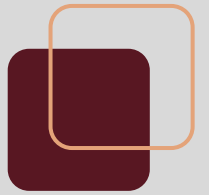
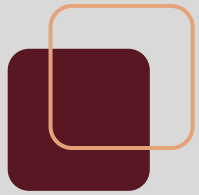


Aspectos avanzados de arquitectura de computadoras Multiprocesadores (I)

Facultad de Ingeniería - Universidad de la República

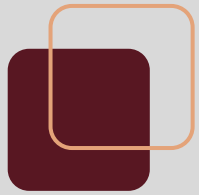
Curso 2017

Motivación



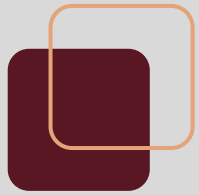
- Explotación de ILP estancada desde 2005 (aproximadamente)
- Consumo eléctrico crece en mayor proporción que la performance si se superan los estándares actuales
- Paralelismo a nivel de thread *más común* en entornos de servidores
- Menor interés en computadoras de escritorio

Taxonomía de Flynn



- SISD: Single Instruction, Single Data
  - Procesos tradicionales ejecutando en CPU *unicore*.
- SIMD: Single Instruction, Multiple Data
  - Vector processor, GPU
- MISD: Multiple Instruction, Single Data
- MIMD: Multiple Instruction, Multiple Data
  - Multicore, Server

Symmetric Multiprocessors (1/6)



- Se denomina multiprocesadores simétricos a aquellos multiprocesadores:
  - En que toda la memoria dista lo mismo de todos los procesadores.
  - Todos los procesadores tienen las mismas características:
    - Funciones similares
    - Pueden realizar accesos de E/S

Symmetric Multiprocessors (2/6)

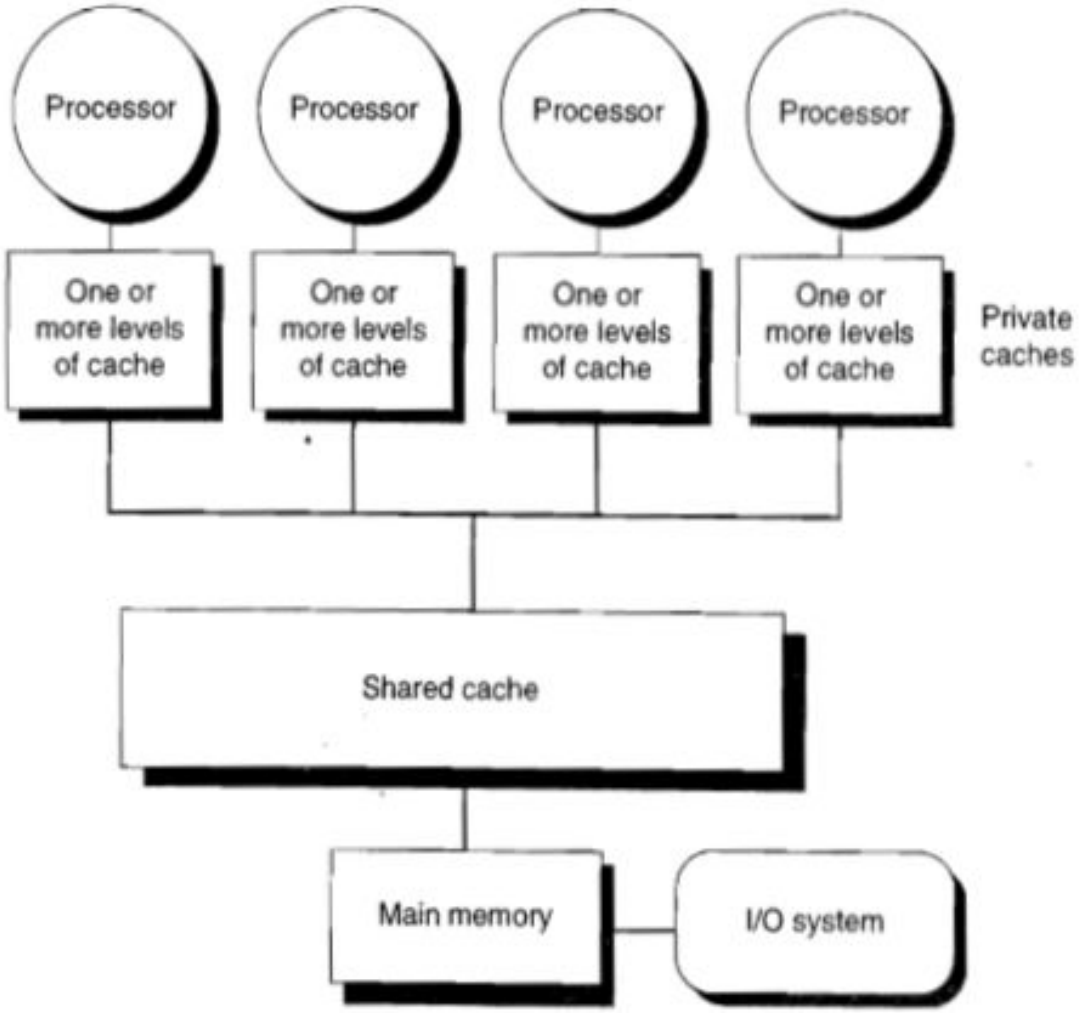
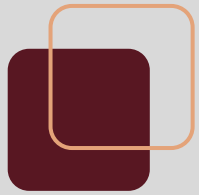


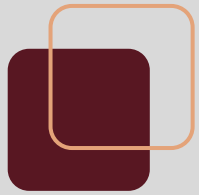
Imagen tomada del libro Computer Architecture, a Quantitative Approach. John L. Hennessy y David Patterson (2011)

Symmetric Multiprocessors (3/6)



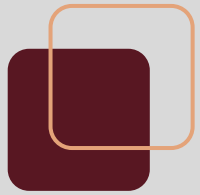
- Dado que toda la memoria dista igual de todos los CPUs, los procesadores simétricos también son conocidos como procesadores UMA (Uniform Memory Access)
- En general, este tipo de procesadores tienen la memoria concentrada en un único módulo, por lo cual también son conocidos como *centralized shared memory processors*

Symmetric Multiprocessors (4/6)



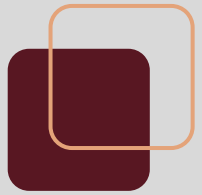
- La gran mayoría de los multiprocesadores comerciales (multicores) son de tipo UMA.
- En estos procesadores, un chip de memoria es capaz de proveer el ancho de banda necesario para el sistema
- A medida que el número de procesadores aumenta, también lo hace el ancho de banda requerido sobre la memoria, siendo un modelo *NUMA* más apropiado (*non-uniform memory access*).

Symmetric Multiprocessors (5/6) Interconexión

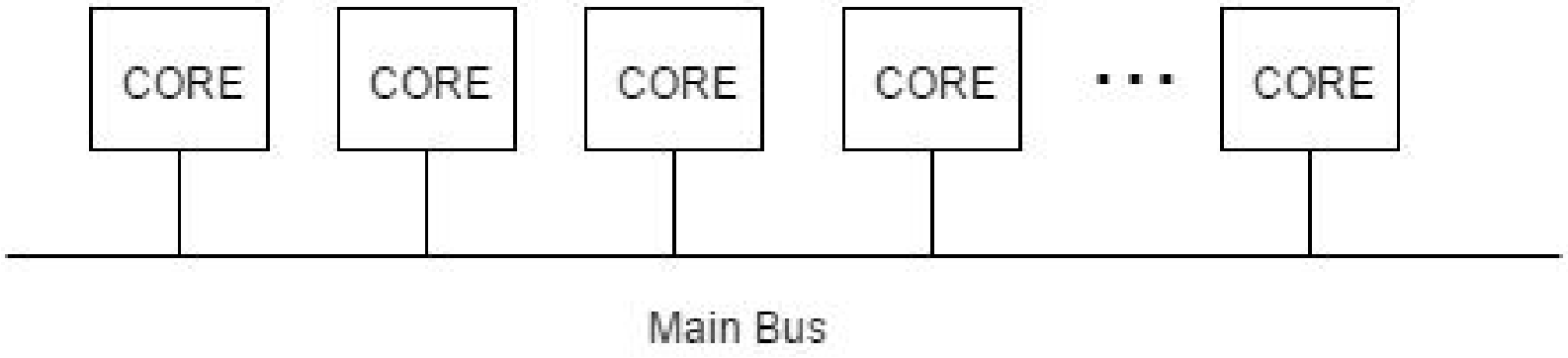


- Otra característica de los procesadores UMA es que típicamente hay un bus central que conecta la memoria principal con el último nivel de caché privada (típicamente L2) o unificada (típicamente L3)
- Por esta característica, típicamente estos procesadores son llamados *bus-based microprocessors.*

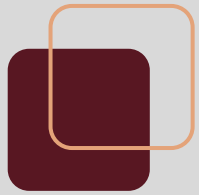
Symmetric Multiprocessors (6/6) Interconexión



- ¿Cuántos CPUs pueden ser conectados?
  - A medida que se agregan CPUs a una conexión de bus, el ancho de banda del mismo comienza a ser una limitante!



Distributed Shared Memory (1/4)



- El diseño alternativo al CSM es aquel donde la memoria está físicamente distribuida: *Distributed Shared Memory*
- Típicamente el acceso a cada chip de memoria se realiza con un tiempo diferente, por lo cual estos procesadores son llamados *NUMA* (Non-Uniform Memory Access)
- Obviamente el acceso es más largo según qué tan lejos se encuentre el CPU del chip accedido

Distributed Shared Memory (2/4)

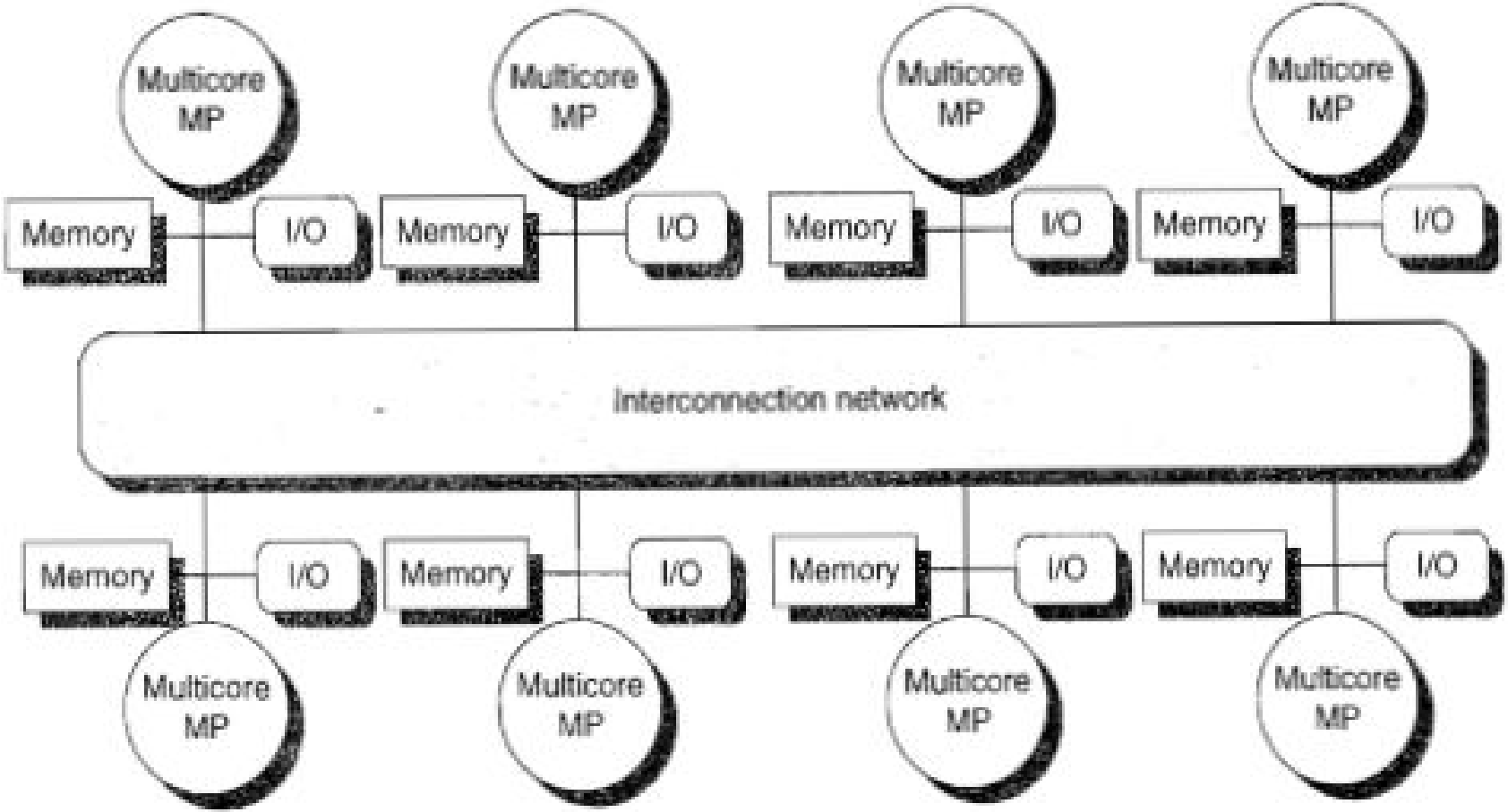
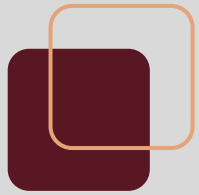


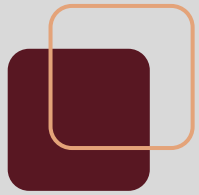
Imagen tomada del libro Computer Architecture, a Quantitative Approach. John L. Hennessy y David Patterson (2011)

Distributed Shared Memory (3/4)



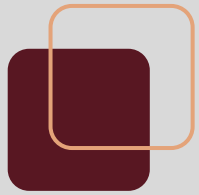
- El esquema general consiste en un chip de memoria DRAM cercano a cada procesador y una *red de interconexión* para manejar el acceso a un chip de memoria diferente al local.
- Esta interconexión típicamente NO es manejada con un bus sino con una *red basada en switches*.

Distributed Shared Memory (4/4)



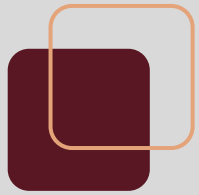
- Características:
  - Acceso a memoria local más rápido que a memoria principal en UMA!
  - Mayor ancho de banda a nivel *global*
  - Compleja comunicación entre procesos!

Memoria Compartida (1/2)



- Procesadores simétricos actuales generalmente disponen de niveles de memoria privados y compartidos.
- Ejemplo:
  - L1 de instrucciones, L1 de datos, privada
  - L2 unificada (instrucciones Y datos), privada
  - L3 compartida entre cores

Memoria Compartida (2/2)



- Cuando un dato privado pasa a una caché privada, el comportamiento es idéntico al de un monoprocesador
- Cuando un dato *compartido* pasa a una caché privada, se deben tomar precauciones pues para una misma posición de memoria dos procesadores podrían ver valores diferentes.
- Esto es conocido como el problema de *coherencia de caché*.

Coherencia de Caché (1/6)

- Write Back:

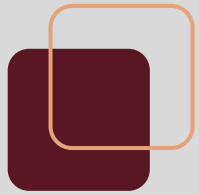
Tiempo	Procesador A	Procesador B	Resultado en Cache
1	Lectura dirección X	-	Read Miss: Se copia el bloque que contiene la dirección X al cache del procesador A.
2	Escritura dirección X, valor Y	-	Write Hit: Se escribe el valor Y en el cache privado de A.
3	-	Lectura dirección X	Read Miss: Se lee el valor desactualizado de X desde la memoria principal..

Coherencia de Caché (2/6)

- Write Through:

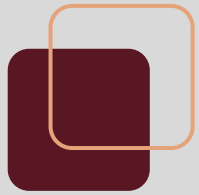
Tiempo	Procesador A	Procesador B	Resultado en Cache
1	Lectura dirección X	-	Read Miss: Se copia el bloque que contiene la dirección X al cache del procesador A.
2	-	Lectura dirección X	Read Miss: Se copia el bloque que contiene la direccioón X al cache del procesador B
3	Escritura dirección X, valor Y	-	Write Hit: Se escribe el valor Y en el cache privado de A y en memoria principal.
4	-	Lectura dirección X	Read Hit: Se lee el valor desactualizado de X desde el cache privado de B.

Coherencia de Caché (3/6)
Definición



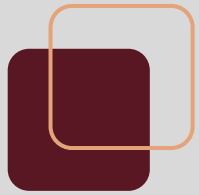
1. Una lectura a una dirección X por un procesador P, que se realiza a continuación de una escritura en esa dirección, devuelve el valor escrito.
2. Una lectura a una dirección X por un procesador P, que se realiza a continuación de una escritura en misma dirección por otro procesador P', devuelve el valor escrito por P' si los accesos están suficientemente separados en el tiempo.
3. Las escrituras a una misma dirección están serializadas (dos escrituras a una misma dirección son vistas por todos los procesadores en el mismo orden).

Coherencia de Caché (4/6)



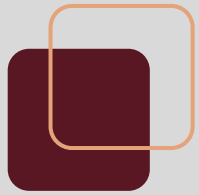
- Protocolos de coherencia de caché: Protocolos que aseguran la coherencia en sistemas multiprocesador.
- Diferentes enfoques según la arquitectura de memoria que se esté usando.

Coherencia de Caché (5/6)
Snooping Protocols



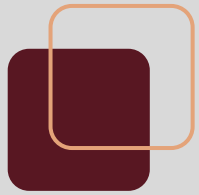
- La idea consiste en que los caches monitoreen o 'husmeen' (snoop) el bus para detectar pedidos a direcciones que ellas contengan. Manteniendo la coherencia en caso de realizarse operaciones sobre sus bloques.
- Requiere un medio común de acceso a la memoria, de modo que todos los cachés puedan detectar todos los accesos realizados

Coherencia de Caché (6/6)
Directory Based Protocols



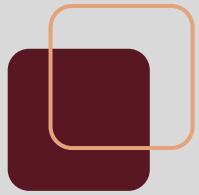
- El estado de cada bloque de memoria se mantiene en un solo lugar, llamado directorio. Cada vez que se desea hacer un acceso a memoria, se avisa al directorio, quien se encarga de mantener la coherencia (por ejemplo actualizando copias del bloque en casos de escritura).
- Son usados fundamentalmente cuando NO existe un medio común de acceso a la memoria.

Snooping Protocols (1/6)



- Se basan en asegurar que sólo un caché puede realizar escrituras sobre un cierto bloque en un cierto momento. Para esto, mantienen bits de estado asociados a cada bloque, indicando si el mismo es un copia de sólo lectura o si es posible escribirlo.
- Una idea posible para asegurar la coherencia es comunicar por el medio común todas las escrituras y que los caches actualicen sus bloques con los valores correctos.

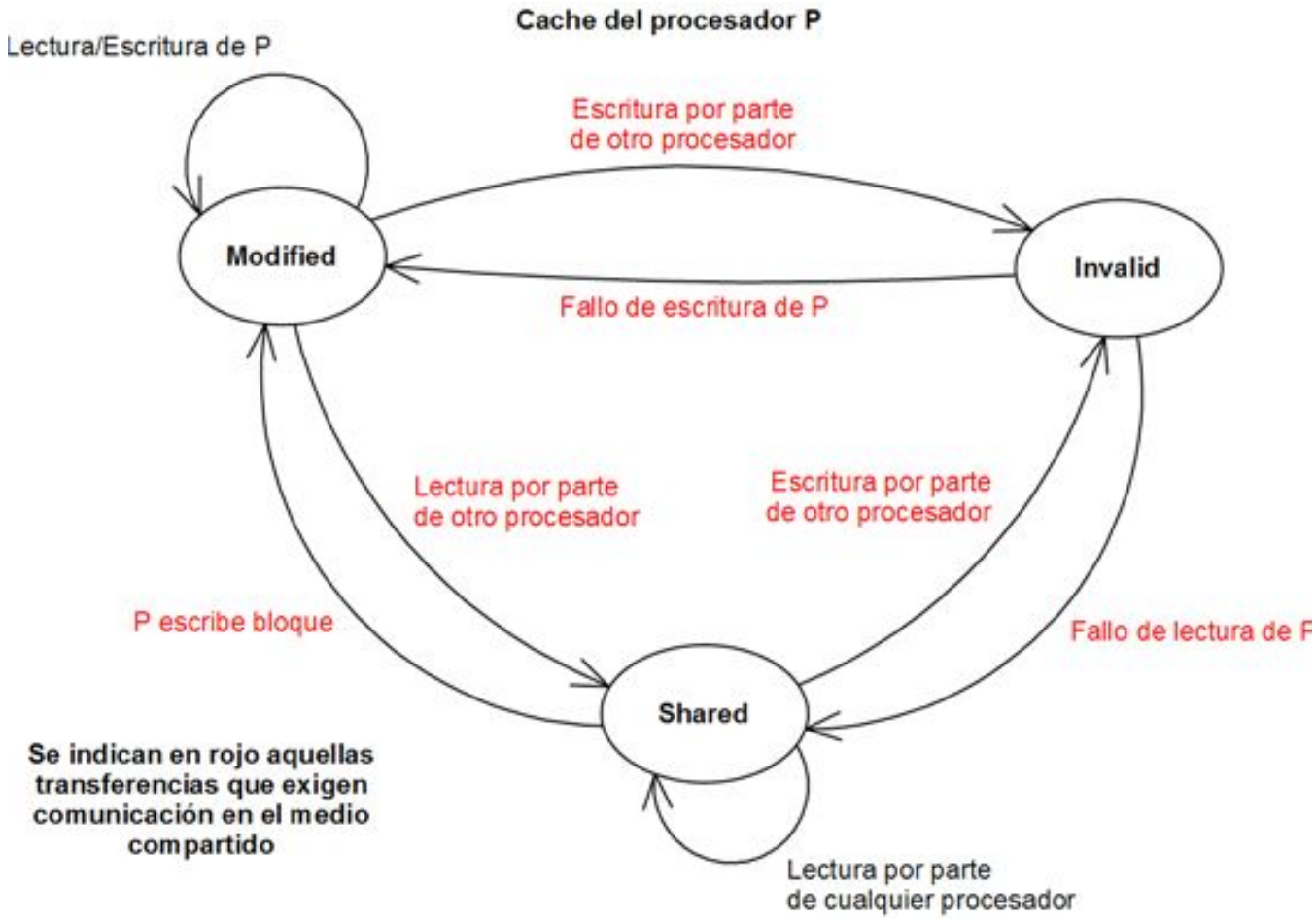
Snooping Protocols (2/6)



- Protocolo MSI (Modified – Shared – Invalid).
- Estados:
  - Invalid (Inválido): El bloque es inválido. Por más que se tiene un valor cacheado del mismo, éste se encuentra desactualizado.
  - Shared (Compartido): El bloque se encuentra en estado compartido, quiere decir que existen otras copias de este bloque y que por tanto es una copia de sólo lectura.
  - Modified (Modificado): La copia del bloque de este cache fue modificada por el CPU adyacente y por tanto es la única copia no inválida del mismo.

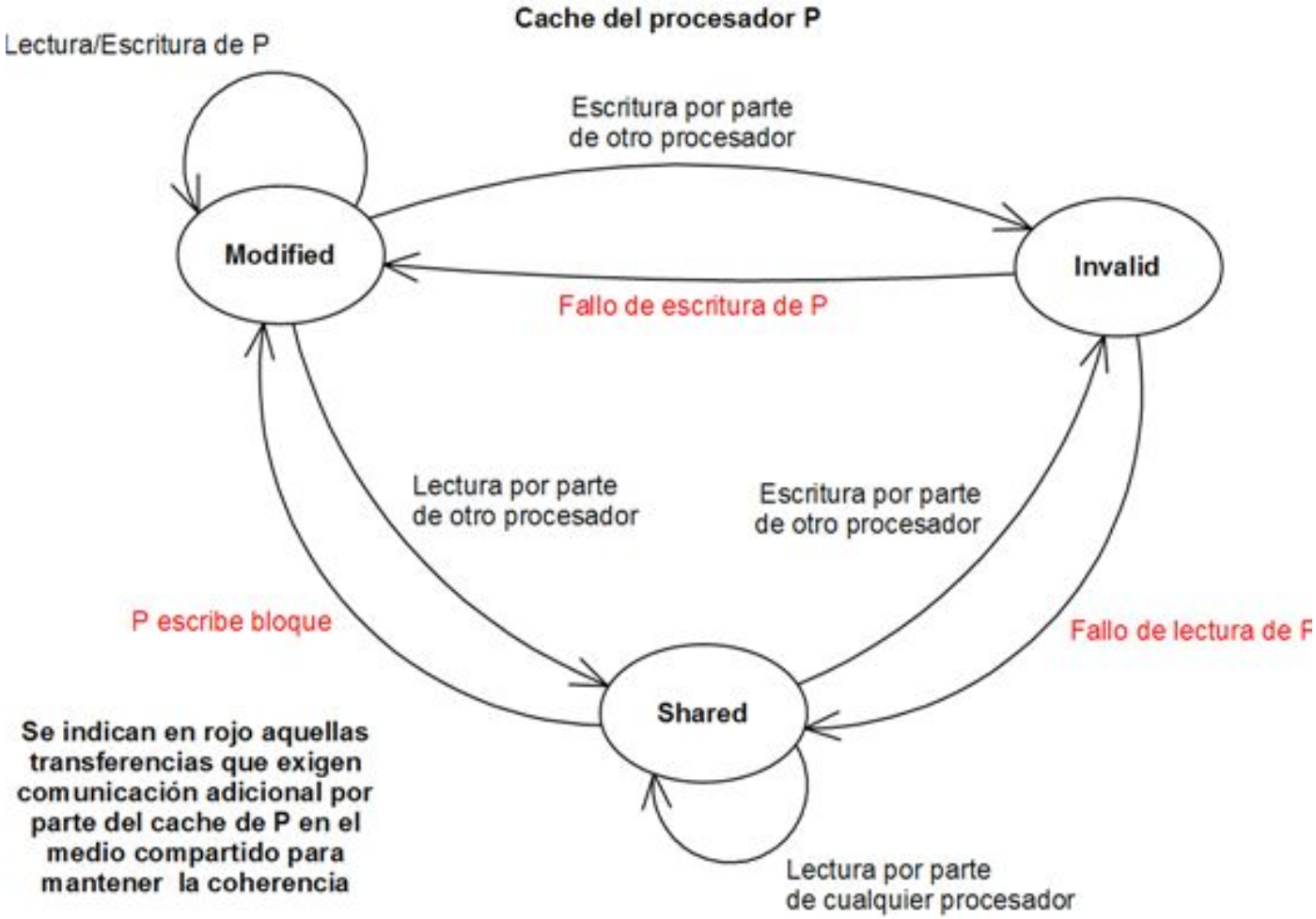
Snooping Protocols (3/6)

- Protocolo MSI

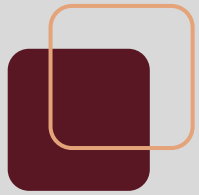


Snooping Protocols (4/6)

- Costo de la coherencia:



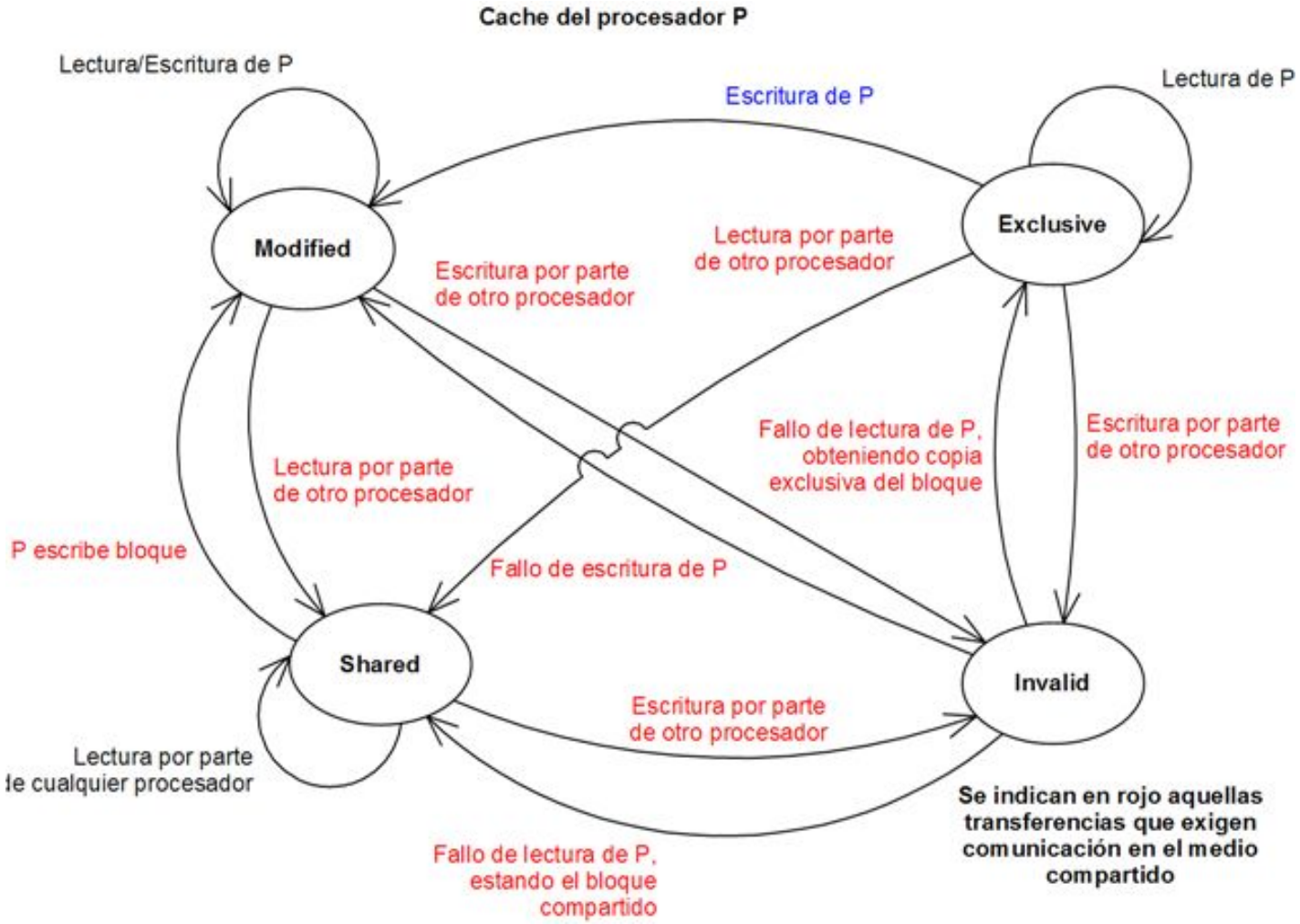
Snooping Protocols (5/6)



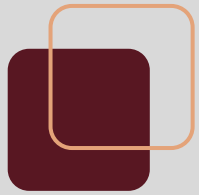
- Protocolo MESI (Enhaced MSI)
- Estados:
  - M-S-I: Mismos que en protocolo MSI
  - Exclusive (Exclusivo pero no modificado): Este caché contiene la única copia del bloque y el mismo no fue modificada.

Snooping Protocols (6/6)

- Protocolo MESI



False Sharing (1/2)

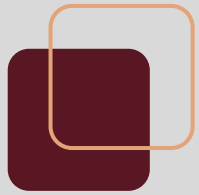


- Como la invalidación se realiza a nivel de bloque, accesos a diferentes direcciones por parte de diferentes procesadores puede causar bloqueos.
  - True Sharing Miss: Se define con este nombre a un cache miss provocado por la invalidación de la misma dirección por otro procesador.
  - False Sharing Miss: Se define con este nombre a un cache miss provocado por la invalidación de otra dirección en el mismo bloque al que se quiere acceder.

False Sharing (2/2)

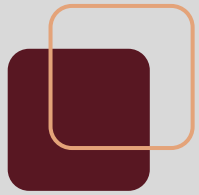
Tiempo	Procesador A	Procesador B	Resultado en Cache
1	Lectura dirección X_1	-	Read Miss: Se obtiene el bloque de memoria y queda en estado Exclusivo
2	-	Lectura dirección X_1	Read Miss: Se obtiene el bloque de memoria y queda en estado compartido.
3	Escritura dirección X_2	-	True share Write Miss. Se escribe la dirección X_2, se pasa a estado modificado y se invalidan las demás copias.
4	-	Lectura dirección X_1	False share miss: Se obtiene la copia modificada del cache de A, aunque X_1 no había sido modificada.

Límites de protocolos de husmeo



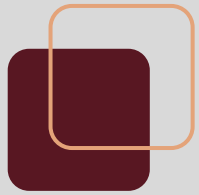
- A medida que aumenta el número de procesadores se presentan los siguientes cuellos de botella:
  - Bus de interconexión (frecuencia)
  - Lógica de control del protocolo (consumo eléctrico)

Límites de protocolos de husmeo



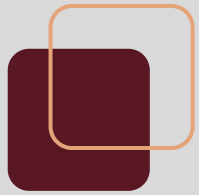
- Es posible implementar snooping en sistemas sin buses sino con enlaces punto a punto, siendo imprescindible la implementación de mensajes broadcast para los *snoop requests* de los cachés.
- Actualmente, la implementación usual para los multiprocesadores grandes (32 y más cores) es el de protocolos de directorio.

Sincronización (1/3)



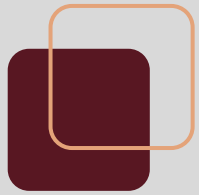
- Algoritmo en un cache write miss:
  a. Obtener la versión más reciente del bloque
  b. Invalidar el bloque
  c. Realizar la escritura
- En la práctica, no es factible realizarlo en un único ciclo, por lo cual se deben tomar precauciones para asegurar la coherencia y consistencia de memoria.

Sincronización (2/3)



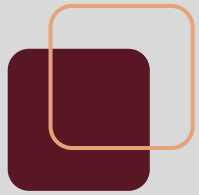
- Antes de comenzar el proceso, el caché compite hasta tener control del bus, y no lo libera hasta que se hayan completado todas las invalidaciones.
- La arbitración asegura que las escrituras se realicen de forma *secuencial*
- Se utiliza una señal especial del bus de control para asegurar que todas las demás cachés completaron el pedido de invalidación.

Sincronización (3/3)



- ¿Y qué sucede en sistemas sin bus?
  - Se deben tomar otras precauciones que aseguren la secuencialidad de las escrituras.
- La situación en que dos o más procesadores quieren escribir un mismo bloque al mismo tiempo (durante el mismo ciclo), se denomina *carrera (race)*

Fin



¿Preguntas?