El protocolo de coherencia más popular es el fisgoneo. Cada cache tiene una copia del bloque de memoria física y una copia de su estado de compartición, pero no se mantiene una información centralizada. Las cachés son accesibles mediante algún medio de transmisión (un bus o una red), y todos los controladores de cache monitorizan o fisgan el medio para determinar si tiene una copia de un bloque que está siendo solicitada en un bus o conmutador de acceso.

Ejemplo de un protocolo de invalidación que funciona con un bus de fisgoneo para un bloque de cache (X) con caches de escritura retardadas.

La tabla muestra un ejemplo de protocolo de invalidación para un bus compartido con fisgoneo y caches de escritura retardada. Para comprender como se asegura la coherencia con este protocolo, consideramos una escritura seguida de una lectura por parte de otro procesador: como la escritura requiere acceso exclusivo, cualquier copia de cache del procesador que hace la lectura debe ser invalidada (de ahí el nombre del protocolo). De este modo, cuando se realiza la lectura, se produce un fallo en la cache y se busca una nueva copia del dato. En caso de escritura, el procesador que escribe tiene el acceso exclusivo y se evita así que otro procesador pueda escribir simultáneamente. Si dos procesadores intentan escribir el mismo dato simultáneamente, uno de los dos ganará la carrera, y la copia del otro procesador se invalidará. Para que el segundo procesador complete su escritura, debe obtener una nueva copia del dato, que ya tendrá el valor actualizado. **Así, este protocolo fuerza también las escrituras en serie.**

Suponemos que inicialmente ninguna cache tiene el valor X y que el valor X en memoria vale 0. Se muestran los contenidos de la CPU y memoria una vez completada la actividad del procesador y el bus. Una celda en blanco indica que no hay actividad o no hay una copia en cache. Cuando se produce el segundo fallo del procesador B, la CPU A responde proporcionando el valor solicitado y cancelando el acceso a memoria. Además, se actualizan el contenido de la cache de B y el valor de X en memoria. Esta actualización de memoria, que ocurre cuando un bloque se comparte, simplifica el protocolo, pero permite rastrear el propietario y forzar que la escritura retardada se realice solo cuando se reemplaza el bloque. Para ello, se requiere la introducción de un estado adicional llamado “propietario”, que indica que un bloque puede compartirse, pero el procesador propietario es el responsable de actualizar cualquier otro procesador y la memoria cuando cambia o se reemplaza el bloque.

Dentro de este protocolo hay dos protocolos de escritura:

- **Write-invalidate**: permite múltiples lectores, pero solo un escritor a la vez. Cada escritura de un bloque compartido debe ser precedido de una invalidación de todas las copias de ese bloque, para prevenir el uso de copias obsoletas. Una vez el bloque se ha hecho exclusivo, las escrituras locales pueden continuar hasta que algún otro procesador requiera el mismo bloque. El procesador que escribe distribuye una señal de invalidación sobre el bus, para que las caches comprueben si tienen una copia.

  Algunas máquinas anteriores (el Encore Multimax, por ejemplo) utilizaron este simple mecanismo. Una solución de coherencia tan poco sofisticada da como resultado un

rendimiento deficiente del sistema debido a un uso muy elevado del bus. Solo sistemas con un tamaño muy pequeño de número de procesadores pueden utilizar este esquema.

Write-update: en lugar de invalidar cada copia del bloque compartido, el procesador que escribe difunde el nuevo dato sobre el bus; entonces se actualiza todas las copias con el nuevo valor. Este esquema difunde continuamente escrituras para datos compartidos, mientras que el de invalidación en escritura suprime las demás copias para que solo haya una copia local para escrituras posteriores. Los protocolos de difusión en escritura permiten, habitualmente, que los bloques se identifiquen como compartidos (difundidos) o privados (locales). Este protocolo actúa de forma similar a una cache de escritura directa para datos compartidos (difundiendo a otras caches) y como una cache escritura retardada para datos privados (los datos modificados salen de la cache sólo cuando se produce un fallo).

Los protocolos Dragon y Firefly pertenecen a esta categoría.
Este protocolo funciona mejor en aplicaciones con un intercambio más estricto. El intercambio secuencial y la migración de procesos puede perjudicar gravemente el rendimiento con difusiones de escritura frecuentes e innecesarias.

Patterson An Example Protocol

Un protocolo de coherencia de fisgoneo se implementa usualmente incorporando un controlador de estado en cada núcleo. Este controlador responde a las solicitudes del procesador en el núcleo y del bus, cambiando el estado del bloque de caché seleccionado y utilizando el bus para acceder a los datos o para invalidarlos. Lógicamente, puede pensar en un único controlador asociado a cada bloque; es decir, las operaciones de indagación o las solicitudes de caché para diferentes bloques se pueden procesar de forma independiente. En implementaciones reales, un solo controlador permite múltiples operaciones a distintos bloques para realizarlas en forma intercalada (es decir, una operación puede iniciarse antes de que se complete otra, incluso aunque solo se permita un acceso de caché o un acceso de bus a la vez). Además, recuerde que, aunque nos referimos a un bus en la siguiente descripción, cualquier red de interconexión que admita una transmisión a todos los controladores de coherencia y sus cachés privadas asociadas puede usarse para implementar la indagación.

El protocolo simple que consideramos tiene tres estados: **inválido, compartido y modificado.** El estado compartido indica que el bloque en la memoria caché privada es potencialmente compartido, mientras que el estado modificado indica que el bloque se ha actualizado en la memoria caché privada; tenga en cuenta que el estado modificado implica que el bloque es exclusivo.

La extensión más común de este protocolo básico es la adición de un estado exclusivo, que describe un bloque que no está modificado pero que solo está en una caché privada.

La Figura 5.6 muestra un diagrama de transición de estados para un solo bloque de caché privada usando un protocolo de invalidación de escritura y una caché de escritura retardada. Para simplificar, los tres estados del protocolo se duplican para representar las transiciones

basadas en las solicitudes del procesador (a la izquierda, que corresponde a la mitad superior de la tabla en la Figura 5.5), en oposición a las transiciones basadas en las solicitudes de bus (a la derecha, que corresponde a la mitad inferior de la tabla en la Figura 5.5). Las palabras en negrita se usan para distinguir las acciones del bus, en oposición a las condiciones de las que depende una transición de estado. El estado en cada nodo representa el estado del bloque de caché privado seleccionado especificado por el procesador o la solicitud de bus.

La mayoría de los cambios de estado indicados por arcos en la mitad izquierda de la Figura 5.6 serían necesarios en una memoria caché de escritura retardada de un único procesador, **siendo la excepción la invalidación de un acierto de escritura en un bloque compartido**. Los cambios de estado representados por los arcos en la mitad derecha de la Figura 5.6 son necesarios solo para la coherencia y no aparecerían en absoluto en un controlador de caché de un solo procesador.

Solo hay una máquina de estado por caché, con estímulos provenientes del procesador conectado o del bus. La Figura 5.7 muestra cómo las transiciones de estado en la mitad derecha de la Figura 5.6 se combinan con las de la mitad izquierda de la figura para formar un solo diagrama de estado para cada bloque de caché.

Las acciones en gris en la Figura 5.7, que manejan los fallos de lectura y escritura en el bus, es esencialmente el componente de indagación del protocolo. Otra propiedad que se conserva en este protocolo, y en la mayoría de los otros protocolos, es que cualquier bloque de memoria en estado compartido siempre está actualizado en la cache compartida externa (L2 o L3, o en memoria si no hay cache compartida), lo que simplifica la implementación. De hecho, no importa si el nivel externo de la cache privada es una cache compartida o memoria principal; la clave es que todos los accesos de los núcleos pasan por ese nivel.

Aunque nuestro protocolo de caché simple es correcto, omite una serie de complicaciones que hacen que la implementación sea mucho más complicada. El más importante de estos es que el protocolo asume que las operaciones son atómicas. Por ejemplo, el protocolo descrito asume que los errores de escritura pueden detectarse, adquirir el bus y recibir una respuesta como una sola acción atómica. En realidad, esto no es cierto. De hecho, incluso un fallo de lectura podría no ser atómica; después de detectar un fallo en la L2 de un multinúcleo, el núcleo debe arbitrar para acceder al bus que conecta a la L3. **Las acciones no atómicas introducen la posibilidad de que el protocolo pueda interrumpirse, lo que significa que alcanza un estado en el que no puede continuar.**

Un multiprocesador construido con múltiples chips de múltiples núcleos tendrá una arquitectura de memoria distribuida y necesitará un mecanismo de coherencia de prácticas por encima y más allá de uno dentro del chip. En la mayoría de los casos, se utiliza algún tipo de esquema de directorio.

Snoop filter:

Cuando se produce una transacción del bus a un bloque de caché específico, todos los fisgones deben fisgonear la transacción de bus. Luego, los fisgones buscan su etiqueta de caché correspondiente para verificar si tiene el mismo bloque de caché. En la mayoría de los casos, la caché no tiene el bloque, ya que un programa paralelo bien optimizado no comparte muchos datos entre los subprocesos. Por lo tanto, la búsqueda de etiquetas de caché por parte del

snooper es generalmente un trabajo innecesario para la caché que no tiene el bloque de caché. Además, la búsqueda de etiquetas perturba el acceso al caché por parte de un procesador e incurre en un consumo de energía adicional.

Una forma de reducir la indagación innecesaria es usar un filtro de espionaje. Un filtro snoop determina si un snooper necesita verificar su etiqueta de caché o no. Un filtro snoop es una estructura basada en directorios y supervisa todo el tráfico coherente para realizar un seguimiento de los estados de coherencia de los bloques de caché. Significa que el filtro snoop conoce las cachés que tienen una copia de un bloque de caché. Por lo tanto, puede evitar que las cachés que no tienen la copia de un bloque de caché realicen la indagación innecesaria. Hay dos tipos de filtros dependiendo de la ubicación del filtro snoop. Uno es un **filtro de origen** que se encuentra en un lado de la caché y realiza el filtrado antes de que los tráficos coherentes lleguen al bus compartido. El otro es un **filtro de destino** que se encuentra al lado del bus y bloquea el tráfico coherente innecesario que sale del bus compartido. El filtro snoop también se clasifica como inclusivo y exclusivo. El filtro snoop inclusivo realiza un seguimiento de la presencia de bloques de caché en cachés. Sin embargo, el filtro snoop exclusivo supervisa la ausencia de bloques de caché en cachés. En otras palabras, un éxito en el filtro snoop inclusivo significa que el bloque caché correspondiente está en cachés. Por otro lado, un éxito en el exclusivo filtro snoop significa que ninguna caché tiene el bloque de caché solicitado.

Directorio:

Además del protocolo de coherencia de cache basado en fisgoneo, existe un protocolo basado en directorio.

Mantiene el estado de compartición de un bloque de la memoria física en una localización, llamada directorio. La implementación tiene un sobrecoste ligeramente mayor que el fisgoneo, pero puede reducir el tráfico entre las caches y, por tanto, escalar a un mayor número de procesadores.

El beneficio de usar bus snooping es que es más rápido porque todas las operaciones son una solicitud/respuesta que ven todos los procesadores. Suele ser así si hay suficiente ancho de banda.

María Jesús Peregrina