Directory-based Cache Coherence:

- Replacement of S blocks
- Serialization
- VN deadlock
- Starvation
- Overflow schemes
- Sparse directory
- Remote access cache
- COMA
- Latency tolerance
- Page migration
- Queue lock in hardware

[From Chapter 8 of Culler, Singh, Gupta]
[SGI Origin 2000 material taken from Laudon and Lenoski, ISCA 1997]
[GS320 material taken from Gharachorloo et al., ASPLOS 2000]

Previous Next

Replacement of S blocks

- Send notification to directory?
    - Can save a future invalidation
    - Does it reduce overall traffic?
- Origin 2000 does not use replacement hints
    - No notification to directory
    - Why?
- Replacements of E blocks are hinted and require acknowledgments also (why?)
- Summary of transaction types
    - Coherence: 9 request transaction types, 6 invalidation/intervention, 39 reply types
    - Non-coherent (I/O, synch, special): 19 requests, 14 replies

Serialization

- Home is used to serialize requests
    - The order determined by the home is final
    - No node should violate this order
    - Example: read-invalidate races
        - P0, P1, and P2 are trying to access a cache block
        - P0 and P2 want to read while P1 wants to write
        - The requests from P0 and P2 reach home first, home replies and marks both in sharer vector; but the reply message to P0 gets delayed in the network
        - P1's write causes home to send out invalidation to P0 and P2; P0's inv. reaches P0 before the read reply
        - P0's hub sends acknowledgment to P1 and also forwards the invalidation to P0's processor cache
        - What happens when P0's reply arrives? Can the data be used?
- Requester's viewpoint
    - When a read reply arrives it finds the OTT entry has the "inv" bit set
    - Under what conditions can it happen?
        - Seen one in the last slide
    - Can replacement hints help?
- What about upgrade-invalidation races?
- What about readX-invalidation races?

VN deadlock

- Origin 2000 has only two virtual networks, but has three-hop transactions
    - Resorts to back-off invalidate or intervention to fall back to strict request-reply
    - Does it really solve the problem or just move the problem elsewhere?
- Stanford DASH has same problems
    - Uses NACKs after a time-out period if the outgoing network doesn't free up
    - Worse compared to Origin because NACKs inflate total traffic and may lead to livelock
    - DASH avoids livelocks by sizing the queues according to the machine size (not a scalable solution)

Previous Next

Starvation

- NACKs can cause starvation
    - Build a FIFO list of waiters either in home memory (Chaudhuri and Heinrich, 2004) or use a distributed linked list (IEEE Scalable Coherent Interface)
        - Former imposes large occupancy on home, yet offers better performance by read combining
        - Latter is an extremely complex protocol with a large number of transient states and 29 stable states, but does distribute the occupancy across the system
    - Origin 2000 devotes extra bits in the directory to raise priority of requests NACKed too many times (above a threshold)
    - Use delay between retries
    - Use Alpha GS320 protocol (will discuss later)

Overflow schemes

- How to make the directory size independent of the number of processors
    - Basic idea is to have a bit vector scheme until the total number of sharers is not more than the directory entry width
    - When the number of sharers overflows the hardware resorts to an “overflow scheme”
        - Dir_iB: i sharer bits, broadcast invalidation on overflow
        - Dir_iNB: pick one sharer and invalidate it
        - Dir_iCV: assign one bit to a group of nodes of size P/i; broadcast invalidations to that group on a write
            - May generate useless invalidations
- Dir_iDP (Stanford FLASH)
    - DP stands for dynamic pointer
    - Allocate directory entries from a free list pool maintained in memory
    - Need replacement hints
    - Still may run into reclamation mode if free list pool is not sized properly at boot time
        - How do you size it?
    - If replacement hints are not supported, assume k sharers on average per cache block (k=8 is found to be good)
    - Reclamation algorithms?
        - Pick a random cache line and invalidate it
- Dir_iSW (MIT Alewife)
    - Trap to software on overflow
    - Software maintains the information about overflown sharers
    - MIT Alewife has directory entry of five pointers and a local bit (i.e. overflow threshold is five or six)
        - Remote read before overflow takes 40 cycles and after overflow takes 425 cycles
        - Five invalidations take 84 cycles while six invalidations take 707 cycles

Sparse directory

- How to reduce the height of the directory?

Observation: total number of cache blocks in all processors is far less than total number of memory blocks

  - Assume a 32 MB L3 cache and 4 GB memory: less than 1% of directory entries are active at any point in time

- Idea is to organize directory as a highly associative cache
- On a directory entry “eviction” send invalidations to all sharers or retrieve line if dirty

Previous Next

Remote access cache

- Essentially a large tertiary cache
    - Captures remote cache blocks evicted from local cache hierarchy
    - Also visible to the coherence protocol: so inclusion must be maintained with processor caches
    - Must be highly associative and larger than the outermost level of cache
    - Usually part of DRAM is reserved for RAC
    - For multiprocessor nodes, requests from different processors to the same cache block can be merged together; also there is a prefetching effect
    - Used in Stanford DASH
    - Disadvantage: latency and space

COMA

- Cache-only memory architecture
    - Solves the space problem of RAC
    - Home node only maintains the directory entries, but may not have the cache block in memory
    - A node requesting a cache block brings it to its local memory and local cache as usual
    - Entire memory is treated as a large tertiary cache
        - Known as the attraction memory (AM)
    - Home as well as any node having a cache block maintain a directory entry for the cache block
    - A request first looks up AM directory state and, if unowned, gets forwarded to home which, in turn, forwards it to one of the sharers
- Cache-only memory architecture
    - To start with home has the cache blocks
    - It retains a cache block until it is replaced by some other migration
    - There is always a master copy of each cache block
        - The last valid copy
    - What happens on a replacement of the master copy?
        - Swap with source of migrating cache block
    - Latency problem remains at the requester
    - Inclusion problems between AM and processor cache hierarchy
        - Complicates the protocol

Previous Next

Latency tolerance

- Page placement
    - Application-directed page placement is used in many cases to minimize the number of remote misses
    - The application provides the kernel (via system call) the starting address and ending address of a chunk of memory (multiple of pages) and also the node number where to map these pages
    - Thus an application writer can specify (based on sharing pattern) which shared pages he/she wants to map in a node's local memory (private pages and stack pages are mapped on local memory by default)
    - The page fault handler of a NUMA kernel is normally equipped with some default policies e.g., round robin mapping or first-touch mapping
    - Examples: matrix-vector multiplication, matrix transpose
- Software prefetching
    - Even after rigorous analysis of a parallel application it may not be possible to map all pages used by a node on its local memory: the same page may be used by multiple nodes at different times (example: matrix-vector multiplication)
    - Two options are available: dynamic page migration (very costly; coming next, stay tuned) or software prefetching
    - Today, most microprocessors support prefetch and prefetch exclusive instructions: use prefetch to initiate a cache line read miss long before application actually accesses it; use prefetch exclusive if you know for sure that you will write to the line and no one else will need the line before you write to it
    - Prefetches must be used carefully
        - Swap with source of migrating cache block
        - Early prefetches may evict useful cache blocks and itself may get evicted before use; may generate extra invalidations or interventions in multiprocessors
        - Late prefetches may not be fully effective, but at least less harmful than early prefetches
        - Wrong prefetches are most dangerous: these bring in cache blocks that may not be used at all in near-future; in multiprocessors this can severely hurt performance by generating extra invalidations and interventions
        - Wrong prefetches waste bandwidth and pollute cache
    - Software prefetching usually offers better control than hardware prefetching
- Software prefetching vs. hardware prefetching
    - Software prefetching requires close analysis of program; profile information may help
    - Hardware prefetching tries to detect patterns in accessed addresses and using the detected patterns predicts future addresses
        - AMD Athlon has a simple next line prefetcher (works perfectly for most numerical applications)
        - Intel Pentium 4 has a very sophisticated stream prefetcher

Previous Next

Page migration

- Page migration changes the existing VA to PA mapping of the migrated page
    - Requires notifying all TLBs caching the old mapping
    - Introduces a TLB coherence problem
- Origin 2000 uses a smart page migration algorithm: allows the page copy and TLB shootdown to proceed in parallel
    - Array of 64 page reference counters per directory entry to decide whether to migrate a page or not: compare requester’s counter against home’s and send an interrupt to home if migration is required
- What does the interrupt handler do?
    - Access all directory entries of the lines belonging to the to-be migrated page
    - Send invalidations to sharers or interventions to owners; at the end all cache lines of that page must be in memory
    - Set the poison bits in the directory entries of all the cache lines of the page
    - Start a block transfer of the page from home to requester at this point (30 µs to copy 16 KB)
- An access to a poisoned cache line from a node results in a bus error which invalidates the TLB entry for that page in the requesting node (avoids broadcast shootdown)
- Until the page is completely migrated and is assigned a physical page frame on target node, all nodes accessing a poisoned line wait in a pending queue
- After the page copy is completed the waiting nodes are served one by one; however, the directory entries and the page itself are moved to a “poisoned list” and are not yet freed at the home (i.e. you still cannot use that physical page frame)
- On every scheduler tick the kernel invalidates one TLB entry per processor
- After a time equal to TLB entries per processor multiplied by scheduling quantum the page frame is marked free and is removed from the poisoned list
- Major advantage: requesting nodes only see the page copy latency including invalidation and interventions in critical path, but not the TLB shootdown latency

Queue lock in hardware

- Stanford DASH
    - Memory controller recognizes lock accesses
        - Requires changes in compiler and instruction set
    - Marks the directory entry with contenders
    - On unlock a contender is chosen and lock is granted to that node
    - Unlock is forced to generate a notification message to home
        - Possibly requires special cache state for lock variables or special uncached instructions for unlock if lock variables are not allowed to be cached

Previous Next