ECSE 425 Lecture 30: Directory Coherence

H&P Chapter 4

Last Time

- Snoopy Coherence
- Symmetric SMP Performance

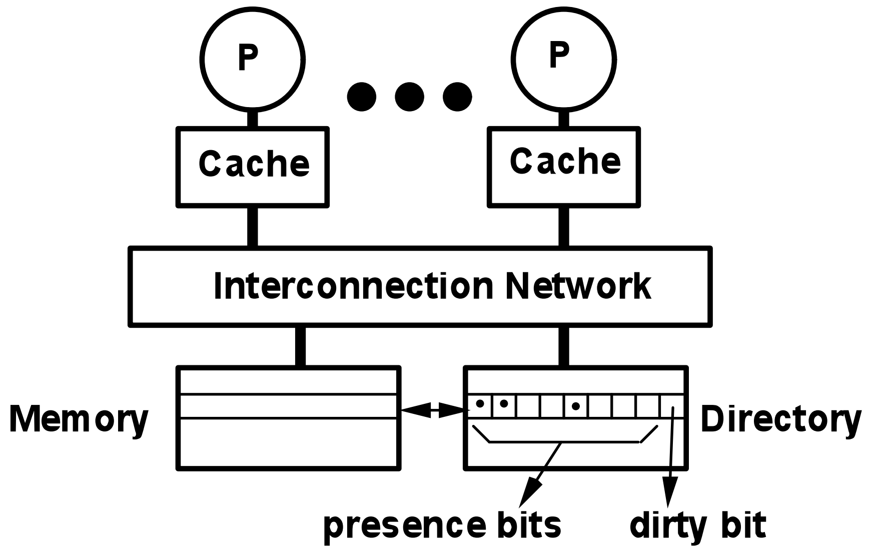
Today

- Directory-based Coherence

A Scalable Approach: Directories

- One directory entry for each memory block
  - Tracks valid copies of cached blocks and their states
- On a miss
  - Find directory entry, look up location of data
  - Communicate only with the node that has valid data
- No broadcast medium necessary
  - In scalable networks, communicate with directory and copies through network transactions
- Many alternatives for organizing directory information
  - Apply to both distributed and centralized memory systems
  - The directory itself can be distributed along with memory

Basic Operation of Directory



- k processors.
- With each cache-block in memory:
  k presence-bits, 1 dirty-bit
- With each cache-block in cache:
  1 valid bit, and 1 dirty (owner) bit

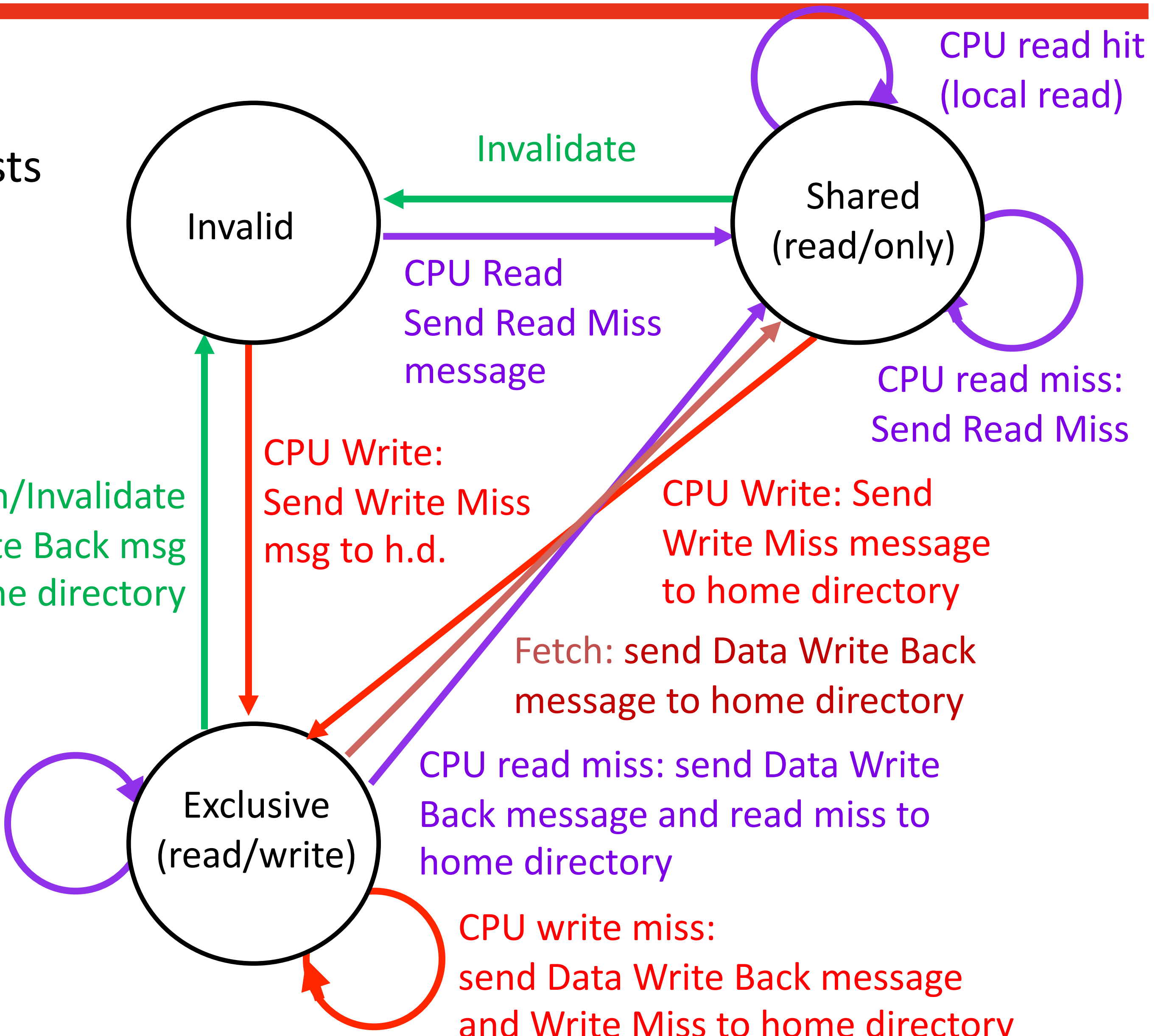
- Read from main memory by processor *i*:
  - If dirty-bit OFF then { read from main memory; turn p[*i*] ON; }
  - if dirty-bit ON then { recall line from dirty proc (cache state to shared); update memory; turn dirty-bit OFF; turn p[*i*] ON; supply recalled data to *i*;}
- Write to main memory by processor *i*:
  - If dirty-bit OFF then { supply data to *i*; send invalidations to all caches that have the block, turning p[*j*] OFF; turn dirty-bit ON; turn p[*i*] ON; ... }
  - ...

Directory Protocol

- No bus; don't want to broadcast:
  - Interconnect no longer a serialization point
  - All messages have explicit responses
- Typically three nodes (proc+mem) involved in any request
  - Local node: where the request originates
  - Home node: location of the directory entry for an address
  - Remote node: has a copy of a cache block, exclusive or shared
- We will assume a simple memory consistency model
  - Writes to non-exclusive data => treats as write miss
  - Processor blocks until access completes
  - Messages received and acted upon in the same order sent

CPU - Cache State Machine

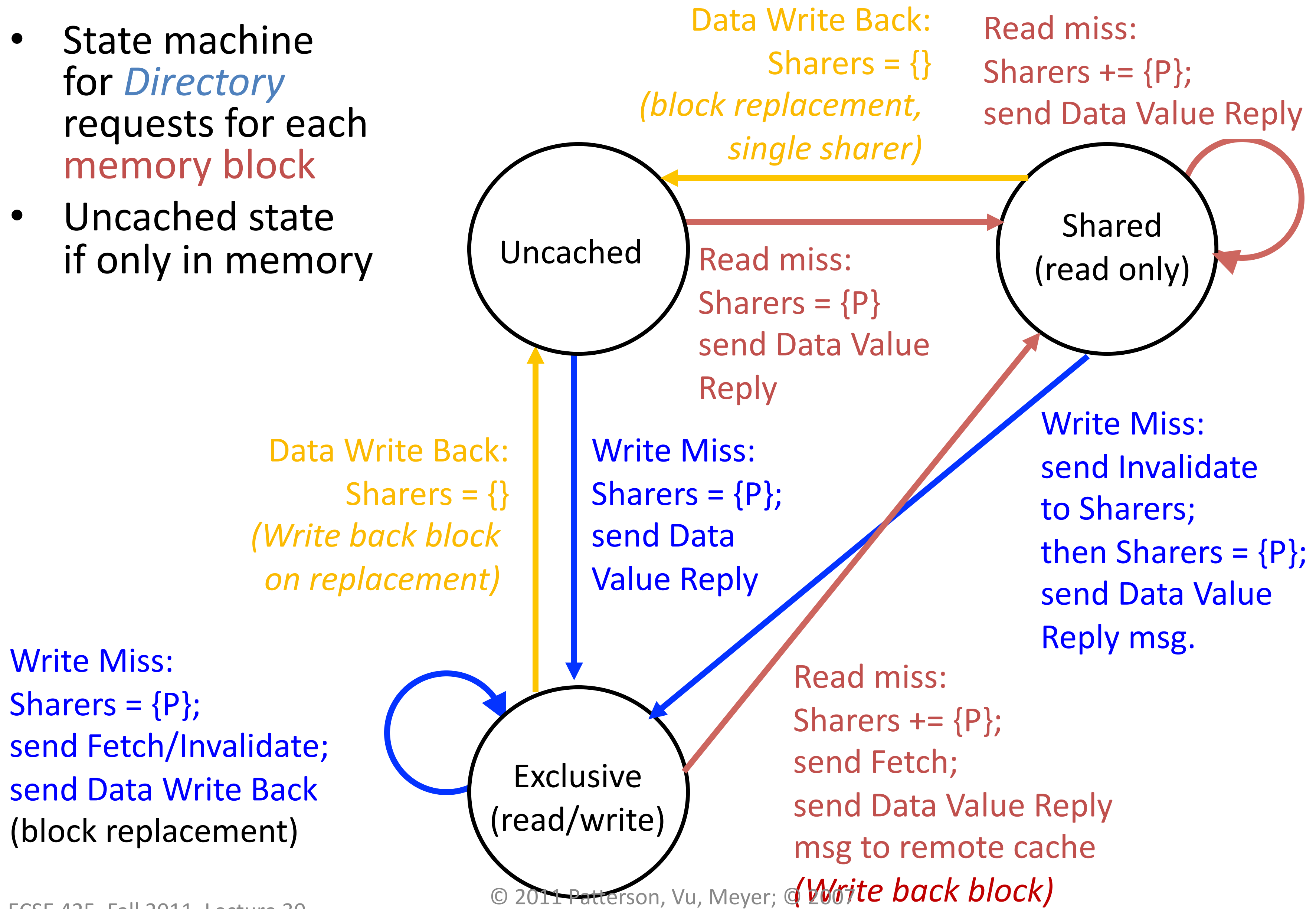
- State machine for *CPU and directory* requests for each memory block
- Invalid state if block is not in cache



Invalid
Shared (read/only)
Exclusive (read/write)
CPU read hit (local read)
Invalidate
CPU Read Send Read Miss message
CPU read miss: Send Read Miss
Fetch/Invalidate send Data Write Back msg to home directory
CPU Write: Send Write Miss msg to h.d.
CPU Write: Send Write Miss message to home directory
Fetch: send Data Write Back message to home directory
CPU read miss: send Data Write Back message and read miss to home directory
CPU read hit CPU write hit (local Rd/Wr)
CPU write miss: send Data Write Back message and Write Miss to home directory

Directory State Machine

- State machine for *Directory* requests for each memory block
- Uncached state if only in memory



Data Write Back: Sharers = {} *(block replacement, single sharer)*
Read miss: Sharers += {P}; send Data Value Reply
Uncached
Shared (read only)
Read miss: Sharers = {P} send Data Value Reply
Data Write Back: Sharers = {} *(Write back block on replacement)*
Write Miss: Sharers = {P}; send Data Value Reply
Write Miss: send Invalidate to Sharers; then Sharers = {P}; send Data Value Reply msg.
Write Miss: Sharers = {P}; send Fetch/Invalidate; send Data Write Back (block replacement)
Exclusive (read/write)
Read miss: Sharers += {P}; send Fetch; send Data Value Reply msg to remote cache *(Write back block)*

Protocol: Uncached Block

- A message sent to the directory causes two actions:
  - The directory is updated
  - More messages are sent to satisfy the specific request
- Block is Uncached; the memory value is up-to-date
  - Read miss: first sharer
    - Requesting processor is sent the value; becomes sole Sharer
    - Block state becomes Shared
  - Write miss: new owner
    - Requesting processor is sent the value; becomes Owner
    - Block state becomes Exclusive; the only valid data is cached

Protocol: Shared Block

- Block is Shared; the memory value is up-to-date
  - Read miss: new sharer
    - Requesting processor is sent the value; becomes Sharer
  - Write miss: new owner
    - Processors in the Sharers set are sent invalidate messages
    - Requesting processor is sent the value; becomes Owner
    - Block state becomes Exclusive; the only valid data is cached

Protocol: Exclusive Block

- Block is Exclusive; the memory value is out-of date, current value is held Owner's cache
  - Read miss: new sharer
    - Directory sends Owner data fetch request
    - Owner sends data to directory; becomes Sharer
    - Block state becomes Shared
    - Requesting processor is sent the value; becomes Sharer

Protocol: Exclusive Block, Cont'd

- Block is Exclusive
  - Data write-back: owner replaces the block
    - Owner sends data to directory; memory value is up-to-date
    - Block state becomes Uncached; the Sharer set is empty
  - Write miss: new owner
    - Directory sends Owner data fetch request
    - Owner responds with data; invalidates local copy
    - Requesting processor is sent the value; becomes Owner
    - Block state remains Exclusive

Example

	Processor 1			Processor 2			Interconnect				Directory			Memory
	P1			P2			Bus				Directory			Memor
step	State	Addr	Value	State	Addr	Value	Actior	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1 Write 10 to A1														
P1: Read A1														
P2: Read A1														
P2: Write 20 to A1														
P2: Write 40 to A2														

A1 and A2 map to the same cache block

Example

Processor 1 Processor 2 Interconnect Directory Memory

	P1			P2			Bus				Directory			Memor
step	State	Addr	Value	State	Addr	Value	Actior	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1 Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1														
P2: Read A1														
P2: Write 20 to A1														
P2: Write 40 to A2														

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect				Directory			Memory
	P1			P2			Bus				Directory			Memor
step	State	Addr	Value	State	Addr	Value	Actior	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1 Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1														
P2: Write 20 to A1														
P2: Write 40 to A2														

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect				Directory			Memory
	P1			P2			Bus				Directory			Memor
step	State	Addr	Value	State	Addr	Value	Actior	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1 Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1				Shar.	A1		RdMs	P2	A1					
	Shar.	A1	10				Ftch	P1	A1	10	A1			10
				Shar.	A1	10	DaRp	P2	A1	10	A1	Shar.	{P1,P2}	10
P2: Write 20 to A1	1													
P2: Write 40 to A2														

Write Back

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect				Directory			Memory
	P1			P2			Bus				Directory			Memor
step	State	Addr	Value	State	Addr	Value	Actior	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1 Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1				Shar.	A1		RdMs	P2	A1					
	Shar.	A1	10				Ftch	P1	A1	10	A1			10
				Shar.	A1	10	DaRp	P2	A1	10	A1	Shar.	{P1,P2}	10
P2: Write 20 to A1							WrMs	P2	A1					10
	Inv.			Excl.	A1	20	Inval.	P1	A1		A1	Excl.	{P2}	10
P2: Write 40 to A2														

A1 and A2 map to the same cache block

Example

	Processor 1			Processor 2			Interconnect				Directory			Memory
	P1			P2			Bus				Directory			Memor
step	State	Addr	Value	State	Addr	Value	Actior	Proc.	Addr	Value	Addr	State	{Procs}	Value
P1 Write 10 to A1							WrMs	P1	A1		A1	Ex	{P1}	
	Excl.	A1	10				DaRp	P1	A1	0				
P1: Read A1	Excl.	A1	10											
P2: Read A1				Shar.	A1		RdMs	P2	A1					
	Shar.	A1	10				Ftch	P1	A1	10	A1			10
				Shar.	A1	10	DaRp	P2	A1	10	A1	Shar.	{P1,P2}	10
P2: Write 20 to A1							WrMs	P2	A1					10
	Inv.			Excl.	A1	20	Inval.	P1	A1		A1	Excl.	{P2}	10
P2: Write 40 to A2							WrBk	P2	A1	20	A1	Unca.	{}	
							WrMs	P2	A2		A2	Excl.	{P2}	0
				Excl.	A2	40	DaRp	P2	A2	0	A2	Excl.	{P2}	0

A1 and A2 map to the same cache block

Implementing a Directory

- We assume operations atomic, but they are not
- Reality is much harder
  - Network buffers are finite; how to avoid deadlock?
  - See Appendix E
- Optimizations: read or write miss when Exclusive
  - P1->Dir; Dir->P2; P2->Dir; Dir->P1
    - Messaging through directory requires four messages
  - P1->Dir; Dir->P2; P2->P1, P2->Dir
    - Short-cut leaves three messages on the critical path

Summary: Snooping vs. Directory

- Snoopy coherence
  - Caches monitor the bus for coherence traffic
    - Processors satisfy requests for dirty data
  - Caches maintain information about sharing
  - Requires a broadcast medium
    - Serializes accesses
    - Simplifies messaging
  - Uniform memory access time, but not scalable
- Directory coherence
  - Directory maintains sharing state
  - Directory coordinates coherence traffic
  - No broadcast medium required
  - Non-uniform memory access time, but scalable

Next Time

- Last lecture!
  - Synchronization
  - Memory consistency