UNIVERSIDAD DE BURGOS

TEMA 1

Arquitecturas multiprocesador: MIMD de memoria compartida (multiprocesadores)

V 2.0

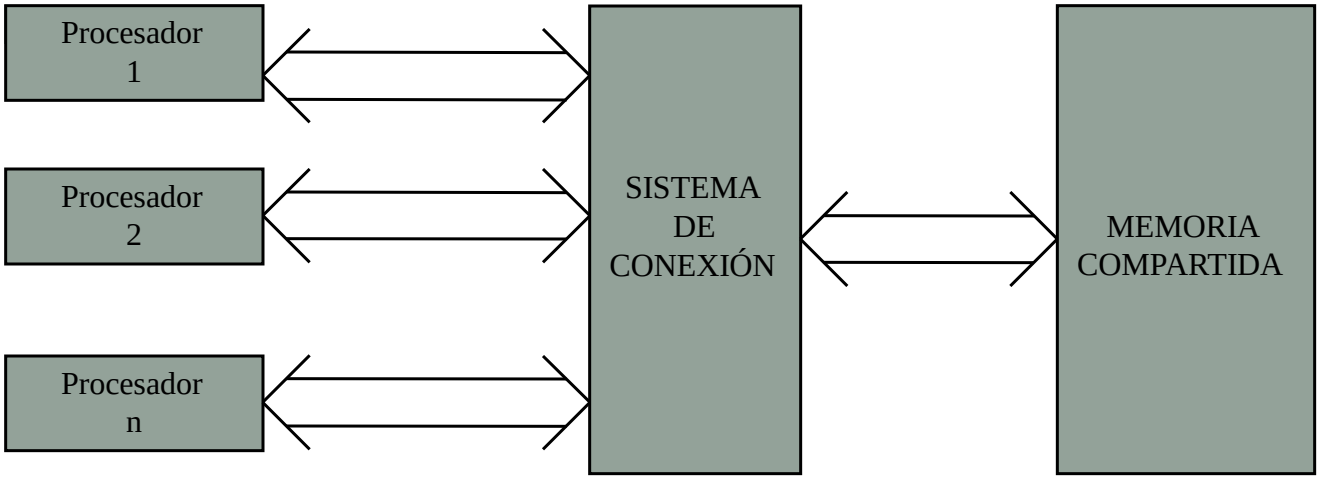
José M. Cámara (checam@ubu.es)

Multiprocesadores y multicomputadores

- Multiprocesadores: son sistemas formados por un cierto número de procesadores que trabajan en paralelo. La comunicación entre ellos se realiza mediante el empleo de variables en memoria compartida.
- Multicomputadores:en este caso los procesadores que trabajan en paralelo disponen de una memoria totalmente independiente del resto. Se caracterizan por tanto por emplear un sistema de memoria distribuida. La intercomunicación se realiza mediante paso de mensajes entre ellos.

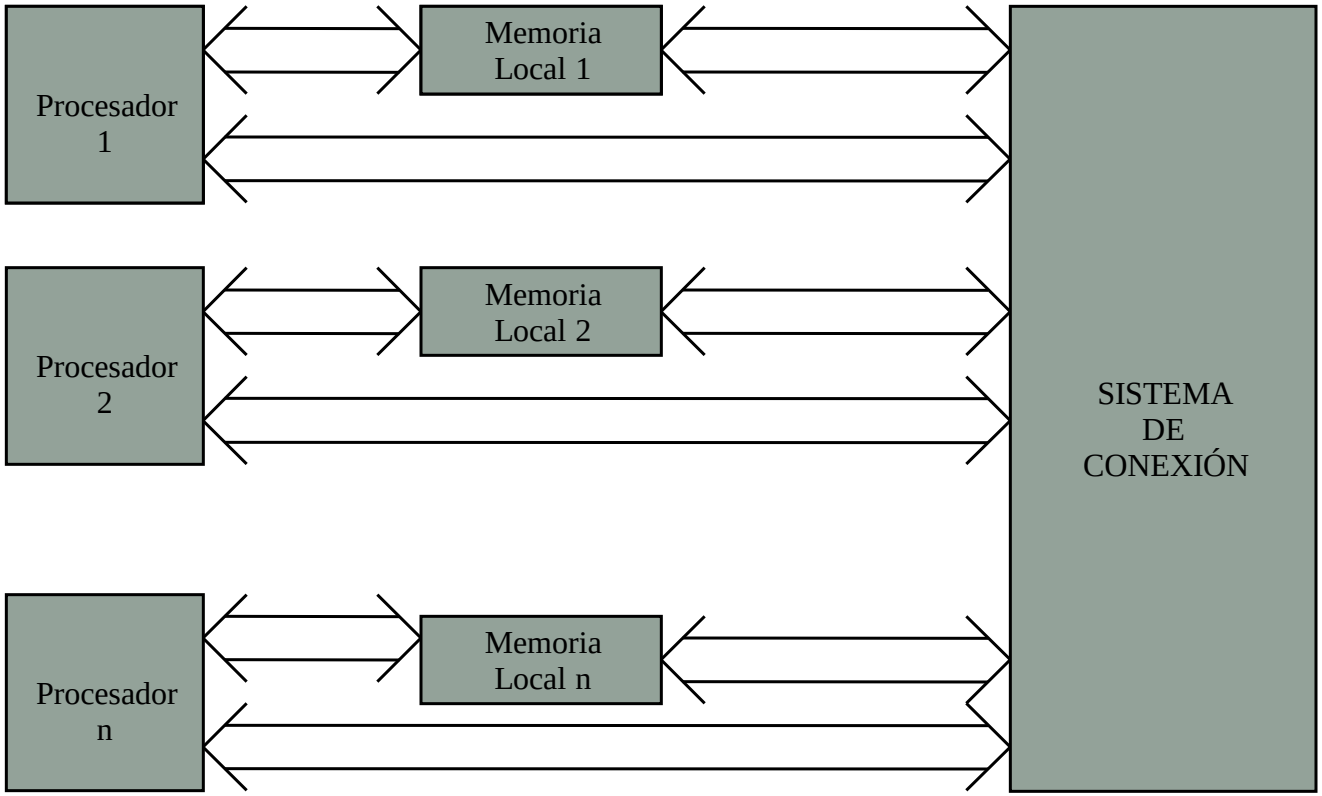
Memoria compartida

- UMA



Memoria compartida

- NUMA



Memoria compartida

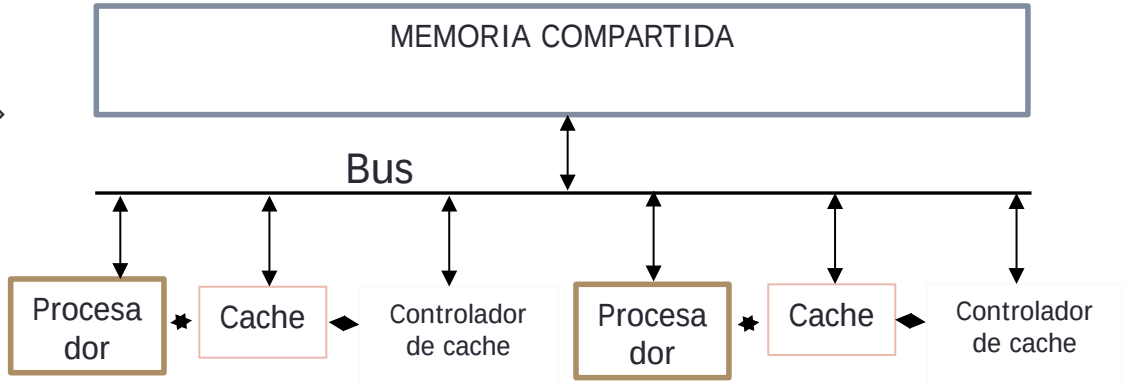
- Problemas de coherencia por:
  - Compartición de datos ⇒
  - Migración de procesos ⇒
  - Entrada-salida ⇒
- Se debe cumplir:
  - Propagación de escritura: las operaciones de escritura son vistas por todos los procesadores.
  - Serialización de escritura: las operaciones de escritura son vistas por todos en el mismo orden.

Coherencia de cache

- Protocolos de observación del bus (snoopy):
  - De escritura invalidación
  - De escritura-actualización
- Protocolos basados en directorios:
  - De mapeo completo
  - Limitados
  - Encadenados

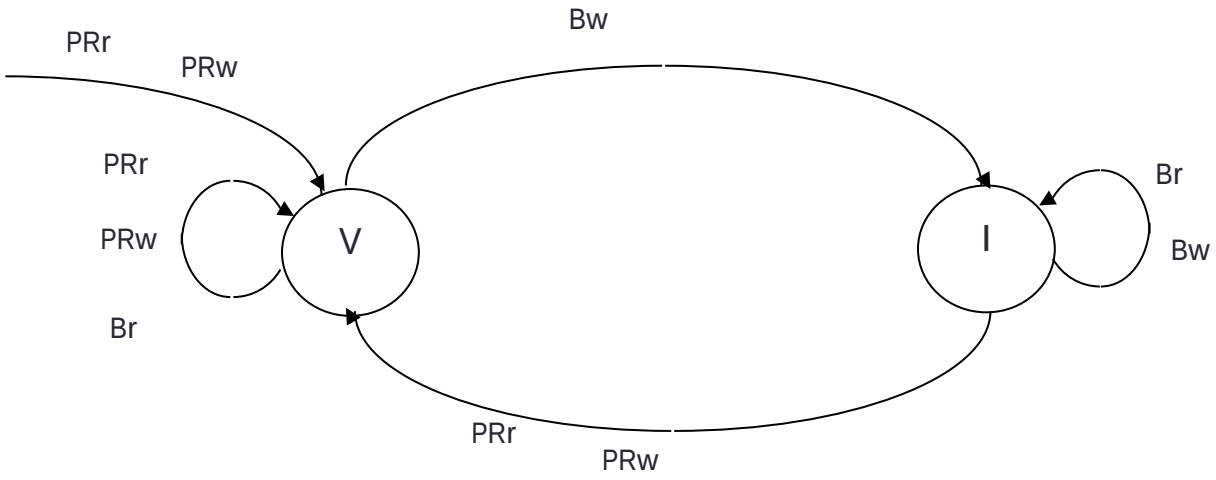
Observación del bus:

- Invalidación:
  - Write-trough
  - Write-back
    - MSI
    - MESI
    - MOESI*
    - MESIF*
- Actualización:
  - Firefly
  - Dragon

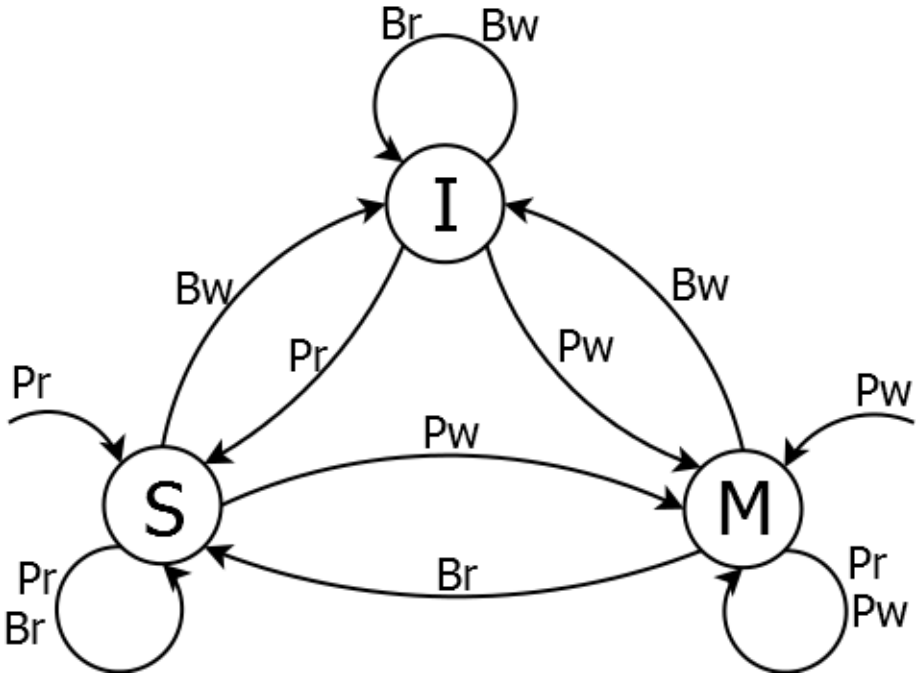


* No implican la existencia de un bus, sino conexiones punto a punto

Write through

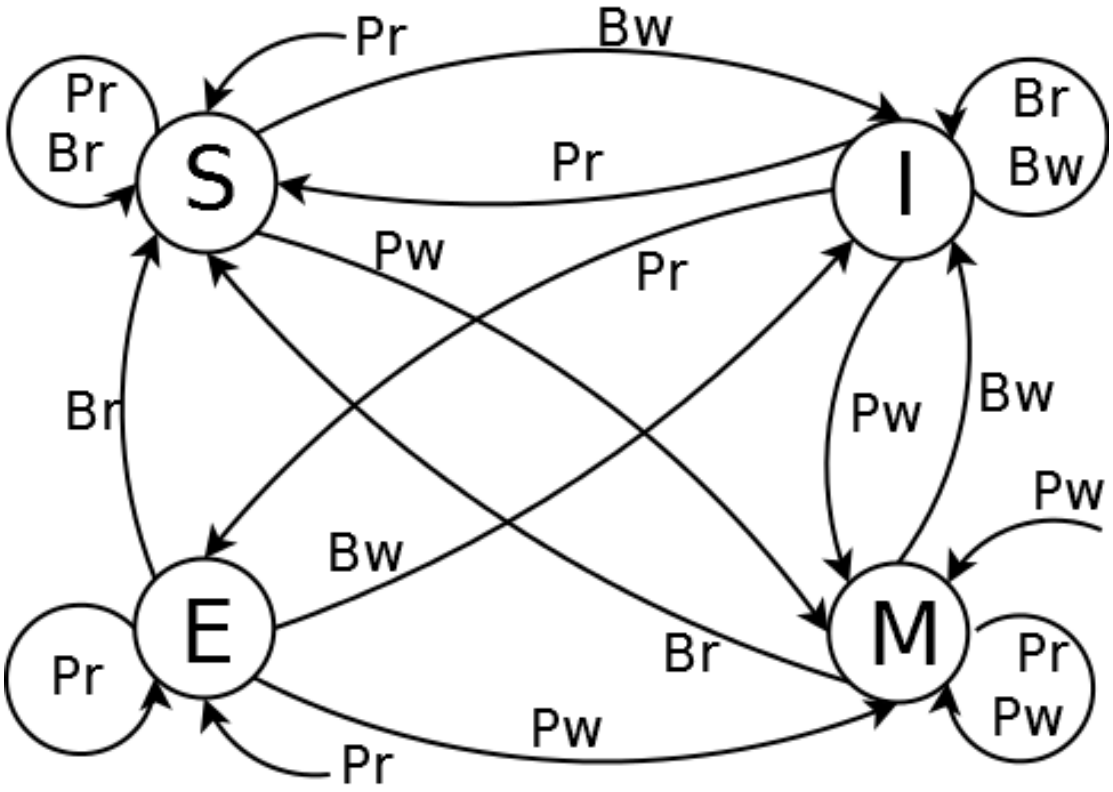


Protocolo MSI



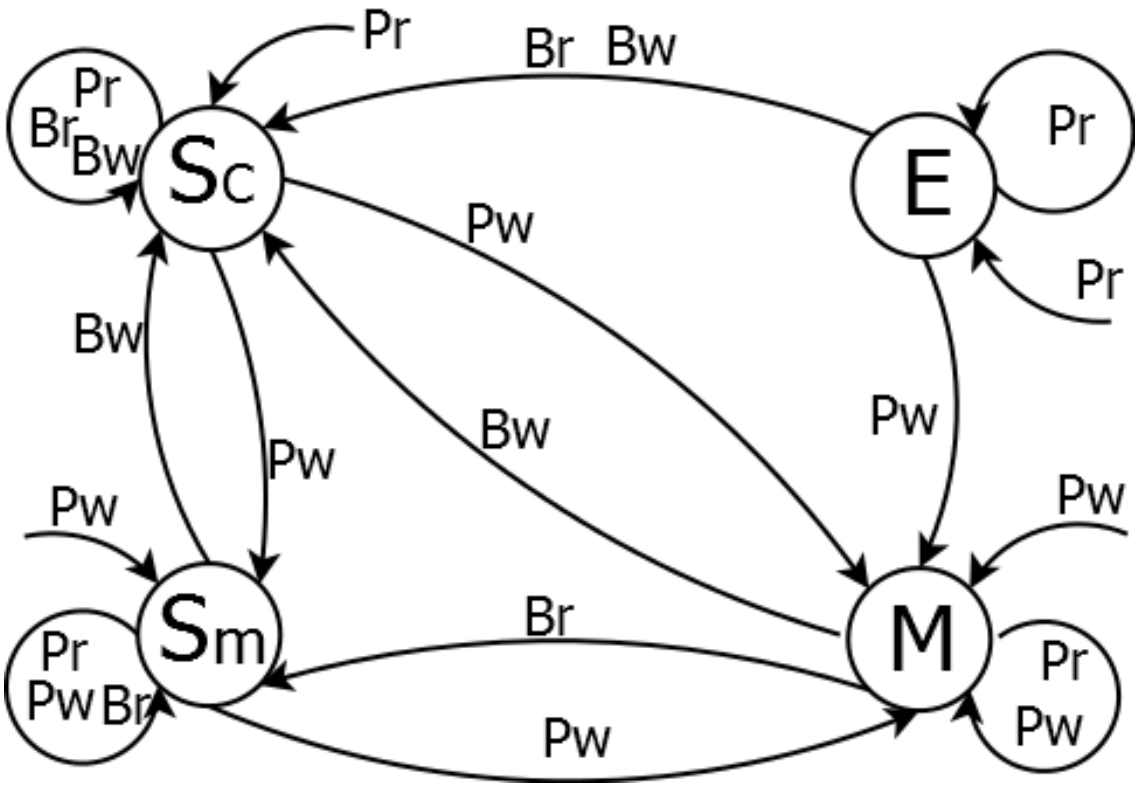
Contenido multimedia disponible

Protocolo MESI



Contenido multimedia disponible

Protocolo Dragon



Contenido multimedia disponible

Directorios

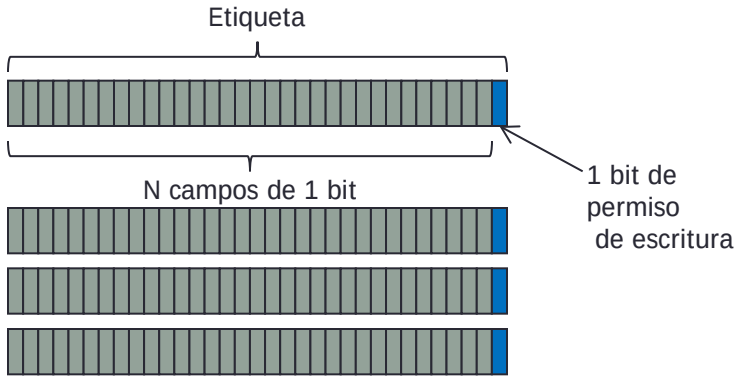
- Almacenados en memoria principal.
- Permiten al controlador de memoria mantener constancia de las copias presentes en las cache locales.
- A cada línea de cache se le asocia una etiqueta.
- Existen 3 posibilidades de implementación:
  - Directorios de mapeo completo
  - Directorios limitados
  - Directorios encadenados

Directorios de mapeo completo

- La etiqueta incluye un campo de un bit por cada posible poseedor de copia (todos los nodos de la máquina).
- Un campo adicional de un bit indica si existe permiso de escritura. Si es así, solamente se encontrará activado el bit correspondiente al nodo que lo ha solicitado.
- Antes de conceder el permiso de escritura, el controlador de memoria invalida las copias existentes hasta ese momento.
- Los directorios de mapeo completo tienen un problema de escalabilidad:
  - El tamaño de las etiquetas crece proporcionalmente al número de nodos de la máquina > mucho espacio de memoria principal dedicado al directorio.

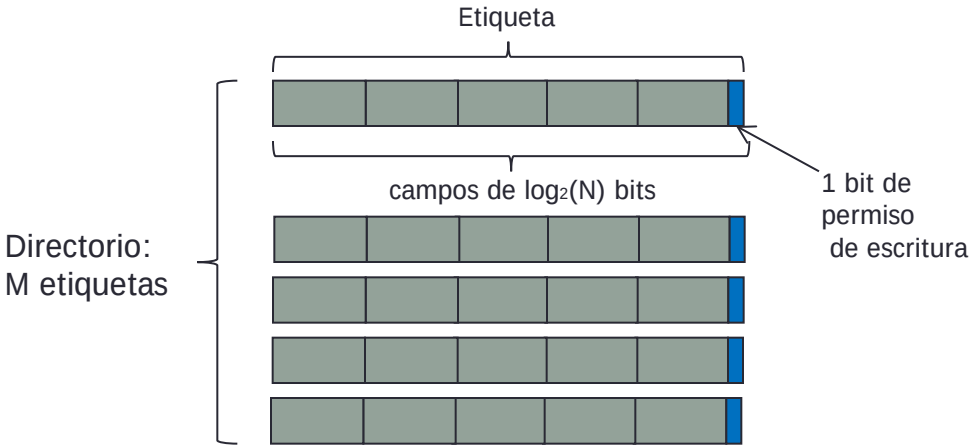
- N procesadores en el sistema
- M líneas de cache en memoria principal

Directorio:
M etiquetas



Directorios limitados

- No existe un campo para cada nodo, sino para unos pocos.
- Los campos necesitan los bits suficientes para codificar a qué nodo representan: $\log_2$ (nº de nodos).
- También existe un campo para señalar si se ha concedido permiso de escritura.
- Si se reduce mucho el nº de campos es posible minimizar el tamaño del directorio.
- En contrapartida, se limita el número de posibles poseedores de copia. Si todos los campos están utilizados y aparece un nuevo peticionario, hay que retirar alguna copia. Esto provoca intercambios adicionales y la implementación de una política de intercambio que permita decidir qué copia se retira.



- N procesadores en el sistema
- M líneas de cache en memoria principal

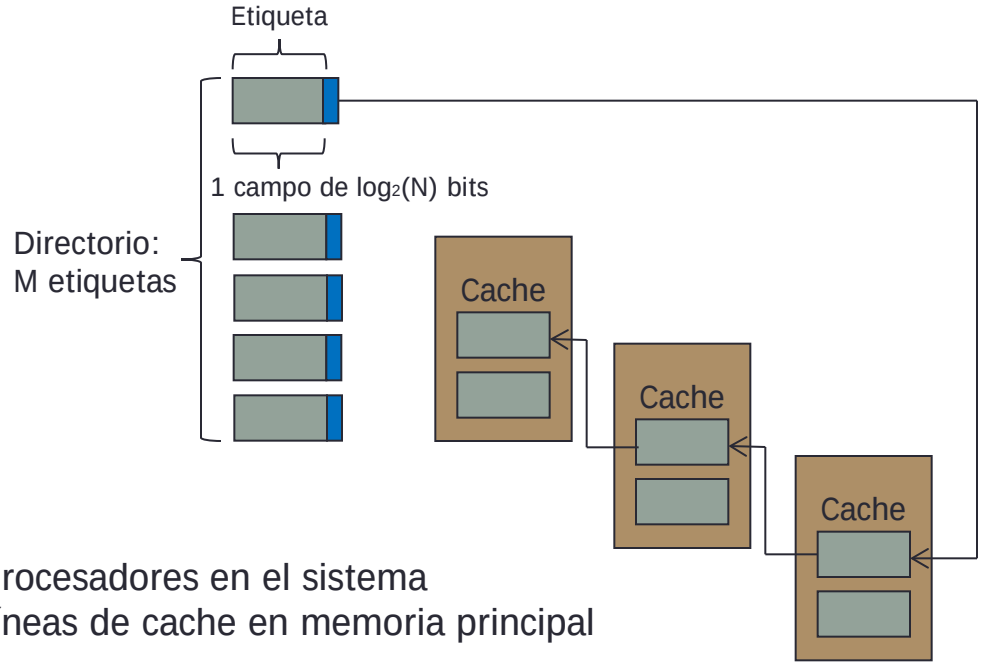
Ejemplo directorios

Máquina con 256 nodos y 2 Gbytes de memoria principal por nodo. Líneas de cache de 64 bytes.

- Directorio de mapeo completo:
  - 512GB/64bytes = $2^{39}/2^{6}=2^{33}$ líneas de cache = etiquetas
  - Cada etiqueta 256 + 1 bits ≈ 2^{8} bits = 2^{5} bytes
  - 2^{33} etiquetas * 2^{5} bytes/etiqueta = 2^{38} bytes para directorio
- Directorio limitado (con 4 posibles poseedores de copia simultáneos):
  - 512GB/64bytes = $2^{39}/2^{6}=2^{33}$ líneas de cache = etiquetas
  - Cada etiqueta ($\log_2(256)$)* 4 campos + 1 bits ≈ 32 bits = 4 bytes
  - 2^{33} etiquetas * 2^{2} bytes/etiqueta = 2^{35} bytes para directorio

Directorios encadenados

- Sólo se conserva un campo en la etiqueta para apuntar al último poseedor de copia.
- Cada poseedor de copia dispone de un puntero al anterior de la lista.
- Un nuevo peticionario se coloca al final de la lista y el controlador de memoria le entrega un puntero al que le precedía.
- Una eventual petición de escritura debe propagarse por toda la lista invalidando cada copia y avisando al anterior. Solo cuando el primero confirma su invalidación, se puede conceder el permiso de escritura.



- N procesadores en el sistema
- M líneas de cache en memoria principal

Se trata de un procedimiento lento. Se minimiza el almacenamiento en memoria principal pero se distribuye entre los diferentes nodos.

Ejemplo de bus: TLSB

- Implementado en máquinas de Alpha como el Alphaserver 8400.
- Segunda mitad de los 90s.
- Bus síncrono con buses de datos y direcciones separados.
- Bus de datos de 256 bits.
- BW máximo de 3,2 Gbytes/s: Bus de datos de 256 bits.

TLSB: Direccionamiento I

- Direccionamiento geográfico de los módulos conectados mediante 3 líneas. Se pueden conectar hasta 9 módulos ya que la dirección 000 es ocupada por el dispositivo 0 y el 8 que tiene que ser de e/s obligatoriamente.
- Direccionamiento virtual para dispositivos como bancos de memoria o CPUs que, de esta manera adquieren una dirección propia en el sistema. Cada módulo puede albergar hasta 8 direcciones virtuales.

TLSB: Direccionamiento II

- Hasta 1 TB de memoria es direccionado mediante direcciones de 40 bits.
- La dirección es decodificada por el solicitante para extraer la dirección virtual del banco de memoria.
- Hasta 16 bancos de memoria son soportados por el sistema.
- Las transferencias emplean líneas de cache de 64 bytes.
- El solicitante puede lanzar una petición de bus al mismo tiempo que compara la etiqueta de cache local. Si se produce un “hit”, la petición se anula.

TLSB: Direccionamiento III

- De los 40 bits de direcciones, se toman los bits 6-39 como dirección de línea de 64 bytes. 2^{34} líneas de cache x 2^{34} bytes /Línea = 2^{40} bytes direccionables.
- El bit 5 se emplea para definir la secuencia en que los dos bloques de 32 bytes aparecen en el bus: High>low o Low>high.
- Los 5 bits restantes codifican la dirección virtual (hasta 16 CPUs y hasta 16 bancos de memoria).

TLSB: Arbitraje

- Cualquier transacción en el bus debe iniciarse con una petición por parte del módulo interesado.
- El arbitraje es distribuido, lo cual implica que no existe un dispositivo que actúe de arbitro. La contienda tiene que ser resuelta entre los distintos módulos.
- La prioridad de los nodos se inicializa a su dirección geográfica.
- En tiempo de funcionamiento, se emplea un mecanismo de Round Robin para actualizar la prioridad.

TLSB: Transferencias I

- Cuando un nodo esclavo está listo para proporcionar los datos que le han sido requeridos, toma control del bus.
- El esclavo activa la línea denominada TLSB_SEND_DATA, a lo que todos los dispositivos tienen tiempo para activar las líneas TLSB_SHARED o TLSB_DIRTY.
- TLSB_SHARED activada indica que algún dispositivo dispone de copia válida y desea seguir disponiendo de ella. Esta línea puede ser activada por cualquier dispositivo en respuesta a cualquier comando.

TLSB: Transferencias II

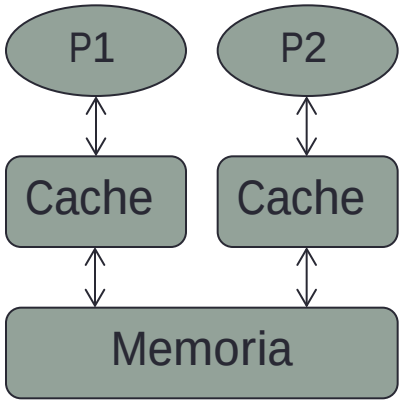
- TLSB_DIRTY activa indica que hay un dispositivo cuya copia es más reciente que la que hay en memoria. En tal caso, él se encargará de volcar los datos al bus.
- TLSB_DIRTY sólo puede ser activada en respuesta a comandos de lectura.
- Las especificaciones del protocolo no establecen el protocolo de coherencia, sino sólo el funcionamiento de las líneas.

Referencias

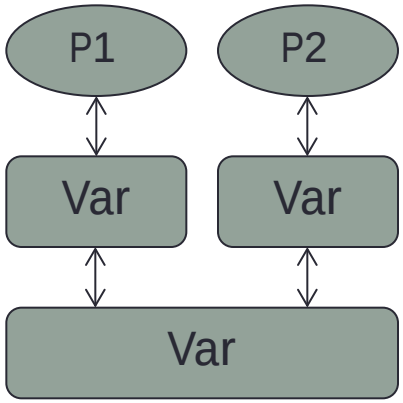
- K. whang. Advances Computer Architectures. McGraw Hill.
- Alphaserver8400 system hadbook.

Compartición de datos

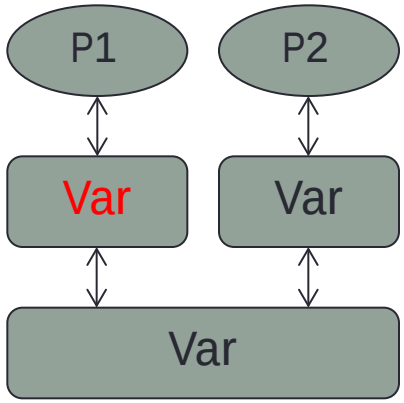
Escenario



1. P1 y P2 leen la misma variable (Var).

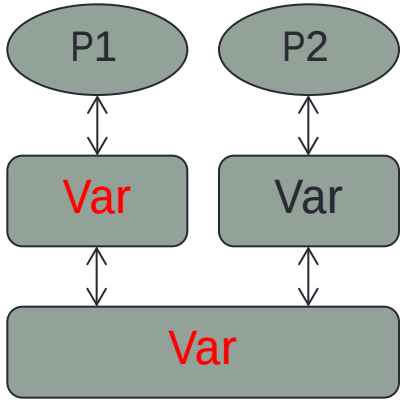


2. P1 modifica Var.

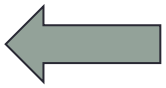


La cache de P2 queda obsoleta, en un sistema “write back” la memoria también.

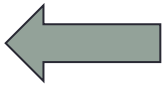
2. P1 modifica Var.



La cache de P2 queda obsoleta, en un sistema “write through” la memoria es actualizada.



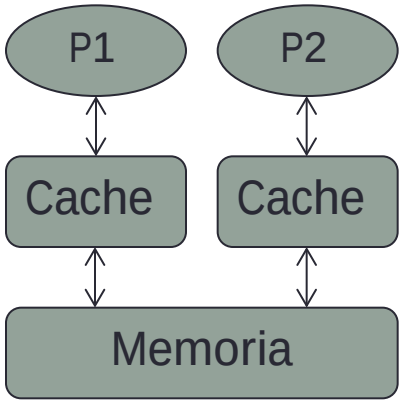
Volver a memoria compartida



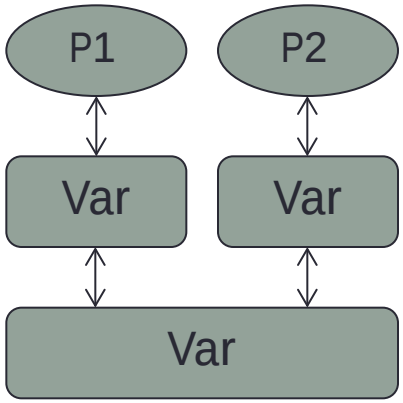
Volver a observación del bus

Migración de procesos

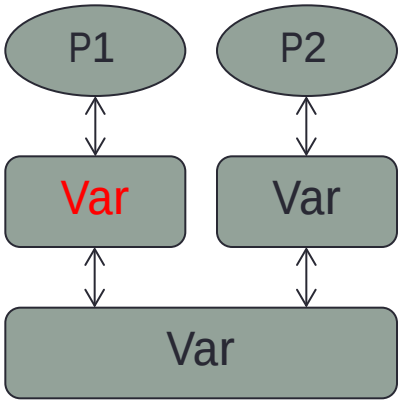
Escenario



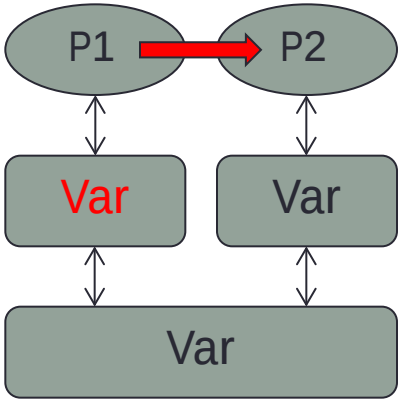
1. P1 y P2 leen la misma variable (Var).



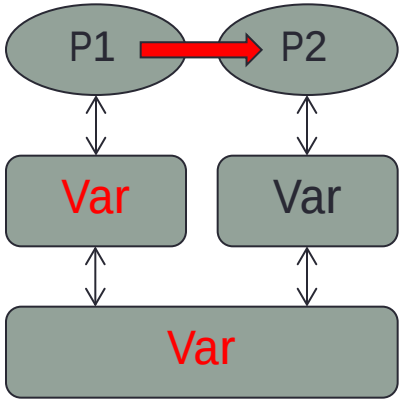
2. P1 modifica Var.



3. El proceso en P1 migra a P2 (WB).



3. El proceso en P1 migra a P2 (WT).



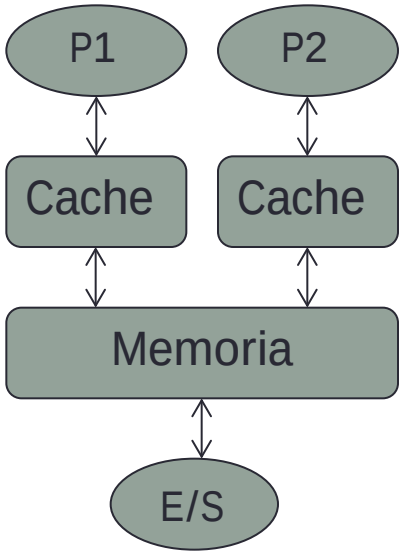
El proceso se encuentra en el nuevo procesador con un valor obsoleto de Var, tanto en “write back” como en “write through”.

Volver

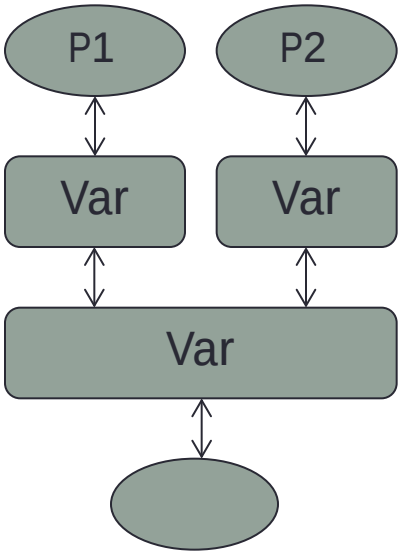
UNIVERSIDAD DE BURGOS

Entrada salida

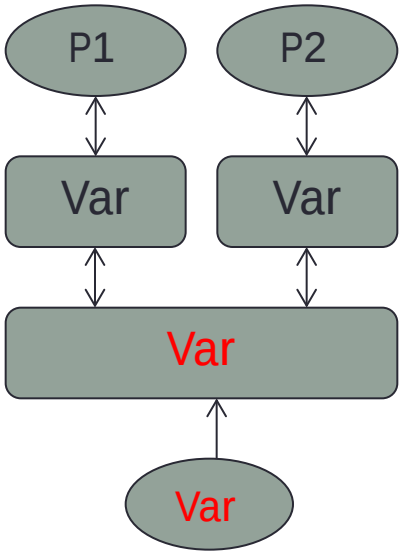
Escenario



1. P1 y P2 leen la misma variable (Var).

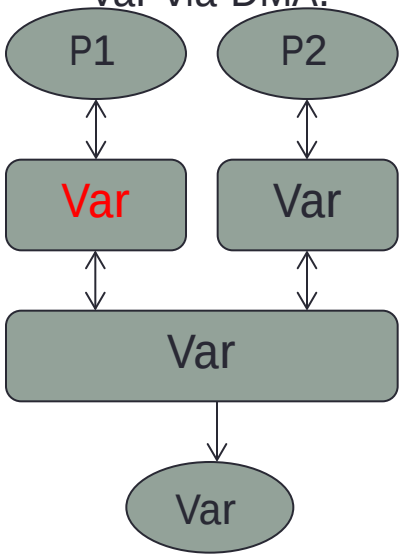


2. E/S introduce un cambio en memoria vía DMA.



Ambas caches quedan obsoletas

2'. P1 modifica Var, E/S lee Var vía DMA.



El valor devuelto por memoria a E/S es obsoleto. Sólo en "write back".

Volver