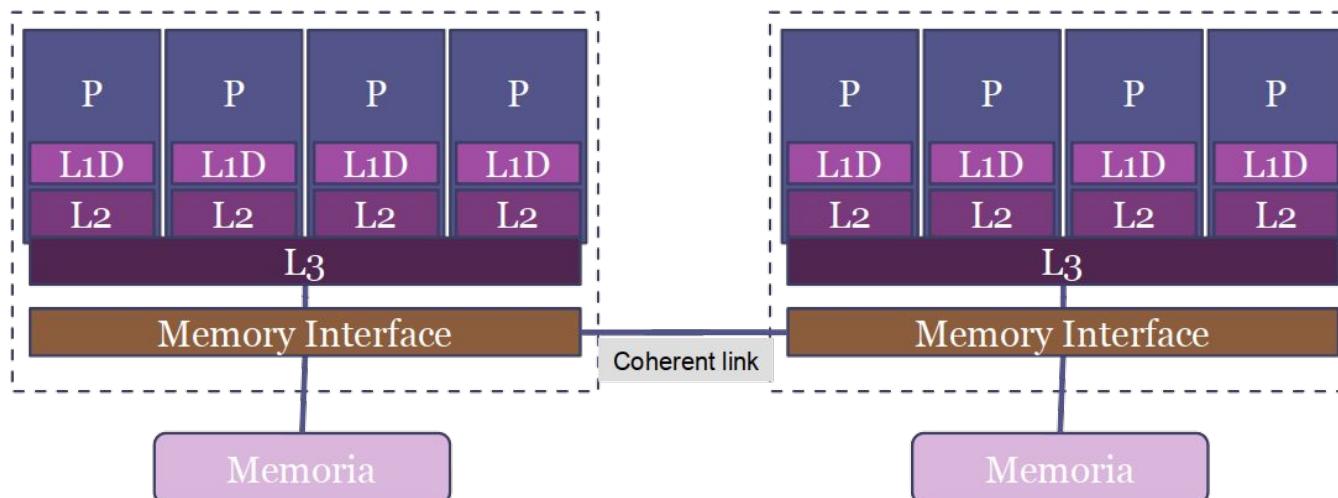
### CC-NUMA

- Cada procesador tiene asociado una cache y una porción propia de la memoria compartida
- La coherencia de memoria se realiza con protocolos basados en directorios
- El acceso a cada línea de memoria se produce mediante hardware
- El soporte de acceso lo proporciona el sistema operativo normalmente



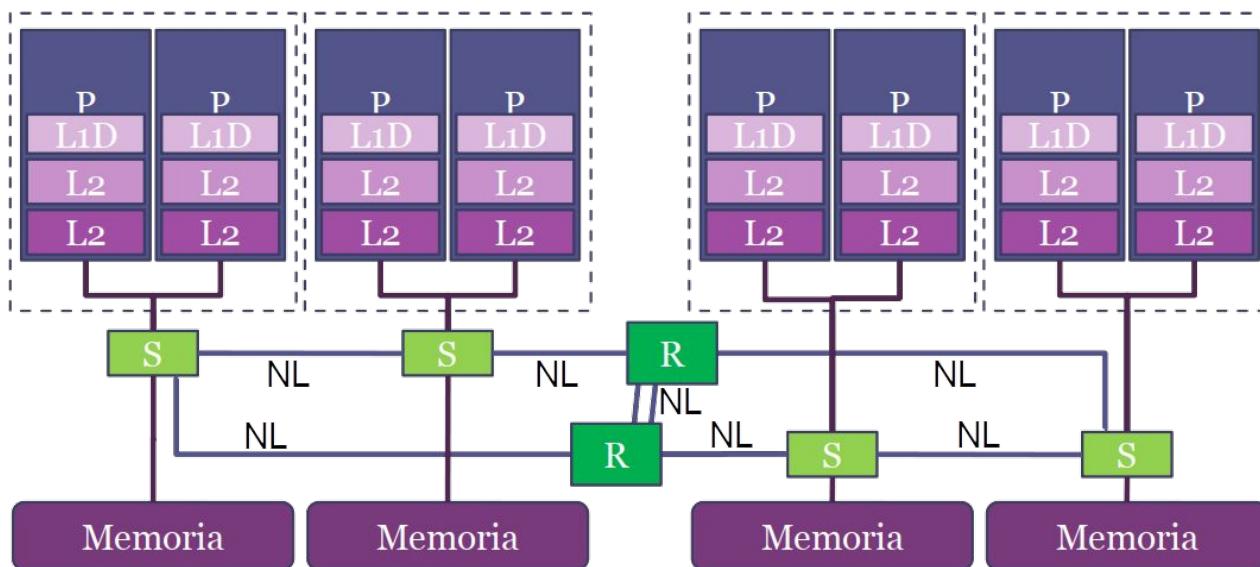
## 7.2 Multiprocesadores

### CC-NUMA



Sistema ccNUMA con dos dominios de localidad (uno por cada socket) y 4 cores cada uno de los dominios

## 7.2 Multiprocesadores CC-NUMA

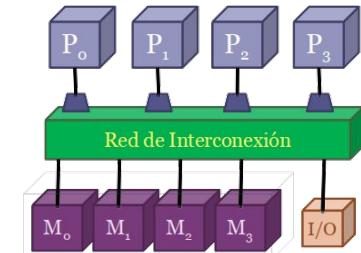


Sistema ccNUMA de SGI Altix con cuatro dominios de localidad, cada uno con un socket (S) y un procesador dual-core en cada dominio. Las conexiones del sistema de memoria son NUMALink (NL) y utilizando dos routers (R). Se usan los accesos a NL solo cuando el acceso a memoria de cada dual-core no es local

## 7.2 Multiprocesadores

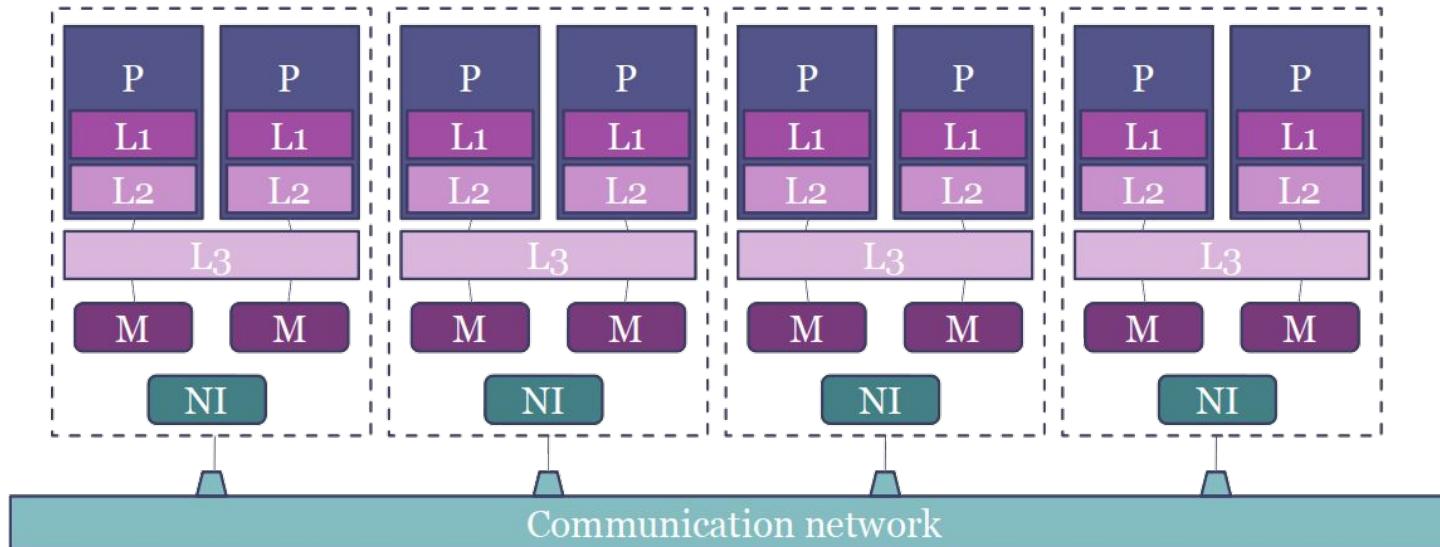
### COMA

- Se trata de una máquina CC-NUMA, pero toda la memoria compartida se trata como una **cache gigante**.
- Al gestionar toda la memoria como una caché , cada línea tiene un contenido y un estado
- Cuando un procesador necesita una línea, se busca y se replica en la porción de memoria compartida local de ese proceso (si no estaba ya).
- Esta arquitectura incrementa las posibilidades de encontrar datos en la zona de memoria local de cada procesador
- La jerarquía de memoria es capaz de gestionar y evitar largas latencias de acceso a memoria mediante adelantamientos de lecturas.



## 7.3 Multiprocesadores y Multicomputadores

### Estructura híbrida simplificada



Sistema híbrido con nodos de memoria compartida tipo CC-NUMA. Se mantienen dos sistemas de comunicación, uno a través de los NI y otro a través del sistema de memoria caché de nivel 3. El modelo de programación para estos sistemas incluyen paso de mensajes y memoria compartida.



---

Gracias.



---

# Trabajos Primera Semana

Grupo 1: BOINC

Grupo 2: BITCOINS Y HPC

Grupo 3: PROFILING Y TRACING, DIFERENCIAS Y SIMILITUDES

Grupo 4: HERRAMIENTAS PARA HPC CON EJEMPLOS CONCRETOS

Grupo 5: CRITERIOS DE CLASIFICACIÓN Y RANKING DE MÁQUINAS DEL TOP 500

Grupo 6: CRITERIOS DE CLASIFICACIÓN Y RANKING DE MÁQUINAS DEL GREEN 500

Grupo 7: SOLUCIONES COMERCIALES PARA HPC DE LOS PRINCIPALES FABRICANTES

