RESPUESTAS

La coherencia es una propiedad que cumple una variable si y sólo si el valor asociado a esta variable es el mismo en todos los niveles jerárquicos de la memoria. Los problemas de coherencia aparecen al aumentar la cantidad de cores en el sistema. Dado que para un MONOCORE no habría problemas con tener incoherente en el cache l1 por ejemplo un dato que se está modificando y su política es write back (dado que si la WRITE POLICY es WRITE-THROUGH sería siempre coherente el dato, con el absurdo uso del bus en todos los niveles de la jerarquía). Entonces, si agregamos más cores al sistema, por ejempo en un sistema SMP (simmetrical multi-procesing) en estos casos podríamos tener dos cores haciendo uso de la misma línea de cache y si no existen reglas para tratar con estos datos de manera coherente, podríamos perder información entre otras cosas. A partir de esta necesidad de mantener la coherencia, surgen los PROTOCOLOS DE COHERENCIA. Esto es, una serie de acciones que deben cumplirse para mantener la coherencia de las variables.

Las WRITE-POLICIES son dos:

* WRITE-THROUGH : siempre que se altere la coherencia del dato (cpu-write) se hace la copia correspondiente en el nivel inferior inmediato de la jerarquía. Uso reiterativo del bus. Mejor para un MONOCORE . Aunque en SMP para transferencias de L1 a L2 a veces no afecta la performance.
* WRITE-BACK (también se la llama COPY-BACK): el core que contiene la línea puede modificarla cuantas veces quiera sin necesidad de escribir en niveles inferiores de la jerarquía en cada modificación. La línea se mantiene DIRTY (incoherente con el resto de las copias) pero no usa reiteradas veces el bus. Sólo al momento de hacer el REPLACEMENT. SMP multicore.

Para utilizar WRITE-BACK en un sistema SMP tendríamos que implementar un PROTOCOLO DE COHERENCIA para no manejar copias incoherentes entre los cores del sistema. Se podría introducir un PROTOCOLO MESI donde los cores que la tienen exclusive la escriben, si la línea es compartida, deben pedir la exclusividad mediante línea RFO y luego sí escribirla. Acá nos vamos a ahorrar toda la explicación de este protocolo y nos vamos a enfocar en el WRITE-BACK. Se hace escritura a niveles inferiores cuando:

* Si la línea está MODIFIED y alguien la quiere leer. Es decir en el bus se pone un READ-MISS. Abortamos la lectura, escribimos, y listo.
* Si está MODIFIED y alguien quiere escribir. Abortamos lectura del dato (asumimos una política WRITE-ALLOCATE), escribimos (write-back) y como ese dato lo va a escribir otro, el nuestro ya no sirve -> INVALID.
* Si la línea está SHARED o EXCLUSIVE, nunca hacemos write-back pues está clean.

Snooping es una clase de PROTOCOLO DE COHERENCIA. Estos puede ser de tipo SNOOPY o DIRECTORY BASED. Para el primero se utiliza un bus que espía las direcciones del bus del sistema del cual se puede saber qué están leyendo los otros cores. Esto permite el WRITE-INVALIDATE y la implementación de PROTOCOLOS como MSI, MESI, MESIF, MOESI.

El protocolo MESI, es un protocolo de coherencia de 4 estados:

* INVALID : la línea en el cache está inválida.

Frente a un READ-MISS se pasa a EXCLUSIVE. Por qué? Esto es porque al enviar en el bus la dire a leer, todos los cores que tengan esta línea la van a pasar a tener en :

* + SHARED si la tenían clean y en exclusive
  + SHARED si ya estaba ahí
  + INVALID si la tenían MODIFIED dado que hacen el write-back para que yo pueda leer la última versión y ellos pasan a tenerla EXCLUSIVE, pero por el hecho de que yo la estoy pidiendo, la ponen SHARED

Todo es tomado por el SNOOP-BUS hacia el controlador cache y la línea SHARED (un bus bidireccional que pasa por todos los cores, más específicamente, al controlador cache de cada core).

Frente a un WRITE-MISS pasamos a MODIFIED. Cómo? Bueno, en principio hay que traerla al cache, entonces WRITE-ALLOCATE. Lo mismo que un READ-MISS, la tenemos EXCLUSIVE y la escribimos.

* MODIFIED : es un estado de línea de cache en el que la variable está DIRTY. Si alguien la quiere leer (READ-MISS en el bus) abortamos el acceso a memoria, hacemos el write-back y la ponemos en EXCLUSIVE. Si alguien la quiere escribir (WRITE-MISS en el bus) la escribimos en memoria, y la invalidamos pues esta ya no sirve dado que otro la quiere escribir. Si tenemos un CONFLIC READ-MISS es decir no tenemos más lugar en nuestro cache y queremos leer algo, la desalojamos haciendo WRITE-BACK previamente y hacemos el PLACEMENT de la nueva; detalle : si esta que traemos se va a EXCLUSIVE y todos los demás que la tienen, la ponen SHARED si la tenían EXCLUSIVE