ARQUITECTURA DE COMPUTADORES II

AUTORES:
David Expósito Singh
Florin Isaila
Daniel Higuero Alonso-Mardones
Javier García Blas
Borja Bergua Guerra

Área de Arquitectura y Tecnología de Computadores
Departamento de Informática
Universidad Carlos III de Madrid

Julio de 2012

TEMA 3:
MP DE MEMORIA COMPARTIDA (III)

Índice (III)

5. Coherencia de caches.

- Uso de caches privadas.
- El problema de coherencia de caches (PCC).
- Posibles soluciones al PCC.
- Políticas para mantener la coherencia.
  - **Invalidación.**
  - **Actualización.**
  - Consideraciones de rendimiento.
  - Decisiones de diseño.
  - Variaciones.
- Implementaciones:
  - Hardware: *Snoopy* y directorios.
  - Software.

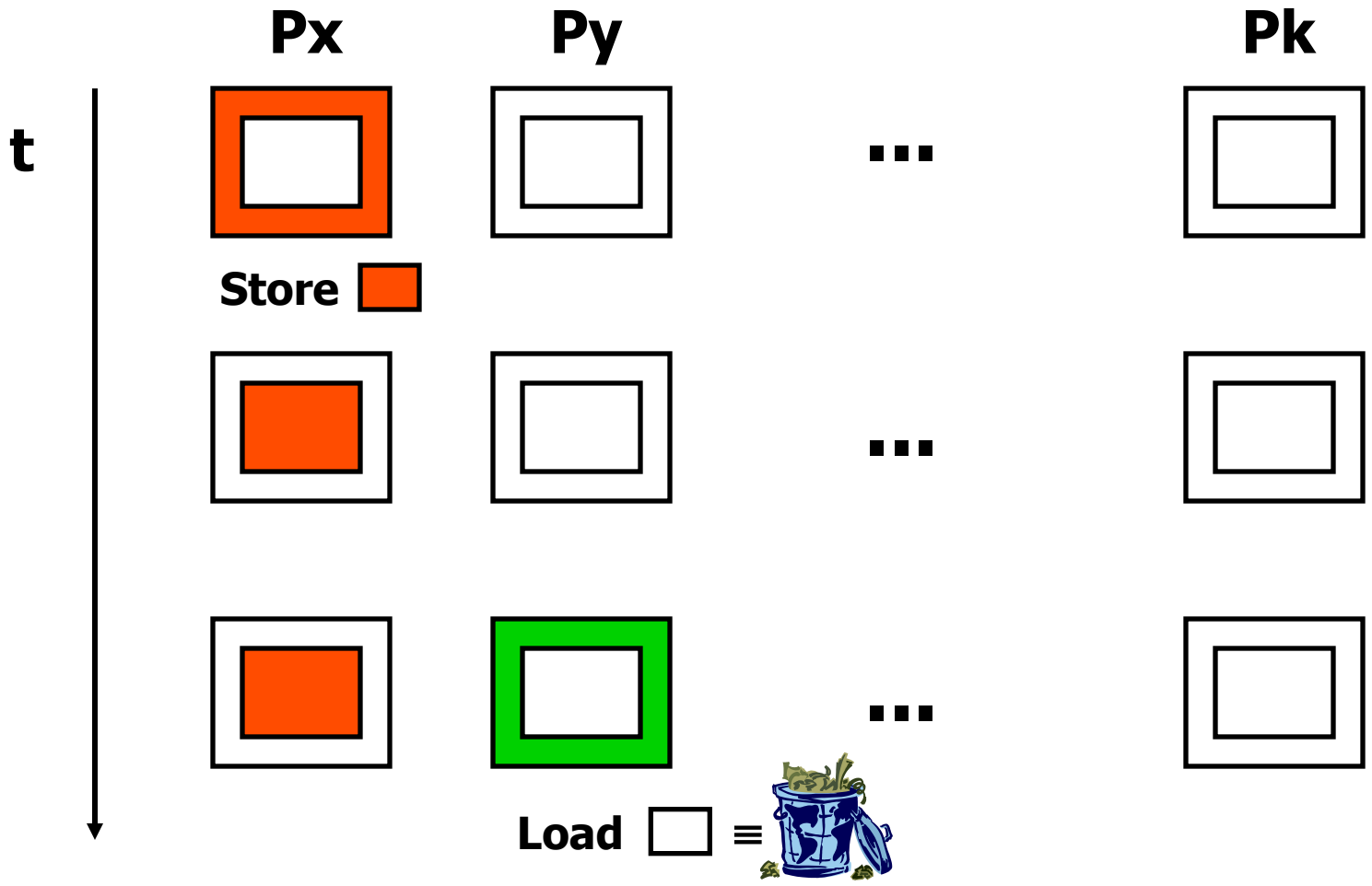
Coherencia de caches (C. CC.)

- **Justificación** del uso de caches privadas:
  - Reducen el tiempo medio de acceso (uso "monoprocesador").
  - Reducen la demanda de ancho de banda en la red de interconexión
- **Problema**:

  Existencia de datos compartidos modificables *(shared writable data)*
  - Copias de una misma variable en diferentes caches.
  - La escritura de un procesador en su copia no se hace visible a los demás que también tienen copia: accederán a valores no actualizados (*stale data*):

  Problema de Coherencia de Caches (PCC)
  - Se necesita realizar alguna acción para solucionar este problema.

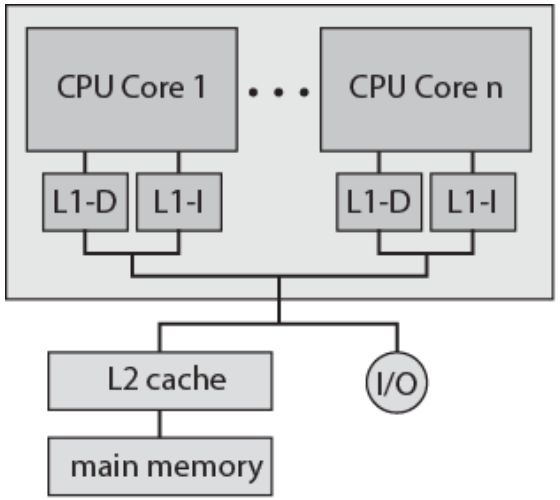
C. CC.: ej. problema



C. CC.: posibles soluciones

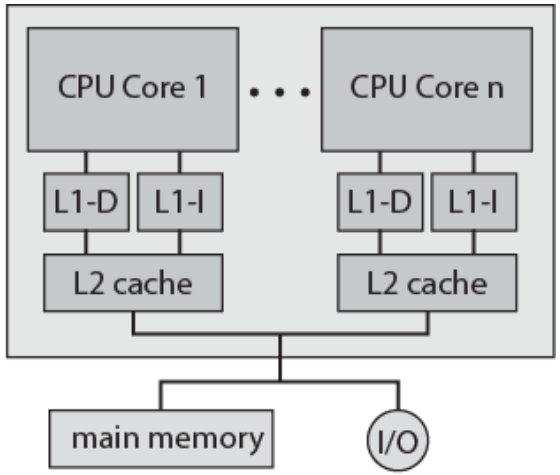
- Posibles soluciones al PCC:
  1. Uso de **caches compartidas**:
     - \+ : no hay problema de coherencia.
     - \+ : reduce el tiempo de acceso.
     - \+ : reduce la latencia en los datos compartidos.
     - – : necesario un mayor ancho de banda que con caches privadas de menor capacidad.
     - – : necesario un mayor tiempo de acceso.
     - Uso real:
       - Mediados de los 80s en parejas de procesadores: *Sequent* y *Encore*.
       - En la actualidad: propuesta para MP "*on-a-chip*"

ARM11 MPCore



(a) Dedicated L1 cache

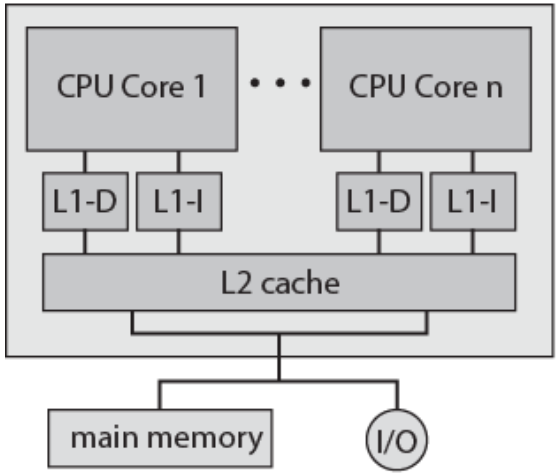
AMD Opteron



(b) Dedicated L2 cache

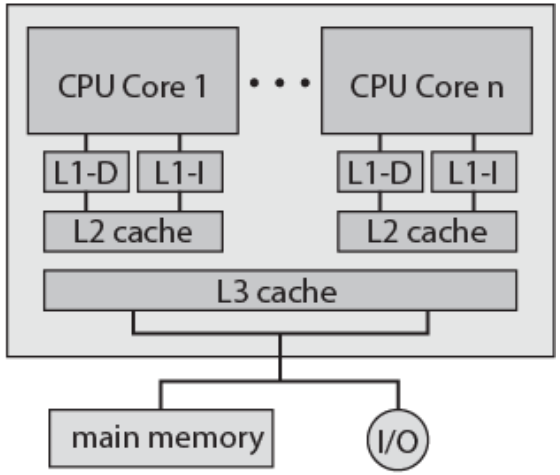
No compartida

Intel Core Duo



(c) Shared L2 cache

Intel Core i7



(d) Shared L3 cache

Compartida

C. CC.: posibles soluciones

2. Distinguir entre información "**cacheable**" y "**no cacheable**": dependiendo de la "semántica" de las direcciones.
   1. Implementación difícil, por ejemplo mediante software.
   2. Ej. "cacheable": instrucciones, datos privados.
   - Ej. "no cacheable": datos compartidos, p.ej., cerrojos, datos en regiones críticas (RR.CC.): mejora inmediata: "cachear" datos en RR.CC. y acción "pro-coherencia" (Invalidar o Actualizar) al salir:

CC: posibles soluciones

3. Copiar en la cache cualquier bloque: **todo "cacheable"**
   - Funcionamiento "monoprocesador": se lleva a la cache cualquier bloque que se necesite: independiente de su "semántica".
   - **Coherencia** mantenida por *hardware*: circuitería añadida que garantiza que no se acceda a una copia no actualizada de un bloque.
   - **Políticas**:
     - Invalidación (I)
     - Actualización (A)
   - Implementaciones:
     - MP basados en buses: *snoopy*
     - MP 'escalables': directorios

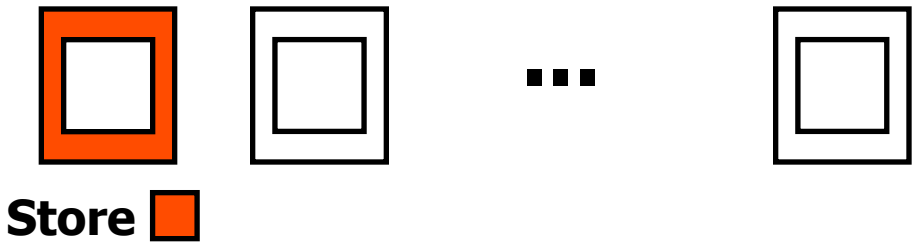
C. CC.: políticas

- Políticas para mantener la coherencia
  - Son independientes de su implementación

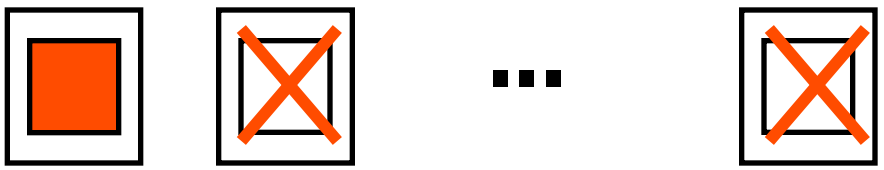
    A) **Invalidación (I)**: *Write-Invalidate*
    - Cuando un procesador modifica su copia de un bloque se señalan como 'no válidas' (inválidas) el resto de las copias existentes.

    B) **Actualización (A)**: *Write-Update/Broadcast*
    - Cuando un procesador modifica su copia de un bloque se actualiza con la modificación el resto de las copias existentes.

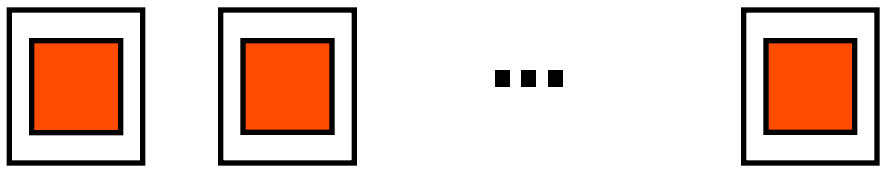
C. CC.: políticas: I vs. A



Invalidación



Actualización



Variación:

write-once (Goodman, 1983)

C. CC.: políticas: rendimiento

- Nomenclatura:
  - **P**: tasa de fallos "monoprocesador: el bloque no está presente en la cache.
  - **F**: tasa de fallos total.
  - **I**: tasa de fallos por invalidación (*invalidation misses*): el bloque está presente en la cache pero en estado 'no-válido'.
  - **T**: tráfico extra debido al mecanismo de coherencia.
  - **B**: tamaño del bloque.
  - **C**: tamaño (capacidad) de la cache.

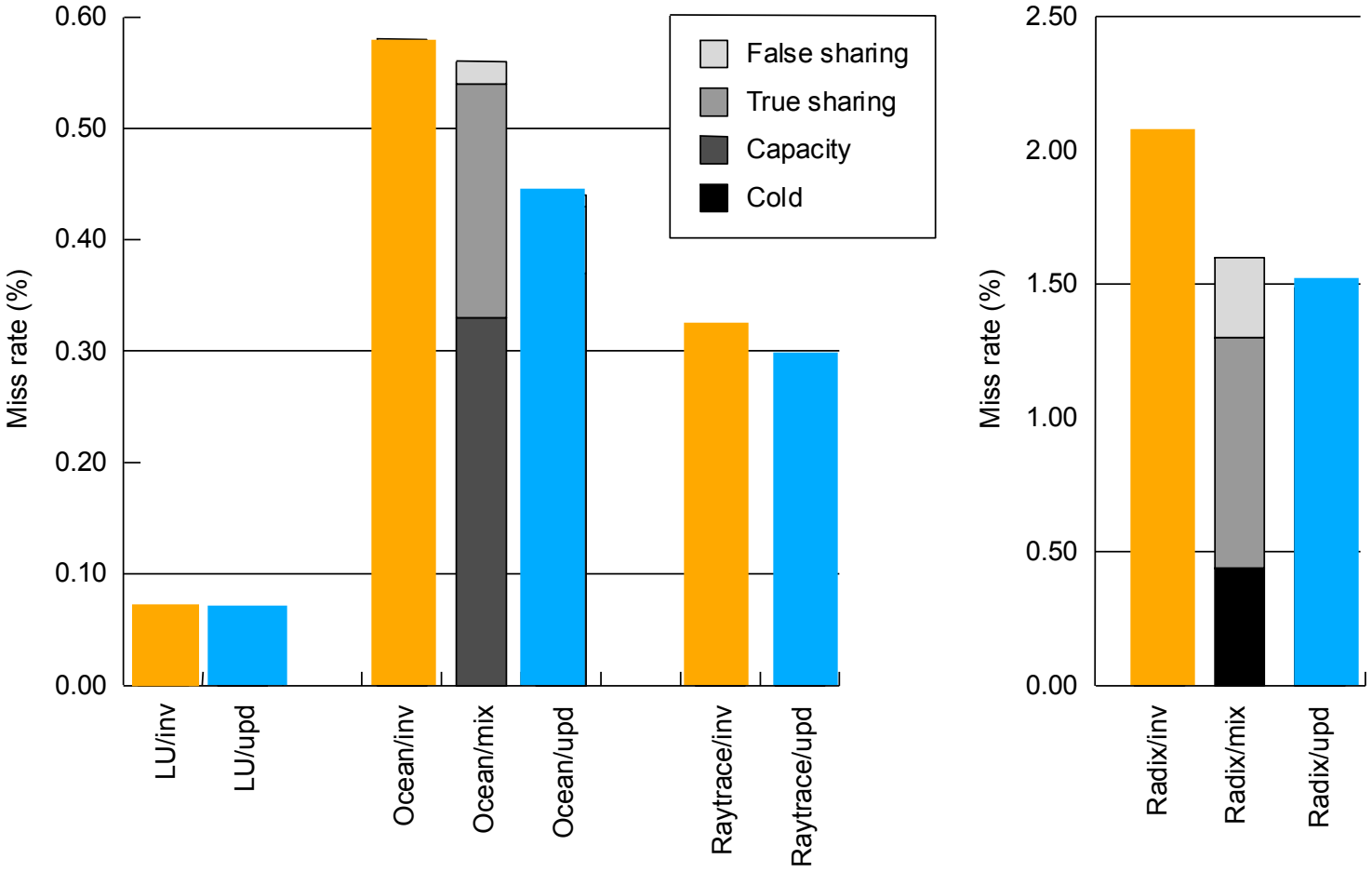
Rendimiento	Invalidación	Actualización
a) Tasa de fallos	$F = P + I$	$F = P$
b) Tráfico añadido	$T_I = T_{invalidar} + T_{fallos\ por\ invalidación}$ *	$T_A >> T_I$ **

* reducido por *write-once*

** Dependerá de la frecuencia de los fallos por Invalidación

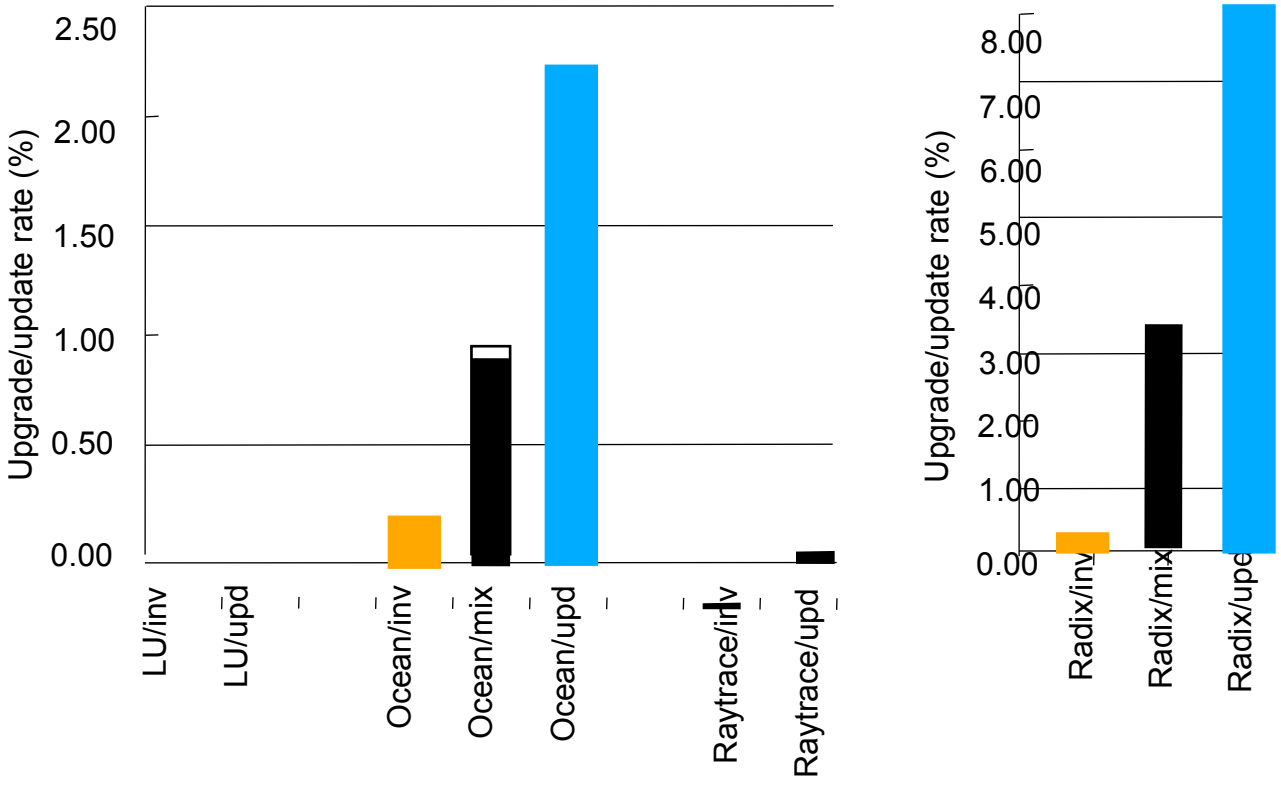
C. CC.: políticas: I vs. A

Culler *et al.*: Ej. **tasa de fallos**



C. CC.: políticas: I vs. A

Culler *et al.*: Ej. **tráfico añadido**



C. CC.: políticas: ¿I ó A?

¿I ó A?

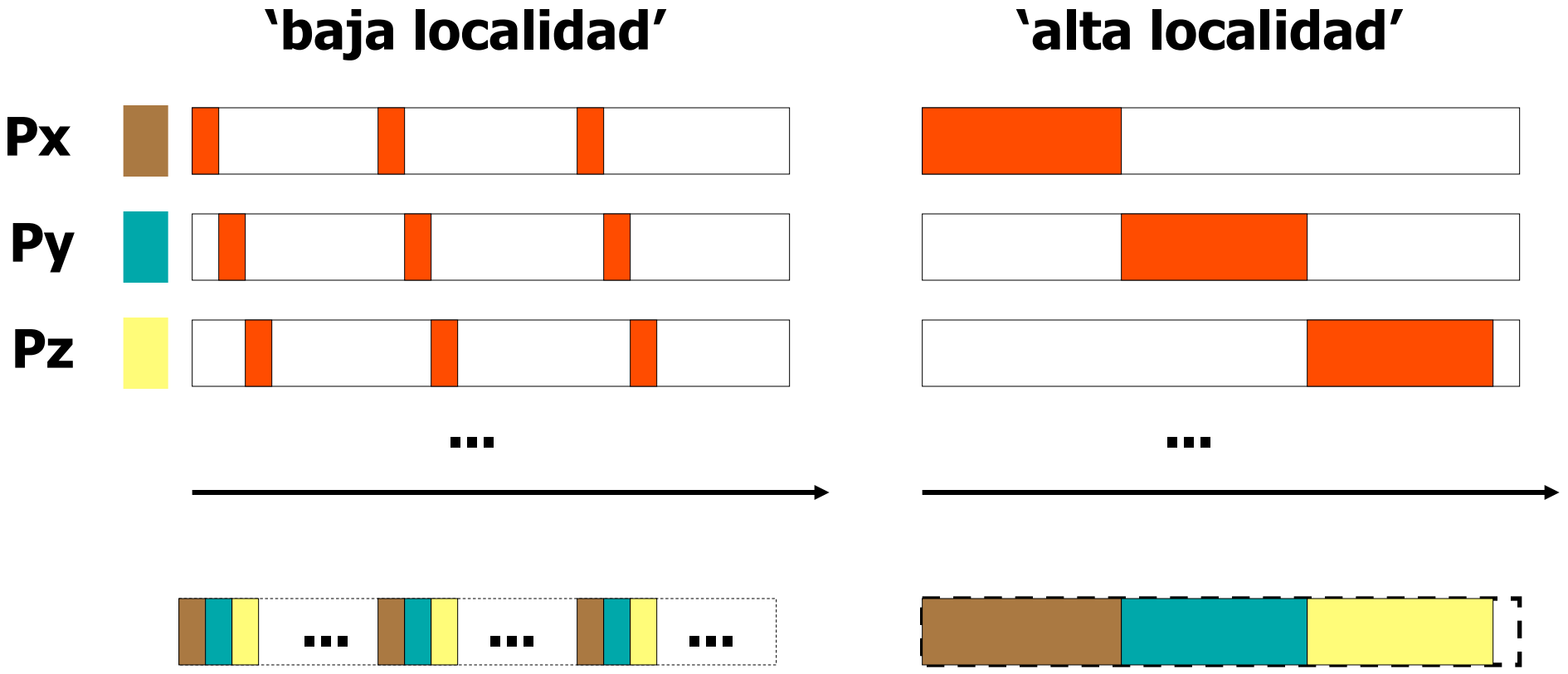
Depende del comportamiento del programa: extremos.

Empíricamente, en la vida real:

1) No se comparte de un modo extremo.
2) Cada bloque puede tener un comportamiento distinto.
3) Y encima éste puede cambiar dinámicamente.

3. CC-I: 'baja localidad' vs. 'alta localidad'

Maneras de compartir un bloque



C. CC.: 'grano fino' vs. 'secuencial'

I ✗ *ping-pong:* **ej. lock**

I ✓

A ✓

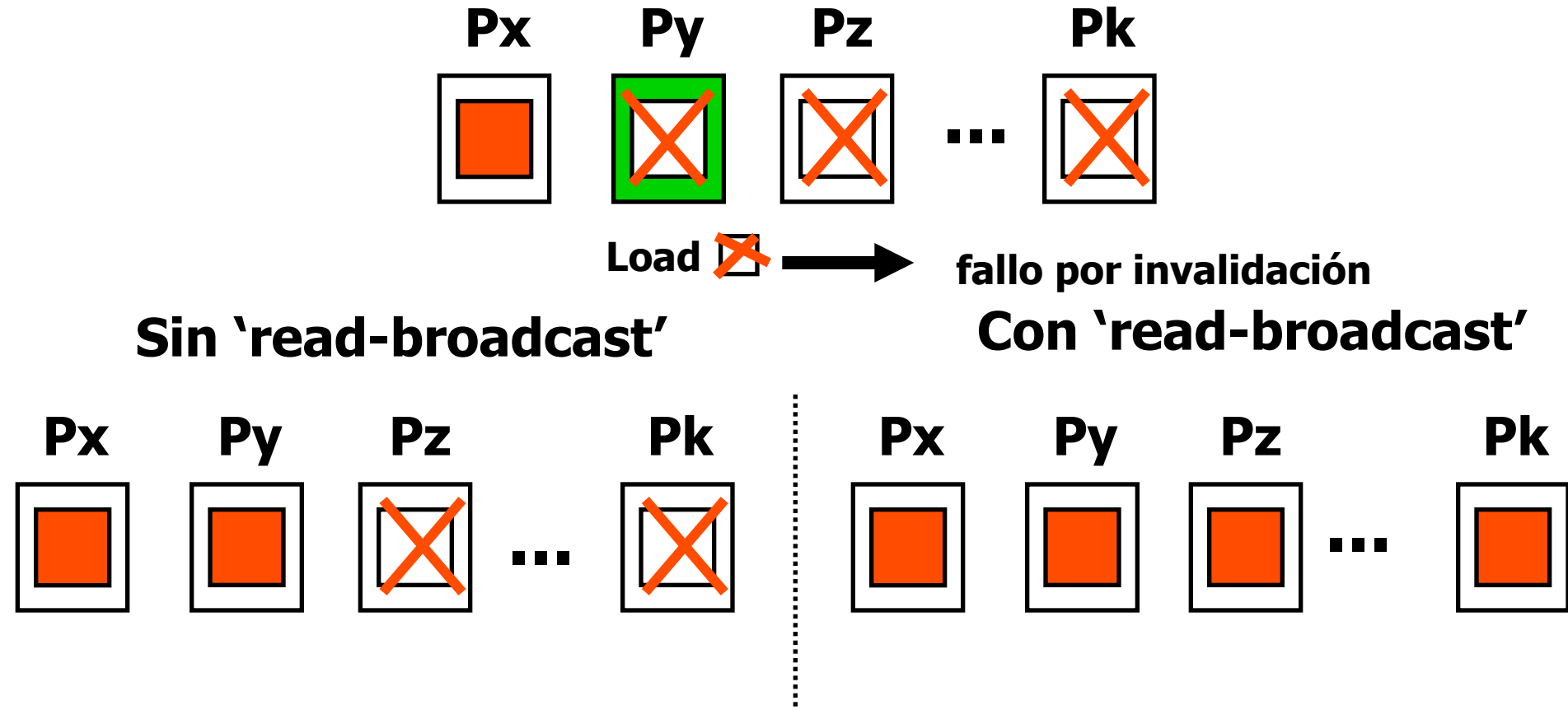
A ✗

act. inútiles

C. CC.: políticas: variaciones

- Variaciones: tratan de paliar defectos.
  - **Invalidación**:
    1. (***write-once***: por defecto, siempre)
    2. ***read-broadcast***: reduce a un fallo global por invalidación: particularmente adecuado para un procesador que escribe y varios que leen.
  - **Actualización**:
    1. Difundir escrituras sólo si el bloque está realmente compartido.
    2. Aunque esté presente en otras caches puede no ser útil: envejecimiento de los bloques: ej. ***competitive snooping***: cuando todos las copias están "envejecidas" => invalidar y no difundir más escrituras.

C. CC.: *read-broadcast*



C. CC.: políticas: vida real

- Vida real:
  - Hay máquinas que permiten **ambas políticas** de coherencia.
  - La **mayoría** de las arquitecturas implementa **Invalidación**: p.ej., muy popular el **MESI** (p.ej., Pentium).
  - **No** suelen implementar **sólo Actualización**:
    - Mucho tráfico en general.
    - Situación peor: varias escrituras por un procesador antes de la lectura por otro: varias operaciones de bus frente a 1 sola en Invalidación.
    - Problemática para protocolos no basados en buses, o sea, 'escalables'.

C. CC.: implementaciones

- Implementaciones:
  - **Hardware**:
    - ***snoopy***: máquinas basadas en buses: *broadcast*
    - **directorios**: máquinas 'escalables': otras redes de intercomunicación: *multicast*
    - "**redes coherentes**": MP con una red de interconexión diseñada expresamente para mantener la coherencia: DDM (COMA), Wisconsin Multicube, KSR1 (tb. COMA): ¿fracasos?
  - **Software**:
    - no realizaciones comerciales
    - cuestión en que parece completamente parada

C. CC.: implementaciones *snoopy*

- Implementaciones *snoopy*:
  - Se utilizan en MP en los que la red de interconexión es un **bus**.
  - Idea básica:
    - Las transacciones en el bus son visibles a todos los procesadores.
    - Los procesadores (realmente, sus controladores de cache) observan lo que pasa en el bus y realizan acciones para preservar la coherencia.

C. CC.: bibliografía

- Libros "de texto":
  - El Culler *et al*. y el Sima *et al*.
- Libros específicos: [compilaciones de "papers"]
  - M. Tomásevic, V. Milutinovic: *The Cache Coherence Problem in Shared-Memory Multiprocessors:* ***Hardware Solutions***, IEEE Computer Society Press, 1993
  - I. Tartalja, V. Milutinovic: *The Cache Coherence Problem in Shared-Memory Multiprocessors:* ***Software Solutions***, IEEE Computer Society Press, 1996