Outline

- **Directory-Based Cache Coherence**
- **Stanford DASH Case Study**
- **SGI Origin Case Study**
- **Advanced Directory Systems**

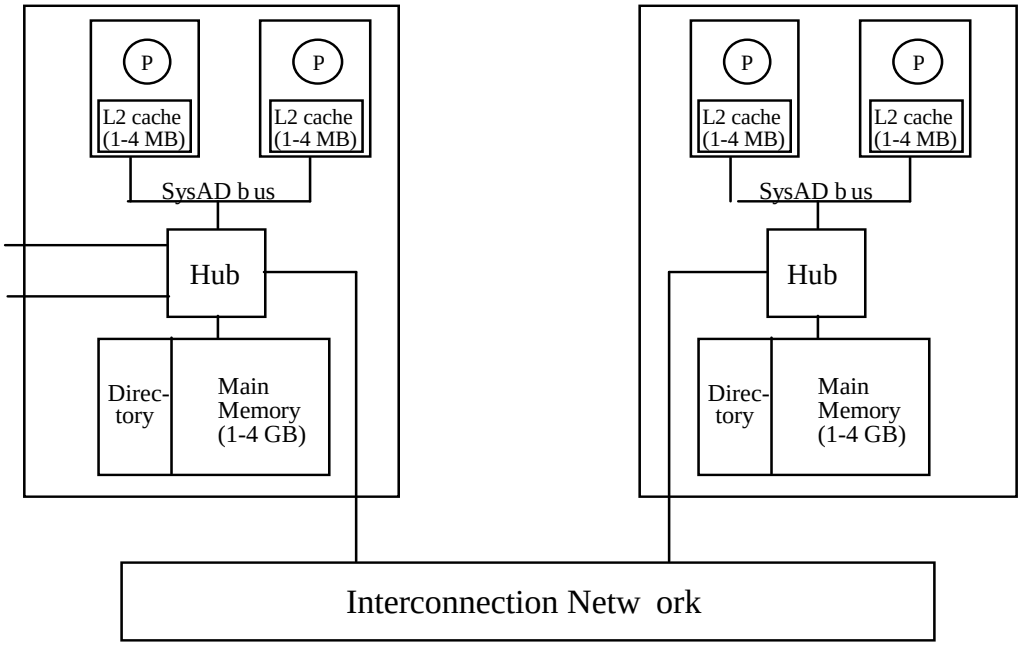
DASH

- First system with directory-based cache coherence
- Academic design (Stanford) that led to SGI Origin
- Also had follow-on at Stanford called FLASH
- DISCUSS DASH PAPER

Outline

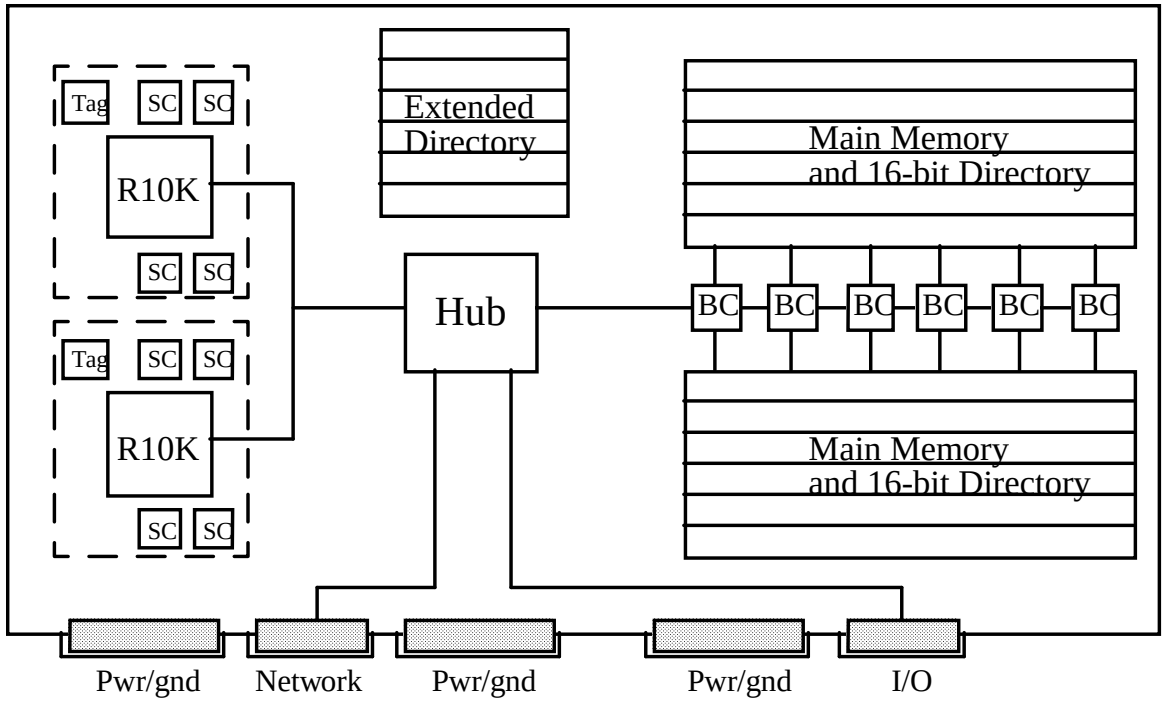
- **Directory-Based Cache Coherence**
- **Stanford DASH Case Study**
- **SGI Origin Case Study**
  - **Overview**
  - **Directory & Protocol States**
  - **Detailed Coherence Protocol Examples**
- **Advanced Directory Systems**

Origin2000 System Overview



- **Single 16”-by-11” PCB (except for Xbow I/O)**
- **Directory state in same or separate DRAMs, accessed in parallel**
- **Up to 512 nodes (1024 processors)**
- **With 195MHz R10K processor, peak 390MFLOPS/780 MIPS**
- **Peak SysAD bus b/w is 780 MB/s, same for Hub to Mem b/w**
- **Hub to router chip and to Xbow is 1.56 GB/s (both are off-board)**

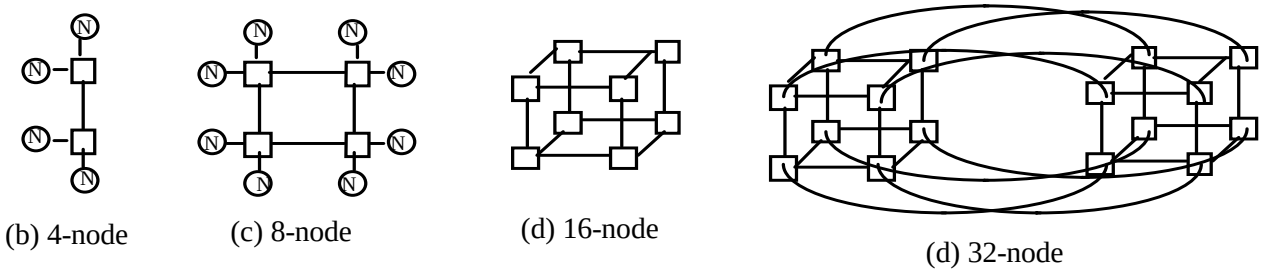
Origin Node Board



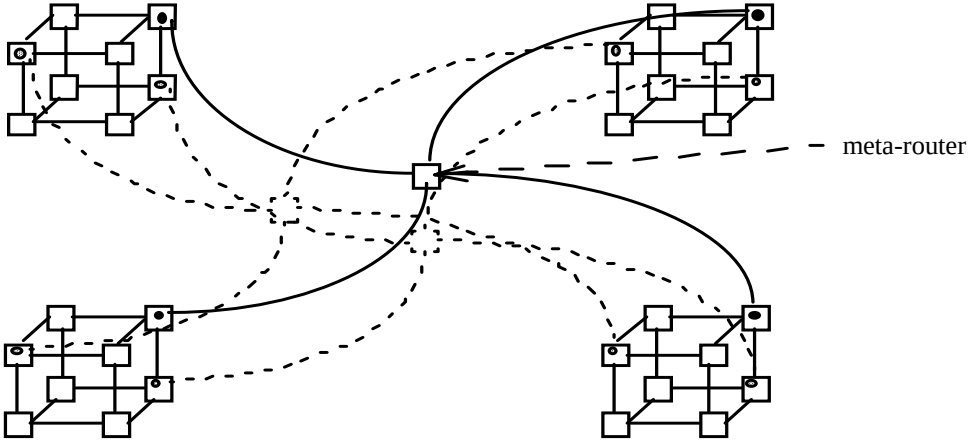
Connections to Backplane

- **Hub is 500K-gate in 0.5 um 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



(b) 4-node (c) 8-node (d) 16-node (d) 32-node



(e) 64-node

- **Each router has six pairs of 1.56MB/s unidirectional links**
  - **Two to nodes, up to 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 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 (like Illinois snooping protocol)**
- **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 → lots of races!**

Races in the Protocol

- **Without point-to-point ordering in the network, messages can bypass each other and arrive at unexpected times**

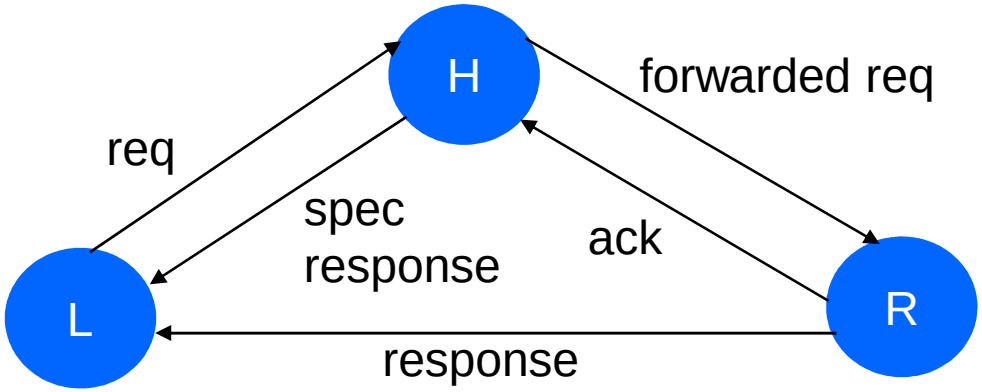
- **Example (all messages involve block B)**
  - **Initially: all caches in Invalid, directory in unowned**
  - **P1 sends GETX to Dir**
  - **Dir receives P1's GETX, responds with data (msg gets delayed)**
  - **P2 sends GETX to Dir**
  - **Dir forwards P2's GETX to P1**
  - **P1 receives Forwarded-GETX … while in state Invalid!**

(1) Handling a Read Miss

- **Hub looks at address**
  - **If remote, sends request to home directory**
  - **If local, looks up directory entry and memory itself**
- **Directory may indicate one of many states**
- **If *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**
    - » **If directory is shared or unowned, data already obtained by Hub**
    - » **If not one of these, speculative memory access is wasted**
- **If 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 is read miss to Exclusive block**
  - **If owner is not home, need to transfer the data from owner to both requestor and home (why to home?)**
  - **Uses reply forwarding for lowest latency and traffic**
    - » **Not strict request-reply (think about deadlock issues …)**
  - **Note: home doesn't know if remote node is in E (unowned!) or M**
    - » **Must speculatively send response to requestor (if in E)**



Actions at Home & Owner

- **At the home:**
  - **Set directory to Busy-Exclusive and NACK subsequent requests**
    - » **General philosophy of protocol (unlike GS320 or Piranha)**
    - » **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 send "sharing writeback" (aka "copyback") with data to home**
  - **If block is clean exclusive**
    - » **Similar, but don't send data (msg to home is called "downgrade")**
- **Home changes state to shared when it receives msg**

(2) Handling a Write Miss

- **Request to home could be upgrade or read-exclusive**
- **If state is busy: NACK**
- **If 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)**
- **If state is shared or exclusive:**
  - **Invalidations must be sent**
  - **Use reply forwarding; i.e. invalidation 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 will be forwarded to requestor**
  - **If RdEx, send "excl. reply with invals pending" to requestor (w/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)

(3) 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 its 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

(4) 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)

Outline

- **Directory-Based Cache Coherence**
- **Stanford DASH Case Study**
- **SGI Origin 2000 Case Study**
- **Advanced Directory Systems**
  - **AlphaServer GS320**
  - **Compaq Piranha**

AlphaServer GS320

- **PRESENTATION**

Compaq Piranha

- One of the first multicores
- Prototype from Compaq
  - Simple cores
  - Directory protocol
  - Goal? Throughput!

Outline

- **Directory-Based Cache Coherence**
- **Stanford DASH Case Study**
- **SGI Origin 2000 Case Study**
- **Advanced Directory Systems**