TEMA 3:

ARQUITECTURAS CON PARALELISMO A NIVEL DE THREAD (TLP)

Lección 3: CONSISTENCIA DEL SISTEMA DE MEMORIA.

Consistencia de memoria: varios procesadores modifican simultáneamente un dato, ¿en qué orden se deben hacer visibles estas escrituras al resto de procesadores?

Coherencia de caché: El problema aparece en el momento en el que introducimos la caché. Un mismo dato puede estar replicado en varias cachés a la vez. Si un procesador modifica el dato, éste debe actualizar o invalidar el resto de cachés. El cambio debe hacerse visible al resto de procesadores en un tiempo finito.

El modelo de **consistencia de memoria** de una herramienta de programación o de un computador es un conjunto de normas que nos indican que valores deberían ver los procesadores en sucesivas lecturas de una o varias direcciones de memoria. Especifica el orden en el que se realizan las operaciones de memoria, operaciones a las mismas o distintas direcciones y emitidas por el mismo o distinto flujo de instrucciones.

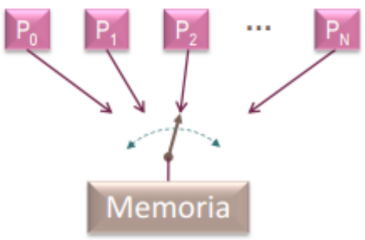
No debe confundirse con el concepto de coherencia. La coherencia solo abarca operaciones de memoria realizadas en una misma dirección de memoria, mientras que la consistencia hace referencia a todas las direcciones de memoria.

La **consistencia secuencial** es modelo de consistencia que espera el programador de las herramientas de alto nivel. El resultado que esperaría el programador por lógica obtener de la ejecución paralela de un programa es el que se obtiene si la consistencia es secuencial. Garantiza que los valores que se ven en memoria se vean en el mismo orden para todas las CPUs. No importa si es en un orden diferente al que se hicieron las escrituras.

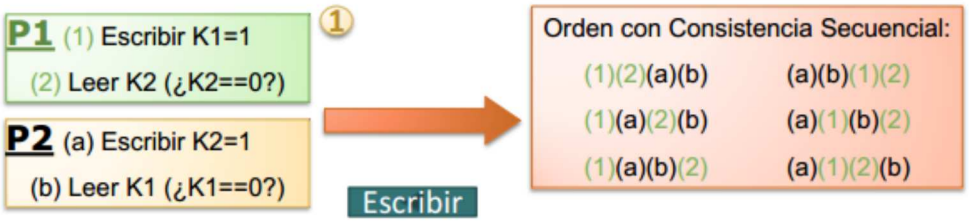
Se deben cumplir dos condiciones para mantener la consistencia secuencial:

- Orden del programa: las operaciones de memoria que ejecuta cada flujo de instrucciones deben realizarse en el orden descrito por el código que ejecuta el flujo de instrucciones.
- Atomicidad / Ejecución atómica: Todos los flujos de instrucciones deben ver el mismo orden en todos los accesos a memoria como si la ejecución de las operaciones de memoria fuese atómica; es decir, como si se ejecutaran de forma instantánea. Esta condición garantiza que todas las operaciones de memoria parecen ejecutarse una detrás de otra en serie, aunque en distintas ejecuciones puedan hacerlo en distinto orden.

La consistencia secuencial presenta el sistema de memoria a los programadores como una memoria global conectada a todos los procesadores a través de un conmutador central.



Ejemplo de consistencia secuencial: Se pueden ejecutar en cualquier orden.



Ningún modelo de consistencia a nivel del hardware garantiza consistencia secuencial debido a la penalización en tiempo de ejecución que supondría no poder alterar el orden de los accesos a memoria independientes del código que ejecuta un flujo. La penalización se agrava si no se permite que las lecturas adelanten a escrituras anteriores. Una escritura requiere transferencias a través de la red. Estas transferencias incrementan su tiempo de ejecución. Los compiladores también reordenan acceso a memoria independientes.

Un **modelo de consistencia relajado** se caracteriza por:

- Los órdenes entre las operaciones de memoria que no garantiza:
  - Orden del programa: Pueden relajar orden del programa. Los modelos relajados pueden permitir que en el código ejecutado en un procesador no se garantice el orden del programa de dos accesos a memoria a distintas direcciones. Difieren en cuanto a los órdenes que permiten relajar; pueden alterar el orden entre:
    1. W ⟶ R: Una escritura y una lectura posterior en el código
    2. W ⟶ W: Una escritura y una escritura posterior.
    3. R ⟶ R: Una lectura y una lectura posterior.
    4. R ⟶ W: Una lectura y una escritura posterior.
  - Atomicidad: Pueden relajar orden global. Hay modelos que permiten que un procesador pueda ver lo que otro ha escrito antes de que esta escritura sea visible al resto de flujos; es decir, antes de que se haya completado globalmente.
- Los mecanismos que hay que añadir para garantizar un orden entre operaciones de memoria cuando resulta necesario para la correcta ejecución de los programas.
- Permiten la finalización anticipada de una operación por parte del procesador.
- ¿Por qué son interesantes los modelos relajados? Porque el modelo de consistencia secuencial produce un cuello de botella, es decir, se comporta como si solo hubiese un único módulo de memoria el cual serializa los accesos.

Modelo que relaja W ⟶ R.

- Permiten que una lectura pueda adelantar a una escritura previa en el orden del programa; pero evita dependencias RAW.
  - Lo implementan los sistemas con buffer de escritura para los procesadores. Desde este buffer se van llevando las escrituras a caché. Hasta que una escritura no deja el buffer no es visible al resto de procesadores. Una lectura puede adelantar a las escrituras del buffer. Además, si un procesador ejecuta una lectura habiendo escrituras en su buffer a la misma dirección, se permite que se pueda obtener el valor de la lectura directamente de la escritura del buffer más reciente a la misma dirección antes de que esta escritura se haya realizado y la puedan ver el resto de los procesadores.

- Para garantizar un orden se pueden usar las instrucciones de lectura-modificación-escritura atómica de una variable. Estas instrucciones, que se introdujeron en los procesadores para implementar funciones de sincronización, permiten garantizar en estos casos el orden W⟶ R porque llevan a caché todas las escrituras pendientes en el buffer de escrituras.

Modelo que relaja W⟶ R y W ⟶W.

- Tiene un buffer de escritura que permite que lecturas adelanten a escrituras en el buffer.
- Permiten que el hardware solape escrituras a memoria a distintas direcciones, de forma que pueden llegar a la memoria principal o a cachés de todos los procesadores fuera del orden del programa.
- En sistemas con este modelo se proporciona hardware para garantizar los dos órdenes. Los sistemas con Sun Sparc implementa un modelo de este tipo.

A los modelos de consistencia que no garantizan ningún orden entre operaciones de acceso a memoria en el código que ejecuta un procesador y que ofrecen instrucciones máquina para establecer el orden se les llama **modelos de consistencia de ordenación débil**.

- La primitiva de sincronización unlock( ) se implementa de forma que los accesos a memoria previos se completen antes que ella y los que vienen detrás solo empiecen cuando termine unlock().

Relaja todas las ordenaciones (RW ⟶ RW).

Tiene una variable de sincronización S que respeta:

- S ⟶ WR
- WR ⟶ S

A los modelos de consistencia que no garantizan ningún orden entre operaciones de acceso a memoria en el código que ejecuta un procesador se le denomina **modelo de consistencia de liberación**. En este modelo se permite aprovechar el paralelismo a nivel de instrucción en mayor grado que el modelo de ordenación débil. Se tiene en cuenta el código de sincronización de adquisición que se usa antes de acceder a memoria compartida, y el de liberación que se usa después de acceder a memoria compartida. Para ello se añaden instrucciones máquina de acceso a memoria de adquisición que garantizan un orden entre el acceso de esta instrucción y de instrucciones de memoria posteriores, e instrucciones máquina de acceso a memoria de liberación que garantizan un orden entre accesos a memoria y anteriores y el acceso a memoria de la propia instrucción con acceso de liberación.

- La primitiva de sincronización unlock( ) se implementa de forma que los accesos a memoria previos se completen antes que ella.

Se respetan:

- SA ⟶ WR: No se puede sobrepasar a SA.
- WR ⟶ SL: SL no puede sobrepasar las que tiene delante.