

《 计算机体系结构 》

实验报告一

|  |  |
| --- | --- |
| 班 级： | **计203** |
| 学 号： | **20002462** |
| 姓 名： | **刘子言** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2022年 11月

**实验名称** Cache性能分析

**实验地点** 信息楼418 **实验日期** 2022.11.16

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**

实验平台采用Cache模拟器。

1. **实验原理**
2. **Cache**

Cache（高速缓冲存储器）是一种特殊的存储器子系统，其中复制了频繁使用的数据以利于快速访问。Cache存储了频繁访问的RAM位置的内容以及这些数据项的存储地址。当处理器引用存储器中的某地址时，Cache便检查是否存有该地址。如果存有该地址，则将数据返回处理器；如果没有保存该地址，则进行常规的存储器访问。因为Cache总是比主RAM存储器的速度快，所以当RAM的访问速度低于微处理器的速度时，常常会使用Cache。

1. **命中与不命中**

（1）命中：要访问的数据在上层存储器找到。

命中率（hit rate）：要访问数据在上层存储器找到的比率；

命中时间（hit time）：进入上层存储器的时间，包含进入时间+判定命中与否时间。

（2）不命中：要访问的数据不在上层存储器，需要从下层存储器读取送到上层存储器，在来访问。

从下层读取数据时，读取包含要访问数据的一整块，根据程序局部性原理，有利于减少接下来的数据访问不命中率；

不命中时间开销（Miss penalty）：从下层将数据替换到上层的时间+将数据送给CPU的时间。

（3）命中时间<<不命中时间开销

（4）不命中的原因

1. **不命中的原因**
2. 首次访问某个数据块；
3. 由于Cache容量有限，将某个数据块丢弃，而后又要访问该数据块；
4. 不同的数据块可以映射到同一个Cache位置，映射冲突也会导致不命中。
5. **相联度与替换算法**
6. 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。

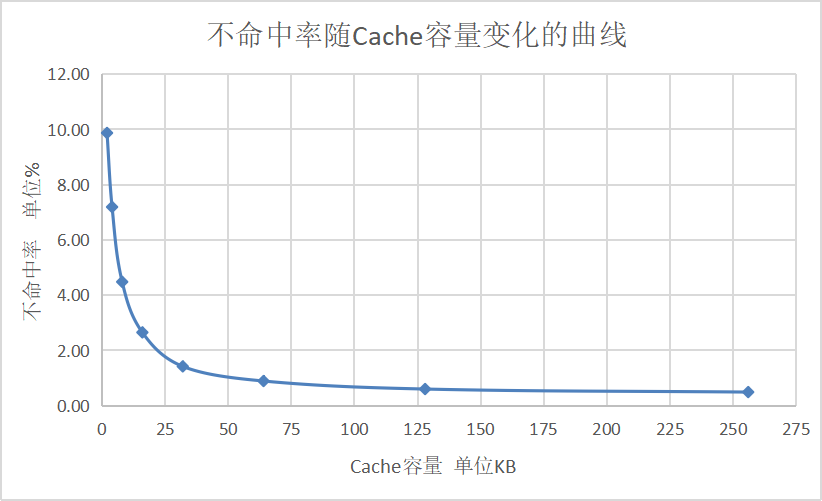
（3）选择地址流文件all.din。方法：选择“访问地址”→“地址流文件”选项，然后单击“浏览”按钮，从本模拟器所在的文件夹下的“地址流”文件夹中选取。

（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% |

（5）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？

结论：在其他条件相同的情况下，随着Cache容量的逐渐增大，不命中率逐渐降低。当Cache容量较小时，不命中率随Cache容量变化较为明显，当Cache容量较大时，不命中率随Cache容量变化的曲线较平缓。

**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

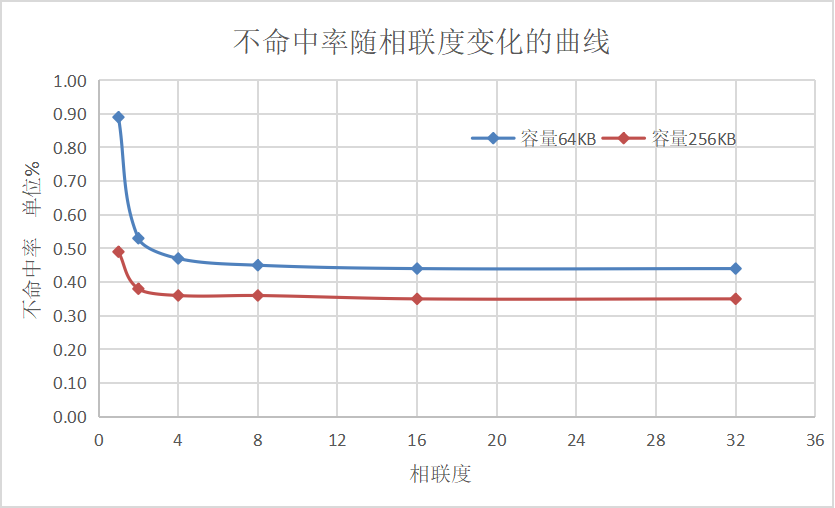
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 相联度 | 1 | 2 | 4 | 8 | 16 | 32 |
| 不命中率 | 0.89% | 0.53% | 0.47% | 0.45% | 0.44% | 0.44% |

（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% |

（5）以相联度为横坐标，画出在容量为64KB和256KB1情况下，不命中率随相联度变化的曲线。



（6）根据该模拟结果，你能得出什么结论？

结论一，在选择同等相联度的情况下，Cache容量为256KB的不命中率均小于容量为64KB的不命中率。这与第1题中得出的结论相映证——在其他条件相同的情况下，随着Cache容量的逐渐增大，不命中率逐渐降低。

结论二，在选择相同Cache容量的情况下，随着相联度的逐渐增加，Cache的不命中率逐渐降低。当相联度较小时，不命中率随Cache相联度变化较为明显，当相联度较大时，不命中率随Cache相联度变化的曲线较平缓。

由此可见，更高的相联度（每组包含更多的块）能够优化Cache命中率。该方法的优点是可以减少冲突导致的不命中（每组块数增加，冲突减少），缺点是增加了命中时间，也增加了能耗。

**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。

由上表分析可知：Cache块大小对不命中率的影响与Cache容量有关。

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% | 2.36% | 1.33% | 2.35% | 1.21% | 3.00% |
| 64KB | 0.53% | 0.75% | 0.47% | 0.77% | 0.45% | 0.76% |
| 256KB | 0.38% | 0.42% | 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。

结论一，相同Cache容量的情况下，适当增加相联度，不命中率会降低。这与第2题中得出的结论相映证——在选择相同Cache容量的情况下，随着相联度的逐渐增加，Cache的不命中率会逐渐降低。

结论二，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**

·实验数据差异性较小的问题

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，该文件的样本最为丰富，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。