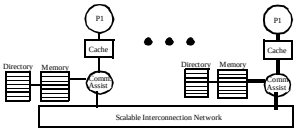
Scalable CC-NUMA Design Study - SGI Origin 2000

CS 258, Spring 99
David E. Culler
Computer Science Division
U.C. Berkeley

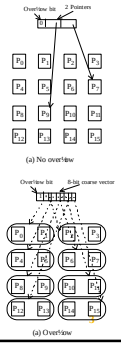
Recap



- Flat, memory-based directory schemes maintain cache state vector at block home
- Protocol realized by network transactions
- State transitions serialized through the home node
- Completion requires waiting for invalidation acks

Overflow Schemes for Limited Pointers

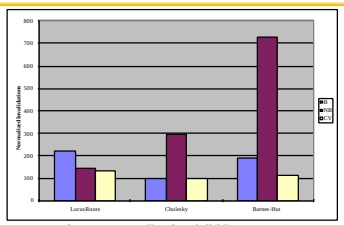
- Broadcast (Dir_iB)
  - broadcast bit turned on upon overflow
  - bad for widely-shared frequently read data
- No-broadcast (Dir_iNB)
  - on overflow, new sharer replaces one of the old ones (invalidated)
  - bad for widely read data
- Coarse vector (Dir_iCV)
  - change representation to a coarse vector, 1 bit per k nodes
  - on a write, invalidate all nodes that a bit corresponds to



Overflow Schemes (contd.)

- Software (Dir_iSW)
  - trap to software, use any number of pointers (no precision loss)
    - » MIT Alewife: 5 ptrs, plus one bit for local node
  - but extra cost of interrupt processing on software
    - » processor overhead and occupancy
    - » latency
      - 40 to 425 cycles for remote read in Alewife
      - 84 cycles for 5 inval, 707 for 6.
- Dynamic pointers (Dir_iDP)
  - use pointers from a hardware free list in portion of memory
  - manipulation done by hw assist, not sw
  - e.g. Stanford FLASH

Some Data

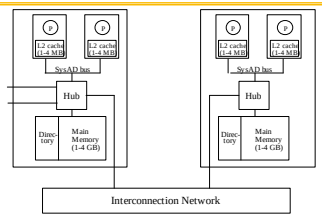


- 64 procs, 4 pointers, normalized to full-bit-vector
- Coarse vector quite robust
- General conclusions:
  - full bit vector simple and good for moderate-scale
  - several schemes should be fine for large-scale

Reducing Height: Sparse Directories

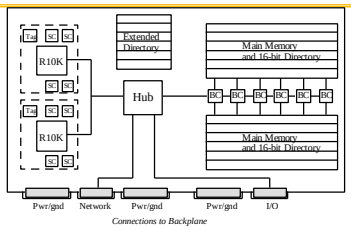
- Reduce M term in P*M
- Observation: total number of cache entries << total amount of memory.
  - most directory entries are idle most of the time
  - 1MB cache and 64MB per node => 98.5% of entries are idle
- Organize directory as a cache
  - but no need for backup store
    - » send invalidations to all sharers when entry replaced
  - one entry per "line"; no spatial locality
  - different access patterns (from many procs, but filtered)
  - allows use of SRAM, can be in critical path
  - needs high associativity, and should be large enough
- Can trade off width and height

Origin2000 System Overview



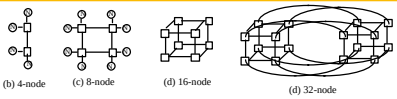
- Single 16"-by-11" PCB
- Directory state in same or separate DRAMs, accessed in parallel
- Upto 512 nodes (1024 processors)
- With 195MHz R10K processor, peak 390MFLOPS or 780 MIPS per proc
- Peak SysAD bus bw is 780MB/s, so also Hub-Mem
- Hub to router chip and to Xbow is 1.56 GB/s (both are off-board)

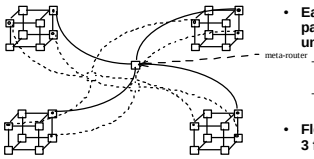
Origin Node Board



- Hub is 500K-gate in 0.5 u CMOS
- Has outstanding transaction buffers for each processor (4 each)
- Has two block transfer engines (memory copy and fill)
- Interfaces to and connects processor, memory, network and I/O
- Provides support for synch primitives, and for page migration (later)
- Two processors within node not snoopy-coherent (motivation is cost)

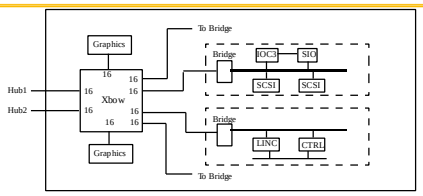
Origin Network





- Each router has six pairs of 1.56MB/s unidirectional links
  - Two to nodes, four to other routers
  - latency: 41ns pin to pin across a router
- Flexible cables up to 3 ft long
- Four "virtual channels": request, reply, other two for priority or I/O

Origin I/O



- Xbow is 8-port crossbar, connects two Hubs (nodes) to six cards
- Similar to router, but simpler so can hold 8 ports
- Except graphics, most other devices connect through bridge and bus
  - can reserve bandwidth for things like video or real-time
- Global I/O space: any proc can access any I/O device
  - through uncached memory ops to I/O space or coherent DMA
  - any I/O device can write to or read from any memory (comm thru routers)

Origin Directory Structure

- Flat, Memory based: all directory information at the home
- Three directory formats:
  - (1) if exclusive in a cache, entry is *pointer* to that specific processor (not node)
  - (2) if shared, *bit vector*: each bit points to a node (Hub), not processor
    - » invalidation sent to a Hub is broadcast to both processors in the node
    - » two sizes, depending on scale
      - 16-bit format (32 procs), kept in main memory DRAM
      - 64-bit format (128 procs), extra bits kept in extension memory
  - (3) for larger machines, *coarse vector*: each bit corresponds to p/64 nodes
    - » invalidation is sent to all Hubs in that group, which each bcast to their 2 procs
  - machine can choose between bit vector and coarse vector dynamically
    - » is application confined to a 64-node or less part of machine?

Origin Cache and Directory States

- Cache states: MESI
- Seven directory states
  - *unowned*: no cache has a copy, memory copy is valid
  - *shared*: one or more caches has a shared copy, memory is valid
  - *exclusive*: one cache (pointed to) has block in modified or exclusive state
  - three *pending* or *busy* states, one for each of the above:
    - » indicates directory has received a previous request for the block
    - » couldn't satisfy it itself, sent it to another node and is waiting
    - » cannot take another request for the block yet
  - *poisoned* state, used for efficient page migration (later)
- Let's see how it handles read and "write" requests
  - no point-to-point order assumed in network

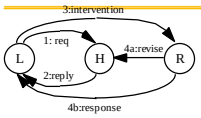
Handling a Read Miss

- Hub looks at address
  - if remote, sends request to home
  - if local, looks up directory entry and memory itself
  - directory may indicate one of many states
- Shared or Unowned State:
  - if shared, directory sets presence bit
  - if unowned, goes to exclusive state and uses pointer format
  - replies with block to requestor
    - » strict request-reply (no network transactions if home is local)
  - also looks up memory speculatively to get data, in parallel with dir
    - » directory lookup returns one cycle earlier
    - » if directory is shared or unowned, it's a win: data already obtained by Hub
    - » if not one of these, speculative memory access is wasted
- Busy state: not ready to handle
  - NACK, so as not to hold up buffer space for long

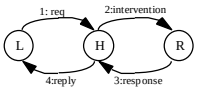
Read Miss to Block in Exclusive State

- Most interesting case
  - if owner is not home, need to get data to home and requestor from owner
  - Uses reply forwarding for lowest latency and traffic
    - » not strict request-reply

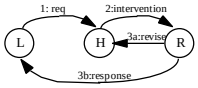
Protocol Enhancements for Latency



(a) Strict request-reply



(a) Intervention forwarding



(a) Reply forwarding

Intervention is like a req, but issued in reaction to req. and sent to cache, rather than memory.

- Problems with "intervention forwarding"
  - replies come to home (which then replies to requestor)
  - a node may have to keep track of P*k outstanding requests as home
    - » with reply forwarding only k since replies go to requestor

Actions at Home and Owner

- At the home:
  - set directory to busy state and NACK subsequent requests
    - » general philosophy of protocol
    - » can't set to shared or exclusive
    - » alternative is to buffer at home until done, but input buffer problem
  - set requestor and unset owner presence bits
  - assume block is clean-exclusive and send speculative reply
- At the owner:
  - If block is dirty
    - » send data reply to requestor, and "sharing writeback" with data to home
  - If block is clean exclusive
    - » similar, but don't send data (message to home is called "downgrade"
- Home changes state to shared when it receives revision msg

Influence of Processor on Protocol

- Why speculative replies?
  - requestor needs to wait for reply from owner anyway to know
  - no latency savings
  - could just get data from owner always
- R10000 L2 Cache Controller designed to not reply with data if clean-exclusive
  - so need to get data from home
  - wouldn't have needed speculative replies with intervention forwarding
- enables write-back optimization
  - do not need send data back to home when a clean-exclusive block is replaced
  - home will supply data (speculatively) and ask

Handling a Write Miss

- Request to home could be upgrade or read-exclusive
- State is busy: NACK
- State is unowned:
  - if RdEx, set bit, change state to dirty, reply with data
  - if Upgrade, means block has been replaced from cache and directory already notified, so upgrade is inappropriate request
    - » NACKed (will be retried as RdEx)
- State is shared or exclusive:
  - invalidations must be sent
  - use reply forwarding; i.e. invalidations acks sent to requestor, not home

Write to Block in Shared State

- At the home:
  - set directory state to exclusive and set presence bit for requestor
    - » ensures that subsequent requests willbe forwarded to requestor
  - If RdEx, send "excl. reply with invals pending" to requestor (contains data)
    - » how many sharers to expect invalidations from
  - If Upgrade, similar "upgrade ack with invals pending" reply, no data
  - Send invals to sharers, which will ack requestor
- At requestor, wait for all acks to come back before "closing" the operation
  - subsequent request for block to home is forwarded as intervention to requestor
  - for proper serialization, requestor does not handle it until all acks received for its outstanding request

Write to Block in Exclusive State

- If upgrade, not valid so NACKed
  - another write has beaten this one to the home, so requestor's data not valid
- If RdEx:
  - like read, set to busy state, set presence bit, send speculative reply
  - send invalidation to owner with identity of requestor
- At owner:
  - if block is dirty in cache
    - » send "ownership xfer" revision msg to home (no data)
    - » send response with data to requestor (overrides speculative reply)
  - if block in clean exclusive state
    - » send "ownership xfer" revision msg to home (no data)
    - » send ack to requestor (no data; got that from speculative reply)

Handling Writeback Requests

- Directory state cannot be shared or unowned
  - requestor (owner) has block dirty
  - if another request had come in to set state to shared, would have been forwarded to owner and state would be busy
- State is exclusive
  - directory state set to unowned, and ack returned
- State is busy: interesting race condition
  - busy because intervention due to request from another node (Y) has been forwarded to the node X that is doing the writeback
    - » intervention and writeback have crossed each other
  - Y's operation is already in flight and has had it's effect on directory
  - can't drop writeback (only valid copy)
  - can't NACK writeback and retry after Y's ref completes
    - » Y's cache will have valid copy while a different dirty copy is written back

Solution to Writeback Race

- Combine the two operations
- When writeback reaches directory, it changes the state
  - to shared if it was busy-shared (i.e. Y requested a read copy)
  - to exclusive if it was busy-exclusive
- Home fwds the writeback data to the requestor Y
  - sends writeback ack to X
- When X receives the intervention, it ignores it
  - knows to do this since it has an outstanding writeback for the line
- Y's operation completes when it gets the reply
- X's writeback completes when it gets writeback ack

Replacement of Shared Block

- Could send a replacement hint to the directory
  - to remove the node from the sharing list
- Can eliminate an invalidation the next time block is written
- But does not reduce traffic
  - have to send replacement hint
  - incurs the traffic at a different time
- Origin protocol does not use replacement hints
- Total transaction types:
  - coherent memory: 9 request transaction types, 6 inval/intervention, 39 reply
  - noncoherent (I/O, synch, special ops): 19 request, 14 reply (no inval/intervention)

Preserving Sequential Consistency

- R10000 is dynamically scheduled
  - allows memory operations to issue and execute out of program order
  - but ensures that they become visible and complete in order
  - doesn't satisfy sufficient conditions, but provides SC
- An interesting issue w.r.t. preserving SC
  - On a write to a shared block, requestor gets two types of replies:
    - » exclusive reply from the home, indicates write is serialized at memory
    - » invalidation acks, indicate that write has completed wrt processors
  - But microprocessor expects only one reply (as in a uniprocessor system)
    - » so replies have to be dealt with by requestor's HUB
  - To ensure SC, Hub must wait till inval acks are received before replying to proc
    - » can't reply as soon as exclusive reply is received
      - would allow later accesses from proc to complete (writes become visible) before this write

Dealing with Correctness Issues

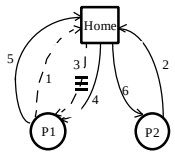
- Serialization of operations
- Deadlock
- Livelock
- Starvation

Serialization of Operations

- Need a serializing agent
  - home memory is a good candidate, since all misses go there first
- Possible Mechanism: FIFO buffering requests at the home
  - until previous requests forwarded from home have returned replies to it
  - but input buffer problem becomes acute at the home
- Possible Solutions:
  - let input buffer overflow into main memory (MIT Alewife)
  - don't buffer at home, but forward to the owner node (Stanford DASH)
    - » serialization determined by home when clean, by owner when exclusive
    - » if cannot be satisfied at "owner", e.g. written back or ownership given up, NACKed bak to requestor without being serialized
      - serialized when retried
  - don't buffer at home, use busy state to NACK (Origin)
    - » serialization order is that in which requests are accepted (not NACKed)
  - maintain the FIFO buffer in a distributed way (SCI)

Serialization to a Location (contd)

- Having single entity determine order is not enough
  - it may not know when all xactions for that operation are done everywhere



1. P1 issues read request to home node for A
2. P2 issues read-exclusive request to home corresponding to write of A. But won't process it until it is done with read
3. Home recieves 1, and in response sends reply to P1 (and sets directory presence bit). Home now thinks read is complete. Unfortunately, the reply does not get to P1 right away.
4. In response to 2, home sends invalidate to P1; it reaches P1 before transaction 3 (no point-to-point order among requests and replies).
5. P1 receives and applies invalideate, sends ack to home.
6. Home sends data reply to P2 corresponding to request 2.

Finally, transaction 3 (read reply) reaches P1.

- Home deals with write access before prev. is fully done
- P1 should not allow new access to line until old one "done"

Deadlock

- Two networks not enough when protocol not request-reply
  - Additional networks expensive and underutilized
- Use two, but detect potential deadlock and circumvent
  - e.g. when input request and output request buffers fill more than a threshold, and request at head of input queue is one that generates more requests
  - or when output request buffer is full and has had no relief for T cycles
- Two major techniques:
  - take requests out of queue and NACK them, until the one at head will not generate further requests or ouput request queue has eased up (DASH)
  - fall back to strict request-reply (Origin)
    - » instead of NACK, send a reply saying to request directly from owner
    - » better because NACKs can lead to many retries, and even livelock

Livelock

- Classical problem of two processors trying to write a block
  - Origin solves with busy states and NACKs
    - » first to get there makes progress, others are NACKed
- Problem with NACKs
  - useful for resolving race conditions (as above)
  - Not so good when used to ease contention in deadlock-prone situations
    - » can cause livelock
    - » e.g. DASH NACKs may cause all requests to be retried immediately, regenerating problem continually
      - DASH implementation avoids by using a large enough input buffer
- No livelock when backing off to strict request-reply

Starvation

- Not a problem with FIFO buffering
  - but has earlier problems
- Distributed FIFO list (see SCI later)
- NACKs can cause starvation
- Possible solutions:
  - do nothing; starvation shouldn't happen often (DASH)
  - random delay between request retries
  - priorities (Origin)

Support for Automatic Page Migration

- Misses to remote home consume BW and incur latency
- Directory entry has 64 miss counters
  - trap when threshold exceeded and remap page
- problem: TLBs everywhere may contain old virtual to physical mapping
  - explicit shootdown expensive
- set directly entries in old page (old PA) to poison
  - nodes trap on access to old page and rebuild mapping
  - lazy shootdown

Synchronization

- R10000 load-locked / store conditional
- Hub provides uncached fetch&op

Back-to-back Latencies (unowned)

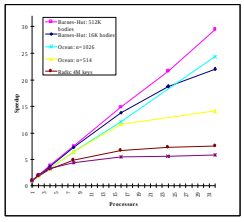
Satisfied in	back-to-back latency (ns)	hops
L1 cache	5.5	0
L2 cache	56.9	0
local mem	472	0
4P mem	690	1
8P mem	890	2
16P mem	990	3

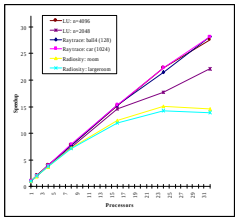
- measured by pointer chasing since ooo processor

Protocol latencies

Home	Owner	Unowned	Clean-Exclusive	Modified
Local	Local	472	707	1,036
Remote	Local	704	930	1,272
Local	Remote	472*	930	1,159
Remote	Remote	704*	917	1,097

Application Speedups





Summary

- In directory protocol there is substantial implementation complexity below the logical state diagram
  - directory vs cache states
  - transient states
  - race conditions
  - conditional actions
  - speculation
- Real systems reflect interplay of design issues at several levels
- Origin philosophy:
  - memory-less: node reacts to incoming events using only local state
  - an operation does not hold shared resources while requesting others