COHERENCIA DE CACHÉ EN ARQUITECTURAS PARALELAS

Jose Alberto Benítez Andrades
71454586A
Ingeniería en Informática
Universidad de León

CLASIFICACIÓN DE LAS ARQUITECTURAS PARALELAS

- Múltiples procesadores → Aumento y mejora en disponibilidad de datos.
- Tipos de ordenadores según el paralelismo de las corrientes de instrucción y datos:
  - Flujo de instrucciones único, flujo de datos único (SISD)
    - Uniprocesador
  - Flujo de instrucciones único, flujo de datos múltiple (SIMD)
    - Multiprocesador, paralelismo a nivel de datos
  - Flujo de instrucciones múltiple, flujo de datos único (MISD)
    - Multiprocesador
  - Flujo de instrucciones múltiple, flujo de datos múltiple (MIMD)
    - Paralelismo a través de hilos.

CLASIFICACIÓN DE LAS ARQUITECTURAS PARALELAS

- El modelo MIMD (hilos)
  - Paralelismo a nivel de hilos
  - Elegida para multiprocesadores de uso general.
  - Son flexibles
  - Tienen ventajas de coste/rendimiento.
- Clusters (tipo de MIMD)
  - Básicos
    - Reunidos por usuarios en lugar de proveedores.
  - Personalizados
    - Explotan gran cantidad de parlaleismo en un único problema.

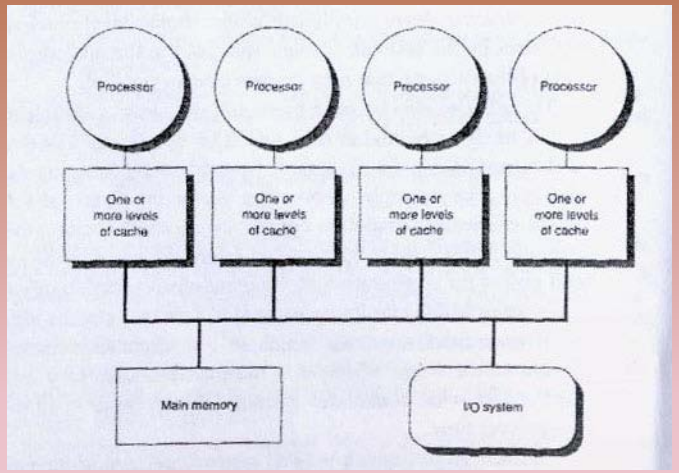
CLASIFICACIÓN DE LAS ARQUITECTURAS PARALELAS

- Clusters
  - La arquitectura multihilo → Ejecución simultánea de varios procesos con espacios de dirección separados.
  - N procesadores → N hilos o procesos a ejecutar.
  - Hilos
    - Procesos de gran escala
    - Procesos programados y manipulados por el S.O
  - Los hilos explotan el paralelismo a nivel de datos

CLASIFICACIÓN DE LAS ARQUITECTURAS PARALELAS

- Grupos de multiprocesadores MIMD:
  - 1er grupo: arquitecturas centralizadas de memoria compartida.
    - Docena de procesadores
    - Grandes cachés
    - Memoria simple
    - Múltiples bancos
  - 2º Grupo: Multiprocesadores con memoria física distribuida.
    - Procesadores centralizados → Incremento rápido en el rendimiento.

CLASIFICACIÓN DE LAS ARQUITECTURAS PARALELAS

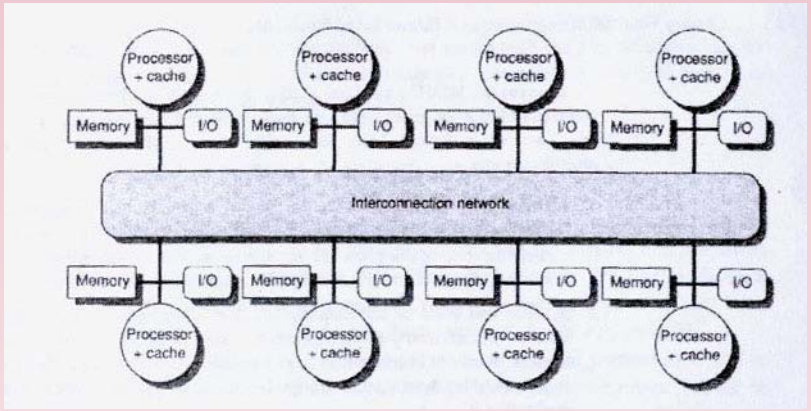


Estructura básica de un multiprocesadore de memoria compartida centralizada.

Múltiples subsistemas de procesadores comparten la misma memoria física, conectada por uno o más buses o un switch. Acceso uniforme a todas las memorias desde todos los procesadores

Arquitectura básica de un multiprocesador de memoria distribuida.

Los nodos individuales pueden contener un número pequeño de procesadores, que deben ser interconectados por un bus pequeño.

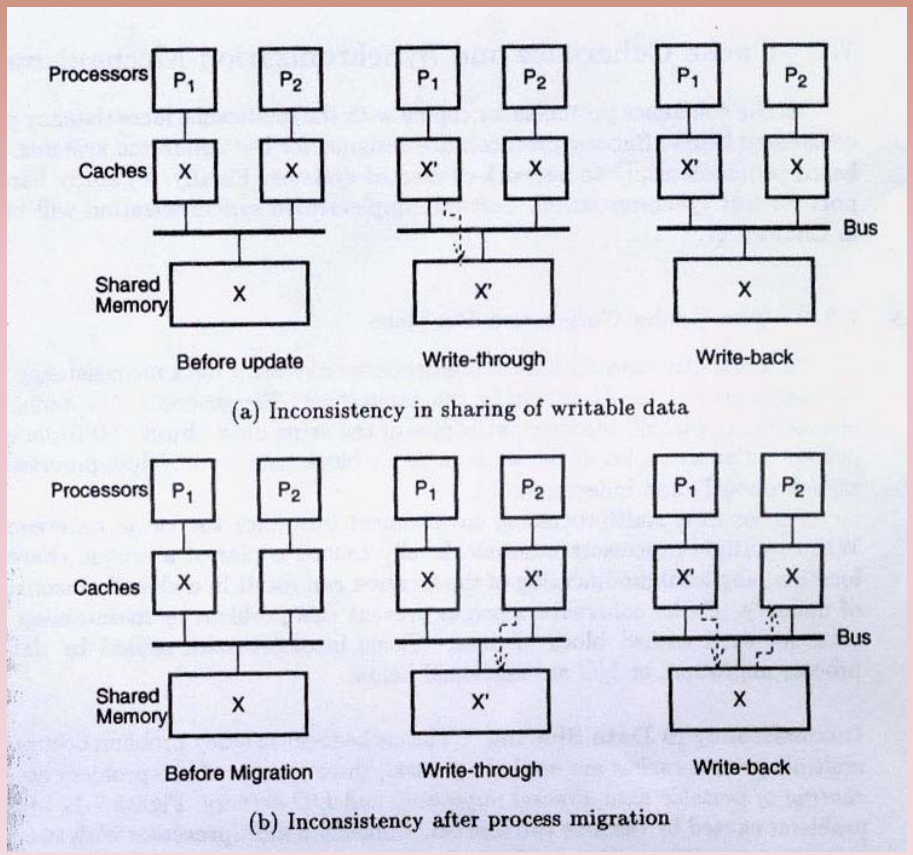


PROBLEMA DE LA COHERENCIA DE LA CACHÉ

- Inconsistencia en la compartición de datos
  - Múltiples caches privadas son usadas:
    - 3 fuentes de problema:
      - Compartición de datos escribible
      - Proceso de migración
      - Actividad de E/S
- Compartición de datos escribible
  - X -> dato compartido referenciado por 2 procesadores
  - P1 escribe nuevo dato X' en caché.
  - Copia en memoria compartida
  - Inconsistencia entre X y X'

PROBLEMA DE LA COHERENCIA DE LA CACHÉ

- Proceso de migración y E/S



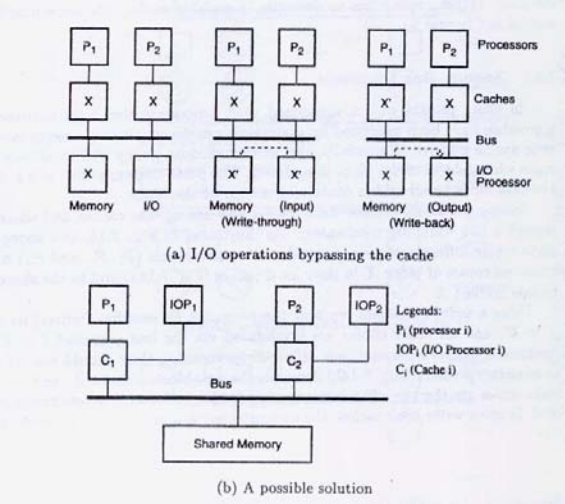
Problemas de coherencia de caché en la compartición de datos y el proceso de migración.

Muestra incoherencia después de un proceso que contiene una variable compartida X y migra del procesador 1 al procesador 2 usando write-back caché.

Solución: Agregar procesadores de E/S a las cachés privadas. Así comparten la CPU.

PROBLEMA DE LA COHERENCIA DE LA CACHÉ

- Dos enfoques de protocolo.
  - Sistemas de memoria basados en bus
    - BUS → Dispositivo conveniente para garantizar coherencia de las cachés, permite observar transacciones de memoria en curso.
    - Si una transacción amenaza → el controlador de caché puede actuar e invalidar la copia local
    - Los protocolos se usan para garantizar la coherencia → protocolos snoopy, por los snoops que hay en cada transacción.



Incoherencia en la caché después de operaciones de E/S y posible solución

MECANISMOS DE SINCRONIZACIÓN

- Protocolos snoopy
  - Cachés privadas vinculadas por un bus común:
    - Política de write-invalidate
    - Política de write-update
- Protocolos basados en directorio
  - Se utilizan en red multietapas ya que los snoopy son muy caros para adaptarse a las capacidades de la red.

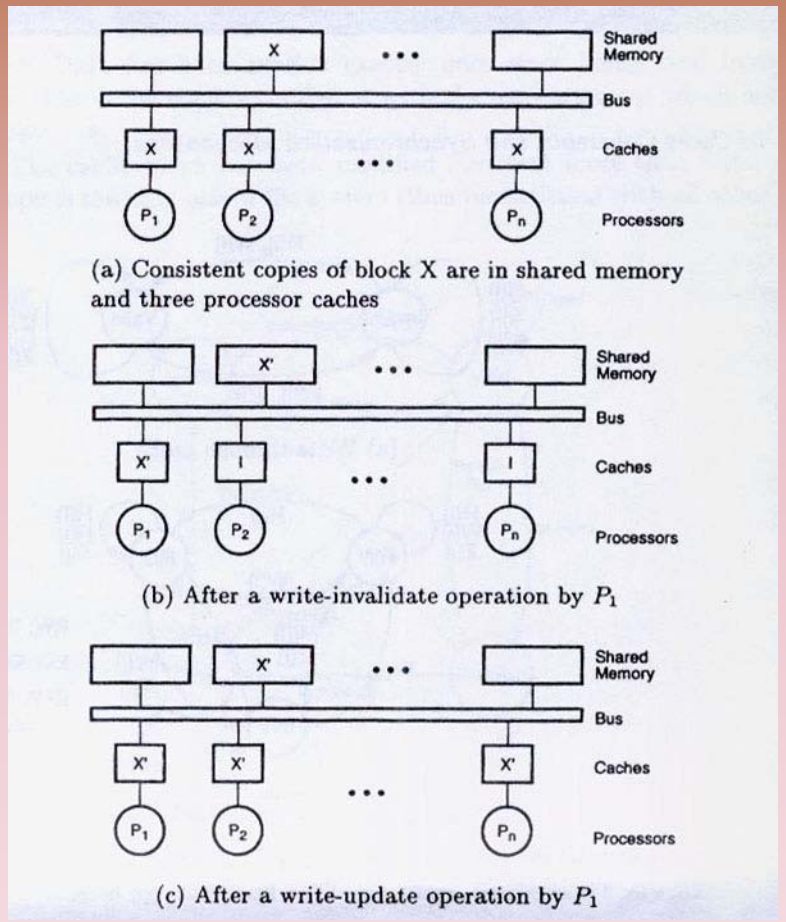
MECANISMOS DE SINCRONIZACIÓN

Protocolos snoopy

- 2 Enfoques para mantener la coherencia
  - Write-invalidate: Invalidará todas las copias remotas cuando un bloque local sea actualizado.
  - Write-update: emitirá el nuevo bloque de datos a todas las cachés contenido una copia del bloque.
- Logran coherencia de datos entre cachés y memorias compartidas a través de un bus.

MECANISMOS DE SINCRONIZACIÓN

Protocolos snoopy



(a) Consistent copies of block X are in shared memory and three processor caches

(b) After a write-invalidate operation by P_1

(c) After a write-update operation by P_1

Usando write-invalidate, P1 modifica sus cachés de X a X', y todas las copias se invalidan mediante el bus.

Los bloques invalidados son "sucios". El protocolo write-update demanda un nuevo contenido de bloque X', que será emitido a todas las cachés mediante el bus.

La copia de memoria es también actualizada si las cachés write-through son utilizadas.

En el uso de cachés write-back, la copia de la memoria es actualizada después del tiempo de reemplazamiento de bloque.

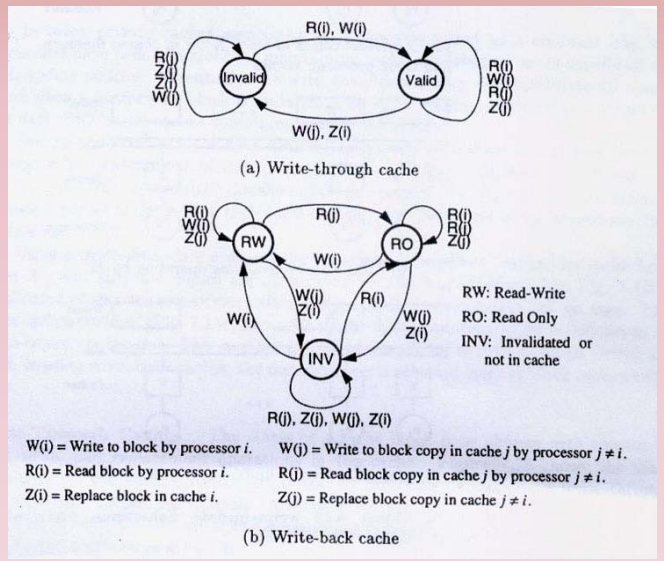
MECANISMOS DE SINCRONIZACIÓN

Protocolos snoopy

- Cachés write-through

Los estados de la copia de un bloque de caché cambia con respecto a operaciones read o write.

La fracción de ciclos de escritura en el bus es mayor que la fracción de ciclos de lectura en una caché write-through, debido a la necesidad de solicitudes de invalidación.



Se observan 2 protocolos snoopy de write-invalidate diseñados por write-back y write-through.

Para cada uno de los 2 estados de caché, hay 6 eventos que pueden llevarse a cabo.

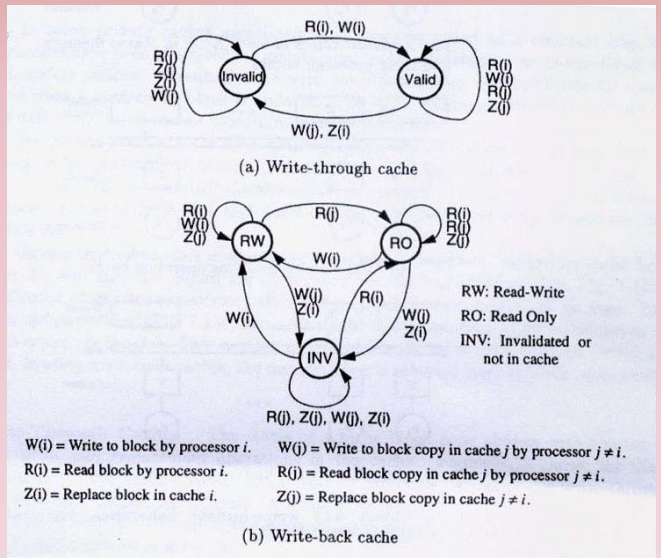
MECANISMOS DE SINCRONIZACIÓN

Protocolos snoopy

- Cachés write-back

El estado validado de una caché write-back puede ser dividido en otros 2 estados: RW(Read-Write) y RO(Read-only)

Pueden existir muchas copias en el estado RO, cada proceso tiene una copia, puede leer la copia segura.
Antes de que un bloque sea modificado, la propiedad de acceso exclusivo debe ser obtenida por un bus de transacciones RO que es emitido a todas las cachés y memorias



El estado INV es equivalente al invalidate mencionado anteriormente.

Cuando la memoria posee un bloque, las cachés pueden contener solo copias RO del bloque.

MECANISMOS DE SINCRONIZACIÓN

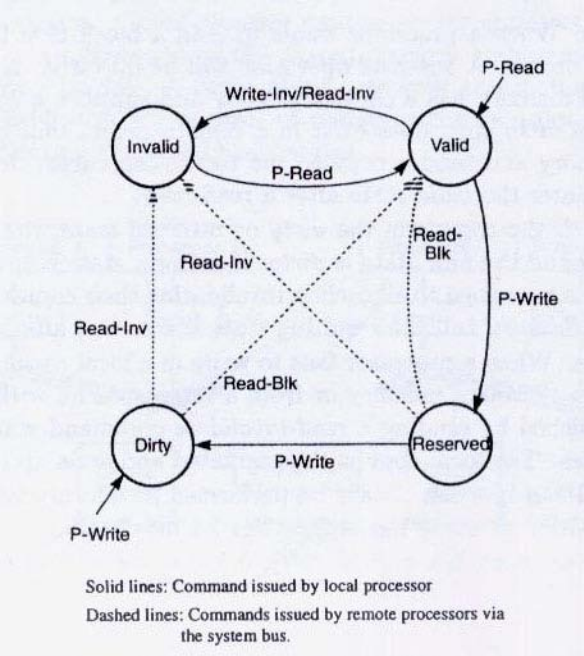
Protocolos snoopy

- Protocolo write-once
  - Goodman propuso un protocolo de coherencia caché para multiprocesadores “bus-based”.
  - Combina la ventajas de invalidaciones write-through y write-back.
  - Primero usará write-through.
  - Después del primer write, la memoria compartida es actualizada usando la política de write-back.
  - 4 estados principales
    - Válido: El bloque coherente con la copia de la memoria, ha sido leído por la memoria compartida y no se ha modificado.
    - Inválido: El bloque no se encuentra en la caché o es incoherente con la copia de la memoria compartida
    - Reservado: Los datos han sido escritos correctamente.
    - Sucio: El bloque de caché ha sido escrito más de una vez y la copia de caché es la única del sistema (incoherente con el resto de copias).

MECANISMOS DE SINCRONIZACIÓN

Protocolos snoopy

- Protocolo write-once



Protocolos de coherencia write-once de Goodman usando la política de write-invalidate en cachés write-back.

Las líneas sólidas son los comandos emitidos por un procesador local con la etiqueta read-miss, write-hit y write-miss.

Read-miss lleva a estado válido.
El 1er Write-hit conduce a reserva
El 2º Write-hit conduce al estado sucio.
Read-invalidate lee un bloque e invalida todas las otras copias.
Write-invalidate invalida todas las copias de un bloque.

MECANISMOS DE SINCRONIZACIÓN

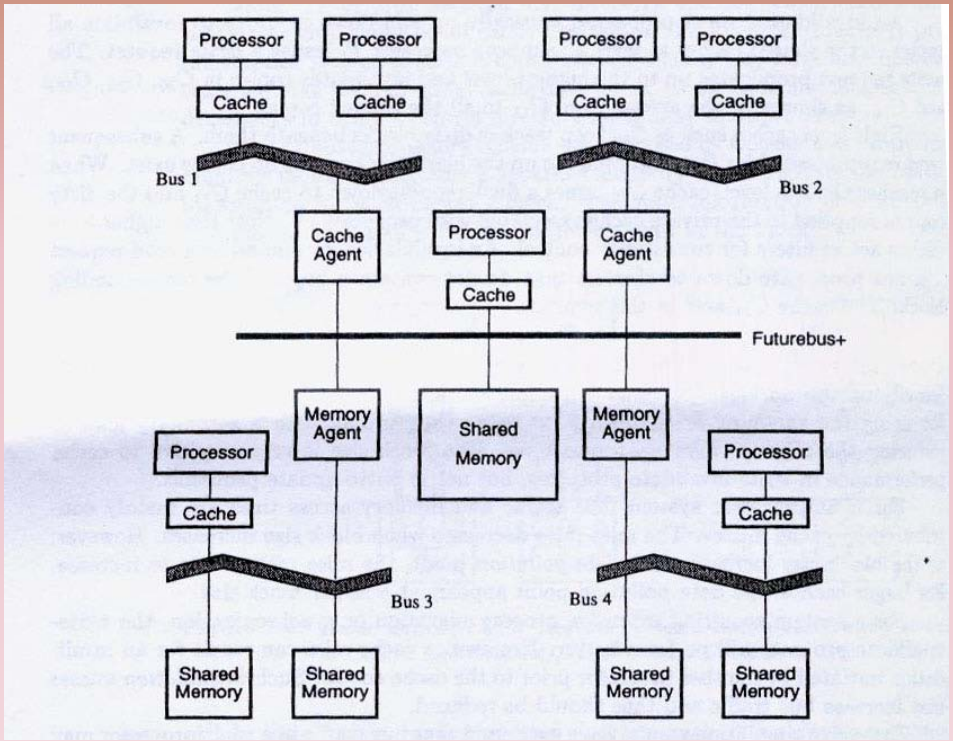
Protocolos snoopy

- Eventos y acciones de caché.
  - El acceso a memoria y los comandos de invalidación desencadenan los siguientes eventos y acciones:
    - Read-miss: El bloque no está en la caché.
    - Write-hit: La escritura puede ser llevada a una salida localmente y el nuevo estado es sucio.
    - Write-miss: La copia vuelve cuando falla al escribir en la caché local.
    - Read-hit: Pueden ser realizados en una caché local sin causar una transición de estado.
    - Block Replacemente: Si una copia está sucia, se vuelve a escribir en la memoria principal mediante esta acción.

MECANISMOS DE SINCRONIZACIÓN

Protocolos snoopy

- Protocolo Futurebus+



Los futurebus+ soportan transacciones de conexión y división.

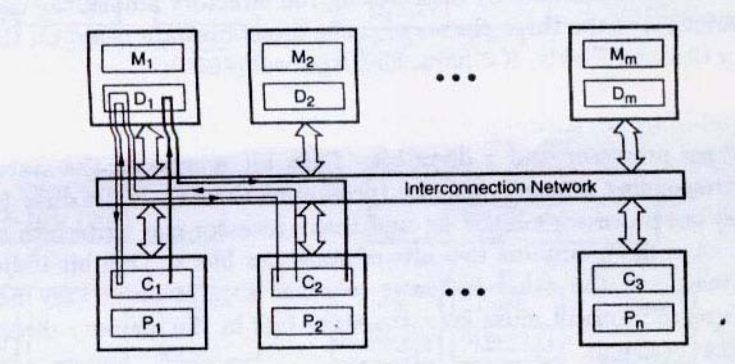
Es una extensión de la arquitectura de memoria-compartida con módulos agente de caché y memoria.

Un agente caché usa transacciones de división para asumir las responsabiilidades de módulos de caché remotos.

MECANISMOS DE SINCRONIZACIÓN

Protocolos basados en directorio

- Se necesitan para redes multietapas para construir multiprocesadores grandes.
- En una red multietapas la coherencia de caché es soportada usando directorios caché para almecnar la información.
- Tang propuso el primer esquema de directorio que usó un directorio central que contenía una copia de todos los directorios caché.



Concepto básico de un esquema de caché basado en directorio.

MECANISMOS DE SINCRONIZACIÓN

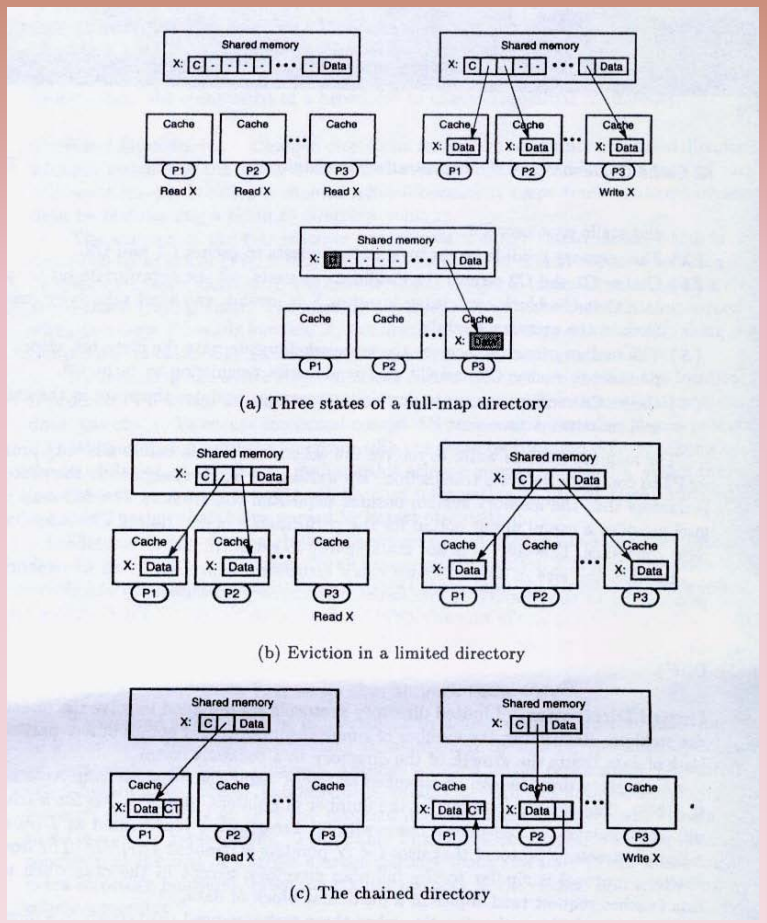
Protocolos basados en directorio

- Los protocolos full-map implementan entradas de directorios con un bit por procesador y un bit sucio.
- Cada bit representa el estado del bloque en la correspondiente caché del procesador.
- Si el bit sucio está activo, entonces un bit de procesador es puesto y el procesador puede escribir en el bloque.
- En la figura que se observa en la siguiente diapositiva se mostraran los 3 estados diferentes de un directorio full-map:

MECANISMOS DE SINCRONIZACIÓN

Protocolos basados en directorio

- Full-map, directorios limitados y directorios encadenados.



Hay 3 estados distintos en un directorio full-map

Los de directorio limitado son diseñados para resolver el problema del tamaño de directorio. Restringen las copias de caché simultáneas.

Los directorios encadenados realizan la escalabilidad de directorios limitados sin restricción de número de copias compartidas de bloques de datos. Este tipo de esquemas de coherencia de caché son llamados esquemas encadenados, porque guardan pistas de copias compartidas de datos por mantener una cadena de punteros de directorio.