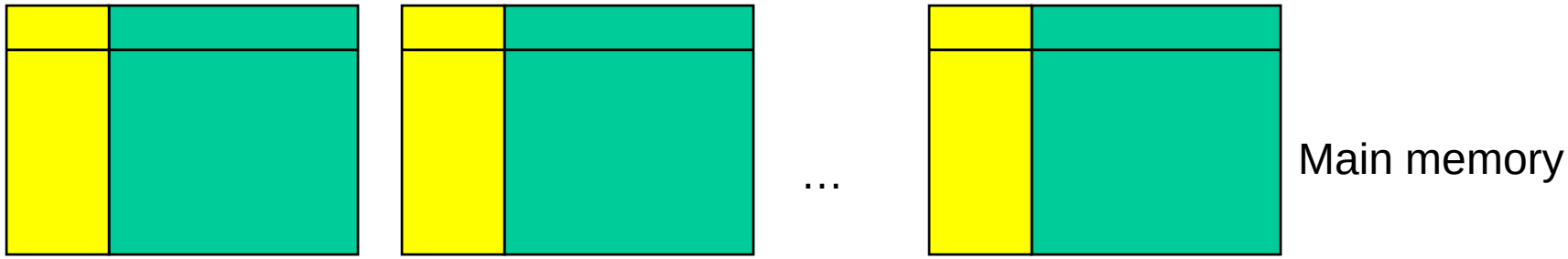
Lecture 5: Directory Protocols

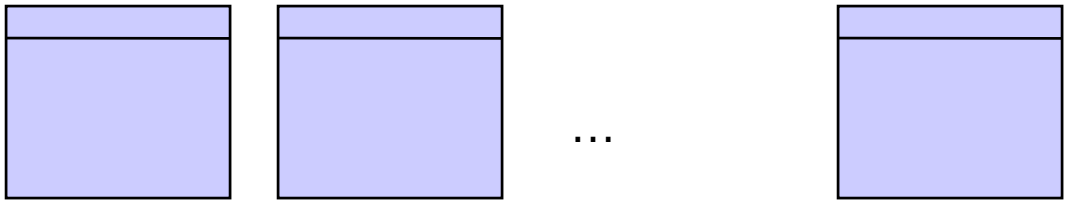
- Topics: directory-based cache coherence implementations

Flat Memory-Based Directories

Block size = 128 B
Memory in each node = 1 GB
Cache in each node = 1 MB

For 64 nodes and 64-bit directory,
Directory size = 4 GB
For 64 nodes and 12-bit directory,
Directory size = 0.75 GB

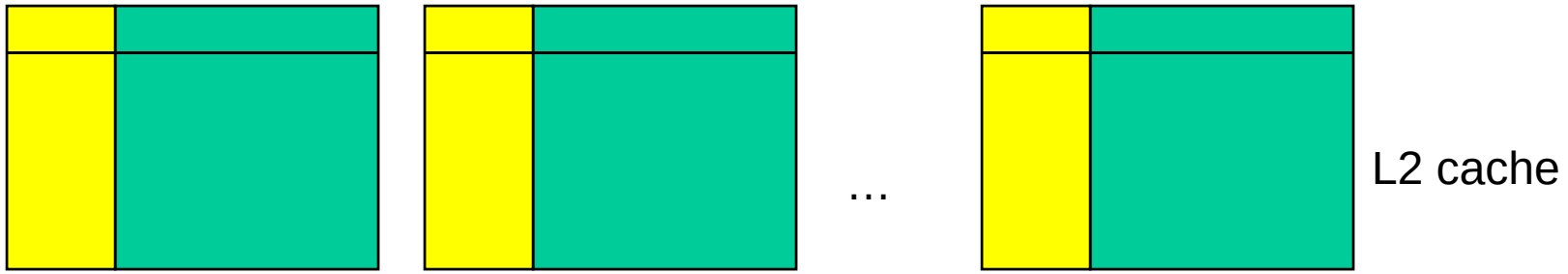


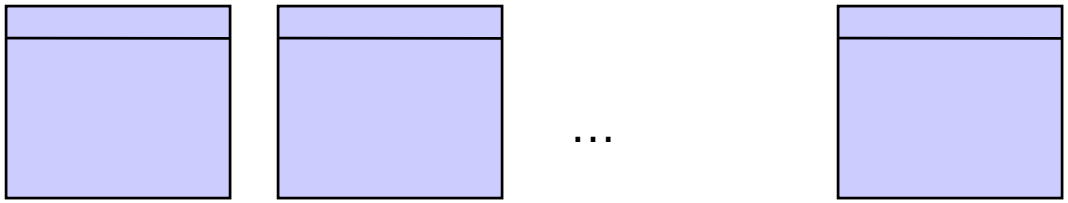


Flat Memory-Based Directories

Block size = 64 B
L2 cache in each node = 1 MB
L1 Cache in each node = 64 KB

For 64 nodes and 64-bit directory,
Directory size = 8 MB
For 64 nodes and 12-bit directory,
Directory size = 1.5 MB



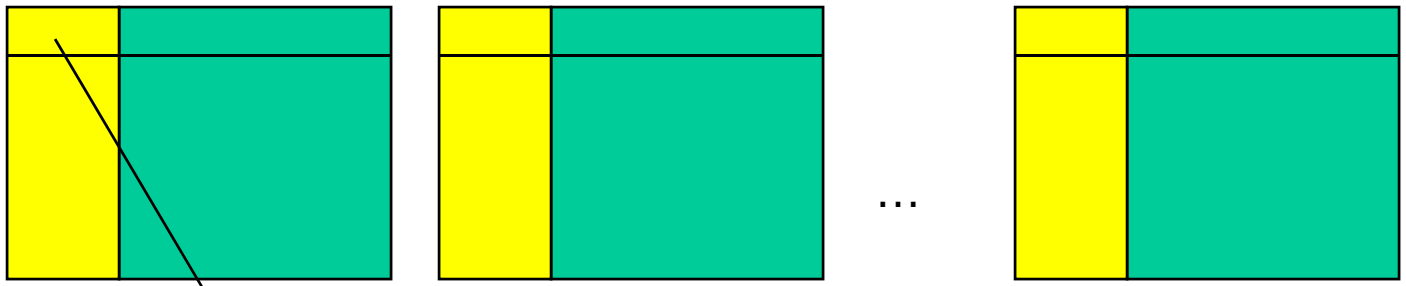


Flat Cache-Based Directories

Block size = 128 B
Memory in each node = 1 GB
Cache in each node = 1 MB

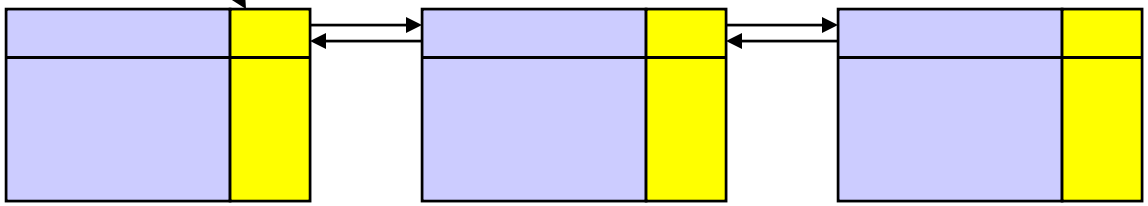
6-bit storage in DRAM for each block;
DRAM overhead = 0.375 GB

12-bit storage in SRAM for each block;
SRAM overhead = 0.75 MB



…

Main memory



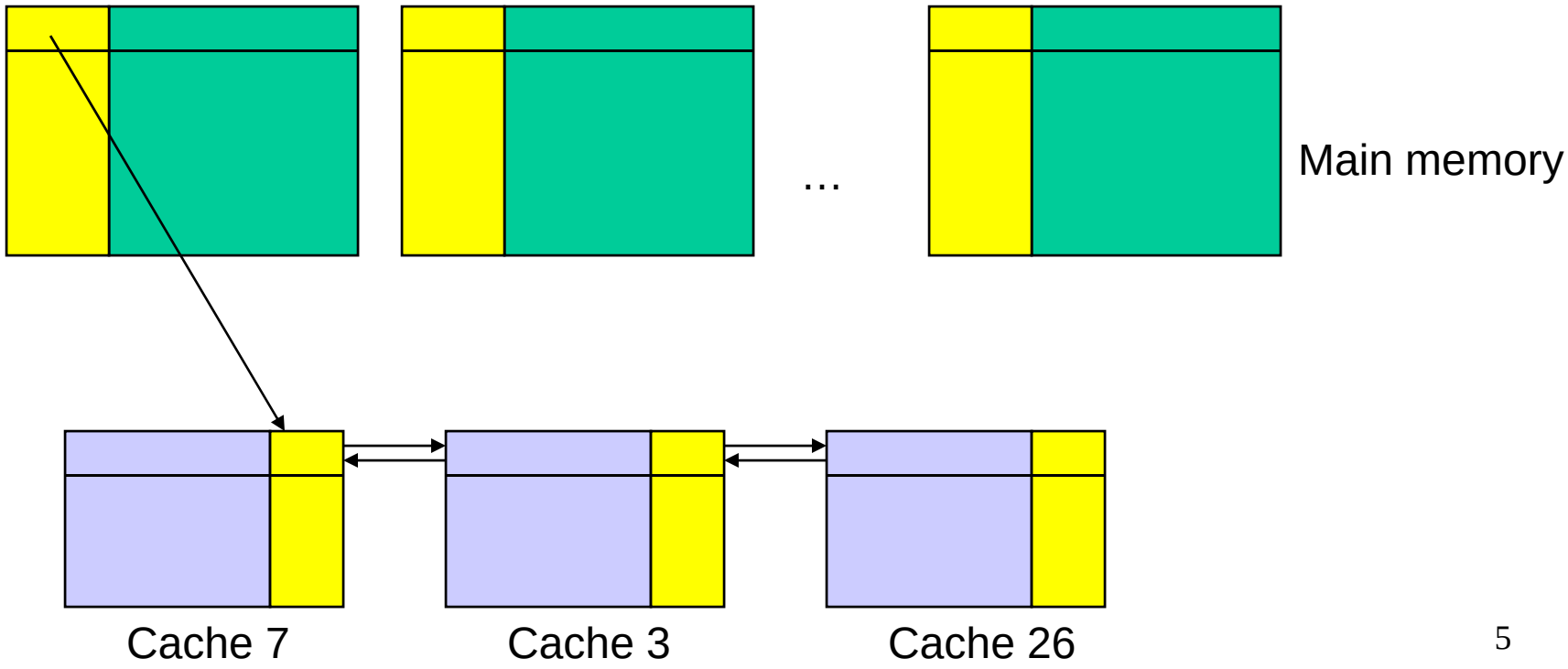
Cache 7 Cache 3 Cache 26

Flat Cache-Based Directories

Block size = 64 B
L2 cache in each node = 1 MB
L1 Cache in each node = 64 KB

6-bit storage in L2 for each block;
L2 overhead = 0.75 MB

12-bit storage in L1 for each block;
L1 overhead = 96 KB



Flat Cache-Based Directories

- The directory at the memory home node only stores a pointer to the first cached copy – the caches store pointers to the next and previous sharers (a doubly linked list)
- Potentially lower storage, no bottleneck for network traffic,
- Invalidates are now serialized (takes longer to acquire exclusive access), replacements must update linked list, must handle race conditions while updating list

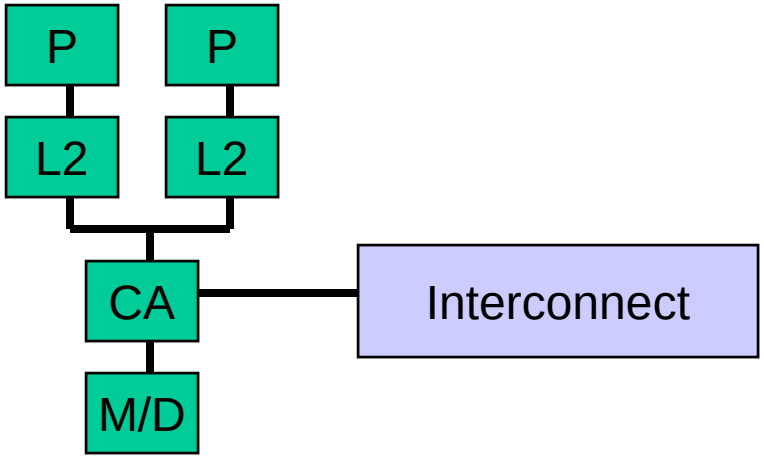
Serializing Writes for Coherence

- Potential problems: updates may be re-ordered by the network; General solution: do not start the next write until the previous one has completed

- Strategies for buffering writes:
  - buffer at home: requires more storage at home node
  - buffer at requestors: the request is forwarded to the previous requestor and a linked list is formed
  - NACK and retry: the home node nacks all requests until the outstanding request has completed

SGI Origin 2000

- Flat memory-based directory protocol
- Uses a bit vector directory representation
- Two processors per node, but there is no snooping protocol within a node – combining multiple processors in a node reduces cost



Protocol States

- Each memory block has seven states
- Three stable states: unowned, shared, exclusive (either dirty or clean)
- Three busy states indicate that the home has not completed the previous request for that block (read, read-excl or upgrade, uncached read)
- Poison state – used for lazy TLB shootdown

Handling Reads

- When the home receives a read request, it looks up memory (speculative read) and directory in parallel

- Actions taken for each directory state:
  - shared or unowned: memory copy is clean, data is returned to requestor, state is changed to excl if there are no other sharers
  - busy: a NACK is sent to the requestor
  - exclusive: home is not the owner, request is fwded to owner, owner sends data to requestor and home

Inner Details of Handling the Read

- The block is in exclusive state – memory may or may not have a clean copy – it is speculatively read anyway
- The directory state is set to busy-exclusive and the presence vector is updated
- In addition to fwding the request to the owner, the memory copy is speculatively forwarded to the requestor
  - Case 1: excl-dirty: owner sends block to requestor and home, the speculatively sent data is over-written
  - Case 2: excl-clean: owner sends an ack (without data) to requestor and home, requestor waits for this ack before it moves on with speculatively sent data

Inner Details II

- Why did we send the block speculatively to the requestor if it does not save traffic or latency?
  - the R10K cache controller is programmed to not respond with data if it has a block in excl-clean state
  - when an excl-clean block is replaced from the cache, the directory need not be updated – hence, directory cannot rely on the owner to provide data and speculatively provides data on its own

Handling Write Requests

- The home node must invalidate all sharers and all invalidations must be acked (to the requestor), the requestor is informed of the number of invalidates to expect

- Actions taken for each state:
  - shared: invalidates are sent, state is changed to excl, data and num-sharers is sent to requestor, the requestor cannot continue until it receives all acks (Note: the directory does not maintain busy state, subsequent requests will be fwded to new owner and they must be buffered until the previous write has completed)

Handling Writes II

- Actions taken for each state:
  - unowned: if the request was an upgrade and not a read-exclusive, is there a problem?
  - exclusive: is there a problem if the request was an upgrade? In case of a read-exclusive: directory is set to busy, speculative reply is sent to requestor, invalidate is sent to owner, owner sends data to requestor (if dirty), and a “transfer of ownership” message (no data) to home to change out of busy
  - busy: the request is NACKed and the requestor must try again

Handling Write-Back

- When a dirty block is replaced, a writeback is generated and the home sends back an ack

- Can the directory state be shared when a writeback is received by the directory?

- Actions taken for each directory state:
  - exclusive: change directory state to unowned and send an ack
  - busy: a request and the writeback have crossed paths: the writeback changes directory state to shared or excl (depending on the busy state), memory is updated, and home sends data to requestor, the intervention request is dropped

Serialization

- Note that the directory serializes writes to a location, but does not know when a write/read has completed at any processor

- For example, a read reply may be floating on the network and may reach the requestor much later – in the meantime, the directory has already issued a number of invalidates, the invalidate is overwritten when the read reply finally shows up – hence, each node must buffer its requests until outstanding requests have completed

Serialization - II

- Assume that a dirty block is being passed from P1 to another writer P2, the “ownership transfer” message from P1 to home takes a long time, P2 receives its data and carries on, P2 does a writeback → protocol must be designed to handle this case correctly

If the writeback is from the node that placed the directory in busy state, the writeback is NACKed

(If instead, the writeback was allowed to proceed, at some later point, if the directory was expecting an “ownership transfer”, it may mis-interpret the “floating” message)

Directory Structure

- The system supports either a 16-bit or 64-bit directory (fixed cost)
- For small systems, the directory works as a full bit vector representation
- For larger systems, a coarse vector is employed – each bit represents p/64 nodes
- State is maintained for each node, not each processor – the communication assist broadcasts requests to both processors

Page Migration

- Each page in memory has an array of counters to detect if a page has more misses from a node other than home
- When a page is moved to a different physical memory location, the virtual address remains the same, but the page table and TLBs must be updated
- To reduce the cost of TLB shootdown, the old page sets its directory state to poisoned – if a process tries to access this page, the OS intervenes and updates the translation

Title

- Bullet