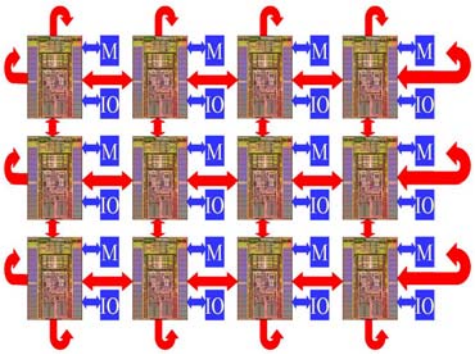
18-742

Lecture 15

COMA & Simple COMA



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

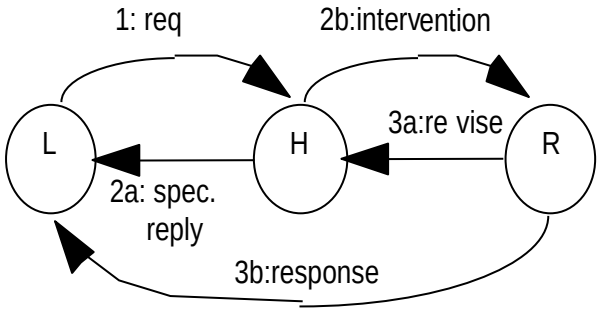
Reader 5:

- **J. Laudon and D. Lenoski, *The SGI Origin: A ccNUMA Highly Scalable Server*, ISCA 1997.**
- **Erik Hagersten, Anders Landin, and Seif Haridi, *DDM--A Cache Only Memory Architecture*, IEEE Computer, Sep. 1992.**
- **Babak Falsafi and David Wood, *Reactive NUMA: A Design to Unify S-COMA and CC-NUMA*, ISCA,1997.**

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

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

Review

- Recall Part 1
  - Directory-Based Cache Coherence
  - SGI Origin 2000 Case Study
- Basic Idea
  - Per-processor cache hierarchies
  - Directory interleaved with memory
- But
  - Limited capacity for replication
  - High design & implementation cost
  - Single hard-wired protocol
  - Limitations of shared physical address space

Outline

- Cache-Only Memory Architecture (COMA)
- Paged-Based Distributed Shared Memory
- Simple-COMA (S-COMA)
- Hierarchical Coherence
- Latency Tolerance

Cache Only Memory Architecture (COMA)

- Make all memory available for migration & replication
- All memory is DRAM cache called Attraction Memory
- Examples
  - Data Diffusion Machine (next)
  - Flat COMA (fixed home for directory but not data)
  - KSR-1 (hierarchy of snooping rings)
- But how do you
  - Find data?
  - Deal with replacements?

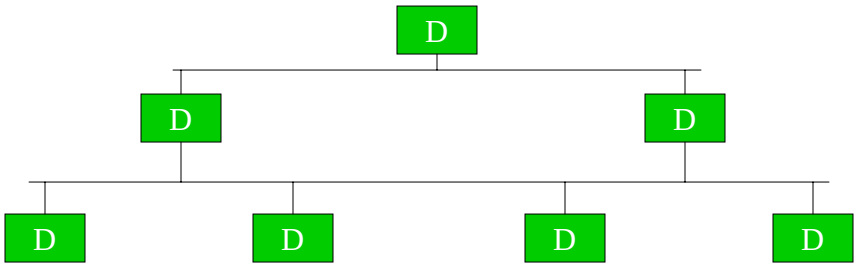
COMA E.g.: Data Diffusion Machine (DDM)

- All hardware COMA
- Attraction Memory => One giant hardware cache
- Maintains both address tags and state
- Data addressed, allocated, & kept coherent in blocks
- Directory info on a per cache-block basis
- Not Home Based:
  - data is migratory => AM attracts data
  - must find a home when replacing the data
  - must find the directory entry before finding the data

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

DDM Directory

- Directory is hierarchical in a tree form
- Each is a set-associative cache if directory info
- Tree maintains inclusion:
- Higher levels keep replica of lower sub-trees



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

DDM Coherence/Placement Protocol

- Simple write-invalidate protocol
- Cache states: Invalid, Exclusive, Shared
- Must traverse the directory:
  - to find a copy on a read or write miss
  - to invalidate on a write to Shared
- Directory is hierarchical set-associative caches
  - Q1: Is the block in my sub-tree?
  - Q2: Does the block exist outside my sub-tree?
  - Request goes up until Q2==no and then down
  - Request goes down until Q1=no or leaf
- On a replacement:
  - for an Exclusive copy, must find another home (HARD!)
  - for a Shared copy, must make sure other copies exist
  - else must find another home

Outline

- Cache-Only Memory Architecture (COMA)
- Paged-Based Distributed Shared Memory
- Simple-COMA (S-COMA)
- Hierarchical Coherence
- Latency Tolerance

Page Based DSM (Shared Virtual Memory)

- **Forget all this hardware!**
- **Implemented shared virtual address space**
  - **On separate computers networked together**
  - **Use virtual memory system to do coherence on pages**
  - **No special hardware; no shared physical address space**
- **Called**
  - **Shared Virtual Memory (SVM) in original paper [Li & Hudak]**
  - **Now called Page-Based (or Software) Distributed Shared Memory**

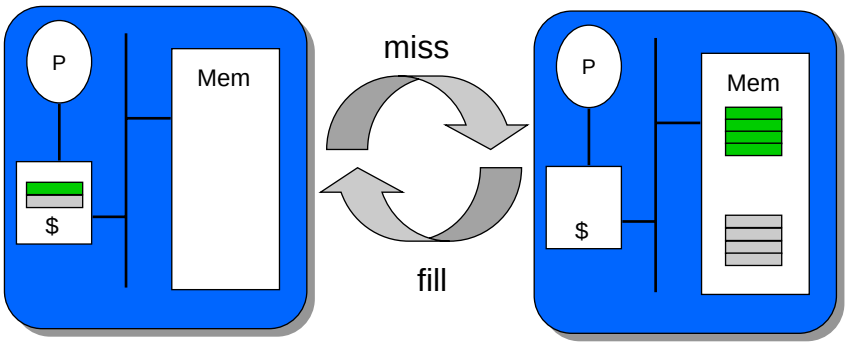
Example

- **P1 read virtual address x**
- **Page fault**
- **Allocate physical frame for page(x)**
- **Request page(x) from home(x)**
- **Set readable page(x)**
- **Resume program**
- **Problems**
  - **False-sharing**
  - **Page fault overhead**
- **Advantages**
  - **Software Coherence protocol**
  - **Low cost**

Simple COMA (S-COMA)

- **COMA**
  - **Block granularity to find/allocate/replace (complex hardware)**
  - **Block granularity for coherence/transfers (good for false sharing)**
- **Software DSM**
  - **Page granularity to find/allocate/replace (use VM: good)**
  - **Page granularity for coherence/transfers (bad for false sharing)**
- **Simple COMA**
  - **Page granularity to find/allocate/replace (use VM: good)**
  - **Block granularity for coherence/transfers (good for false sharing)**
  - **Blocks act like sub-blocks on page**

Cache-Coherent Non-Uniform Memory Architecture (CC-NUMA)

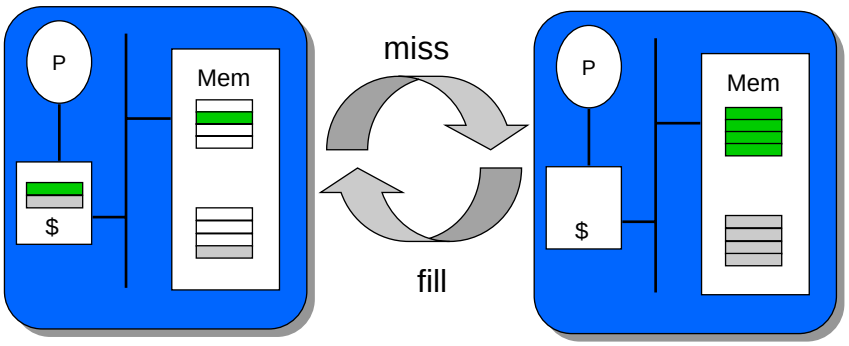


✓ **Fast allocation of block frames**

X **Limited storage capacity**

- **frequent capacity/conflict misses for data with long-term "reuse"**

Simple COMA Memory Architecture (S-COMA)



✓ **Handles capacity/conflict misses locally**

X **Large DRAM page allocation overhead (VM)**

X **Large fragmentation**

Why Reactive NUMA (R-NUMA)?

Remote data behave in two ways

1. **Reuse pages:**
- **Frequently accessed data structures**
- **Incur capacity/conflict misses**

→ **Favor S-COMA**

2. **Communication pages:**
- **Exchange producer/consumer data**
- **Incur coherence misses**

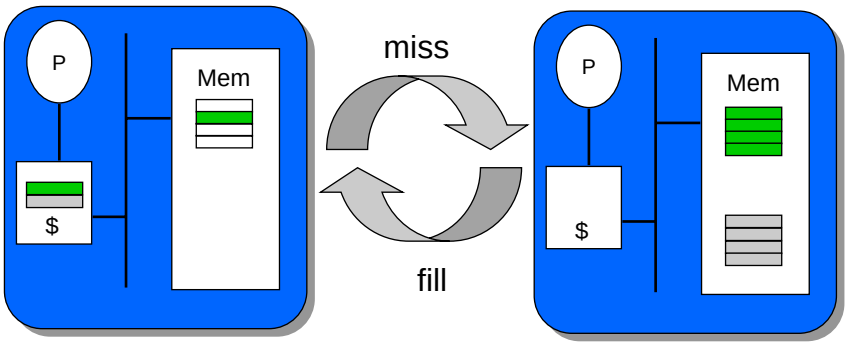
→ **Favor CC-NUMA**

What is Reactive NUMA (R-NUMA)?

Dynamically selects between CC-NUMA & S-COMA

- **Initially selects a page to be CC-NUMA**
- **Uses "reuse counters" per page**
  - **Count capacity/conflict misses**
  - **How can it tell between these and coherence misses?**
- **When count reaches threshold, switches to S-COMA**
  - **When does it switch back?**

Miss Handling in R-NUMA



- **Initially both the green and gray pages are CC-NUMA**
- **Eventually the green page moves to S-COMA**
  - **Gray (communication), green (reuse)**

R-NUMA's Performance

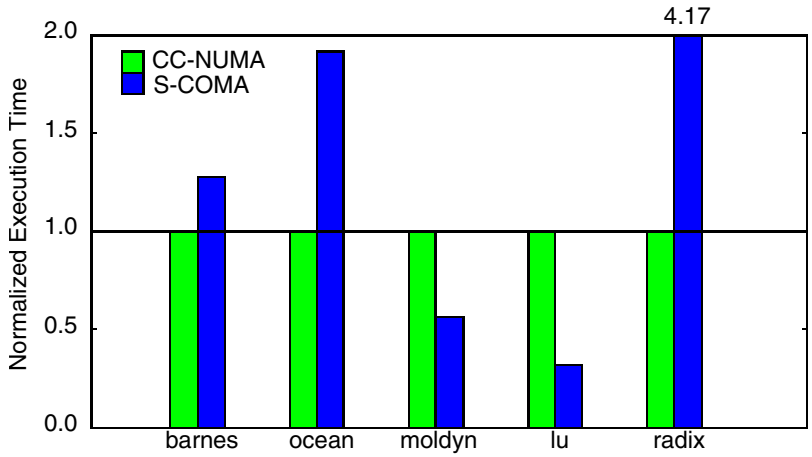
Performance depends on

- How fast and how accurately it can detect reuse pages
- How fast it can switch between CC-NUMA & S-COMA

Analytical worst-case bound:

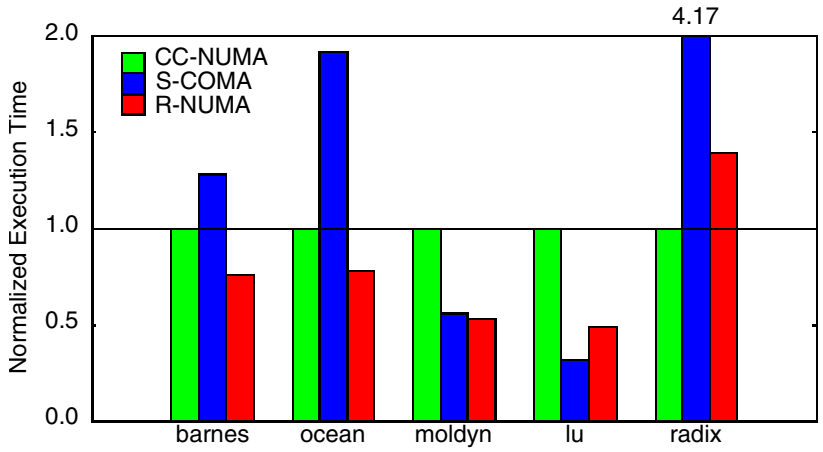
- Large CC-NUMA & S-COMA performance gap (> 10)
- R-NUMA closes performance gap to within 3 of best

Base Performance Comparison



Large performance gap between the two

Performance Compared to R-NUMA



- **R-NUMA often outperforms both**

Sun Wildfire

- **[Hagersten/Koster HPCA99]**
- **Begin with up to four SMP nodes**
- **Add pseudo-processor board to each as proxy for rest of system**
- **Uses R-NUMA (calls it Coherent Memory Replication)**
- **A hierarchical methods of building parallel machines**