Tarea 1
Protocolos de coherencia de cache

Emannuel Esquivel-Lopez
email: ema11412@estudiantec.cr
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Abstract*—The study of the high efficiency of programs is currently a very important issue, so this also involves the various error control methods, mainly generated in the systems cache, which is why there are various protocols for controlling the coherence of cache, such as MESI or MOESI, which allow us to control this type of errors and thus efficiently manage the resources of the programs, as well as increase efficiency, so its study is very important and how it is also structured implementation.**

***Palabras clave*—MESI, MOESI, cache, coherency**

I. Introducción

Los sistemas actuales son extremadamente complejos a nivel de hardware ya que llevaron muchos años a su elaboración, por lo que un elemento muy importante para su estudio, el cual es el procesador, este es el encargado de las operaciones elementales y logica interna del computador, además de esto el cache es un elemento también el cual interactúa directamente con el procesador, por lo que lo hace una memoria extremadamente rápida comparándola por ejemplo con la memoria principal.

El echo de que esta memoria sea una memoria de alta velocidad nos da la facilidad de respuesta inmediata en algunos casos donde esta es utilizada y ayuda a explotar ciertas capacidad, como la localidad de distintos datos y así poder aumentar la eficiencia de algunos programas los cuales sus datos últimamente accesados depende entre si, pero que pasa cuando tenemos mas de un programa o aplicación, llamado $cliente$, los cuales acceden a datos en la cache, los cuales pueden ser datos compartidos o no, llega un conflicto el cual consiste en la incuerencia de cache, o inconsistencia.

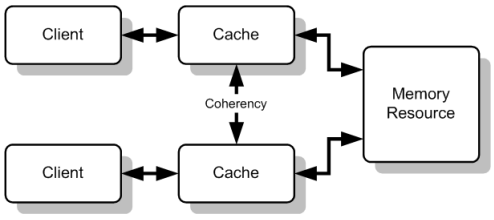

Figure 1. Coherencia de cache

Este problema es generado ya que hay casos donde se debe de leer y/o escribir en cache por distintos procesadores a distintas memoria cache, si solo fuera lectura esta inconsistencia no existiría, por lo que debe existir métodos o protocolos los cuales permitan erradicar estos problemas o bien poder tratarlos de la mejor manera para ya sea el caso algún procesador no accese a datos desactualizados o no esperados por este,estos métodos utilizados son MESI y MOESI, ademas de estos hay un protocolo tambien muy conocido para esto llamado protocolo de directorio.

II. Desarrollo

II-A. MESI

Este protocolo es un protocolo muy importante ya que es de los mas utilizados para el control de la coherencia de cache, este admite write-back (WB) y write-through (WT) para cache.[1]

Su nombre proviene directamente de los estados posibles de linea de cache, esto ya que para la resolucion de este problema se descomponen las lineas de cache en 4 estados los cuales estan basados en las siguientes caracteristicas:

- Modificado (M): Este estado indica que el cache ha sido modificado previamente por lo que los datos deben ser escritos en memoria antes de su próxima lectura.
- Exclusiovo (E): Quiere decir que el cache tiene una copia del bloque, esta no ha sido modificada por lo que puede ser accesada y leída.
- Compartido (S $shared$): En este estado nos dice que varios procesadores (mas de uno) tiene este bloque es su respectivo cache y no ha sido modificado.
- Invalido (I): Indica un estado no valido de linea de cache.

Lo anterior se puede ver reflejado en su respectivo diagrama de estados.

II-B. MOESI

Para este estado esta claro que se comparten significados, ya que posee 4 estados en comun los cuales son M, E, S, I, pero con diferencia este posee un nuevo estado:

- Propio (O $owned$): Este estado de linea de cache puede verse como un estado el cual es compartido y el mismo esta disponible pero si se desea modificar este solo esta disponible para el cache actual. Por lo que si este es modificado la linea sera tratada como propia, y esta liego sera compartida con las demás lineas de cache.

Podemos ver su diagrama a continuación.

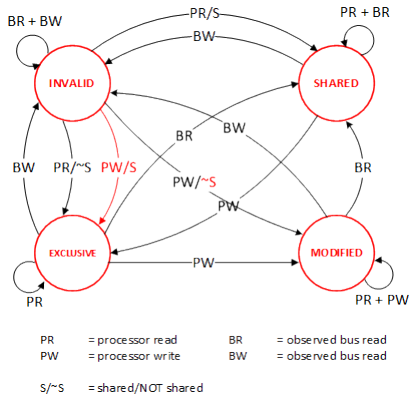

Figure 2. Diagrama de estados MESI

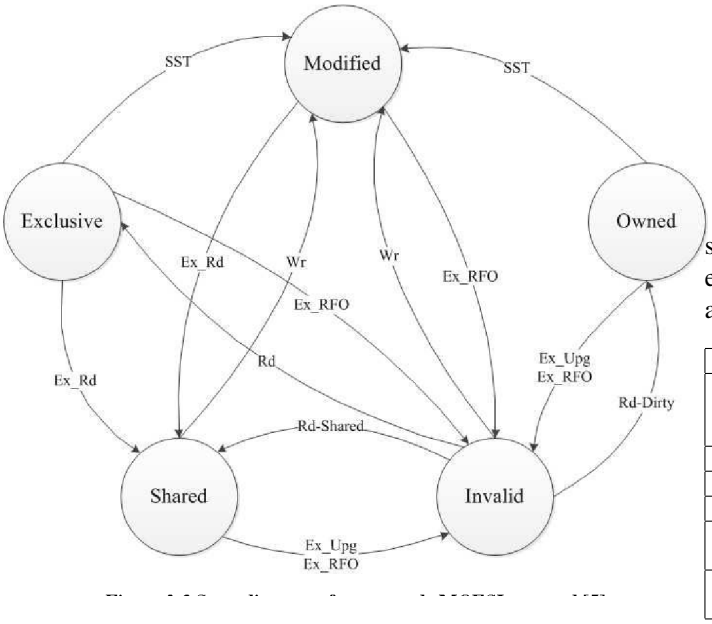

Figure 3. Diagrama de estados MOESI [2]

II-C. Protocolo de directorio

Como fue visto anteriormente es un protocolo de coherencia de cache muy utilizado, este básicamente funciona como los otros ya que permite controlar ese problema solo que lo hace mediante el uso de un bus, también llamado bus compartido, este básicamente pregunta al bus el estado de los directorios para así no ser saturado. [3]

Podemos ver unas ventajas y desventajas a continuación:

- Escalabilidad: Es una parte muy importante por la cual muchos se decantan por este método, por lo que tiene gran flexibilidad al aumento creciente de trabajo, o nodos de trabajo, pero a su vez si este numero crece desmesuradamente podría ser fatal y surgir muchos problemas
- Simplicidad: Es prácticamente el punto mas importante en este protocolo ya que este nos permite organizar todo el trafico que pasa en el sistema, asegura la atomicidad de todas las señales además de que no habrá que esforzarse para garantizar orden del trafico.

Podemos ver el diagrama en la figura 4.

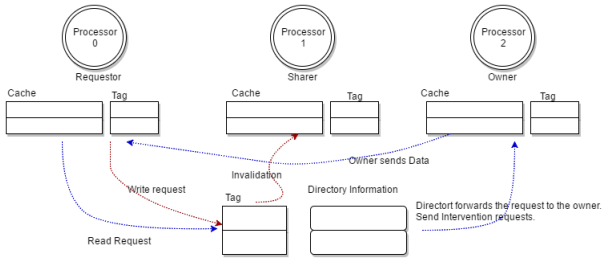

Figure 4. Protocolo directorio

Este tipo de protocolo separa el bus en los siguientes nodos:

- Nodo: El solicitante ya sea de escritura o lectura en bloque de memoria.
- Nodo directorio: Es el que mantiene el estado de cada bloque del cache del sistema, el nodo solicitante se comunica con este.
- Nodo propietario: Posee el estado mas reciente de memoria.
- Nodo compartido: Nodos que comparten copias del bloque de memoria.

Para este protocolo existen una serie de estados los cuales son muy importantes de concer, los cuales nos dicen los estados iniciales y finales, solicitud de petición al bus y la acción a tomar en cuenta [3].

Inicial	Solicitud	Respuesta	Nuevo
U	Rd RdX	Obtener bloque de memoria actualizada Envio de bloque usando mensaje (ReplyD) Si no hay cliente el directorio pasa a EM	EM
EM	Rd	Responde intervención al owner (Int)	S
	RdX	Envié la invalidación al owner (Inv)	
S	Rd	Responde con bloque de memoria (ReplyD)	
	RdX	Responde con bloque de memoria (ReplyD) Invalida a los clientes (Inv)	EM
	Upgr	Invalida a los clientes (Inv) Notifica que puede actualizar	EM

III. Conclusiones

Como se vio anteriormente los protocolos de coherencia de cache son de mucha importancia para la optimización significativa del sistema ya que sin estos la producion de software funcional significativo se veria reducido ya que estos problemas seguirian apareciendo.

El protocolo MOESI tiene una ventaja sobre los demás protocolos ya que con sus estados de Owner provee un estado propio de acceso al cache de un nivel inferior a la memoria principal, ya que los otros protocolos tienen un precio mas caro para acceder a esta.

References

[1] T. Suh, D. M. Blough, and H.-H. Lee, "Supporting cache coherence in heterogeneous multiprocessor systems," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 2. IEEE, 2004, pp. 1150–1155.

[2] H. Altwaijry and D. S. Alzahrani, "Improved-moesi cache coherence protocol," *Arabian Journal for Science and Engineering*, vol. 39, no. 4, pp. 2739–2748, 2014.

[3] D. Chaiken, C. Fields, K. Kurihara, and A. Agarwal, "Directory-based cache coherence in large-scale multiprocessors," *Computer*, vol. 23, no. 6, pp. 49–58, 1990.

[4] X. Zhang, “Verification strategy of cache coherence for opensparc t 2 multi-processor systems under the direction of dr,” 2013.
[5] J. Zebchuk, V. Srinivasan, M. K. Qureshi, and A. Moshovos, “A tagless coherence directory,” in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 423–434.