Memoria compartida simétrica

Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid

Creciente importancia de multiprocesadores

- Caída de la eficiencia en uso de silicio y energía al explotar mayor nivel de ILP.
  - El coste de silicio y energía crece más rápidamente que el rendimiento.
- Interés creciente en servidores de alto rendimiento.
  - Cloud computing, *software as a service*, . . .
- Crecimiento de aplicaciones intensivas en datos.
  - Enormes cantidades de datos en Internet.
  - *Big data analytics*.

TLP: Paralelismo a nivel de hilo

- TLP implica la existencia de múltiples contadores de programa.
  - Asume MIMD.
  - Uso generalizado de TLP fuera de computación científica relativamente reciente.
  - Nuevas aplicaciones:
    - Aplicaciones empotradas.
    - Desktop.
    - Servidores de alta gama.

Multiprocesadores

- Un **multiprocesador** es un computador formado por procesadores altamente acoplados con:
  - **Coordinación y uso** típicamente controlados por un **sistema operativo único**.
  - **Compartición de memoria** mediante un **único espacio de direcciones compartido**.

- **Modelos de software**:
  - **Procesamiento paralelo**: Conjunto de hilos acoplados que cooperan.
  - **Procesamiento de peticiones**: Ejecución de procesos independientes originados por usuarios.
  - **Multiprogramación**: Ejecución independiente de múltiples aplicaciones.

- La aproximación más común:
  - De 2 a decenas de procesadores.
  - Memoria compartida.
    - Implica memoria compartida.
    - No necesariamente implica una única memoria física.

- **Alternativas**:
  - **CMP** (*Chip MultiProcessors*) o *multi-core*.
  - Múltiples chips.
    - Cada uno puede ser (o no) *multi-core*.
  - **Multicomputador**: Procesadores débilmente acoplados que no comparten memoria.
    - Usados en computación científica de gran escala.

- **Aprovechamiento** de un multiprocesador:
  - Con **n** procesadores se necesitan **n** hilos o procesos.

- **Identificación** de hilos:
  - Identificados explícitamente por programador.
  - Creados por el sistema operativo a partir de peticiones.
  - Iteraciones de un bucle generadas por compilador paralelo (p. ej. OpenMP).

Identificación realizada a alto nivel por el programador o software de sistema con hilos con un número **suficiente** de instrucciones a ejecutar.

Multiprocesadores y memoria compartida

SMP: Symmetric Multi-Processor

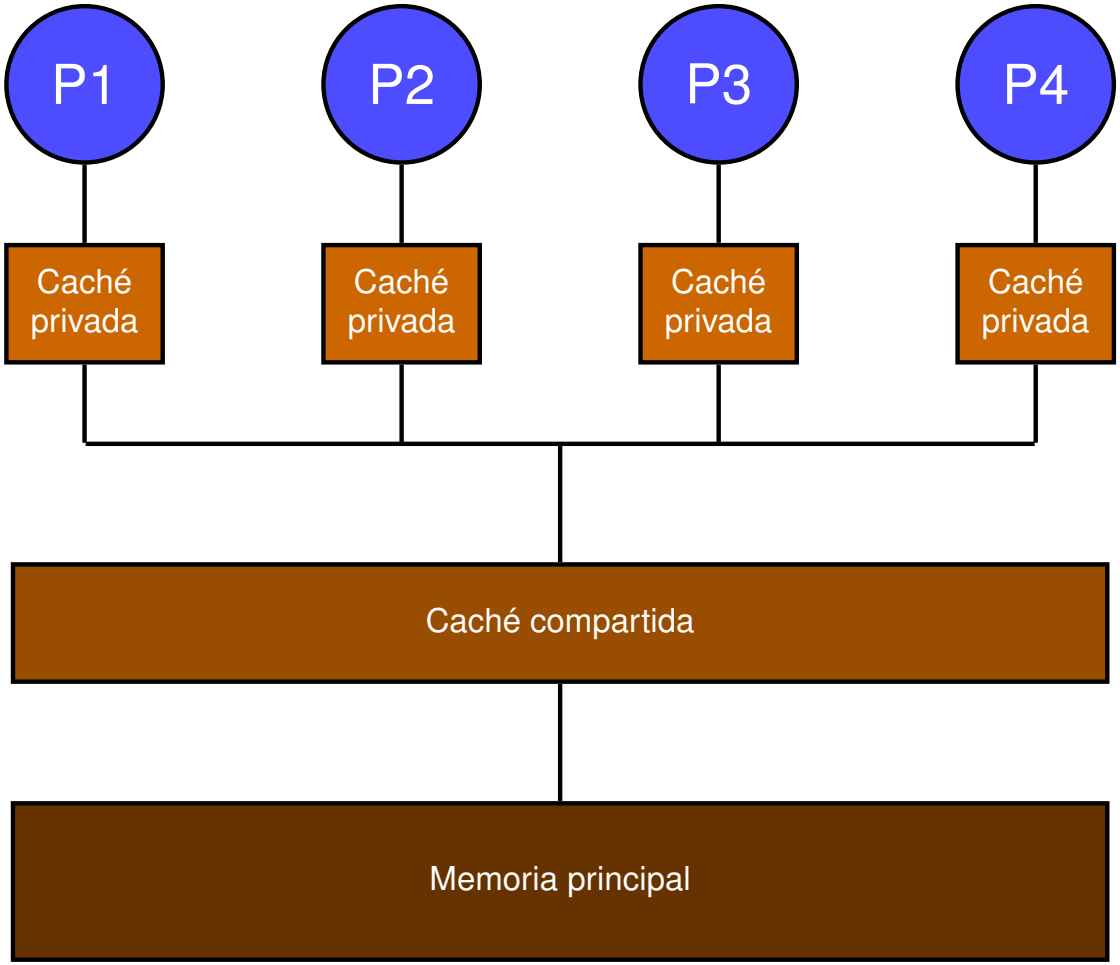
- Memoria compartida centralizada.
- Comparten una memoria centralizada única a la que todos acceder por igual.
- Todos los multi-core son SMP.
- **UMA**: Uniform Memory Access
  - La latencia de memoria es uniforme.

DSM: Distributed Shared Memory

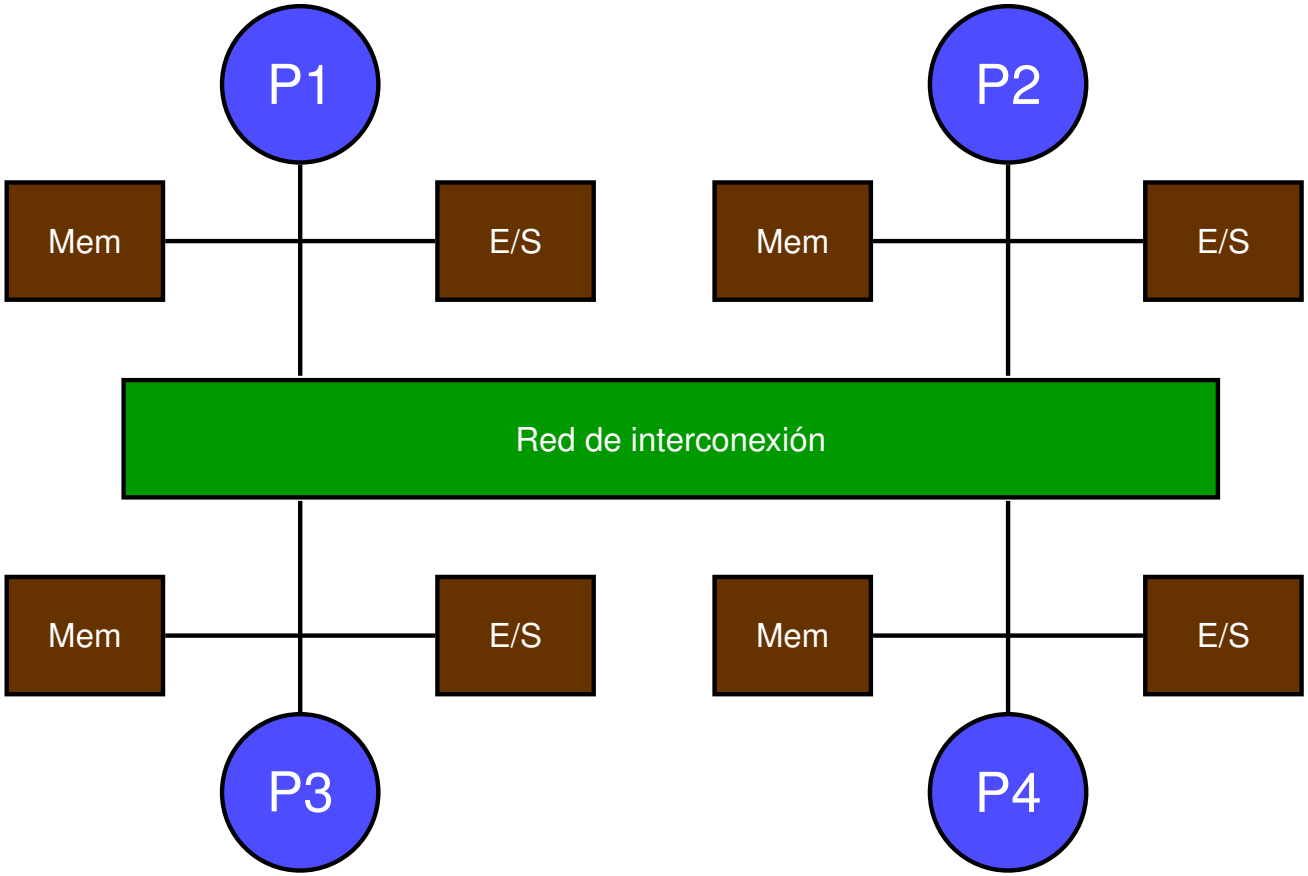
- Memoria compartida distribuida
- La memoria se distribuye entre los procesadores.
- Necesaria cuando hay muchos procesadores.
- **NUMA**: Non Uniform Memory Access.
  - La latencia depende de la ubicación del dato accedido.

- Comunicación mediante acceso a memoria compartidas globales.

SMP: Symmetric MultiProcessor



DSM: Distributed Shared Memory



SMP y jerarquía de memoria

- ¿Por qué usar memoria centralizada?
  - Cachés grandes multi-nivel reducen la demanda de ancho de banda sobre memoria principal.

- **Evolución**:
  1. Mono-núcleo con memoria en **bus compartido**.
  2. Conexión de memoria a **bus separado** solamente para memoria.

Memoria caché

- **Tipos de datos** en memoria caché:
  - **Datos privados**: Datos usados por un único procesador.
  - **Datos compartidos**: Datos usados por varios procesadores.

- **Problema** con datos compartidos:
  - El dato puede replicarse en múltiples caché.
  - Reduce la contención.
    - Cada procesador accede a su copia local.
  - Si dos procesadores modifican sus copias . . .
    - ¿Coherencia de caché?

Coherencia de caché

Thread 1

```
lw $t0, dirx
addi $t0, $t0, 1
sw $t0, dirx
```

Thread 2

```
lw $t0, dirx
```

- **$t0** inicialmente 1.
- Asumiendo escritura inmediata.

Proceso	Instrucción	Caché P1	Caché P2	Memoria principal
T1	Inicialmente	No presente	No presente	1
T1	**lw $t0, dirx**	1	No presente	1
T1	**addi $t0, $t0, 1**	1	No presente	1
T2	**lw $t0, dirx**	1	1	1
T1	**sw $t0, dirx**	0	1	1

Incoherencia de caché

- ¿Por qué se da la incoherencia?
  - **Dualidad de estado**:
    - **Estado global** → **Memoria principal**.
    - **Estado local** → **Caché privada**.

- Un sistema de memoria es **coherente** si cualquier lectura de una posición devuelve el valor más reciente que se haya escrito para esa posición.

- **Dos aspectos**:
  - **Coherencia**: ¿Qué valor devuelve una lectura?
  - **Consistencia**: ¿Cuándo obtiene una lectura un valor escrito?

Condiciones para la coherencia

- **Preservación de orden de programa**:
  - Una lectura del procesador P sobre la posición X posterior a una escritura del procesador P sobre la posición X, sin escrituras intermedias de X por otro procesador, siempre devuelve el valor escrito por P.
- **Vista coherente de la memoria**:
  - Una lectura de un procesador sobre la posición X posterior a una escritura por otro procesador sobre la posición X, devuelve el valor escrito si las dos operaciones están suficientemente separadas en el tiempo y no hay escrituras intermedias sobre X.
- **Serialización de escrituras**:
  - Dos escrituras sobre la misma posición por dos procesadores son vistas en el mismo orden por todos los procesadores.

Consistencia de memoria

- Define en qué momento un proceso que lee verá una escritura.

- **Coherencia** y **consistencia** son complementarias:
  - **Coherencia**: Comportamiento de lecturas y escrituras a una única posición de memoria.
  - **Consistencia**: Comportamiento de lecturas y escrituras con respecto a accesos a otras posiciones de memoria.

- Existen distintos modelos de consistencia de memoria.
  - **Dedicaremos una sesión específica a este problema**.

Multiprocesadores coherentes

- Un **multiprocesador coherente** ofrece:
  - **Migración** de datos compartidos.
    - Un dato puede moverse a una caché local y usarse de forma transparente.
    - Reduce latencia de acceso a dato remoto y demanda de ancho de banda a la memoria compartida.
  - **Replicación** de datos compartidos leídos simultáneamente.
    - Se realiza copia del dato en caché local.
    - Reduce latencia de acceso y contención de las lecturas.
- **Propiedades críticas para el rendimiento**:
  - **Solución**: Protocolo hardware de mantenimiento de coherencia de caché.

Clases de protocolos de coherencia de caché

- Basados en **directorio**:
  - El estado de compartición se mantiene en un directorio.
  - **SMP**: Directorio centralizado en memoria o en caché de más alto nivel.
  - **DSM**: Para evitar cuello de botella se usa un directorio distribuido (más complejo).
- **Snooping** (espionaje):
  - Cada caché mantiene el estado de compartición de cada bloque que tiene.
  - Las cachés accesibles mediante medio de multidifusión (bus).
  - Todas las cachés monitorizan el medio de multidifusión para determinar si tienen una copia del bloque.

Mantenimiento de la coherencia

- **Invalidación de escrituras**:
  - Garantiza que un procesador tiene **acceso exclusivo** a un bloque antes de realizar una escritura.
  - **Invalida** el resto de copias que puedan tener otros procesadores.

- **Actualización de escrituras** (difusión de escrituras):
  - Difunde todas las escrituras a **todas las cachés** para modificar el bloque.
  - Consume **más ancho de banda**.

- Estrategia más común $\Rightarrow$ **Invalidación**.

Uso del bus de memoria

- **Invalidación**.
  - El procesador adquiere el bus y difunde la dirección a invalidar.
  - Todos los procesadores espían el bus.
  - Cada procesador comprueba si tienen en caché la dirección difundida y la invalidan.
- No puede haber dos escrituras simultáneas:
  - El uso exclusivo del bus serializa las escrituras.
- **Fallos de caché**:
  - **Escritura inmediata** (write through):
    - La memoria tiene la última escritura realizada.
  - **Post-escritura** (write back):
    - Si un procesador tiene una copia modificada, contesta al falló de caché del otro procesador.

Implementación

- **Invalidación**:
  - Se aprovecha el **bit de validez** (V) asociado a cada bloque.
- **Escrituras**:
  - Necesidad de saber si hay **otras copias** en caché.
    - Si no hay otras copias no hay que **difundir escritura**.
  - Se añade **bit de compartición** (S) asociado a cada bloque.
  - Cuando hay escritura:
    - Se genera **invalidación** en bus.
    - Se pasa de **estado compartido** a **estado exclusivo**.
    - No hace falta enviar nuevas invalidaciones.
  - Cuando hay **fallo de caché** en otro procesador:
    - Se pasa de **estado exclusivo** a **estado compartido**.

Protocolo básico

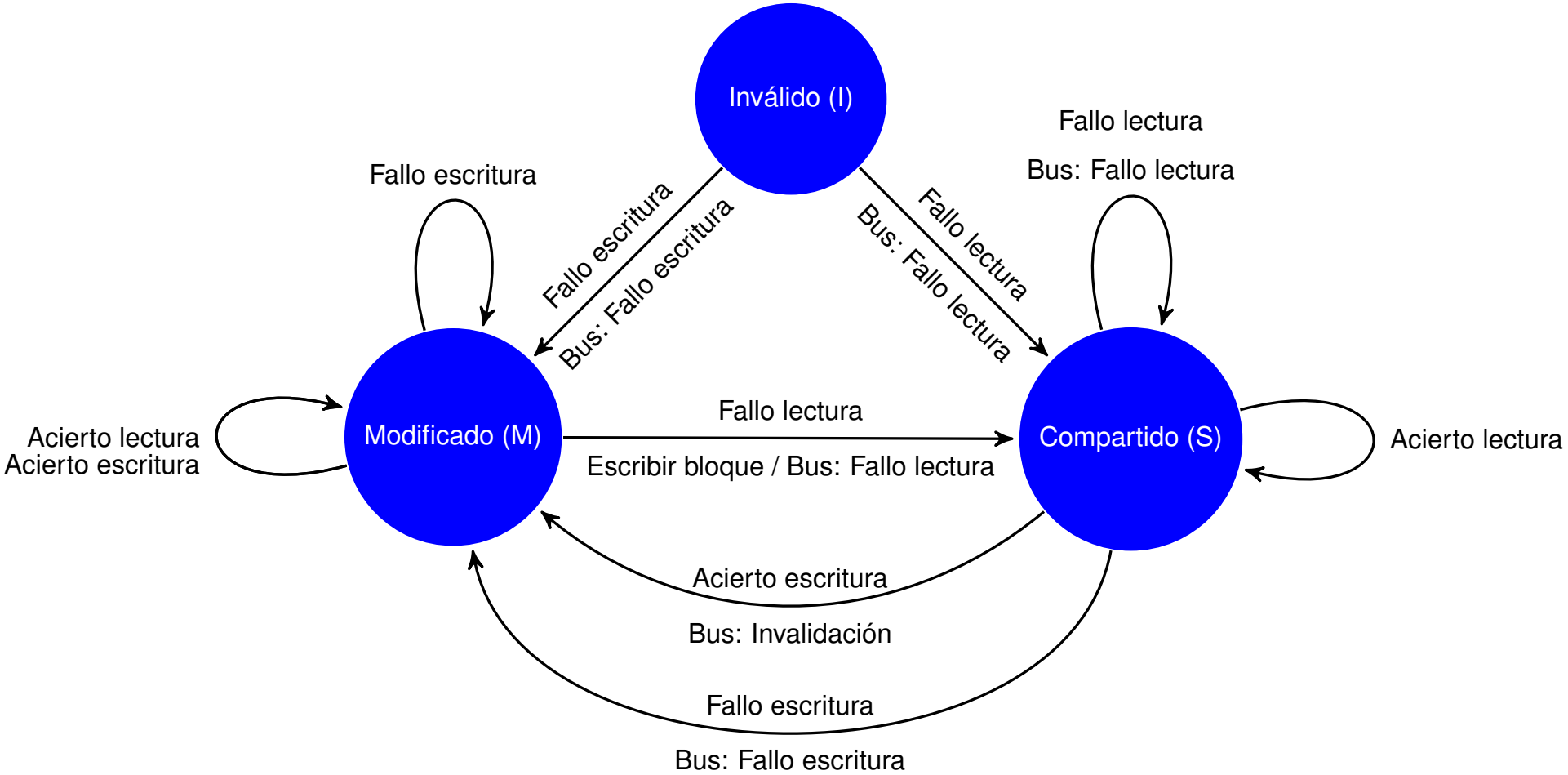
- Basado en una **máquina de estados** para **cada bloque de caché**:
  - **Cambios de estado** generados por:
    - Peticiones del procesador.
    - Peticiones del bus.
  - **Acciones**:
    - Cambios de estado.
    - Acciones sobre el bus.

- Aproximación simple con **tres estados**:
  - **M**: El bloque ha sido modificado.
  - **S**: El bloque está compartido.
  - **I**: El bloque ha sido invalidado.

Acciones generadas por el procesador

Petición	Estado	Acción	Descripción
Acierto de lectura	S → S	Acierto	Leer dato de caché local
Acierto de lectura	M → M	Acierto	Leer dato de caché local
Fallo de lectura	I → S	Fallo	Difundir fallo de lectura en bus.
Fallo de lectura	S → S	Remplazo	Fallo de conflicto de dirección. Difundir fallo de lectura en bus.
Fallo de lectura	M → S	Remplazo	Fallo de conflicto de dirección. Escribir bloque y difundir fallo de lectura.
Acierto de escritura	M → M	Acierto	Escribir dato en caché local.
Acierto de escritura	S → M	Coherencia	Invalidación en bus.
Fallo de escritura	I → M	Fallo	Difundr fallo de escritura en bus.
Fallo de escritura	S → M	Remplazo	Fallo de conflicto de dirección. Difundir fallo de escritura en bus.
Fallo de escritura	M → M	Remplazo	Fallo de conflicto de dirección. Escribir bloque y difundir fallo de escritura.

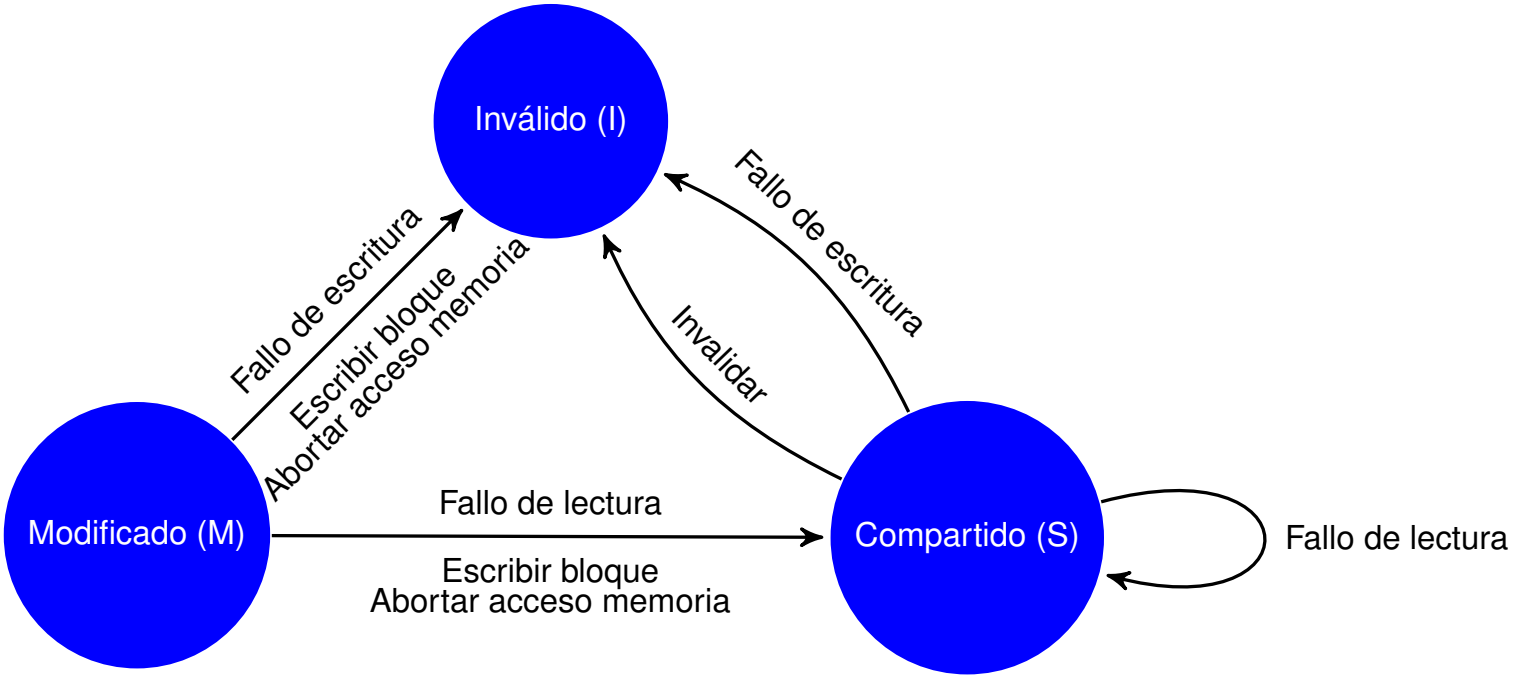
Protocolo MSI: Acciones de procesador



Acciones generadas por el bus

Petición	Estado	Acción	Descripción
Fallo de lectura	S $\rightarrow$ S	–	Memoria compartida sirve el fallo
Fallo de lectura	M $\rightarrow$ S	Coherencia	Intento de compartir dato. Se coloca bloque en bus.
Invalidar	S $\rightarrow$ I	Coherencia	Intento de escribir bloque compartido. Invalidar el bloque.
Fallo de escritura	S $\rightarrow$ I	Coherencia	Intento de escribir bloque compartido. Invalidar el bloque.
Fallo de escritura	M $\rightarrow$ I	Coherencia	Intento de escribir bloque que es. exclusivo en algún sitio. Se escribe (*write-back*) el bloque.

Protocolo MSI: Acciones de bus



Complejidad del protocolo MSI

- El protocolo asume que las operaciones son **atómicas**.
  - **Ejemplo**: Se asume que se puede detectar un fallo de escritura, adquirir el bus y recibir una respuesta en una única acción sin interrupción.
- Si las operaciones **no son atómicas**:
  - Posibilidad de deadlock y/o carreras.
- **Solución**:
  - El procesador que envía invalidación mantiene **propiedad del bus** hasta que la invalidación llega al resto.

Extensiones a MSI

- **MESI**:
  - Añade **estado exclusivo** (E) que indica que el bloque reside en una única caché pero no está modificado.
  - Escritura de un bloque E **no genera invalidaciones**.
- **MESIF**:
  - Añade **estado forward** (F): Alternativa a S que indica qué nodo debe responder a una petición.
  - Usado en Intel Core i7.
- **MOESI**:
  - Añade **estado poseído** (O) que indica que el bloque está desactualizado en memoria.
  - Evita escrituras a memoria.
  - Usado en AMD Opteron.

Rendimiento

- El uso de políticas de coherencia de caché tiene impacto sobre tasa de fallos.

- **Aparecen fallos de coherencia**:
  - Fallos de **compartición verdadera** (true sharing):
    - Un procesador escribe en bloque compartido en invalida.
    - Otro procesador lee de bloque compartido.
  - Fallos de **compartición falsa** (false sharing):
    - Un procesador escribe en bloque compartido e invalida.
    - Otro procesador lee una palabra distinta del mismo bloque.

Resumen

- Multiprocesador como computador con procesadores altamente acoplados con coordinación, uso y compartición de memoria.
- Multiprocesadores clasificados en SMP (Symetric multiprocessor) y DSM (Distributed Shared Memory).
- Dos aspectos a considerar en la jerarquía de memoria: coherencia y consistencia.
- Dos alternativas en la coherencia de caché: directorio y espionaje (*snooping*).
- Los protocolos de espionaje no requieren un elemento centralizado.
  - Pero generan más tráfico de bus.

Referencias

- **Computer Architecture. A Quantitative Approach**
  5th Ed.
  Hennessy and Patterson.
  Secciones: 5.1, 5.2, 5.3.

- **Ejercicios recomendados**:
  - 5.1, 5.2, 5.3, 5.4, 5.5, 5.6.

Memoria compartida simétrica

Arquitectura de Computadores

J. Daniel García Sánchez (coordinador)
David Expósito Singh
Javier García Blas
Óscar Pérez Alonso
J. Manuel Pérez Lobato

Grupo ARCOS
Departamento de Informática
Universidad Carlos III de Madrid