第五章 存储系统

第五讲 降低Cache不命中率

谢长生
武汉光电国家研究中心
华中科技大学计算机科学与技术学院

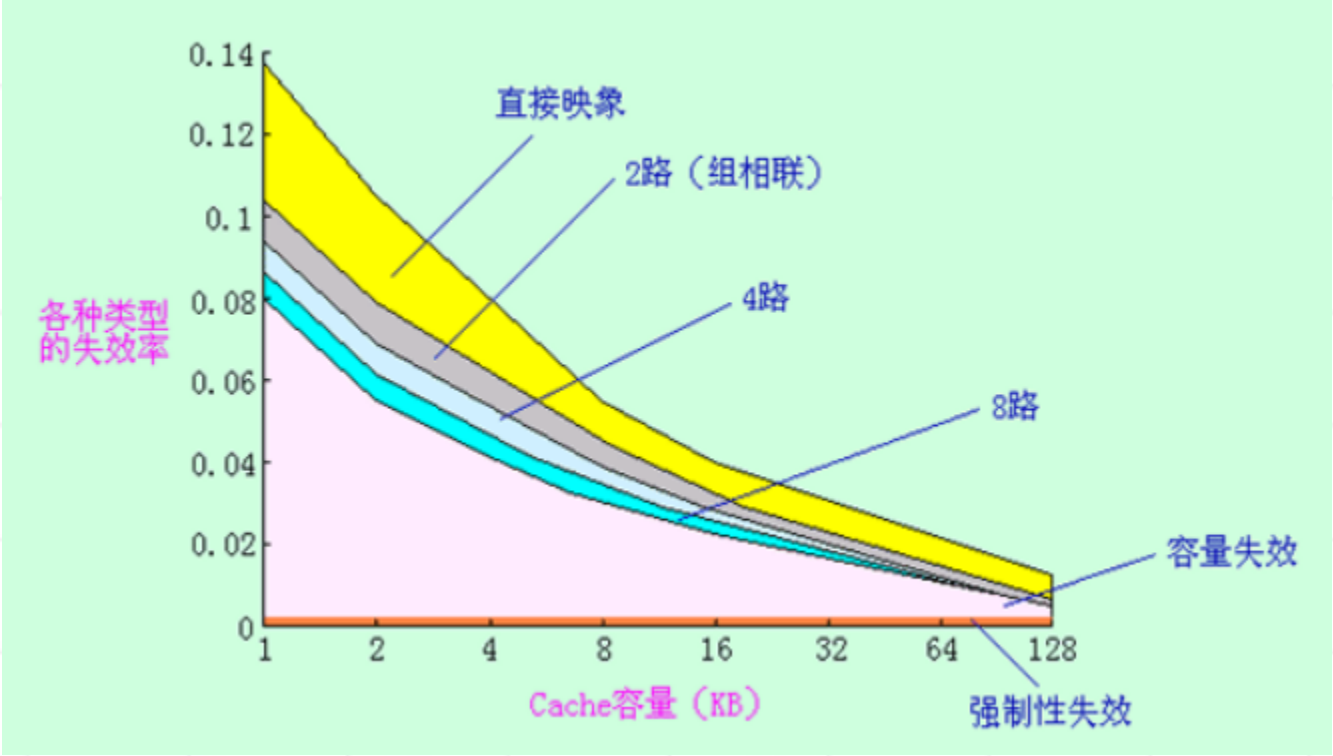
华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

5.5.1 三种类型的不命中（3C）

- 强制性不命中(Compulsory miss)
  - 当第一次访问一个块时，该块不在Cache中，需从下一级存储器中调入Cache。(冷启动不命中，首次访问不命中)
- 容量不命中(Capacity miss)
  - 如果程序执行时所需的块不能全部调入Cache中，则当某些块被替换后，若又重新被访问，就会发生不命中。
- 冲突不命中(Conflict miss)
  - 在组相联或直接映象Cache中，若太多的块映象到同一组(块)中，则该组中某个块被别的块替换(即使别的组或块有空闲位置)，然后又被重新访问的情况。(碰撞不命中，干扰不命中)

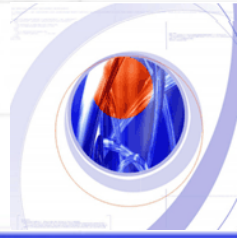
5.5 降低Cache不命中率

- 相联度越高，冲突不命中就越少；
- 强制性不命中和容量不命中不受相联度的影响；
- 强制性不命中不受Cache容量的影响，但容量不命中却随着容量的增加而减少。

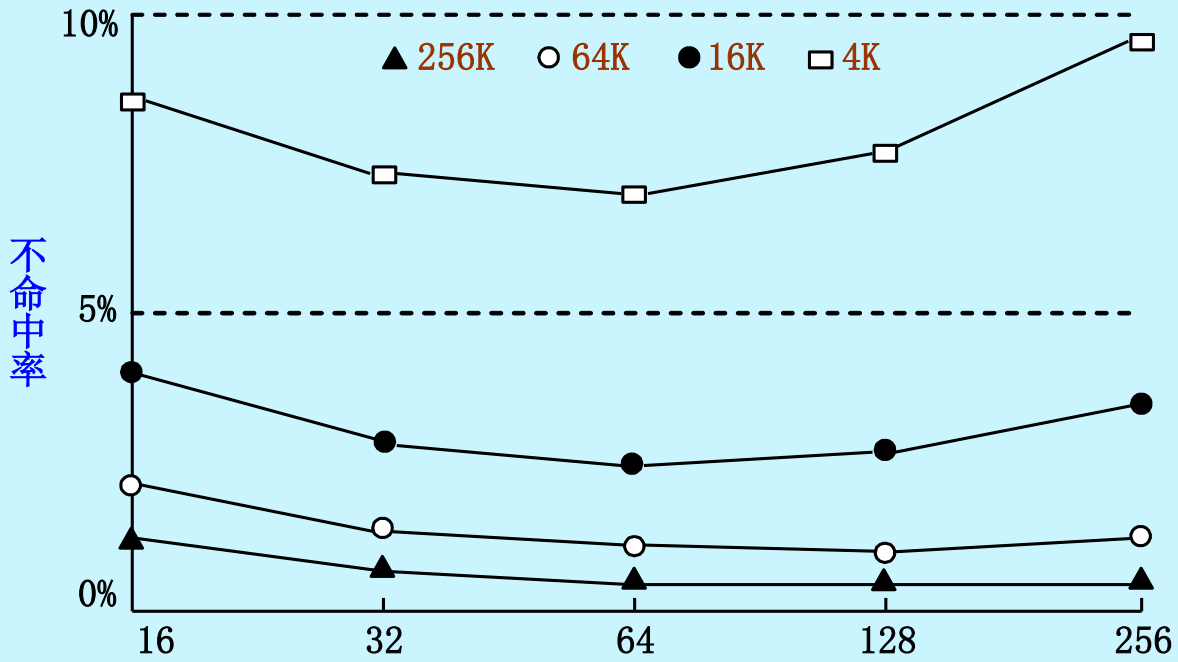


5.5 降低Cache不命中率

- 减少三种不命中的方法
  - 强制性不命中：增加块大小，预取
  - 容量不命中：增加容量
  - 冲突不命中：提高相联度
- 许多降低不命中率的方法会增加命中时间或不命中开销



5.5.2 增加Cache块大小



1. 不命中率与块大小的关系
   - 对于给定的Cache容量，当块大小增加时，不命中率开始是下降，后来反而上升了。

     原因：
     - 一方面它减少了强制性不命中；
     - 另一方面，由于增加块大小会减少Cache中块的数目，所以有可能会增加冲突不命中。
   - Cache容量越大，使不命中率达到最低的块大小就越大。
2. 增加块大小会增加不命中开销

5.5.3 增加Cache的容量

1. 最直接的方法是增加Cache的容量
   - 缺点:
     - 增加成本
     - 可能增加命中时间
2. 这种方法在片外Cache中用得比较多

5.5.4 提高相联度

1. 采用相联度超过8的方案的实际意义不大。
2. 2:1 Cache经验规则

   容量为N的直接映象Cache的不命中率和容量为N/2的两路组相联Cache的不命中率差不多相同。
3. 提高相联度是以增加命中时间为代价。

5.5.5 伪相联 Cache 列相联 Cache

1. 多路组相联的低不命中率和直接映象的命中速度

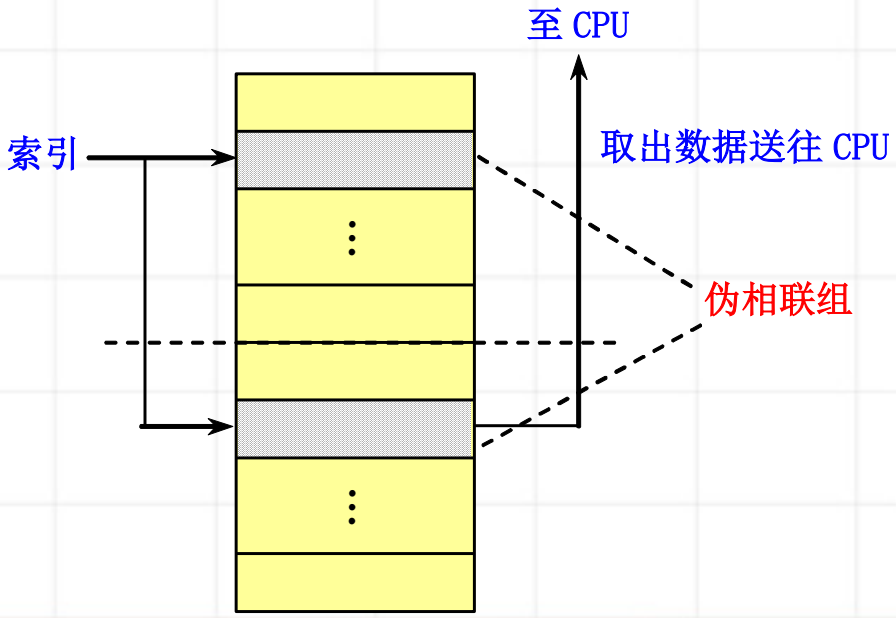
	优 点	缺 点
直接映象	命中时间小	不命中率高
组相联	不命中率低	命中时间大

2. 伪相联Cache的优点

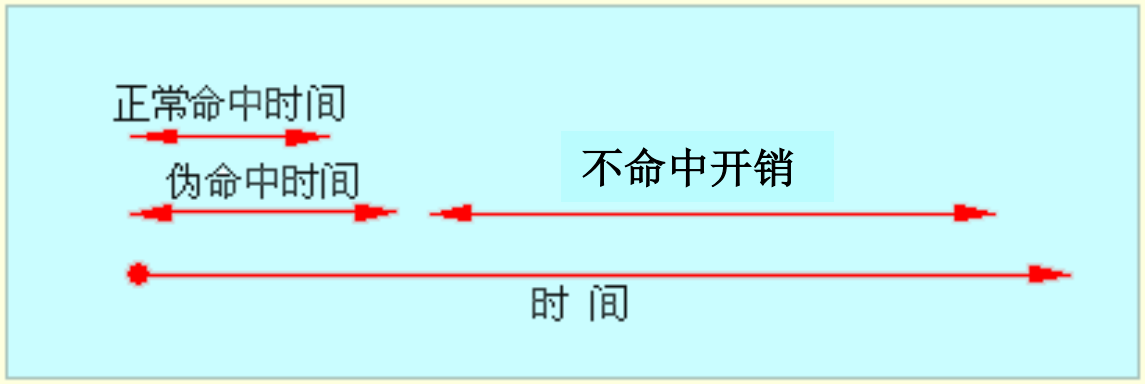
- 命中时间小
- 不命中率低

3. 基本思想及工作原理

在逻辑上把直接映象Cache的空间上下平分为两个区。对于任何一次访问，伪相联Cache先按直接映象Cache的方式去处理。若命中，则其访问过程与直接映象Cache的情况一样。若不命中，则再到另一区相应的位置去查找。若找到，则发生了伪命中，否则就只好访问下一级存储器。



4. 快速命中与慢速命中



5. 缺点：

多种命中时间

5.5.6 硬件预取

1. 指令和数据都可以预取
2. 预取内容既可放入Cache，也可放在外缓冲器中。

   例如：指令流缓冲器
3. 指令预取通常由Cache之外的硬件完成
4. 预取效果

   - Joppi的研究结果
     - 指令预取 (4KB，直接映象Cache，块大小＝16字节)
       - 1个块的指令流缓冲器： 捕获15%～25%的不命中
       - 4个块的指令流缓冲器： 捕获50%
       - 16个块的指令流缓冲器：捕获72%
   - 预取应利用存储器的空闲带宽，不能影响对正常不命中的处理，否则可能会降低性能。

5.5.7 编译器控制的预取

在编译时加入预取指令，在数据被用到之前发出预取请求。

1. 按照预取数据所放的位置，可把预取分为两种类型：
   - 寄存器预取：把数据取到寄存器中。
   - Cache预取：只将数据取到Cache中。
2. 按照预取的处理方式不同，可把预取分为：
   - 故障性预取：在预取时，若出现虚地址故障或违反保护权限，就会发生异常。

- 非故障性预取：在遇到这种情况时则不会发生异常，因为这时它会放弃预取，转变为空操作。

本节假定Cache预取都是非故障性的，也叫做非绑定预取。

3. 在预取数据的同时，处理器应能继续执行。

只有这样，预取才有意义。

4. 编译器控制预取的目的

使执行指令和读取数据能重叠执行。

5. 每次预取需要花费一条指令的开销

- 保证这种开销不超过预取所带来的收益
- 编译器可以通过把重点放在那些可能会导致不命中的访问上，使程序避免不必要的预取，从而较大程度地减少平均访存时间。

5.5.8 编译器优化

基本思想：通过对软件进行优化来降低不命中率。

（特色：无需对硬件做任何改动）

1. 程序代码和数据重组

- 可以重新组织程序而不影响程序的正确性
  - 把一个程序中的过程重新排序，就可能会减少冲突不命中，从而降低指令不命中率。
  - 把基本块对齐，使得程序的入口点与Cache块的起始位置对齐，就可以减少顺序代码执行时所发生的Cache不命中的可能性。

（提高大Cache块的效率）

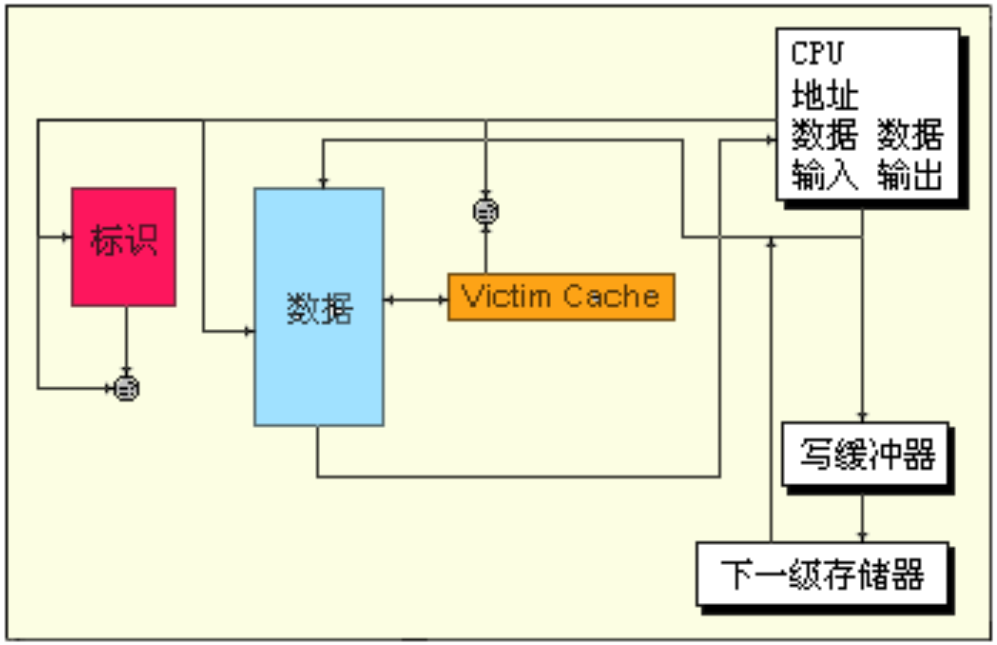
- 如果编译器知道一个分支指令很可能会成功转移，那么它就可以通过以下两步来改善空间局部性：
  - 将转移目标处的基本块和紧跟着该分支指令后的基本块进行对调；
  - 把该分支指令换为操作语义相反的分支指令。
- 数据对存储位置的限制更少，更便于调整顺序。

5.5.9 “牺牲” Cache

1. 一种能减少冲突不命中次数而又不影响时钟频率的方法。

2. 基本思想

   - 在Cache和它从下一级存储器调数据的通路之间设置一个全相联的小Cache，称为“牺牲”Cache（Victim Cache）。用于存放



3. 作用

- 对于减小冲突不命中很有效，特别是对于小容量的直接映象数据Cache，作用尤其明显。
- 例如

  项数为4的Victim Cache:

  能使4KB Cache的冲突不命中减少20%～90%

谢谢大家！