



BROWN

Computer System Design Lecture 32: Protocols and Synchronization

Prof. R. Iris Bahar
EN164
April 20, 2007

Reading: Chapter 4, section 4.4, 4.5



BROWN

Performance Improvements

- What determines performance on a multiprocessor?
 - What fraction of the program is parallelizable?
 - How does memory hierarchy performance change?
- New form of cache miss: *coherence miss*
 - such a miss would not have happened if another processor did not write to the same cache line
- *False coherence miss*: the second processor writes to a different word in the same cache line
 - this miss would not have happened if the line size equaled one word



BROWN

How Do Cache Misses Scale?

	Compulsory	Capacity	Conflict	Coherence	
				True	False
Increasing cache capacity					
Increasing processor count					
Increasing block size					
Increasing associativity					



BROWN

How Do Cache Misses Scale?

	Compulsory	Capacity	Conflict	Coherence	
				True	False
Increasing cache capacity	No affect	Decrease	Decrease	Slight increase	Slight increase
Increasing processor count	Slight increase	No affect	No affect	Increase	Increase (some)
Increasing block size	Decrease	Slight decrease	Slight decrease	Decrease	Increase
Increasing associativity	No affect	No affect	Decrease	Slight increase?	Slight increase?

Results may vary with the applications being run on the processors



BROWN

Simplifying Assumptions

- All transactions on a read or write are atomic
 - on a write miss, the miss is sent on the bus, a block is fetched from memory/remote cache, and the block is marked exclusive
- Potential problem if the actions are non-atomic: P1 sends a write miss on the bus, P2 sends a write miss on the bus:
 - since the block is still invalid in P1, P2 does not realize that it should write after receiving the block from P1
 - instead, it receives the block from memory
- Fix by keeping track of more state:
 - Don't acquire the block unless all outstanding transactions have completed



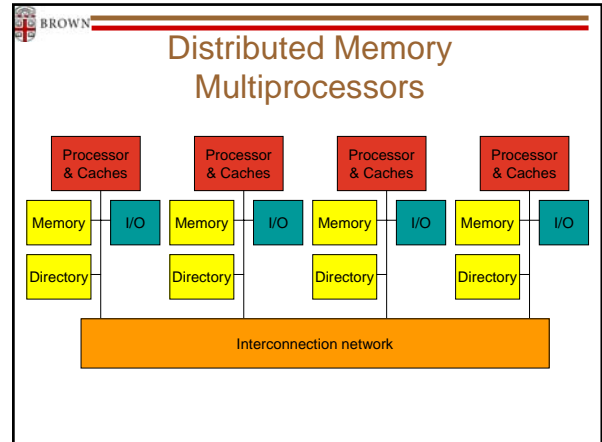
BROWN

Coherence in Distributed Memory Multiprocessor

- Distributed memory systems are typically larger
 - bus-based snooping may not work well
- Option 1: software-based mechanisms
 - message-passing systems or software-controlled cache coherence
- Option 2: hardware-based mechanisms
 - directory-based cache coherence

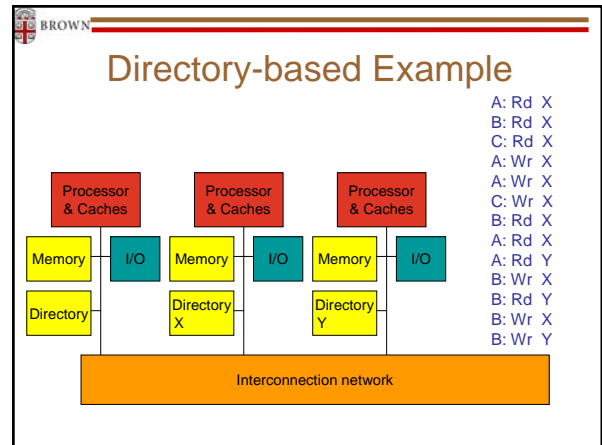
Directory-based Cache Coherence

- The physical memory is distributed among all processors
- The directory is also distributed along with the corresponding memory
- The physical address is enough to determine the unique location of memory
- The (many) processing nodes are connected with a scalable interconnect (not a bus)
 - messages are no longer broadcast, but routed from sender to receiver
 - the directory keeps track of sharing state



Cache Block States

- What are the different states a block of memory can have within the directory?
- We need information for each cache so that invalidate messages can be sent
- The block state is also stored in the cache for efficiency
- The directory now serves as the arbitrator:
 - if multiple write attempts happen simultaneously, the directory determines the ordering



DSM Example

	A	B	C
A: Rd X	S		
B: Rd X	S	S	
C: Rd X	S	S	S
A: Wr X	E	I	I
A: Wr X	E	I	E
C: Wr X	I	I	I
B: Rd X	I	S	S
A: Rd X	S	S	S
A: Rd Y	S (Y)	S (X)	S (X)
B: Wr X	S (Y)	E (X)	I
B: Rd Y	S (Y)	S (Y)	I
B: Wr X	S (Y)	E (X)	I
B: Wr Y	I	E (Y)	I

Directory Actions

- If block is in uncached state:
 - Read miss: send data, make block shared
 - Write miss: send data, make block exclusive
- If block is in shared state:
 - Read miss: send data, *add node to sharers list*
 - Write miss: send data, *invalidate sharers*, make exclusive
- If block is in exclusive state:
 - Read miss: ask owner for data, write to memory, send data, make shared, *add node to sharers list*
 - Data write back: write to memory, make uncached
 - Write miss: ask owner for data, write to memory, send data, *update identity of new owner*, remain exclusive

Example Protocol

<i>Message</i>	<i>Source</i>	<i>Destination</i>	<i>Action</i>
Read miss	Local cache	Home directory	Request data and make proc. a read sharer
Write miss	Local cache	Home directory	Request data and make proc. the exclusive owner
Invalidate	Home directory	Remote cache	Invalidate a shared copy of the data
Fetch	Home directory	Remote cache	Fetch block, send to home directory, remote cache state set to shared
Fetch / invalidate	Home directory	Remote cache	Fetch block, send to home directory, invalidate block
Data value reply	Home directory	Local cache	Return data from home
Data writeback	Remote cache	Home directory	Writeback data to home