# 1 问题描述

有 NUMA 型结构如图 1-1 所示。该结构共有 32 个节点，以十进制给与各节点编号。

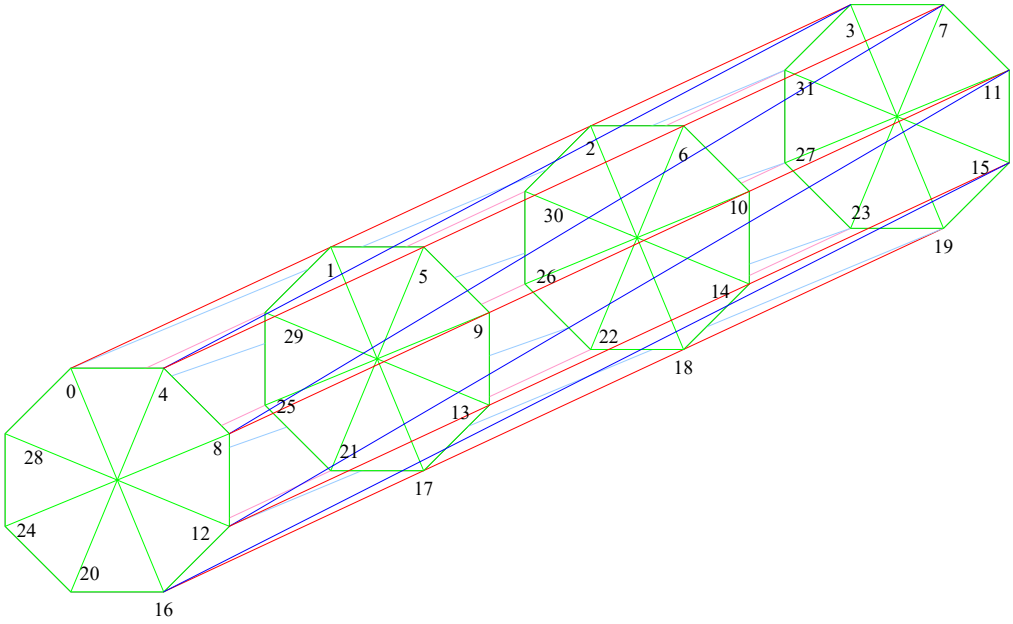


图 1-1 NUMA 型结构

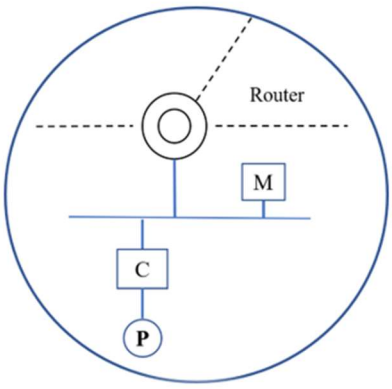


图 1-2 路由结构

给定号码 n 的连接方式：

$$n \rightarrow \mathrm{mod}((n+4^{i}),32), i\text{为}0,1,2$$

（1）请给出用于该结构的数据一致性管理方案，并评价其效率的优劣。

（2）请给出该拓扑结构的路由算法，并证明其正确。

# 2 一致性管理方案

## 2.1 问题由来

Cache作为提高系统性能的一种常用手段在并行计算机系统中得到了很多应用。但是，在并行处理机系统中的私有Cache会引起Cache中的内容相互之间以及共享存储之间互不相同的问题。出现数据不一致问题的原因通常有三个[1]：不同处理器对各自缓存的同一高速缓存行的不同拷贝的异步写操作；多处理器系统中的进程迁移(process migration)；绕过高速缓存拷贝拥有者的I/O操作。

因此，欲使某一存储器系统被认为是在高速缓存行级一致的，则对高速缓存行 X 在高速缓存中的所有拷贝的存取操作皆需满足如下要求[2]：

①若处理器P在完成对X的写操作后再对X进行读操作，而此间没有其他的处理器对X进行写操作，则返回由P写入的值；

②若在处理器Q完成对X的写操作后由处理器P进行X的读操作，并且写操作和读操作是完全分离的，在此期间也没有其它处理器对X进行写操作，则返回Q写入的值；

③多个不同的处理器对同一高速缓存行的写操作总是按所有处理器所见到的存储器事件的顺序而串行化。

## 2.2 常见的一致性管理协议

根据系统结构的不同，解决Cache一致性问题的协议机制主要有两类。在采用基于总线互联结构的系统中，由于系统中每个处理机都能察觉到存储器系统正在进行的活动，在某个活动破坏了Cache一致性时，Cache控制器将采取相应的动作使有关的拷贝无效或更新。这种保持Cache一致性的协议称为监听协议。在使用监听协议时，可采用：写无效(Write-Invalidate)和写更新(Write-Update)这两种策略。

但是，很多并行系统并不采用基于总线互连的结构，在这些系统中处理机无法对存储器系统的活动进行监听，这类系统的Cache一致性问题一般采用基于目录的方法来解决。

大多数 CC-NUMA 多处理机系统采用基于目录的高速缓存一致性协议[3]。比较经典的有HP/Convex Exemplar X-class, SGI/Cray Origin 2000[4],Sequent NUMA-Q 2000等。

目前已有的商业CC-NUMA都主要采用全映射(full-map),有限映

射(limited)和链式(chained)目录一致性协议。

## 2.3 本题的一致性管理方案

本题设计了一种后写无效化的一致性协议保证数据一致性，在 Cache 中设置三个标志位：Valid/Invalid，Exclusive/Shared，Owner/Non-owner。

| Valid/Invalid | Exclusive/Shared | Owner/Non-owner | DATA |
|---|---|---|---|

图 2-1 Cache 每页结构

Valid/Invalid 表示 Cache 中该页是否有效，如果该位为 Invalid 表示无效，Valid 表示有效。处理器写 Cache 中一页时，应考虑是否需要将其他 Cache 中该页状态改为 Invalid。处理器读标志位为 Invalid 的页时，需要考虑从何处获得正确副本的问题。

Exclusive/Shared 表示 Cache 中该页是否独占，Exclusive 表示独占，Shared 表示共享。

Owner/Non-owner 表示目前的处理器对某页的所有权，Owner 代表享有所有权，Non-owner 则是没有所有权。在系统中如果有一个处理器是某页的 Owner，则其他处理器在读该页时从 Owner 获取。

在内存 M 中，采用 full-map 法，将每一页按图 2-2 表示。

| Exclusive/Shared | PU0 | PU1 | … | PU31 | Owner | DATA |
|---|---|---|---|---|---|---|

图 2-2 内存每页结构

Exclusive/Shared，该状态位为 Exclusive 时，内存与 Cache 中的内容不一致；该状态位为 Shared 时内存与 Cache 中的内容一致。

PUi(i=0,1…31)表示第 i 个处理单元的 Cache 中是否存在本页。

Owner 占 5bit，使用二进制表示哪一个处理单元对本页享有所有权。

Cache 共存在五种状态：S-N(Shared-Non-owner)，S-O(Shared-Owner)，E-O(Exclusive-Owner)，I(Invalid)，V(Valid)。

（1）假设处理器 A 对 Cache 进行写操作，对于任何一种状态，均采用同一种操作。首先向内存 M 发出写要求，内存根据 Owner 位，向 Owner 发送 Ownership 请求。Owner 收到请求后，向所有使用该页的 Cache 发布无效化信息。待接收完所有返回的无效化完成信息后，移交 Ownership，将 Owner 修改为处理器 A 的标号。处理器 A 对应

的内存修改为 Exclusive，Owner 修改为本处理器的标号。此时处理器 A 可以对 Cache 任意修改，修改后将 Cache 置于 E-O。

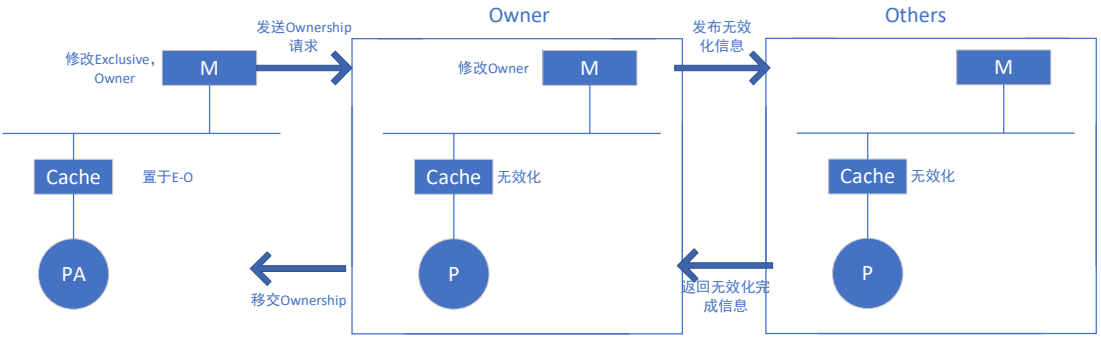


图 2-3 处理器写过程

（2）假设处理器 A 对 Cache 进行读操作，Cache 不是 I 状态，则直接读取。

（3）假设处理器 A 对 Cache 进行读操作，Cache 是 I 状态。访问处理器 A 的内存 M，M 根据 Owner 位不断查找，最终查找到指定的内存。

若内存中该页的状态位为 Exclusive，则将 Cache 中的内容写回内存，将 Cache 状态置于 S-O，内存状态置于 Shared。将页依次写回 A 的内存和 Cache，并将 A 的 Cache 置于 S-N 状态。

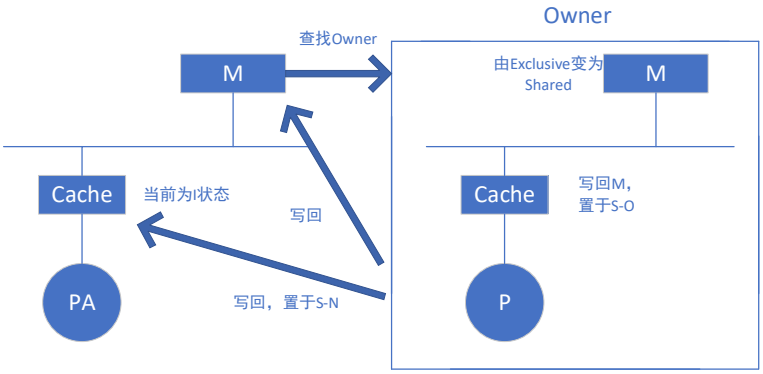


图 2-4 处理器读过程（状态位为 Exclusive）

若内存中该页的状态位 Shared，则直接将页依次写回 A 的内存和 Cache，并将 A 的 Cache 置于 S-N 状态。

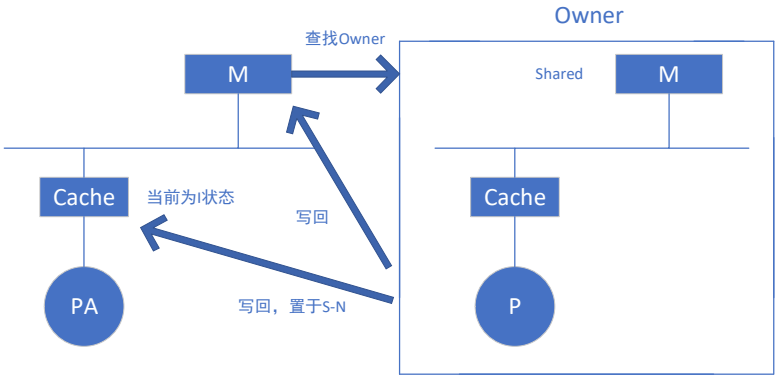


图 2-5 处理器读过程（状态位为 Shared）

## 2.4 效率优劣评价

使用后写无效化协议，在某一页修改时，标志该页无效，十分简单易行。但是当处理器交替写时，容易产生 Ping Pong 效应。在并行机中，这种需求经常出现，使得 Bus 上十分繁忙。

因此，本一致性管理方案，更适用于进程连续处理倾向较强的工作场景。

全映射目录的基本思想是该目录包含全局范围内共享的所有高速缓存行的信息。即存储模块的每一个高速缓存行对应一个目录项，每个目录项包含 P 个指针，P 是指结点的个数，这些指针通过位向量标识。位向量的每一位与一个结点相对应，用于指出该结点远程 Cache 中有无该高速缓存行的拷贝。

假定全映射目录系统有 P 个结点，每个结点的存储器拥有 M（bit），每个高速缓存行含 B 位，也即每个结点的存储器包含 C=M/B 个高速缓存行。其全映射目录结构如上图所示。则整个目录表占用的存储空间是（忽略状态标志位）：

$$P \times M / B \times P = O(P^2)$$

可见它与结点个数呈平方倍增长，这显然会对系统的扩展性造成严重的负面影响。因此，全映射目录对于含结点数量太多的多机系统不适用。

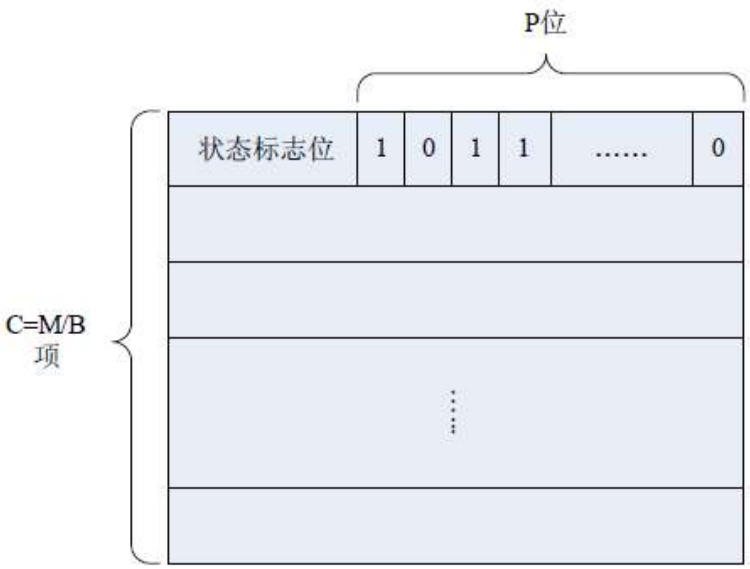


图 2-6 全映射目录

但是如果处理器个数适中，则全映射是一个十分有效的协议,能够高效寻找到需要更新的 PU 位置并进行更新。本题处理器个数适中，选用全映射目录的代价可以接受。

许多著名的多机系统，也采用了本题一致性方案类似的方法。例如，斯坦福的 DASH 多机系统[5]由 64 个处理器组成，4 个为一个结点，共 16 个结点，结点内的 Cache 一致性采用侦听策略，组间 Cache 一致性采用全映射目录机制。整个目录表占有存储器资源的 13.3%。

# 3 路由算法

## 3.1 条件分析

使用二进制表示每一个节点，见表 3-1。

| | | | |
|---|---|---|---|
| 00000 | 00001 | 00010 | 00011 |
| 00100 | 00101 | 00110 | 00111 |
| 01000 | 01001 | 01010 | 01011 |
| 01100 | 01101 | 01110 | 01111 |
| 10000 | 10001 | 10010 | 10011 |
| 10100 | 10101 | 10110 | 10111 |
| 11000 | 11001 | 11010 | 11011 |
| 11100 | 11101 | 11110 | 11111 |

表 3-1 节点二进制表示

号码 n 的连接方式为：

$$n \to \mathrm{mod}((n+4^i),32), i为0,1,2$$

二进制数的后两位表示该节点位于哪一个八边形。

二进制数的前三位表示该节点位于八边形的哪一个位置。

若两个节点的二进制数第一位不相等，其余各位相等，则表示两个节点在同一个八边形的对称位置，根据给定连接方式，此时一步可达。

在同一个八边形内，任意两节点，最多两步可达。

## 3.2 路由算法设计

假设源地址为$S_4S_3S_2S_1S_0$，目的地址为$D_4D_3D_2D_1D_0$。

先比较$S_1S_0$与$D_1D_0$，若相等则两地址在同一个八边形上，不相等则移动至同一个八边形。

再比较$S_3S_2$与$D_3D_2$，若相等则两地址在同一个八边形的同一个位置或者是同一个八边形的对称位置；若不相等，通过$S_3S_2 \pm 01$，向逆时针或顺时针方向移动，使得这两位数相等。

最后比较$S_4$与$D_4$，若相等则已达目的地址；若不相等则取反，移动到八边形对称位置。

## 3.3 路由算法正确性证明

设源地址为$S_4S_3S_2S_1S_0$，目的地址为$D_4D_3D_2D_1D_0$。

①比较$S_1S_0$与$D_1D_0$，若$S_1S_0$=$D_1D_0$，输出$S_4S_3S_2D_1D_0$。若$S_1S_0$>$D_1D_0$，$S_1S_0$-01直至$S_1S_0$=$D_1D_0$，输出$S_4S_3S_2D_1D_0$。同理，若$S_1S_0$<$D_1D_0$，$S_1S_0$+01，直至$S_1S_0$=$D_1D_0$，输出$S_4S_3S_2D_1D_0$。

②比较$S_3S_2$与$D_3D_2$，若$S_3S_2$=$D_3D_2$，输出$S_4D_3D_2D_1D_0$。若$S_3S_2 \neq D_3D_2$，$S_1S_0 \pm n(01), n \leq 2$（溢出只保留低两位，去掉高位）,直至$S_3S_2$=$D_3D_2$，输出$S_4D_3D_2D_1D_0$。

③比较$S_4$与$D_4$，若$S_4$=$D_4$，输出$D_4D_3D_2D_1D_0$。若$S_4 \neq D_4$，$S_4$取反，输出$D_4D_3D_2D_1D_0$。

证毕。

示例：源地址 00100（4），目的地址 11010（26）。

$S_4S_3S_2S_1S_0$=00100，$D_4D_3D_2D_1D_0$=11010。

$S_1S_0$+01+01=10=$D_1D_0$

$S_3S_2$+01=10=$D_3D_2$

$S_4 \xrightarrow{\text{取反}} 1=D_4$

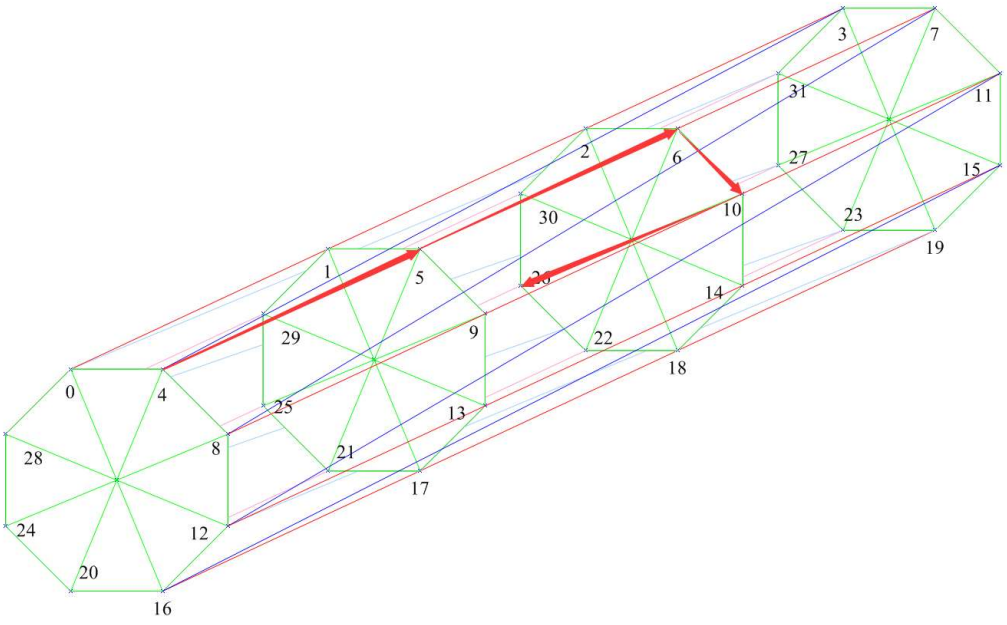


图 3-1 路由示例

# 参考文献

[1] 郑纬民, 汤志忠. 计算机系统结构(第2版)[M].清华大学出版社,1998.
[2] Will Stallings, Computer Organization and Architecture Designing for Performance, Fifth Edition[J], Pearson Education, 2001, pp623-625
[3] Stenstrom P,A Survey of Cache Coherence Schemes for Multiprocessors[J], IEEE Computer,Vo1.23, N0.6, June 1990, pp12-24
[4] J. Laudon, D. Lenoski. The SGI Origin: A cc-NUMA Highly Scalable Server[J]. Proceedings of 24th Annual International Symposium on Computer Architecture, pp.241-251, 1997
[5] D.E.Leiserson et al., The DASH Prototype: Logic and Performance[J]. IEEE Trans. On Parallel and Distributed Systems, Jan. 1993, pp 41-61.
[6] Lenoski D E , Laudon J P , Gharachorloo K , et al. The directory-based cache coherence protocol for the DASH multiprocessor[J]. ACM SIGARCH Computer Architecture News, 1990.
[7] 张毅. CC-NUMA多机系统Cache一致性研究[D].重庆大学,2008.
[8] 庞征斌. 基于SMP的CC-NUMA类大规模系统中Cache一致性协议研究与实现[D].国防科学技术大学,2007.
[9] DavidE.Culler, Culler, Singh,等. 并行计算机体系结构[M]. 机械工业出版社, 2003.
[10] Cypher R , Singhal A . Shared memory multiprocessing system employing mixed broadcast snooping and directory based coherency protocols[J]. 2003.