

Computer Architecture
ELE 475 / COS 475
Slide Deck 15: Directory Cache
Coherence

David Wentzlaff

Department of Electrical Engineering
Princeton University



Performance of Symmetric Shared-Memory Multiprocessors

Cache performance is combination of:

1. Uniprocessor cache miss traffic
2. Traffic caused by communication
 - Results in invalidations and subsequent cache misses
- Adds 4th C: *coherence miss*
 - Joins Compulsory, Capacity, Conflict
 - (Sometimes called a *Communication* miss)

Coherency Misses

1. **True sharing misses** arise from the communication of data through the cache coherence mechanism
 - Invalidates due to 1st write to shared block
 - Reads by another CPU of modified block in different cache
 - Miss would still occur if block size were 1 word
2. **False sharing misses** when a block is invalidated because some word in the block, other than the one being read, is written into
 - Invalidation does not cause a new value to be communicated, but only causes an extra cache miss
 - Block is shared, but no word in block is actually shared
⇒ miss would not occur if block size were 1 word

Example: True v. False Sharing v. Hit?

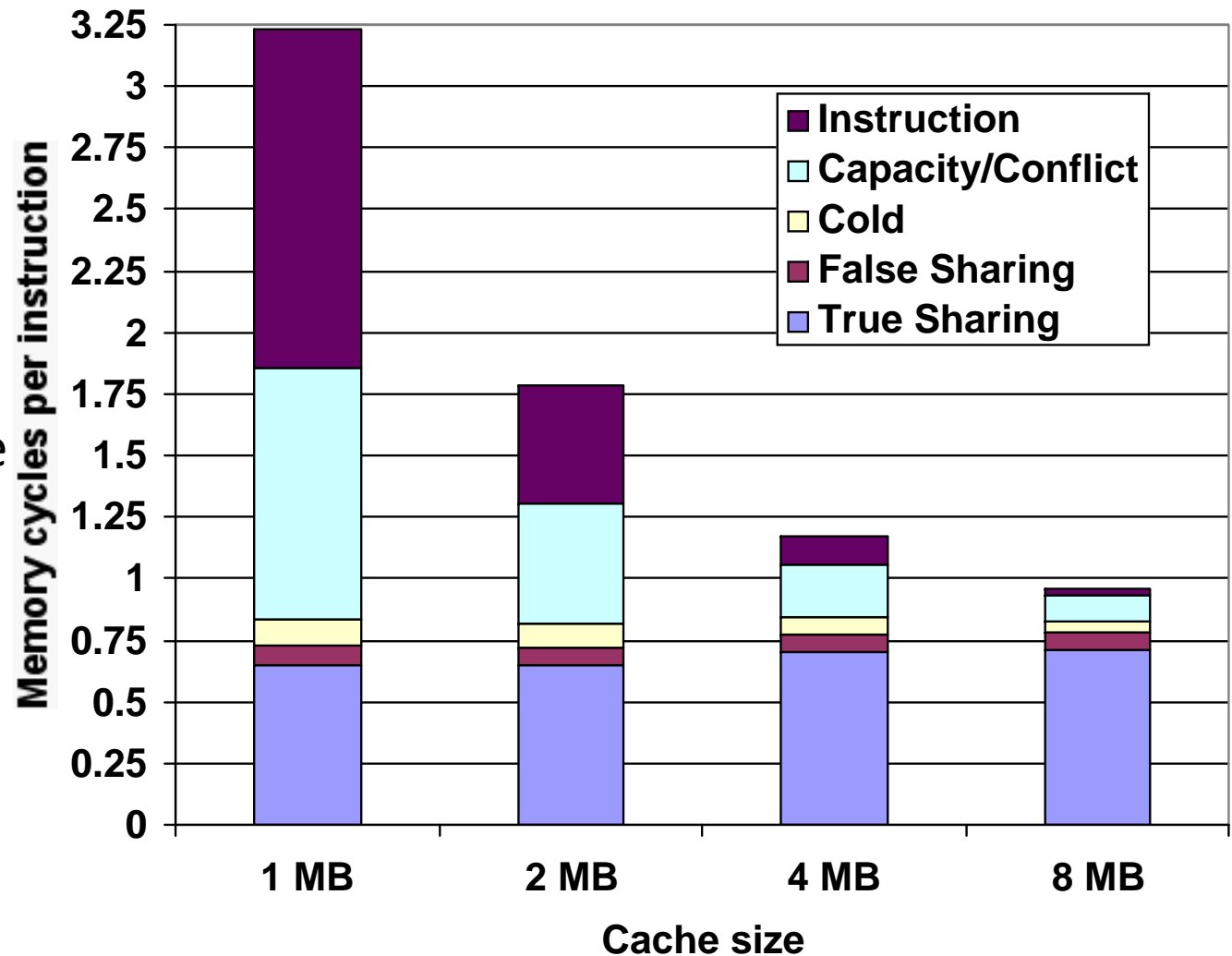
- Assume x1 and x2 in same cache block.
P1 and P2 both read x1 and x2 before.

Time	P1	P2	True, False, Hit? Why?
1	Write x1		True miss; invalidate x1 in P2
2		Read x2	False miss; x1 irrelevant to P2
3	Write x1		False miss; x1 irrelevant to P2
4		Write x2	False miss; x1 irrelevant to P2
5	Read x2		True miss; invalidate x2 in P1

MP Performance 4 Processor

Commercial Workload: OLTP, Decision Support (Database), Search Engine

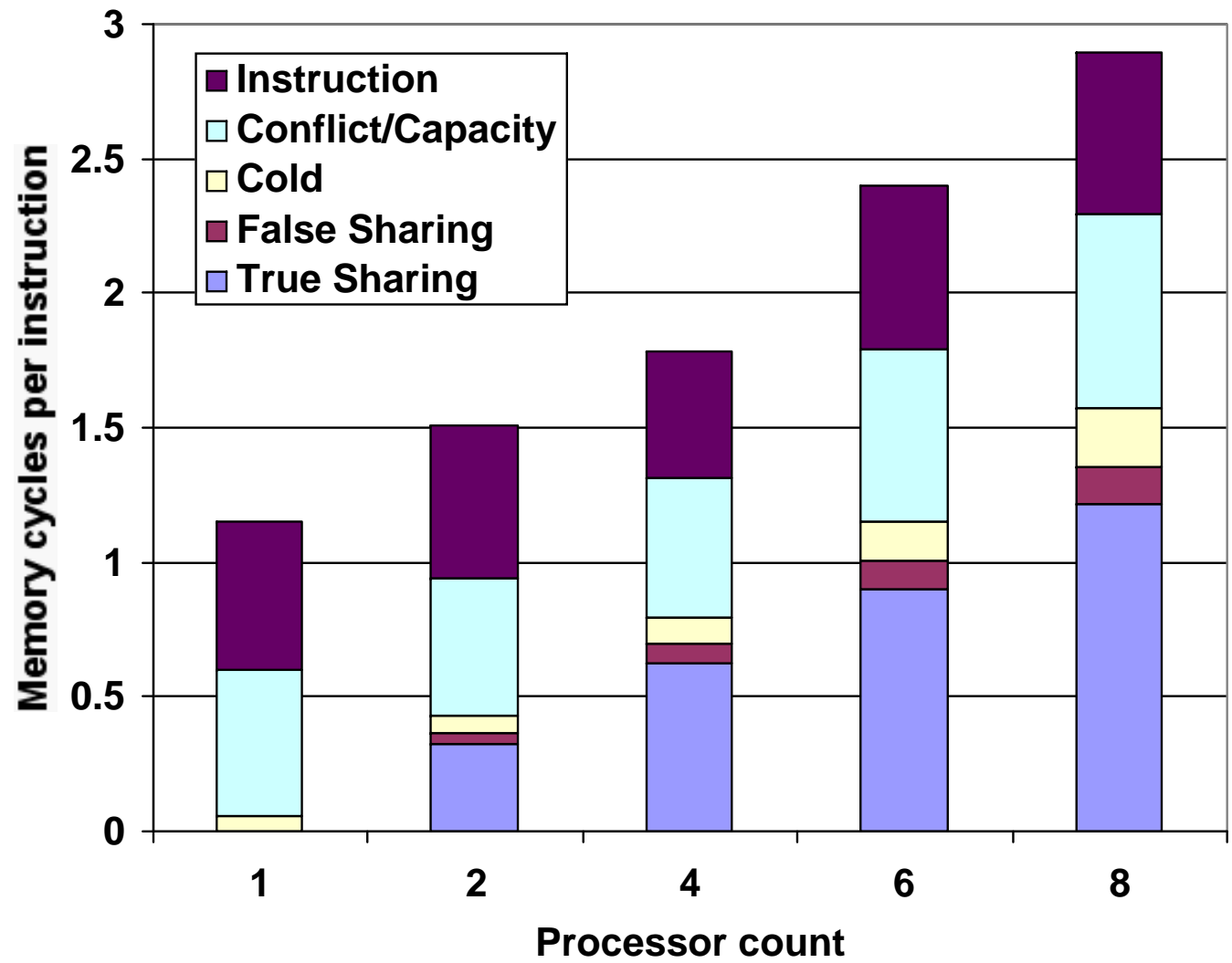
- True sharing and false sharing unchanged going from 1 MB to 8 MB (L3 cache)
- Uniprocessor cache misses improve with cache size increase (Instruction, Capacity/Conflict, Compulsory)



MP Performance 2MB Cache

Commercial Workload: OLTP, Decision Support (Database), Search Engine

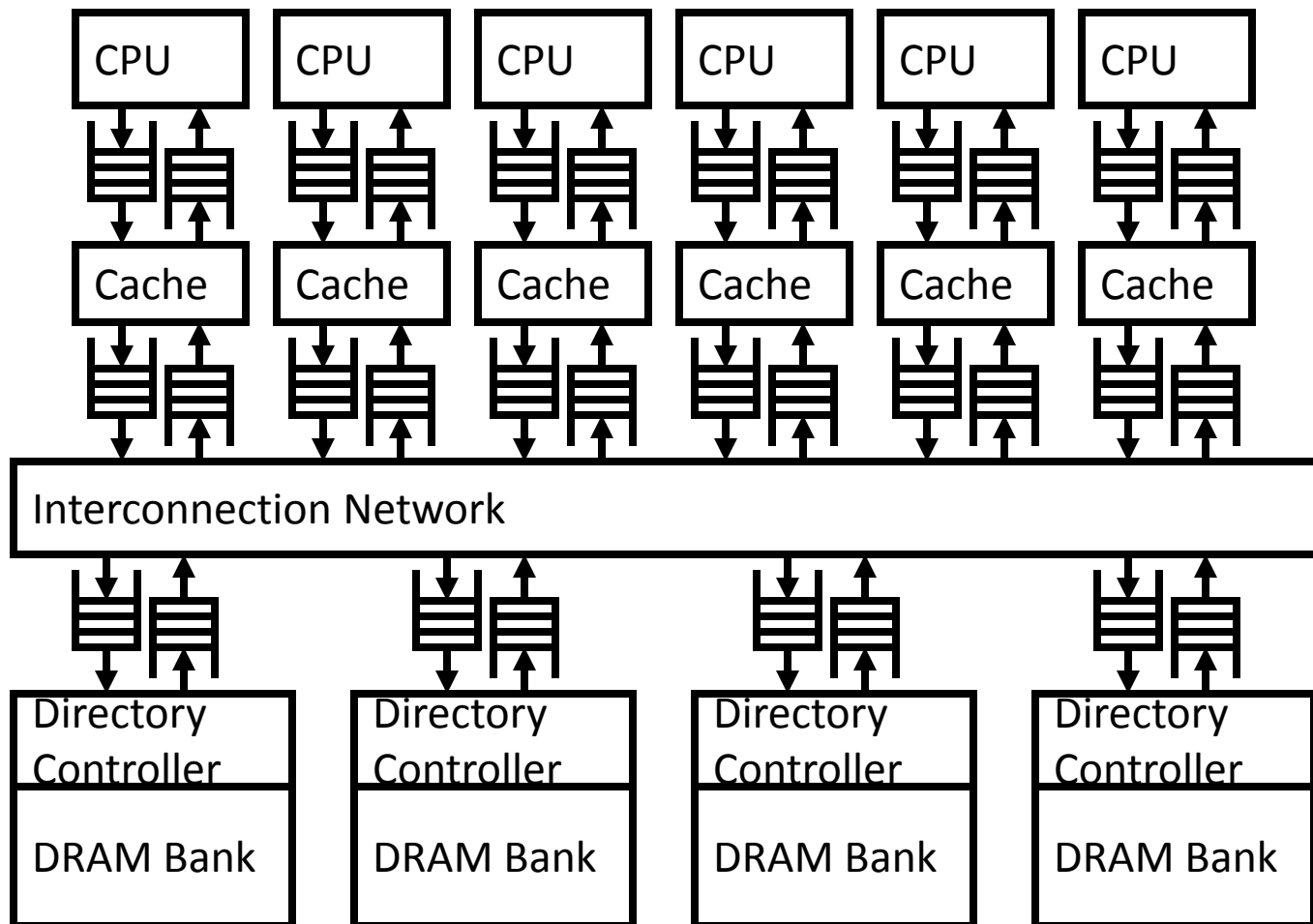
- True sharing, false sharing increase going from 1 to 8 CPUs



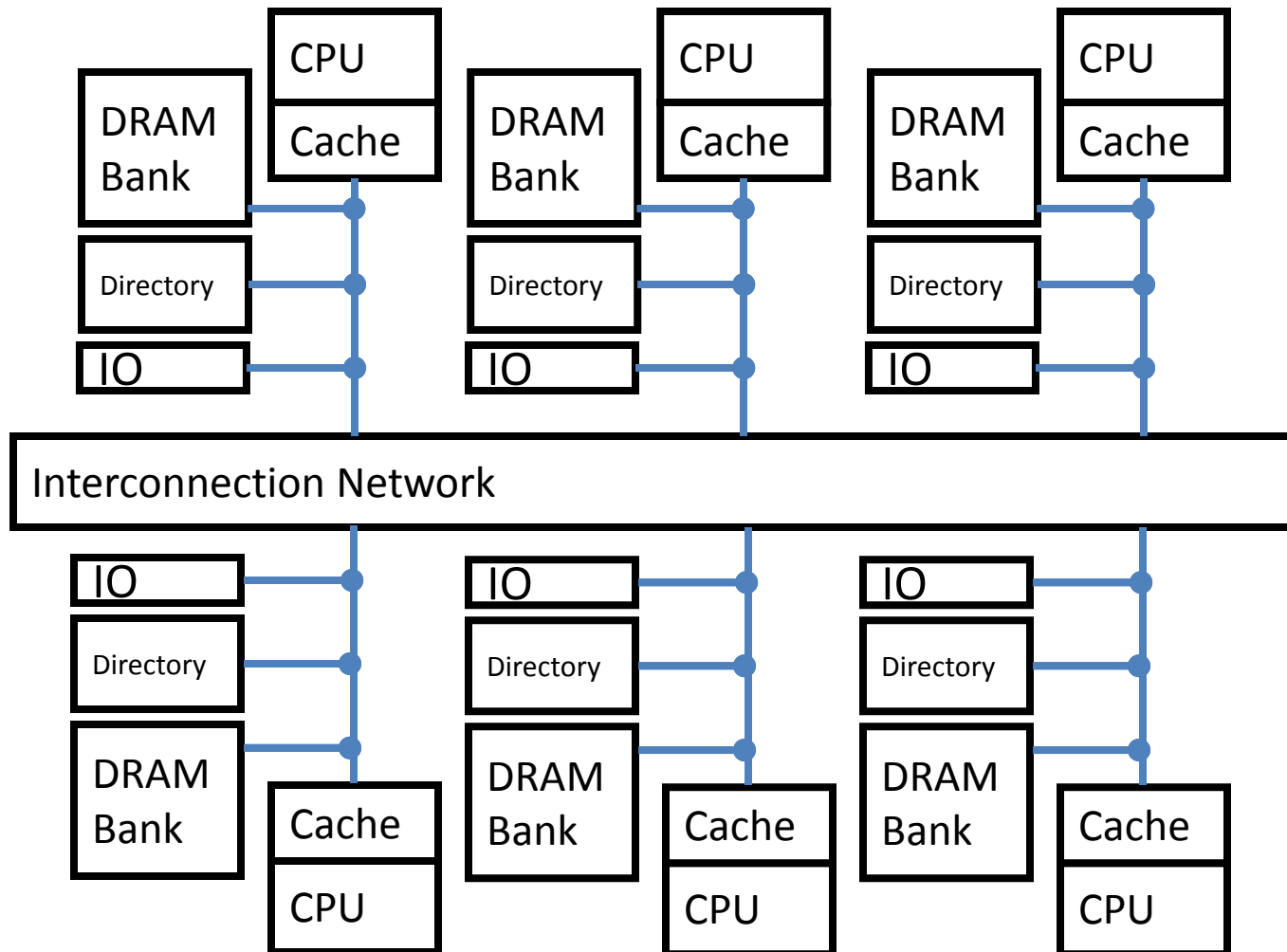
Directory Coherence Motivation

- Snoopy protocols require every cache miss to broadcast
 - Requires large bus bandwidth, $O(N)$
 - Requires large cache snooping bandwidth, $O(N^2)$ aggregate
- Directory protocols enable further scaling
 - Directory can track all caches holding a memory block and use point-to-point messages to maintain coherence
 - Communication done via scalable point-to-point interconnect

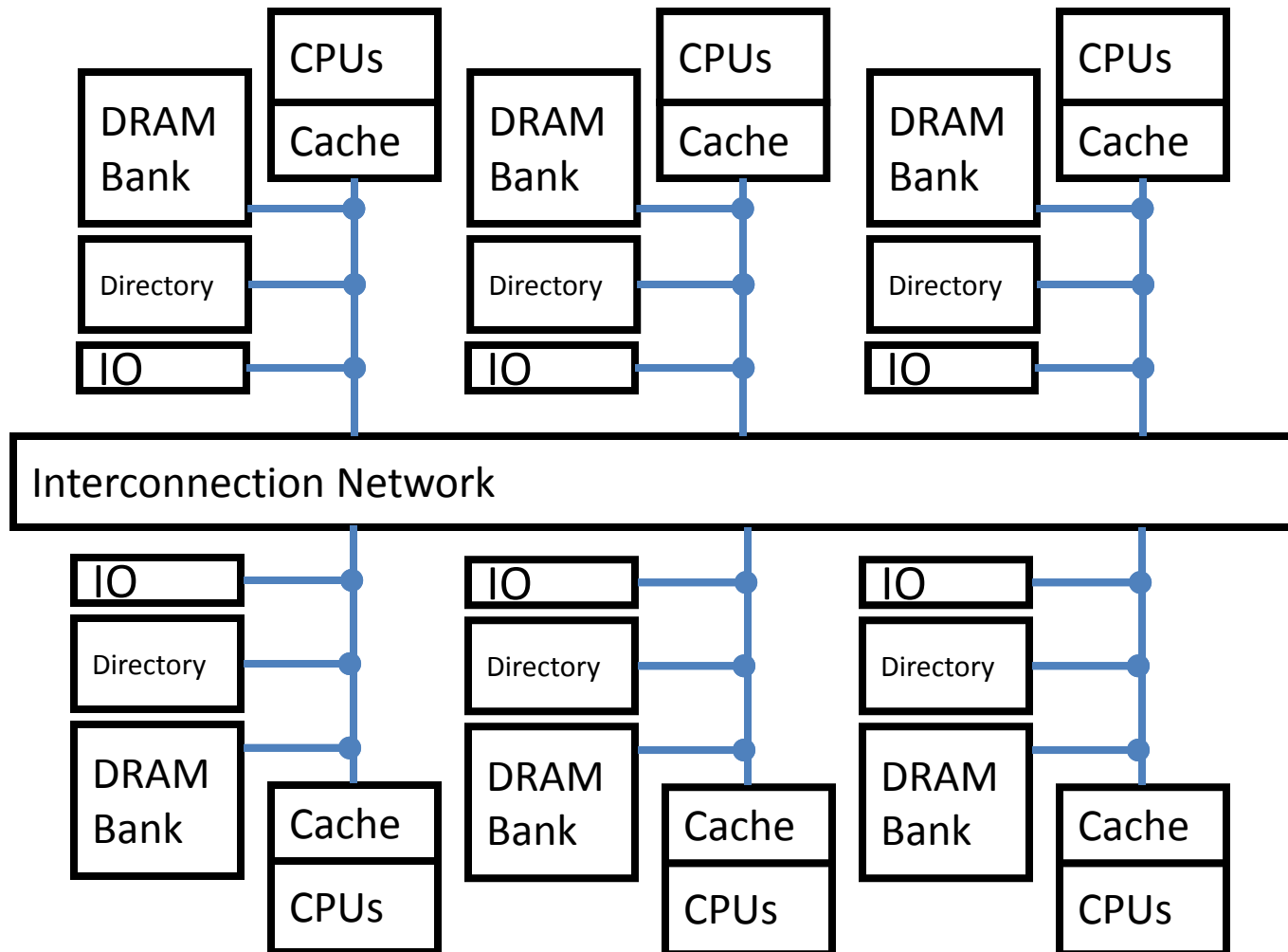
Directory Cache Coherence



Distributed Shared Memory



Multicore Chips in a Multi-chip System



Non-Uniform Memory Access (NUMA)

- Latency to access memory is different depending on node accessing it and address being accessed
- NUMA does not necessarily imply ccNUMA

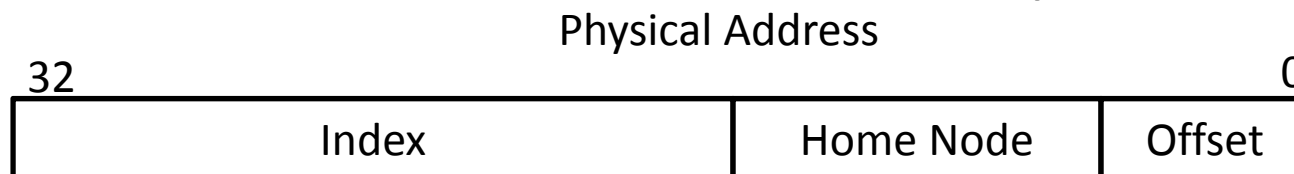
Address to Home Directory

High Order Bits Determine Home (Directory)



- OS can control placement in NUMA
- Homes can become hotspots

Low Order Bits Determine Home (Directory)



- OS loses control on placement
- Load balanced Well

Basic Full-Map Direcorry

State	Sharers/Owner
S	1011001100011
U	xxxxxxxxxxxxxxxxx
U	xxxxxxxxxxxxxxxxx
E	0001000000000
...	

State: State that the directory believes cache line is in {Shared, Uncached, Exclusive, (Pending)}

Sharers/Owner: If State == Shared:
Bit vector of all of the nodes in system that have line in shared state.

If State == Modified:

Denotes which node has line in cache exclusively

Cache State Transition Diagram

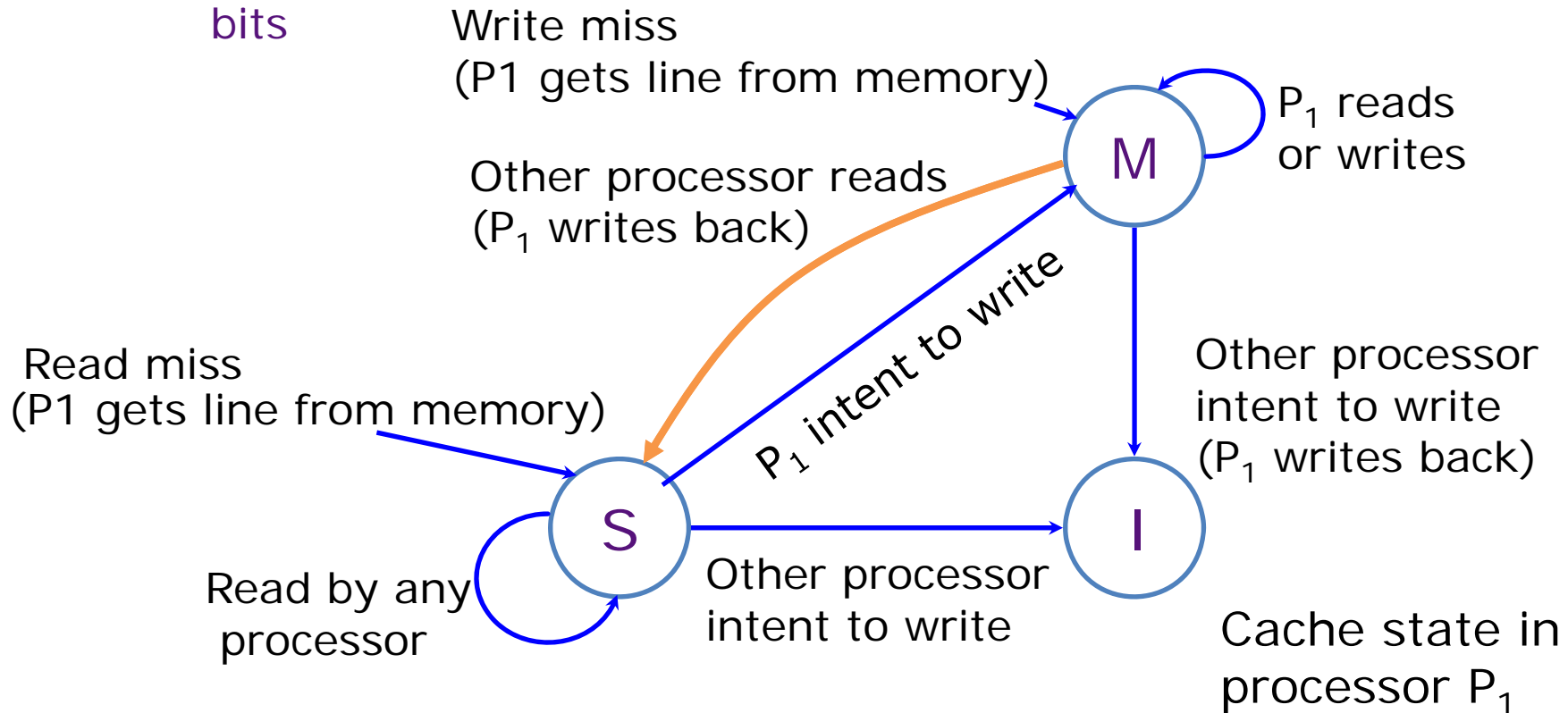
The MSI protocol

Each cache line has state bits

M: Modified

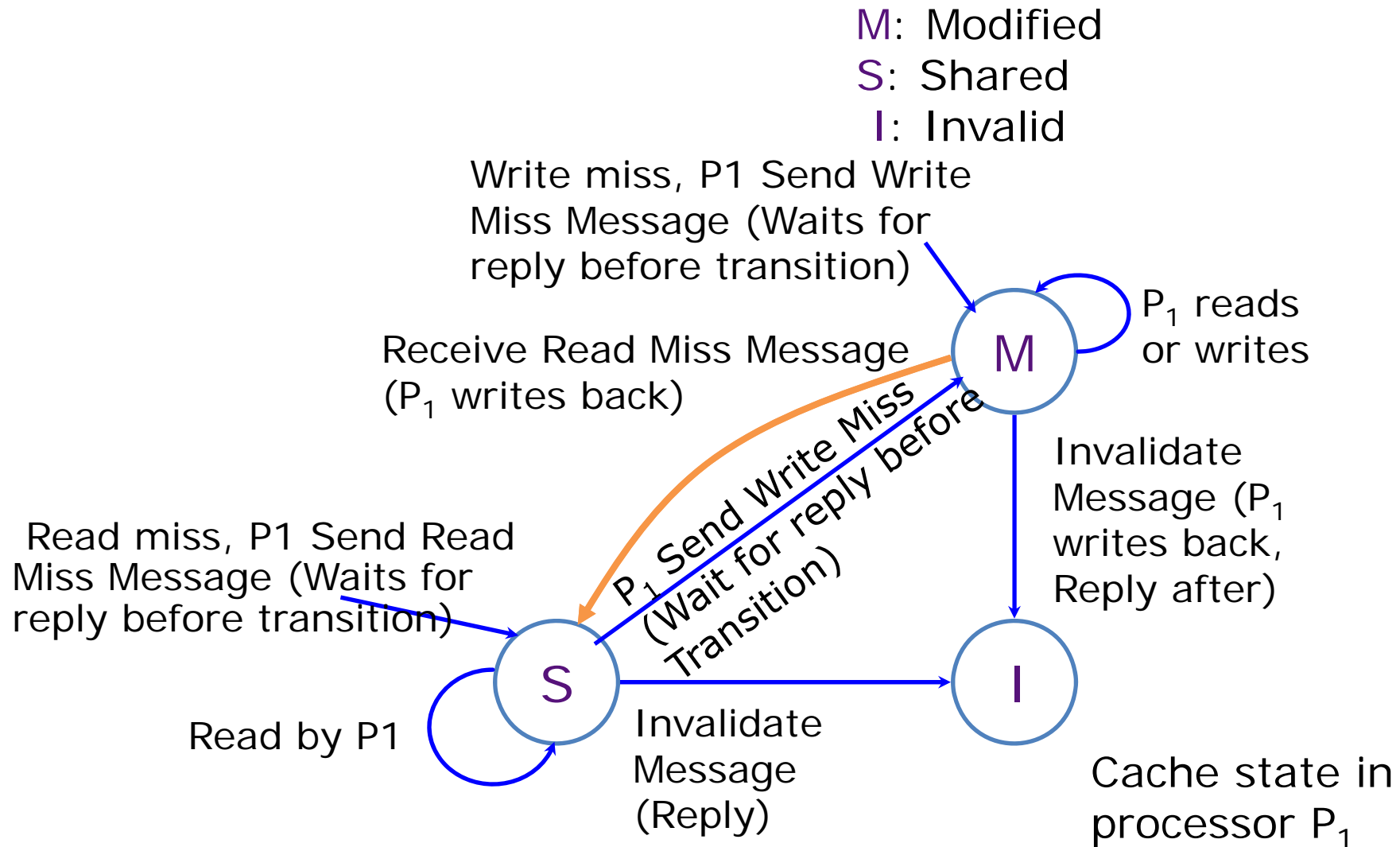
S: Shared

I: Invalid



Cache State Transition Diagram

For Directory Coherence



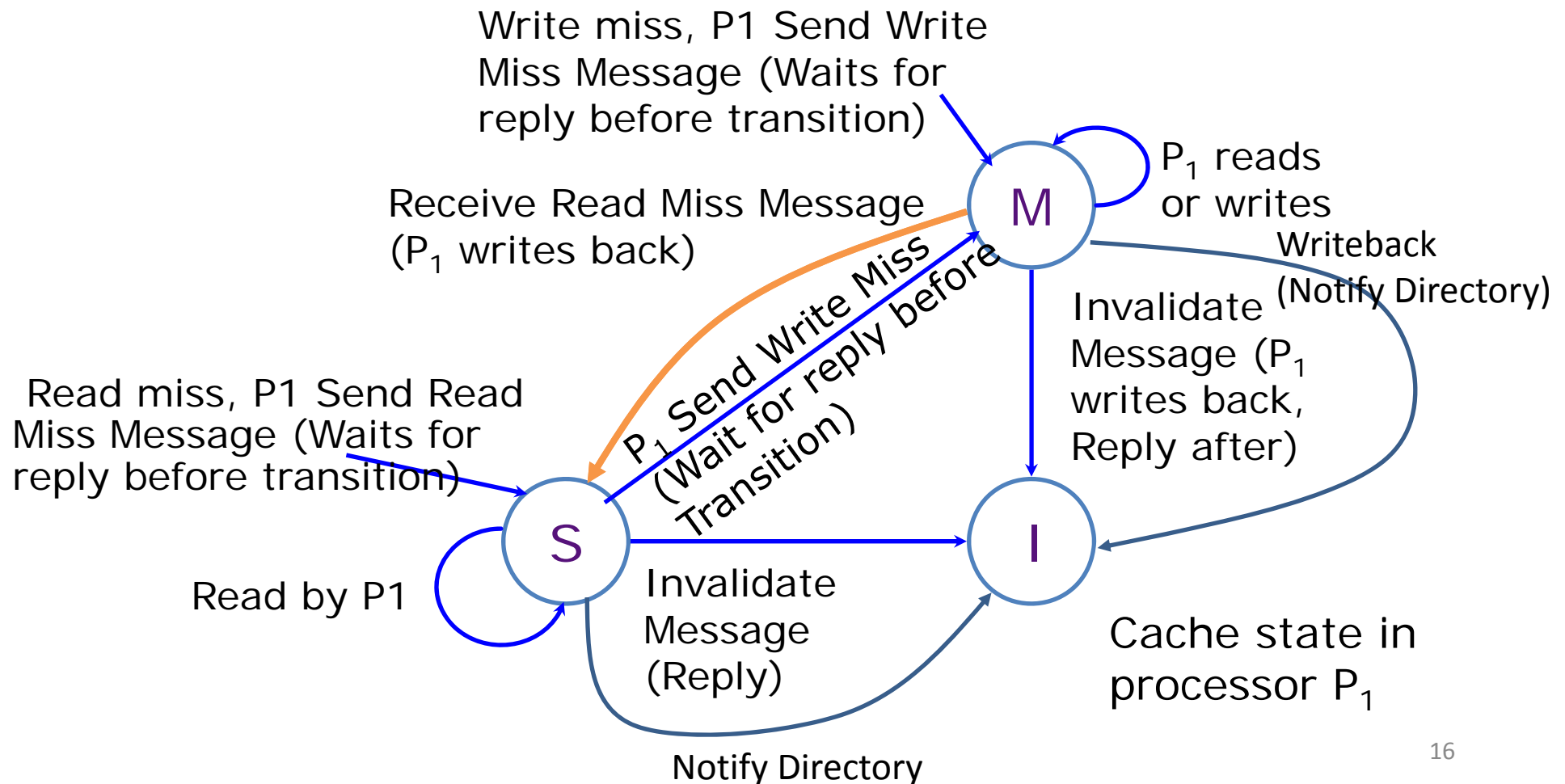
Cache State Transition Diagram

For Directory Coherence

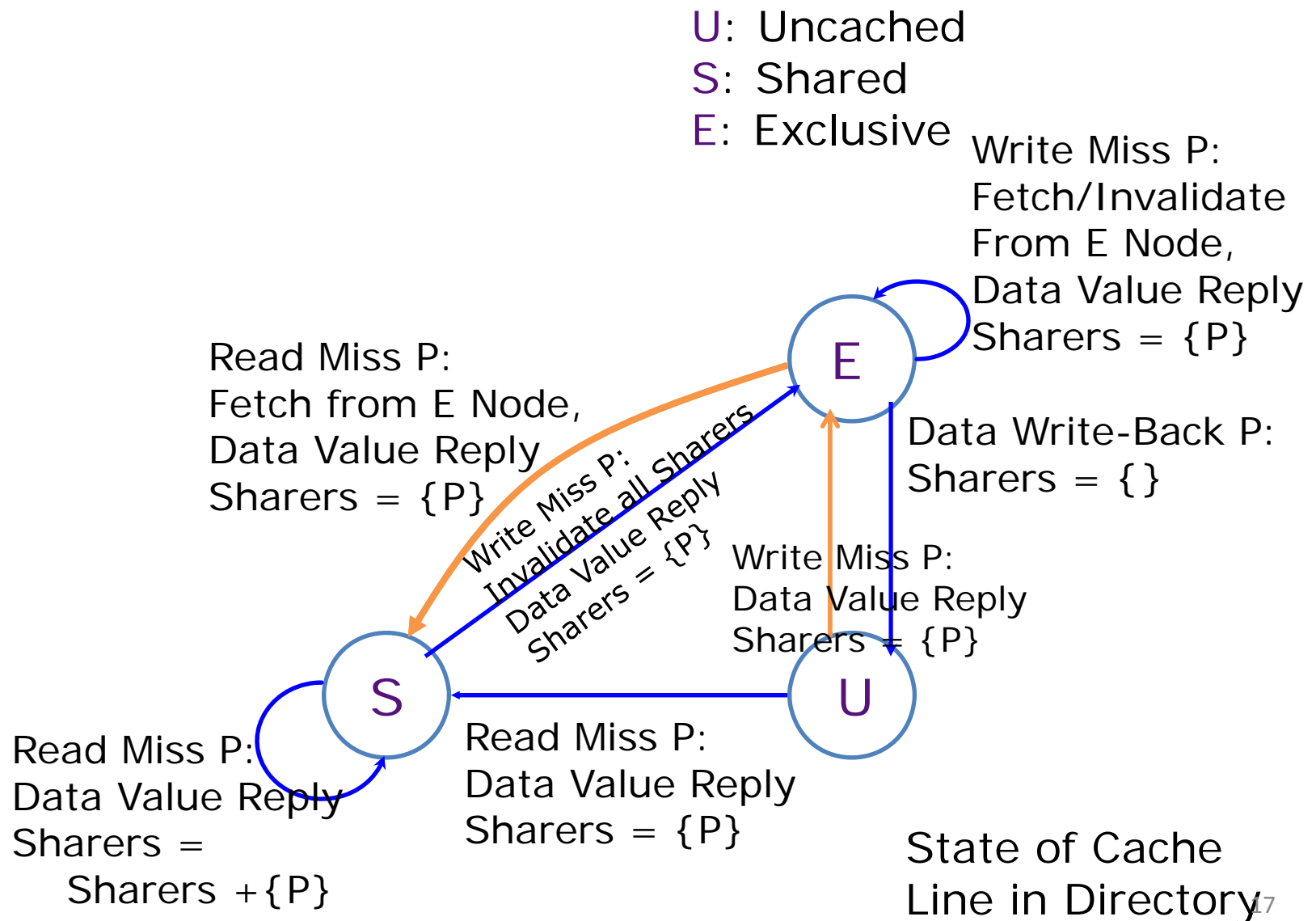
M: Modified

S: Shared

I: Invalid



Directory State Transition Diagram



Message Types

Message type	Source	Destination	Message contents	Function of this message
Read miss	Local cache	Home directory	P, A	Node P has a read miss at address A; request data and make P a read sharer.
Write miss	Local cache	Home directory	P, A	Node P has a write miss at address A; request data and make P the exclusive owner.
Invalidate	Local cache	Home directory	A	Request to send invalidates to all remote caches that are caching the block at address A.
Invalidate	Home directory	Remote cache	A	Invalidate a shared copy of data at address A.
Fetch	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; change the state of A in the remote cache to shared.
Fetch/invalidate	Home directory	Remote cache	A	Fetch the block at address A and send it to its home directory; invalidate the block in the cache.
Data value reply	Home directory	Local cache	D	Return a data value from the home memory.
Data write-back	Remote cache	Home directory	A, D	Write-back a data value for address A.

Multiple Logical Communication Channels Needed

- Responses queued behind requests can lead to deadlock
- Many different message types, need to determine which message type can create more messages
- Segregate flows onto different logical/physical channels

Memory Ordering Point

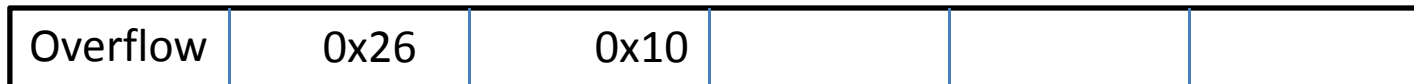
- Just like in bus based snooping protocol, need to guarantee that state transitions are atomic
- Directory used as ordering point
 - Whichever message reaches home directory first wins
 - Other requests on same cache line given negative acknowledgement (NACK)
- NACK causes retry from other node
- Forward progress guarantee needed
 - After node acquires line, need to commit at least one memory operation before transitioning invalidating line

Scalability of Directory Sharer List Storage

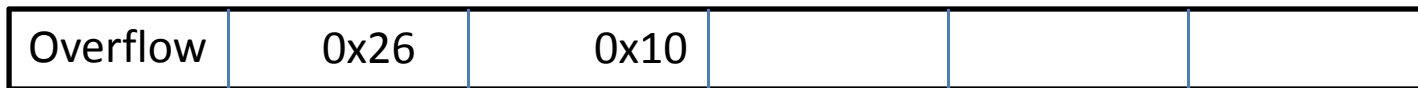
- Full-Map Directory (Bit per cache)



- Limited Pointer (Keep list of base-2 encoded sharers)



- LimitLess (Software assistance)



Beyond Simple Directory Coherence

- On-chip coherence (Leverage fast on-chip communications to speed up or simplify protocol)
- Cache Only Memory Architectures (COMA)
- Large scale directory systems (Scalability of directory messages and sharer list storage)

SGI UV 1000 (Origin Descendant)

Maximum Memory: 16TB

Maximum Processors: 256

Maximum Cores: 2560

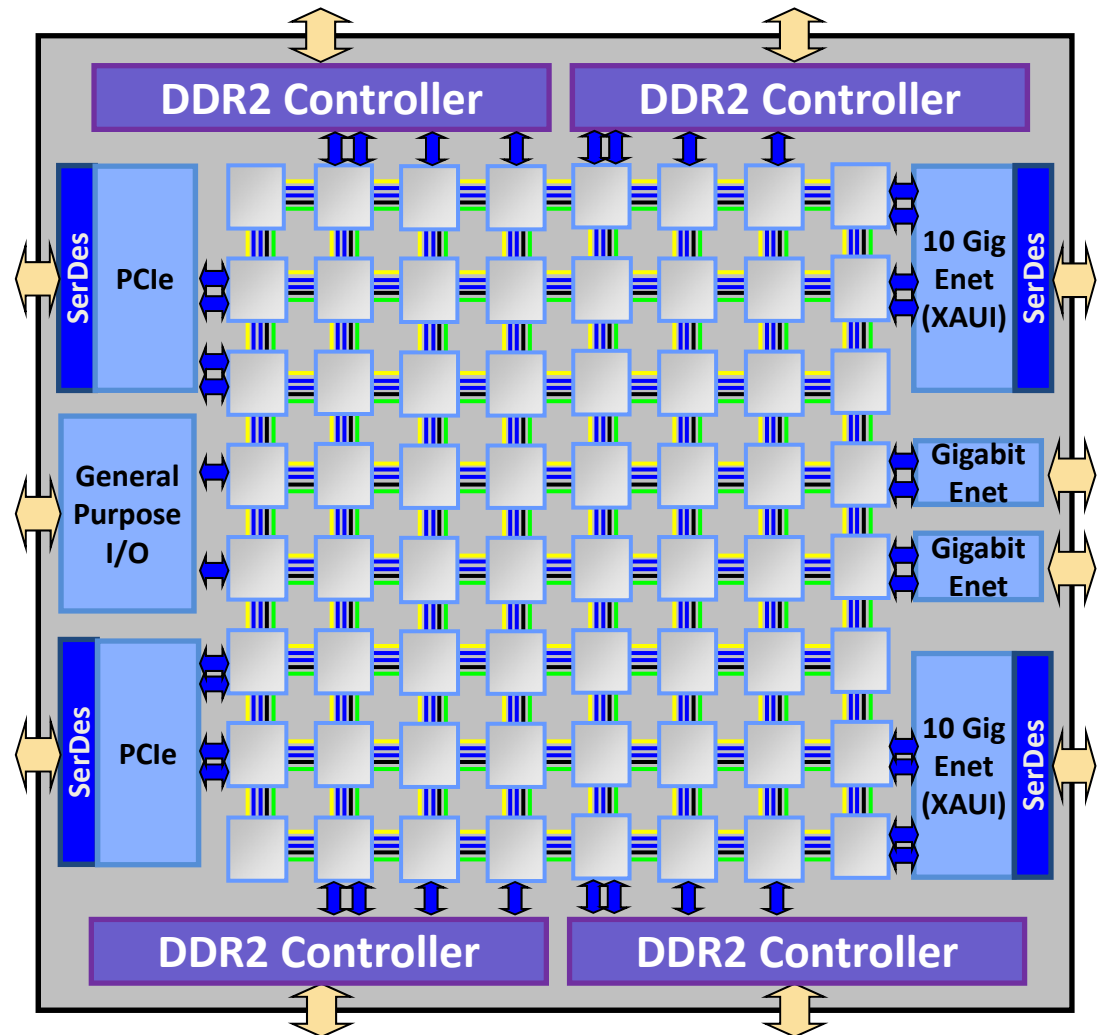
Topology: 2D Torus



Image Credit: SGI

TILE64Pro

- 64 Cores
- 2D Mesh
- 4 Memory Controllers
- 3 Memory Networks
 - Divide different flows of traffic
- Each node can be a home



Beyond ELE 475

- Computer Architecture Research
 - International Symposium on Computer Architecture (ISCA)
 - International Symposium on Microarchitecture (MICRO)
 - Architectural Support for Programming Languages and Operating Systems (ASPLOS)
 - International Symposium on High Performance Computer Architecture (HPCA)
- Build some chips / FPGA
- Parallel Computer Architecture
- ELE 580A Parallel Computation (Princeton Only)
 - Graduate Level, Using Primary Sources

Acknowledgements

- These slides contain material developed and copyright by:
 - Arvind (MIT)
 - Krste Asanovic (MIT/UCB)
 - Joel Emer (Intel/MIT)
 - James Hoe (CMU)
 - John Kubiatowicz (UCB)
 - David Patterson (UCB)
 - Christopher Batten (Cornell)
- MIT material derived from course 6.823
- UCB material derived from course CS252 & CS152
- Cornell material derived from course ECE 4750

Copyright © 2012 David Wentzlaff