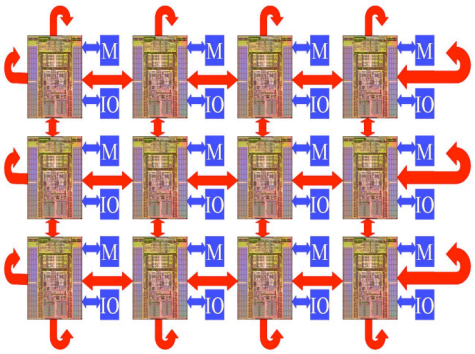
18-742

Lecture 14

Distributed Shared Memory



Spring 2005

Prof. Babak Falsafi

http://www.ece.cmu.edu/~ece742

Slides developed in part by Profs. Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith, and Singh of University of Illinois, Carnegie Mellon University, University of Wisconsin, Duke University, University of Michigan, and Princeton University.

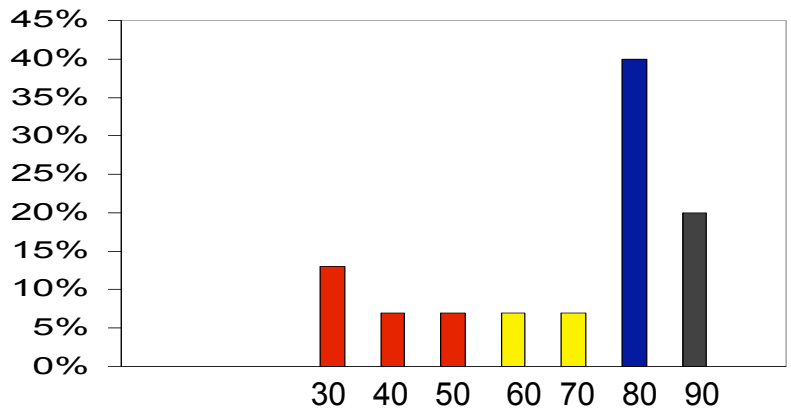
Readings

Chapter 8 of Culler & Singh

Reader 5:

- **D. V. James, A. T. Laundrie, S. Gjessing, and G. S. Sohi, *Distributed-Directory Scheme: Scalable Coherent Interface*, IEEE Computer 23(6): 74-77, June 1990.**
- **Chaiken et al., *Directory-Based Cache Coherence Protocols for Large-Scale Multiprocessors*, IEEE Computer, 19-58, June 1990.**

Announcements



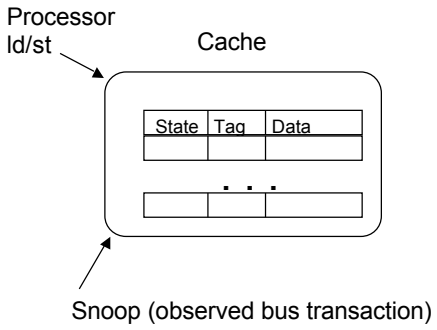
- **Mean = 74**
- **Below 60, please stop by and talk to me**

Outline

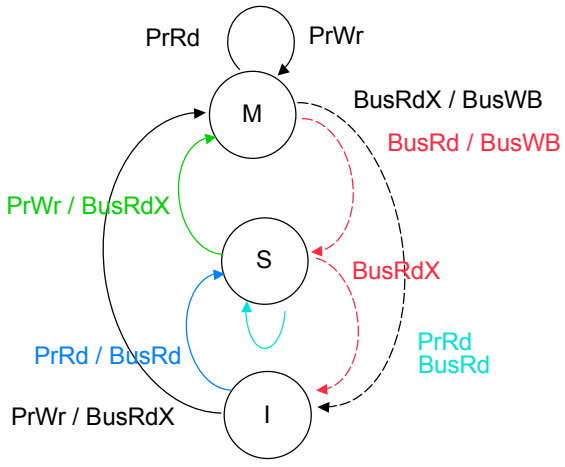
- **Directory-Based Cache Coherence**
  - **Review**
  - **Basic Idea**
  - **Some Variations**
- **SGI Origin 2000 Case Study**
- **Memory Consistency Models Revisited**

Review: Snooping Coherence

- **Controller updates state of blocks in response to processor and snoop events and generates bus xactions**
- **Often have duplicate cache tags**
- **Snoopy protocol**
  - **set of states**
  - **state-transition diagram**
  - **actions**
- **Basic Choices**
  - **write-through vs. write-back**
  - **invalidate vs. update**

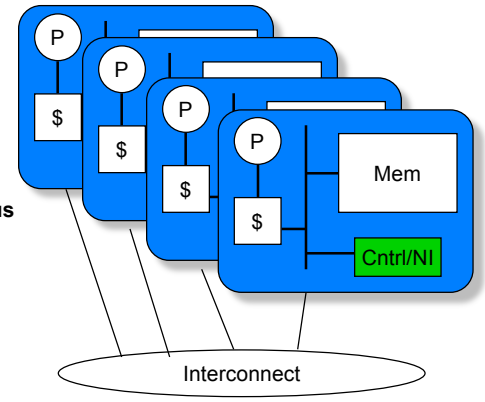


Review: MSI State Diagram



Large Scale Shared Memory Multiprocessors

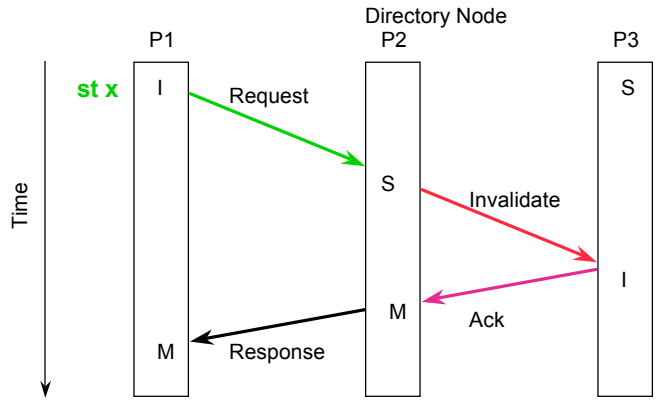
- 100s to 1000s of nodes (processors) with single shared physical address space
- Use General Purpose Interconnection Network
  - Still have cache coherence protocol
  - Use messages instead of bus transactions
  - No hardware broadcast
- Communication Assist



Directory Based Cache Coherence

- Avoid broadcast request to all nodes on a miss
  - traffic
  - time
- Maintain directory of which nodes have cached copies of the block (directory controller + directory state)
- On a miss, send message to directory
  - communication assist
- Directory determines what (if any) protocol action is required
  - e.g., invalidation
- Directory waits for protocol actions to finish and then responds to the original request

Directory Example



Centralized Directory

- **Single directory that contains a copy of all nodes cache tags**

Problems

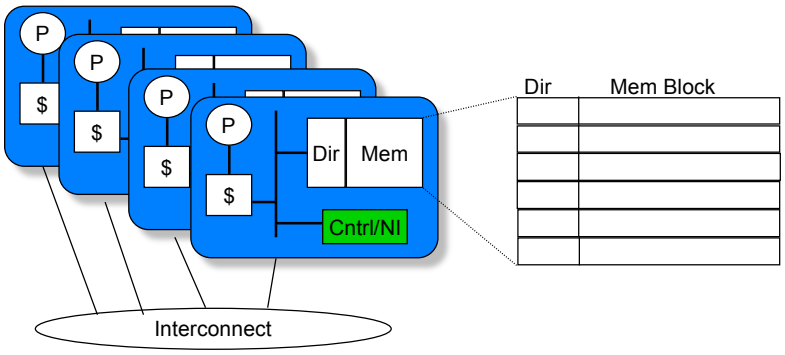
- **Bottleneck (1000s of processors…)**
- **Directory changes with number of nodes**

Positives

- **Send Invalidates/Updates only to nodes that have copy of block**

Distributed Directory

- Distribute Directory among memory modules
- Maintain directory for each memory block
  - memory block = coherence block size: block's home node = node with directory



Directory Nomenclature

- Dir_iX
- Directory of *i* pointers (*i* <= Total number of nodes)
- X specifies what to do on Shared to Modified transition
  - B => Broadcast
  - NB => No Broadcast
  - SW => Software
- Dir_N = full-map directory
  - Bit vector per memory block
  - Bit per node in system
  - No need to broadcast

Coarse Vector and Sparse Directories

Coarse Vector

- Instead of full-map or broadcast, indicate a set of nodes that may have the block
- Reduces space requirements
- Many applications have near neighbor sharing

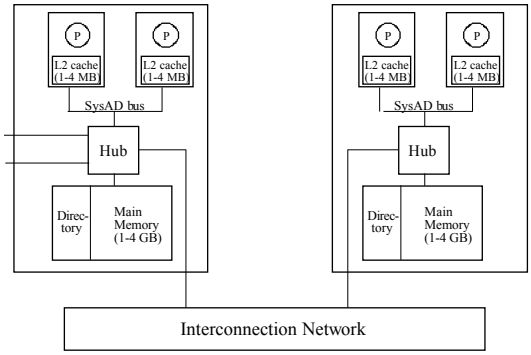
Sparse

- Not all of memory is in processor caches
- Cache of directory entries at memory

Outline

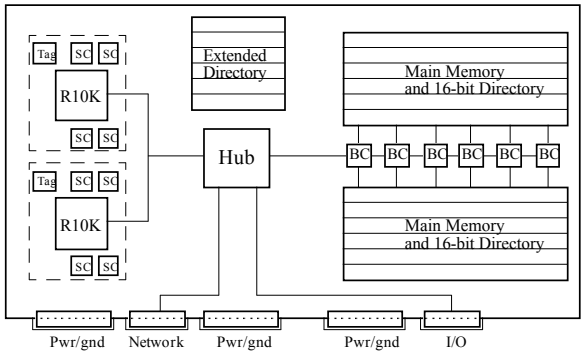
- Directory-Based Cache Coherence
- SGI Origin 2000 Case Study
  - Overview
  - Directory & Protocol States
  - Detailed Coherence Protocol Examples
- Memory Consistency Models Revisited

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/780 MIPS**
- **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 of-board)**

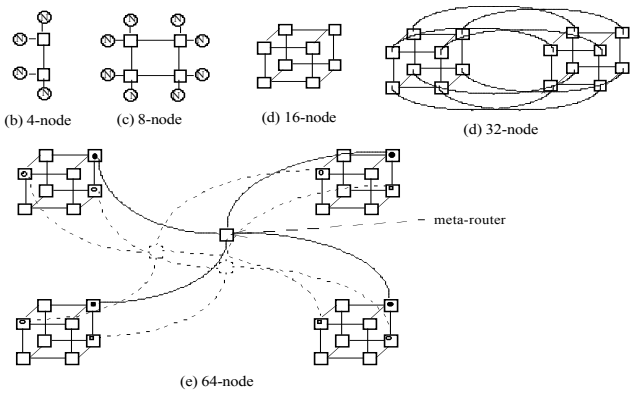
Origin Node Board



Connections to Backplane

- **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**
- **Two processors within node not snoopy-coherent (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, two for priority or I/O

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
- Ignore coarse vector in discussion for simplicity

Origin Cache and Directory States

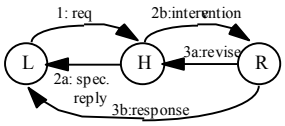
- Cache states: MESI
- Seven directory states
  - *unowned*: no cache has a copy, memory copy is valid
  - *shared*: one or more caches has a shared copy, memory is valid
  - *exclusive*: one cache (pointed to) has block in modified or exclusive state
  - three *pending* or *busy* states, one for each 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)
  - actually, also looks up memory speculatively to get data
    - » directory lookup returns one cycle earlier
    - » if directory is shared or unowned, 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**



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 and unset appropriate 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 msg**

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 forwards 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 the 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)
    - » 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