Proyecto II: Análisis de protocolos de coherencia MESI y MOESI

Yordi Brenes Roda
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
ybrenesr@estudiantec.cr

Luis Andrey Zúñiga Hernández
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
lazh@estudiantec.cr

Fátima Alicia Leiva Chinchilla
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
2019153089@estudiantec.cr

Brian Wagemans Alvarado
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
briwag88@estudiantec.cr

I. Marco teórico

En una arquitecura de computadores, la coherencia de caché hace referencia a la uniformidad de los datos compartidos que son almacenados en múltiples caches. Si varios sistemas mantienen cache de una memoria común es normal que se produzca incoherencia entre los datos. Los protocolos de coherencia de caché son normas que se siguen al acceder a memoria para mantener la coherencia de los datos [1].

Los protocolos pueden tener dos tipos de políticas de escritura.

1) Write-through: La modificación de un elemento es actualizada en memoria principal.
2) Write-back: Se actualiza memoria principal en el próximo remplazo de caché.

Existen diferentes tipos de estados en los que se puede estar un elemento en caché

1) Modified(M): El elemento de caché solo está presente en la caché actual y está modificado. Ese necesario que la caché escriba el dato en memoria principal.
2) Shared(S): Contiene datos no modificados, por lo que puede aparecer en múltiples memorias.
3) Invalid(I): Un dato se hace invalido si otro procesador modificó ese dato. El estado invalido equivale a no tener el dato en memoria.
4) Exclusive(E): El elemento de caché solo está presente en este caché y concuerda con memoria principal.
5) Owned(O): El caché es el único dueño de este contenido. Es el responsable de manejar las solicitudes de lectura.

A. Protocolo de coherencia MESI

Consiste en una máquina de estado finito . Los cambios de estado pueden ser generados por una solicitud de lectura/escritura del procesador dueño de la caché (figura 1) o una solicitud desde el bus de datos (figura 2), lo que significa que otro procesador no tiene el elemento en caché o este no está actualizado.

A continuación se muestran los estados y sus características (tabla I).

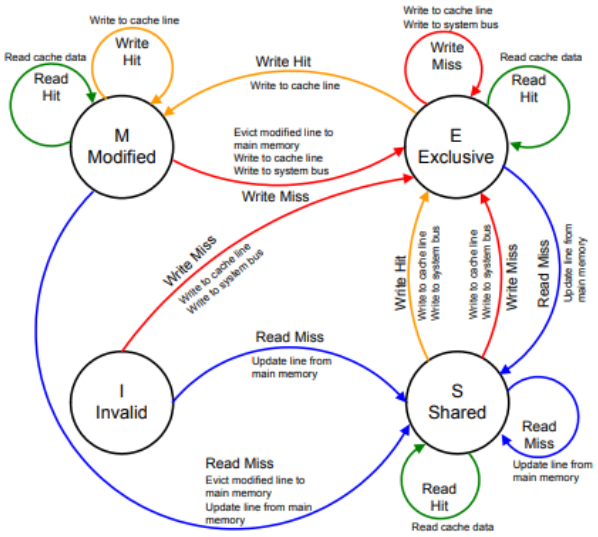

Fig. 1: Maquina de estado finito protocolo MESI, para el PE. Fuente [2]

TABLA I: Estados y caracteristicas del protocolo MESI

Tipo de memoria	**M**	**E**	**S**	**I**
Elemento valido?	Si	Si	Si	No
La copia en memoria	Desact.	Valida	Valida	-
Escritura en este elemento	No va al bus	No va al bus	Va al bus y se actualiza cache	va al bus directamente
Hay una copia en otro caché?	No	No	Es posible	Es posible

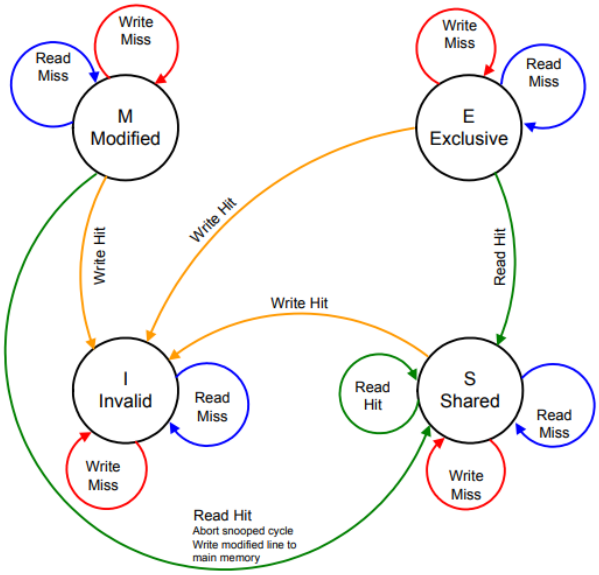

Fig. 2: Maquina de estado finito protocolo MESI, desde el bus. Fuente [1]

B. Protocolo de coherencia MOESI

El protocolo MOESI utiliza los estados que emplea MESI y además utiliza un estado extra O. Este estado extra permite representar elementos que han sido modificados y compartidos. Hacer esto evita la necesidad de acceder a memoria antes de compartir los datos. En MOESI, se debe poder transferir datos de un procesador a otro.

En la figura 3, se muestra la máquina de estado finito.

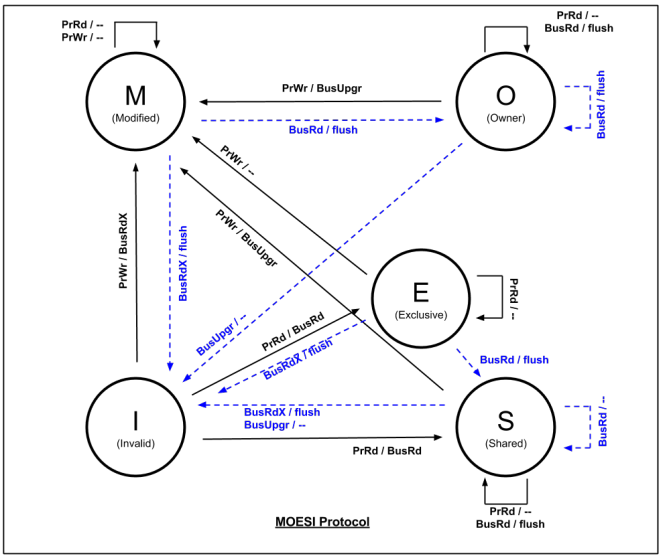

Fig. 3: Maquina de estado finito protocolo MOESI. Fuente [3]

C. MESI vs MOESI

El estado MOESI será más complejo de implementar debido a que al tener un estado extra, existe un mayor número de transiciones posibles. Además, para ambos modelos se debe tener una microarquitectura que permita leer datos de un caché a otro. Como el estado O de MOESI, permite que se hagan menos accesos a memoria, el estado MOESI es más eficiente de forma teórica [4].

II. Introducción

El siguiente texto tiene como objetivo hacer una comparación del protocolo de coherencia MESI y MOESI utilizando un simulador. La microarquitectura modelada consiste de 3 Procesing Elements (PEs), cada uno con caché privada y una única memoria compartida por medio de un bus (figura **??**). Las caché tienen una política de write-back. Las dimensiones de las cachés y memoria compartida se muestran en la tabla **??**. Cada PE se ejecutará de forma paralela, esto será modelado utilizando threads.

TABLA II: Instrucciones de control de flujo de la arquitectura

Tipo de memoria	**Número de entries**	**tamaño de dato**
Memoria compartida	16	32 bits
caché privada	4	32 bits

III. Métricas de desempeño

Para comparar el desempeño de ambos protocolos se utilizaran varias métricas.

La primera es el tiempo de ejecución. El acceso a memoria no es inmediato, acceder a memoria caché y memoria principal conlleva un tiempo. El tiempo de acceso a cada memoria generalmente depende de la jerarquía. En el sistema simulado solamente existe una memoria cache y una memoria compartida (figura 4).

Como la memoria caché es la de primer acceso, esta es más pequeña y rápida que la memoria principal. Para comparar como los protocolos afectan el tiempo de ejecución de los procesadores, se propusieron diferentes tiempos de acceso para cada acceso a memoria (tabla III). Los tiempos se establecieron basándose en [5]

1) Tiempo de ejecución: Usando los tiempos especulados, se puede calcular el tiempo de los accesos. Por ejemplo, el tiempo de acceso total t_T a memoria compartida será

$$t_T = t_\$ + t_{bus} + t_{mem}$$

o en caso de hit

$$t_T = t_\$$$

Donde

1)

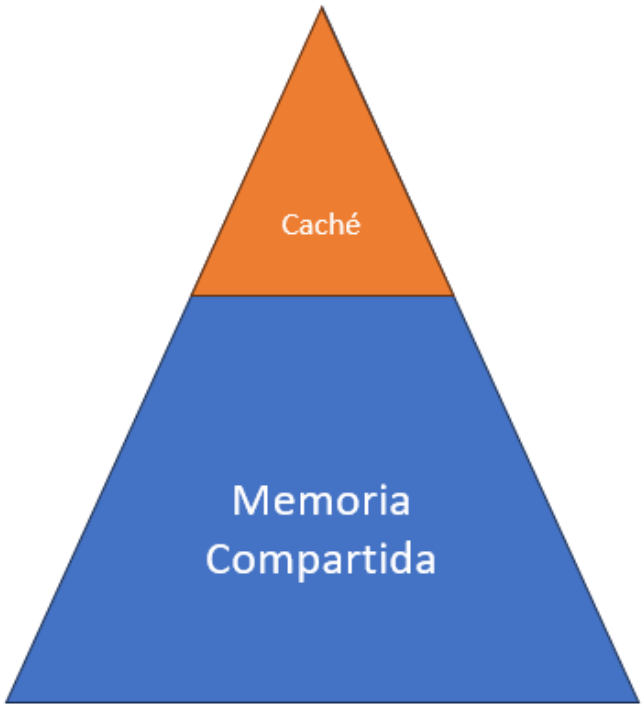

Fig. 4: Jerarquía de memoria

TABLA III: Tiempos de acceso para el sistema simulado

Memoria	**Descripción**	**tiempo** [ns]
caché	Tiempo en acceder a caché	1,1
otro caché	Tiempo que le toma al bus acceder a otro caché	3,3
memoria compar-tida	Tiempo que toma acceder a memoria	62,9

2) t_T: tiempo total
3) $t_\$$: tiempo de acceso al caché
4) t_{bus}: tiempo del bus
5) t_{mem}: tiempo de acceso a memoria

Entonces, a menor tiempo de ejecución, mejor será el rendimiento.

2) Otros parámetros: Otros parámetros que se leyeron fueron la cantidad de transacciones, read requests, write requests e invalidaciones.

1) transacciones: Cualquier acción que tome el bus, por ejemplo, leer, escribir, cambiar el estado de una linea de caché.
2) read request: Cualquier solicitud de lectura al bus.
3) write request: Solicitud para que el bus escriba en memoria.
4) invalidar: Cambiar el estado de una linea de caché a I.

Los read request también representan el número de caché misses, ya que para realizar un read requst necesariamente el dato no debe estar en caché (o estar invalido, que a efectos prácticos, es lo mismo). Puesto que el bus solo escribirá sobre una linea de memoria, el write request también representa el número de accesos a memoria. Además, un dato invalido no puede ser leído ni alterado. Cuanto menor sea el número de lineas en estado I, mejor será el desempeño de un sistema. Esto se cumple para ambos protocolos.

IV. Resultados

Para cada protocolo se corrieron 30 instrucciones aleatorias por hilo. Se utilizó el mismo código para ambas corridas. Esto se hizo 5 veces para cada protocolos, a continuación se presentan los resultados finales (tabla IV).

TABLA IV: Datos promedio de los reportes de los protocolos MESI y MOESI

Medición	**MESI**	**MOESI**
Transacciones totales	348	328
Read Requests	48,0	42,8
Write Requests	24,8	15,2
Invalidaciones	13,8	15,8
Coste	4886,22	3954,38

En la siguiente tabla se muestra el porcentaje de diferencia entre MESI y MOESI

TABLA V: Porcentaje de cambio entre ambos protocolos respecto a MESI

Medición	**Porcentaje de cambio respecto a MESI**
Transacciones totales	5,75
Read Requests	10,83
Write Requests	38,71
Invalidaciones	-14,49
Tiempo de ejecución[ns]	19,07

V. Análisis de resultados

Los resultados simulados concuerdan con los esperados, El write request es mucho mayor en el protocolo MESI que en el MOESI, lo cual tiene sentido, ya que el estado O, reduce la cantidad de accesos a memoria compartida. Otra gran diferencia se da en el tiempo de ejecución, que depende de la cantidad de accesos a memoria caché, otras cachés y memoria compartida. Sin embargo, como el coste de acceso a memoria compartida es tan grande en comparación a los otros dos, este será el parametro que más afecte el tiempo de ejecución. Que el tiempo de ejecución de MESI sea más alto concuerda con que el número de acceso a memoria sea mayor.

El tiempo de ejecución es una forma de comparar el efecto que tiene el protocolo en el desempeño de el protocolo, sin embargo, este tiempo no toma en cuenta la proximidad física de otros cachés, el ancho de banda de la caché, el tiempo que tardan las señales en ir de un caché a otro, entre otros.

Aunque existe una diferencia en los parámetros de read request, transacciones totales e invalidaciones. La diferencia es muy pequeña como para concluir algo con seguridad. Un mayor número de datos podría reducir la varianza de los datos medidos y dar una predicción más precisa de las diferencias entre MESI y MOESI, principalmente en parámetros como invalidaciones, read requests y write requests.

Finalmente, tomar en cuenta otros parámetros para calcular el tiempo de ejecución puede dar un tiempo más aproximado a la realidad.

A. Conclusiones

El tiempo de ejecuión de MOESI resultó ser más corto que el tiempo del protocolo MESI. Además, el protocolo MOESI realiza menos accesos a memoria que el MESI. El protocolo de coherencia de caché MOESI resultó ser más eficiente que el protocolo MESI. Se recomienda tomar un mayor número de datos para esclarecer mejor las diferencias entre protocolos para parámetros como transacciones totales, read requests e invalidaciones.

REFERENCIAS

[1] Redis, "Cache coherence." [Online]. Available: https://redis.com/glossary/cache-coherence/

[2] F. J. Jimenez, "TEACHING THE CACHE MEMORY COHERENCE WITH THE MESI PROTOCOL SIMULATOR." [Online]. Available: http://e-spacio.uned.es/fez/eserv/taee:congreso-2006-1112/S3E04.pdf

[3] R. K. Ibrahim, "Design of MOESI protocol for multicore processors based on FPGA," 2021. [Online]. Available: http://e-spacio.uned.es/fez/eserv/taee:congreso-2006-1112/S3E04.pdf

[4] "Slide view: Parallel Computer Architecture and Programming," 2016. [Online]. Available: http://15418.courses.cs.cmu.edu/spring2016/lecture/snoopcoherence/slide

[5] "Lab 4: Caching, Lab 4: Caching - HackMD." [Online]. Available: https://cs.brown.edu/courses/csci1310/2020/assign/labs/lab4.html