5.3.6 Cache替换算法

刘 芳 副教授
国防科学技术大学计算机学院

5.3.6 Cache替换算法

什么时候需要进行Cache替换？

- Cache行数<<主存块数；
- 主存块和Cache行：多对1；
- 当一个新的主存块需要复制到Cache中时，如果Cache中的对应行已经全部被占满，怎么办？

选择淘汰掉一个Cache行中的块

5.3.6 Cache替换算法

什么时候需要进行Cache替换？

例：某Cache采用二路组相联映射，其数据区容量为16块。假定第0组的两个行分别存放了主存第0块和第8块，此时需调入主存第16块，根据映射关系，它只能放到Cache的第0组。第0组中必须调出一块，如何选择调出哪一块？

淘汰策略问题/替换算法

5.3.6 Cache替换算法

什么时候需要进行Cache替换？

直接映射（Direct Mapped）Cache

- 映射唯一，无条件用新信息替换老信息

N路组相联（N-way Set Associative）Cache

- 每个主存数据有N个Cache行可选择，需考虑替换哪一行

全相联（Fully Associative）Cache

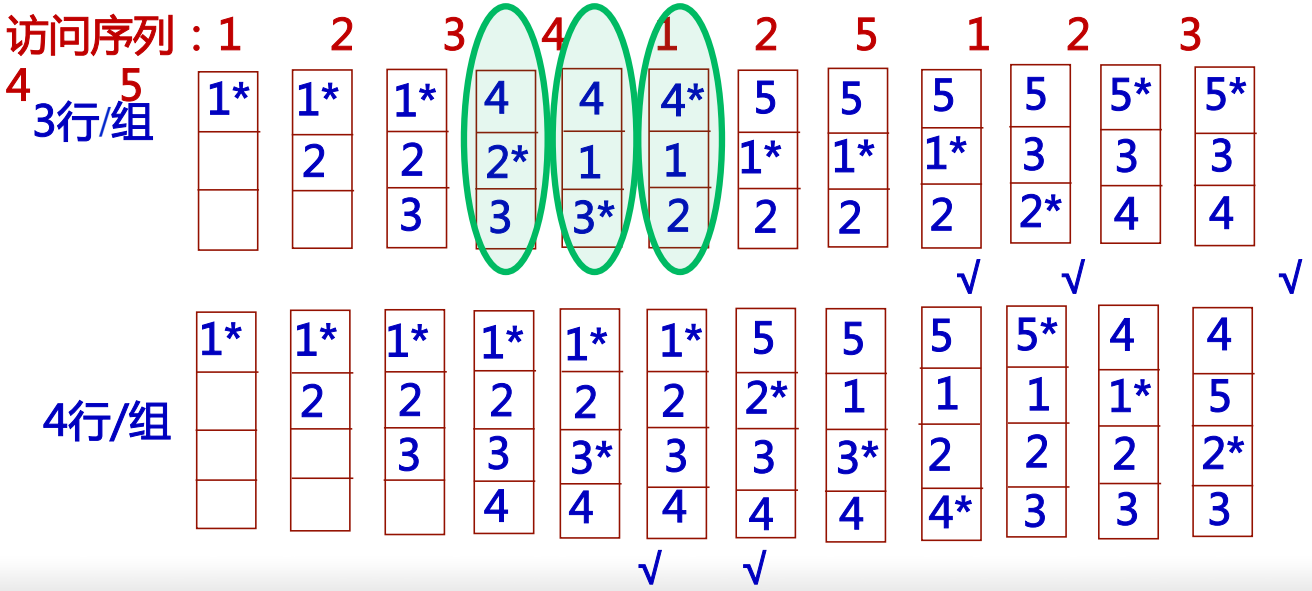
- 每个主存数据可存放到Cache任意行中，需考虑替换哪一行

常用替换算法：
先进先出FIFO、最近最少用LRU、最不经常使用LFU、随机替换

先进先出（First In First Out，FIFO）算法

基本思想：总是把最先进入的那一块替换掉

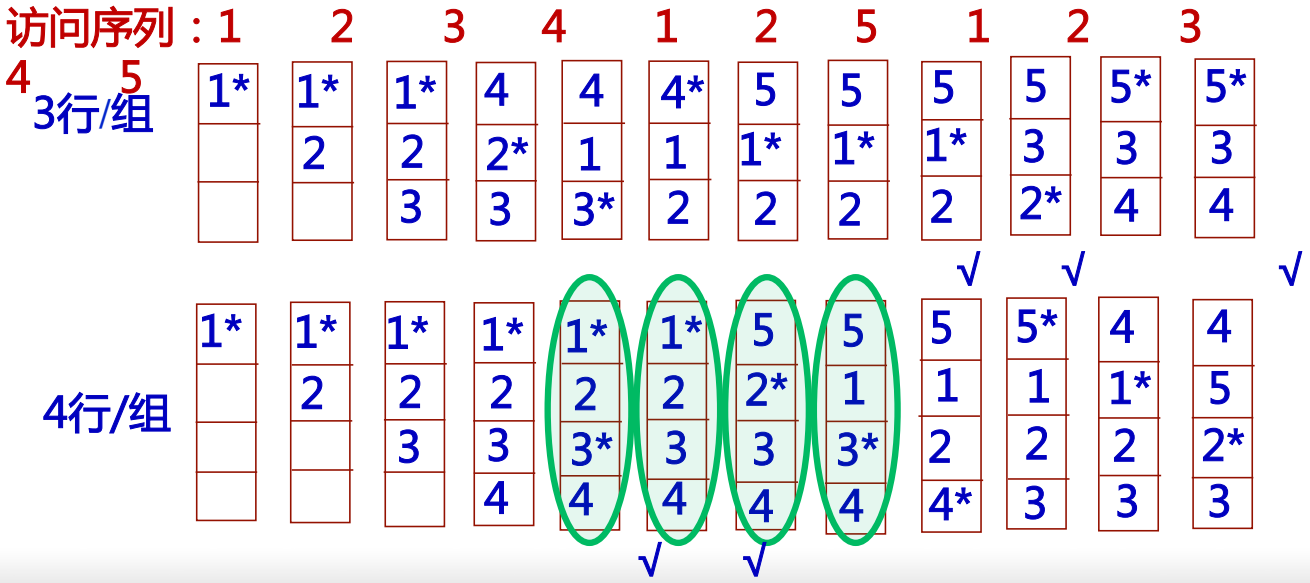
例：假定主存中的5块{1,2,3,4,5}同时映射到Cache同一组中，对于同一访存序列，考察3行/组、 4行/组的情况。



先进先出FIFO算法

基本思想：总是把最先进入的那一块替换掉

例：假定主存中的5块{1,2,3,4,5}同时映射到Cache同一组中，对于同一访存序列，考察3行/组、 4行/组的情况。

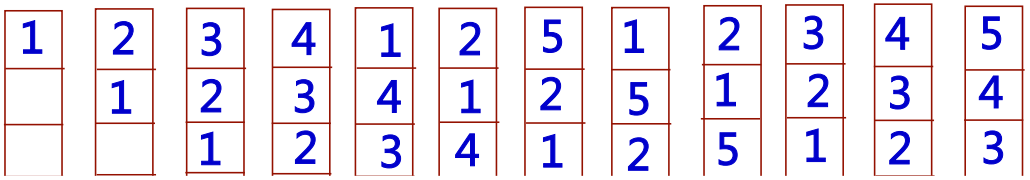


最近最少使用（Least Recently Used，LRU）算法

基本思想：总是把最近最少用的那一块淘汰掉，利用时间局部性

例：假定主存中的5块{1,2,3,4,5}同时映射到Cache同一组中，对于同一地址流，考察3行/组、 4行/组、 5行/组的情况。

访问序列： 1 2 3 4 1 2 5 1 2 3 4 5



LRU算法的命中率随组中行数的增大而提高

3行/组
4行/组
5行/组

最近最少使用LRU算法

当分块局部化范围（即：某段时间集中访问的存储区）超过了Cache存储容量时，命中率会变得很低。极端情况下，假设地址流是1,2,3,4,1,2,3,4,1,……，而Cache每组只有3行，那么，不管是FIFO，还是LRU算法，其命中率都为0。这种现象称为抖动(Thrashing / PingPong)

LRU算法具体实现：通过给每个Cache行设定一个计数器，根据计数值来记录这些主存块的使用情况。这个计数值称为LRU位

最近最少使用LRU算法

计数器变化规则

- 每组4行时，计数器设2位。计数值越小，则说明越被常用
- 命中时，被访问行的计数器置0，其他行计数器加1，其余不变
- 未命中且该组未满时，新行计数器置为0，其余全加1
- 未命中且该组已满时，计数值最大的那一行中的主存块被淘汰，新行计数器置为0，其余加1

访问序列：1 2 3 4 1 2 5 1 2 3 4 5

0	1	1	1	2	1	3	1	0	1	1	1	2	1	0	1	1	1	2	1	3	1	0	5
		0	2	1	2	2	2	3	2	0	2	1	2	2	2	0	2	1	2	2	2	3	4
				0	3	1	3	2	3	3	3	0	5	1	5	2	5	3	5	0	4	2	3
						0	4	1	4	2	4	3	4	3	4	3	4	0	3	1	3	1	2

注：表中蓝色表示计数器的值，红色表示Cache中存放的数据

随机（Random）替换算法

基本思想：随机地从候选的槽中选取一个淘汰，与使用情况无关

模拟试验表明，随机替换算法在性能上只稍逊于LRU算法，而且代价低！

5.3.6 Cache替换算法

例：假定计算机系统有一个容量为32K×16位的主存，且有一个4K字的4路组相联Cache，主存和Cache之间的数据交换块的大小为64字。

物理地址是按照字编址还是字节编址
完全需要看内存的内部组成

1、试分析Cache的结构和主存地址的划分

答：假定主存按字编址，每字为16位。

Cache：4K字=64*64字/行＝16组 × 4行 / 组 × 64 字 / 行

主存：32K字=512块 × 64字 / 块 = = 2^5 ×2^{4k}块× 64字 / 块

主存地址划分为：

标志位	组号	字号
5	4	6

5.3.6 Cache替换算法

例：假定计算机系统有一个容量为32K×16位的主存，且有一个4K字的4路组相联Cache，主存和Cache之间的数据交换块的大小为64字。

设Cache开始为空，处理器顺序地从存储单元0、1、…、4351中取数，一共重复10次。Cache比主存快10倍，设采用LRU策略。

2、分析：采用Cache后速度提高了多少？

这里指的内存存储单元大小为一个字，而不是指块的大小

答：处理器顺序地从存储单元0、1、…、4351中取数

4352/64=68，

处理器的访问过程是对前68块连续访问10次

5.3.6 Cache替换算法

	第0 行	第1 行	第2 行	第3 行
第0组	0/64	16/	32	48
第1组	1/65	17	33	49
第2组	2/66	18	34	50
第3组	3/67	19	35	51
第4组	4	20	36	52
……	……	……	……	……
……	……	……	……	……
第15组	15	31	47	63

处理器的访问过程是对前68块连续访问10次

LRU算法分析：

第一次循环：对于每一块只有第一字未命中，其余都命中，缺失68次。

5.3.6 Cache替换算法

	第0 行	第1 行	第2 行	第3 行
第0组	0/64	16/0	32	48
第1组	1/65	17	33	49
第2组	2/66	18	34	50
第3组	3/67	19	35	51
第4组	4	20	36	52
……	……	……	……	……
……	……	……	……	……
第15组	15	31	47	63

处理器的访问过程是对前68块连续访问10次

LRU算法分析：

第一次循环：对于每一块只有第一字未命中，其余都命中，缺失68次。

5.3.6 Cache替换算法

	第0 行	第1 行	第2 行	第3 行
第0组	0/64	16/0	32	48
第1组	1/65	17/1	33	49
第2组	2/66	18/2	34	50
第3组	3/67	19/3	35	51
第4组	4	20	36	52
……	……	……	……	……
……	……	……	……	……
第15组	15	31	47	63

LRU算法分析：

第一次循环：对于每一块只有第一字未命中，其余都命中，缺失68次。

5.3.6 Cache替换算法

	第0 行	第1 行	第2 行	第3 行
第0组	0/64/48	16/0/64	32/16	48/32
第1组	1/65/49	17/1/65	33/17	49/33
第2组	2/66/50	18/2/66	34/18	50/34
第3组	3/67/51	19/3/67	35/19	51/35
第4组	4	20	36	52
……	……	……	……	……
……	……	……	……	……
第15组	15	31	47	63

LRU算法分析：

第一次循环：对于每一块只有第一字未命中，其余都命中，缺失68次。

后9次循环：有20块的第一字未命中，其余都命中。

5.3.6 Cache替换算法

	第0 行	第1 行	第2 行	第3 行
第0组	0/64/48	16/0/64	32/16	48/32
第1组	1/65/49	17/1/65	33/17	49/33
第2组	2/66/50	18/2/66	34/18	50/34
第3组	3/67/51	19/3/67	35/19	51/35
第4组	4	20	36	52
……	……	……	……	……
……	……	……	……	……
第15组	15	31	47	63

LRU算法分析：

第一次循环：对于每一块只有第一字未命中，其余都命中，缺失68次。

后9次循环：有20块的第一字未命中，其余都命中。

命中率p： (43520−68−9×20)/43520=99.43%

速度提高：$t_m/t_a=t_m/(pt_c+(1-p)t_m)=10/(p+10\times(1-p))=9.5$倍

这里我还是有意见的平均访存时间应该是命中时间+缺失率*缺失代价