COHERENCIA DE CACHÉ

Protocolo Snoopy

Adrián José Hurtado García
Juan Manuel Ortíz García-Márquez
Arquitectura de computadores Paralelos y distribuidos

Índice

Introducción....2
Implementación Snoopy....2
Coherencia snooping en un bus simple compartido....2
Problemas y Soluciones....5
Snoopy jerárquico....5
Problema de la jerarquía....6
Bibliografía....7

Introducción

En el protocolo Snoopy la memoria se conecta a los procesadores para espiarlos a través de buses. Dependiendo lo que encuentre, enviará señales de control a los procesadores para tomar decisiones.

Cuando se detecta una actualización de un dato, o se invalidan las copias (protocolo de invalidación) o se actualizan las copias (protocolo de actualización).

Implementación Snoopy

Dentro de la implementación del protocolo Snoopy podemos encontrar distintas formas de llevar a cabo de su implementación en un computador. Nosotros veremos cómo realizar la coherencia de la implementación snoopy en un bus simple compartido.

Coherencia snooping en un bus simple compartido

El concepto de su funcionamiento es fácil de entender pues, todos los procesadores y todas las memorias pueden ver las transacciones que se están realizando entre las distintas memorias y memorias cachés del sistema. El controlador de caché del bus supervisa las etiquetas de las líneas involucradas y reacciona si es necesario comprobando el contenido y el estado de dicho elemento en la memoria local. Este bus proporciona un punto de serialización entre dos elementos A y B antes o después de realizar la lectura/escritura del dato en la memoria.

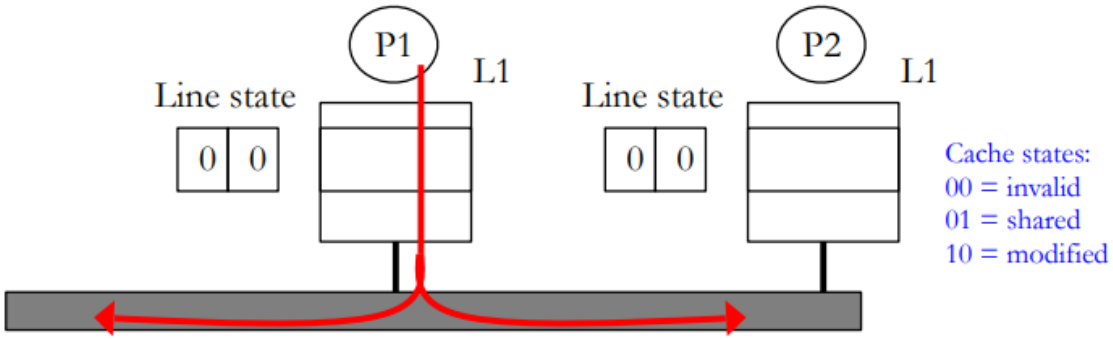

Ilustración 1 - Bus simple compartido

Este sería el resumen de un bus simple compartido, en este caso, por dos procesadores P1 y P2. Cada procesador posee su propia línea de estado que puede tomar los valores: 00 (inválido), 01 (compartido) y 10 (modificado).

Entonces, ¿cuándo provee la memoria el dato solicitado? Primero debemos esperar a que se deshabilite la invalidez del dato, y una vez que esto se produzca, deberemos mirar que el estado actual del dato en memoria no sea ni compartido ni modificado. Si no estuviera en alguno de esos estados, el dato estaría listo para enviarse al procesador, pero, si por el contrario estos datos son los que se encuentran en la línea de estado, los datos quedarán marcados como inválidos y no se realizará el envió de datos que se ha solicitado por parte del procesador.

Todo esto genera un problema de escritura atrasada, por lo que se implementa un buffer *Write-back*

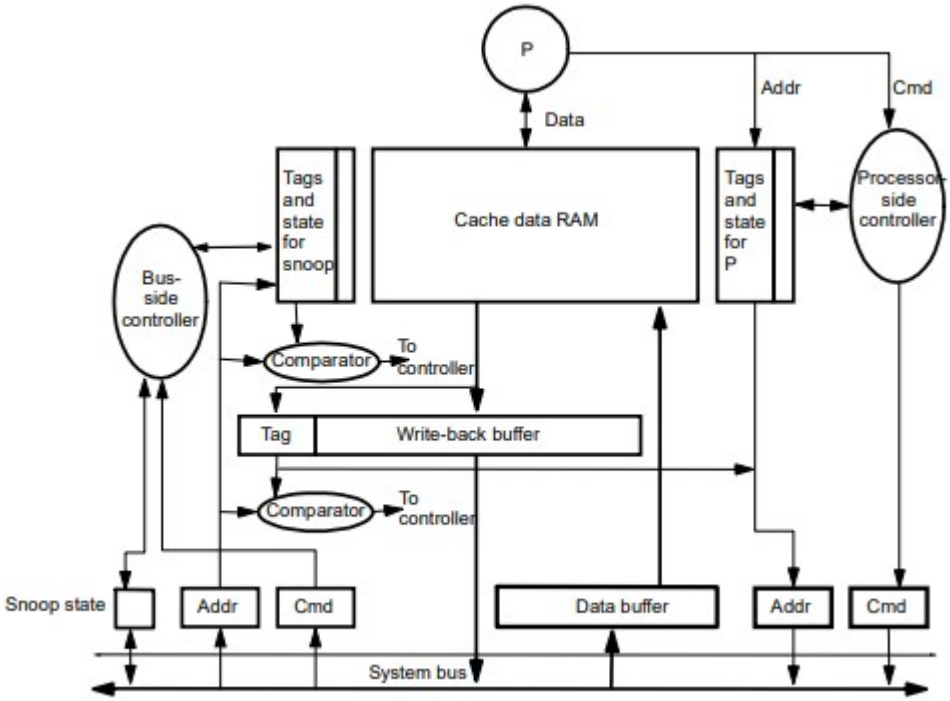

Ilustración 2 - Buffer Write-back

Con el buffer write-back, los datos inicialmente sólo se escriben en la caché. El dato escrito aparece reflejado en la memoria principal sólo cuando el bloque que lo contiene es reemplazado. Se incorpora un bit más (bit dirty), el cual se pone a uno cada vez que la CPU escribe en el bloque de caché.

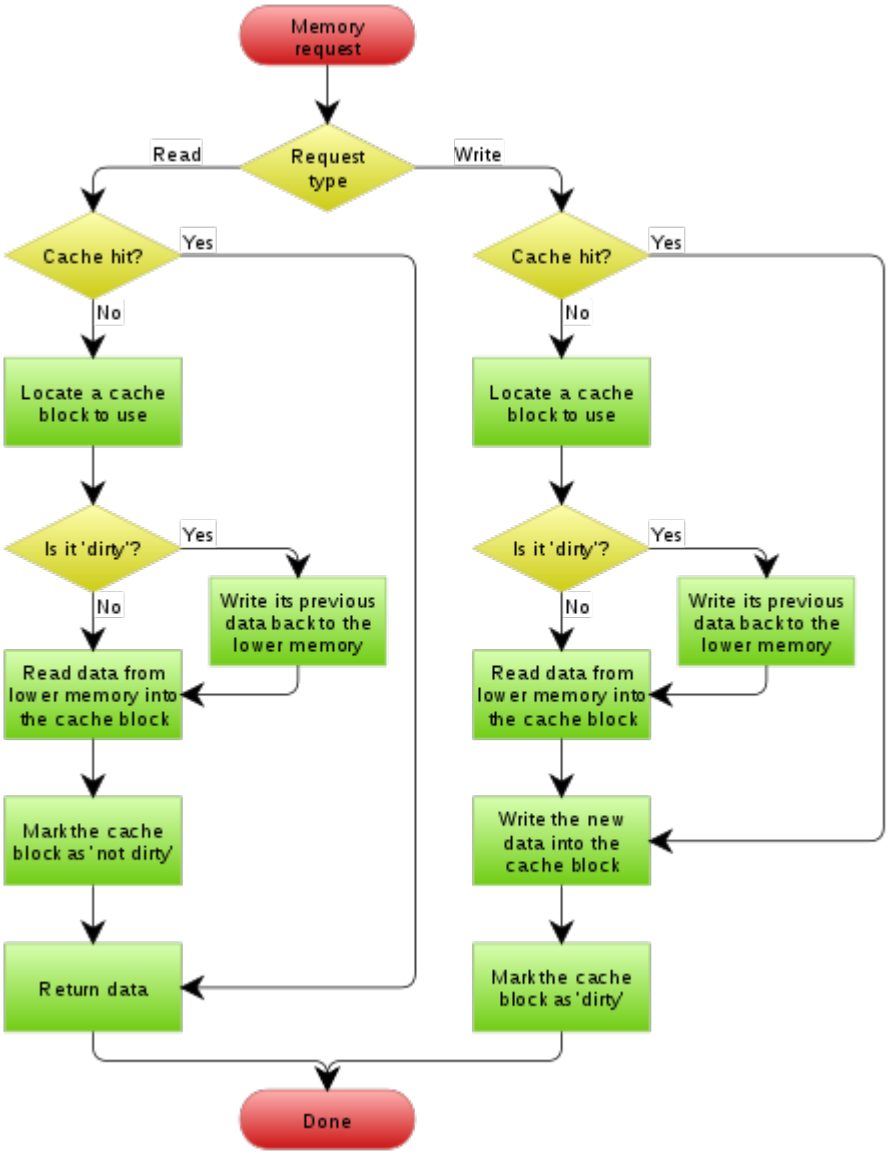

Ilustración 3 - Estructura funcionamiento Write-back

Tenemos una caché con 3 bits extras, estos bits son:

- V (Valid) = Movimiento de datos válido.
- D (Dirty) = significa que el dato en la caché no es el mismo que en la memoria.
- S (Shared) = Compartida.

Los anteriores estados son los que puede tomar la caché. Esta caché contiene un valor que puede ser leído o escrito (Read/Write). Al escribir sobre la caché, estamos cambiando el valor de esta. Cada valor estará o en memoria principal (lenta de acceder) o en una o más cachés locales (rápidas de acceder). Cuando un bloque es cargado en la caché, esta es cargada como válida.

Si a la hora de realizar una lectura, esta es errónea, la solicitud de lectura se transmite al bus. Todos los controladores de las distintas memorias caché que posee el sistema monitorizan el bus. Si uno de los controladores ha guardado esa dirección en caché y su estado es *Dirty,* cambia el estado a *Valid* y envía la copia al nodo solicitante.

El estado Valid significa que la línea de caché es actual. Pero si se produce un error a la hora de escribir (intentar escribir ese valor pero no se encuentra en la caché), el bus snoopy garantizará que las otras cachés se configuren como no válidas. Este estado de No valida significa que una copia solía existir en la caché pero ya no es la que se encuentra actualmente.

Problemas y Soluciones

1º Problema: Surge cuando procesador y bus acceden a memoria caché.
Solución: Se duplica el directorio de la caché, no es solución total pero reduce los conflictos.

2º Problema: En una petición BR, la memoria principal es muy lenta y la caché implica complejidad.
Solución: Se solicita a MP el bloque, si la caché proporciona antes se cancela la lectura a MP, si MP acaba antes se espera hasta que cache confirmen que no hay ningún M.

3º Problema: Las operaciones pueden solaparse generando grandes conflictos mezclando acciones de distintos procesadores sobre el mismo bloque.
Solución: Asegurar la atomicidad, es decir ejecutar una operación sin interferencias.

Snoopy jerárquico

Un snoopy jerárquico hace referencia a la posibilidad de tener varios sistemas de coherencia de caché unidos entre sí. Estas cachés estarán unidas por un bus con suficiente ancho de memoria para poder manejar a todos los procesadores a la vez. Este ancho de banda debe ser escalable y con facilidad de expansión.

Para tratar esta escalabilidad utilizamos la arquitectura NUMA (Non-Uniform Memory Access). Esta arquitectura se basa en el principio de memoria distribuida

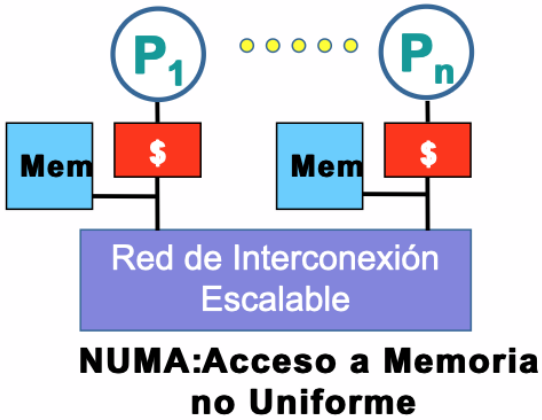

Ilustración 4 - NUMA

Problema de la jerarquía

Un problema que podríamos tener es la coherencia de los datos por ejemplo en las memorias cachés L1 y L2. Los datos que se manejan en la caché L1 podrían no ser visibles en la caché L2.

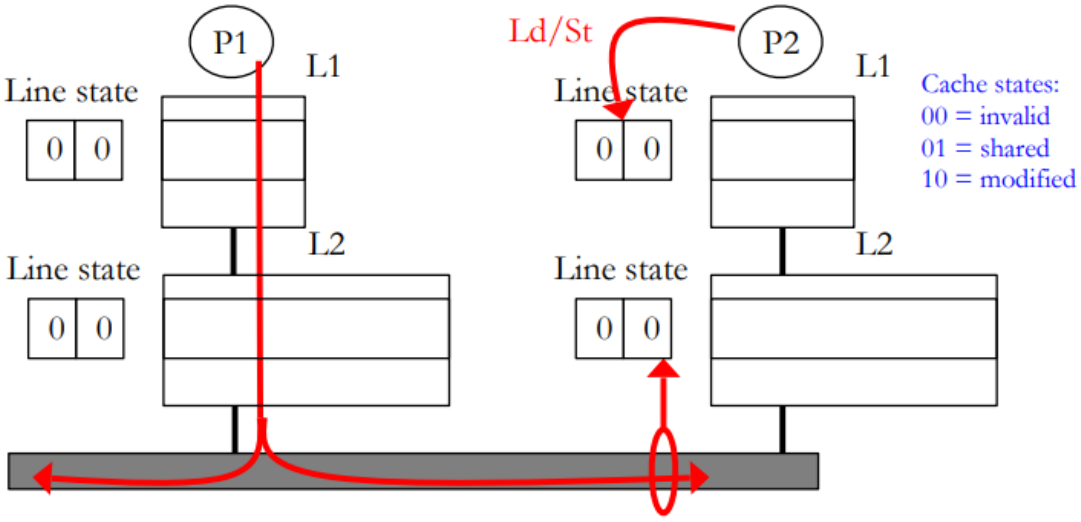

Ilustración 5 - Jerarquía de caché

¿Cómo podemos solucionar este problema de inconsistencia? Manteniendo la propiedad de inclusión. Esto es que las líneas en la caché L1 deberán estar en la caché L2. Gracias a esta redundancia de datos nos aseguramos de que no haya riesgo de perder datos relevantes. Otra posible solución sería propagar las transacciones de coherencia a L1 también. Propagar todas, sin importar si son o no relevantes. Habría que añadir un bit más a los estados de L2 para saber si los datos están o no en L1. Si está presente en L2, pero los bits de inclusión dicen que no están en L1, no hay necesidad de propagar la transacción.

Bibliografía

- http://www.inf.ed.ac.uk/teaching/courses/pa/Notes/lecture05-snooping.pdf
- http://15418.courses.cs.cmu.edu/fall2016content/lectures/12_snoopimpl/12_snoopimpl_slides.pdf
- https://en.wikipedia.org/wiki/Bus_snooping
- https://en.wikipedia.org/wiki/Cache_(computing)#Writing_policies