计算机体系结构

周学海
xhzhou@ustc.edu.cn
0551-63606864
中国科学技术大学

第7章　多处理器及线程级并行

7.1 引言

7.2 集中式共享存储器体系结构

7.3 分布式共享存储器体系结构

7.4 存储同一性

7.5 同步与通信

review 1/2

- **并行计算机体系结构：SISD, SIMD, MISD, MIMD**
- **MIMD 的通信模型及存储器结构**
  - 地址空间的组织模式：共享存储(多处理机) vs. 非共享存储(多计算机)
  - 通信模型：LOAD /STORE指令 vs. 消息传递
- **共享存储的MIMD结构**
  - 集中式共享存储（SMP） vs. 分布式共享存储（DSM）
- **共享存储器结构的存储器行为**
  - Cache一致性问题 (Coherence)：使得多处理机系统的Cache像单处理机的Cache一样对程序员而言是透明的
  - 存储同一性问题(Consistency)：在多线程并发执行的情况下，提供一些规则来定义正确的共享存储器行为。通常允许有多种运行顺序

review 2/2

- **Cache 一致性（定义）**
  - 处理器P对X写之后又对X进行读，读和写之间没有其它处理器对X进行写，则读的返回值总是**写进的新值**。
  - 处理器对X写之后，另一处理器对X进行读，读和写之间无其它写，则读X的返回值应为**写进的新值**。
  - 对同一单元的写是顺序化的，即**任意两个处理器对同一单元的两次写，从所有处理器看来顺序是相同的。**
- **共享数据块的跟踪：监听和目录**
  - Cache一致性协议实现：写作废和写更新
- **集中式共享存储体系结构**
  - Snoopy Cache-Coherence Protocols

7.2-2 集中式共享存储结构的Cache一致性协议

01 **基本思路**

02 MSI协议

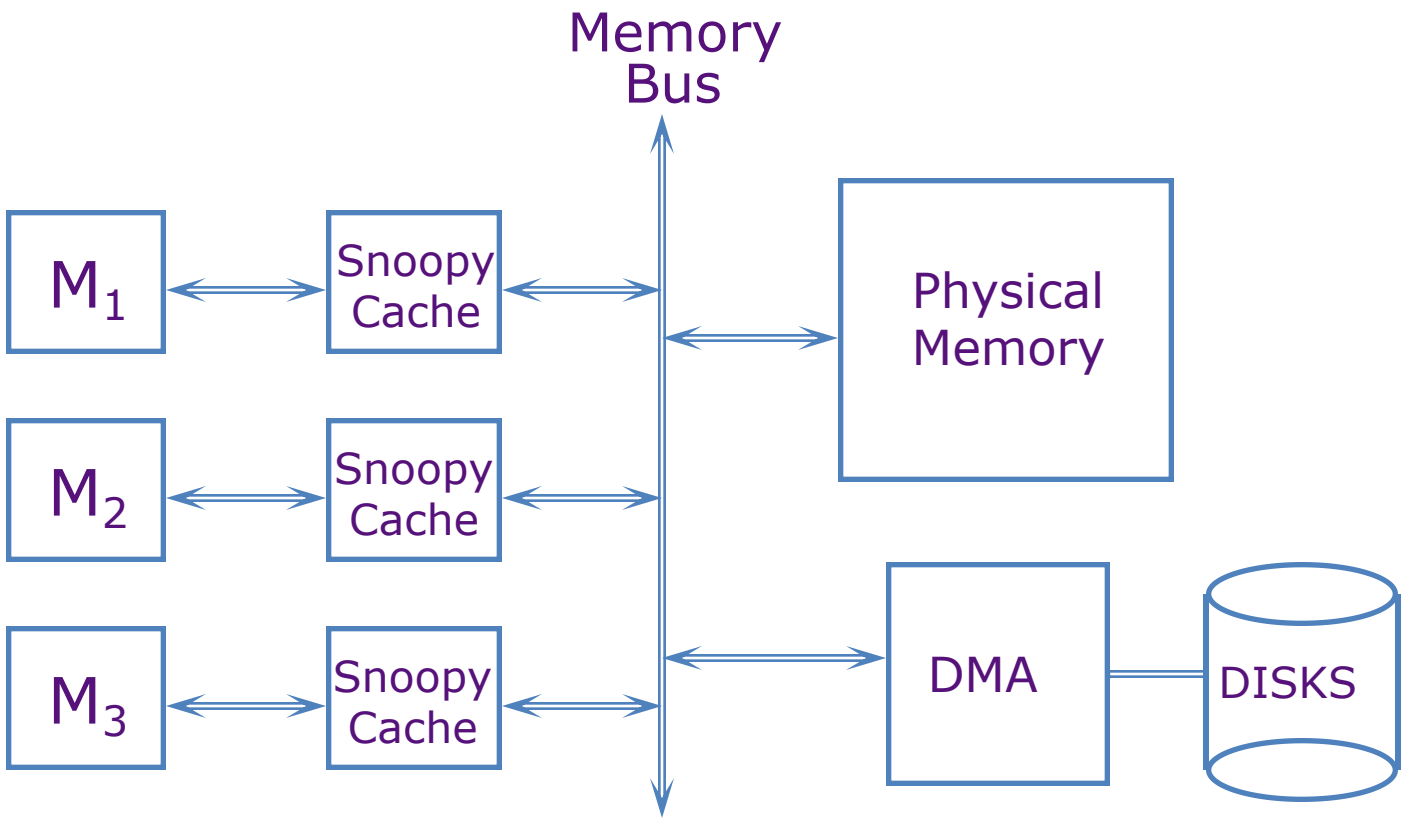
03 MESI、MOESI协议

04 Cache一致性引起的失效（缺失）

4. 监听协议的基本实现技术

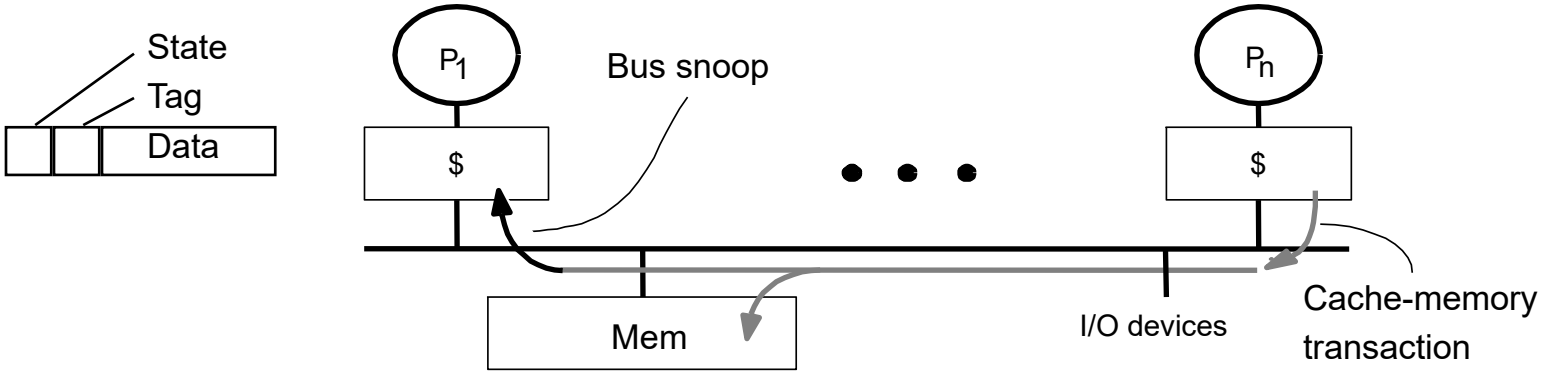
- **小规模多处理机中实现写作废协议的关键利用总线进行作废操作,每个块的有效位使作废机制的实现较为容易。**
- **写直达Cache，因为所有写的数据同时被写回主存，则从主存中总可以取到最新的数据值。**
- **对于写回Cache，得到数据的最新值会困难一些，因为最新值可能在某个Cache中，也可能在主存中。**
- **在写回Cache条件下的实现技术**
  - 用Cache中块的标志位实现监听过程。
  - 给每个Cache块加特殊的状态位说明它是否为共享。

Shared Memory Multiprocessor



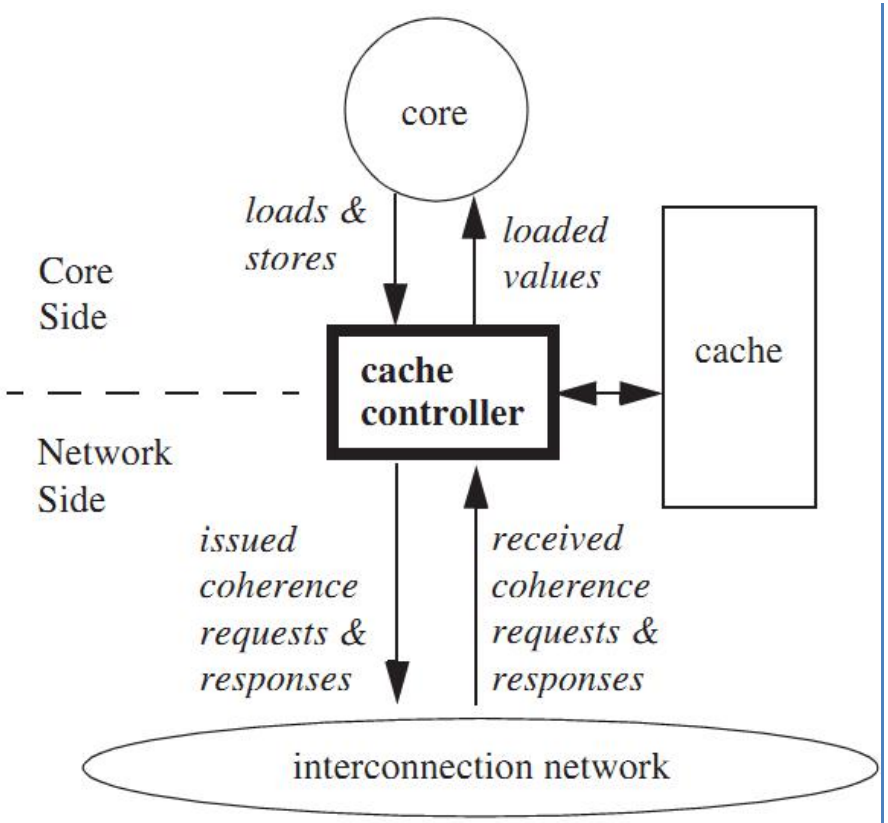
利用监听机制来保持处理器看到的存储器的视图一致

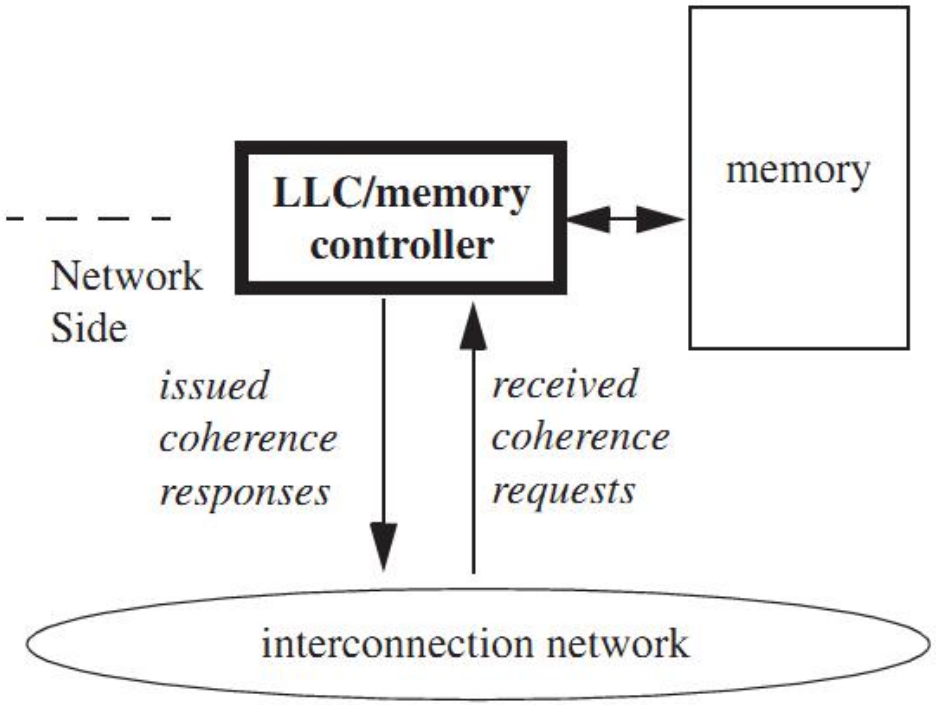
Snoopy Cache-Coherence Protocols



- **总线作为广播的媒介&Caches可知总线的行为**
  - 总线上的事务对所有Cache是可见的
  - 这些事务对所有控制器以同样的顺序可见
- **Cache 控制器监测 （snoop）共享总线上的所有事务**
  - 根据Cache中块的状态不同会产生不同的事务
  - **通过执行不同的总线事务来保证Cache的一致性**
    - Invalidate, update, or supply value

一致性的实现

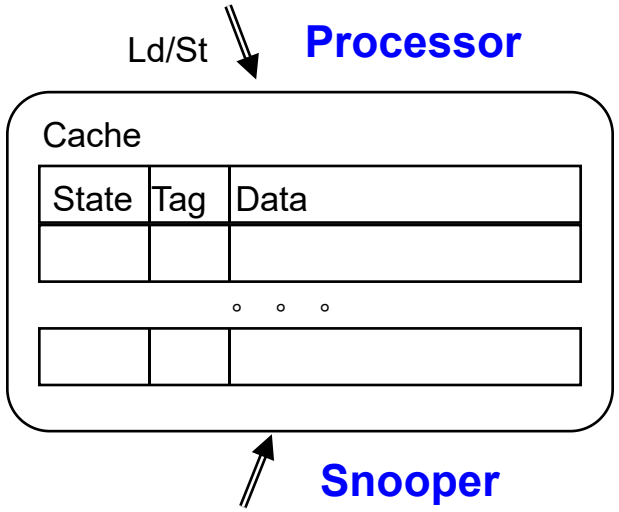




Coherence controller: 实现一组有限状态机 （逻辑上说，每个块对应一个独立的、但又规则一致的有限状态机）
左边：Cache controller 作为Coherence controller.
右边：memory controller 作为Coherence controller

Implementing a Snooping Protocol

- **Cache 控制器接收两方面的请求输入：**
  - 处理器的请求 (load/store)
  - 监测器 （snooper)的总线请求/响应
- **Cache 控制器根据这两方面的输入产生动作**
  - 更新Cache块的状态
  - 提供数据
  - 产生新的总线事务



7.2-2 集中式共享存储结构的Cache一致性协议

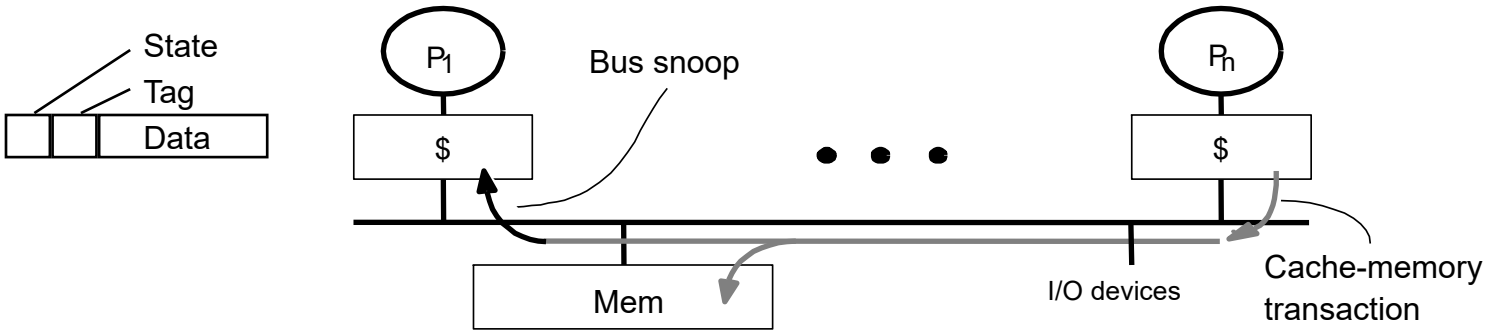
01 基本思路

02 **MSI协议**

03 MESI、MOESI协议

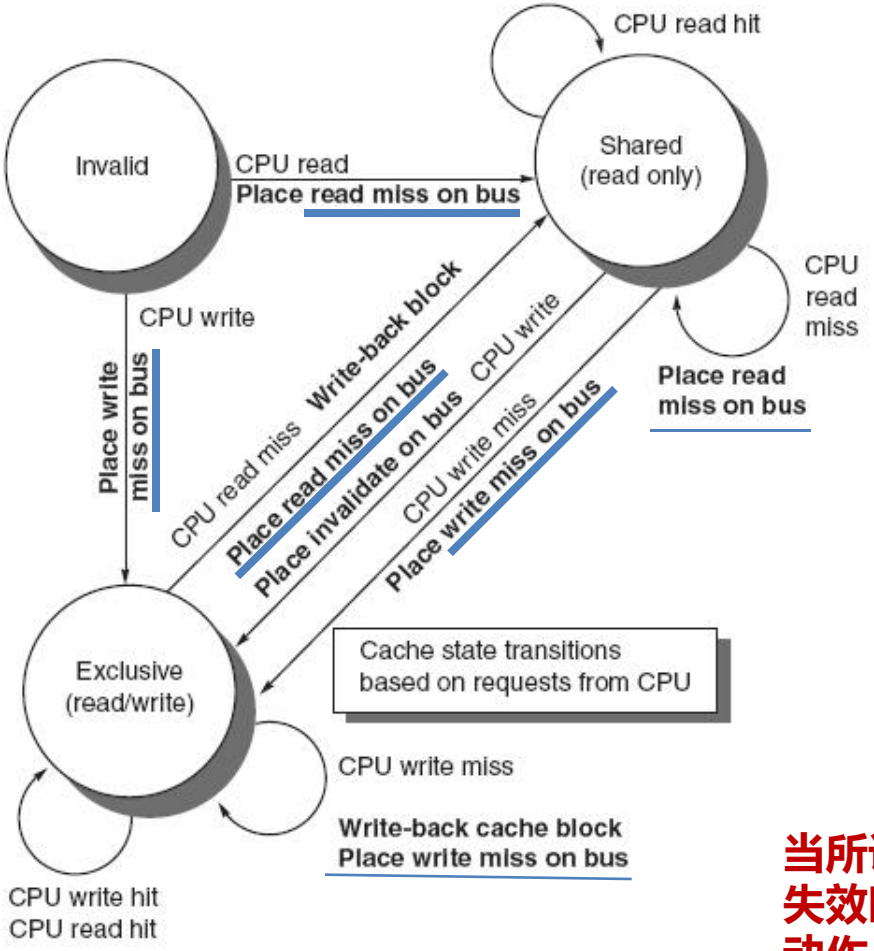
04 Cache一致性引起的失效（缺失）

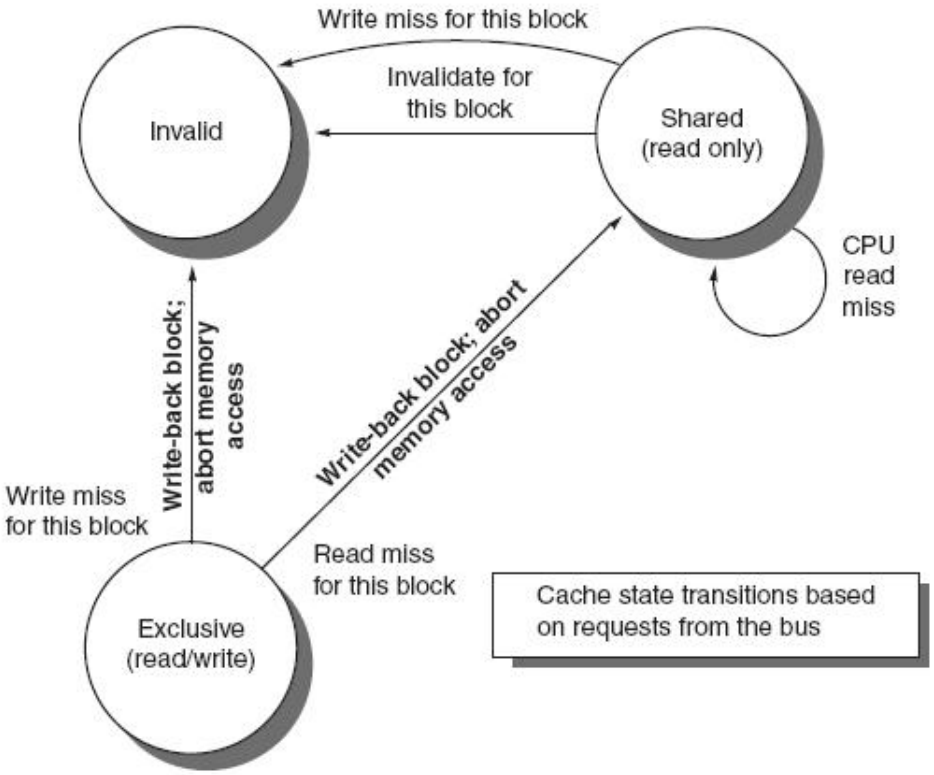
MSI Write-Back Invalidate Protocol



- **3 states:**
  - Modified: 仅该cache拥有修改过的、有效的该块copy
  - Shared: 该块是干净块，其他cache中也可能含有该块，存储器中的内容是最新的
  - Invalid: 该块是无效块（invalid）
- **4 bus transactions:**
  - Read Miss : 服务于Read Miss on Bus
  - Write Miss: 服务于Write Miss on Bus,得到一个独占的块
  - Invalidate: 作废该块在其他处理器中的Copy
  - Write back：替换操作将修改过的块写回
- **写操作时，作废所有其他块**
  - 当有多个副本，直到Invalidate transaction出现在总线上，写操作才算完成
  - 写串行化：总线事务在总线上串行化

MSI Snoopy Cache Coherence Protocol





当所访问的块的最新数据在某个私有Cache时，在读写失效时，数据的提供者是拥有该块数据的私有Cache。
动作：Write-back block; abort memory access

MSI Snoopy Cache Coherence Protocol

Request	Source	State Transition	Action and Explanation
Read Hit	Processor	Shared or Modified	Normal Hit: Read data in private data cache (no transaction)
Read Miss	Processor	Invalid → Shared	Normal Miss: Place **read miss on bus**, change state
Read Miss	Processor	Shared	Replace block: Place **read miss on bus**
Read Miss	Processor	Modified → Shared	**Write-Back** block, Place **read miss on bus**, change state
Write Hit	Processor	Modified	Normal Hit: Write data in private data cache (no transaction)
Write Hit	Processor	Shared → Modified	Coherence: Place **invalidate on bus** (no data), change state
Write Miss	Processor	Invalid → Modified	Normal Miss: Place **write miss on bus**, change state
Write Miss	Processor	Shared → Modified	Replace block: Place **write miss on bus**, change state
Write Miss	Processor	Modified	**Write-Back** block, Place **write miss on bus**
Read Miss	Bus	Shared	**Serve read miss** from shared cache or memory
Read Miss	Bus	Modified → Shared	Coherence: **Write-Back & Serve read miss**, change state
Invalidate	Bus	Shared → Invalid	Coherence: **Invalidate shared block** in other private caches
Write Miss	Bus	Shared → Invalid	Coherence: **Invalidate shared block** in other private caches
Write Miss	Bus	Modified → Invalid	Coherence: **Write-Back & Serve write miss, Invalidate**

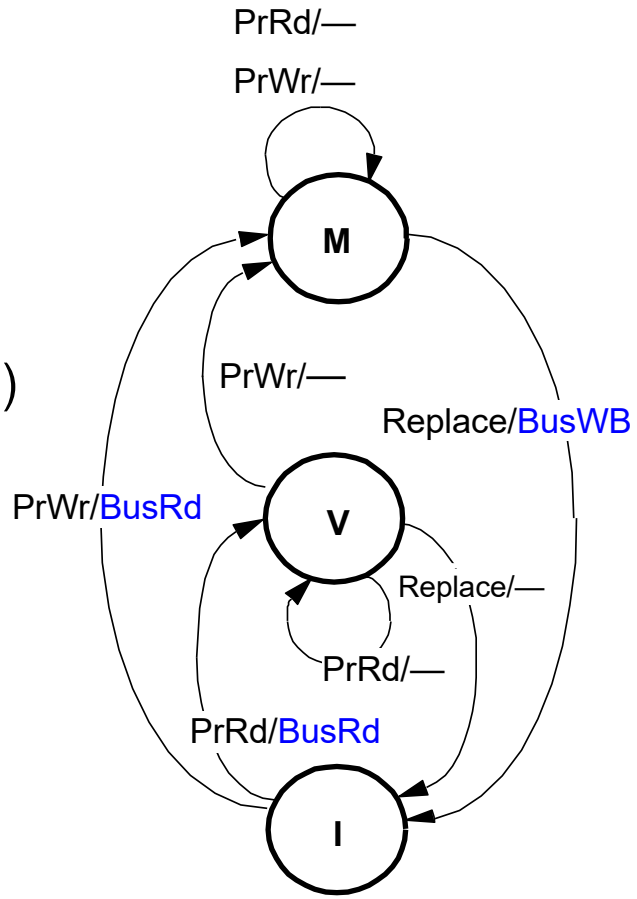
Example on MSI Cache Coherence

Request	Processor P1			Processor P2			Bus			Memory	
	State	Addr	Value	State	Addr	Value	Proc	Addr	Action	Addr	Value
P1: Write 10 to A1							P1	A1	Wr Miss	A1	15
	M	A1	10								
P1: Read A1 (Hit)	M	A1	10								
P2: Read A1							P2	A1	Rd Miss		
	S	A1	10				P1	A1	Wr Back	A1	10
				S	A1	10	P2	A1	Transfer		
P2: Write 20 to A1							P2	A1	Invalidate		
	I	A1	10	M	A1	20				A1	10
P2: Write 40 to A2				M	A2	40				A2	25

- Assume that A1 and A2 map to same cache block
- Initial cache state is invalid

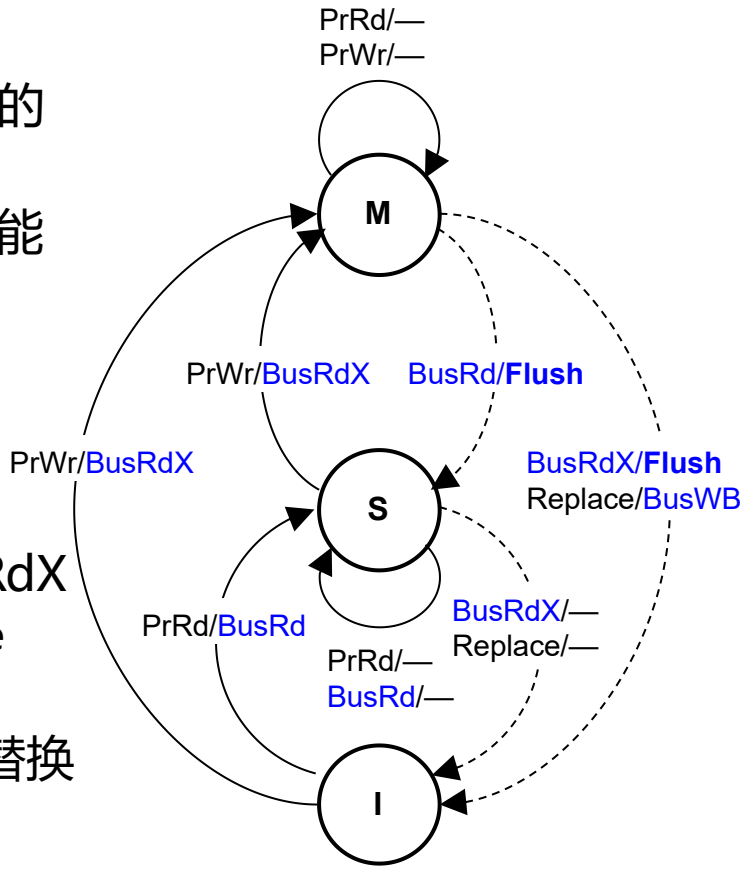
Write-back Cache

- **Cache块状态**
  - Invalid, Valid (clean), Modified (dirty)
- **Processor / Cache 操作**
  - PrRd, PrWr, block Replace
- **总线事务**
  - Bus Read (BusRd), Write-Back (BusWB)
  - 仅传送cache-block
- **针对Cache一致性的块状态调整**
  - Treat Valid as Shared
  - Treat Modified as Exclusive
- **引入新的总线事务**
  - Bus Read-eXclusive (BusRdX)
  - BusRdX -> Bus read with intention to write



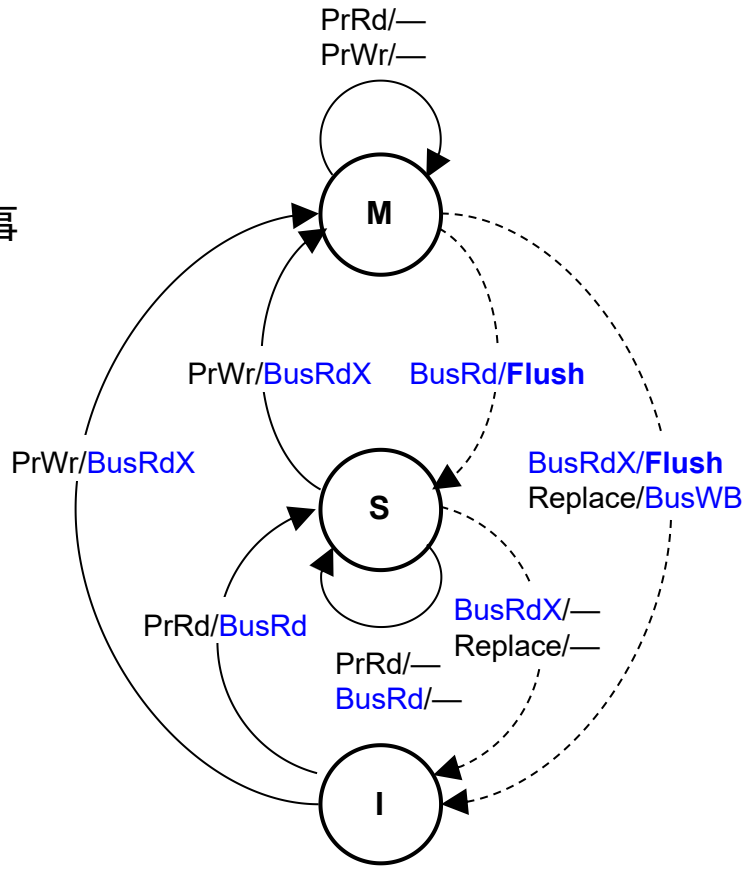
MSI Write-Back Invalidate Protocol

- **3 states:**
  - Modified: 仅该cache拥有修改过的、有效的该块copy
  - Shared: 该块是干净块，其他cache中也可能含有该块，存储器中的内容是最新的
  - Invalid: 该块是无效块（invalid）
- **4 bus transactions:**
  - Bus Read: 读失效时产生BusRd总线事务
  - Bus Read Exclusive（总线排他读）: BusRdX
    - BusRdX -> Bus read with intention to write
    - 得到独占的（exclusive）cache block
  - Bus Write-Back: BusWB用于cache 块的替换
  - Flush on BusRd or BusRdX
    - Cache将数据块放到总线上（而不是从存储器取数据）完成 Cache-to-cache的传送，并更新存储器

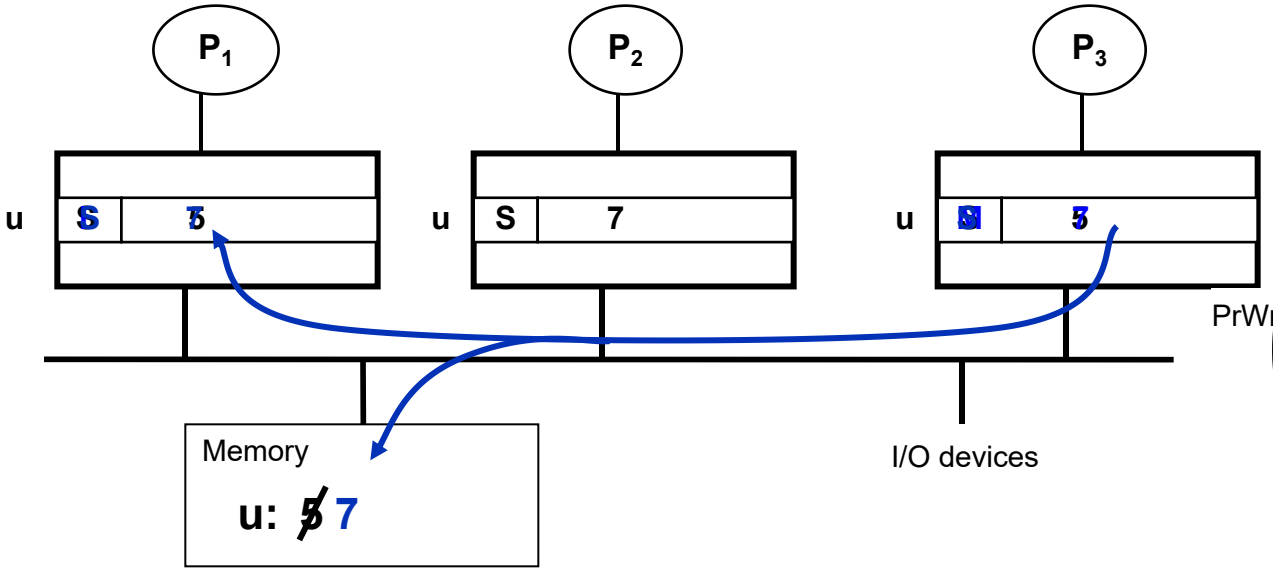


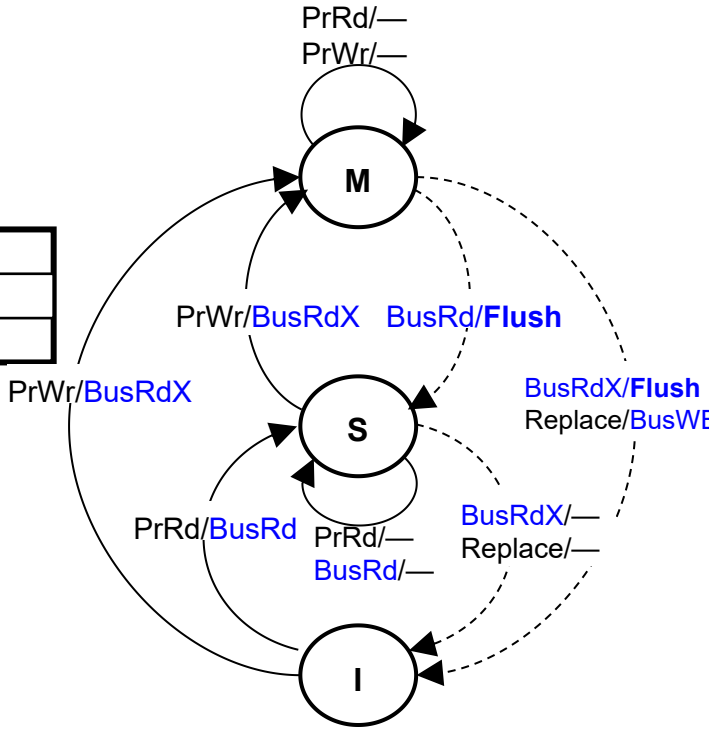
State Transitions in the MSI Protocol

- **Processor Read**
  - Cache miss ⇒ 产生BusRd事务
  - Cache hit（S or M) ⇒ 无总线动作
- **Processor Write**
  - 当在非Modified状态时，产生总线BusRdX 事务，BusRdX 导致其他Cache中的对应块作废（invalidate）
  - 当在Modified状态时，无总线动作
- **Observing a Bus Read**
  - 如果该块是 Modified, 产生Flush总线事务
    - 更新存储器和有需求的Cache
    - 引起总线事务的Cache块状态⇒ Shared
- **Observing a Bus Read Exclusive**
  - 作废相关block
  - 如果该块是modified, 产生Flush总线事务



Example on MSI Write-Back Protocol





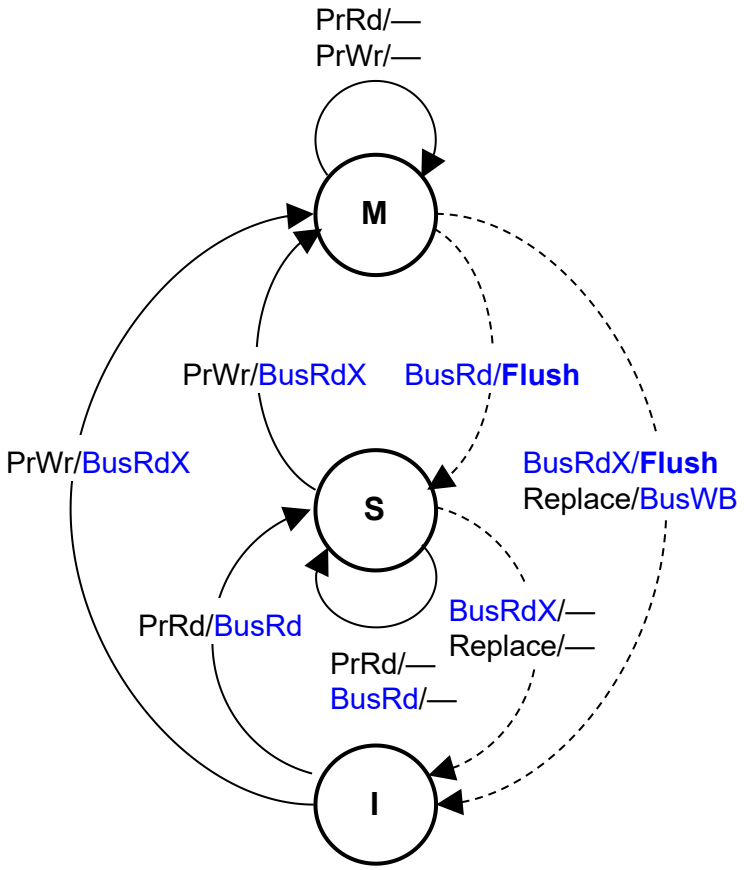
Processor Action	State P1	State P2	State P3	Bus Action	Data from
1. P1 reads u	S			BusRd	Memory
2. P3 reads u	S		S	BusRd	Memory
3. P3 writes u	I		M	BusRdX	Memory
4. P1 reads u	S		S	BusRd, Flush	P3 cache
5. P2 reads u	S	S	S	BusRd	Memory

Lower-level Design Choices

- **当 M态的块观察到BusRd 时，变迁到哪个态**
  - M → S or M → I 取决于访问模式
- **Transition to state S**
  - 如果不久会有本地读操作，而不是其他处理器的写操作
  - 比较适合于经常发生读操作的访问模式
- **Transition to state I**
  - 经常发生其他处理器写操作
  - 比较适合数据迁移操作：即本地写后，其他处理器将会发出读和写请求，然后本地又进行读和写。即连续的对称式访问模式。
- **不同选择方案会影响存储器的性能**

Satisfying Coherence

- **写传播(Write propagation)**
  - 对一个shared 或invalid块的写，其他cache都可见
    - 使用Bus Read-exclusive (BusRdX) 事务，Bus Read-exclusive 事务作废其他Cache中的块
    - 其他处理器在未看到该写操作的效果前体验到的是Cache Miss
- **写串行(Write serialization)**
  - 所有出现在bus上的写操作(BusRdX)被总线串行化
    - 所有处理器（包括发出写操作的处理器）以同样的方式排序
    - 首先更新发出写操作的处理器的本地cache，然后处理其他事务
  - 并不是所有的写操作都会出现在总线上
    - 对modified 块的写序列来自同一个处理器（P）将不会产生总线事务
    - 同一处理器是串行化的写：由P进行读操作将会看到串行序的写序列
    - 其他处理器对该块的读操作：会导致一个总线事务，这保证了写操作的顺序对其他处理器而言也是串行化的。



7.2-2 集中式共享存储结构的Cache一致性协议

01 基本思路

02 MSI协议

03 **MESI、MOESI协议**

04 Cache一致性引起的失效（缺失）

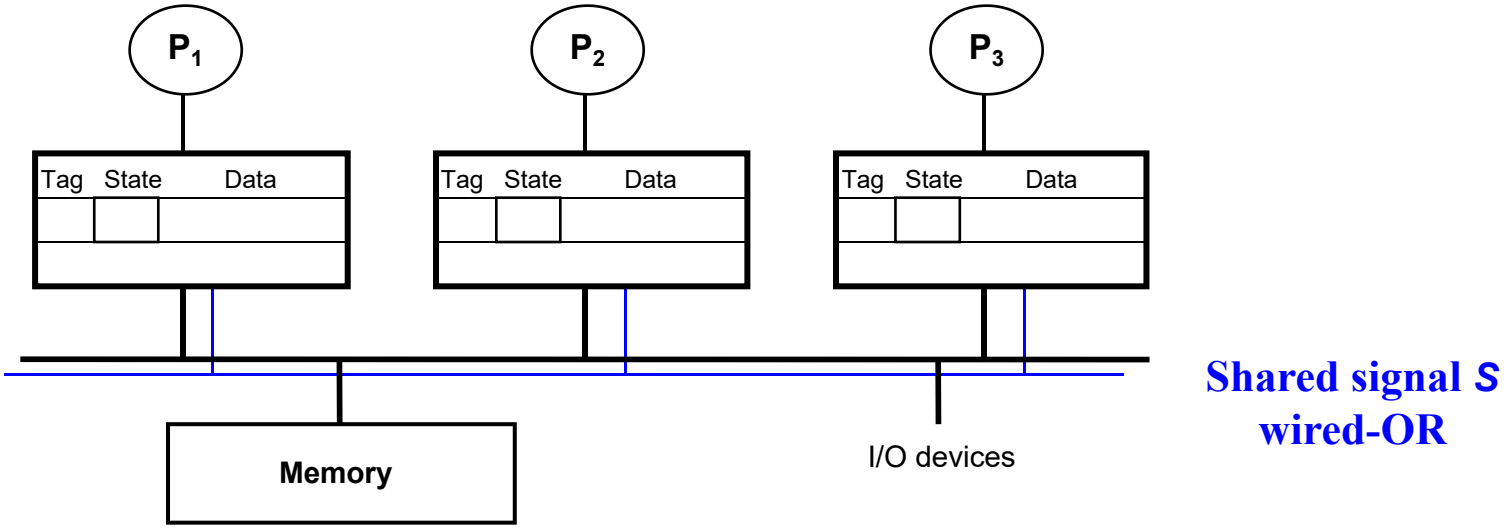
MESI Write-Back Invalidation Protocol

- **MSI Protocol的缺陷：**
  - 读进并修改一个block，产生2 个总线事务
    - 首先是读操作产生 BusRd (I→S)，并置状态为Shared, 写更新时产生 BusRdX (S→M)
    - 即使一个块是Cache独占的这种情况仍然存在
    - 使用多道程序负载时，这种情况很普遍
- **增加exclusive state, 减少总线事务**
  - Exclusive state 表示仅当前Cache包含该块，并且是干净的块
  - 区分独占块的“clean”和“dirty”
  - 一个处于exclusive state的块，更新时不产生总线事务

Four States: MESI

- **M: Modified**
  - 仅当前Cache含有该块，并且该块被修改过
  - 内存中的Copy是陈旧的值
- **E: Exclusive or exclusive-clean**
  - 仅当前Cache含有该块，并且该块没被修改过
  - 内存中的数据是最新的
- **S: Shared**
  - 多个Cache中都含有本块，而且都没有修改过
  - 内存中的数据是最新的
- **I: Invalid**
- **也称Illinois protocol**
  - 首先是由Illinois的研究人员研制并发表论文
  - MESI协议的变种广泛应用于现代微处理器中

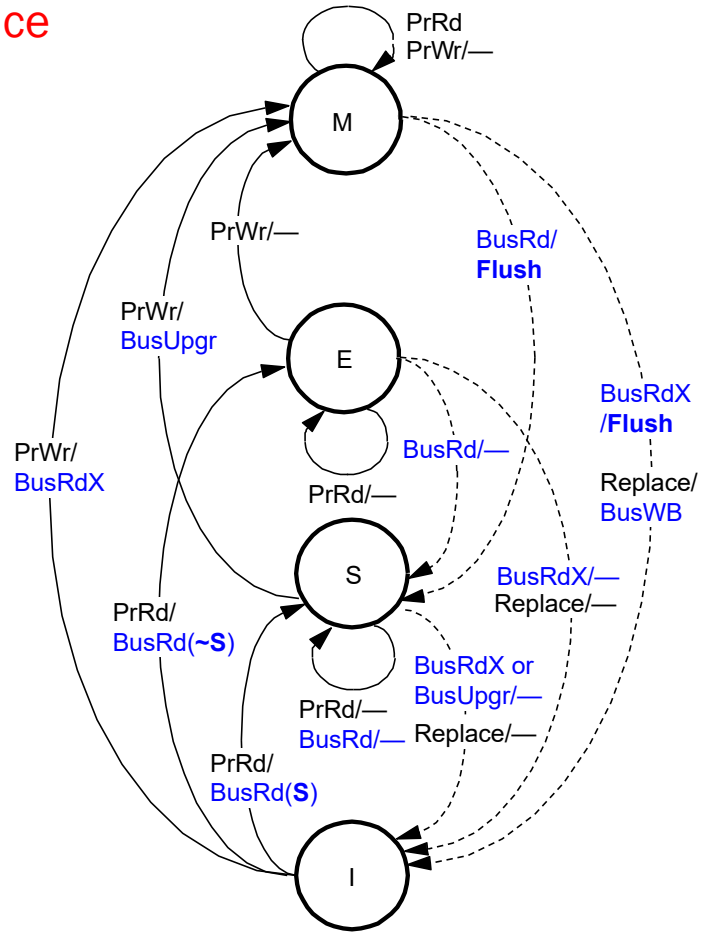
Hardware Support for MESI



- **总线互连的新要求**
  - 增加一个称为shared signal S, 必须对所有Cache控制器可用
  - 可以实现成 wired-OR line
- **所有cache controllers 监测 BusRd**
  - 如果所访问的块的状态是（state S, E, or M)
  - 请求Cache 根据shared signal选择E或S

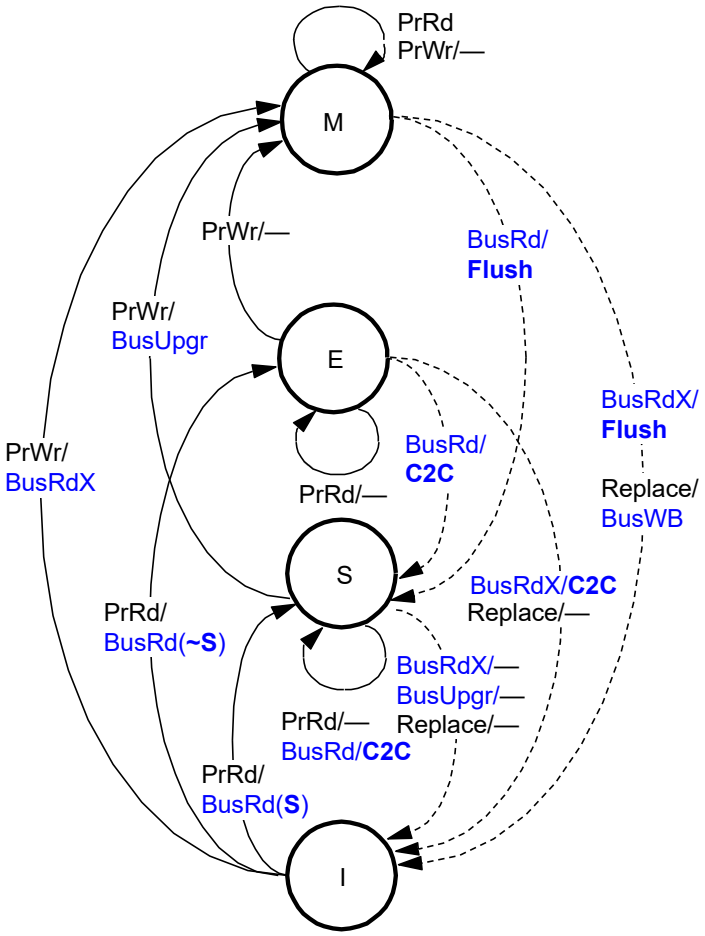
MESI State Transition Diagram

- **Processor Read**
  - 读失效时产生BusRd事务，进而有可能产生replace动作?
  - BusRd(S) => shared line asserted
    - 在其他Cache中有有效的copy
    - Goto state S
  - BusRd(~S) => shared line not asserted
    - 在其他Cache中不存在该块
    - Goto state *E*
  - 读命中时不产生总线事务
- **Processor Write**
  - 该Cache块的状态转至 *M*
  - 在I或S态(命中）产生 BusRdX / BusUpgr
    - 作废其他Cache中的Copies
  - Cache块处于状态E 和M时，不产生总线事务
  - 写失效时，E或S态的块有可能引起replace动作
  - 写失效时，M态的块有可能引起replace和BusWB



MESI State Transition Diagram – cont’d

- **Observing a BusRd**
  - 该块的状态从E降至 S
    - 因为存在其他copy
  - 该块从*M降至*S
    - 将引起更新过的块刷新操作
    - 刷新内存和其他有需求的Cache
- **Observing a BusRdX or BusUpgr**
  - 将作废相应的 block
  - 对于处于modified状态的块，将产生flush事务
  - BusUpgr：仅引起作废其他块，不产生读块操作
- **Cache-to-Cache (C2C) Sharing**
  - 原来的Illinois version支持这种共享
  - 由Cache 提供数据，而不是由内存提供数据



MESI Lower-level Design Choices

- **在E或S态时，由谁为BusRd/BusRdx事务提供数据**
  - Original, Illinois MESI: cache, 因为它假设Cache比memory更快
- **但 cache-to-cache 共享增加了实现的复杂性**
  - 这种实现的代价高于从memory获取数据
  - 存储器如何知道它该提供数据 (must wait for caches)
  - 如果多个Cache共享数据，要有Selection过程
- **当块状态为Modified，总线上的刷新（Flushing）数据操作**
  - 需要更新的块以及存储器 接收数据，但存储器速度比Cache的速度慢。
  - 是否可以仅让Cache接收数据，memory不接收数据?
    - 这就要求第5个状态: Owned state ⇒ MOESI Protocol
    - Owned 态是共享的Modified态，此时存储器不是最新数据
    - 该块可以被多个Cache共享，但所有者（owner)只有一个

MOESI中的Owned 和Shared 状态

- **Owned位。**
  - O位为1表示在当前Cache 块中包含的数据是当前处理器系统最新的数据拷贝，而且在其他CPU中一定具有该Cache块的副本，其他CPU的Cache块状态为S。
  - 如果存储器的数据在多个CPU的Cache中都具有副本时，有且仅有一个CPU的Cache块状态为O，其他CPU的Cache块状态只能为S。
  - 与MESI协议中的S状态不同，状态为O的Cache块中的数据与存储器中的数据并不一致。
- **Shared位。**
  - 当Cache块状态为S时，其包含的数据并不一定与存储器一致。
  - 如果在其他CPU的Cache中不存在状态为O的副本时，该Cache块中的数据与存储器一致；
  - 如果在其他CPU的Cache中存在状态为O的副本时，Cache块中的数据与存储器不一致。

7.2-2 集中式共享存储结构的Cache一致性协议

01 基本思路

02 MSI协议

03 MESI、MOESI协议

04 **Cache一致性引起的失效（缺失）**

Performance of Symmetric Shared-Memory Multiprocessors

- **Cache performance 由两部分构成：**
  - 单处理器 cache miss的通信量（Traffic）
  - 通信引起的通信量：由于作废机制导致后面的访问失效
- **Coherence misses**
  - 有时也称为 Communication miss
  - 4th C of cache misses along with Compulsory, Capacity, & Conflict.

Coherency Misses

- **由于对共享块的写操作引起**
  - 共享块在多个本地Cache有副本
  - 当某处理器对共享块进行写操作时会 作废其他处理器的本地Cache的副本
  - **其他处理器对共享块进行读操作时 会有**coherence miss
- **True sharing misses： Cache coherence 机制引起的数据通信**
  - 通常是不同的处理器写或读 同一个变量
  - **对共享块（S态块）某一字的第一次写操作作废其他cache中的共享块**
  - **处理器试图读一个存在于其他处理器的cache中并且已经修改过的字（modified），这会导致失效，并将当前cache中的对应块写回**
  - **即时块大小为1个字，失效仍然会发生**
- **False sharing misses： 由于某个字在某个失效块中**
  - 读写同一块中的不同变量
  - 失效并没有通过通信产生新的值，仅仅是产生了额外的失效
  - 块是共享的，但块中没有真正共享的字
    ⇒**如果块的大小为1个字，那么就不会产生这种失效**

Example of True & False Sharing Misses

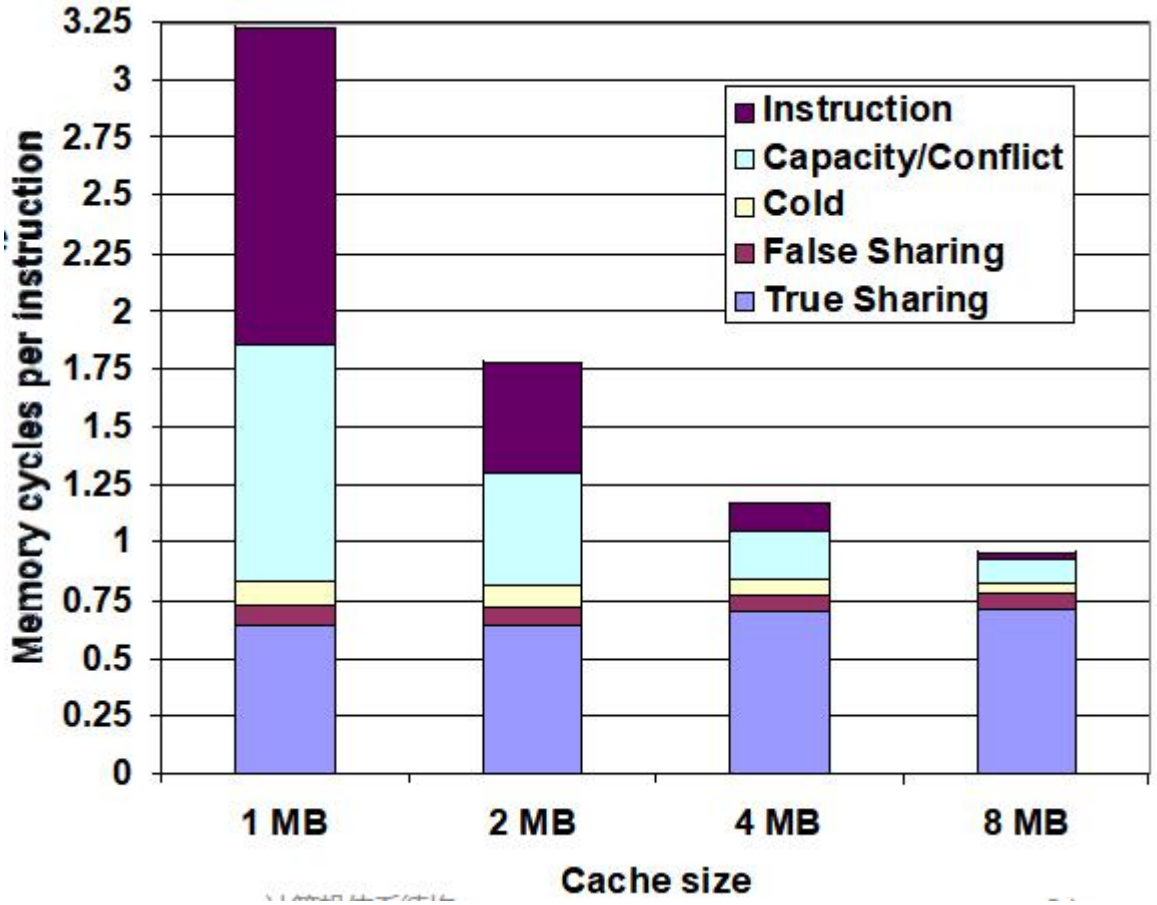
变量X和Y 属于同一cache块

初始状态为： P1和P2均读取了共享变量X, Block(X, Y) 在P1, P2中处于Shared 态

Request	P1 Cache State	P2 Cache State	Explanation
P1: Write X	Shared (X , Y)	Shared (X , Y)	True Sharing Miss (P2 read X)
	Modified (X , Y)	Invalid (X , Y)	P1 invalidates block (X , Y) in P2
P2: Read Y	Modified (X , Y)	Invalid (X , Y)	False Sharing Miss (Y not modified)
	Shared (X , Y)	Shared (X , Y)	Write-Back & Copy block from P1 to P2
P1: Write X	Shared (X , Y)	Shared (X , Y)	False Sharing Miss (P2 did not read X)
	Modified (X , Y)	Invalid (X , Y)	P1 invalidates block (X , Y) in P2
P2: Write Y	Modified (X , Y)	Invalid (X , Y)	False Sharing Miss (P1 did not read Y)
	Invalid (X , Y)	Modified (X , Y)	Write-Back & Copy block from P1 to P2
P1: Read Y	Invalid (X , Y)	Modified (X , Y)	True Sharing Miss (P2 modified Y)
	Shared (X , Y)	Shared (X , Y)	Write-Back & Copy block from P2 to P1

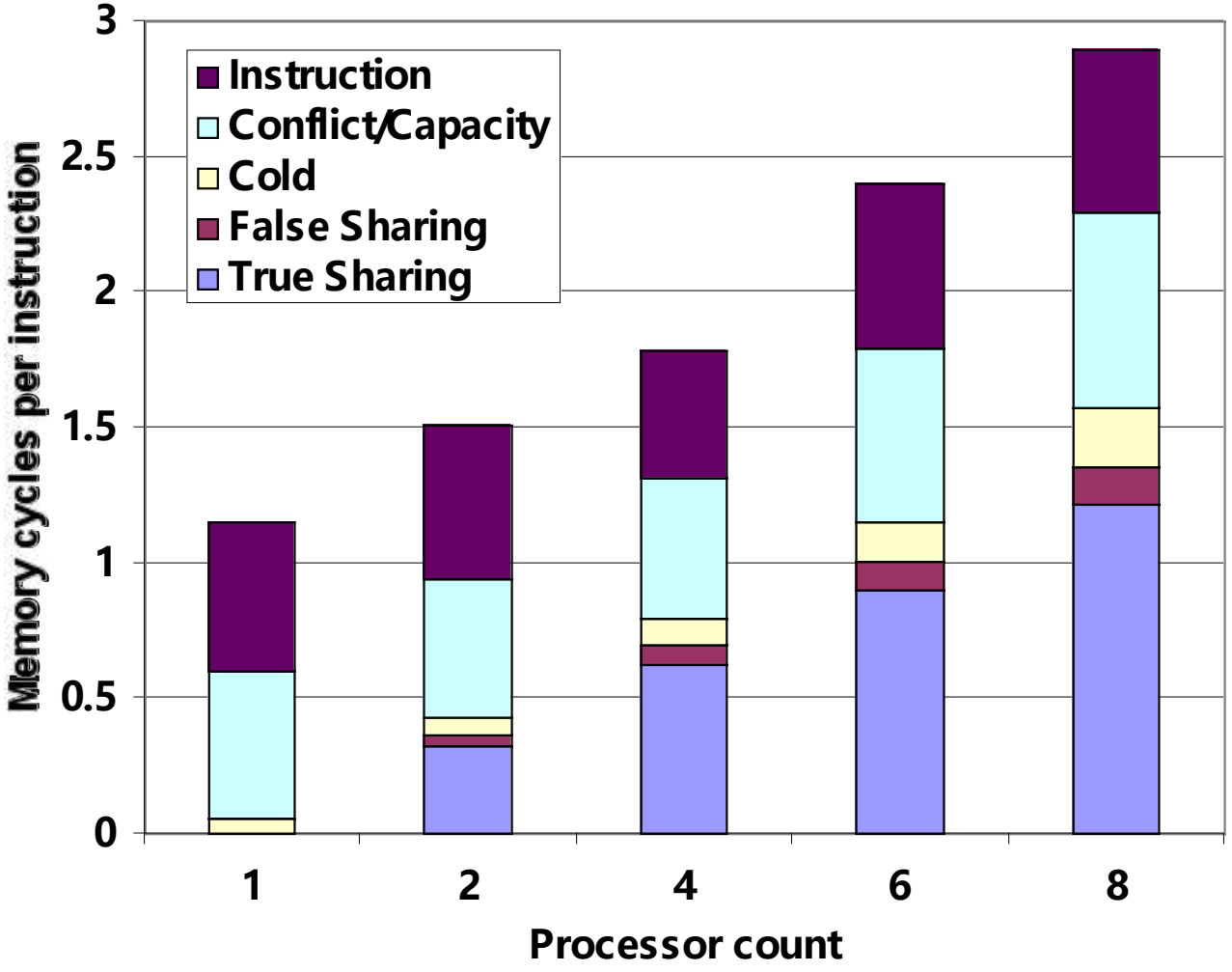
MP Performance 4 Processor Commercial Workload: OLTP, Decision Support (Database), Search Engine

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



MP Performance 2MB Cache
Commercial Workload: OLTP, Decision Support (Database), Search Engine

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



- 考虑一个8处理器的多核芯片，其中每个处理器都有自己的L1和L2缓存，并且在L2缓存之间的共享总线上执行窥探(snooping)。
  - 假设L2的平均访问时间是15个周期，不管是否存在一致性失效。
  - 假设时钟速率为3.0 GHz, CPI为0.7，Load/Store的频率为40%。
- 如果我们的目标是不超过50%的L2带宽被一致性失效流量消耗，那么每个处理器的最大一致性失效率是多少?
  - Cache cycles available = clock rate/CyclesPerRequest*50%
  - Cache cycles available = MemoryReferences/clock/processor * clock rate * processor count * CMR
  - CMR 为coherence miss rate

7.3 分布式共享存储器体系结构

01 **问题分析**

02 基于目录的一致性协议

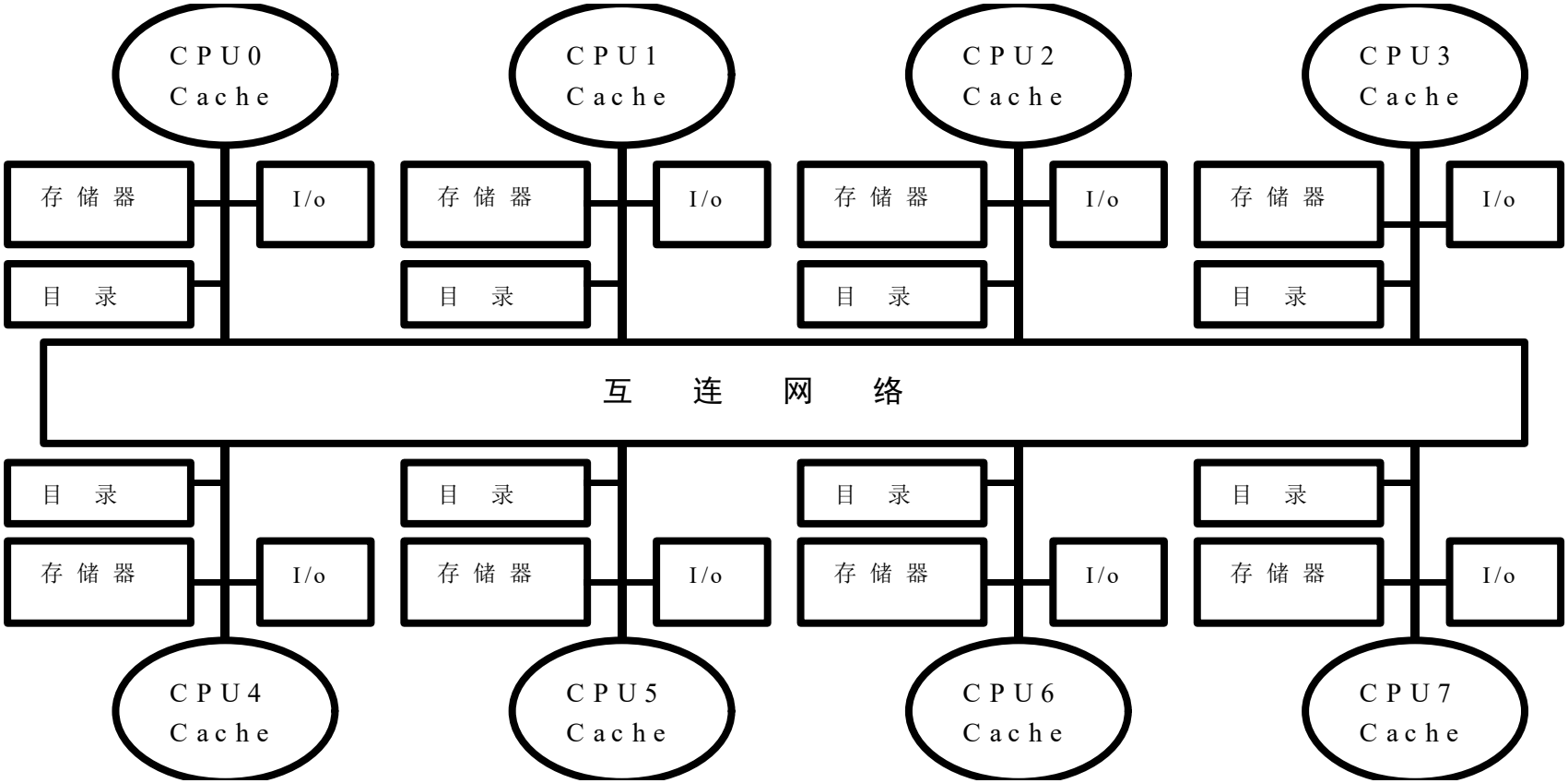
Limitations of Snooping Protocols

- **总线的可扩放性收到一定限制**
  - 总线上能够连接的处理器数目有限
  - 共享总线存在竞争使用问题
  - 在由大量处理器构成的多处理器系统中，监听带宽是瓶颈
- **解决方案之一：片上互连网络→并行通信**
  - 多个处理器可并行访问共享的Cache banks
  - 允许片上多处理器包含有更多的处理器
  - 可扩放性仍然受到限制。
- **在非总线或环的网络上监听是比较困难的**
  - 必须将一致性相关信息广播到所有处理器，这是比较低效的
- **如何不采用广播方式而保持 cache coherence**
  - 使用目录（directory)来记录每个 Cached 块的状态
  - 目录项说明了哪个私有Cache包含了该块的副本

解决Cache一致性问题的关键

- **寻找替代监听协议的一致性协议。**
- **目录协议**
  - 目录：用于记录共享块相关信息的数据结构，它记录着可以进入Cache的每个数据块的访问状态、该块在各个处理器的共享状态以及是否修改过等信息。
- **对每个结点增加目录表后的分布式存储器的系统结构**

分布式共享存储器体系结构



存储器分布于各结点中，所有的结点通过网络互连。访问可以是本地的，也可是远程的。

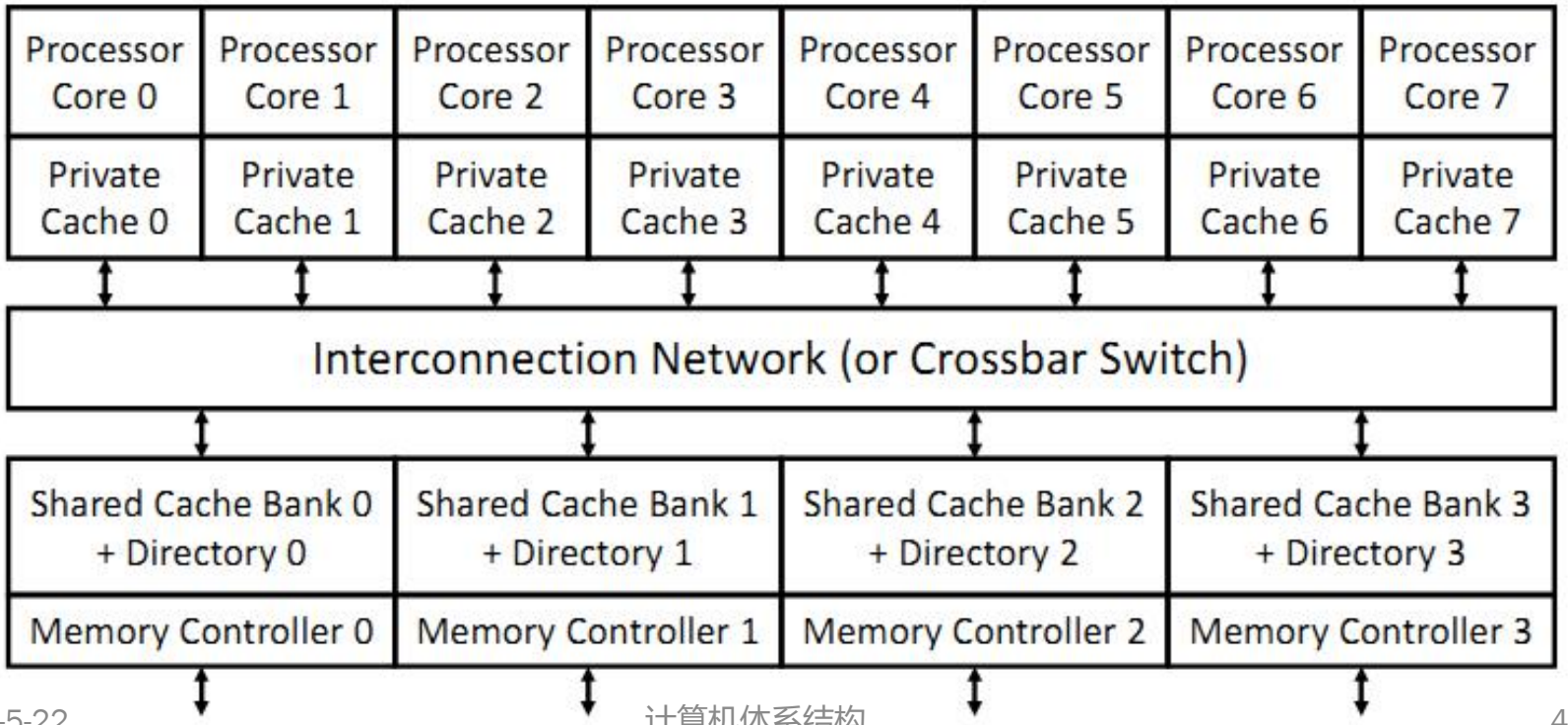
7.3 分布式共享存储器体系结构

01 问题分析

02 **基于目录的一致性协议**

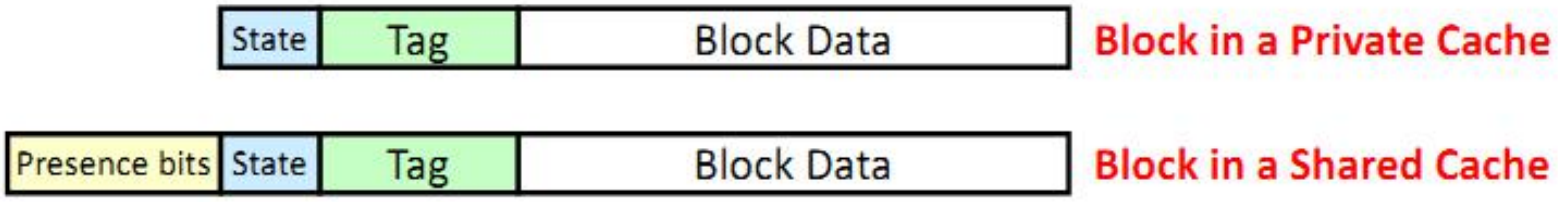
Directory in a Chip Multiprocessor

- **目录在所有处理器共享的最外层Cache中**
  - 目录记录了每个私有Cache中块的相关信息
- **最外层Cache分成若干个banks，以便并行访问**
  - Cache的banks数可以与cores的数量相同，也可以不同



Directory in the Shared Cache

- **Shared Cache 包含所有的私有Cache**
  - 共享Cache是私有cache块的超集
  - Example: Intel Core i7
- **目录在共享cache中**
  - 共享cache中的每个块增加若干presence bits
  - 如果有k个processors那么共享cache中每个块含有presence bits(k位) + state位
  - Presence bits 指示了包含该块copy的cores
  - 每个块都有其在私有cache和共享cache中的状态信息
  - 在私有Cache中块的状态：State = M (Modified), S (Shared), or I (Invalid)



一些术语

- **本地或私有Cache （Local (or Private) Cache)**
  - 处理器请求的源
- **目录（Home Directory)**
  - 存放Cache块相关信息
  - 目录使用presence bits 和 state 追踪cache块
- **远程Cache（Remote Cache)**
  - 该Cache中包含一个Cache块的副本，处于modified 或shared 态
- **Cache一致性：即要保证Single-Writer, Multiple-Readers**
  - 如果一个块在本地Cache中处于Modified态，那么只有一个有效的副本存在（共享的Cache和存储器还没有更新）
- **无总线，不用广播方式到所有处理器核**
  - 所有消息都有显式的回复

States for Local and Shared Cache

对于本地（私有）cache 块，存在3种状态:

1. Modified: 仅当前Cache具有该块修改过的副本

2. Shared: 该块可能在多个Cache中有副本

3. Invalid: 该块无效

对于共享Cache中的块，存在4种状态:

1. Modified: 只有一个本地Cache是这个块的拥有者

只有一个本地Cache具有该块修改后的副本

2. Owned: 共享Cache是modified块的拥有者

Modified block被写回到共享Cache，但不是内存

处于owned态的块可以被多个本地Cache共享

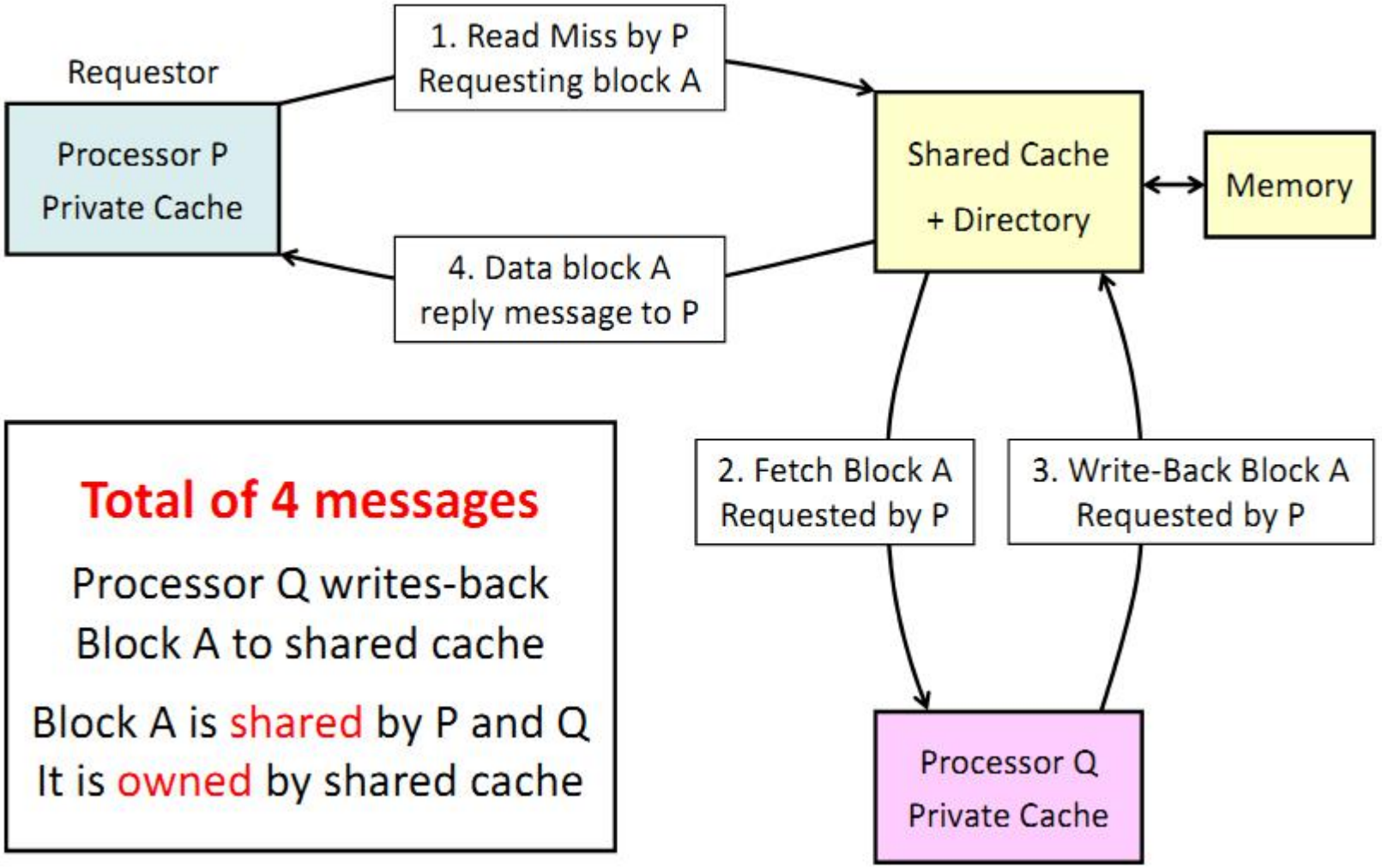
3. Shared: 该块可能被复制到多个cache中

4. Uncached: 该块不在任何本地或共享Cache中

Read Miss by Processor P

- **Processor P 发送 Read Miss 消息给 Home directory**
- **Home Directory: block 是 Modified态**
  - Directory 发送 Fetch message 给拥有该块的remote cache
  - Remote cache发送 Write-Back message 到 directory (shared cache)
  - Remote cache 将该块状态修改为shared
  - Directory 将其所对应的共享块状态修改为 owned
  - Directory 发送数据给P, 并将对应于P的presence bit置位
  - P的Local cache 将所接收到的块状态置为 shared
- **Home Directory: block 是Shared or Owned态**
  - Directory发送数据给P，并将对应 P的presence bit置位
  - P的Local cache 将所接收到的块状态置为 shared
- **Home Directory: Uncached -> 从存储器中获取块**

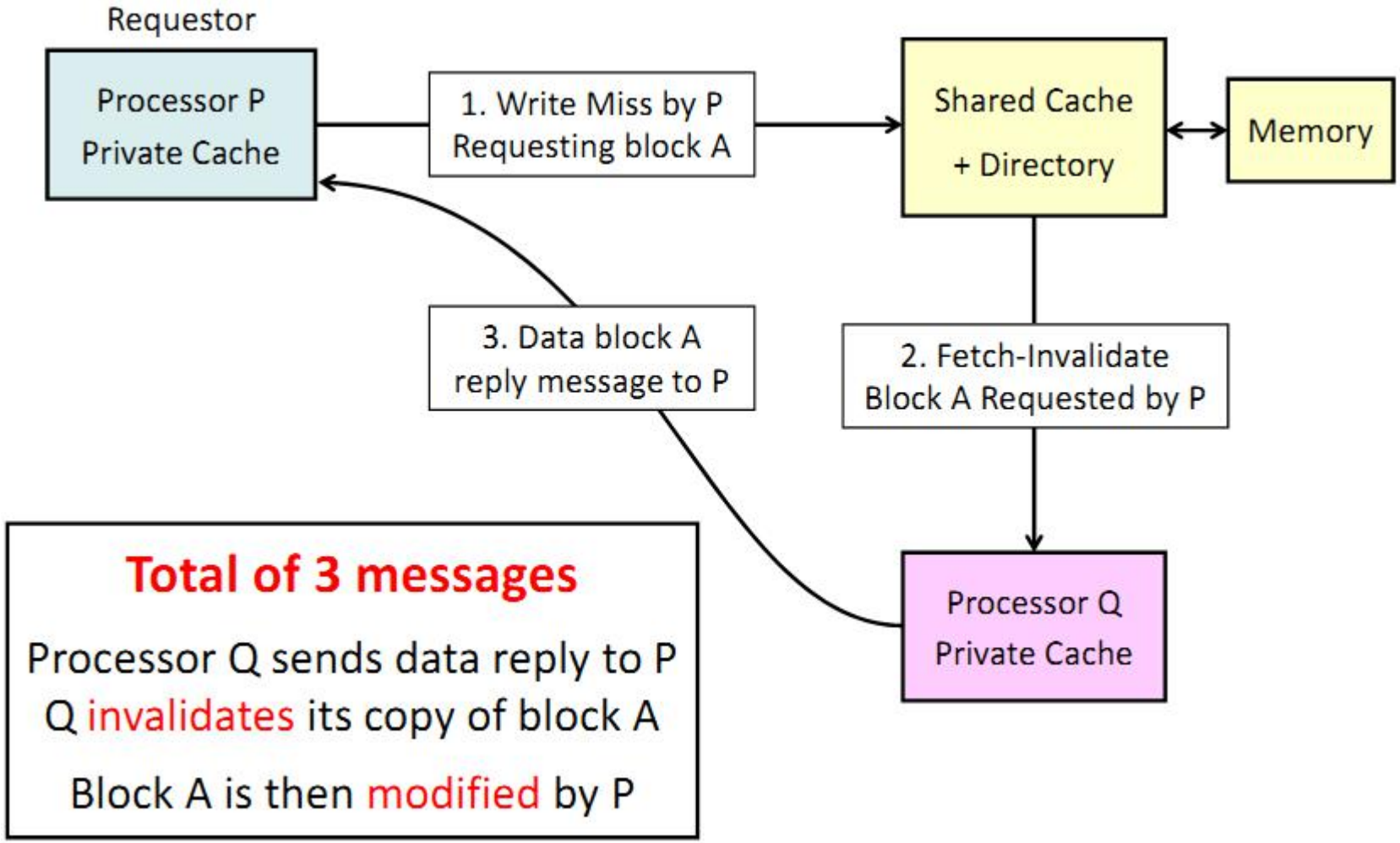
Read Miss to a Block in Modified State



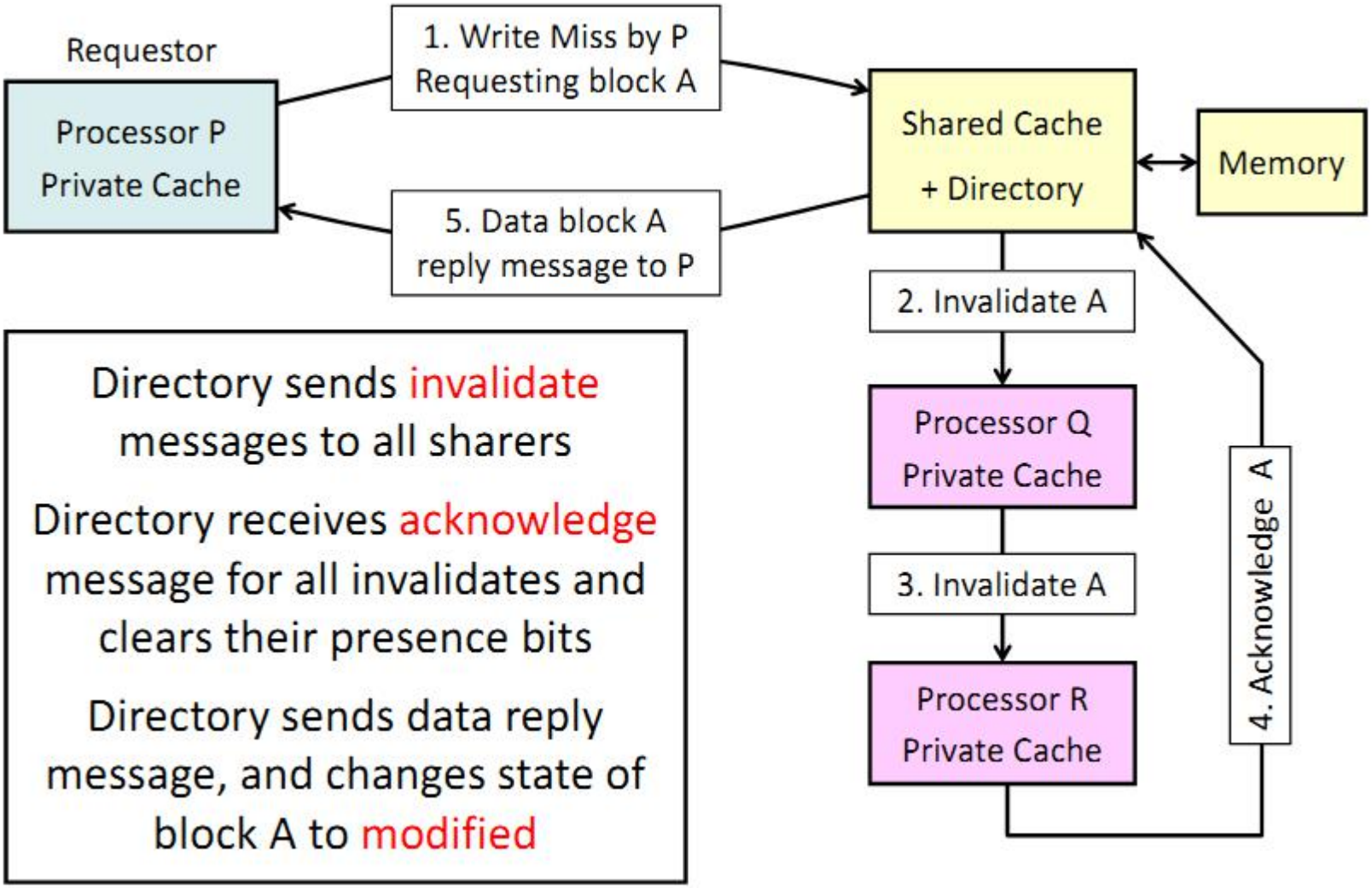
Write Miss Message by P to Directory

- **Home Directory: block 是Modified态**
  - Directory 发送 Fetch-Invalidate message 给处理器Q的Cache (Remote Cache)
  - 处理器Q的cache 直接发送数据应答消息给P
  - Q的cache将对应块的状态修改为invalid
  - P的cache (Local) 将接收到的块的状态信息修改为modified
  - Directory 将对应于Q的 presence bit复位，并将对应于P的 presence bit 置位
- **Home Directory: block 是 Shared or Owned态**
  - Directory 根据presence bit位给所有的共享者发送invalidate messages
  - Directory接收 acknowledge消息并将对应的presence bits复位
  - Directory 发送数据回复信息给P, 并将P对应的 presence bit 置位
  - P的cache 和directory 将该块的状态修改为 modified
- **Home Directory: Uncached -> 从存储器获取数据**

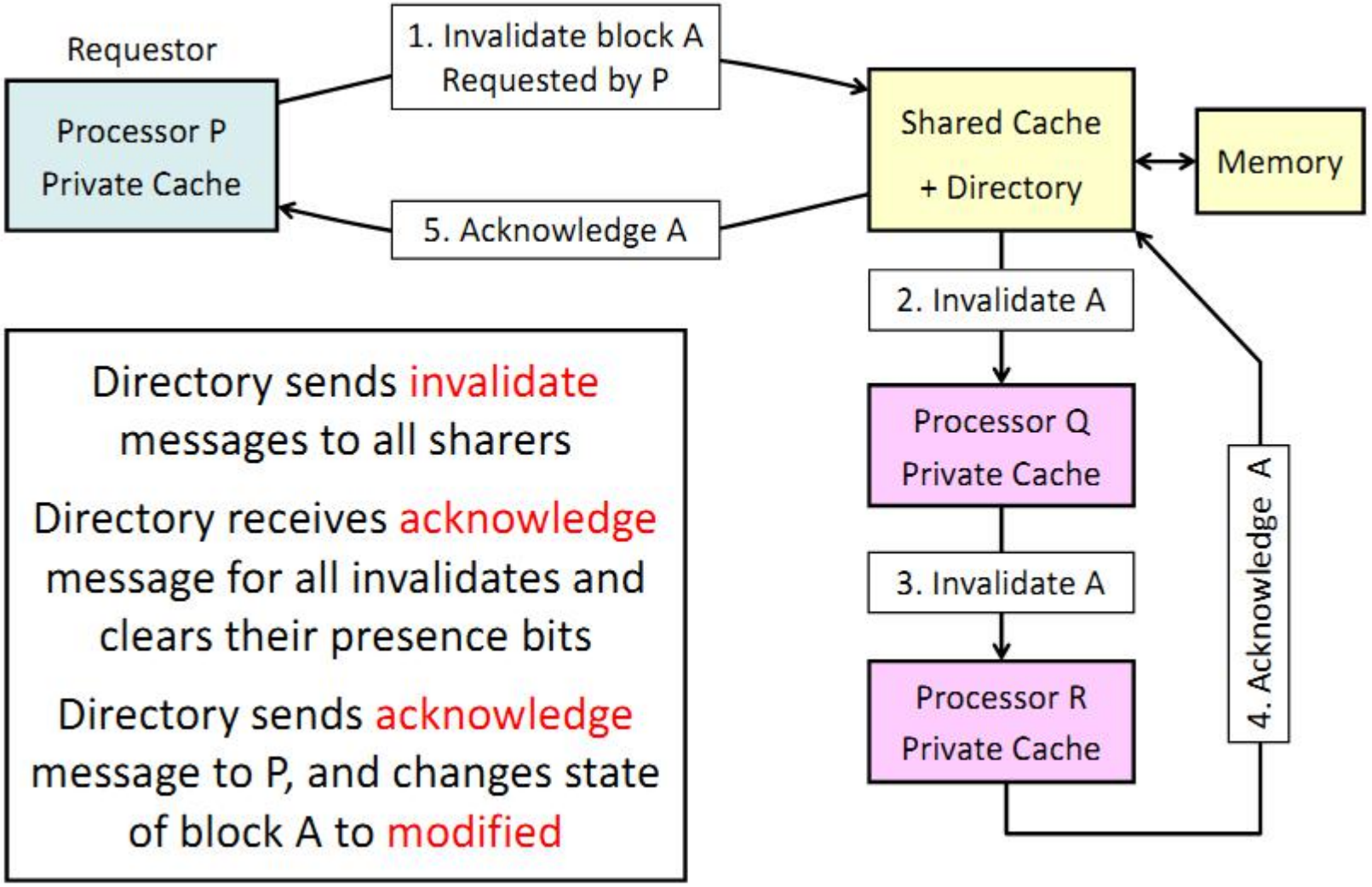
Write Miss to a Block in Modified State



Write Miss to a Block with Sharers



Invalidating a Block with Sharers

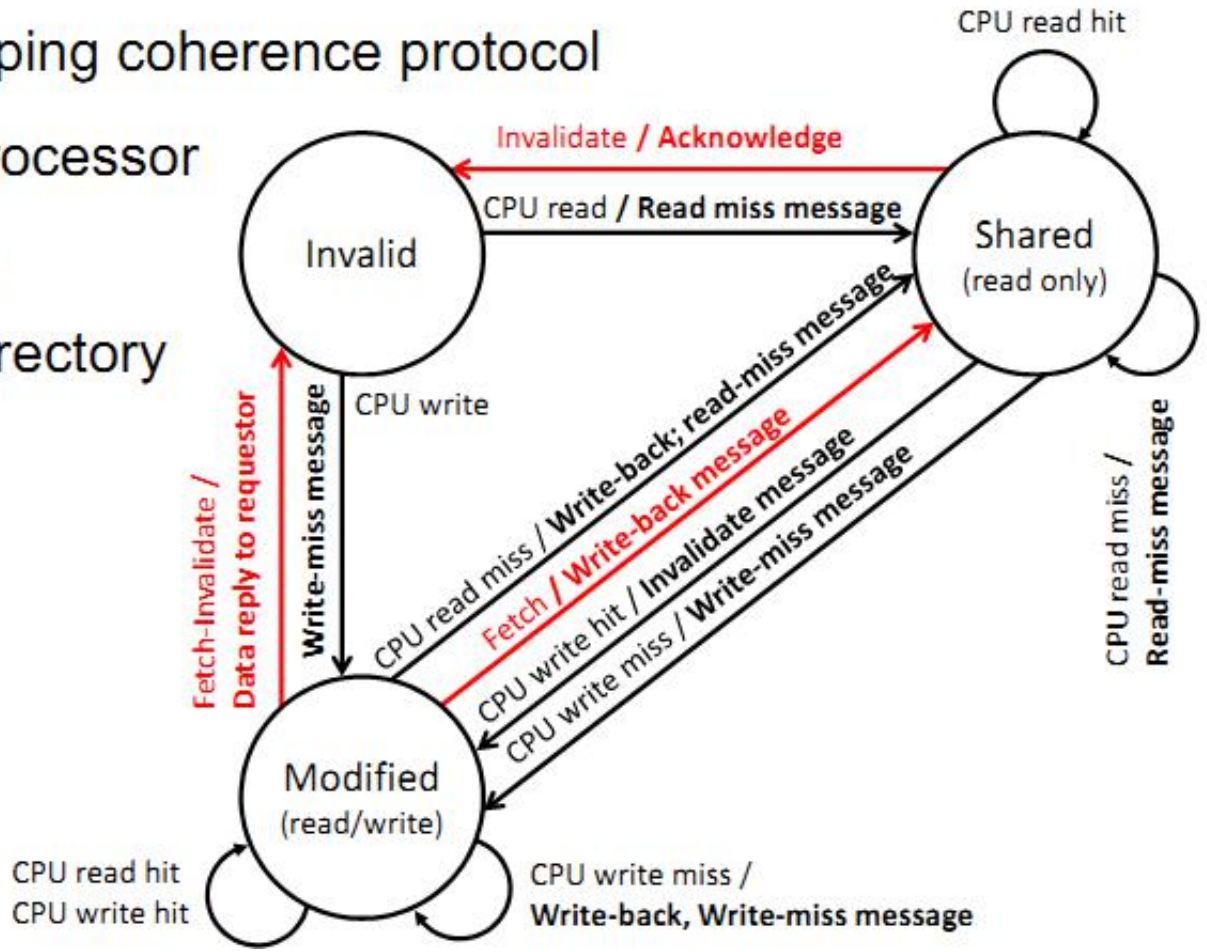


Directory Protocol Messages

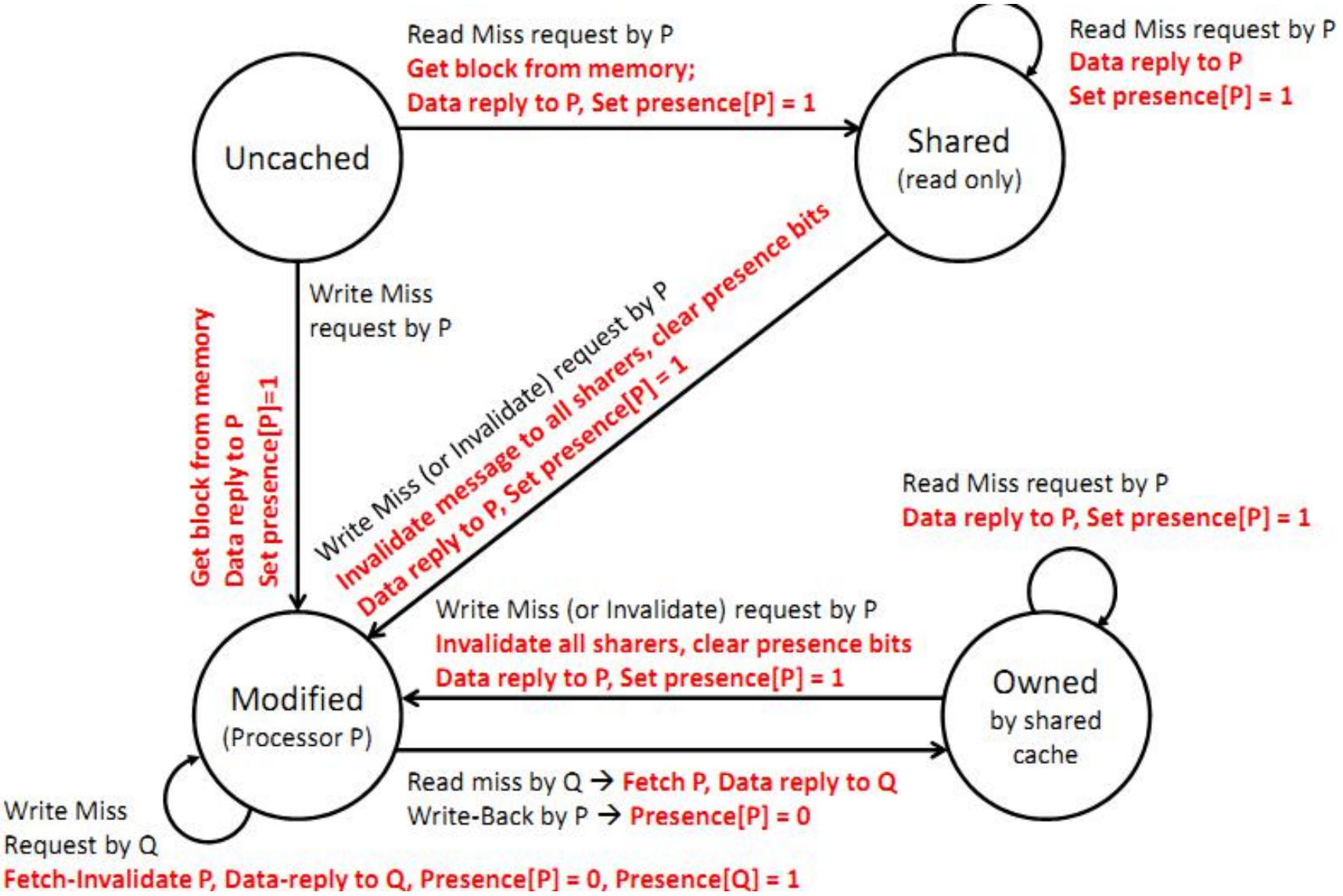
Message Type	Source	Destination	Message Function
Read Miss	Local Cache	Home Directory	Processor P has a read miss at address A Request data and make P a read sharer
Write Miss	Local Cache	Home Directory	Processor P has a write miss at address A Request data and make P the exclusive owner
Invalidate	Local Cache	Home Directory	Processor P wants to invalidate all copies of the same block at address A in all remote caches
Invalidate	Home Directory	Remote Caches	Directory sends invalidate message to all remote caches to invalidate shared block at address A
Acknowledge	Remote Cache	Home Directory	Remote cache sends an acknowledgement message back to home directory after invalidating last shared block A
Acknowledge	Home Directory	Local Cache	Directory sends acknowledgment message back to local cache of P after invalidating all shared copies of block A
Fetch	Home Directory	Remote Cache	Directory sends a fetch message to a remote cache to fetch block A and to change its state to shared
Fetch & Invalidate	Home Directory	Remote Cache	Directory sends message to a remote cache to fetch block A and to change its state to invalid
Data Block Reply	Directory or Cache	Local Cache	Directory or remote cache sends data block reply message to local cache of processor P that requested data block A
Data Block Write Back	Remote Cache	Home Directory	Remote Cache sends a write-back message to home directory containing data block A

MSI State Diagram for a Local Cache

- Three states for a cache block in a local (private) cache
- Similar to snooping coherence protocol
- Requests by processor
  - **Black arrows**
- Requests by directory
  - **Red arrows**

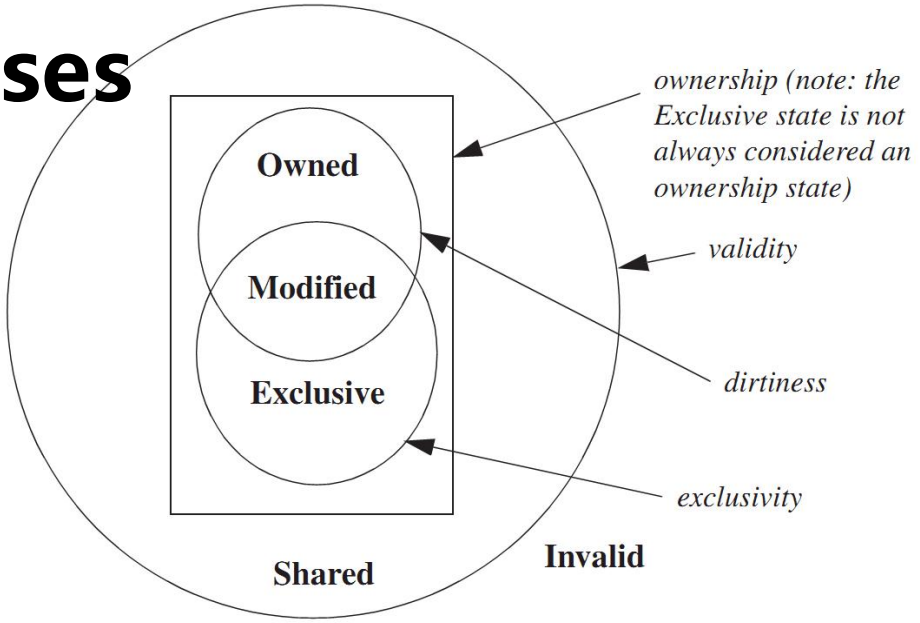


MOSI State Diagram for Directory



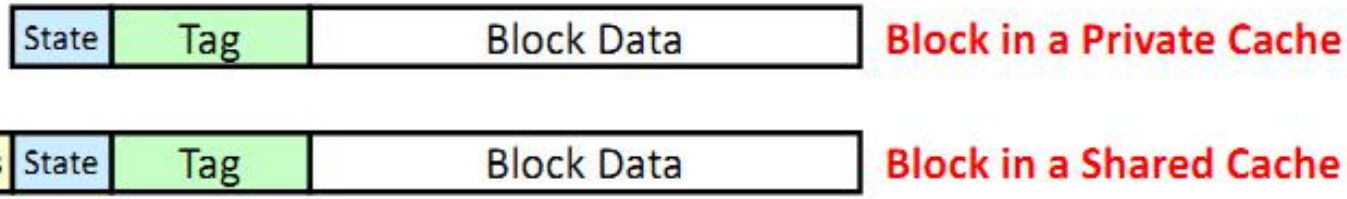
Summary

- **共享数据块的跟踪：监听和目录**
- **Cache一致性协议实现：写作废和写更新**
- **集中式共享存储 Cache一致性协议**
  - Snooping协议：MSI，MESI，MOESI
- **Coherency Misses**
  - True Sharing
  - False Sharing



Summary

- **分布式共享存储的Cache一致性协议**
  - Cache块的状态：
    - 私有Cache中块的状态 /目录中Cache块的状态



  - 状态迁移过程：状态迁移图
- **存储同一性问题**
  - Consistency研究不同处理器访问存储器操作的定序问题，即所有处理器发出的所有访问存储器操作(所有地址）的全序
  - Coherence研究不同处理器访问存储器相同地址操作的定序问题，即访问每个Cache块的局部序问题

Acknowledgements

- **These slides contain material developed and copyright by:**
  - John Kubiatowicz (UCB)
  - Krste Asanovic (UCB)
  - David Patterson (UCB)
  - Chenxi Zhang (Tongji)
- **UCB material derived from course CS152、CS252、CS61C**
- **KFUPM material derived from course COE501、COE502**