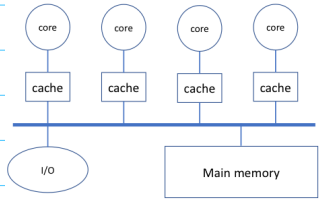
Introducción a las arquitecturas paralelas

Memoria Centralizada Compartida

Sistema con varios cores cada uno con su caché privada accediendo a la memoria principal centralizada y I/O a través de un bus común.



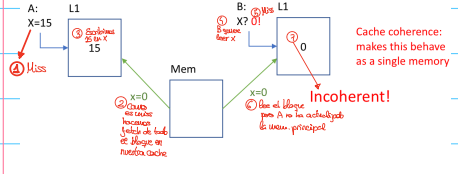
→ - Multicore
- UMA: Uniform Memory Acces (en tiempo !!)
- SMP: Symmetric Multi Processor

Problema de Coherencia de Caché

Vemos que hay mem. compartida, por lo que pensamos:

CPU A escribe x=15 → CPU B lee x, ve 15 NO sucede

Esto sucede ya que cada core tiene su propia caché privada L1 (si no sería muy lento todo)



Para que ambas CPU tengan el mismo valor, hay que buscar coherencia

Como obtener coherencia

1) No usar caches → mal rendimiento

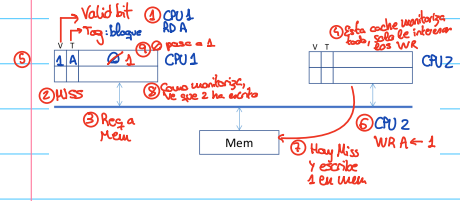
2) Todos los cores comparten una caché L1 → mal rendimiento

3) Caches privadas con write-through → incoherente, una CPU que solo lee un valor de su caché, siempre leerá el mismo, aunque otras CPU hayan modificado el valor en mem. principal

4) Forzar lectura en una caché para ver escrituras hechas por otras
   - Desde la CPU escritora – Protocolos de coherencia
     - Transmitir las escrituras para actualizar otras caches → WRITE-UPDATE } Escoger UNA política
     - Prevenir Hits en otras CPUs tras escribir → WRITE-INVALIDATE

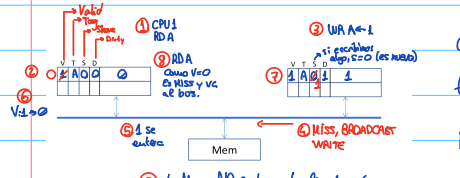
- Desde la CPU lectora – Mecanismos de coherencia
  - Escuchar los writes transmitidos en el bus compartido → SNOOPING → Escoger UN Protocolo
  - El estado de compartición de cada línea de Mem. se mantiene centralizado en una ubicación → Directory (?)

WRITE-UPDATE + SNOOPING



Si los dos escriben a la vez, se tiene que acordar quien lo hace primero

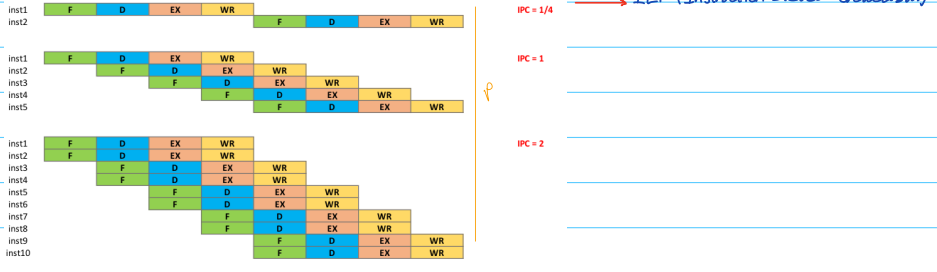
WRITE-INVALIDATE + SNOOPING



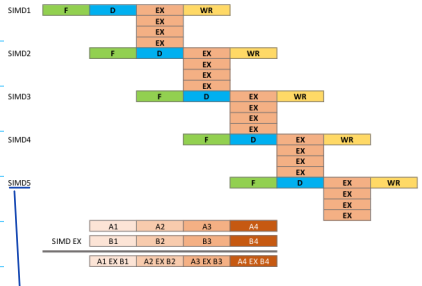
Cuando hay varias WR, el primero se transmite para invalidar los otros CPU, pero los demás ya se hacen en local

Si Mem NO se ha actualizado aún CPU 2 se entera del intento de lectura y le da a CPU 1 el dato, S: 0 → 1

Paralelismo uniprocesador



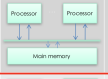
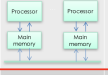
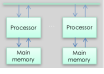
→ ILP (Instruction Level Paralelism)



→ DLP (data-level parallelism)

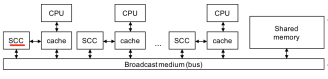
→ Single Instruction Multiple Data

Clasificación de las arquitecturas multiprocesador

Memory architecture	Address space(s)	Connection	Model for data sharing	Names
(Centralized) Shared-memory architecture	Single shared address space, uniform access time		Load/store instructions from processors	• SMP (Symmetric Multi-Processor) architecture • UMA (Uniform Memory Access) architecture
Distributed-memory architecture	Single shared address space, non-uniform access time		Load/store instructions from processors	• DSM (Distributed-Shared Memory architecture • NUMA (Non-Uniform Memory Access) architecture
	Multiple separate address spaces		Explicit messages through network interface card	• Message-passing multiprocessor • Cluster Architecture • Multicomputer

SNOOPING - Mecanismo de coherencia basado en transmisión

- La coherencia de caché se mantiene en granularidad de LÍNEA DE CACHÉ (no en los datos dentro de cada línea de la caché)
- Cada línea que tiene una copia de memoria física, mantiene su estatus de compartición
- El bus ordena las transacciones visibles para todas las cachés
- Las cachés monitorizan y actúan mediante la ayuda del SCC (Snoopy Cache Controllers)



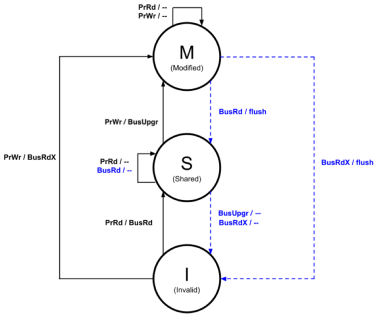
Cachés Dual-ported

Enviar mail

Protocolo simple de write-invalidate + SNOOPING

Una línea de caché puede tener 3 estados

- Modified: copia dirty de la línea
- Shared: Copia limpia de la línea
- Invalid: Copia inválida de la línea (o no existe en caché)



Eventos CPU:

- PrWr
- PrRd

Eventos bus (causados por el controlador de caché)

- BusRd: Pedir copia solo lectura
- BusRdx: Pedir copia para escribir
- BusUpgr: Pedir permiso para modificar la línea existente → causa invalidación de las copias
- Flush: Poner línea en el bus (se ha pedido a Write-back)

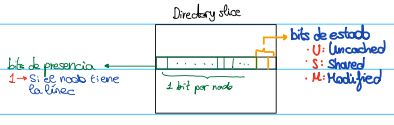
Non-Uniform Memory Access (NUMA)

Snooping funcionará en una NUMA pequeña, pero no serviría para escalar en sistemas grandes (demasiado tráfico)

→ Alternativa:

Guardar la información sobre el estado de cada línea en MP → Directory

El directory se divide en "slices", un slice por nodo



En NUMA tenemos diferentes maneras de enfocar los nodos

nodo Home: Donde se aloja la línea actualmente, el procesador que accedió por primera vez a la línea (first-touch)

Local: Nodo con el procesador que accede a la línea

Remote: Resto de nodos que contienen:

- Dirty copy (Owner)
- Clean copies (Reader)

Contenido de una página = Página / Línea → Normalmente asumiremos P=L (1 línea/página)

Comandos **local ⟷ home:**

home → envía DReply ⇝ / envía ACK ⇝

- RdReq: pide la copia de una línea sin intención de modificar.
- WrReq: pide la copia de una línea con intención de modificar.
- UpgrReq: pide permiso para modificar una línea existente, invalidando todo el resto copias.

home ⟷ remote:

remote → envía DReply → • Fetch: pide al nodo remote (owner) una copia de la línea.

remote → envía ACK → • Invalidate: pide al nodo remote (reader) que invalide su copia

Numa Nodes