

## ECE666: Advanced Computer Systems (Really... Parallel Computer Architecture)

Instructor: Mithuna Thottethodi

Spring 2007  
Course webpage:  
<http://www.ece.purdue.edu/~mithuna/ece666>  
Course newsgroup: `purdue.class.ece666`

## Post Mid-term Outline

- Post-midterm outline
  - Scalable parallel machines
  - Directory Coherence
  - HW/SW tradeoffs
  - Interconnection Networks
  - Advanced Topics
    - Transactions
  - Last week: Project presentations
    - Two per lecture. 20 mins presentation + 15 mins of Q&A

ECE666, Spring 2007

(2)

© Mithuna Thottethodi, 2005,2007

## Context

- Recall Part 1
  - Directory-Based Cache Coherence
  - SGI Origin Case Study
  - Memory Consistency Models Revisited
- 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

ECE666, Spring 2007

(3)

© Mithuna Thottethodi, 2005,2007

## Outline

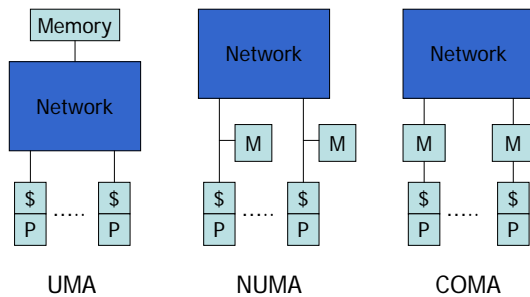
- Nomenclature UMA vs. NUMA vs CC-NUMA
- Cache-Only Memory Architecture (COMA)
- Paged-Based Distributed Shared Memory
- Simple-COMA (S-COMA)
- Hierarchical Coherence

ECE666, Spring 2007

(4)

© Mithuna Thottethodi, 2005,2007

## Nomenclature



ECE666, Spring 2007

(5)

© Mithuna Thottethodi, 2005,2007

## 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?

ECE666, Spring 2007

(6)

© Mithuna Thottethodi, 2005,2007

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

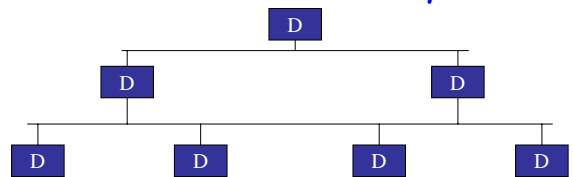
- All hardware COMA
- **Attraction Memory**
  - One giant hardware cache
- Maintains both address tags and state
- Data addressed, allocated, & kept coherent in blocks
- Directory info on a per cache-block basis
- **Not Home Based:**
  - data is migratory => AM attracts data
  - must find a home when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

ECE666, Spring 2007

(7)

© Mithuna Thottethodi, 2005,2007

## DDM Directory



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

ECE666, Spring 2007

(8)

© Mithuna Thottethodi, 2005,2007

## 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

ECE666, Spring 2007

(9)

© Mithuna Thottethodi, 2005,2007

## Outline

- Nomenclature UMA vs. NUMA vs CC-NUMA
- Cache-Only Memory Architecture (COMA)
- Paged-Based Distributed Shared Memory
- Simple-COMA (S-COMA)
- Hierarchical Coherence

ECE666, Spring 2007

(10)

© Mithuna Thottethodi, 2005,2007

## 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

ECE666, Spring 2007

(11)

© Mithuna Thottethodi, 2005,2007

## Example

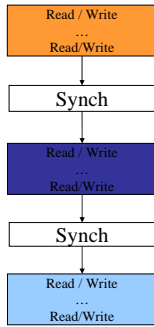
- P1 read virtual address x
- Page fault
- Allocate physical frame for page(x)
- Request page(x) from home(x)
- Set readable page(x)
- Resume program
- Problems
  - False-sharing
  - Page fault overhead
- Advantages
  - Software Coherence protocol
  - Low cost

ECE666, Spring 2007

(12)

© Mithuna Thottethodi, 2005,2007

## Recall: Weak Ordering

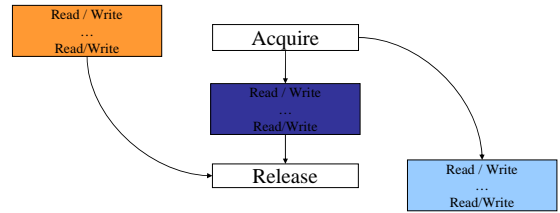


ECE666, Spring 2007

(13)

© Mithuna Thottethodi, 2005,2007

## Recall: Release Consistency

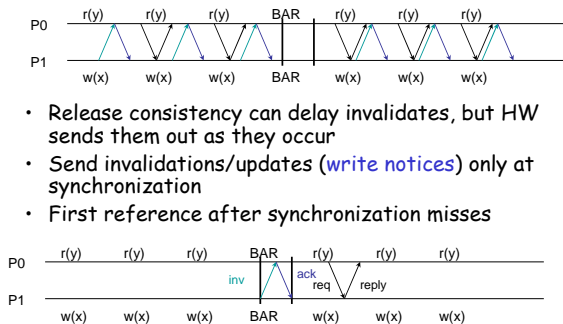


ECE666, Spring 2007

(14)

© Mithuna Thottethodi, 2005,2007

## Relaxed Consistency



ECE666, Spring 2007

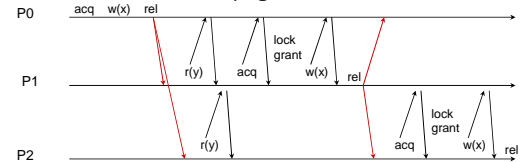
(15)

© Mithuna Thottethodi, 2005,2007

## Eager Release Consistency

- Propagate write notices at release operations

- X & Y on same page



Sends invalidations to too many processors  
 Separate messages for invalidation and acquire  
 Invalidate pages too early increasing page faults

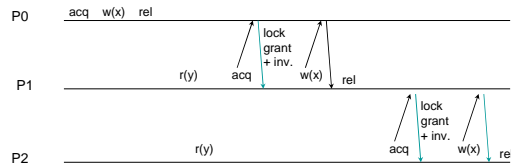
ECE666, Spring 2007

(16)

© Mithuna Thottethodi, 2005,2007

## Lazy Release Consistency (TreadMarks)

- Piggyback write notices with acquire operations
  - X & Y on same page



- + fewer messages
- what about false sharing with multiple writes to same page?

ECE666, Spring 2007

(17)

© Mithuna Thottethodi, 2005,2007

## LRC vs. RC

- Changes in behavior
- Difficult to port existing programs
  - Fence operations
    - Not recognized by the SVM layer
  - Synchronization operations

ECE666, Spring 2007

(18)

© Mithuna Thottethodi, 2005,2007

## Multiple Writer Protocol

- $x$  &  $y$  on same page P1 writes  $x$ , P2 writes  $y$
- Don't want delays associated with constraint of exclusive access
- Allow each processor to modify its local copy of a page between synchronization points
- Make things consistent at synchronization point

ECE666, Spring 2007

(19)

© Mithuna Thottethodi, 2005,2007

## Implementing a multiple-writer protocol

- Need to get "latest" data from each processor
- Exploit mutual exclusion, to capture modifications made by one processor
- Create twin (copy) of unmodified page, compute diff to send w/ write notice
  - **eager**: send at release
  - **lazy**: send with response to acquire
- Multiple-Writer in Lazy Release Consistency
  - which diffs (vector timestamp)?
  - which nodes diffs come from?

ECE666, Spring 2007

(20)

© Mithuna Thottethodi, 2005,2007

## Outline

- Nomenclature UMA vs. NUMA vs CC-NUMA
- Cache-Only Memory Architecture (COMA)
- Paged-Based Distributed Shared Memory
- Simple-COMA (S-COMA)
- Hierarchical Coherence

ECE666, Spring 2007

(21)

© Mithuna Thottethodi, 2005,2007

## 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

ECE666, Spring 2007

(22)

© Mithuna Thottethodi, 2005,2007

## S-COMA-like Examples

- Wisconsin Typhoon [Reinhardt et al. ISCA 1994]
  - On access VM system check if page present
  - On access HW/SW checks block state
  - Failure invokes a user-level protocol in SW
  - Wonderful flexibility, but SW slow & users don't want to write protocols
- Sun Wildfire [Hagersten/Koster HPCA99]
  - Begin with up to four SMP nodes
  - Add pseudo-processor board to each as proxy for rest of system
  - Can run CC-NUMA directory protocol
  - Can selectively use S-COMA (called Coherent Memory Replication)
  - Selects between with competitive algorithm [Falsafi/Wood ISCA97]
  - A hierarchical method of building parallel machines

ECE666, Spring 2007

(23)

© Mithuna Thottethodi, 2005,2007

## A Taxonomy of Issues

- Allocation/Replication
  - cache line vs page
- Access Control (Coherence)
  - cache line vs page
  - HW vs SW
- Protocol Processing
  - HW vs SW
- Communication
  - cache line vs page
  - HW vs SW (message passing)

ECE666, Spring 2007

(24)

© Mithuna Thottethodi, 2005,2007

## Papers to Read

- Treadmarks
- R-NUMA
  - Acknowledgement
    - Some slides created by past ECE 666 students

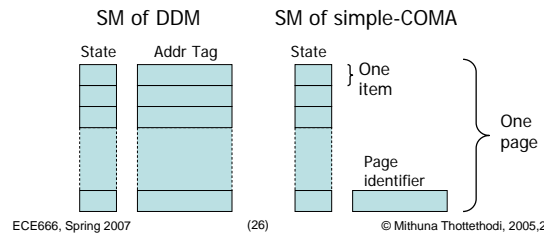
ECE666, Spring 2007

(25)

© Mithuna Thottethodi, 2005,2007

## Simple COMA

- *Page* for allocate and replace (MMU + SW).
- *Block* for Coherence (Hardware).



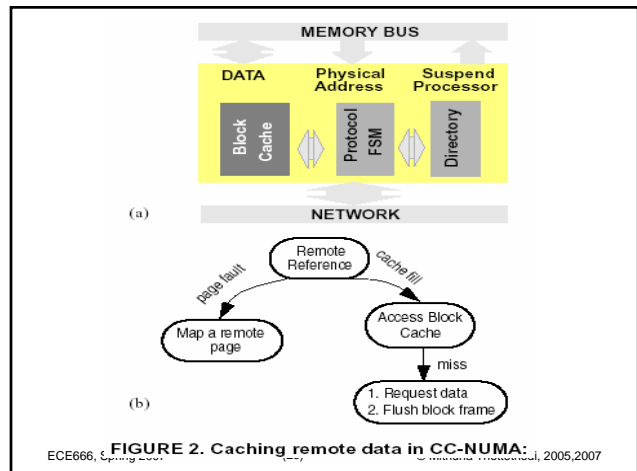
## Best of both worlds

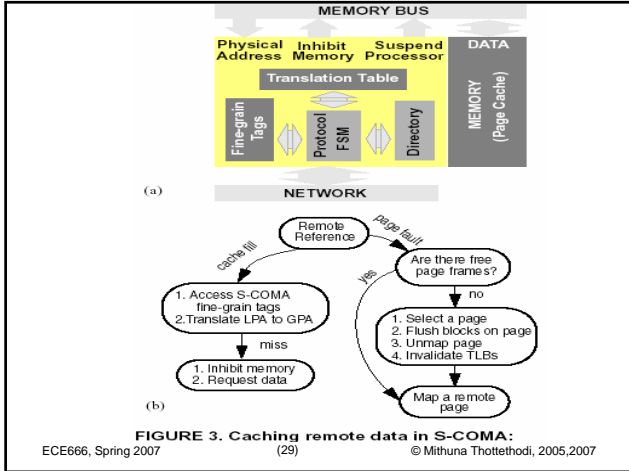
- CC-NUMA (DSM-Distributed Shared Memory)
  - +: keeps the local data close to the processor
  - : ( small cache) high miss rate for remote access
- S-COMA (Physically DSM, Logically SMP)
  - +: (large memory) low remote access miss rate
  - : sparse pages, page alloc. overheads

ECE666, Spring 2007

(27)

© Mithuna Thottethodi, 2005,2007

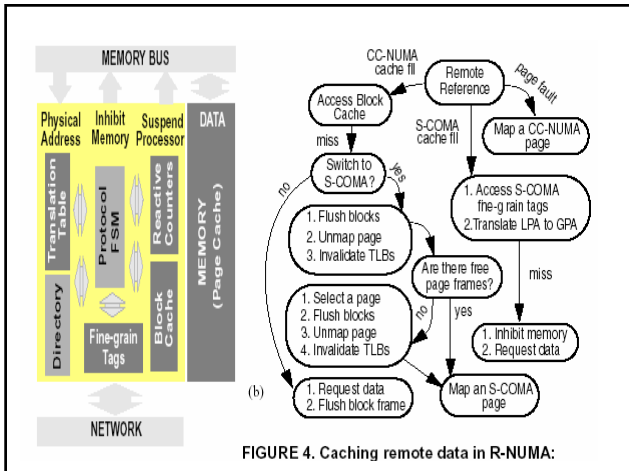




## Mechanisms for R-NUMA

- Reuse page (S-COMA)  $\leftrightarrow$  Communication Page (CC-NUMA)
- The switching point: preset threshold value.
- Combines the RAD hardware for CC-NUMA and S-COMA
  - Common: Protocol FSM, Directory
  - CC-NUMA: Block Cache
  - S-COMA: Page Cache, Fine-grain Tags
  - Additional: Reactive Counter Per Node Per Page

ECE666, Spring 2007 (30) © Mithuna Thottethodi, 2005,2007

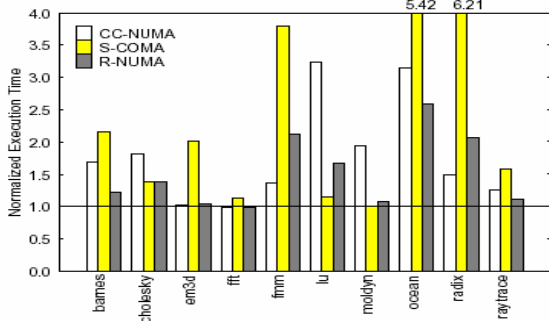


## Qualitative WC Analysis of RNUMA

- Worst Case R-NUMA vs. CC-NUMA
 
$$\frac{O_{R-NUMA}}{O_{CC-NUMA}} = \frac{TC_{refetch} + C_{relocate} + C_{allocate}}{TC_{refetch}} \quad (EQ 1)$$
- Worst Case R-NUMA vs. S-COMA
 
$$\frac{O_{R-NUMA}}{O_{S-COMA}} = \frac{TC_{refetch} + C_{relocate} + C_{allocate}}{C_{allocate}} \quad (EQ 2)$$
- Resulting Threshold Value T
 
$$\frac{O_{R-NUMA}}{O_{CC-NUMA}} = \frac{O_{R-NUMA}}{O_{S-COMA}} = 2 + \frac{C_{relocate}}{C_{allocate}} \quad (EQ 3)$$

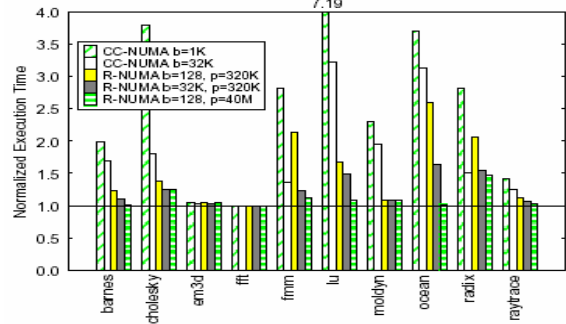
ECE666, Spring 2007 (32) © Mithuna Thottethodi, 2005,2007

### CC-NUMA, S-COMA, and R-NUMA Performance Comparison



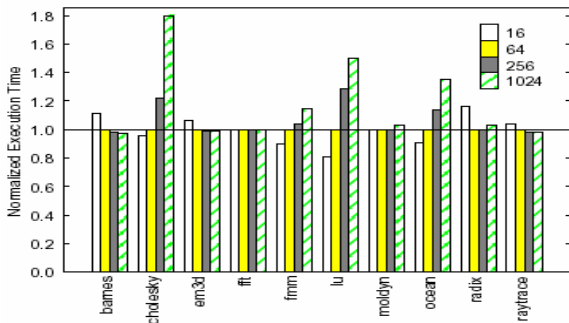
ECE666, Spring 2007 (33) © Mithuna Thottethodi, 2005,2007

### Performance Sensitivity of CC-NUMA and R-NUMA to Cache Size



ECE666, Spring 2007 (34) © Mithuna Thottethodi, 2005,2007

### Performance Sensitivity of R-NUMA to Relocation Threshold Value



ECE666, Spring 2007 (35) © Mithuna Thottethodi, 2005,2007

### R-NUMA Summary

- R-NUMA outperforms CC-NUMA and S-COMA in general
- Little additional hardware
- R-NUMA is less sensitive to hardware configurations than CC-NUMA and S-COMA

ECE666, Spring 2007 (36) © Mithuna Thottethodi, 2005,2007

## Other SW/HW tradeoffs

- Coherence
  - SVM can do page based coherence in S/W
  - Block level S/W coherence also possible
    - Blizzard-S, Shasta
    - See paper reading list
  - Can the compiler do it?
    - Eliminate run-time solutions
    - Figure out coherence (conservatively) at compile time

ECE666, Spring 2007

(37)

© Mithuna Thottethodi, 2005,2007

## Compiler Directed Coherence

- Compiler manages cache
  - No directory hardware (too complex, not scalable)
  - Divide program into computation units (epochs)

```
repeat /* computation of vector  $x_{i+1} = Ax_i + b$  */
doall j = 1 to N /* create N parallel tasks */
  xtemp[j] = b[j];
epoch 1 | for k = 1 to N
        |   xtemp[j] = xtemp[j] + A[j,k]*x[k];
        | end doall
doall j = 1 to N /* create N parallel tasks */
epoch 2 | x[j] = xtemp[j]; /* write new vectors */
        | end doall /* end of epoch */
until all vectors computed
```

ECE666, Spring 2007

(38)

© Mithuna Thottethodi, 2005,2007

## Compiler Directed Coherence (Continued)

- Indiscriminate Invalidation
  - Invalidate entire cache after each epoch
  - Uses uncached read/write as one option
- Selective Invalidation
  - Invalidate only cache lines that may introduce incoherence
  - Assumes write-through caches
- Timestamp
  - Clock per data structure incremented at end of epoch
  - Timestamp cache block with clock +1 when written
  - Valid if timestamp > clock
- **BUT POINTERS!**
- In general has only been shown to work for one word cache blocks

ECE666, Spring 2007

(39)

© Mithuna Thottethodi, 2005,2007

## Hierarchical Coherence

- Most solutions so far are flat
  - E.g., a directory that points to 1K processors
- Use hierarchy
  - Intra-node coherence (e.g., snooping in SMP node)
  - Inter-node coherence (e.g., directory between nodes)
- Why?
  - Divide & conquer markets (e.g., sell node)
  - Divide & conquer complexity (but must interface protocols)

ECE666, Spring 2007

(40)

© Mithuna Thottethodi, 2005,2007

## Advantages of Multiprocessor Nodes

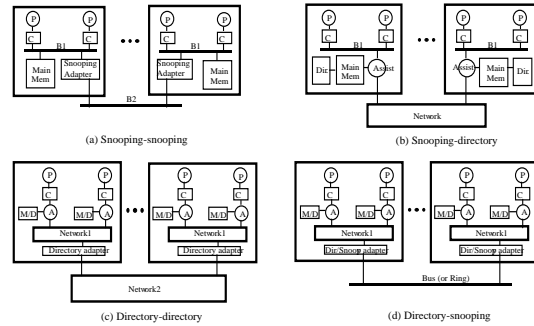
- amortization of node fixed costs over multiple processors
  - applies even if processors simply packaged together but not coherent
- can use commodity SMPs
  - More compelling with CMPs
- less nodes for directory to keep track of
- much communication may be contained within node (cheaper)
- nodes prefetch data for each other (fewer "remote" misses)
- combining of requests (like hierarchical, only two-level)
- can even share caches (overlapping of working sets)
- benefits depend on sharing pattern (and mapping)
  - good for widely read-shared: e.g. tree data in Barnes-Hut
  - good for nearest-neighbor, if properly mapped
  - not so good for all-to-all communication

ECE666, Spring 2007

(41)

© Mithuna Thottethodi, 2005,2007

## Example Two-level Hierarchies



ECE666, Spring 2007

(42)

© Mithuna Thottethodi, 2005,2007

## Disadvantages of Coherent MP Nodes

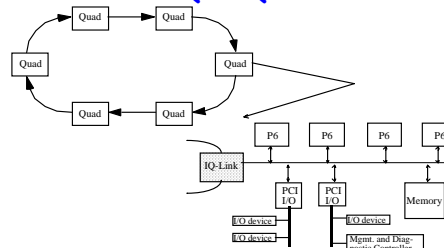
- Bandwidth shared among nodes
  - all-to-all example
  - applies to coherent or not
- Bus increases latency to local memory
- With coherence, typically wait for local snoop results before sending remote requests
- Snoopy bus at remote node increases delays there too, increasing latency and reducing bandwidth
- Overall, may hurt performance if sharing patterns don't comply

ECE666, Spring 2007

(43)

© Mithuna Thottethodi, 2005,2007

## NUMA-Q IQ-Link Board



- Use of high-volume SMPs as building blocks
- Quad bus is 532MB/s split-transaction in-order responses
  - limited facility for out-of-order responses for off-node accesses
- Cross-node interconnect is 1GB/s unidirectional ring
- Larger SCI systems built by bridging multiple rings

ECE666, Spring 2007

(44)

© Mithuna Thottethodi, 2005,2007

## NUMA-Q cont.

- IQ-Link is Key
  - local directory: {home, fresh, gone} + pointer
  - "L3" cache for remote data (tags + data)
  - Internal bus & external network interfaces
  - Two ASICs + memory; protocols use microcode
- Global Protocol
  - Coherence between nodes, not processors
  - Data in processor cache either
    - from local memory (directory knows)
    - in L3 cache (inclusion)
- Local Protocol
  - MESI Snooping
  - IQ-Link asserts "delayed reply" if
    - access to local memory & directory says "gone"
    - access to remote memory (but L3?)