UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Materia: Sistemas Operativos

Profesor: Gunnar Eyal Wolf Iszaevich

MECANISMOS PARA MANTENER LA COHERENCIA EN CACHÉ

- Ayala Hernández María Fernanda
- Portilla Hermenegildo Elizabeth

Semestre 2025-2

Grupo: 06

Ayala Hernández María Fernanda
Portilla Hermenegildo Elizabeth

Mecanismos para mantener la coherencia en caché

Puntos a Abordar:

- ¿Qué es la memoria caché?

  Antes, la velocidad del procesador y la memoria principal estaba equilibrada, pero con el tiempo la CPU se volvió mucho más rápida que la memoria, generando una brecha de rendimiento. Cuando el procesador solicita datos no disponibles de inmediato, debe detenerse (stall), lo que desperdicia tiempo.
  La respuesta para reducir esa espera es la memoria caché. Esta es una memoria de alta velocidad, temporal, de menor capacidad, de rápido acceso, situada entre la memoria principal y el procesador, y diseñada para resolver las diferencias de velocidad entre una CPU muy rápida y una memoria principal muy lenta. Lo hace almacenando una copia de los datos de uso frecuente. De esta forma, cuando el procesador necesita acceder a cierta información, primero busca en la memoria caché. Si los datos están allí (lo que se conoce como *cache hit*), el acceso es mucho más rápido; si no están (*cache miss*), entonces se recurre a la memoria principal.

  La programación y transferencia de información entre la caché y la memoria principal se realiza automáticamente por el hardware. Cuando la CPU lee datos o instrucciones, simultáneamente guarda esa información en un bloque de caché. De este modo, si el procesador necesita acceder a esos mismos datos nuevamente, puede obtenerlos directamente del bloque de caché correspondiente.

  Por esta razón, durante operaciones frecuentes, es fundamental que el contenido de la memoria caché y el de la memoria principal se mantengan consistentes, ya que cualquier desincronización puede afectar negativamente el rendimiento del sistema.

  El intercambio de datos entre la caché y la memoria principal se realiza en unidades llamadas bloques (o líneas de caché). Lógicamente, la memoria caché está dividida en varias líneas de caché, cada una correspondiente a un conjunto específico de direcciones de memoria. Por lo tanto, la mínima unidad de intercambio de datos entre la memoria caché y la principal es la línea de caché.

- ¿Qué es la coherencia en caché y su importancia?

  La coherencia de caché se refiere a la consistencia y sincronización de los datos almacenados en diferentes memorias caché dentro de un sistema multiprocesador o multinúcleo.

  En sistemas multiprocesador o multinúcleo, cada procesador tiene su propia caché. Aunque esto mejora el rendimiento, plantea un problema: mantener la consistencia de los datos compartidos. Si varias cachés almacenan copias del mismo bloque de memoria y una de ellas lo modifica, deben actualizarse las demás para evitar inconsistencias.

  Este fenómeno se conoce como el problema de coherencia de caché. Si no se controla, puede generar errores en la ejecución del software o incluso corrupción de datos.

  Para garantizar que todas las cachés tengan versiones actualizadas y consistentes de los datos compartidos, se emplean protocolos de coherencia, que coordinan las operaciones entre cachés y aseguran que el acceso a los datos compartidos sea siempre consistente y confiable en todo el sistema.

- Mecanismos de coherencia en caché

A. Fisgones (snooping)

  En sistemas multiprocesador, los protocolos de coherencia de caché basados en bus compartido utilizan la técnica de snooping para mantener la consistencia entre cachés. Esta técnica permite que cada procesador supervise el bus en busca de transacciones de memoria, e invalide sus propios datos si detecta que otro procesador ha modificado una dirección compartida (invalidación cruzada).
  Aunque efectivos, estos protocolos pueden disminuir el rendimiento debido al aumento del tráfico en el bus y a las pérdidas de caché causadas por las invalidaciones.
  Existen dos políticas principales:

1. *Política de escritura-invalidación:* al escribir en un bloque, se invalidan las copias en otras cachés.

2. *Política de escritura-actualización:* al escribir, se actualizan todas las copias del bloque en las demás cachés, reduciendo transferencias futuras pero aumentando el uso del bus.

B. Por directorio:

En arquitecturas donde no hay bus compartido, se usan protocolos basados en directorios. Un directorio lleva un registro de qué procesadores tienen una copia de cada bloque de memoria y en qué estado. Cuando una caché desea acceder o modificar un bloque, primero debe consultar el directorio.
Cada entrada del directorio corresponde a un bloque de memoria que contiene punteros (los cuales indican en qué caches se encuentra una copia de ese bloque) y un bit "sucio" (que marca si algún procesador tiene una versión modificada del bloque que aún no ha sida escrita en la memoria principal).
Ya que mencionaremos los dos tipos de organización de directorios, sería bueno explicar un poco más al respecto.

- **Directorio centralizado:** Toda la información se guarda en un solo punto. Es fácil de implementar, pero puede convertirse en un cuello de botella si muchos procesadores acceden a la vez.
- **Directorio distribuido:** Cada módulo de memoria gestiona la información de sus propios bloques. Mejora el rendimiento y la escalabilidad.

Algunas formas de estructurar internamente los directorios son:

- Directorios full-map: Cada entrada del directorio tiene N bits, siendo N el número de procesadores. Cada bit indica si un procesador tiene una copia válida del bloque.
  Se incluye un bit extra de "sucio".
- Directorios limitados: En lugar de usar N bits, se asigna un número fijo de bits por entrada, independientemente del número de procesadores.
- Directorios encadenados: Se permite un número ilimitado de copias. La información se organiza como una lista enlazada distribuida entre la memoria principal y las cachés.El primer puntero está en la memoria principal y apunta a la primera caché con copia, que a su vez apunta a la siguiente, y así sucesivamente.

C. Otros protocolos basados en snooping:

- MSI: Es uno de los protocolos más básicos, cada línea de caché puede estar en uno de los 3 estados.
  M- Modified: El bloque ha sido modificado en la caché y ya no coincide con la memoria principal. Solo esa caché posee el bloque. Antes de que otra caché lo lea, debe escribirse en memoria (write-back).
  S- Shared: El bloque puede estar presente en varias cachés y coincide con la memoria (está sin modificar). Se permiten operaciones de lectura, pero para escribir se debe invalidar primero en bloque en las demás cachés (otras copias).
  I-Invalid: El bloque no es válido en la caché.

- MESI(ILLINOIS): Protocolo que es una extensión del MSI añadiendo además el estado Exclusive.
  E- Exclusive: Solo está el bloque en este caché y coincide con la memoria.
  Se puede modificar localmente (ya que no hay copias), pasando a Modified sin invalidar otros cachés.
  Esto evita el tráfico en el bus al generar modificaciones.

- MOSI(BERKELEY): También se deriva de MSI pero añade el estado Owned.
  O-Owned: El bloque está modificado, pero puede compartirse. Está caché es responsable de responder a otros procesadores si lo solicitan, sin necesidad de escribirlo en memoria.
  Evitándose entonces la escritura inmediata en memoria al compartir bloque modificados entre cachés.

- MOESI: Es una versión más compleja y eficiente de MESI pues incorpora el estado Owned, esto permite mayor eficiencia al compartir bloques modificados sin recurrir al acceso a memoria principal. Similar al caso de MOSI, permite que un bloque modificado y compartido sea enviado directamente desde una caché a otra. Mejorando el rendimiento al reducir el tráfico al sistema de memoria y aumentando la reutilización de datos compartidos entre procesadores.

- MESIF: Extiende MESIF con el estado Forward.
  F-Forward: Entre todas las cachés que tienen un bloque en estado Shared, una sola se designa como la "reenviadora" oficial.
  Esta caché es la única que responde a solicitudes externas del bloque, evitando que múltiples respondan al mismo tiempo.

- Actividad planeada

  La actividad que se planea presentar tiene como objetivo demostrar, mediante un ejemplo práctico en C++, cómo pueden surgir problemas de coherencia en caché cuando múltiples hilos acceden y modifican variables compartidas sin el uso de mecanismos adecuados de sincronización.
  En sistemas multiprocesador, cada núcleo suele tener su propia caché. Si varios hilos acceden a variables compartidas sin coordinación, pueden generarse inconsistencias entre los valores en memoria principal y los almacenados en caché. Este experimento simula dicha situación, evidenciando que un hilo puede ver valores desactualizados o inconsistentes, a pesar de que otro hilo ya haya realizado modificaciones.

Bibliografía:

[1] A. C. J. Lisandro and S. M. R. Leonardo, "Cache memory coherence protocol for distributed systems." https://ve.scielo.org/scielo.php?script=sci_arttext&pid=S0254-07702007000200008

[2] «Memorias», *Arquitectura de Computadoras. Licenciatura En Informática A Distancia*. http://ecampus.fca.unam.mx/ebook/imprimibles/informatica/arquitectura_computadoras/Unidad_7.pdf (accedido 17 de abril de 2025).

[3] G. Wolf, E. Ruiz, F. Bergero, y E. Meza, *Fundamentos de sistemas operativos*, 1.a ed. Ciudad de Mexico, Universidad Nacional Autónoma de México, Instituto de Investigaciones Económicas : Facultad de Ingeniería, México, 2015.

[4] Universidad Europea de Madrid, *Multiprocesadores Coherencia De Cache*. [En línea]. Disponible en: https://www.cartagena99.com/recursos/alumnos/apuntes/ININF1_M10_U4_T3.pdf

[5] Universidad Europea de Madrid, *Coherencia de cachés*. [En línea]. Disponible en: https://www.cartagena99.com/recursos/alumnos/apuntes/INF_EST_COM_U6_R2_T_PDF.pdf

[6] «Cache coherence - redis», *Redis*, 2 de junio de 2023. https://redis.io/glossary/cache-coherence/

[7] «Coherencia de memoria caché». IBM, 16 de enero de 2025. https://www.ibm.com/docs/es/aix/7.3?topic=architecture-cache-coherency

[8] K. E. Limited, «Introduction to Cache Coherence with Solutions», 15 de septiembre de 2023. https://www.linkedin.com/pulse/introduction-cache-coherence-solutions-kuke-electronics

[9] P. Pandey, «Cache coherence», *SlideShare*, 11 de noviembre de 2019. https://www.slideshare.net/slideshow/cache-coherence-192322944/192322944

[10] M. D. Shirayuki, «4.3.2 Coherencia de caché.pptx», *Scribd*. https://www.scribd.com/presentation/242775447/4-3-2-Coherencia-de-cache-pptx

[11] «Mecanismos para mantener la coherencia en caché», *Slideserve*, 01 de agosto de 2014. https://www.slideserve.com/bina/sistemas-operativos-6-3-mecanismos-para-mantener-la-coherencia-en-cach