5.3.5 Cache的一致性问题

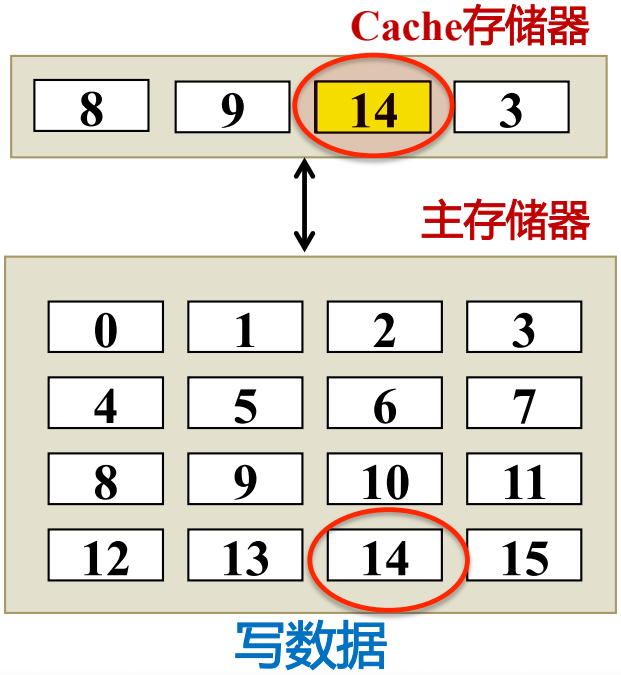
刘 芳 副教授
国防科学技术大学计算机学院

5.3.5 Cache的一致性问题

Cache的一致性问题指什么？

Cache中的内容是主存的副本

情况1：当Cache中的内容进行更新时，而没有改变主存中的相应内容时，Cache和主存之间产生了不一致(inconsistent)



5.3.5 Cache的一致性问题

Cache的一致性问题指什么？

情况2：当多个设备都允许访问主存时

例：I/O设备可通过DMA 方式直接读写主存时，如果Cache中的内容被修改，则I/O设备读出的对应主存单元的内容无效；若I/O设备修改了主存单元的内容，则对应Cache行中的内容无效。

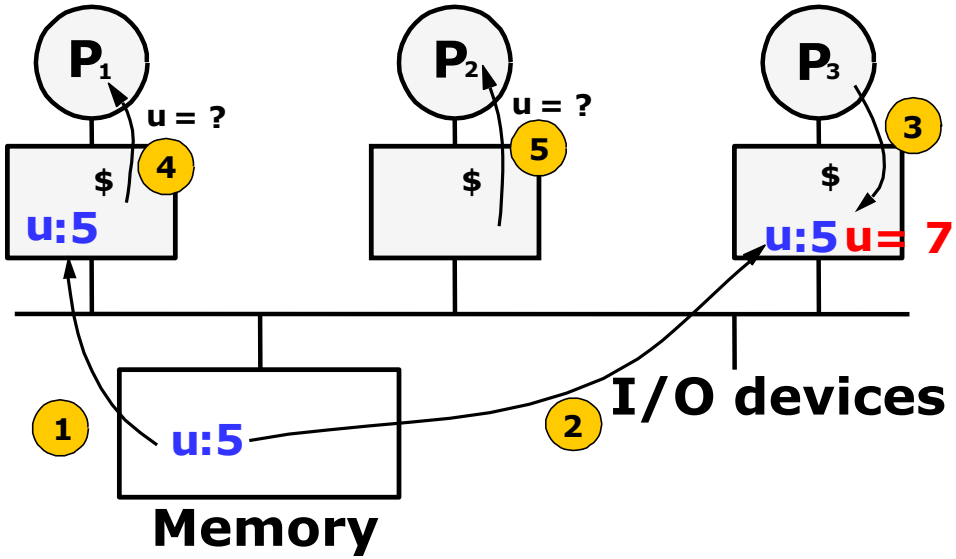
情况3：当多个CPU都有各自私有的Cache并且共享主存时

例：某个CPU修改了自身Cache中的内容，则对应的主存单元和其他CPU中对应的Cache行的内容都要变为无效。

5.3.5 Cache的一致性问题

处理器私有Cache引入的问题

- 同一变量拷贝可能出现在多个处理器私有Cache中
- 某处理器写操作可能对其它处理器是不可见的



P3私有Cache中的变量u被更新后，各处理器读到的是不同的u值

程序不能容忍这样的错误， 但这种现象却很常见!

如何保持Cache一致性呢？

当CPU写存储地址命中Cache时

需要研究Cache写机制

- Write Through (写直达、写通过、直写)
- Write Back (写回、一次性写、回写)

Cache的写机制：Write Through（写直达、写通过、直写）

当一个写操作产生时，新值同时写到Cache和主存的块中

写直达会带来什么影响?

Memory is too slow(>100Cycles)

10%的存数指令会使CPI增加到：1.0+100×10%=11

同步更新！

在Cache和主存之间使用写缓冲(Write Buffer)

- 当一个数据等待被写入主存时，先将其存入写缓冲；
- 在把数据写入Cache和写缓冲后，处理器继续执行命令；
- 当写主存操作结束后，写缓冲里的数据释放

Cache的写机制：Write Back (写回、一次性写、回写)

当一个写操作产生时，新值仅被写到Cache中，而不被写入主存
写回方式会带来什么影响?

没有同步更新！

- 大大降低主存带宽需求，提高系统性能，控制可能很复杂
- 每个Cache行都设置一个修改位（“dirty bit-脏位”），如果对应Cache行中的主存块被修改，就同时置修改位为“1“，如果修改位为“0“，则说明对应主存块没有被修改过
- 只有当修改位为“1“的块从cache中替换出去时，才把它写回主存

如何保持Cache一致性呢？

写命中(Write Hit)
要写的单元已经在Cache中

- Write Through
  (写通过、写直达、直写)
- Write Back
  (一次性写、写回、回写)

写不命中(Write Miss)
要写的单元不在Cache中

- Allocate-on-miss (写分配)
- No-allocate-on-write (写不分配)

5.3.5 Cache的一致性问题

写不命中时如何处理？

Allocate-on-miss (写分配)

将该主存块装入Cache，然后更新Cache行中的相应单元 ；

试图利用空间局部性，但增加了从主存读入一个块的开销

No-allocate-on-write (写不分配)

直接写主存，不把被写数据放入Cache；

减少了从主存读一个块的开销，但没有利用空间局部性

1、写直达Cache可用写分配或写不分配

2、写回Cache通常用写分配

5.3.5 Cache的一致性问题

问题1：以下描述的是哪种写策略？　　写直达、写分配！

问题2：如果用写不分配，则如何修改算法？

ON REFERENCE TO Mem[X]: Look for X among tags...

HIT: *X == TAG(i), for some cache line i*

READ: return DATA[I]
WRITE: change DATA[I]; Start Write to Mem[X]

MISS: *X not found in TAG of any cache line*

REPLACEMENT SELECTION:
Select some line k to hold Mem[X]

READ: Read Mem[X]
Set TAG[k] = X, DATA[k] = Mem[X]

WRITE: Start Write to Mem[X]
~~Set TAG[k] = X, DATA[k] = new Mem[X]~~

问题3：以下算法描述的是哪种写策略？

写回、写分配！

ON REFERENCE TO Mem[X]: Look for X among tags...

HIT: *X = TAG(i) , for some cache line I*

READ: return DATA(i)
WRITE: change DATA(i); ~~Start Write to Mem[X]~~

MISS: *X not found in TAG of any cache line*

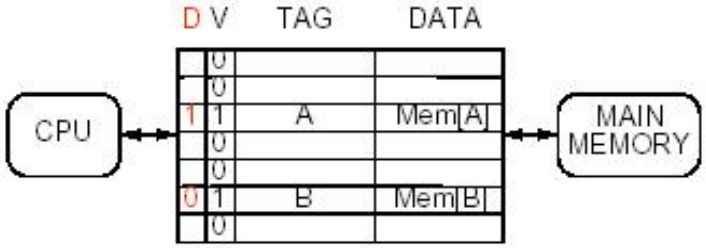
REPLACEMENT SELECTION:
~~Select some line k to hold Mem[X]~~
Write Back: Write Data(k) to Mem[Tag[k]]

READ: Read Mem[X]
Set TAG[k] = X, DATA[k] = Mem[X]

WRITE: ~~Start Write to Mem[X]~~
Set TAG[k] = X, DATA[k] = new Mem[X]

5.3.5 Cache的一致性问题

Write-back w/ "Dirty" bits



ON REFERENCE TO Mem[X]: Look for X among tags...

HIT: *X = TAG(i) , for some cache line I*
READ: return DATA(i)
WRITE: change DATA(i); ~~Start Write to Mem[X]~~ D[i]=1

MISS: *X not found in TAG of any cache line*
REPLACEMENT SELECTION:
Select some line k to hold Mem[X]
If D[k] == 1 (Write Back) Write Data(k) to Mem[Tag[k]]
READ: Read Mem[X]; Set TAG[k] = X, DATA[k] = Mem[X], D[k]=0
WRITE: ~~Start Write to Mem[X]~~ D[k]=1
Set TAG[k] = X, DATA[k] = new Mem[X]