

《 计算机体系结构 》

实验报告

|  |  |
| --- | --- |
| 班 级： | **计213** |
| 学 号： | **21013134** |
| 姓 名： | **徐昊博** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2023年11月

**实验名称** **Cache性能分析实验地点信息楼418实验日期11/22**

1. **实验目的**

（1）加深对Cache基本概念、基本组织结构以及工作原理的理解。

（2）掌握Cache容量、相关度、块大小对Cache性能的影响

（3）掌握降低Cache不命中率的各种方法以及它们对提高Cache性能的好处

（4）理解LRU与随机法的基本思想以及对Cache性能的影响。

1. **实验设备**

Cache模拟器，计算机

1. **实验原理**

缓存（Cache）的命中和不命中是描述在对数据进行访问时，系统是否能够在缓存中找到所需数据的过程。

命中：当CPU需要访问某个数据时，它首先检查缓存中是否已经存在该数据。如果数据在缓存中找到了，这个事件被称为“命中”，CPU可以直接从缓存中获取所需数据，这样访问速度就非常快速。命中率是描述在所有数据访问中，有多少比例的数据可以在缓存中找到。高命中率通常意味着系统的性能较好，因为大部分数据都可以从快速的缓存中获取。

不命中：如果所需数据不在缓存中，这个事件被称为“不命中”。不命中会导致CPU需要从更慢的主存（或者更慢的层级存储）中获取数据，这通常会导致额外的延迟，影响系统性能。

缓存命中率是一个重要的性能指标，它直接影响了系统的响应速度和效率。设计高效的缓存策略、合理设置缓存大小、优化缓存替换算法等可以有效提高命中率，从而提升系统性能。

相联度与替换算法

1. 直接相联（n=1）

对于直接相联，利用模运算直接映射；

1. 组相联和全相联（n≥2）

对于组相联和全相联，主要有以下三种不同的算法：

·随机算法：随机选择一块被替换；

·LRU算法：近期被用到的块大概率还会被再次使用，所以选择近期最久没用的块当做被替换的块；

·FIFO算法：先进先出，最先进来的块被替换。

1. **实验操作及运行结果**

**1、Cache容量对不命中率的影响**

（1）启动MyCache

（2）单击“复位”按钮，把各参数设置为默认值。

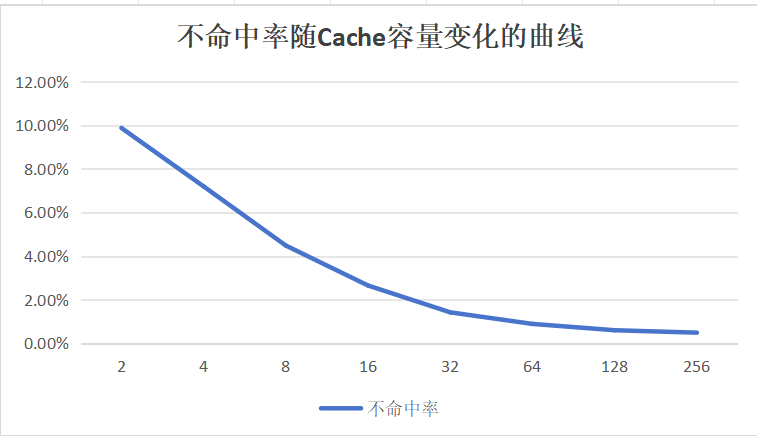
（3）选择地址流文件all.din。方法：选择“访问地址”→“地址流文件”选项，然后单击“浏览”按钮，从本模拟器所在的文件夹下的“地址流”文件夹中选取。

（4）选择不同的Cache容量，分别执行模拟器（单击“执行到底”），在表1.1中记录各种情况下的不命中率。

表1.1 不同容量下Cache的不命中率

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 容量KB | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
| 不命中率 | 9.87% | 7.19% | 4.48% | 2.65% | 1.42% | 0.89% | 0.60% | 0.49% |

1. 以容量为横坐标，画出不命中率随Cache容量变化的曲线。



（6）根据该模拟结果，你能得出什么结论？

随着Cache容量的不断增大，访问地址的不命中率逐步降低。

**2、相联度对不命中率的影响**

（1）单击“复位“按钮，把各参数设置为默认值。

（2）选择地址流文件all.din。

（3）选择不同的Cache相联度，分别执行模拟器，在下表记录各种情况的不命中率。

表1.2 当Cache容量为64KB时，不同相联度下Cache的不命中率

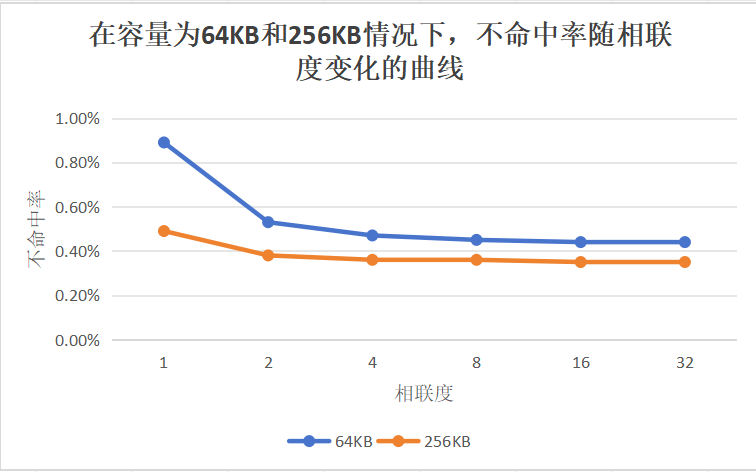
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 相联度 | 1 | 2 | 4 | 8 | 16 | 32 |
| 不命中率 | 0.89% | 0.53% | 0.47% | 0.45% | 0.44% | 0.44% |

（4）把Cache容量改为256KB，重复（3）中工作，并填表1.3

表1.3 当Cache容量为256KB时，不同相联度下Cache的不命中率

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 相联度 | 1 | 2 | 4 | 8 | 16 | 32 |
| 不命中率 | 0.49% | 0.38% | 0.36% | 0.36% | 0.35% | 0.35% |

1. 以容量为横坐标，画出在容量为64KB和256KB1情况下，不命中率随Cache容量变化的曲线。



（6）根据该模拟结果，你能得出什么结论？

在相同的相联度下，Cache的容量越大，不命中率越低，当Cache容量不变时，相联度越高，不命中率越低。

**3、Cache块大小对不命中率的影响**

（1）单击复位按钮，把参数设置为默认值。

（2）选择地址流文件all.din。

（3）选择不同的Cache块大小，不同的Cache容量，分别执行模拟器，记录各种情况下的不命中率。

表1.4 各种情况下Cache的不命中率

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 块大小（B） | Cache容量（KB） | | | | | |
| 2 | 8 | 32 | 64 | 128 | 512 |
| 16 | 12.02% | 5.79% | 1.86% | 1.26% | 0.95% | 0.71% |
| 32 | 9.87% | 4.48% | 1.42% | 0.89% | 0.60% | 0.42% |
| 64 | 9.36% | 4.03% | 1.20% | 0.71% | 0.43% | 0.27% |
| 128 | 10.49% | 4.60% | 1.08% | 0.64% | 0.35% | 0.20% |
| 256 | 13.45% | 5.35% | 1.19% | 0.67% | 0.34% | 0.16% |

（4）分析Cache块大小对不命中率的影响。

当Cache容量≤64KB时：在同一Cache容量下，Cache的不命中率随着块大小的增大呈先降低后升高的趋势。Cache容量≤8KB时，块大小为64B的情况下不命中率最低，优化程度最好；8KB＜Cache容量≤64KB时，块大小为128B的情况下不命中率最低，优化程度最好；

当Cache容量≥128KB时：在同一Cache容量下，Cache的不命中率随着块大小的增大而不断降低。

由此可见，适当增大块尺寸有利于优化Cache命中率。该方法的优点是能够减少首次访问的不命中情况，缺点是会增加映射冲突不命中，增加不命中的时间开销。

**4、替换算法对Cache不命中率的影响**

（1）单击复位按钮，把参数设置为默认值。

（2）选择地址流文件all.din。

（3）对于不同的替换算法、Cache容量和相联度，分别执行模拟器，记录各种情况下的的不命中率。

表1.5 LRU和随机替换算法的不命中率比较

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cache  容量 | 相联度 | | | | | |
| 2路 | | 4路 | | 8路 | |
| LRU | 随机算法 | LRU | 随机算法 | LRU | 随机算法 |
| 16KB | 1.71% | 2.08% | 1.33% | 2.48% | 1.21% | 2.82% |
| 64KB | 0.53% | 0.64% | 0.47% | 0.73% | 0.45% | 0.73% |
| 256KB | 0.38% | 0.39% | 0.36% | 0.37% | 0.36% | 0.36% |
| 1MB | 0.35% | 0.35% | 0.35% | 0.35% | 0.35% | 0.35% |

（4）分析不同替换算法对Cache不命中率的影响。

由上表分析可知：在其他条件相同的条件下，采用LRU算法比采用随机算法得到的不命中率要低。

所以，相同Cache容量的情况下，适当增加相联度，不命中率会降低。对于大容量的Cache，LRU与随机算法性能相差不大；但是对于小容量的Cache，LRU性能优于随机算法，能够较明显地降低不命中率。

1. **实验中出现的问题和解决方法**

问题：在实验中，起初是选择采用地址流文件eg.din，来进行各种条件下Cache性能的测试，但观察发现，测试中一共有1000组数据，得到的数据差异性较小，不能够明显地反映Cache不命中率随着各种条件的改变而产生的变化规律，这样会影响得出的实验结论的准确性。

解决方案：最终选择换用地址流文件all.din来进行实验，该文件的样本有约两百多万个地址，然后执行模拟器，最终得到的不命中率数据也具有一定的差异性（数据间差值比原来明显），再结合表格、图像等辅助手段，可以进一步总结出影响Cache性能的因素及相关规律。