Tema 10

Sistemas Multiprocesadores

10.1 Multiprocesadores Simétricos (SMP)

A medida que aumenta la siempre creciente demanda de mayores prestaciones, y conforme el coste de los microprocesadores se reduce, los fabricantes han introducido los sistemas SMP. El término SMP, *sistema multiprocesador simétrico*, se refiere a la arquitectura hardware del sistema multiprocesador y al comportamiento del sistema operativo que utiliza dicha arquitectura. Un SMP es un computador con las siguientes características:

1) Tiene dos o más procesadores similares de capacidades comparables.
2) Los procesadores comparten la memoria principal y la E/S, y están interconectados mediante un bus u otro tipo de sistema de interconexión, de manera que el tiempo de acceso a memoria es aproximadamente el mismo para todos los procesadores.
3) Todos los procesadores comparten los dispositivos de E/S, pero pueden hacerlo bien a través de los mismos canales, o bien a través de otros caminos de acceso al mismo dispositivo.
4) Todos los procesadores pueden desempeñar las mismas funciones (de ahí el término simétrico).
5) El sistema está controlado por un sistema operativo que posibilita la interacción entre los procesadores y sus programas.

La última de las características anteriores apunta a una de las diferencias de los SMP respecto a los sistemas multiprocesadores débilmente acoplados, como son los 'clusters'. En estos, la unidad de interacción es normalmente un mensaje o un fichero completo. Sin embargo, en un SMP, la interacción se puede producir a través de elementos de datos individuales, y puede existir un elevado nivel de cooperación entre procesadores. De ahí que esten clasificados como sistemas fuertemente acoplados.

A continuación se relacionan las ventajas potenciales de un SMP respecto a una arquitectura monoprocesador:

- **Prestaciones**: Si el trabajo a realizar por un computador puede organizarse de forma que diferentes partes puedan realizarse en paralelo, concurrentemente, entonces un sistema

con varios procesadores proporcionará mejores prestaciones que uno con un sólo procesador del mismo tipo.

- **Disponibilidad**: Como en un SMP todos los procesadores pueden realizar las mismas funciones, un fallo en un procesador no hará que el computador se detenga.
- **Crecimiento incremental**: Se pueden aumentar las prestaciones del sistema añadiendo más procesadores.
- **Escalado**: Los fabricantes pueden ofrecer una gama de productos con diferentes precios y prestaciones, en función del número de procesadores que configuran el sistema.

Sin embargo, los beneficios anteriores no son beneficios garantizados, sino potenciales. Por su parte, el sistema operativo debe proporcionar herramientas y funciones que permitan explotar el paralelismo del programa y proyectarlo sobre los diferentes procesadores de un SMP.

Los sistemas SMP poseen una característica muy atractiva, y es que la existencia de varios procesadores es transparente al usuario. Es el sistema operativo el que posibilita la sincronización entre los procesadores, y la planificación de los hilos o de los procesos, puesto que es el responsable de asignarlos a los distintos procesadores.

10.1.1. Organización

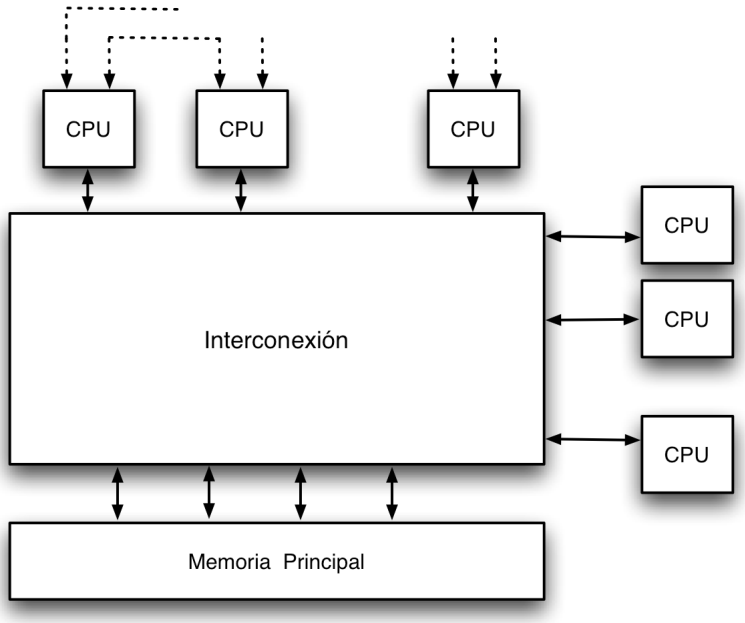

figura 10.1. Diagrama de bloques de un sistema multiprocesador.

Existen dos o más CPUs, cada una de las cuales contiene: a) unidad de control, b) ALU, c) registros y d) posiblemente caché. Cada CPU tiene acceso a una memoria principal compartida y a los dispositivos I/O a través de un mecanismo de interconexión. Los procesadores pueden

comunicarse entre ellos a través de la memoria (mensajes e información de estados almacenados en áreas comunes). También pueden intercambiar directamente señales, indicadas por las líneas punteadas en la figura 10.1. La memoria se organiza frecuentemente de manera que se pueden realizar accesos simultáneos a bloques independientes. En algunas configuraciones, cada CPU puede tener también su memoria principal privada y canales I/O además de los recursos compartidos.

La organización de un sistema multiprocesador puede clasificarse de la siguiente forma:

- Tiempo compartido o Bus Común.
- Memoria Multipuerto.
- Unidad de Control Central.

10.1.1.1. Bus de tiempo compartido

Es el mecanismo más simple para implementar un sistema multiprocesador. La estructura e interfaces son básicamente los mismos de un sistema con un procesador único. Para facilitar las transferencias DMA desde los procesadores de I/O, se añaden las siguientes características:

- **Direccionamiento**: Se pueden distinguir los módulos del bus para determinar la fuente y el destino de los datos.
- **Arbitraje**: Existe un mecanismo para arbitrar peticiones de control del bus, utilizando algún tipo de esquema de prioridades. Los módulos I/O también pueden funcionar temporalmente como master.
- **Tiempo Compartido**: Cuando un módulo está controlando el bus, los módulos restantes no están autorizados y deben suspender, si es necesario, la operación hasta que se les asigne el acceso al bus.

Las principales ventajas de esta estructura son:

- **Simplicidad**, ya que la estructura es la misma que en un sistema uniprocesador.
- **Flexibilidad**: Es fácil expandir el sistema añadiendo más CPUs.
- **Fiabilidad**: El bus es esencialmente un medio pasivo, por lo que en principio no debe producir fallos en el sistema.

La principal desventaja consiste en que todas las referencias a memoria pasan a través de un bus común, por lo que la velocidad del sistema está limitada por el ciclo de bus. Para mejorar esto, es conveniente equipar cada CPU con una memoria caché. Ésta reduce significativamente el número de accesos.

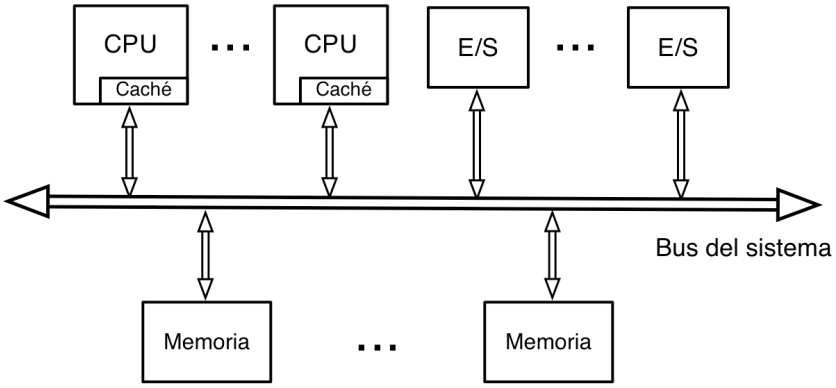

figura 10.2. Diagrama de bloques de un sistema multiprocesador.

Si se utiliza memoria caché, se necesitan algunas consideraciones de diseño. Puesto que cada caché contiene datos tomados de la memoria compatida, si se altera un dato en una caché después de un resultado, virtualmente esto puede invalidar el dato de otra caché. Para prevenir esto, las restantes CPUs deben ser alertadas de ello y tomar las medidas necesarias (problema de la *coherencia de la caché*).

10.1.1.2. Memoria multipuerto

La aproximación de memoria multipuerto permite el acceso independiente y directo a los módulos de memoria principal por parte de cada CPU y módulo I/O. Se necesita una lógica asociada a la memoria para resolver conflictos de acceso. El método más utilizado es asignar permanentemente prioridades designadas a cada puerto de memoria.

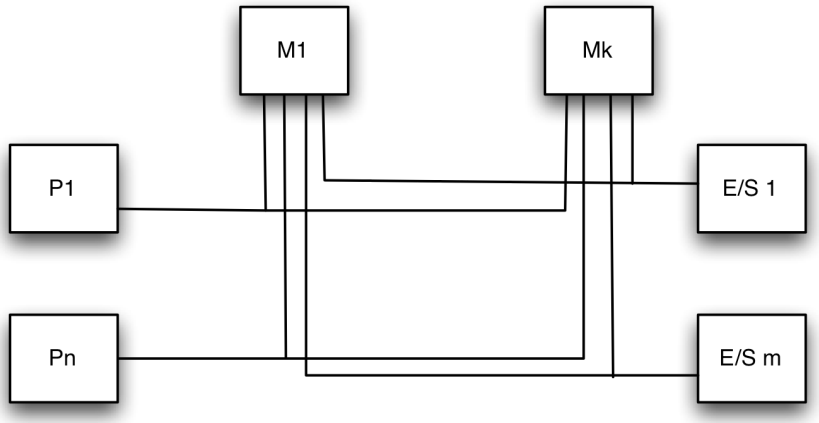

figura 10.3. Diagrama de bloques de un sistema multiprocesador.

Esta aproximación es más compleja que la aproximación de bus, requiriendo gran cantidad de lógica añadida al sistema. Las características de funcionamiento son mejores puesto que cada procesador y cada I/O disponen de caminos independientes a cada módulo de memoria. Otra ventaja es la posibilidad de configurar porciones de memoria como 'privadas' de una CPU y/o un módulo I/O. Esto permite incrementar la seguridad contra accesos no autorizados y para el almacenamiento de rutinas de recuperación en áreas de memoria no modificables por otros procesadores.

En esta estructura, sólo puede utilizarse el método 'write through' (que se verá más adelante) ya que no puede alertarse a los otros procesadores de modificaciones en una caché.

10.1.1.3. Unidad de Control Central

La unidad de control central proporciona canales de datos separados para cada sentido entre módulos independientes: CPU, memoria y I/O. El controlador memoriza las peticiones e implementa funciones de arbitraje y temporización. Puede pasar también mensajes de control y estado entre CPUs, y alertar de modificación de cachés.

Puesto que toda la lógica de coordinación del multiprocesador está concentrada en la unidad de control, los interfaces de I/O, memoria y CPU se descargan prácticamente de estas funciones, lo que le proporciona la flexibilidad y simplicidad de la aproximación de bus. La principal desventaja es la complejidad de la unidad de control, y la posibilidad de convertirse en un 'cuello de botella'.

Esta estructura es usual en sistemas multiprocesadores con mainframes.

10.2. Grandes Computadores SMP (IBM S/390)

La mayoría de los PC y estaciones de trabajo de tipo SMP utilizan una estrategia de interconexión basada en bus. Sin embargo, existen otro tipo de organizaciones donde el mecanismo de interconexión no es un bus. Por ello, resulta ilustrativo analizar una aproximación alternativa al bus común, que se utiliza en las implementaciones más recientes de la familia de grandes computadores (mainframes) IBM S/390.

La figura 10.4 muestra un esquema que corresponde a la organización general del SMP S/390. Esta familia de sistemas es escalable, e incluye, desde computadores monoprocesador con un módulo de memoria principal, en su configuración más básica, hasta sistemas con diez procesadores y cuatro módulos de memoria, en la gama alta.

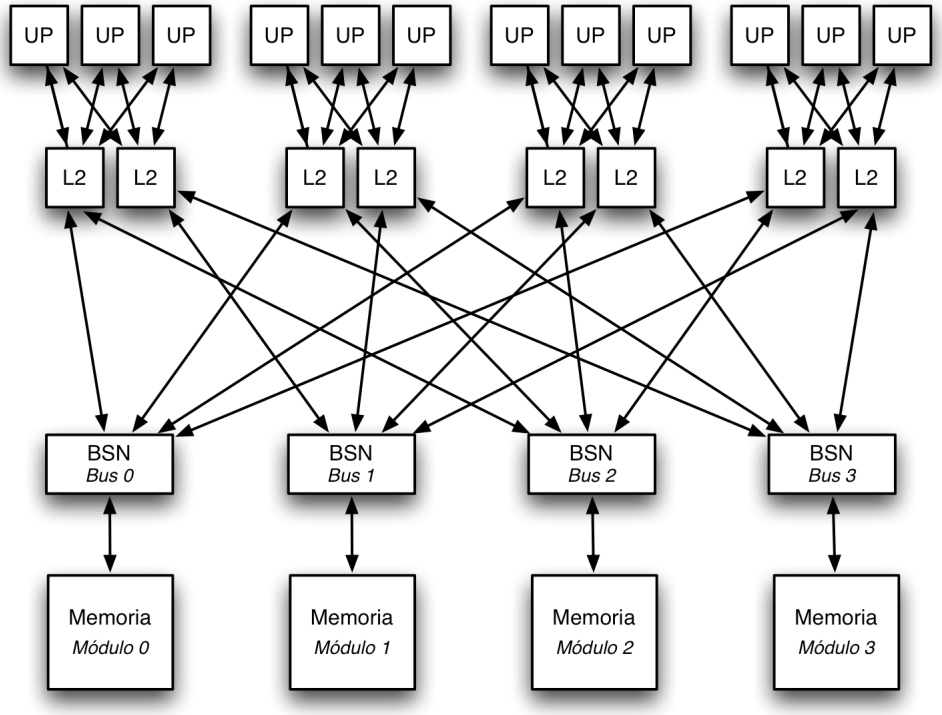

figura 10.4. Organización del SMP S/390 de IBM.

Los componentes básicos son los siguientes:

- **Procesador (PU, *Processor Unit*):** Los procesadores son de tipo CISC, en los que las instrucciones más frecuentemente ejecutadas se encuentran cableadas. Cada procesador incluye una caché interna L1 unificada (almacena datos e instrucciones) de 64 KB, con un acceso de un solo ciclo.

- **Caché L2:** Cada caché L2 es de 384 KB. Las cachés L2 se organizan de una forma especial, en grupos de dos, de forma que cada grupo puede recibe accesos de tres procesadores, y proporciona acceso a todo el espacio de memoria principal, tal como se representa en la figura 10.4.

- **Adaptador bus-red de interconexión (BSN, *Bus Switching Network*):** Los BSN interconectan las cachés L2 y la memoria principal. Cada BSN incluye también una caché de tercer nivel (L3) de 2 MB.

- **Módulos de memoria:** Cada módulo dispone de 8 GB de memoria, con una capacidad total de 32 GB.

Hay una serie de características en la configuración del SMP S/390 interesantes para su discusión:

- Interconexión conmutada.
- Cachés L2 compartidas.
- Cachés L3.

Interconexión conmutada

Esta es una estrategia para evitar el cuello de botella que impone el esquema de bus compartido en los SMP tradicionales, y que afecta a la escalabilidad cuando el sistema se amplía. El S/390 se enfrenta a este problema de dos formas:

- La memoria principal se distribuye en cuatro módulos. De esta forma, existen cuatro caminos independientes a memoria, y el tráfico de cargas desde memoria se reduce en un factor de cuatro.
- Las conexiones de los procesadores (más en concreto de las cachés L2) a cada módulo de memoria no se produce como en un bus compartido, sino más bien como en un enlace punto-a-punto. Cada enlace conecta un grupo de tres procesadores, a través de una caché L2, con un BSN. La tarea del BSN es encaminar datos entre sus cinco enlaces (cuatro enlaces con memorias L2 y uno con un módulo de memoria), realizando así la función de conmutador. Con respecto a los enlaces con cachés L2, el BSN conecta estos cuatro enlaces físicos a un bus de datos lógico, reenviando una señal procedente de uno de los cuatro enlaces con las cachés L2 a cada uno de las tres cachés L2 restantes. Esto es necesario para permitir mantener la coherencia de la caché.

Nótese que, aunque hay cuatro módulos de memoria, cada caché L2 sólo tiene dos puertos físicos y, por tanto, sólo puede conectarse a dos de los cuatro módulos de memoria. Esto se debe a que cada caché L2 sólo almacena datos de la mitad de la memoria principal, por lo que se necesitan dos cachés para dar servicio a toda la memoria principal. Por otro lado, cada procesador tiene dos puertos para poder conectarse a ambas cachés.

Cachés L2 compartidas

En una configuración típica de sistema SMP con caché de dos niveles, cada procesador tiene cachés L1 y L2 propias. Esto ocurría también en las primeras versiones del SMP S/390 (G3), donde cada procesador tenía una caché L2 propia. Sin embargo, últimamente ha aumentado el interés por la utilización de cachés L2 compartidas, como es el caso que nos ocupa. Las razones que llevaron a considerar el uso de una o más cachés compartidas fueron:

- Al cambiar de la versión G3 a la G4, IBM pasó a utilizar procesadores con doble velocidad. Manteniendo la organización G3, se hubiese producido un aumento significativo del tráfico a través del bus. El BSN podría haber llegado a ser un cuello de botella de haber mantenido la organización G3, a no ser que se hubiese mejorado significativamente el bus.

- El análisis de las cargas de trabajo de un S/390 típico mostraba un elevado nivel de instrucciones y datos compartidos por los procesadores. Esto se puede aprovechar con la utilización de cachés compartidas.

A primera vista, compartir la caché L2 puede parecer una mala idea. Puede pensarse que el acceso a memoria de los procesadores será más lento, debido a que los procesadores deben pugnar por el acceso a la caché L2. Sin embargo, como varios procesadores comparten un elevado volumen de datos, en realidad una caché compartida puede incrementar el rendimiento en lugar de disminuirlo, ya que los datos compartidos, que se encuentran en la caché compartida, se obtienen mucho más rápidamente que si accediera a ellos a través del bus.

Caché L3

Otra característica interesante del SMP S/390 es la utilización de un tercer nivel de caché (L3). Estas cachés L3 se encuentran en los BSN a modo de buffer entre las cachés L2 y uno de los módulos de memoria. De esta forma, la caché L3 reduce el retraso de acceso a los datos que no estan en ninguna de las cachés L1 y L2 del procesador que los solicita. El uso de una cache L3 permite que el dato esté disponible mucho más rápidamente que si hubiese que acceder a la memoria principal, en el caso de que dicho dato se encuentre en esta caché.

La tabla 10.1 muestra las prestaciones que se obtienen en este SMP para el caso de una carga de trabajo típica de una aplicación comercial que carga considerablemente tanto el bus como la memoria del S/390. Como vemos, la penalización de acceso al almacenamiento es el tiempo, en ciclos, transcurrido entre la petición del dato a la jerarquía de cachés, y el momento en el que se obtiene el primer bloque de datos de 16 bytes. La caché L1 muestra un porcentaje de aciertos del 89 %, de forma que el 11% de los restantes accesos a memoria deben resolverse en el nivel L2, L3, o en memoria. De este 11%, un 5% se resuelven en la caché L2, y así sucesivamente. Utilizando estos tres niveles de caché, sólo un 3% de las referencias necesitan acceder a memoria. Sin el tercer nivel, la proporción de accesos a memoria principal se duplicaría. Aunque el porcentaje de aciertos en la cache L3 es muy bajo, su presencia se ve justificada por el ahorro

de ciclos de penalización, ya que de lo contrario este 3% de accesos supondría más del doble de ciclos de penalización.

Subsistema de memoria	Penalización de acceso (en ciclos de reloj)	Tamaño de caché	Porcentaje de aciertos (%)
Caché L1	1	32 KB	89
Caché L2	5	256 KB	5
Caché L3	14	2 MB	3
Memoria	32	8 MB	3

Tabla 10.1. Porcentajes típicos de acierto de caché en una configuración S/390.

10.3. Coherencia de la Caché

En los sistemas multiprocesador actuales, una organización esencial para conseguir unas prestaciones razonables es disponer de uno o dos niveles de caché asociados a cada procesador. Esto, no obstante, origina un problema, conocido como el problema de la 'coherencia de caché'. La esencia del problema es la posibilidad de que existan varias copias del mismo dato simultáneamente en diferentes cachés. En este caso, si los procesadores actualizan sus copias, puede producirse una visión inconsistente de la memoria.

Cuando se utilizan cachés, existen dos políticas de escritura usuales:

- **Post-escritura (*Write back*):** Las operaciones de escritura se realizan usualmente sólo en la caché. La memoria principal sólo se actualiza cuando la línea de caché correspondiente se reemplaza.
- **Escritura directa (*Write through*):** Todas las operaciones de escritura se realizan en memoria principal a la vez que en la caché, asegurándose así de que el contenido de la memoria principal siempre es válido.

Resulta evidente que una política de post-escritura puede ocasionar inconsistencia. Por ejemplo, supóngase que dos cachés contienen la misma línea, y la línea se actualiza en una de ellas, entonces la otra caché tendrá un valor no válido. Por tanto, las lecturas siguientes a dicha línea producirán resultados no válidos. Incluso con la política de escritura directa puede existir inconsistencia, a no ser que las otras cachés comprueben los accesos a la memoria principal o reciban algún tipo de notificación directa de la escritura realizada.

En esta sección se revisan distintas aproximaciones al problema de la coherencia de caché.

El objetivo de un protocolo de coherencia de caché es situar las variables locales utilizadas recientemente en la caché apropiada, y mantenerlas allí para las distintas escrituras y lecturas, al mismo tiempo que se mantiene la consistencia de las variables compartidas que pudieran encontrarse en varias cachés al mismo tiempo.

Existen dos aproximaciones básicas para minimizar los problemas asociados a la coherencia de los datos en las cachés de primer o de segundo nivel de los procesadores constituyentes de un sistema multiprocesador: software y hardware.

10.3.1. Soluciones Software

Intentan evitar la necesidad de circuitería y lógica hardware adicional, trasladando el problema al compilador y el sistema operativo. Los mecanismos basados en el compilador realizan un análisis del código para determinar qué datos pueden producir problemas, marcándolos de forma que el sistema operativo o el hardware impidan su paso a caché ('non-cacheable'). La solución más sencilla y conservadora sería impedir que cualquier dato compartido pase a caché. No obstante, la compartición puede ser sólo de lectura, o en momentos diferentes de tiempo. De cualquier manera, se reduce el rendimiento debido a caché. Aunque se han desarrollado modelos más eficientes, las soluciones software no resultan demasiado eficientes.

10.3.2. Soluciones Hardware

Éstas son las que generalmente se denominan 'protocolos de coherencia de caché'. Estas soluciones permiten reconocer dinámicamente, en el momento de la ejecución, las situaciones de inconsistencias potenciales. Puesto que el problema se considera sólo en el momento en que aparece, existe un uso más efectivo de las cachés, mejorándose las prestaciones en relación a las aproximaciones software. Además, estas aproximaciones son transparentes para el programador y el compilador, reduciendo la complejidad del desarrollo del software.

Cuestiones asociadas son: a) el lugar donde se encuentra la información de estado de los bloques de datos, b) organización de esa información, c) dónde se impone la coherencia, y d) mecanismos para imponerla.

En general, los esquemas software se pueden dividir en dos categorías: protocolos de directorio y protocolos de sondeo (*snooping*).

10.3.2.1. Protocolos de directorio

Recogen y mantienen la información acerca de dónde residen las copias de los bloques. Usualmente hay un controlador centralizado que es parte del controlador de memoria principal, y un directorio que se almacena en memoria principal. El directorio contiene información de estado global en relación con los contenidos de las diferentes cachés locales. Cuando un controlador individual de una caché hace una petición, el controlador centralizado comprueba y emite las órdenes para la transferencia de datos entre memoria y caché o entre diferentes cachés. Mantiene también actualizada la información de estado, conociendo así qué procesadores tienen una copia de cada bloque, de forma que cualquier acción local que pueda afectar al estado global de una línea, debe comunicarse al controlador central.

Antes de que un procesador pueda escribir en un bloque de su caché, debe solicitar al controlador el acceso exclusivo a dicho bloque. Antes de ceder este tipo de acceso, el controlador envía un mensaje a todos los procesadores con una copia del bloque, forzando a que invaliden su copia. Después de recibir el reconocimiento de invalidación de cada uno de esos procesadores, cede el control exclusivo al procesador que lo solicitó. Cuando otro procesador intenta leer una línea cedida para acceso exclusivo de otro procesador, enviará una notificación de fallo de lectura al controlador. Entonces, el controlador manda una orden al procesador que posee la línea requerida, para que lo vuelva a escribir en memoria principal, y después la línea puede compartirse para lectura por el procesador original y el que solicitaba el acceso.

La principal desventaja de este método es que se convierte en un cuello de botella y hay un alto coste de comunicación entre controladores. Para resultar eficientes, se utilizan con sistemas de gran escala, que disponen de varios buses o estructuras de interconexión complejas.

10.3.2.2. Protocolos de sondeo (Snoopy protocols)

Distribuyen la tarea de mantener la coherencia entre todos los controladores de caché. Cada caché debe reconocer cuando un bloque de los que contiene está compartido con otras cachés. Cuando en una caché se realiza una modificación de un bloque compartido, debe anunciarse a las otras cachés mediante algún mecanismo de difusión (*broadcast*). Cada controlador caché debe sondear (*snoop*) la red para observar las notificaciones que se difunden y actuar en consecuencia.

La difusión (*broadcast*) se implementa más fácilmente en sistemas de bus compartido, dado que este proporciona una forma sencilla de realizar la difusión y el sondeo. Pero, dado que uno de los objetivos de utilizar cachés locales es evitar accesos al bus, hay que limitar su uso de forma que no invaliden el ahorro de bus que implica la utilización de cachés.

Las aproximaciones básicas del protocolo de sondeo son:

- **Invalidar si escritura ('write-invalidate'):** Varios procesadores pueden leer pero tan sólo uno puede escribir en un mismo bloque compartido en un momento dado. Cuando se quiere hacer una escritura en dicho bloque, el controlador primero envía una notificación a los demás, invalidando dicho bloque en las otras cachés y haciéndolo exclusivo. A partir de aquí, el procesador puede hacer escrituras locales en ese bloque hasta que sea solicitado por otra caché.
- **Actualizar si escritura ('write-update'):** Cuando un procesador desea escribir en un bloque compartido, difunde la palabra a escribir para que el resto de controladores la escriban en su caché.

Las prestaciones de estas dos aproximaciones dependen del número de cachés en el sistema y del patrón de lecturas/escrituras, por lo que no puede considerarse ninguna mejor a priori. La primera es la aproximación más utilizada en procesadores comerciales como los basados en Pentium II y PowerPC.

10.3.2.3. Protocolo MESI

Es un ejemplo del protocolo de invalidar si escritura. En este protocolo se marca el estado de cada línea de la caché mediante dos bits adicionales en el campo de etiqueta. Cada línea puede estar en uno de estos cuatro estados: (a) modificado ('Modified', contenido distinto que en memoria principal y sólo disponible en esta caché), (b) exclusivo ('Exclusive', contenido similar que en memoria principal y no está en otra memoria caché), (c) compartido ('Shared': contenido similar que en memoria principal que puede estar en otra caché), y (d) no-válido ('Invalid': no contiene datos válidos). Por esta razón este protocolo se denomina MESI.

La figura 10.5 muestra el diagrama de estados del protocolo MESI. Cada línea de la caché tiene sus propios bits de estado y su propia implementación del diagrama de estados. La figura 10.5a muestra las transiciones que se producen debido a las acciones iniciadas por el procesador al que pertenece la caché. La figura 10.5b describe las transiciones ocasionadas por eventos que se producen en el bus común.

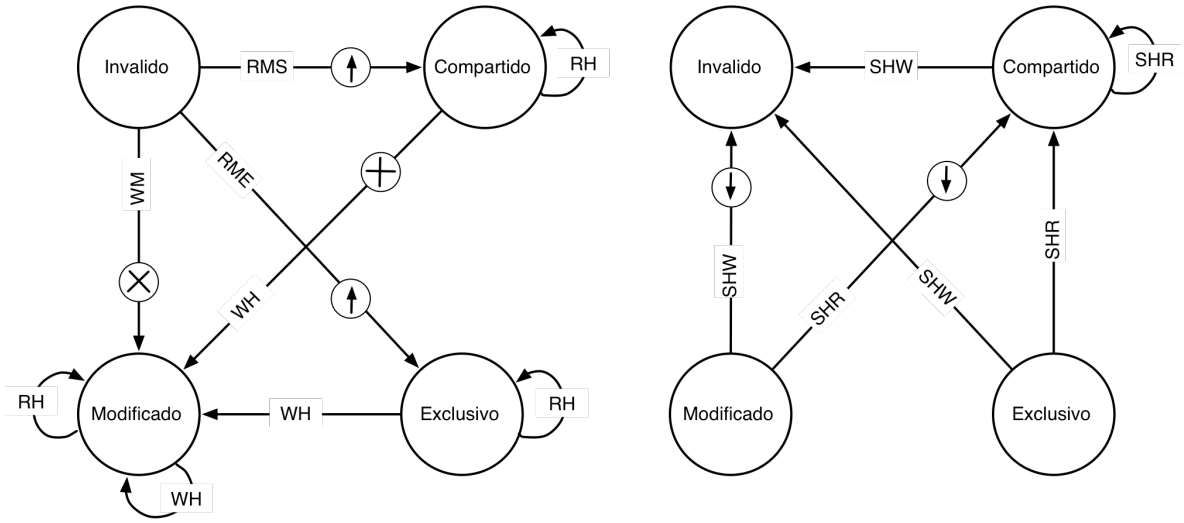

figura 10.5. Diagrama de transición de estados MESI.

Las posibles transiciones del protocolo MESI son:

Fallo de lectura

Cuando se produce fallo de lectura en la caché local, el controlador inicia una lectura en memoria principal. Para ello, inserta una señal en el bus, que alerta a los otros controladores para que sondeen la transacción. Hay varias posibilidades:

- Si un controlador tiene ese bloque en su caché como exclusivo, devuelve una señal indicando que comparte el bloque y lo pasa a estado compartido. El controlador inicial lee el bloque y lo etiqueta como compartido.
- Si uno o varios controladores ya tenían ese bloque en su caché como compartido, mandan una señal para que el controlador inicial lo cargue también como compartido.
- Si un controlador tiene una copia modificada del bloque, le indica al procesador inicial que vuelva a intentar el acceso. Mientras, toma el control del bus, escribe el bloque modificado en memoria principal, y pasa el estado de modificado a compartido. A continuación, el controlador inicial intenta de nuevo la lectura, dándose el caso anterior.
- Si el bloque no está en otra caché, ningún controlador envía señal. El controlador inicial lee el bloque como exclusivo.

Acierto de lectura

Cuando se produce un acierto en una lectura en la caché local, el procesador simplemente lee el dato solicitado. No hay ningún cambio de estado.

Fallo de escritura

Cuando se produce fallo de escritura en la caché local, el controlador inicia una lectura en memoria principal pata traer el bloque sobre el que quiere escribir. Para ello inserta una señal en el bus que indica lectura-para-modificación. El bloque se carga directamente como modificado. Con respecto a las otras cachés, hay dos escenarios posibles previos a la carga del bloque:

- Otro procesador puede haber modificado una copia del bloque en su caché. Entonces, indica al procesador inicial que hay una copia modificada, por lo que el inicial deja libre el bus y espera que el otro escriba en memoria principal su copia modificada y pase su estado a no válido (puesto que el procesador inicial va a modificarla). A continuación, el procesador inicial repite el proceso y copia el bloque en su caché, lo modifica, y lo marca como modificado.
- Ningún procesador tiene copia modificada. En este caso no responden con ninguna señal y el procesador inicial carga el bloque y lo modifica. Mientras tanto, si algún otro procesador tiene copia no modificada (exclusiva o compartida), la pasa a estado no válido.

Acierto de escritura

Hay tres posibilidades, en función del estado del bloque a escribir:

- Compartido: Antes de actualizar, debe conseguir el acceso exclusivo al bloque, por lo que envía una señal al resto. Todo procesador que tenga una copia la pasa de estado compartido a no válido. Después, el procesador inicial modifica el bloque y lo pasa de estado compartido a modificado.
- Exclusivo: modifica el bloque y lo pasa a estado modificado.
- Modificado: sólo actualiza el valor.

10.3.2.4. Consistencia de cachés L1-L2

Hasta ahora se han descrito los protocolos de coherencia de caché en términos de la cooperación entre las cachés conectadas al mismo bus o a otro sistema de interconexión utilizado por un SMP. Normalmente, estas cachés son cachés L2. Sin embargo, en este caso, cada procesador también tiene una caché L1 que no está conectada directamente al bus y que, por lo tanto, no puede participar en un protocolo de sondeo. Esta caché L1 también necesita un esquema para mantener la coherencia de los datos en los dos niveles de caché y en todas las cachés del SMP.

La estrategia utilizada consiste en extender el protocolo MESI (o cualquier otro protocolo de coherencia de caché) a las cachés L1. Cada línea de la caché L1 incluye bits para indicar su estado. Básicamente, el objetivo es que para cualquier línea que esté presente tanto en caché L2 como en su correspondiente caché L1, el estado de la línea en L1 debe seguir el trayecto del estado de la línea en L2. Una forma sencilla de hacer esto es usar una política de escritura directa en la caché L1, pero escribiendo sobre la caché L2, y no sobre la memoria principal. La política de escritura directa en L1 hace que cualquier modificación en una línea de L1 se propague a la caché L2, haciéndose visible a las otras cachés L2.

El uso de la política de escritura directa en L1 requiere que el contenido de L1 sea un subconjunto del contenido de L2. Esto sugiere, además, que la asociatividad de L2 debería ser igual o mayor que la asociatividad de L1. Esta política de escritura directa para L1 es la que se utiliza en el SMP IBM S/390.

Cuando la caché L1 utiliza una política de post-escritura, la relación entre las dos cachés se hace más compleja.