Arquitectura e Ingeniería de Computadores

Modulo III. Multiprocesadores

Tema 8. **Coherencia**. Consistencia. Sincronización.

Coherencia cache escalable

- Esquema
  - Multiprocesadores escalables
  - Coherencia cache escalable
  - Protocolos basados en Directorio
  - Organización del Directorio
    - Directorios planos basados en memoria
    - Directorios planos basados en cache

Recordatorio: Protocolos de Coherencia en cache (1)

Protocolos para implementar las **políticas de coherencia** en caches

Dependen de la red de interconexión

Recordatorio: Protocolos de Coherencia en cache (2)

- **Broadcast eficiente/ Buses compartidos**
  - Las operaciones de invalidación o actualización se pueden enviar de forma simultanea a todos los controladores.

Buses: Protocolo Snoopy

Observación del Bus

- El controlador de cache de cada procesador espía los paquetes que hay en el bus y actúa en consecuencia (invalidando o actualizando la copia que tiene de un bloque)

Recordatorio: Protocolos de Coherencia en cache (3)

- **Protocolos de coherencia *Snoopy***
  - Invalidación
    - **Snoopy 2 estados**
    - **MSI**
    - **MESI (Illinois)**
    - **MOESI**
  - Actualización
    - **Dragón**
- **Medio de comunicación = bus**
  - Ventajas
    - Orden = Arbitraje de bus
  - Desventajas
    - Escalabilidad limitada
      - Longitud y nº de conexiones

Recordatorio: Protocolos de Coherencia en cache (4)

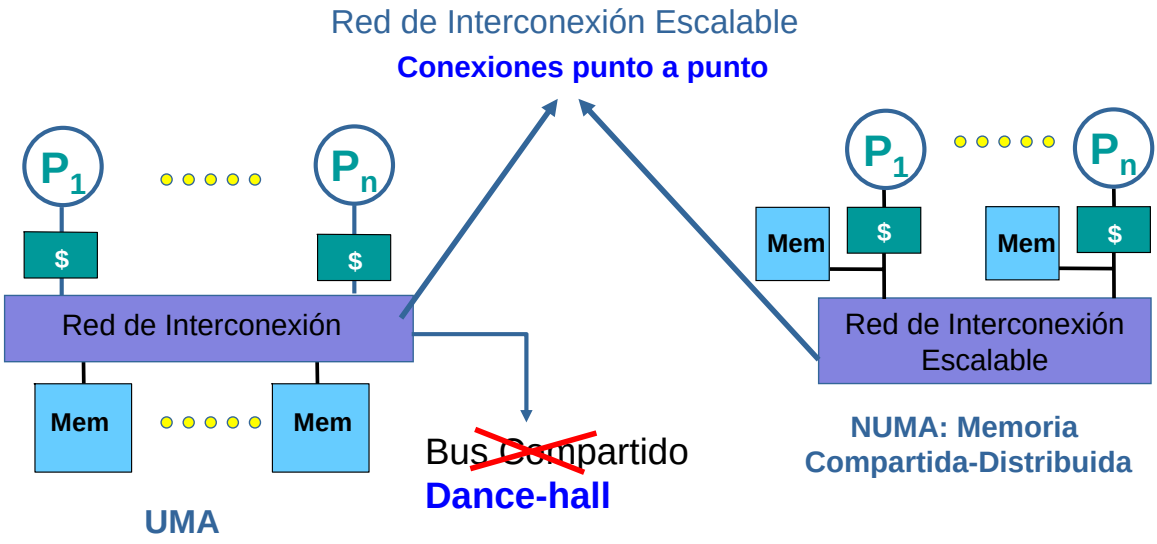
- **Redes con difusión costosa de implementar**
- **Redes en las que se requiere mayor escalabilidad**
  - La actualización/invalidación se envía únicamente a aquellas caches que tienen una copia del bloque.

Protocolos Basados en Directorio

Se indica en que caches existe copia y en que estado está dicha copia

Protocolos basados en Directorio (1)

- **Donde se aplican:**
  - Sistemas de memoria distribuida
  - **Sistemas de memoria compartida**
    - UMA con Dance-hall
    - **NUMA**: Memoria Compartida Distribuida



Protocolos basados en Directorio (2)

- Ahora **No hay Bus Compartido:**
  - **No se ven las transacciones de los otros nodos (procesadores)**
    - Cómo saber el estado del bloque en otras caches
    - Cómo comunicarse con las otras copias para:
      - obtener datos,
      - invalidarlas/actualizarlas

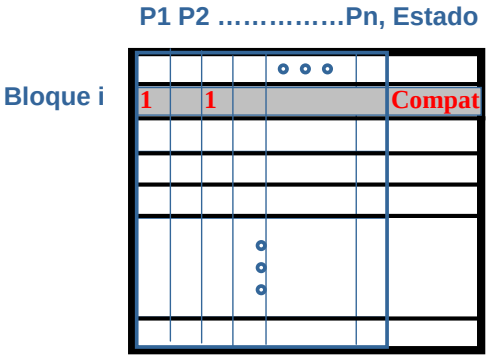
- **Desafío**: Arquitectura NUMA
  - Mantener coherencia mediante hw con granularidad de bloque cache al igual que en las máquinas basadas en bus

Protocolos basados en Directorio (3)

- **Se basan en usar un Directorio**: Cada bloque de memoria tiene asociado:
  - **Una entrada en el directorio** con información sobre:
    - Las caches con copia valida de ese bloque
    - Estado del bloque en memoria
      - Actualizado, no actualizado, compartido..

Información global

Directorio



Posible implementación del directorio

  - **Estado en Caches**: **Información Local**

Protocolos basados en Directorio (4)

- **Modo de actuar**:
  - Si se produce un **Fallo de Lectura**
    - **Comunicación con el directorio**
      - Búsqueda de bloque
        - Lo proporciona la memoria o la cache que lo ha modificado
  - Si se produce una **Escritura**
    - **Comunicación con el directorio**
      - Información de caches con copia del bloque
      - Si es fallo de escritura, Búsqueda de bloque
        - Lo proporciona la memoria o la cache que lo ha modificado
    - **Comunicación con los nodos que comparten el bloque**
      - Para Invalidar/ actualizar
- Las Comunicaciones son **punto-a-punto**
  - **Envío de mensajes**

Protocolos basados en Directorio (5)

- **Coherencia en los protocolos basados en directorio**

¿Cumple los 2 requisitos de coherencia?

- **Propagación de las escrituras:**
  - Se logra con la información del directorio, y enviando los mensajes correspondientes a los nodos que tengan copias.

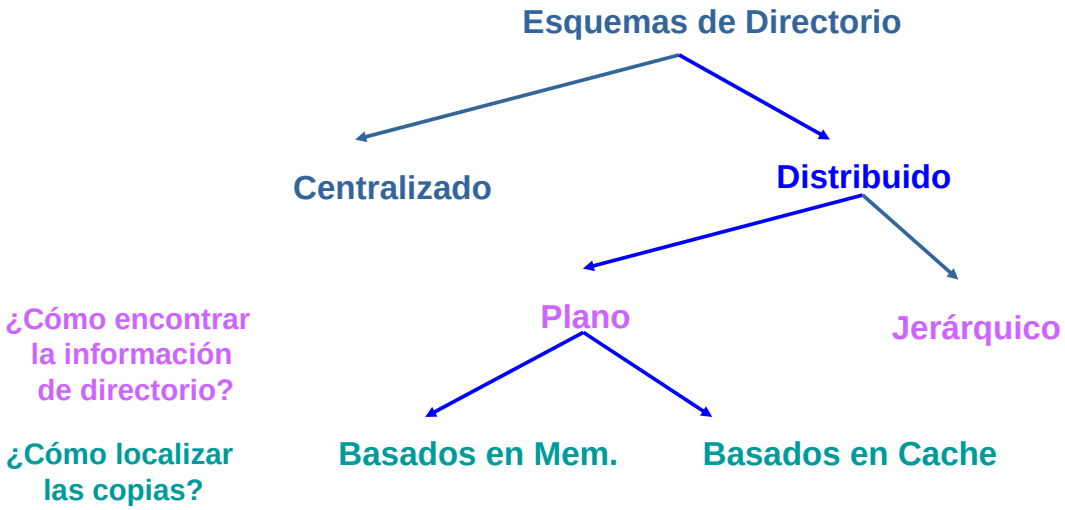
- **Serialización de escrituras**
  - El orden total lo impone acceso al directorio

Protocolos basados en Directorio (6)

- La Escalabilidad en estos protocolos está determinada por:

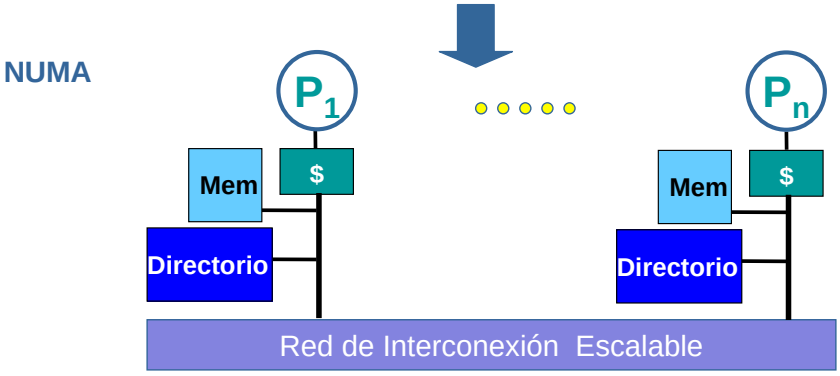
  - Rendimiento
    - Demanda de BW (Tráfico):
      - Transacciones de red por fallo (multiplicado por la frecuencia de fallos)
    - Latencia:
      - Transacciones red en camino crítico del fallo (espera)
    - Estos dos factores dependen de
      - La organización del directorio
      - Cómo esté optimizado el flujo de transacciones de red
  - Sobrecarga de memoria
    - Depende de la organización del directorio

Organización del Directorio (1)



Organización del Directorio (2)

- **Directorio centralizado**
  - Contiene información de todos los bloques de memoria
  - Atiende todas las peticiones de acceso a memoria generadas por el sistema
    - **Supone cuello de botella**
    - Primeros Esquemas de Directorio: en Mainframes de IBM, Centralizados .
- **Directorio distribuido entre los módulos de memoria principal**
  - Cada modulo de memoria tiene un directorio con información de los bloques que contiene

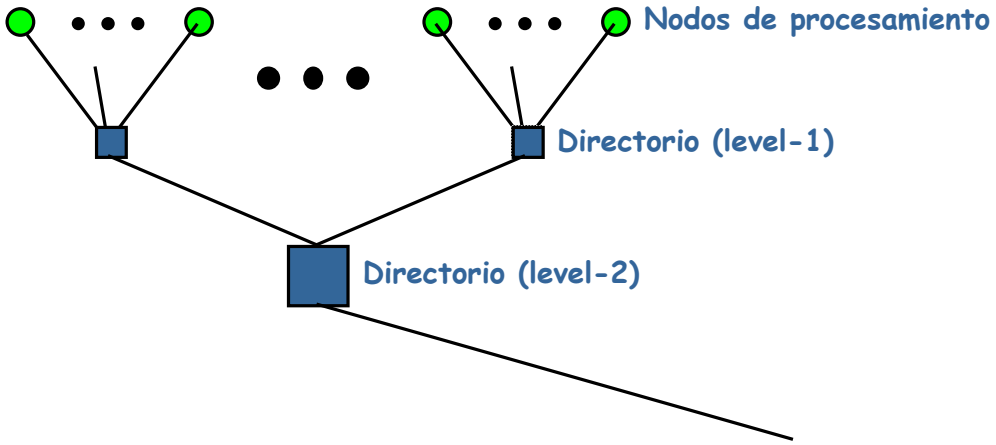


Organización del Directorio (2)

- **Esquema Plano**
  - El origen de la información de directorio para un bloque se encuentra en un lugar fijo
    - Este suele venir determinado por la dirección del bloque
  - Ante un fallo de cache
    - Se consulta el directorio
      - Si no está en el propio nodo se envía un transacción de red directamente al nodo donde se encuentra
- Tipos
  - **Basado en Memoria**
    - La información de directorio acerca de un determinado bloque está almacenada en el nodo origen de dicho bloque
    - Stanford DASH/FLASH, SGI Origin, MIT Alewife, HAL
  - **Basado en cache**
    - La información de directorio acerca de un determinado bloque no está totalmente almacenada en el nodo origen sino distribuida entre las propias copias
    - IEEE SCI, Sequent NUMA-Q

Organización del Directorio (3)

- **Esquema Jerárquico** (Data Diffusion Machine)
  - La información del directorio está organizada en una estructura de datos jerárquica (no necesariamente física)
    - **Hojas del árbol**: los nodos de procesamiento con su porción de memoria
    - **Nodos intermedios**: información del directorio relativa al subarbol
      - Qué bloques locales están replicados fuera del subárbol
      - Qué hijos tienen copia de bloques remotos



Organización del Directorio (4)

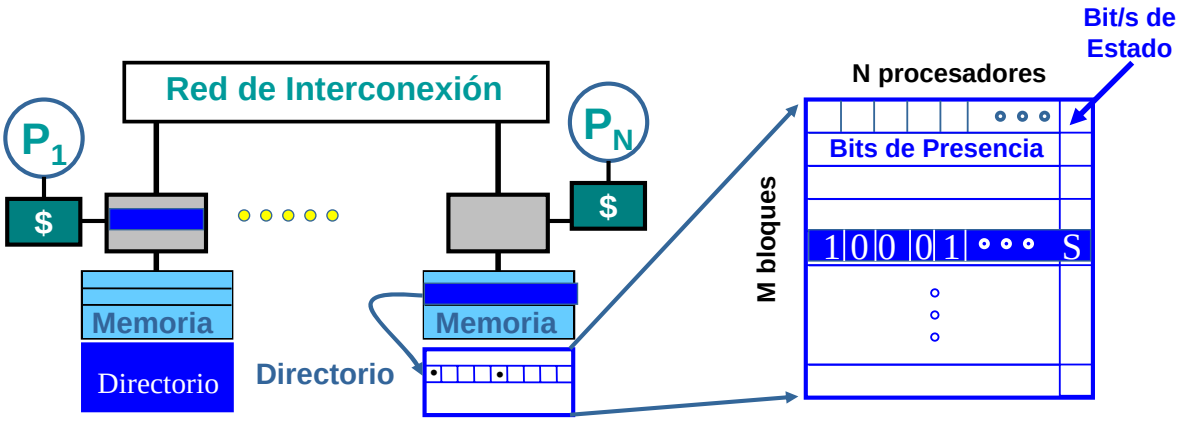
- **Ventajas** potenciales de los **directorios jerárquicos**
  - En un fallo de lectura a un bloque cuyo origen se encuentra lejano en la topología de la red de interconexión:
    - Se envía al procesador una **copia más cercana** en la topología
    - No tiene que recorrer todo el camino hasta el nodo origen
  - Las peticiones de varios nodos pueden combinarse en un ancestro común de la jerarquía
- **Inconvenientes** potenciales de los **directorios jerárquicos**
  - Las transacciones de red necesarias para consultar la jerarquía puede ser mucho más numerosas que las que generan en una consulta en directorios planos
  - Mayores requisitos de latencia y ancho de banda

Directorios Distribuidos Planos

Basados en Memoria

Directorios Planos Basados en memoria (1)

- **Plano Basado en Memoria**
  - La información de directorio acerca de un determinado bloque está almacenada en el nodo origen de dicho bloque
  - Implementación más natural: **Full Bit Vector**
    - 1 bit de presencia por nodo
    - Estado: uno o más bits
      - Depende del protocolo implementado
      - Más simple: un único bit (bit de modificación)



Directorios Planos Basados en memoria (2)

- **Diseño del protocolo:**
  - Definir **Política de actualización** de la memoria principal
    - Escritura inmediata
    - **Post-escritura**
  - Definir **Política de coherencia** de caché
    - Escritura con **invalidación**
    - Escritura con actualización

Directorios planos basados en memoria (3)

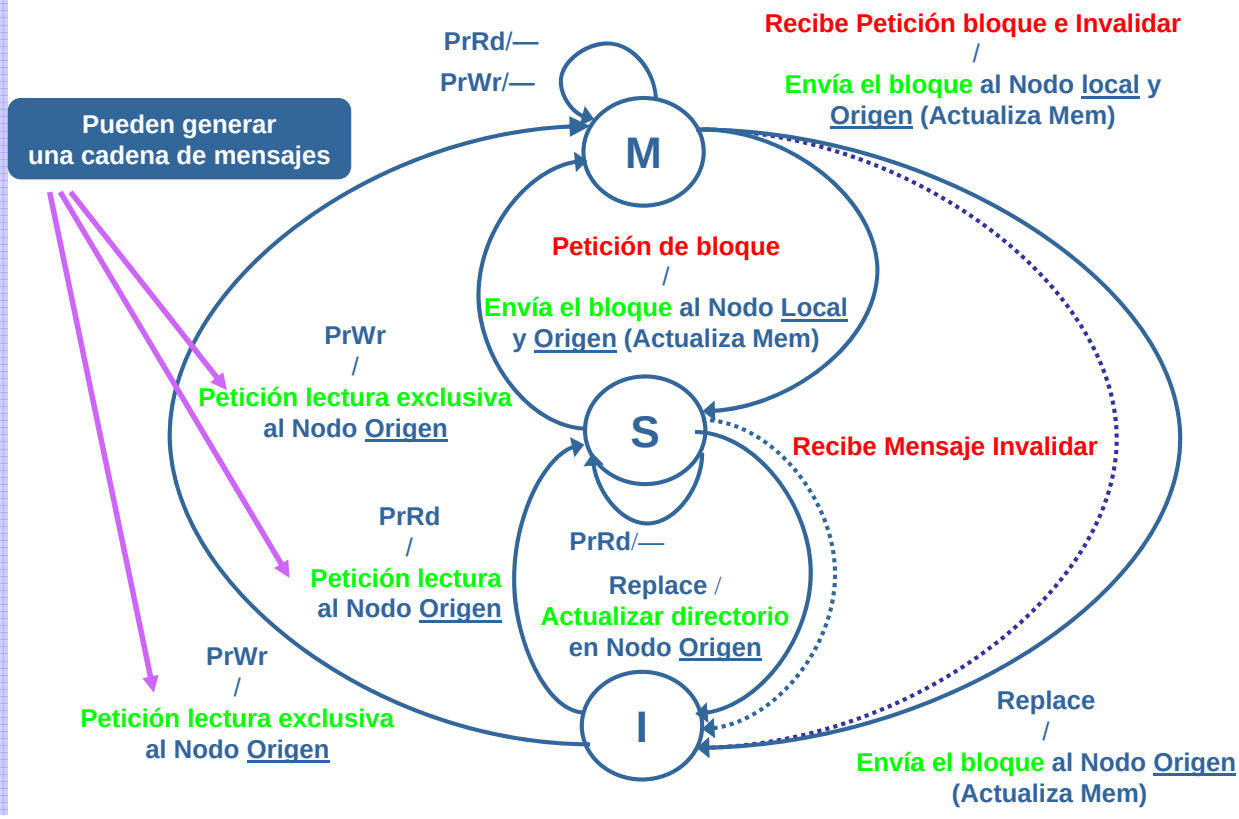
- Transferencias **Punto-a-Punto**
  - **Necesidad de paso de mensajes**
    - **Petición:**
      - Mensaje enviado por un nodo (llamado solicitante) a otro nodo
      - Puede ser:
        - Un Fallo Lectura, una Escritura, una Post-Escritura
        - La petición de un bloque de su cache
        - Una invalidación
    - **Respuesta:**
      - Mensaje enviado al nodo solicitante en respuesta a su petición
      - Pueden contener:
        - Un bloque de memoria
        - Identificador de nodo
        - Confirmación de invalidación

Directorios planos basados en memoria (4)

- Nodos que pueden estar implicados en una transferencia de mensajes:
  - Nodo Peticionario (local):
    - Contiene el procesador que emite una petición del bloque
  - Nodo Origen (home):
    - Nodo en cuya porción de memoria principal se encuentra el bloque (contiene el directorio)
  - Nodo Compartidor (shared)
    - Nodo con una copia válida de bloque en cache
  - Nodo Propietario (owner):
    - Tiene una copia del bloque en su cache en estado modificado. Su cache tiene la única copia válida del bloque. La memoria no está actualizada.

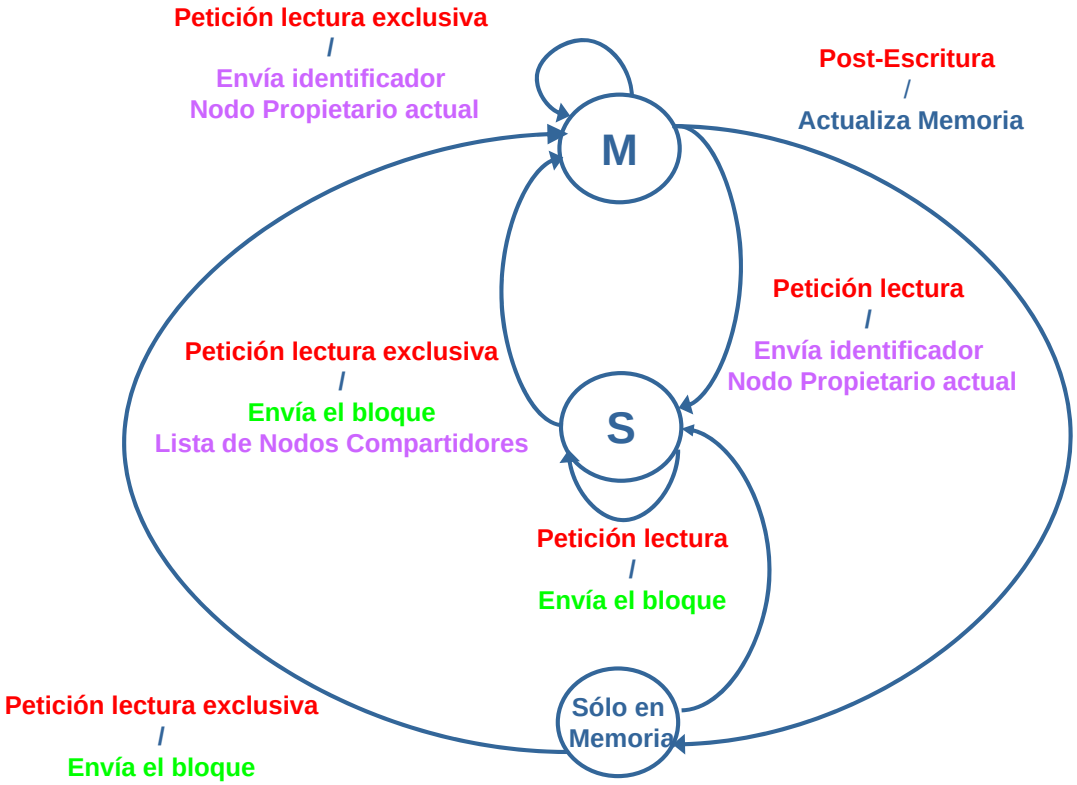
Directorios planos basados en memoria (5)

Transiciones para un bloque de cache: **Protocolo MSI**



Directorios planos basados en memoria (6)

Transiciones para un bloque en **el Directorio**



Directorios planos basados en memoria (7)

Algoritmo de funcionamiento del protocolo del Directorio:

- Tenemos:
  - 1 nivel de cache, 1 procesador por nodo
  - Política de reemplazamiento **post-escritura**
  - Política de coherencia **invalidación y** Protocolo de coherencia en las caches **MSI**

- **Nodo$_i$ Solicita un bloque (Fallo de lectura/escritura)**
  - Buscar a que porción de memoria principal hay que acceder.
    - **Si es Local** →
      - El nodo origen es el propio nodo Local → acceso directo a la memoria
      - Cuando recibe el bloque actualiza su estado en la cache
        - Estado= Compartido
    - **Si es Remota** →
      - Establece comunicación con el nodo cuya memoria principal contiene el bloque (nodo origen) → Envió de mensajes
      - Cuando recibe el bloque actualiza su estado en la cache
        - Estado= Modificado

Directorios planos basados en memoria (8)

Algoritmo de funcionamiento del protocolo del directorio: **Fallo de Lectura**

- **Nodo Origen: Recibe la petición del Nodo$_i$ y le envía respuesta**
  - Consulta la entrada de ese bloque en el directorio
    - Bit-Estado=Compartido o Sólo en Memoria →
      - Envía el bloque de memoria a la cache del Nodo$_i$
      - Actualiza el directorio
        - Bit-presencia[Nodo$_i$]=SI
        - Bit-Estado=Compartido
    - Bit-Estado=Modificado →
      - Envía la identidad del nodo que tiene el bloque actualizado
      - Actualiza el directorio
        - Bit-presencia (Nodo$_i$)=SI
        - Bit-Estado=Compartido
      - **Nodo Local** establece **comunicación con el nodo propietario**
        - En su cache el bloque pasa a estado **Compartido**
        - Envía el bloque de su cache a la cache del Nodo$_i$
        - Envía el bloque de su cache a la memoria del nodo origen para actualizar la memoria (post-escritura)

Directorios planos basados en memoria (9)

Algoritmo de funcionamiento del protocolo del directorio: **Fallo de Escritura**

- **Nodo Origen: Recibe la petición del Nodo$_i$ y le envía respuesta**
  - Consulta la entrada de ese bloque en el directorio
    - Bit-Estado=Compartido o Sólo en Memoria →
      - Envía la lista de identidades de los nodos que comparten el bloque
      - Envía el bloque de memoria a la cache del Nodo$_i$
      - Actualiza el directorio
        - Bit-Estado=Modificado
        - Bit-presencia (Nodos compartidos) =NO
        - Bit-presencia (Nodo$_i$)=SI
      - **Nodo Local** establece **comunicación con los nodos compartidores**
        - En cada una de las caches de esos nodos el bloque pasa a estado **Inválido**
    - Bit-Estado=Modificado →
      - Envía la identidad del nodo que tiene el bloque actualizado
      - Actualiza el directorio
        - Bit-presencia (Nodo propietario) =NO
        - Bit-presencia (Nodo$_i$)=SI
        - Deja Bit-Estado=Modificado
      - **Nodo Local** establece **comunicación con el nodo propietario**
        - En la cache de este nodo el bloque pasa a estado **Inválido**
        - Envía el bloque de su cache a la cache del Nodo$_i$

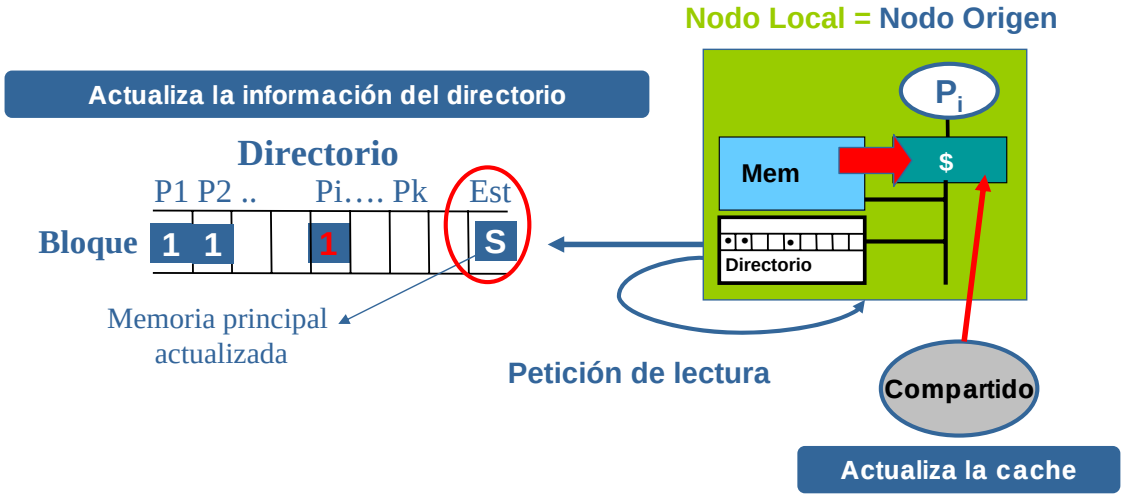
Directorios planos basados en memoria (10)

Algoritmo de funcionamiento del protocolo:

- **Nodo$_i$ Solicita post-escritura**
  - Buscar a que porción de memoria principal hay que acceder.
- **Nodo Origen: Recibe la petición del Nodo$_i$ y le envía respuesta**
  - **Si es Local** →
    - Se escribe directamente en memoria
  - **Si es Remota** →
    - **Nodo$_i$** establece **comunicación con el nodo origen** y se envía el bloque a su porción de memoria principal
  - Actualiza el directorio
    - Bit-Estado=Sólo en Memoria (la memoria se acaba de actualizar)
    - Bit-presencia (Todos los Nodos) =NO

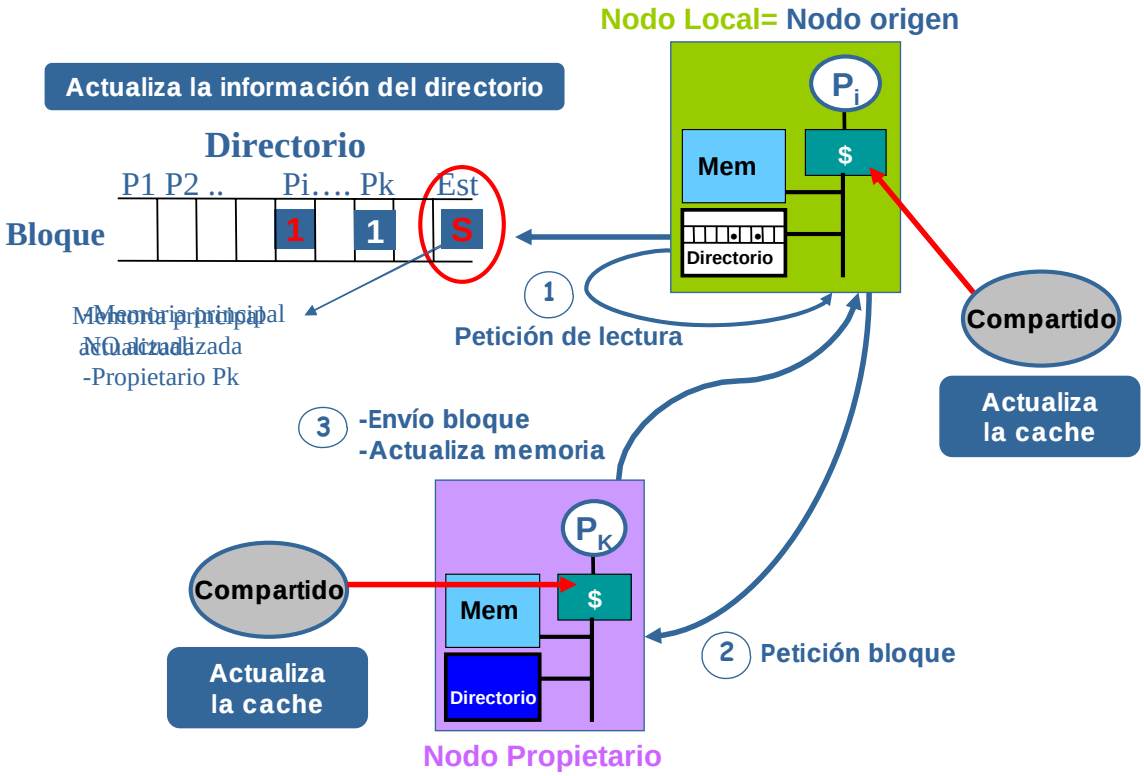
Directorios planos basados en memoria (11)

- Fallo de lectura Local sobre Bloque Compartido



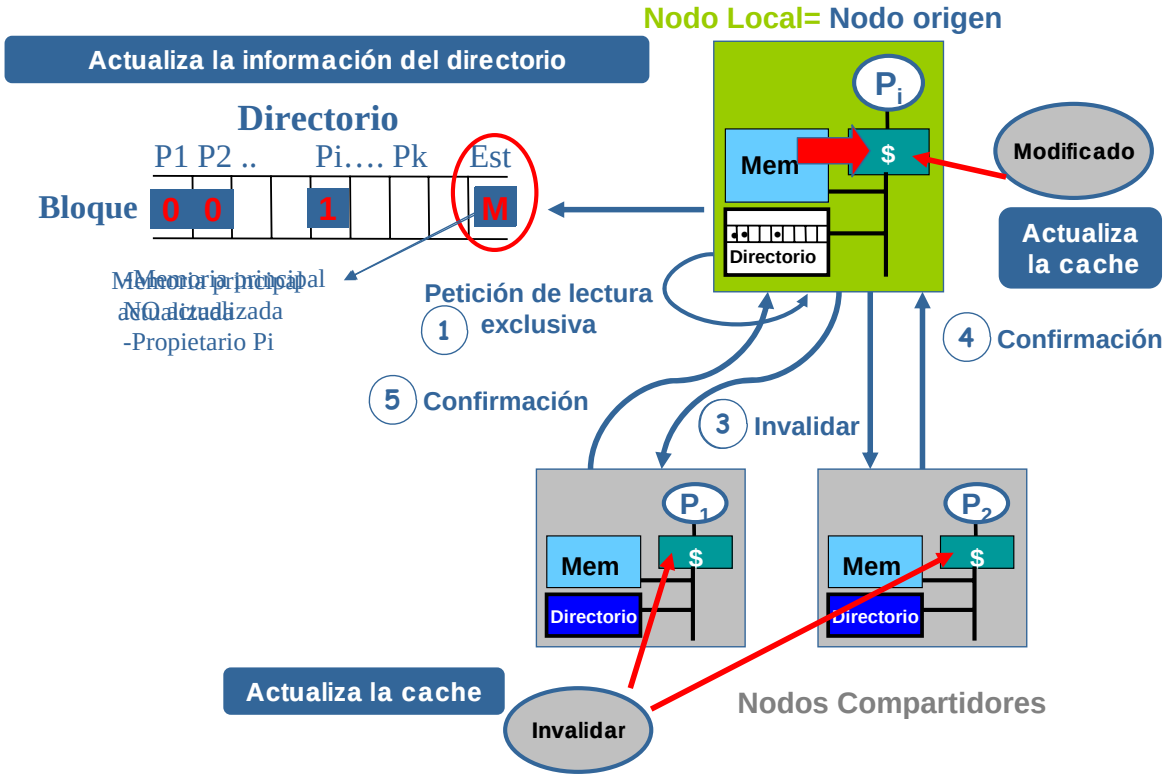
Directorios planos basados en memoria (12)

- Fallo de lectura Local sobre Bloque Modificado



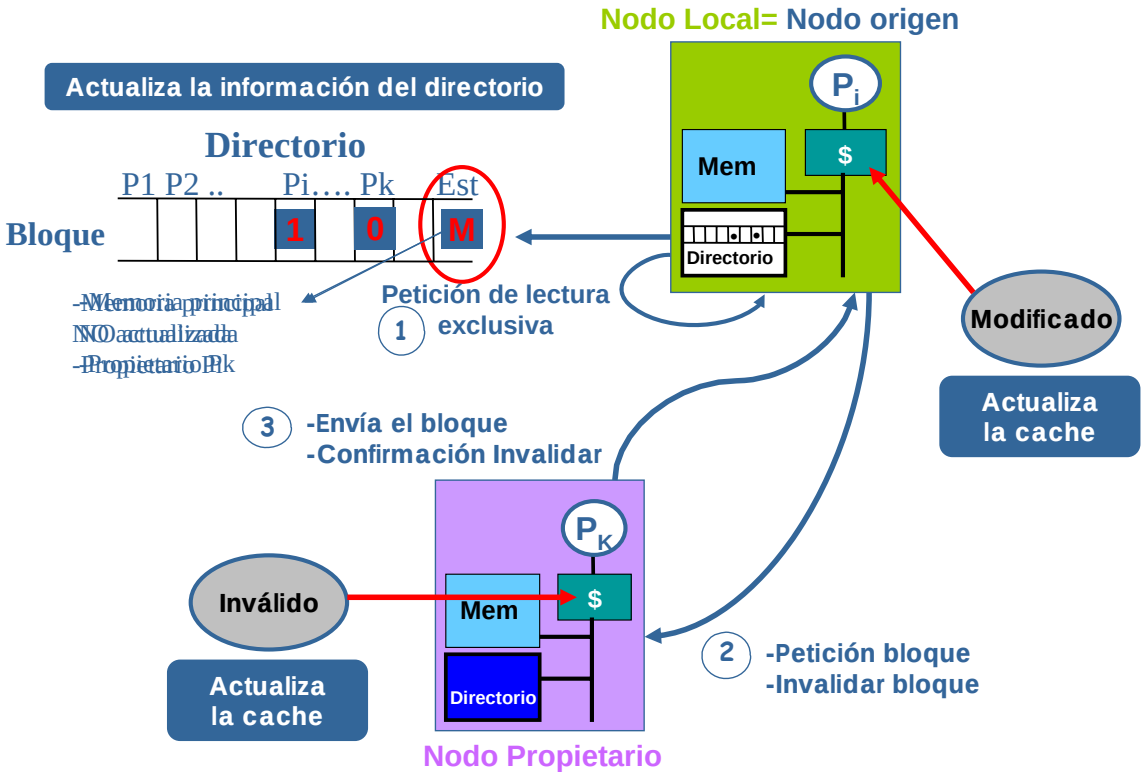
Directorios planos basados en memoria (13)

- Fallo de escritura Local sobre Bloque Compartido



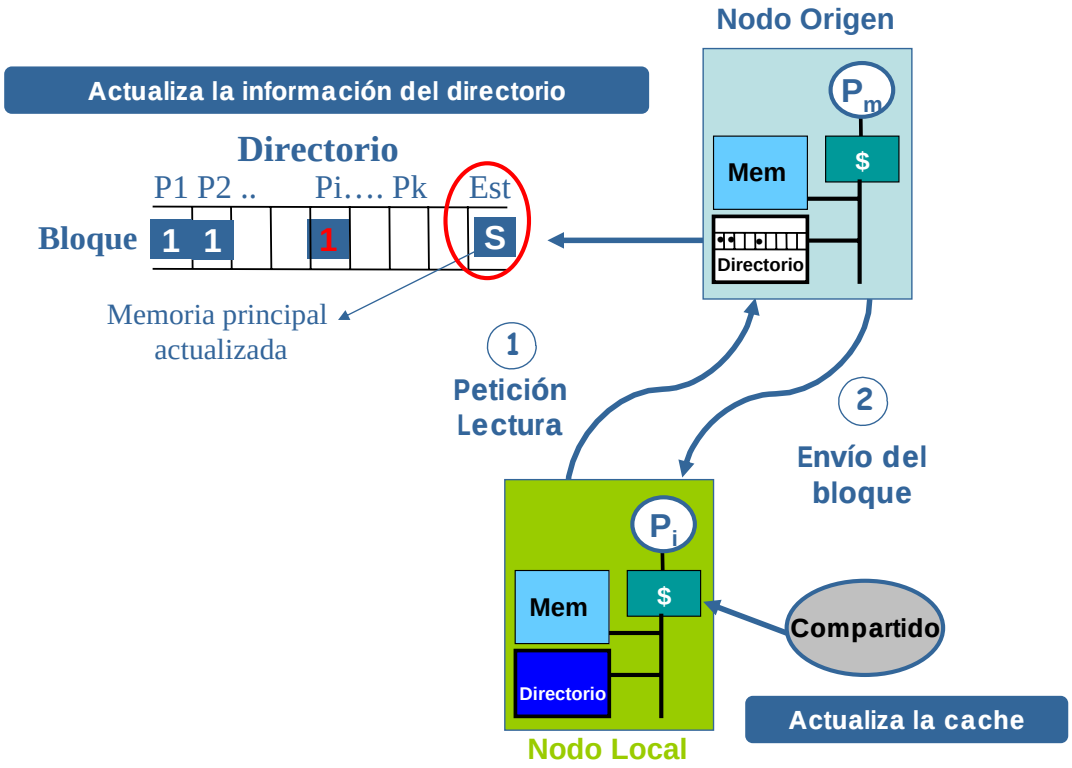
Directorios planos basados en memoria (14)

- Fallo de escritura Local sobre Bloque Modificado



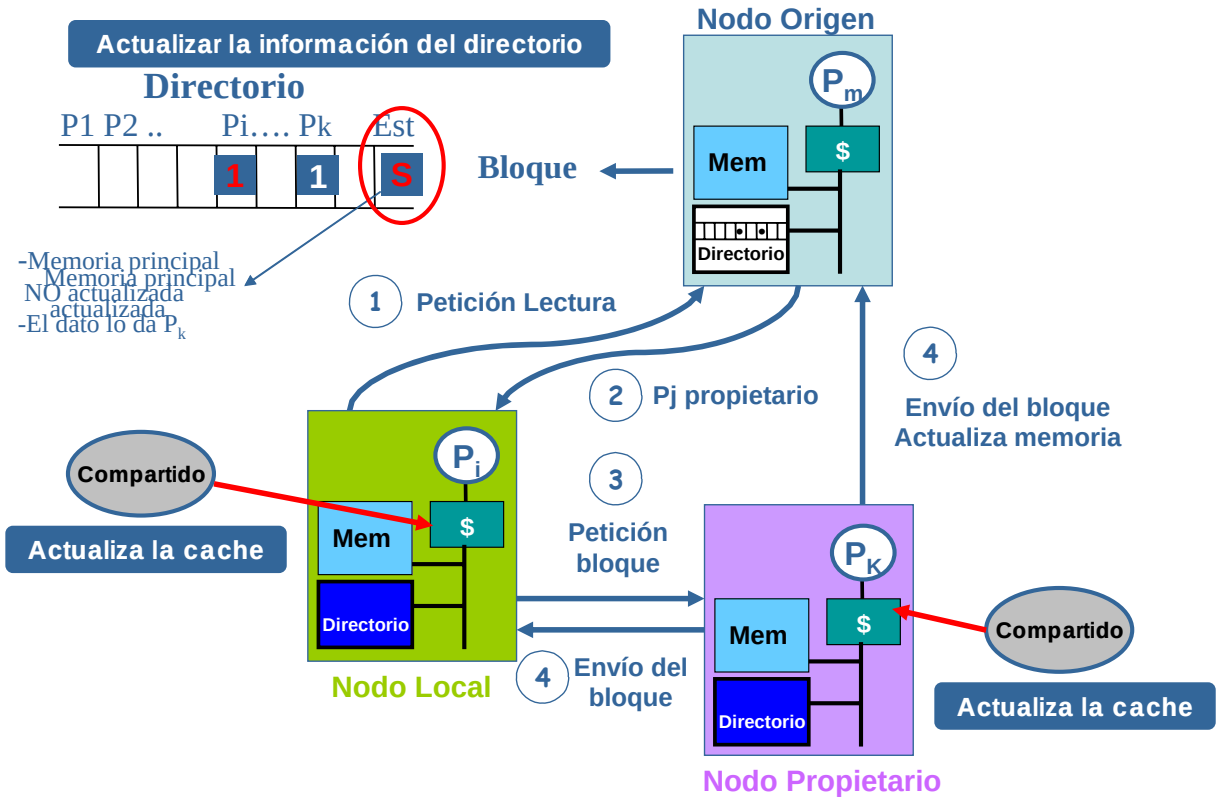
Directorios planos basados en memoria (15)

- Fallo de lectura Remoto sobre Bloque Compartido



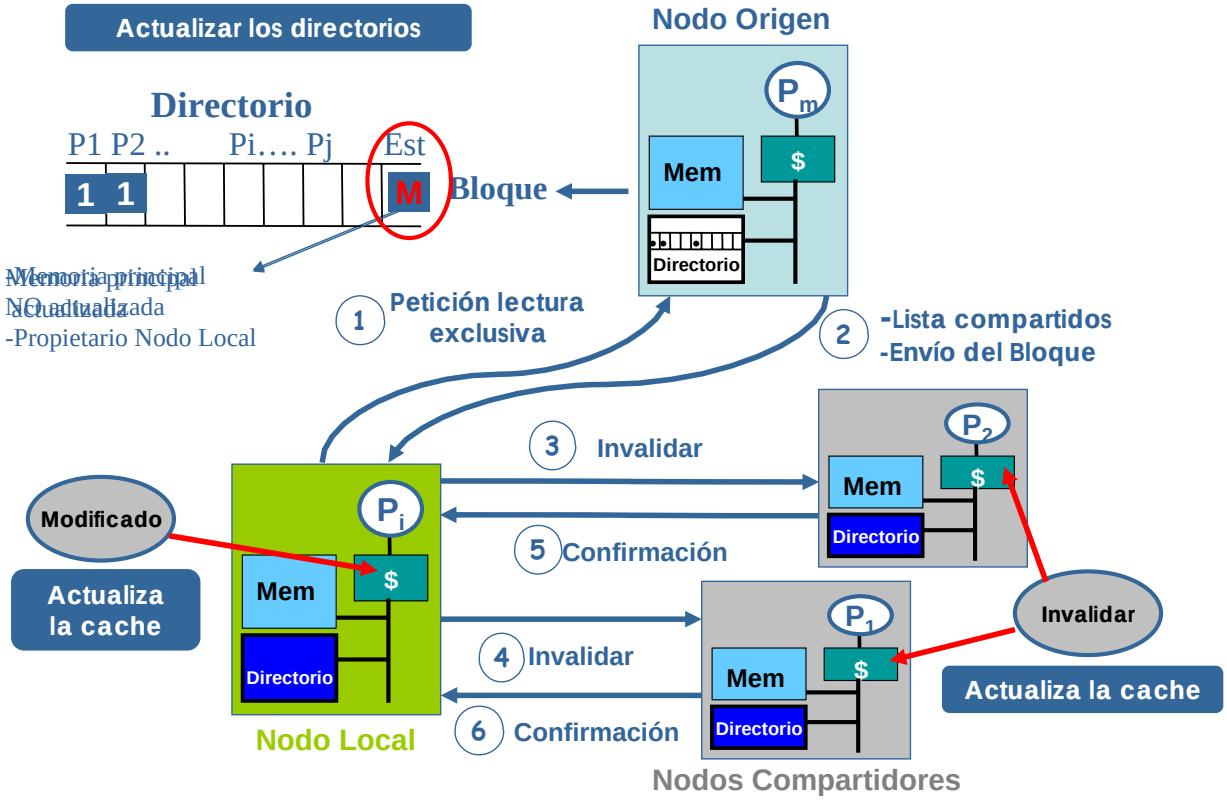
Directorios planos basados en memoria (16)

- Fallo de lectura Remota sobre Bloque Modificado



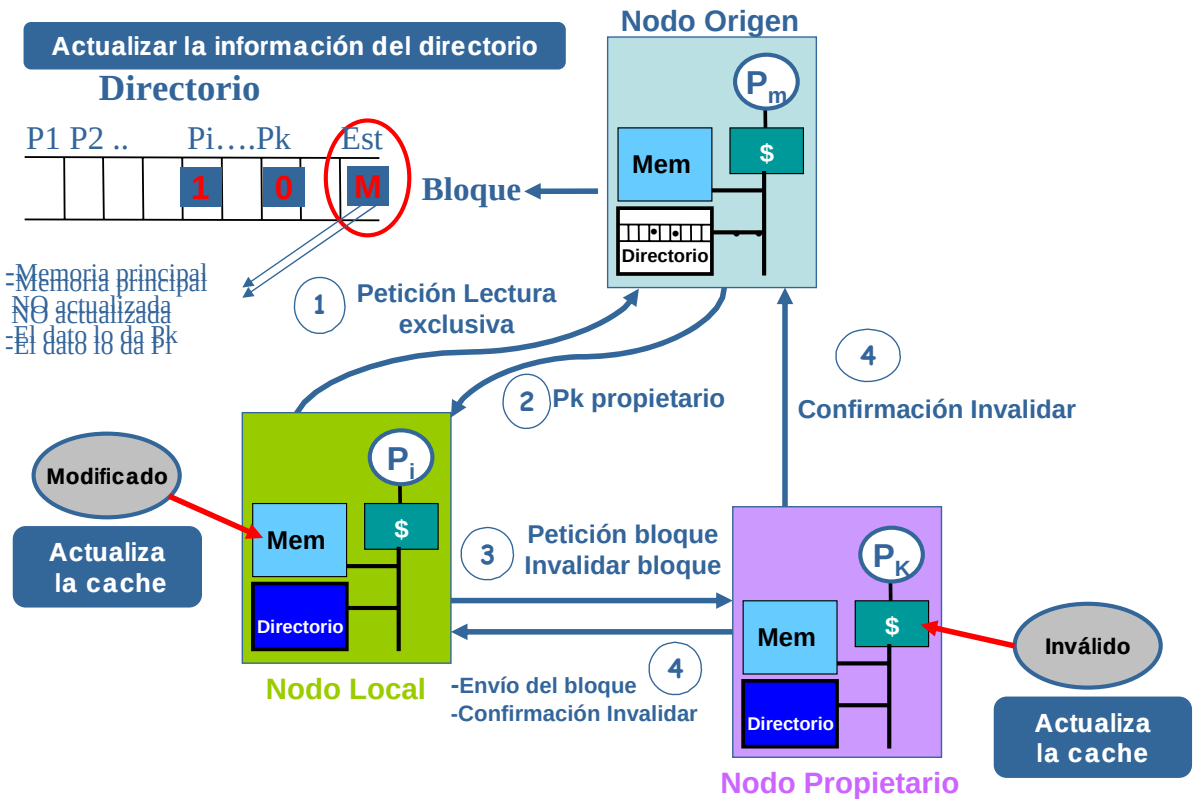
Directorios planos basados en memoria (17)

- Fallo/acierto de escritura sobre Bloque Compartido

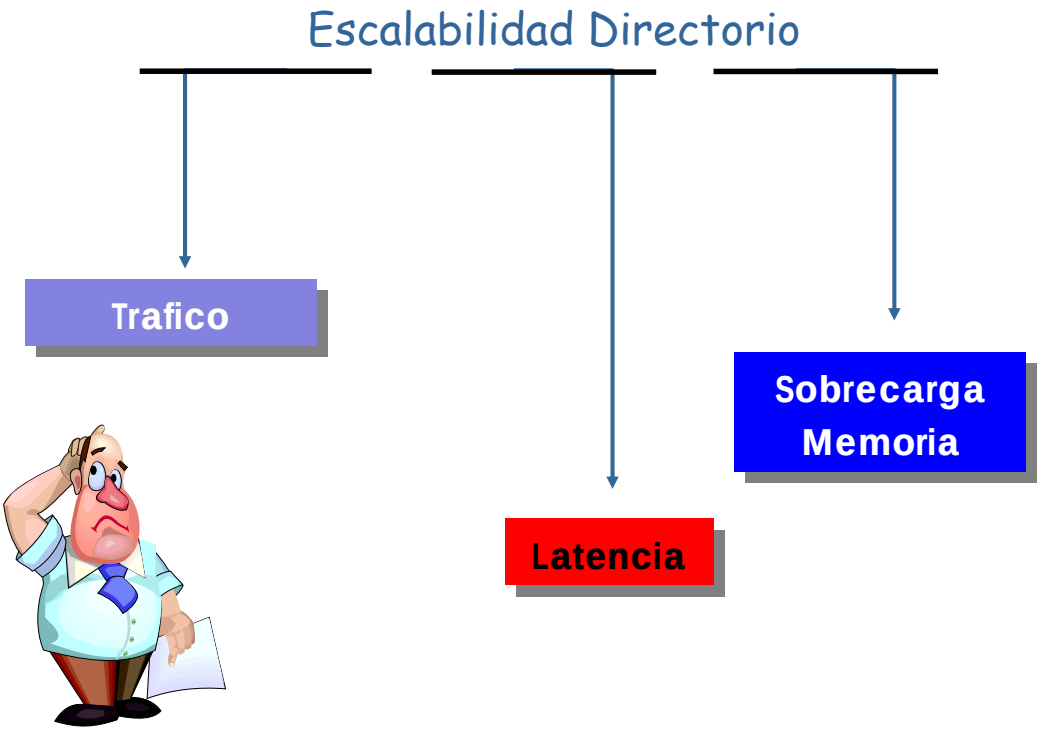


Directorios planos basados en memoria (18)

- Fallo de Escritura Remota sobre Bloque Modificado



Directorios planos basados en memoria (19)



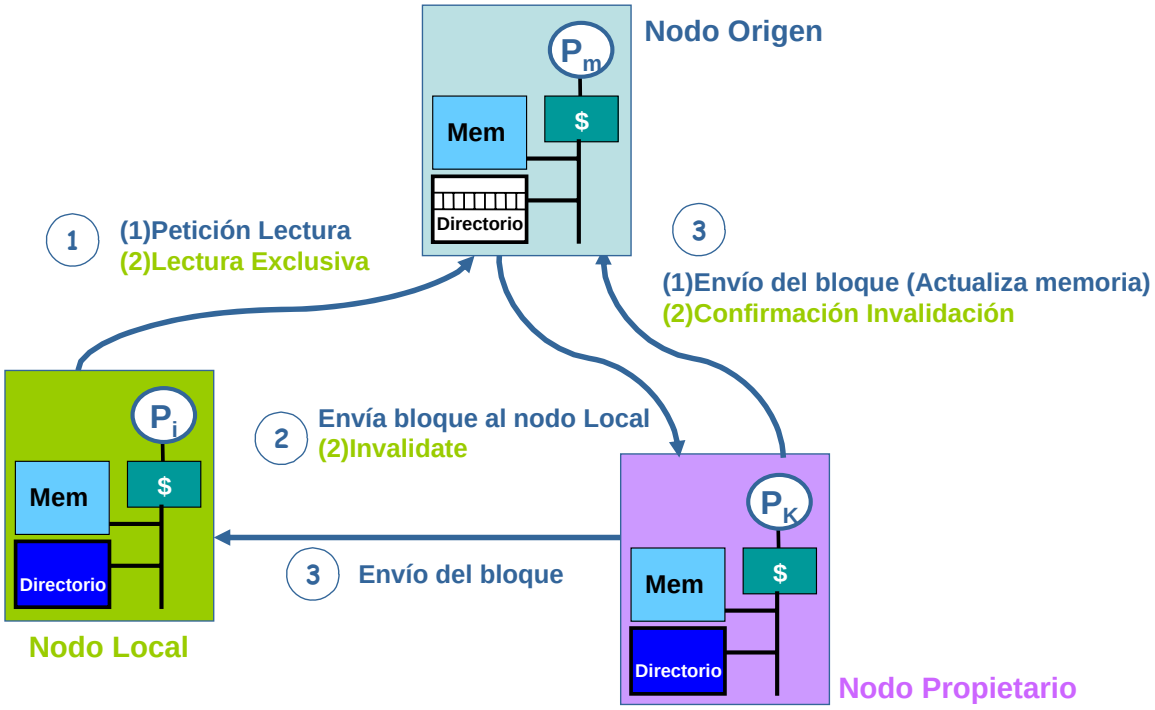
Directorios planos basados en memoria (20)

- **Rendimiento en las Escrituras (mensajes de invalidación)**
  - Trafico (BW): Nº de Mensajes
    - Proporcional nodos que comparten el bloque
  - Latencia: Nº de Mensajes en camino crítico:
    - Todos disponibles en nodo origen
    - Envío en paralelo ← VENTAJA

- **Sobrecarga de almacenamiento**
  - Tamaño del directorio en un nodo:
    - **(Nº de nodos + bit de estado) * Nº de bloques de memoria**
  - Ejm: Si tamaño de bloque: 64 bytes, con:
    - 64 nodos: sobrecarga del directorio 12.5%
    - 256 nodos: sobrecarga del directorio 50%
    - 1024 nodos: sobrecarga del directorio 200%

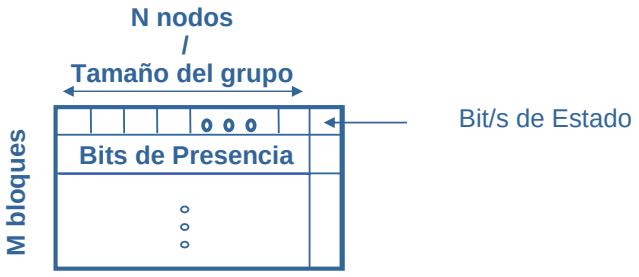
Directorios planos basados en memoria (16)

- **Reducir tráfico y latencia: forwarding**
  - Cuando se solicita un bloque en estado Modificado El nodo origen se comunica directamente con el nodo propietario



Directorios planos basados en memoria (21)

- **Reducir Sobrecarga de almacenamiento**
  - **Alternativa: Vector de bit de presencia asignado a grupos**
    - Cada bit de presencia representa un grupo de nodos



  - **Sobrecarga de almacenamiento**
    - Tamaño del directorio de un nodo=
      (N /tamaño del grupo + bit de estado) * Bloques de memoria
  - **Rendimiento en las Escrituras**
    - Se tienen que enviar invalidaciones a todos los nodos asignados a un bit de presencia, tengan o no copia del bloque

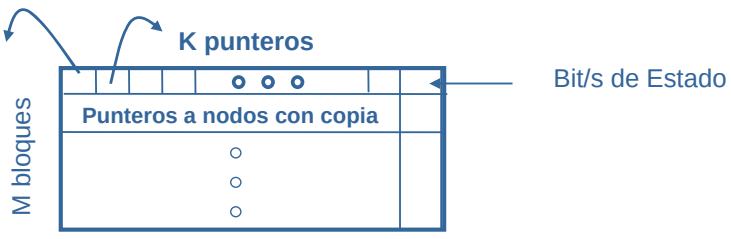
Otras Alternativas

Directorios Planos **Basados en Caches**

- La información de directorio acerca de un determinado bloque no está totalmente almacenada en el nodo origen sino distribuida entre las propias copias

Directorios planos basados en caches (1)

- **Directorio limitado**
  - En lugar de vector de bits de presencia → Punteros a caches (nodos) con copia del bloque
    - Nodos con copia (k) < Nodos totales (N)
    - Cada puntero necesita $\log_2 N$ bits para codificar el nodo



  - **Sobrecarga de almacenamiento**
    - Tamaño del directorio de un nodo=

      (K*log$_2$N + bits de estado) * Bloques de memoria
  - **Fallo de lectura de un bloque en una cache**
    - Si está llena la entrada del bloque en el directorio hay que invalidar una de las caches con copia para ubicar la nueva cache
    - Necesidad de implementar algoritmo de reemplazo

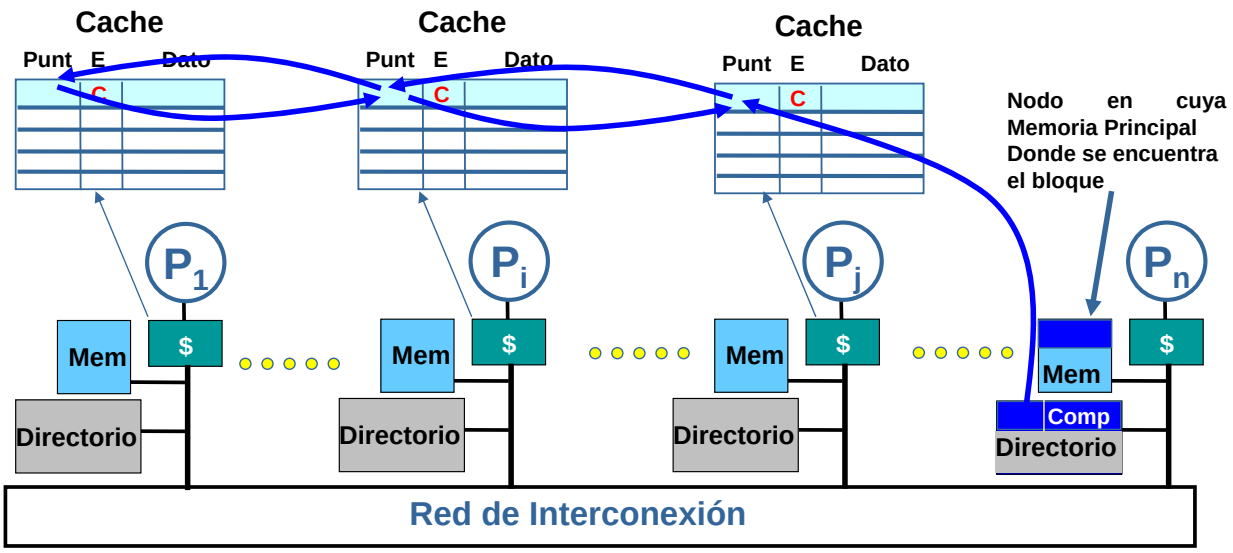
Directorios planos basados en caches (2)

- **Directorio encadenado**:
  - Las entradas del directorio quedan reducidas a Bits de estado y Puntero a la última cache que accedió al bloque
  - El resto de los nodos con copia se encuentra unidos mediante una lista enlazada o doblemente enlazada
    - La última cache tiene un puntero a la penúltima cache que accedió al bloque y así sucesivamente
    - Si un nuevo nodo solicita una copia compartida del bloque, se inserta como nueva cabecera
      - Implica menos transferencias que insertarlos al final de la lista
  - **Sobrecarga de almacenamiento**
    - Tamaño del directorio de un nodo=

($\log_2$N + bits de estado) * Bloques de memoria + $\log_2$N *Bloques de cache

Directorios planos basados en caches (3)

- **Directorio encadenado**:



Directorios planos basados en caches (4)

Algoritmo de funcionamiento del protocolo del directorio:

- Lo que **cambia** respecto a los directorios planos basados en memoria es **la actualización del directorio**:

- **$Nodo_i$ Solicita un bloque por fallo de lectura**
  - **Actualización del directorio** → envió de mensajes
    - **$Nodo_i$** envía petición al nodo origen de ese bloque para determinar la identidad del nodo cabecera
    - **Nodo origen** responde con la identidad del nodo cabecera
    - **$Nodo_i$** envía al nodo cabecera solicitud **inserción en la cabeza de la lista** (se convierte en el nuevo nodo cabecera)

- **$Nodo_i$ Solicita Reempalzamiento (aunque no se produzca post-escritura)** →
  - **Actualización del directorio** → Eliminación de la lista
    - Añade bastante complejidad al protocolo: es necesario llevar a cabo coordinación con nodo anterior y posterior en la lista

Directorios planos basados en caches (5)

Algoritmo de funcionamiento del protocolo del directorio:

- **$Nodo_i$ Solicita un bloque por acierto/fallo de escritura**
  - **Actualización del directorio** → envió de mensajes
    - **$Nodo_i$** envía petición al nodo origen de ese bloque para determinar la identidad del nodo cabecera
    - **Nodo origen** responde con la identidad del nodo cabecera
    - **$Nodo_i$** puede ser nuevo, o estar ya en la lista de nodos que comparten el bloque
    - Recorrer la lista para **invalidar las sucesivas copias.**
      - Sólo se queda él en la lista
      - Acuses de recibo de invalidaciones se envían al **$Nodo_i$**

Directorios planos basados en caches (6)

- **Rendimiento en las Escrituras (mensajes de invalidación)**
  - Trafico (BW): Nº de Mensajes
    - Proporcional al número de nodos que comparten el bloque (semejante a los basados en memoria), pero están distribuidos
  - Latencia: Nº de Mensajes en camino crítico:
    - También es proporcional al número de nodos que comparten el bloque (los mensajes se serializan). Esto es peor que en los de memoria !!

- **Sobrecarga de almacenamiento**
  - Menor que en los directorios basados en memoria. Mayor escalabilidad
  - Tamaño del directorio en un nodo:

(log$_2$N + bits de estado) * Bloques de memoria + log$_2$N *Bloques de cache

Directorios planos basados en caches (7)

- **Ventajas con esquemas basados en memoria**
  - Menor sobrecarga de directorio.
    - Número de Punteros a nodo cabecera:
      - Proporcional al número de bloque de memoria de la máquina
    - Número de Punteros siguiente y anterior:
      - Proporcional al número de bloques de cache en la máquina (mucho menor que el número de bloques de memoria)
    - Lista enlazada:
      - Punteros siguiente
    - Lista doblemente enlazada:
      - Punteros siguiente y anterior:
  - La lista enlazada guarda el orden realizado de accesos
  - El trabajo realizado por los controladores para enviar las invalidaciones no está centralizado en un nodo sino distribuido entre los nodos que comparten el bloque

Resumen

- Coherencia cache escalable
  - Protocolos basados en Directorio
  - Organización del Directorio
    - Directorios planos basados en memoria
    - Directorios planos basados en cache
  - Escalabilidad Directorio
    - Sobrecarga almacenamiento (basado en memoria)
    - Latencia
    - Tráfico (BW)