ARQUITECTURA DE MEMORIA COMPARTIDA

- INTRODUCCIÓN A ARQUITECTURAS MULTIPROCESADOR
- ARQUITECTURA DE MEMORIA COMPARTIDA CENTRALIZADA
- ALTERNATIVAS DE COHERENCIA DE CACHE
- PROTOCOLOS DE ESPIONAJE
- RESUMEN

Multiprocesadores

Los multiprocesadores son computadores que tienen memoria compartida y se pueden clasificar en UMA (*uniform memory access*), NUMA (*nonuniform memory access*) y COMA (*cache only memory access*).

Los nombres que reciben las arquitecturas multiprocesador UMA y NUMA tienen que ver con el tiempo de acceso a la memoria principal, no teniéndose en cuenta la diferencia de tiempo entre un acierto o un fallo en caché. De lo contrario, también deberíamos considerar arquitectura NUMA a cualquier sistema con jerarquía de memoria, incluidos los uniprocesadores.

En cuanto a la arquitectura COMA, que no tuvo mucho éxito, se basa en tener la memoria compartida como si fuera una gran caché.

UMA

En este tipo de arquitectura, como bien dice su nombre, todos los accesos a memoria tardan el mismo tiempo. Seguramente podemos pensar que es difícil que tengamos el mismo tiempo de acceso si la memoria, aunque compartida, está dividida en módulos a los que se accede a través de una red de interconexión basada en *switches*. Eso es cierto y, para conseguir esta uniformidad de acceso, se tiene que aumentar el tiempo de los accesos más rápidos.

Switch

Es un dispositivo que interconecta dos o más segmentos de red (*links*), pasando datos de un segmento a otro según la configuración de conexiones y la dirección que deba tomar el dato a transferir.

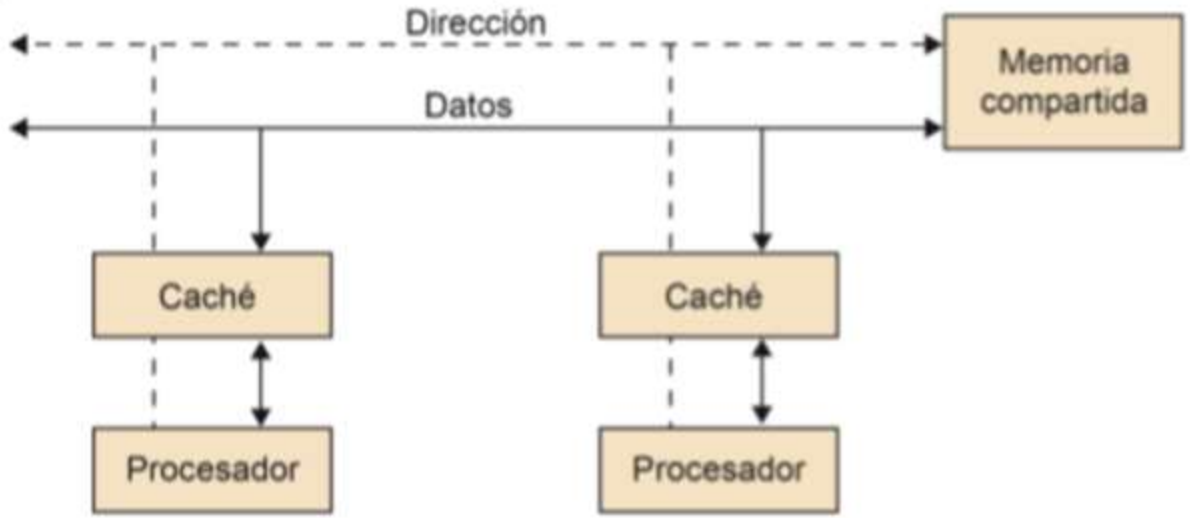
¿Por qué se busca esta uniformidad de tiempo de acceso? Porque para los programadores es más fácil deducir qué parámetros ayudarán a mejorar sus programas si el tiempo de acceso es igual para cualquier acceso. De lo contrario, como pasa con las arquitecturas NUMA , deberían ser mucho más conscientes de la arquitectura que tiene el computador y de cómo están distribuidos los datos en la arquitectura en cuestión.

La figura muestra un sistema multiprocesador donde los procesadores se conectan a la memoria compartida utilizando un bus (dato y direcciones). Este tipo de máquinas son fáciles de construir. Sin embargo, el hecho de que todos los procesadores accedan al mismo bus para acceder a memoria puede significar un cuello de botella. Una alternativa a tener un único bus sería tener varios buses, de tal forma que se distribuirían los accesos a los módulos de memoria. También se podría usar una red de interconexión más compleja, como una red *crossbar*, que permita evitar conflictos entre los accesos a las memorias que no van al mismo módulo.

Conexiones mediante bus

Las conexiones a través de un bus son fáciles de construir pero pueden tener mayor contención de acceso a memoria que otras redes de interconexión.

Multiprocesador de memoria compartida a través de un bus.



Por otra parte, una forma de reducir la contención de los accesos a memoria* es reduciendo la cantidad de estos accesos por parte de los procesadores, y se puede conseguir incorporando jerarquías de memoria en los mismos. De esta forma, si los programas explotan la localidad de datos, los accesos a memoria se reducirán.

*** La jerarquía de memoria ayuda a reducir la contención de acceso a memoria si se explota la localidad de datos.**

Las máquinas UMA se hicieron más y más populares desde aproximadamente el año 2000, año en el que podríamos decir que hubo un punto de inflexión en la tendencia de los diseños de las arquitecturas de los uniprocesadores, debido básicamente al consumo energético y a la consecuente bajada de rendimiento en las aplicaciones. La aparición de los procesadores CMP o *chip multi-processors* fueron el resultado natural de intentar aprovechar las mejoras tecnológicas existentes, y aprovechar así mejor el área del chip, sin aumentar considerablemente el consumo energético.

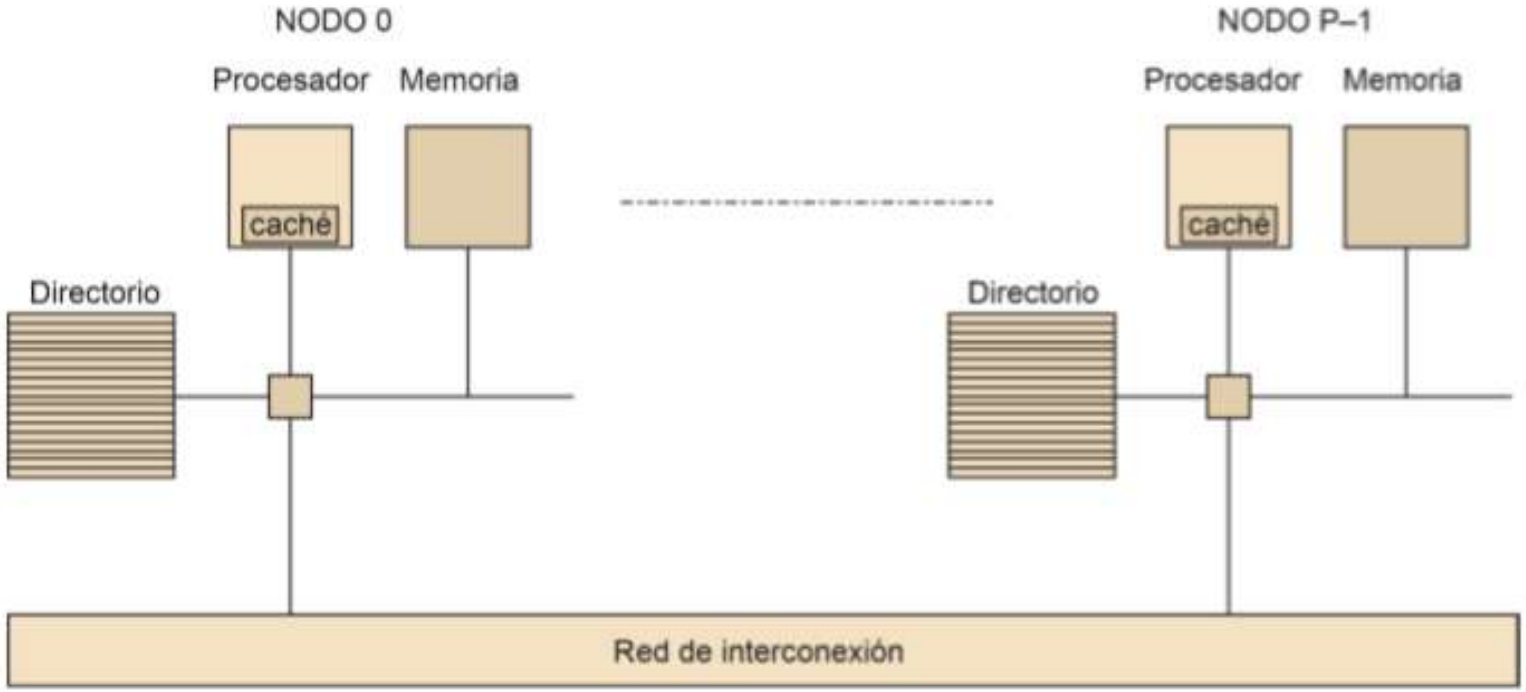
Los procesadores CMP son procesadores de tipo UMA que incorporan más de una unidad de procesamiento dentro de un único chip.

NUMA

En los multiprocesadores NUMA, a diferencia de los UMA, los accesos a memoria pueden tener tiempos distintos. En estas máquinas la memoria también está compartida, pero los módulos de memoria están distribuidos entre los diferentes procesadores con el objetivo de reducir la contención de acceso a memoria.

El módulo de memoria que está junto a un procesador en un mismo nodo recibe el nombre de memoria local a este procesador. Así, los accesos de un procesador a su memoria local suelen ser mucho más rápidos que los accesos a la memoria local de otro procesador (memoria remota). La figura muestra un esquema básico de la distribución de memoria en este tipo de multiprocesadores, donde se observa la distribución de los módulos entre los diferentes nodos. Estos nodos tienen un bus local para acceder a la memoria local, y se conectan a la red de interconexión para acceder a la memoria situada en otros nodos. También observamos un *hardware* llamado directorio que sirve para mantener la coherencia de los datos en el módulo. La redes de interconexión típicas de estos sistemas son las redes de tipo *tree* y las de bus jerárquico.

Multiprocesador NUMA de memoria compartida.



Desde el punto de vista del programador, éste deberá ser consciente de la arquitectura NUMA para poder acercar los datos a la memoria local del procesador que tenga que operar con esos datos. De esta forma se podrán reducir los accesos a memorias locales de otros procesadores (memorias remotas) y, como consecuencia, se mejorará el tiempo de ejecución de la aplicación.

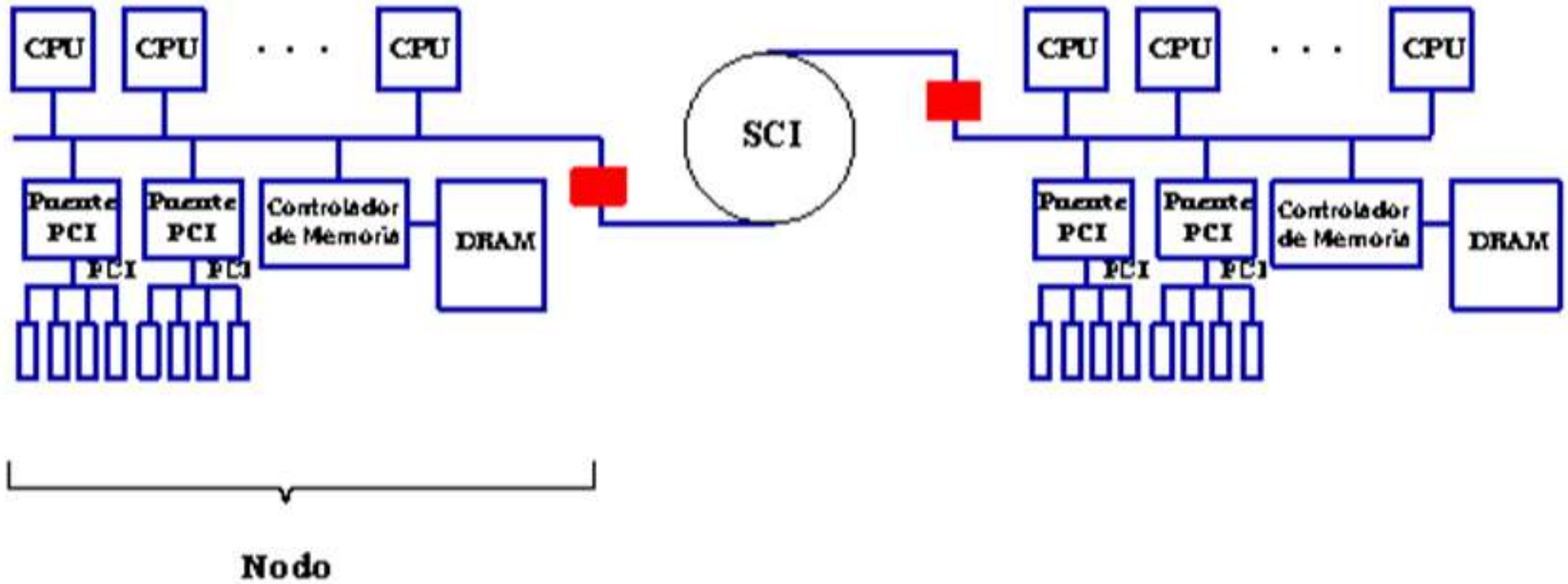
Mapeo de datos

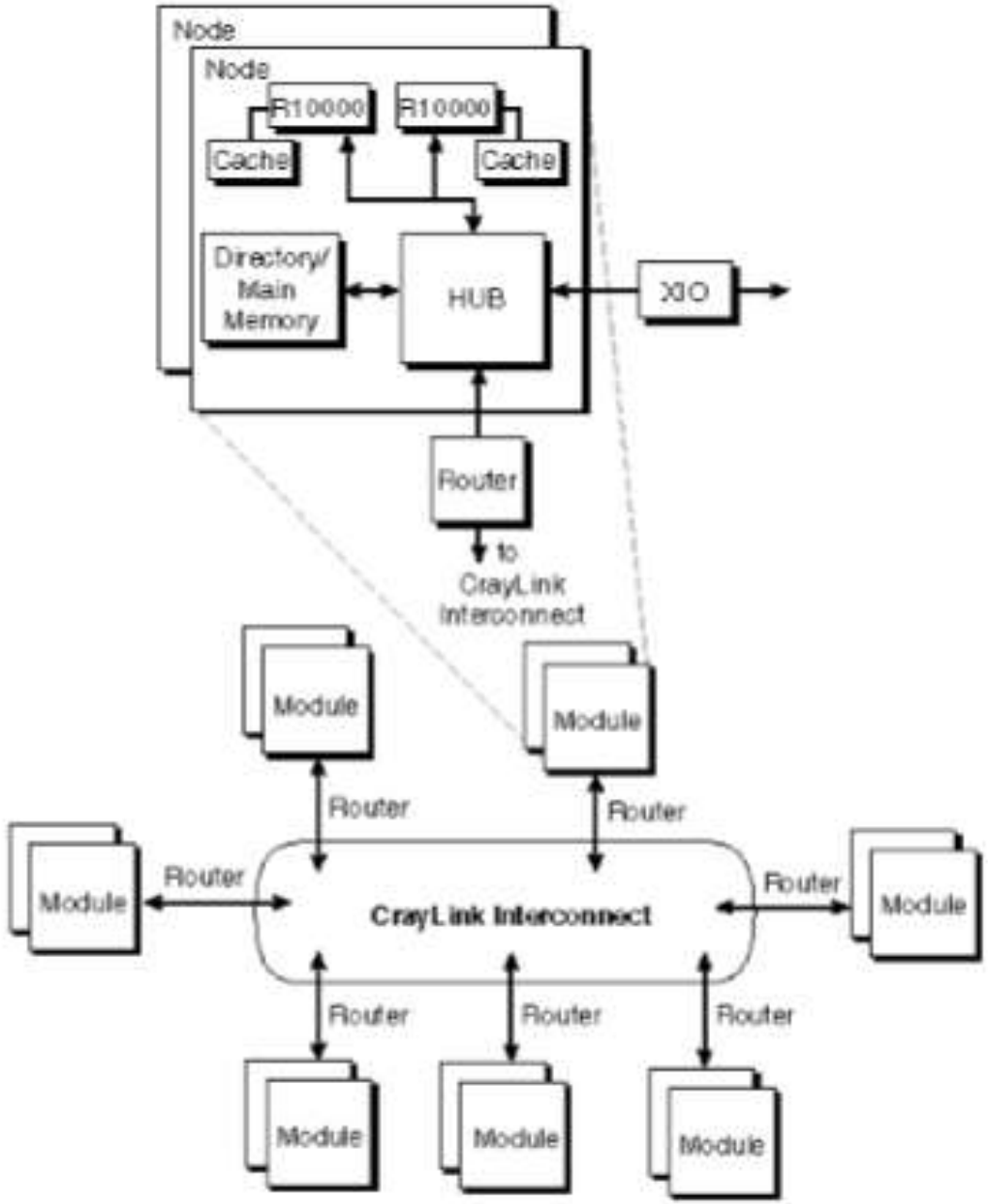
El programador deberá ser consciente del mapeo de los datos para reducir el número de accesos a la memoria local de otros procesadores.

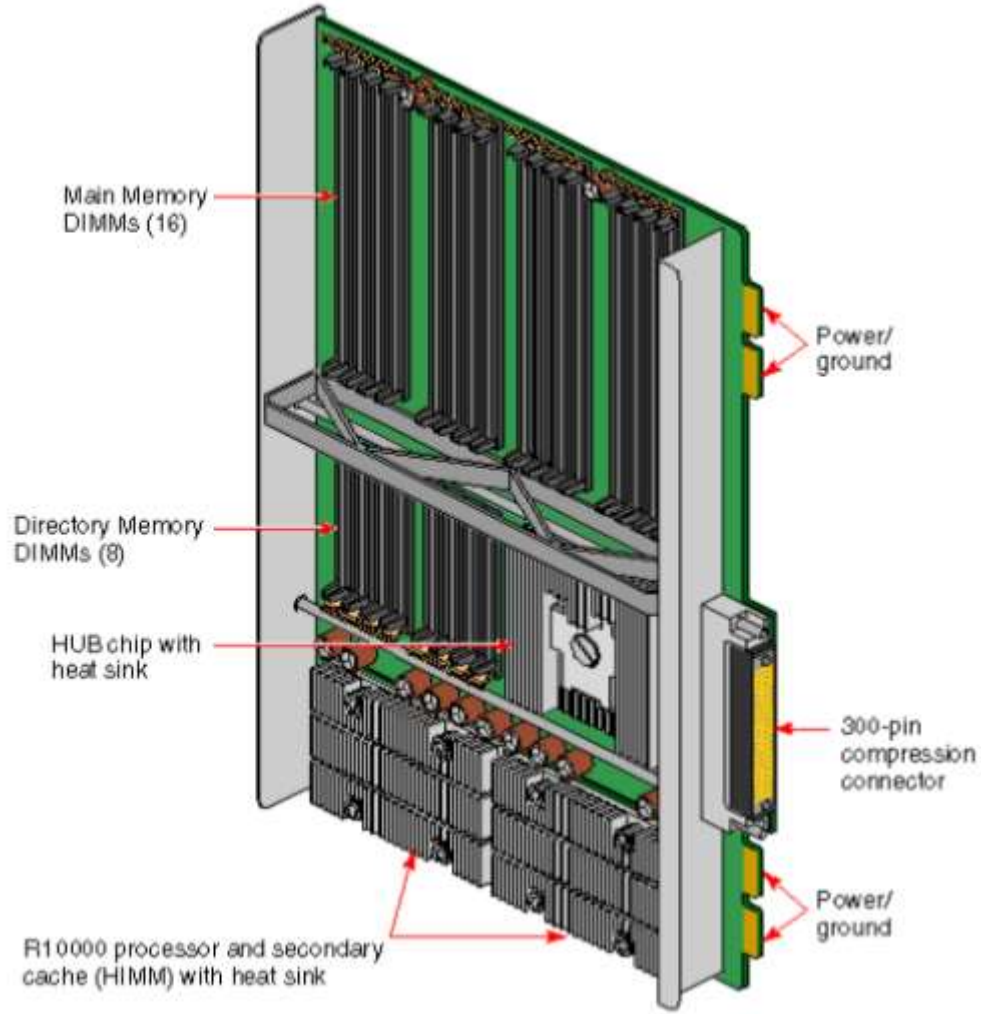
Por otra parte, al igual que pasaba con las arquitecturas UMA, también se incorporaron las cachés para reducir la contención de memoria. Pero además, con estas cachés también se intenta ocultar la diferencia de tiempo de acceso a memoria entre memoria local y remota. Según tengan o no caché, los multiprocesadores NUMA se clasifican en *cache coherence* NUMA (ccNUMA) o *Non-Coherence* NUMA respectivamente. El término de coherencia o no coherencia proviene de mantenerla o no en los datos duplicados en las cachés.

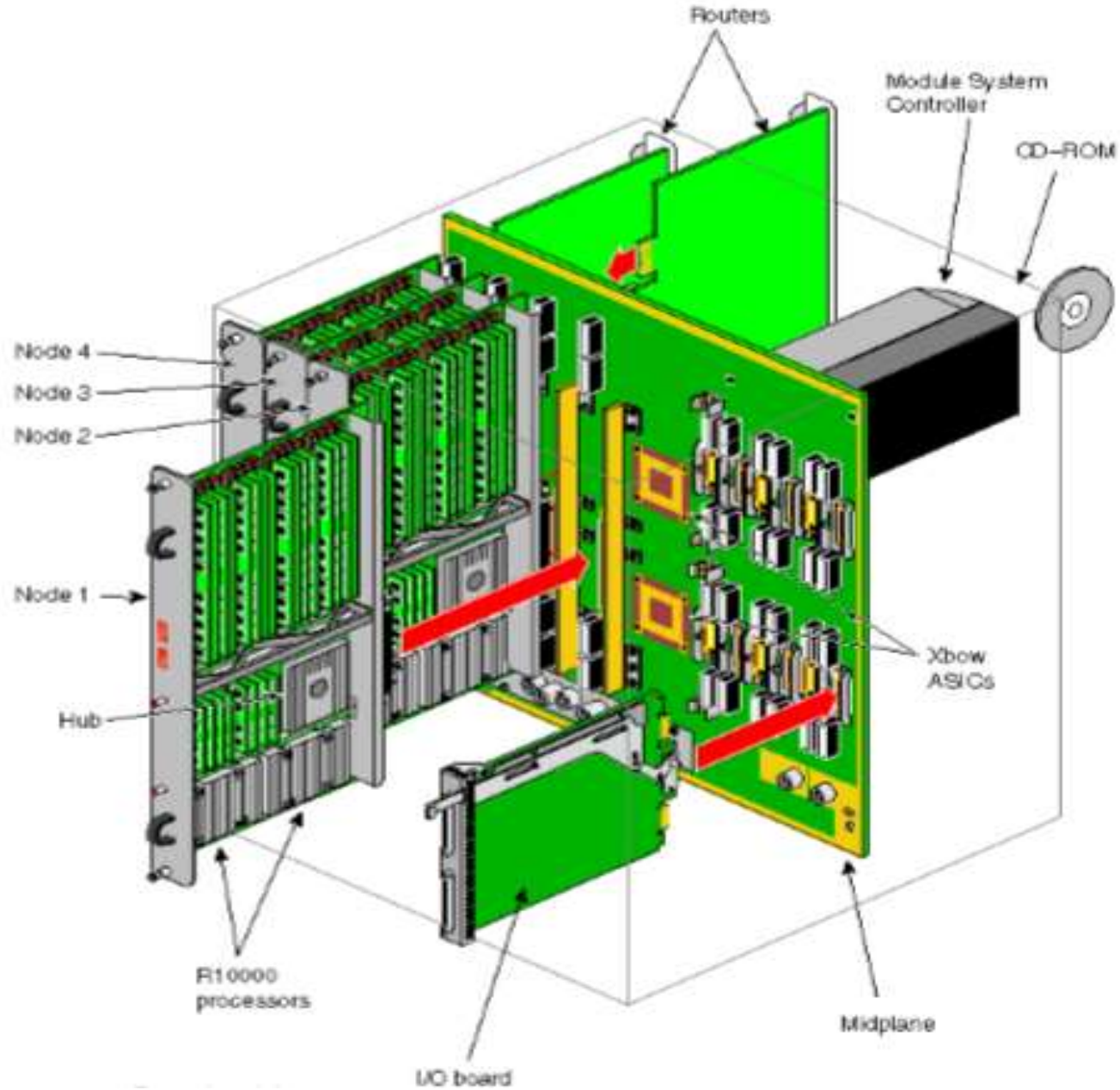
Coherencia de los datos

Mantener la coherencia de los datos significa que el último valor de un dato debe ser visto por todos los procesadores.


CPU
CPU
CPU
SCI
CPU
CPU
CPU
Puente PCI
Puente PCI
Controlador de Memoria
DRAM
PCI
PCI
Puente PCI
Puente PCI
Controlador de Memoria
DRAM
PCI
PCI
Nodo


Node
Node
R10000
R10000
Cache
Cache
Directory/
Main
Memory
HUB
XIO
Router
to
CrayLink
Interconnect
Module
Module
Router
Router
Module
Router
CrayLink Interconnect
Router
Module
Router
Router
Router
Module
Module
Module


Main Memory
DIMMs (16)
Power/
ground
Directory Memory
DIMMs (8)
HUB chip with
heat sink
300-pin
compression
connector
Power/
ground
R10000 processor and secondary
cache (HIMM) with heat sink


Routers
Module System Controller
CD-ROM
Node 4
Node 3
Node 2
Node 1
Hub
Xbow ASICs
R10000 processors
I/O board
Midplane

COMA

Aunque las arquitecturas ccNUMA ayudan a ocultar la diferencia de tiempo entre accesos locales y remotos, esto dependenderá de la localidad de datos que se explote y la capacidad de las cachés. Así, en el caso de que el volumen de los datos a los que se accede sea mayor que la capacidad de la caché, o el patrón de acceso no ayude, volveremos a tener fallos de caché y el rendimiento de las aplicaciones no será bueno.

Las arquitecturas COMA (*cache only memory access*) intentan solucionar este problema haciendo que la memorial local a cada procesador se convierta en parte de una memoria caché grande. En este tipo de arquitecturas, las páginas de memoria desaparecen y todos los datos se tratan como líneas de caché. Sin embargo, con esta estrategia surgen nuevos interrogantes: ¿cómo se localizan las líneas de caché? y cuando una línea se tiene que reemplazar, ¿qué sucede si esta es la última? Estas preguntas no tienen fácil solución y pueden requerir de soporte *hardware* adicional. Normalmente se implementan utilizando un mecanismo de directorio distribuido.

Arquitecturas COMA

Las arquitecturas *cache only memory access* tratan la memoria principal como una memoria caché grande.

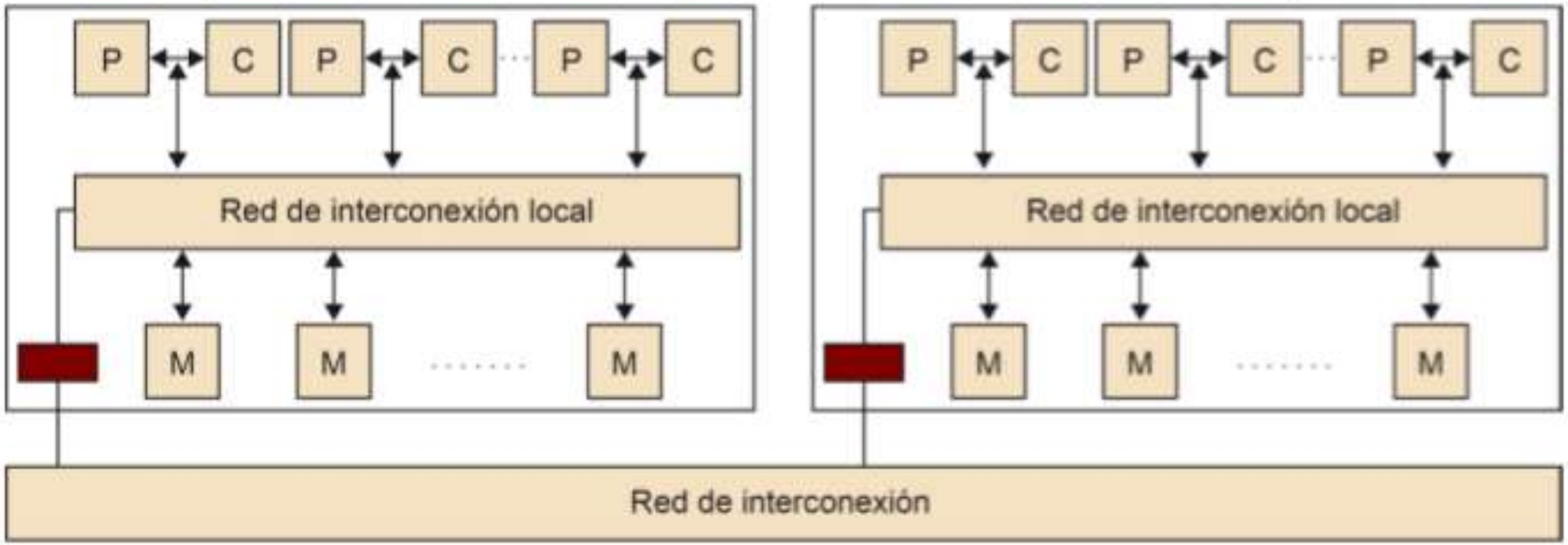
Así, aunque este tipo de máquinas prometían mejor rendimiento que las ccNUMA, se han construido pocas. Las primeras COMA que se construyeron fueron la KSR-1 y la *data diffusion machine*.

Multicomputadores

La característica más importante de estos sistemas es que tienen la memoria distribuida. Esta arquitectura también es conocida como arquitectura basada en el paso de mensajes, ya que los procesos deben realizar comunicaciones (mensajes) a través de la red de interconexión para poder compartir datos. Estas máquinas, por consiguiente, no están limitadas por el ancho de banda de memoria, sino más bien por el de la red de interconexión.

La figura muestra el esquema básico de un multicomputador donde cada nodo tiene parte de la memoria distribuida. Los nodos pueden contener desde un simple uniprocesador como un sistema multiprocesador.

Esquema básico de un multicomputador donde cada nodo es un multiprocesador.



MPP

Los sistemas MPP (*massively parallel processors*) son multicomputadores que tienen sus CPU conectadas con una red de interconexión estática de altas prestaciones (baja latencia y elevado ancho de banda) especialmente diseñada para el sistema. Son sistemas grandes, en comparación con un sistema CMP, pero que no suelen tener un número muy elevado de CPU debido al coste económico que supondría mantener una red de interconexión de altas prestaciones. En general, son sistemas difícilmente escalables.

Las características principales de los sistemas MPP son:

- Utilizan microprocesadores estandard, pero también pueden incorporar *hardware* específico o aceleradores (por ejemplo, ASIC*, FPGA**, GPU).

- La red de interconexión que incorporan es propietaria, especialmente diseñada, con muy baja latencia y un ancho de banda elevado.

- Suelen venir con *software* propietario y librerías para gestionar la comunicación.

- Tienen una capacidad de almacenamiento de entrada/salida elevada, ya que se suelen utilizar para trabajar con grandes volúmenes de datos que se han de procesar y almacenar.

- Tienen mecanismos de tolerancia de fallos del *hardware*.

MULTIPROCESADORES

IMPORTANCIA

Creciente importancia de multiprocesadores

- Caída de la eficiencia en uso de silicio y energía al explotar mayor nivel de ILP.
  - El coste de silicio y energía crece más rápidamente que el rendimiento.
- Interés creciente en servidores de alto rendimiento.
  - Cloud computing, *software as a service*, . . .
- Crecimiento de aplicaciones intensivas en datos.
  - Enormes cantidades de datos en Internet.
  - *Big data analytics*.

TLP: Paralelismo a nivel de hilo

- TLP implica la existencia de múltiples contadores de programa.
  - Asume MIMD.
  - Uso generalizado de TLP fuera de computación científica relativamente reciente.
  - Nuevas aplicaciones:
    - **Aplicaciones embebidas**
    - Desktop.
    - Servidores de alta gama.

Multiprocesadores

- Un **multiprocesador** es un computador formado por procesadores altamente acoplados con:
  - **Coordinación y uso** típicamente controlados por un **sistema operativo único**.
  - **Compartición de memoria** mediante un **único espacio de direcciones compartido**.

- **Modelos de software**:
  - **Procesamiento paralelo**: Conjunto de hilos acoplados que cooperan.
  - **Procesamiento de peticiones**: Ejecución de procesos independientes originados por usuarios.
  - **Multiprogramación**: Ejecución independiente de múltiples aplicaciones.

- La aproximación más común:
  - De 2 a decenas de procesadores.
  - Memoria compartida.
    - Implica memoria compartida.
    - No necesariamente implica una única memoria física.

- **Alternativas**:
  - **CMP** (*Chip MultiProcessors*) o *multi-core*.
  - Múltiples chips.
    - Cada uno puede ser (o no) *multi-core*.
  - **Multicomputador**: Procesadores débilmente acoplados que no comparten memoria.
    - Usados en computación científica de gran escala.

- **Aprovechamiento** de un multiprocesador:
  - Con **n** procesadores se necesitan **n** hilos o procesos.

- **Identificación** de hilos:
  - Identificados explícitamente por programador.
  - Creados por el sistema operativo a partir de peticiones.
  - Iteraciones de un bucle generadas por compilador paralelo (p. ej. OpenMP).

Multiprocesadores y memoria compartida

SMP: Symmetric Multi-Processor

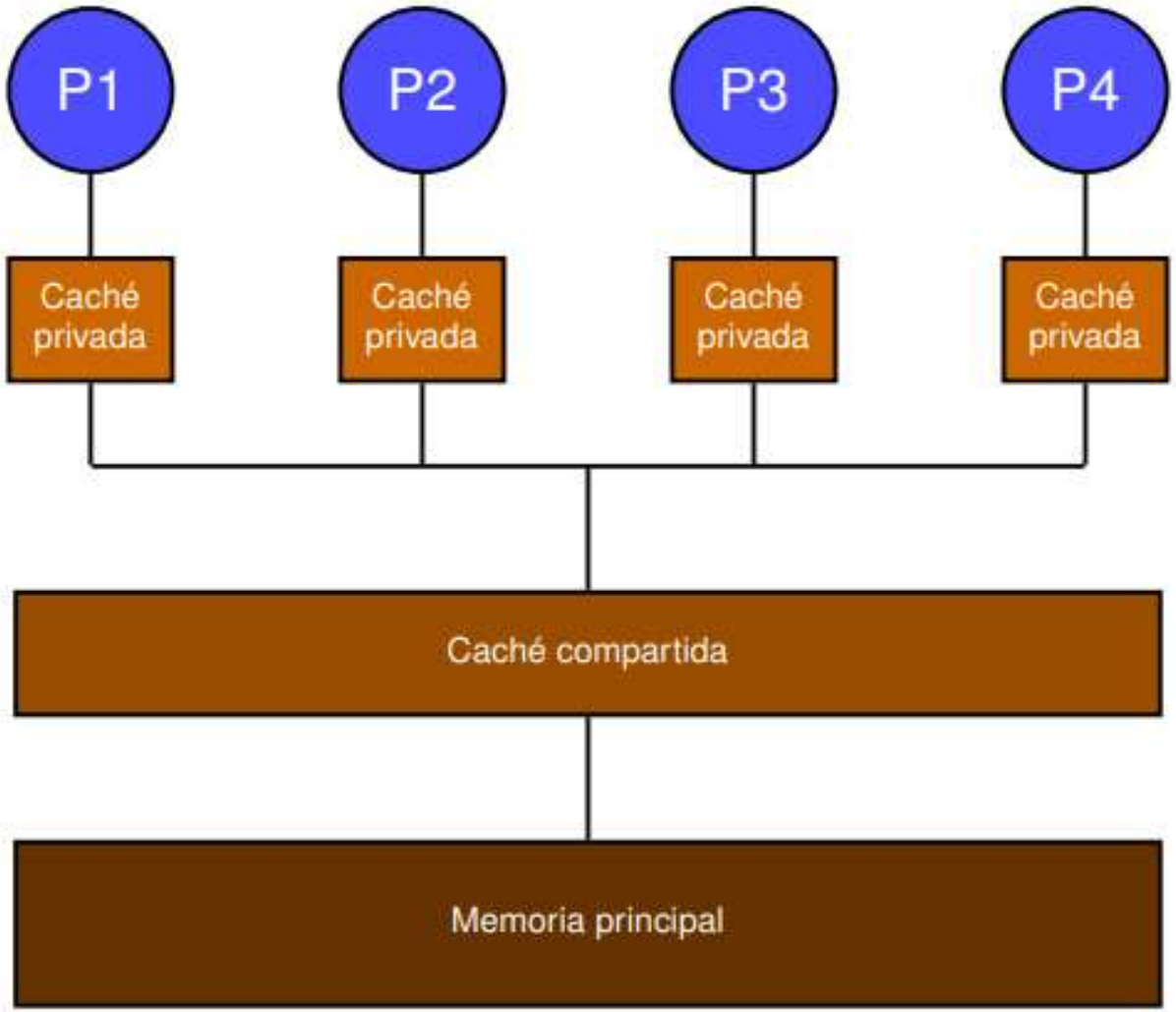
- Memoria compartida centralizada.
- Comparten una memoria centralizada única a la que todos acceder por igual.
- Todos los multi-core son SMP.
- **UMA**: Uniform Memory Access
  - La latencia de memoria es uniforme.

DSM: Distributed Shared Memory

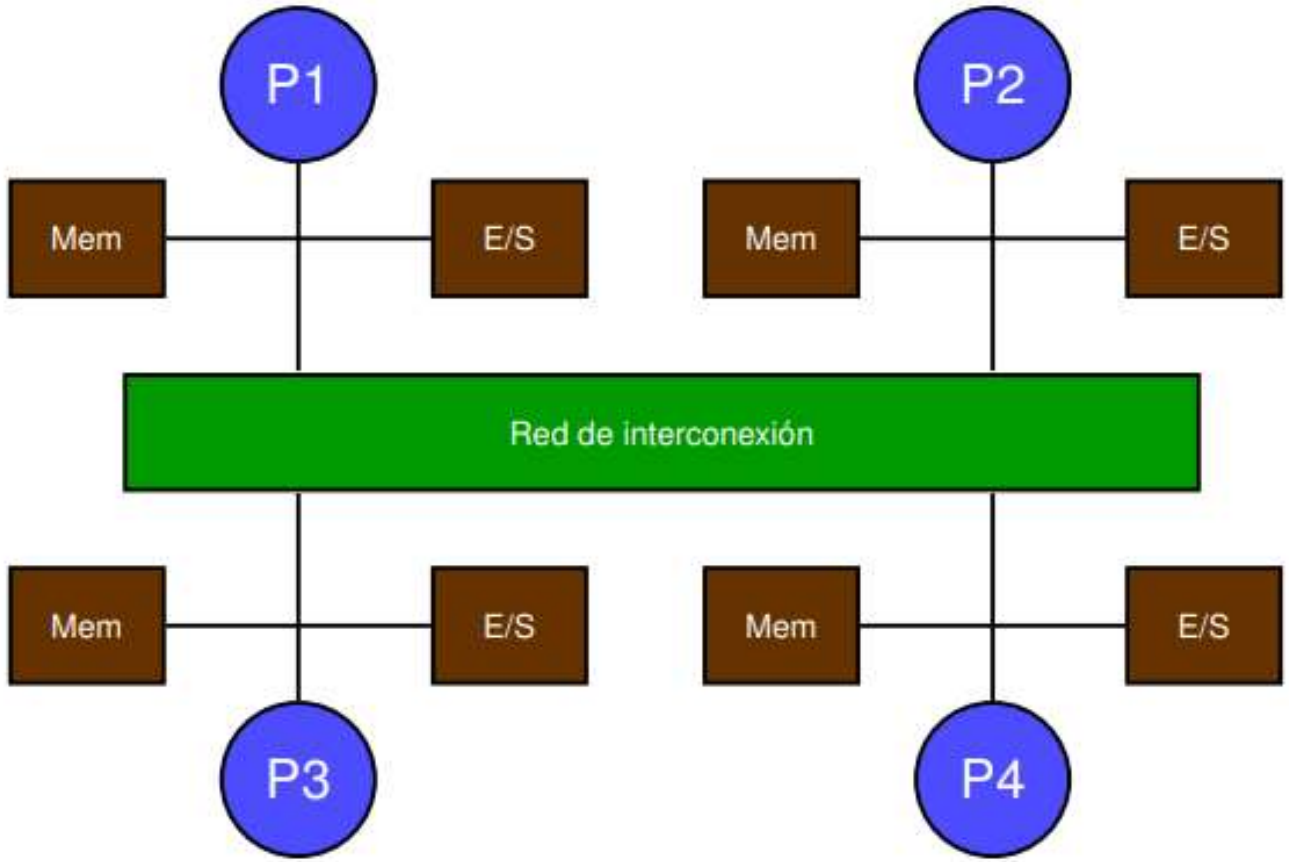
- Memoria compartida distribuida
- La memoria se distribuye entre los procesadores.
- Necesaria cuando hay muchos procesadores.
- **NUMA**: Non Uniform Memory Access.
  - La latencia depende de la ubicación del dato accedido.

- Comunicación mediante acceso a memoria compartidas globales.

SMP: Symmetric MultiProcessor



DSM: Distributed Shared Memory



SMP y jerarquía de memoria

- ¿Por qué usar memoria centralizada?
  - Cachés grandes multi-nivel reducen la demanda de ancho de banda sobre memoria principal.

Memoria caché

- **Tipos de datos** en memoria caché:
  - **Datos privados**: Datos usados por un único procesador.
  - **Datos compartidos**: Datos usados por varios procesadores.

Memoria caché

- **Tipos de datos** en memoria caché:
  - **Datos privados**: Datos usados por un único procesador.
  - **Datos compartidos**: Datos usados por varios procesadores.

- **Problema** con datos compartidos:
  - El dato puede replicarse en múltiples caché.
  - Reduce la contención.
    - Cada procesador accede a su copia local.
  - Si dos procesadores modifican sus copias . . .
    - ¿Coherencia de caché?

Incoherencia de caché

- ¿Por qué se da la incoherencia?
  - **Dualidad de estado**:
    - **Estado global** → **Memoria principal**.
    - **Estado local** → **Caché privada**.

- Un sistema de memoria es **coherente** si cualquier lectura de una posición devuelve el valor más reciente que se haya escrito para esa posición.

- **Dos aspectos**:
  - **Coherencia**: ¿Qué valor devuelve una lectura?
  - **Consistencia**: ¿Cuándo obtiene una lectura un valor escrito?

Condiciones para la coherencia

- **Preservación de orden de programa**:
  - Una lectura del procesador P sobre la posición X posterior a una escritura del procesador P sobre la posición X, sin escrituras intermedias de X por otro procesador, siempre devuelve el valor escrito por P.
- **Vista coherente de la memoria**:
  - Una lectura de un procesador sobre la posición X posterior a una escritura por otro procesador sobre la posición X, devuelve el valor escrito si las dos operaciones están suficientemente separadas en el tiempo y no hay escrituras intermedias sobre X.
- **Serialización de escrituras**:
  - Dos escrituras sobre la misma posición por dos procesadores son vistas en el mismo orden por todos los procesadores.

Consistencia de memoria

- Define en qué momento un proceso que lee verá una escritura.

- **Coherencia** y **consistencia** son complementarias:
  - **Coherencia**: Comportamiento de lecturas y escrituras a una única posición de memoria.
  - **Consistencia**: Comportamiento de lecturas y escrituras con respecto a accesos a otras posiciones de memoria.

- Existen distintos modelos de consistencia de memoria.

Multiprocesadores coherentes

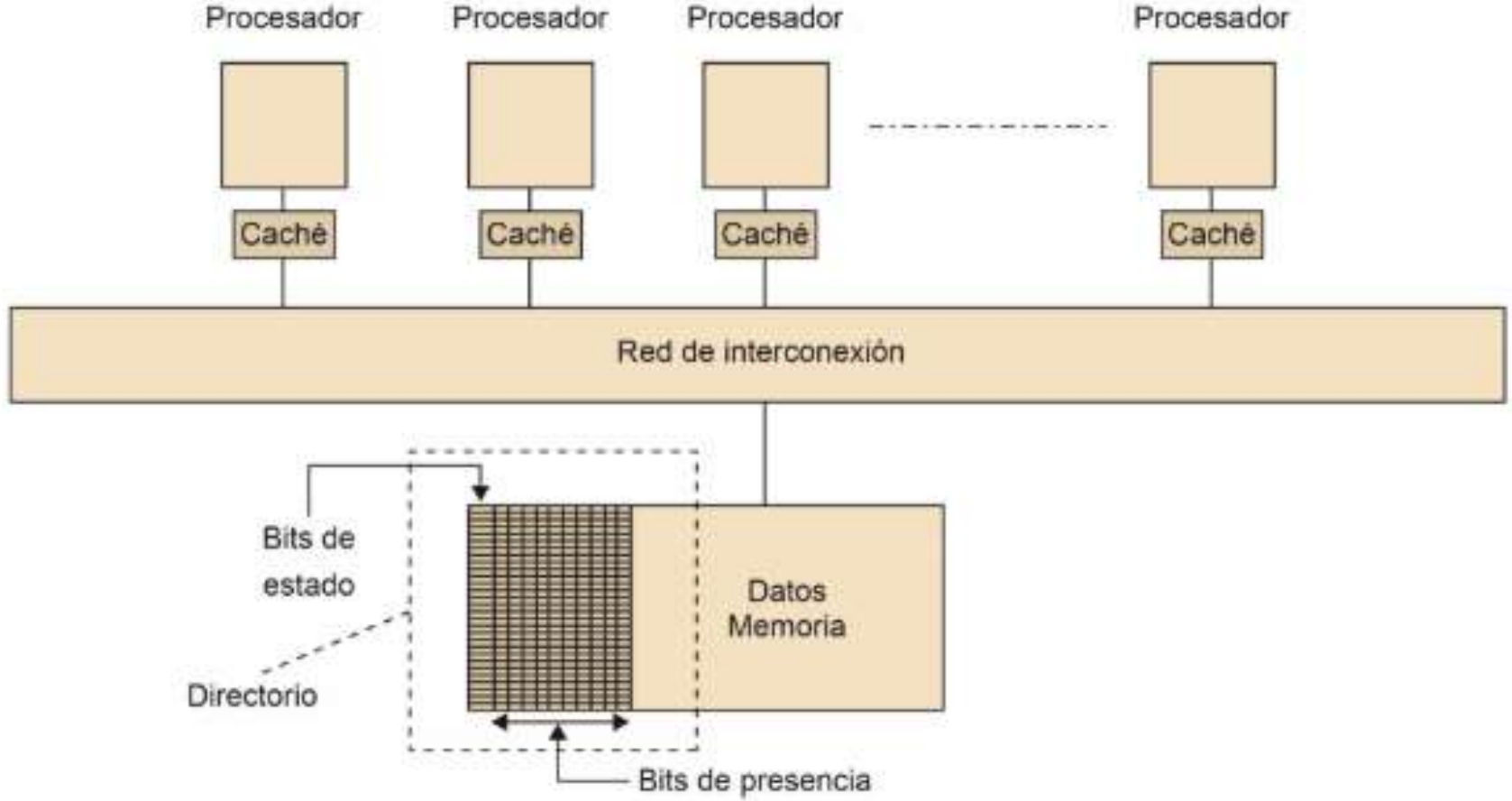
- Un **multiprocesador coherente** ofrece:
  - **Migración** de datos compartidos.
    - Un dato puede moverse a una caché local y usarse de forma transparente.
    - Reduce latencia de acceso a dato remoto y demanda de ancho de banda a la memoria compartida.
  - **Replicación** de datos compartidos leídos simultáneamente.
    - Se realiza copia del dato en caché local.
    - Reduce latencia de acceso y contención de las lecturas.
- **Propiedades críticas para el rendimiento**:
  - **Solución**: Protocolo hardware de mantenimiento de coherencia de caché.

Clases de protocolos de coherencia de caché

- Basados en **directorio**:
  - El estado de compartición se mantiene en un directorio.
  - **SMP**: Directorio centralizado en memoria o en caché de más alto nivel.
  - **DSM**: Para evitar cuello de botella se usa un directorio distribuido (más complejo).

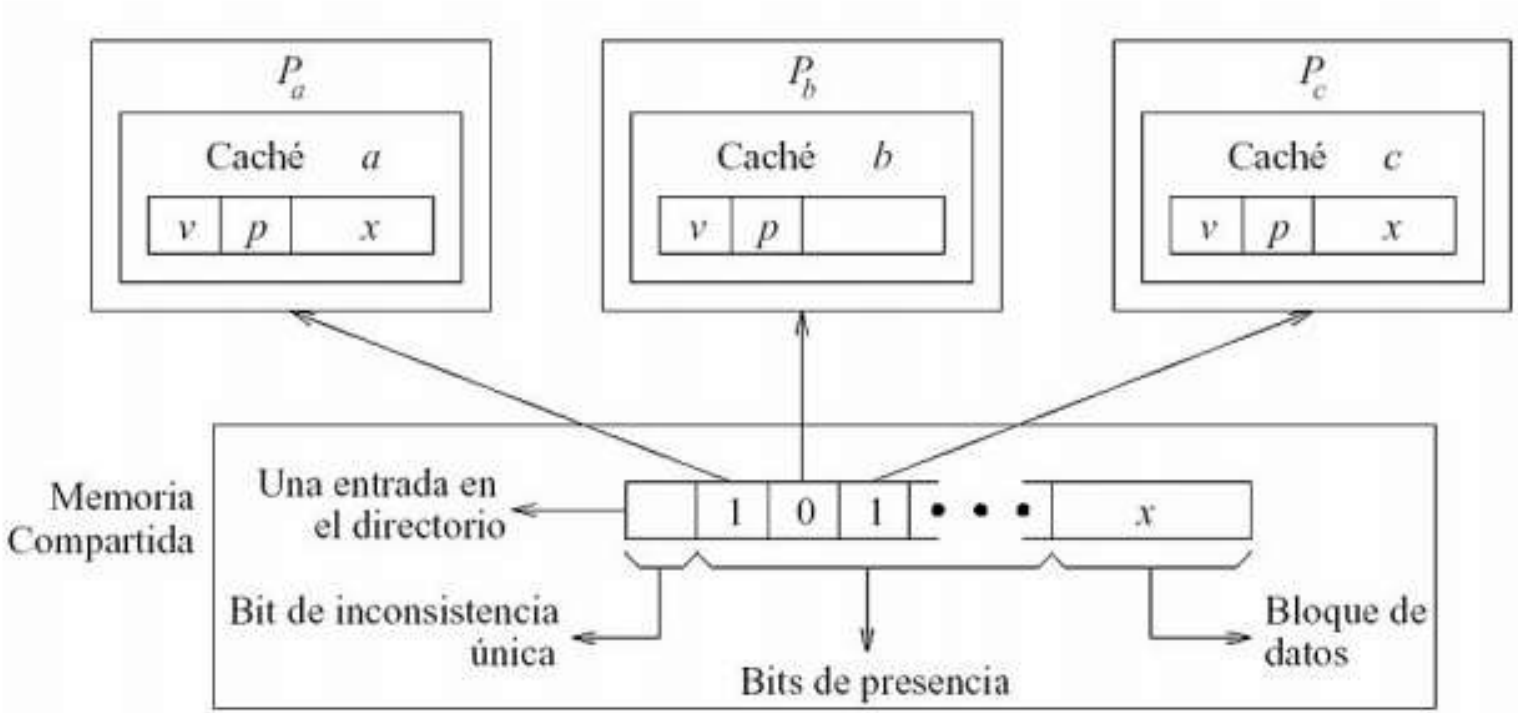
- **Snooping** (espionaje):
  - Cada caché mantiene el estado de compartición de cada bloque que tiene.
  - Las cachés accesibles mediante medio de multidifusión (bus).
  - Todas las cachés monitorizan el medio de multidifusión para determinar si tienen una copia del bloque.

Sistema multiprocesador formado por una memoria principal compartida con el sistema de directorios de coherencia de caché centralizado.



- Mantener un único identificador del procesador que tiene el valor de ese bloque de memoria. De esta forma sólo se podría tener una copia del dato. Esto no permite lecturas concurrentes sin provocar accesos en la memoria principal para actualizar qué procesador tiene la copia en cada momento. La ventaja es que es escalable desde el momento en que no guardas más que un entero independientemente del número de procesadores.

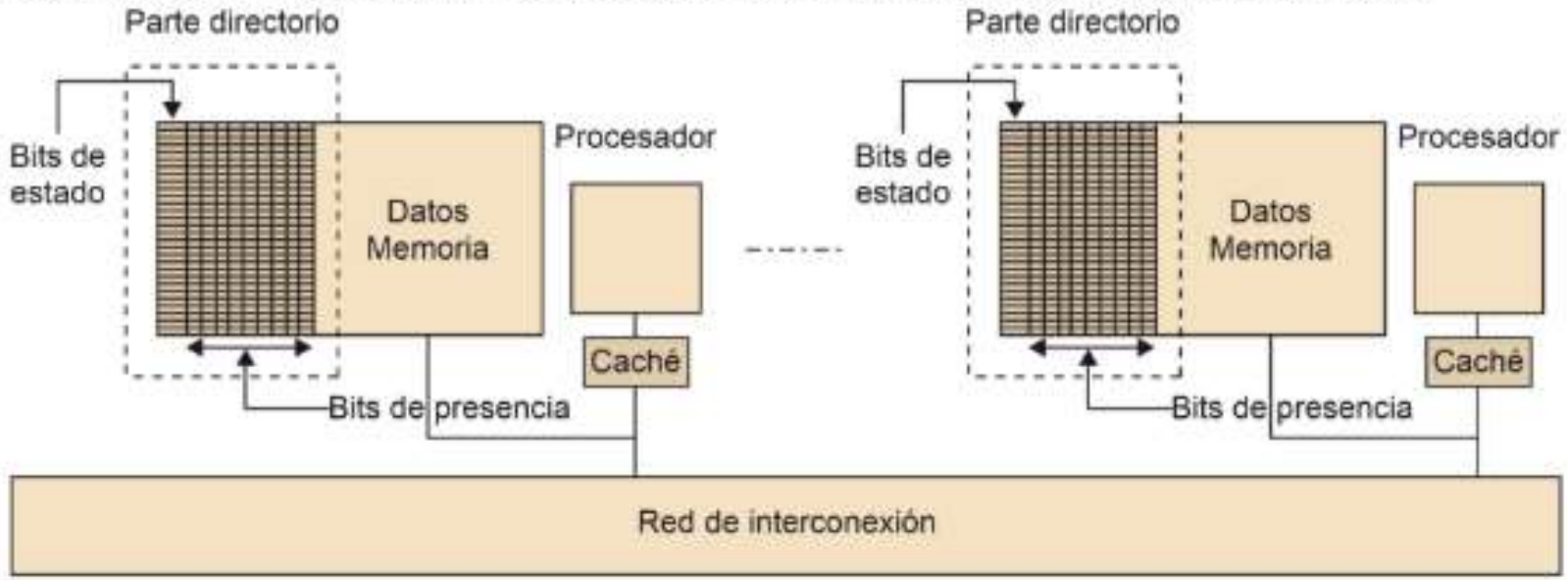
El directorio centralizado (protocolo de mapeado completo) mantiene un directorio en el cual cada entrada tiene un bit, llamado bit de presencia, por cada una de las cachés del sistema. El bit de presencia se utiliza para especificar la presencia en las cachés de copias del bloque de memoria. Cada bit determina si la copia del bloque está presente en la correspondiente caché. Por ejemplo, en la figura siguiente, las cachés de los procesadores Pa y Pc contienen una copia del bloque de datos x, pero la caché del procesador Pb no. Aparte, cada entrada del directorio tiene un bit llamado bit de inconsistencia única. Cuando este bit está activado, sólo uno de los bits de presencia está a uno, es decir, sólo existe una caché con la copia de ese bloque o línea, con lo que sólo esa caché tiene permiso para actualizar la línea



Directorio en el protocolo de mapeado completo.

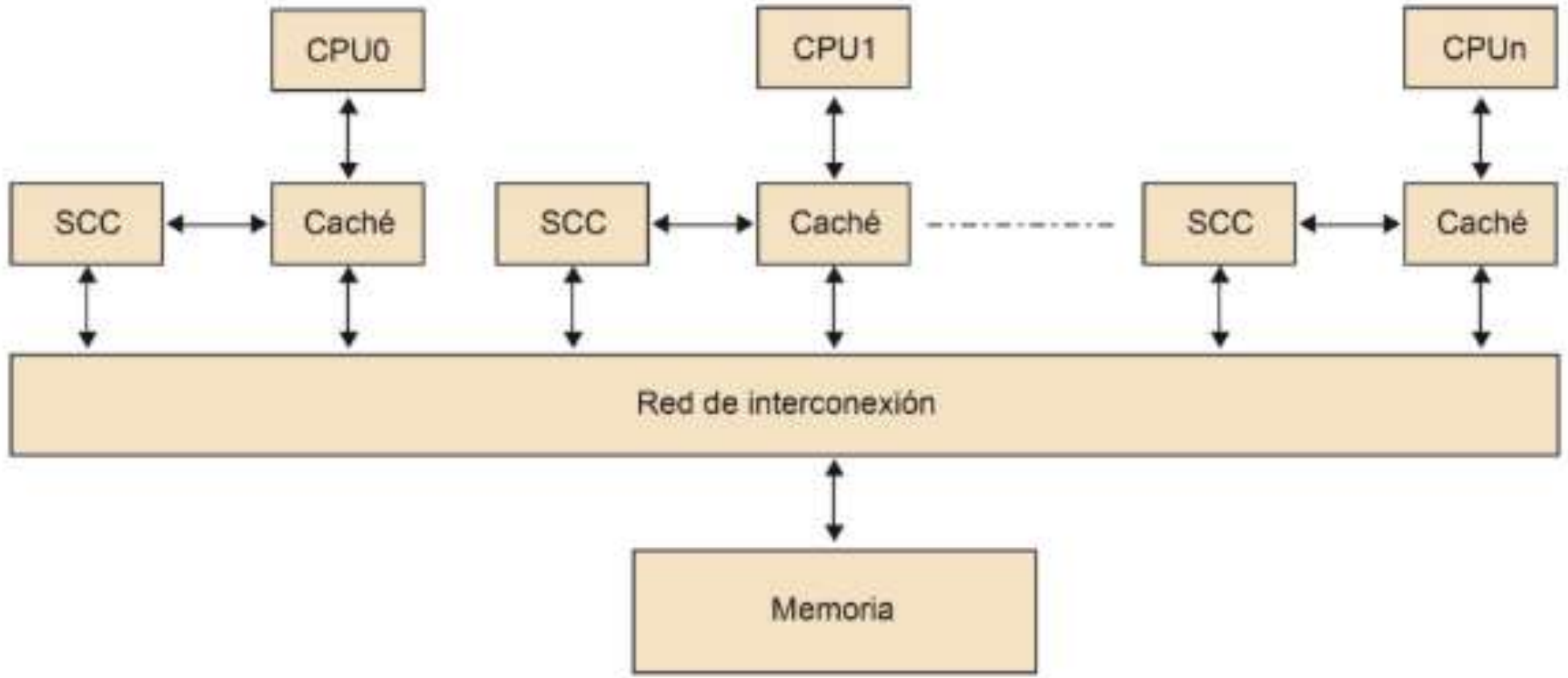
Cada caché por su parte tiene dos bits en cada entrada de la caché. El primero es el bit de validación e indica si la copia es válida o no. Si el bit es cero, indica que la copia no es válida, es decir, la copia se puede quitar de la caché. El otro bit, llamado bit de privacidad sirve para indicar si la copia tiene permiso de escritura, es decir, cuando este bit es uno entonces es la única copia que existe de esta línea en las cachés y por tanto tiene permiso para escribir

Sistema multiprocesador formado por una memoria principal compartida, físicamente distribuida, con el sistema de directorios de coherencia de caché distribuido.



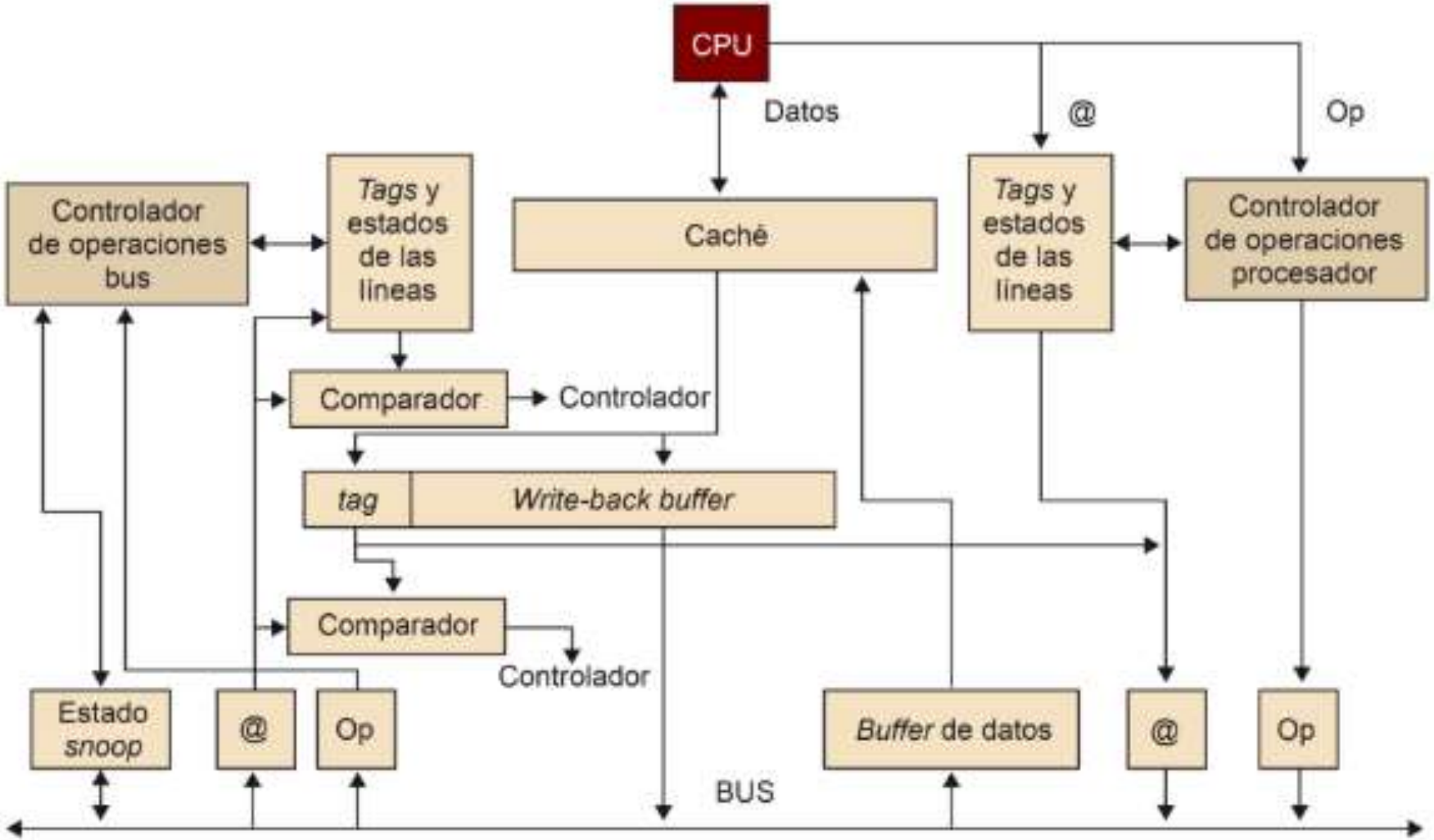
En un sistema con un protocolo basado en directorio, cada vez que se accede a un dato se tiene que consultar y posiblemente actualizar los bits de presencia y de estado en el directorio de memoria. Por lo tanto, si la memoria está formada por un solo módulo, tanto los accesos a memoria como la actualización del directorio se pueden convertir en cuellos de botella. Una posible solución es distribuir la memoria físicamente en módulos entre los nodos del sistema multiprocesador, repartiendo de la misma forma el directorio entre los diferentes módulos, para aumentar así el paralelismo en la gestión de accesos a la memoria principal y las transacciones a través del bus.

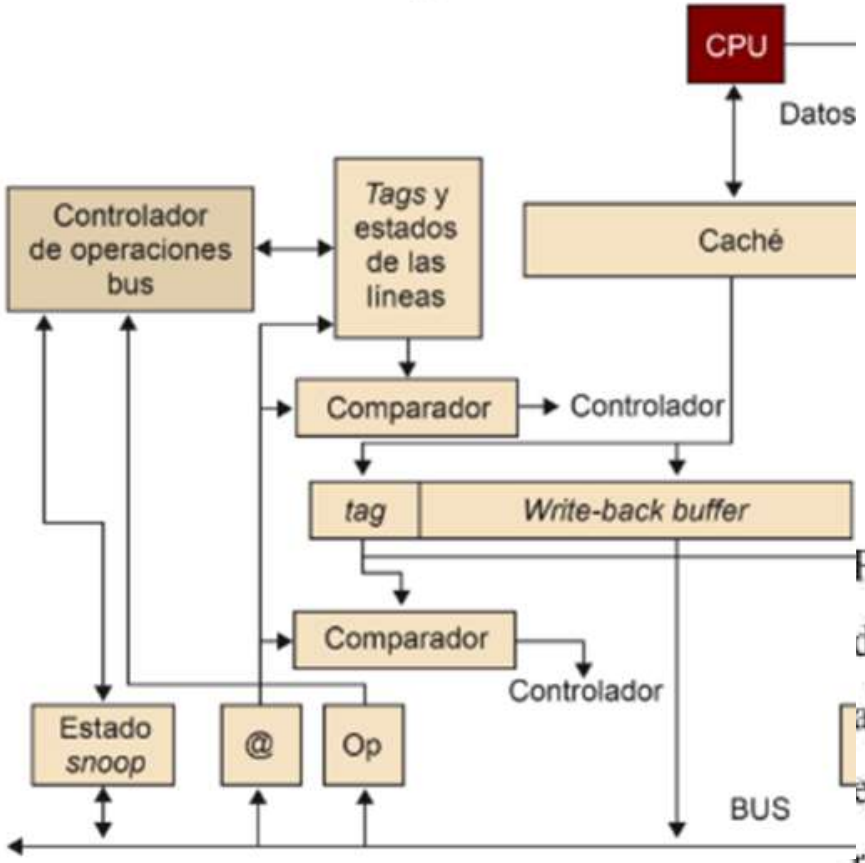
Esquema básico de un sistema multiprocesador con memoria compartida basado en un bus de interconexión y *Snoopy* cache *coherence*.



Un esquema básico de un sistema multiprocesador con una serie de procesadores, una caché asociada por procesador, y un hardware dedicado de Snoopy cache coherence (SCC). Todos ellos están conectados a la red de interconexión (normalmente un bus) que conecta la memoria compartida con los procesadores.

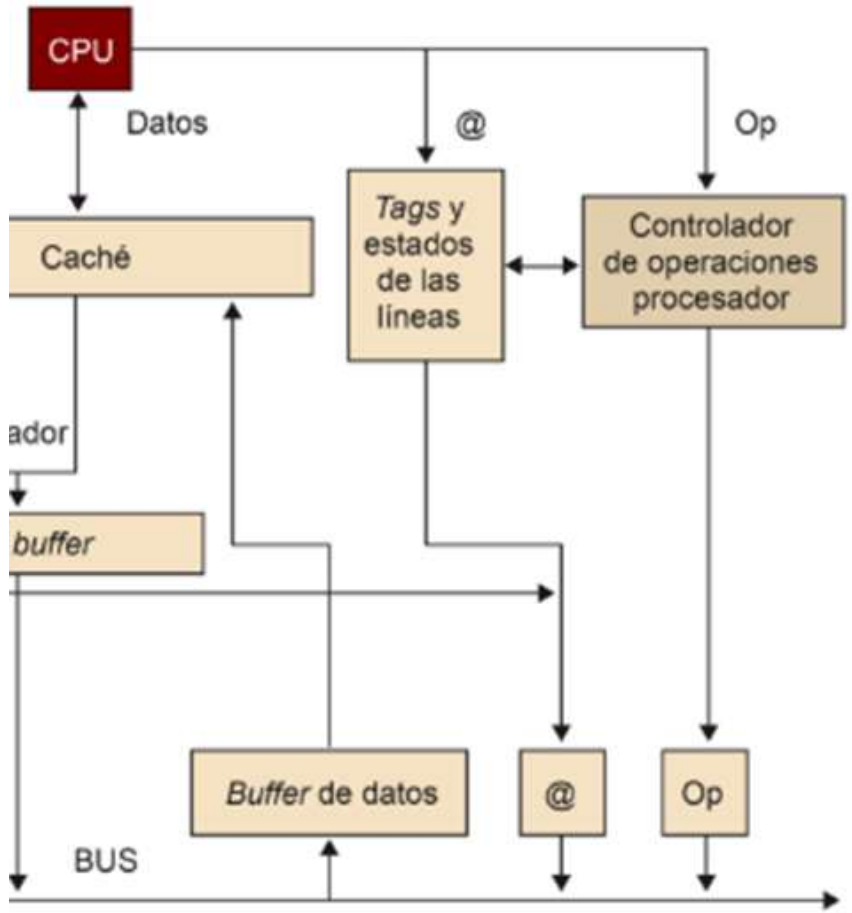
Detalle del *Snoopy cache coherence hardware.*





Por otro lado, si nos fijamos en el lado izquierdo de la figura, veremos que hay un *hardware* dedicado a sondear/recibir los comandos u operaciones que vienen del bus (Op conectado al BUS) o interconexión con memoria y que se conecta con el controlador del bus para enviarle la información. La dirección que viene del bus se lee y se pasa a los dos comparadores existentes: uno para comparar con los *tags* de la caché y otro para compararlo con los *tags* del *write-back buffer*. El *write-back buffer* es un *buffer* para agrupar escrituras a una misma línea de caché. En caso de tener coincidencia en alguna línea o en el *write-back buffer*, se deberá actuar en consecuencia según el protocolo de coherencia seguido. También observamos que hay una conexión llamada estado *Snoopy* que le sirve al procesador para informar al bus de que tiene una copia del dato del cual se está haciendo una lectura por parte de otro procesador, o viceversa, para que informen al procesador de que no tendrá exclusividad en el dato que está leyendo de memoria.

SCC. Si nos fijamos en la parte derecha del esquema, veremos un *hardware* dedicado (controlador del procesador) que recibe comandos del procesador y que está conectado a la información de los *tags* y del estado de las líneas de caché. Así, dependiendo de los comandos u operaciones que se efectúen en el procesador, el controlador del procesador, mirando el estado de las líneas de caché y sus *tags*, decidirá enviar operaciones al bus y/o cambiar el estado de alguna línea de caché.



Mantenimiento de la coherencia

- **Invalidación de escrituras**:
  - Garantiza que un procesador tiene **acceso exclusivo** a un bloque antes de realizar una escritura.
  - **Invalida** el resto de copias que puedan tener otros procesadores.

- **Actualización de escrituras** (difusión de escrituras):
  - Difunde todas las escrituras a **todas las cachés** para modificar el bloque.
  - Consume **más ancho de banda**.

- Estrategia más común $\Rightarrow$ **Invalidación**.

Uso del bus de memoria

- **Invalidación**.
  - El procesador adquiere el bus y difunde la dirección a invalidar.
  - Todos los procesadores espían el bus.
  - Cada procesador comprueba si tienen en caché la dirección difundida y la invalidan.
- No puede haber dos escrituras simultáneas:
  - El uso exclusivo del bus serializa las escrituras.
- **Fallos de caché**:
  - **Escritura inmediata** (write through):
    - La memoria tiene la última escritura realizada.

Implementación

- **Invalidación**:
  - Se aprovecha el **bit de validez** (V) asociado a cada bloque.
- **Escrituras**:
  - Necesidad de saber si hay **otras copias** en caché.
    - Si no hay otras copias no hay que **difundir escritura**.
  - Se añade **bit de compartición** (S) asociado a cada bloque.
  - Cuando hay escritura:
    - Se genera **invalidación** en bus.
    - Se pasa de **estado compartido** a **estado exclusivo**.
    - No hace falta enviar nuevas invalidaciones.
  - Cuando hay **fallo de caché** en otro procesador:
    - Se pasa de **estado exclusivo** a **estado compartido**.

Protocolo básico

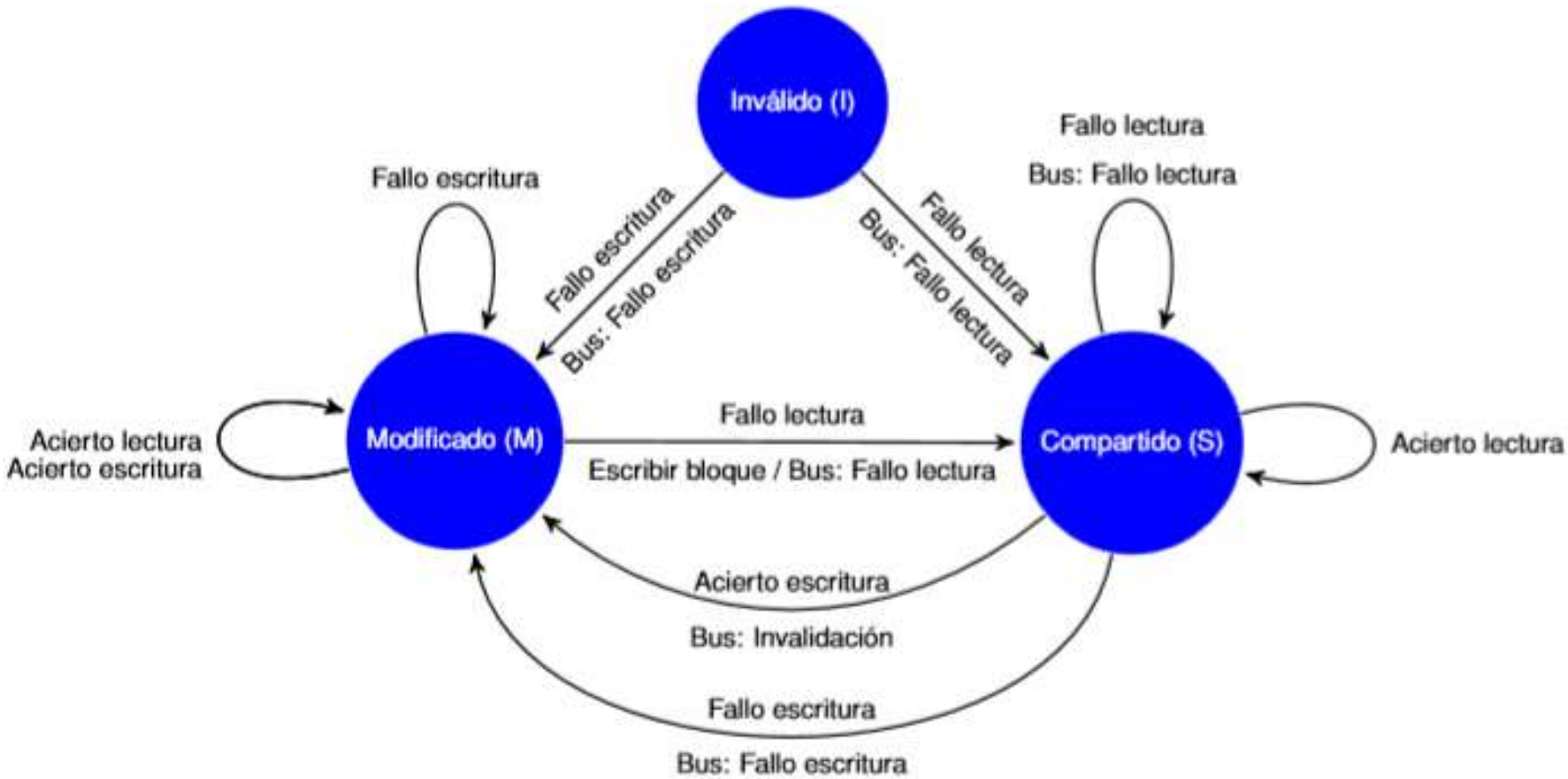
- Basado en una **máquina de estados** para **cada bloque de caché**:
  - **Cambios de estado** generados por:
    - Peticiones del procesador.
    - Peticiones del bus.
  - **Acciones**:
    - Cambios de estado.
    - Acciones sobre el bus.

- Aproximación simple con **tres estados**:
  - **M**: El bloque ha sido modificado.
  - **S**: El bloque está compartido.
  - **I**: El bloque ha sido invalidado.

Acciones generadas por el procesador

Petición	Estado	Acción	Descripción
Acierto de lectura	S → S	Acierto	Leer dato de caché local
Acierto de lectura	M → M	Acierto	Leer dato de caché local
Fallo de lectura	I → S	Fallo	Difundir fallo de lectura en bus.
Fallo de lectura	S → S	Remplazo	Fallo de conflicto de dirección. Difundir fallo de lectura en bus.
Fallo de lectura	M → S	Remplazo	Fallo de conflicto de dirección. Escribir bloque y difundir fallo de lectura.
Acierto de escritura	M → M	Acierto	Escribir dato en caché local.
Acierto de escritura	S → M	Coherencia	Invalidación en bus.
Fallo de escritura	I → M	Fallo	Difundr fallo de escritura en bus.
Fallo de escritura	S → M	Remplazo	Fallo de conflicto de dirección. Difundir fallo de escritura en bus.
Fallo de escritura	M → M	Remplazo	Fallo de conflicto de dirección. Escribir bloque y difundir fallo de escritura.

Protocolo MSI: Acciones de procesador



Acciones generadas por el bus

Petición	Estado	Acción	Descripción
Fallo de lectura	S → S	–	Memoria compartida sirve el fallo
Fallo de lectura	M → S	Coherencia	Intento de compartir dato. Se coloca bloque en bus.
Invalidar	S → I	Coherencia	Intento de escribir bloque compartido. Invalidar el bloque.
Fallo de escritura	S → I	Coherencia	Intento de escribir bloque compartido. Invalidar el bloque.
Fallo de escritura	M → I	Coherencia	Intento de escribir bloque que es. exclusivo en algún sitio. Se escribe (*write-back*) el bloque.

Protocolo MSI: Acciones de bus

Complejidad del protocolo MSI

- El protocolo asume que las operaciones son **atómicas**.
  - **Ejemplo**: Se asume que se puede detectar un fallo de escritura, adquirir el bus y recibir una respuesta en una única acción sin interrupción.

- Si las operaciones **no son atómicas**:
  - Posibilidad de deadlock y/o carreras.

- **Solución**:
  - El procesador que envía invalidación mantiene **propiedad del bus** hasta que la invalidación llega al resto.

Se da una situación de “*deadlock*” cuando dos o más *threads* se bloquean mutuamente, esperando el uno al otro, sin progresar ninguno de ellos.

Carreras:
un programa sufre este problema cuando el resultado puede ser correcto o incorrecto dependiendo de cómo se de la casualidad de que se entrelacen los *threads*. Básicamente el problema aparece cuando varios *threads* hacen ciclos de modificación de una variable compartida (lectura + modificación + escritura) y el ciclo completo se ve interrumpido

Extensiones a MSI

- **MESI**:
  - Añade **estado exclusivo** (E) que indica que el bloque reside en una única caché pero no está modificado.
  - Escritura de un bloque E **no genera invalidaciones**.
- **MESIF**:
  - Añade **estado forward** (F): Alternativa a S que indica qué nodo debe responder a una petición.
  - Usado en Intel Core i7.
- **MOESI**:
  - Añade **estado poseído** (O) que indica que el bloque está desactualizado en memoria.
  - Evita escrituras a memoria.
  - Usado en AMD Opteron.

Resumen

- Multiprocesador como computador con procesadores altamente acoplados con coordinación, uso y compartición de memoria.
- Multiprocesadores clasificados en SMP (Symetric multiprocessor) y DSM (Distributed Shared Memory).
- Dos aspectos a considerar en la jerarquía de memoria: coherencia y consistencia.
- Dos alternativas en la coherencia de caché: directorio y espionaje (*snooping*).
- Los protocolos de espionaje no requieren un elemento centralizado.
  - Pero generan más tráfico de bus.