CompSci131

Parallel and Distributed Systems

Prof. A. Veidenbaum

Today's topics

- **Cache Coherence**
- **Coherence protocols**

- **Reading assignment:**
  - **Today: Cache coherence (lecture notes only)**
  - **Next time: more coherence, the rest of 7.2**
  - **After that: 7.3**

Last Lecture Covered

- **Causal Consistency**
- **Grouping operations**

Cache coherence vs consistency

- Caches are replicas of shared memory
  - But not full replicas, as Cache_size << Memory_size
  - A process (core) accesses its local cache only
    - » A conit here is a 64Byte block (aligned), aka cache line
      - Identified by its memory address
- Thus there is a consistency problem
  - But caches are hardware managed, so need simplicity
- Coherence is a special case of consistency
  - Deals with consistency for a single address/block
- Finally, a cache may not contain a requested block
  - Has to be brought in from memory or other replicas

Cache Terminology

- **Processor Pi issues Rd/Wr requests to its local cache**
- **A cache hit:**
  - **Pi reads/writes an address A from Ci (i.e. finds it there)**
- **A cache miss:**
  - **Pi tries accessing A, but it's not in Ci**
    - » **Is read from memory into Ci at this point**
- **A write-back:**
  - **The space occupied by a modified block is needed, so the block is written back to memory**
    - » **Not when the write happens (a write-back cache)**

Cache coherence protocols

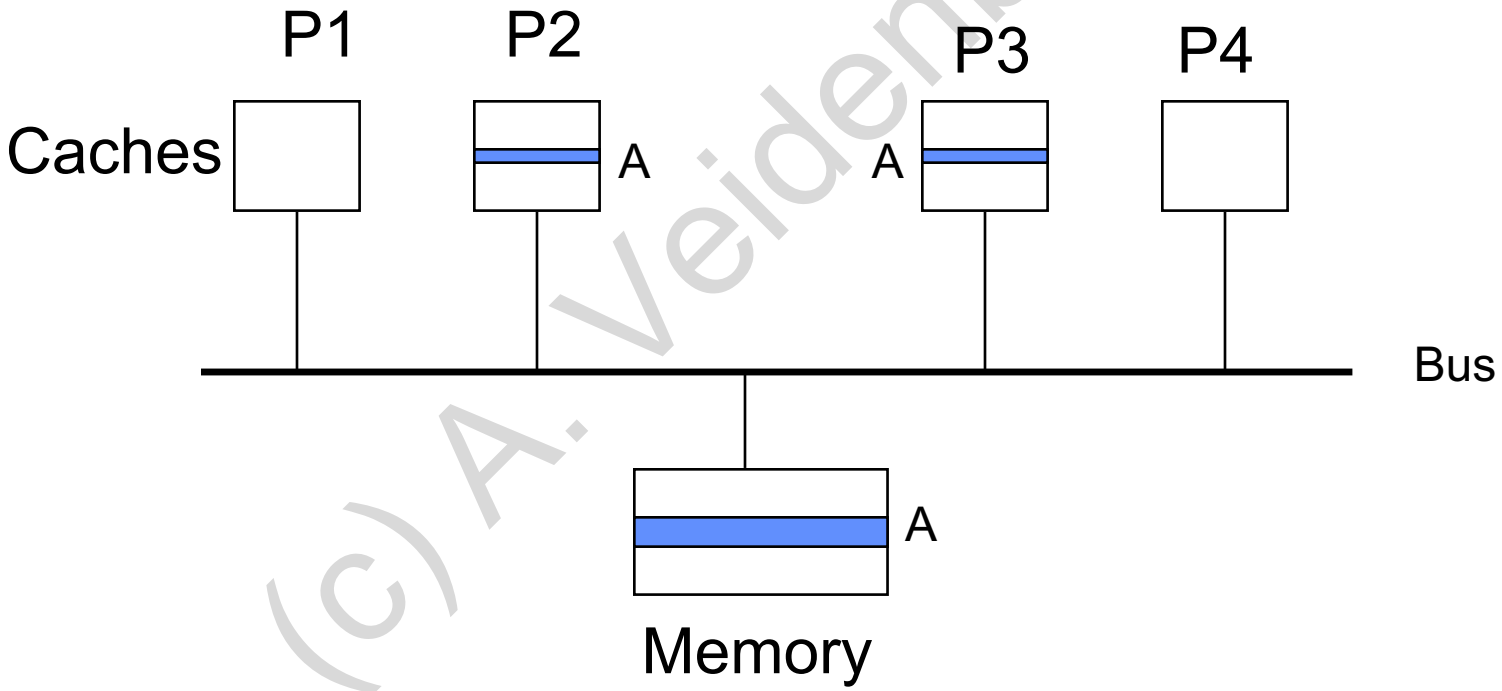
- **Coherence guarantees update of all existing replicas**
  - ***Of a given block***
- **Many shared memory coherence protocols**
  - **With hardware support**
- **General idea –a write causes coherence enforcement**
- **Also possible in software**
  - **Middleware-based systems with caches use software protocols for coherence**

SMP Protocols

- Some slides courtesy of Prof. J.-L. Baer
- Caches and memory are connected by a bus
  - This means every bus access acts as a broadcast
    - » E.g. like an ethernet cable
    - » Bus access is arbitrated, one unit at a time
- A number of possible design choices
  - They are variants of MOESI or MESI protocols
    - » 1st letters of states – Modified, Exclusive, Shared, Invalid

Cache Coherence: Review

- Initial state: P2 reads A; P3 reads A
  - From memory



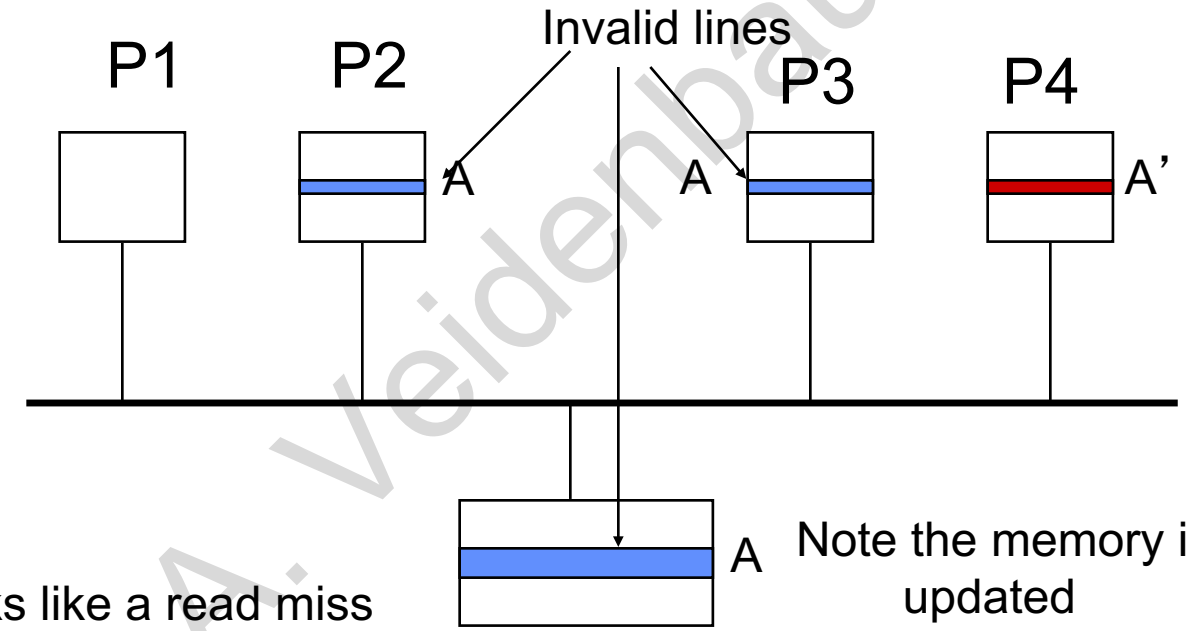
Cache coherence (shared-bus)

- **Now P4 wants to write to A – *and writes are the issue!***
  - ***It is a write miss for P4, it fetches the block from memory***
- **Broadcast an invalidation message for address A**
  - **the address is snooped by the P2,P3 caches (and all others)**
  - **Seeing a write, P2 and P3 invalidate their copies**
    - » **This is called a write-invalidate protocol**
      - **Note that the copy in memory is *not up-to-date* any longer**
        - **Because it is write-back cache**
- **Now P4 writes to block A in its cache**
  - ***An alternative implementation is to request ownership on a read miss***

)

A Write-invalidate example

- P2 and P3 have read line A
- P4 has a write miss on an element of line A



Note the memory is not updated

- A write miss looks like a read miss
  - Brings the old value of A to P4
- Followed by the broadcast of an invalidation and a write

Memory Update

- **What happens next time P2 wants to read A?**
  - ***Cannot fetch the block from memory!***
- **But the read request is broadcast on the bus!**
  - **the address is *snooped* by P4's caches**
    - » **Now is a good time to update memory!**
  - **C4 can make C2 abort its read, then write back to memory**
  - **Now C2 will retry an get it from memory!**

Snoopy Cache Coherence Protocols

- Need protocols to perform all these actions
  - And in the right order
- Use FSM and associate a state with each cache line

- Let us start with 3 states:
  - Invalid (I) - not present in this cache
  - Shared (S) - one or more unmodified, up to date copies
  - Modified (M) – The only up-to-date copy

- 4th or 5th state can be added for performance
  - MESI: E stands for Exclusive
  - MOESI protocols: adds O for Ownership

State Transitions for a Given Cache Line

- Occur in a given cache in response to

1. Processor requests to its cache
   - Read miss, write miss, write on shared line

2. Bus snooping of other processor actions on the bus
   - Read miss by Q would make P's line transit from M to S
   - Write miss by Q might make P's line transit from S to I
     - » write invalidate protocol

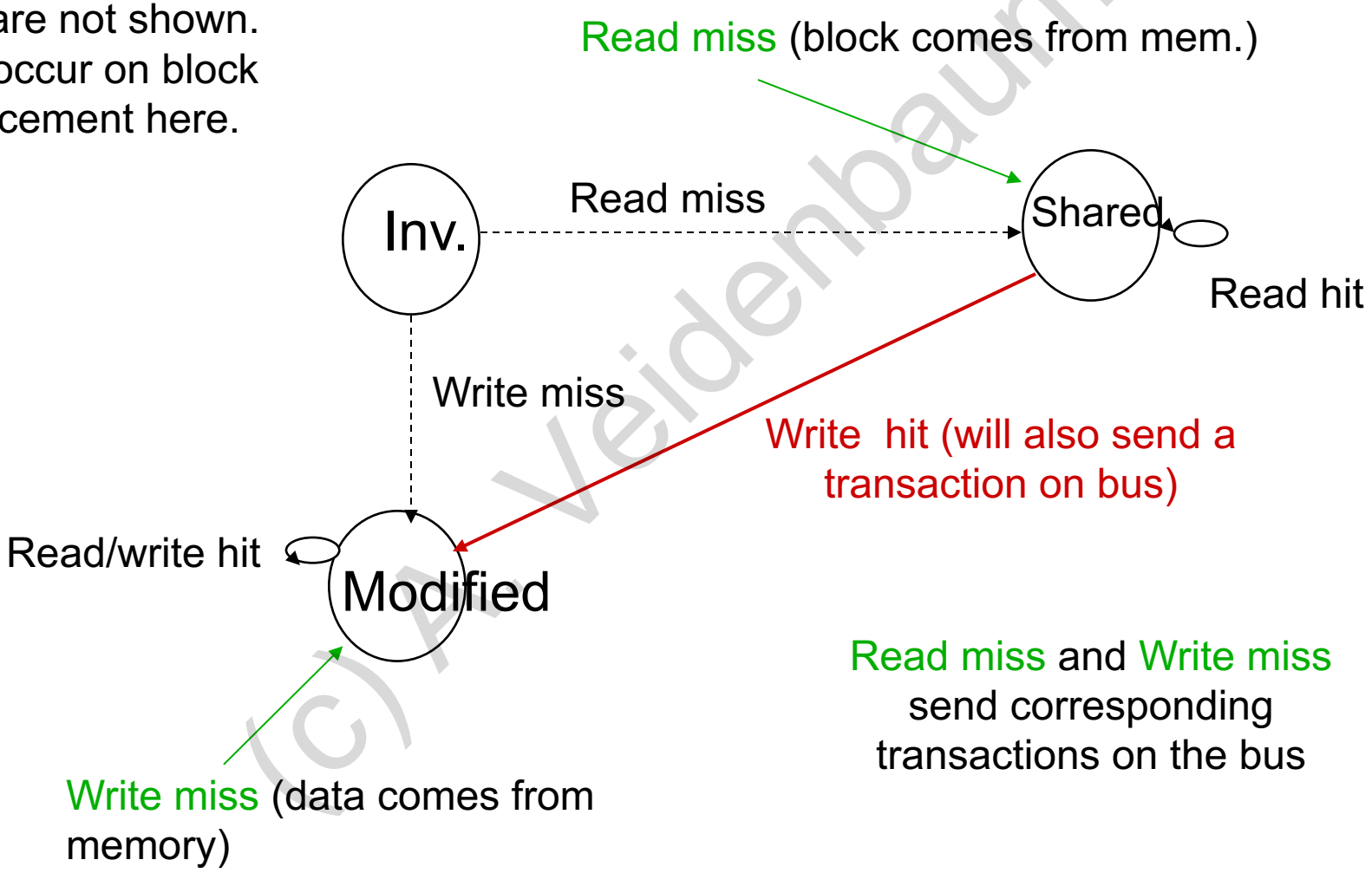
Basic Write-invalidate Protocol (write-back write-allocate caches)

- Uses 3 states for each cache line
  - Invalid
  - Shared (read only – can be shared)
  - Modified (only valid copy in the system)

- Different state transitions are
  - Induced by the processor attached to the cache
  - Induced by snooping on the bus

3-State Protocol: Processor Actions

Transitions to Invalid state are not shown. They occur on block replacement here.



Read miss and Write miss send corresponding transactions on the bus

Basic 3 State Protocol: Transitions from Bus Snooping