CA(12)《线程级并行性》课后作业

1. 多核同步多线程多处理器如图 1 所示：

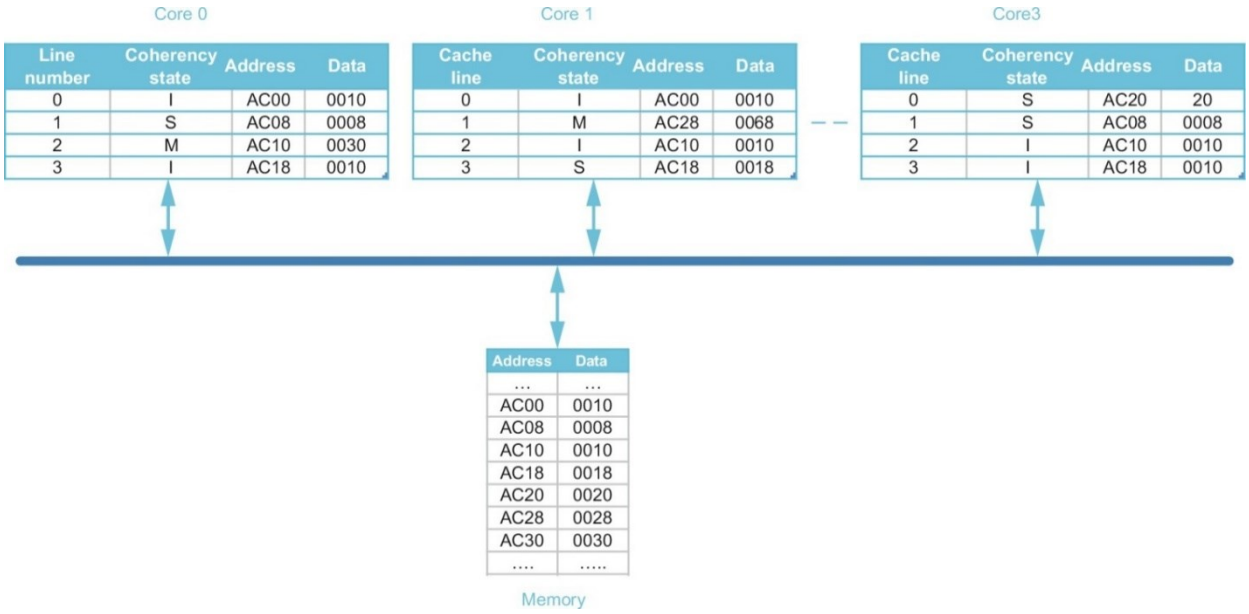

图 1 多核多处理器

图中只显示了 Cache 内容。每个核拥有一个私有 Cache，使用监听一致性协议维持一致性，监听一致性协议如下图所示：

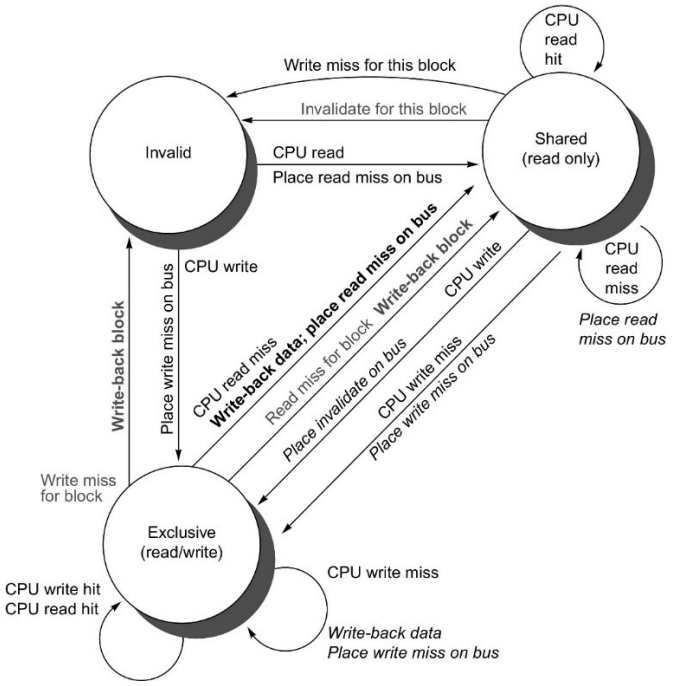

图 2 Cache 一致性状态转移图（黑色表示局部处理器引起的状态转移，灰色表示总线动作引起的状态转移）

每个 Cache 采取直接映射方式，有 4 个 Cache 行(line)，每个 Cache 行保存 2 个字节。为了进一步简化，内存中的整个行地址显示在 Cache 的地址域中，标签通常存在这个域中。三种一致性状态修改、共享和无效分别记为 M, S 和 I。

假设 Cache 和内存状态初始为图 1 显示的内容，CPU 操作或操作序列的形式如下：

读操作：`Ccore#:R,<address>`

写操作：`Ccore#:W,<address><--<value written>`

读操作和写操作每次一个字节。显示经过下面给出的动作之后，Cache 和内存的结果状态（即，一致性状态，标签，数据）。仅显示经过状态改变的 Cache 行：`C0.L0:(I,AC20,0001)`表示在核 0 的 Cache 行 0 呈现一个“无效”一致性状态`(I)`、存储着内存 `AC20` 的内容，数据为 `0001`。另外，使用 `M:<address><--value` 表示内存状态的改变。

下述 7 个问题之间没有关联，可以认为 Cache 和内存都是从初始状态开始的。

(1) (10) `C0:R,AC20`

(2) (10) `C0:W,AC20<--80`

(3) (10) `C3:W,AC20<--80`

(4) (10) `C1:R,AC10`

(5) (10) `C0:W,AC08<--48`

(6) (10) `C0:W,AC30<--78`

(7) (10) `C3:W,AC30<--78`

2. 考虑图 3 表示的分布式共享内存系统，它由 8 个处理器核节点组成点对点互联的三维立方体。

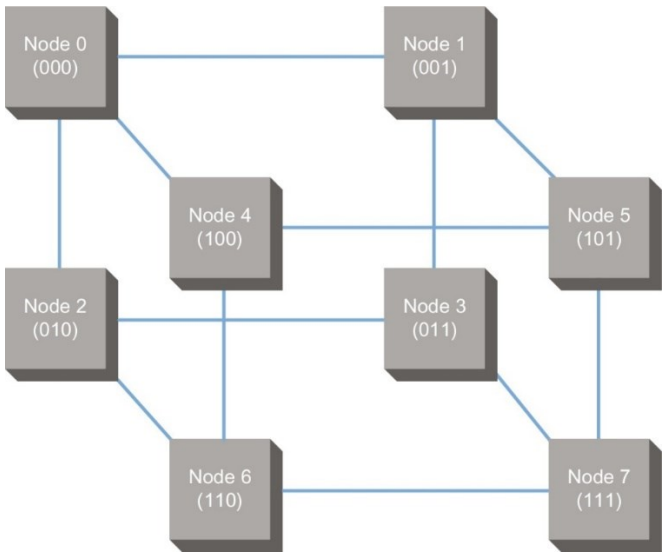

图 3. 分布式共享内存的多核多处理器系统

为了简化，假设下面的减化配置：

- 每个节点为单个处理器核，拥有直接映射 L1 数据缓存，专用缓存控制器。
- L1 数据缓存拥有 2 个缓存行，每行 B 个字节。
- L1 缓存状态分别记为 `M`（修改）、`S`（共享）和 `I`（无效），缓存项形式如下：

  `1:S,M3,0xabcd`

  表示：Cache 行 1 处于共享状态，包含内存块 `M3`，且块的数值是 `0xabcd`。
- 系统内存由 8 个内存块组成（即，每个节点一个内存块），且分布在 8 个节点中，每个节点拥有一个内存块，节点 `C1` 拥有的内存块是 `M1`。
- 每个内存块包含 B 个字节，由存储在内存块中的一致性目录项跟踪。
- 每个内存目录项的状态分别记为 `DM`（目录修改）、`DS`（目录共享）和 `DI`（目录无效）。另外，目录项使用比特矢量列出块的所有者，每个比特表示一个节点。内存块和相关的目录项示例如下：

  `M3:0xabcd,DS,00000011-->`

  表示：内存块 `M3`（在节点 `C3` 中）包含数值 `0xabcd`，由节点 `0` 和 `1` 共享。

读/写表示法

为了描述读/写事务，使用如下表示：

读：`Ci#:R,<Mi>`

写：`Ci#:W,<Mi><--<value written>`

如：

`C3:R,M2` 表示在节点 `3` 中的处理器核从内存块 `M2` 的地址发出读取事务（地址可能已经缓存在 `C3` 中）

`C0:W,M3<--0018` 表示在节点 `0` 中的处理器核发出写入事务（数据 `0x0018`）到内存块 `M3` 的地址中（地址可能已经缓存在 `C0` 中）

消息(message)

目录一致性机制依赖于图 4 目录协议描述的命令消息和数据消息的交换，命令消息的例子是读请求，数据消息的例子是读响应（包括数据）。

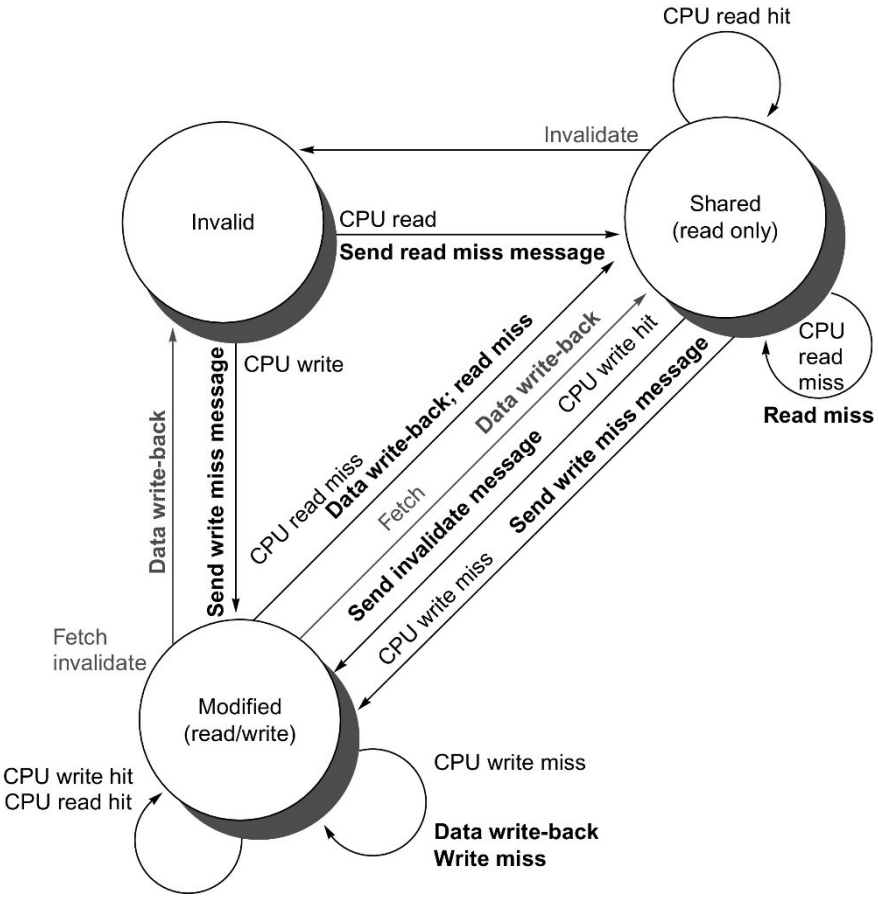

图 4. 在基于目录系统中单个 cache 块的状态转移图（局部处理器请求显示为黑色，主目录的请求显示为灰色）

- 在同一节点中发出的结束消息不跨越任何节点间的链接。
- 具有不同源/目节点的消息通过节点间的链路传输，这些消息从一个缓存控制器发送到另一个缓存控制器，从一个缓存控制器发送到目录控制器，或从一个目录控制器发送到一个缓存控制器。
- 从源节点到不同目标节点的消息是静态路由的。
  - ✧ 静态路由算法选择源节点和目标节点之间的最短路径
  - ✧ 通过考虑源索引和目标索引的二进制表示（如：节点 `C1` 的索引是 `001`，节点 `C4` 的索引是 `100`）来确定最短路径，然后，从一个节点移动到消息尚未跨越的相邻节点。
    - ➢ 例如，从节点 `6` 到节点 `0`（`110-->000`），路径为 `110-->100-->000`。
    - ➢ 因为存在多条最短路径（如上例中的路径 `110-->010-->000` 是另一条最短路径），假设通过首先反转与目标索引中相应比特不同的最低有效位来选择路径。例如，从节点 `1` 到节点 `6` 的移动（`001-->110`），路径是 `001-->000-->010-->110`。
  - ✧ 任何消息移动的最长路径是 3 个链（等于节点索引二进制表示中的比特数目）。
- 一个节点可以同时处理最多三条来自/发送到不同相邻节点链路的消息，前提是它们当中没有两个节点竞争相同链路资源。下面例子说明了**消息发送/接收到/来自/通过**节点 `0`。

  消息：从 `001-->010;010-->000`（到缓存/目录控制器）；`100-->001`，**OK**（不同目标）

消息：从 001-->010;000-->001（从缓存/目录控制器）;100-->001，**Not OK**（两条消息的目的地是节点 001）。

在目标争用的情形下，将优先级分配给：

a) 消息目的节点（本例中节点 001）缓存或目录控制器；然后

b) 消息从一个节点转发到另一个节点（本例中通过节点 000）；然后

c) 来自节点（本例中节点 000）缓存或目录控制器的消息。

- 假设传输和服务延迟如下表所示

消息类型	缓存控制器	目录控制器	链
无数据	2 个周期	5 个周期	10 个周期
带数据	$(3+[B/4])$个周期	$(6+10\times B)$个周期	$(4+B)$个周期

- 如果消息通过节点转发，则节点首先接收消息，然后再沿路径发送到另一个节点。
- 假设每个缓存控制器和目录控制器都具有无限容量来将消息排队，并按照 FCFS (First Cache First Send) 顺序为它们提供服务。

假设初始化所有缓存行无效，在内存 Mi 中数据是字节 i（$0\text{x}00 \le i \le 0\text{x}07$）重复与块容量一样的次数。假设连续的请求是完全序列化的，也就是说，直到完成（相同或不同处理器核）早期发出的请求之前，没有处理器核发出一致性请求。

对下面每个问题，

(1) 显示在给定事务序列完成之后缓存和目录控制器（包括数据值）的最终状态（即，一致性状态，共享者/所有者，标签，数据），和

(2) 显示转移的消息（选择消息类型的合适格式）

a) (10)

```
C3: R, M4
C3: R, M2
C7: W, M4 <-- 0xaaaa
C1: W, M4 <-- 0xbbbb
```

b) (10)

```
C3: R, M0
C3: R, M2
C6: W, M4 <-- 0xaaaa
C3: W, M4 <-- 0xbbbb
```

c) (10)

```
C0: R, M7
C3: R, M4
C6: W, M2 <-- 0xaaaa
C2: W, M2 <-- 0xbbbb
```