实验二 Cache 模拟器实验报告

PB15111662 李双利

一、实验目的

1. 加深对 Cache 的基本概念、基本组织结构以及基本工作原理的理解；
2. 掌握 Cache 容量、相联度、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这些方法对提高 Cache 性能的好处；
4. 理解 LRU 与随机法的基本思想以及它们对 Cache 性能的影响。

二、实现要求

设计与实现一个 Cache 模拟器，能模拟处理器中 Cache 的行为。处理器访存有三种类型：读指令、读数据和写数据，给出访存的地址和类型，我们的 Cache 的模拟器能够进行模拟这种带有 Cache 的访存行为，并能给出统计信息，如访存次数、Cache 命中次数、命中率等。

1．基本要求：模拟器中必须具备下列配置项

   a. 能够设置 Cache 总的大小
   b. 能够设置 Cache 块的大小
   c. 能够设置 Cache 的映射机制：直接映射、n-路组相联
   d. 能够设置 Cache 的替换策略：LRU、FIFO …
   e. 能够设置 Cache 的写策略：写回法、写直达法

2．较高要求：模拟器中可以选择支持下列配置

   a. 能够设置将 Cache 分为数据 Cache 和 指令 Cache
   b. 能够设置预取策略
   c. 能够设置写不命中的调块策略
   d. 有友好的操作界面，如使用界面来配置 Cache

三、设计思想

（1）基本设计

a）建立三个类分别表示 Cache、Block 和 Instruction。

b）其中 Cache 类用于实例化一个 cahce，即每次运行程序时，都会新建一个 cache 对象，根据前端界面上在 box 里的选择 index，得到 cache 的大小和 block 的大小作为参数传入。同时会初始化 cache，将 cache 模拟相关的各种参变量都初始化为 0。

c）从文件读取指令后存到字符串数组中，然后在分步执行时，每次从数组中读取一条指令，实例化一个 Instruction 对象，传入的参数是字符串，由于要将指令根据位数划分为 tag 标记域、组号、块偏移和块内偏移，所以要将原来的十六进制字符串转为二进制，然后就能根据原地址转换为 tag 标记域、组号、块偏移和块内偏移，用于在 cache 的读写等操作。

d）cache 读写的过程是在读取指令之后，传入组号到读写方法中，在该组号内进行遍历，如果找到对应 tag，表示读/写命中，否则的话 cache 未命中，进行块的替换策略。

（2）映射机制

Cache 模拟器支持 n-路组相联的映射机制，根据界面中的 index 选择，决定 cahce 是多少路的组相联，从而指定在 cache 中有多少组以及每组里面的块的数目。这样在读写的时候会取每一组里面找到 tag 标记域是否对应。

（3）替换策略

模拟器支持三种替换策略

a）FIFO

先进先出的替换策略是每次替换块时，选取的被替换块是在当前组中

最先进入的块，即按照队列的 FIFO 策略。实现建立一个记录进入时间的数组，一个块需要替换时首先遍历这个 FIFO 记录数组，然后从中选择最先进入的块并且替换即可。

b）LRU

LRU 策略是每次替换的块是最近最少使用的那个块，算法根据数据的历史访问记录来进行淘汰数据，其核心思想是“如果数据最近被访问过，那么将来被访问的几率也更高”。所以用一个 LRU 数组记录每一个块的最近使用情况，每次替换时即从这个数组中找到最近最少使用的那个块进行替换。

c）随机替换策略

每次替换时，随机选择一个块进行替换即可。

（4）写策略

a）写直达

当 cache 写命中时，cache 与主存同时发生写修改。所以在该策略下进行实现时，修改 cache 的同时会向主存中写入，这样 cache 的脏块标记为 false。

b）写回法

当 CPU 对 cache 写命中时，只修改 cache 的内容不立即写入主存，只当此行被换出时才写回主存。所以每次写入 cache 时，会出现不一致的情况，cache 块的脏块标记需要设为 true。

（5）预取策略

采用不命中预取时，如果某次读数据或者读指令未命中，那么首先将不命中块替换到 cache 中去，然后根据预取策略接着访问下一个地址的读操作，由局部性原理可知该预取策略的合理性。需要主要的是接着访问的下一地址的命中/缺失也要记录到 cache 模拟的统计中。

（5）不命中调块策略

a）按写分配

先从内存中现将该块调入到 cache 中，然后将数据写入 cache 中对应

的位置，所以要调用块的替换方法进行替换。

b）不按写分配

可以不将该数据调入 cache 中，直接写到内存中。主存的访问次数加一，无需进行 cache 内的块替换。

（6）界面优化

四、实验分析和结论

在设计的 cache 模拟器上分别针对 Cache 容量、映射机制、块大小和替换算法进行实验并且记录数据进行作图分析。（以下实验执行的均为 spice.din 测试程序）

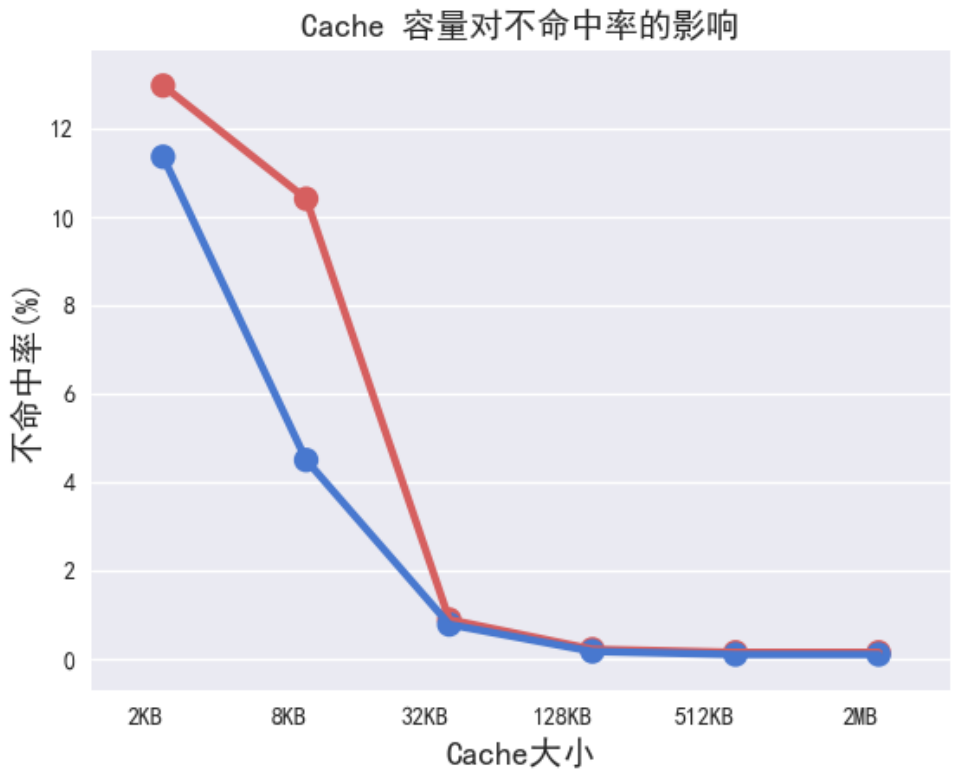
（1）Cache 容量对不命中率的影响

设定 Cache 的相联度为直接映射，替换策略为 LRU，预取策略为不预取，写策略为写回法，写不命中调块策略为按写分配，在块大小分别为 64B 和

128B 条件下改变 Cache 容量进行实验，得到：

Cache 容量	2KB	8KB	32KB	128KB	512KB	2MB
不命中率（块:64B）	12.97%	10.41%	0.89%	0.22%	0.15%	0.15%
不命中率（块:128B）	11.36%	4.5%	0.78%,	0.17%	0.1%	0.1%

绘制曲线得到：



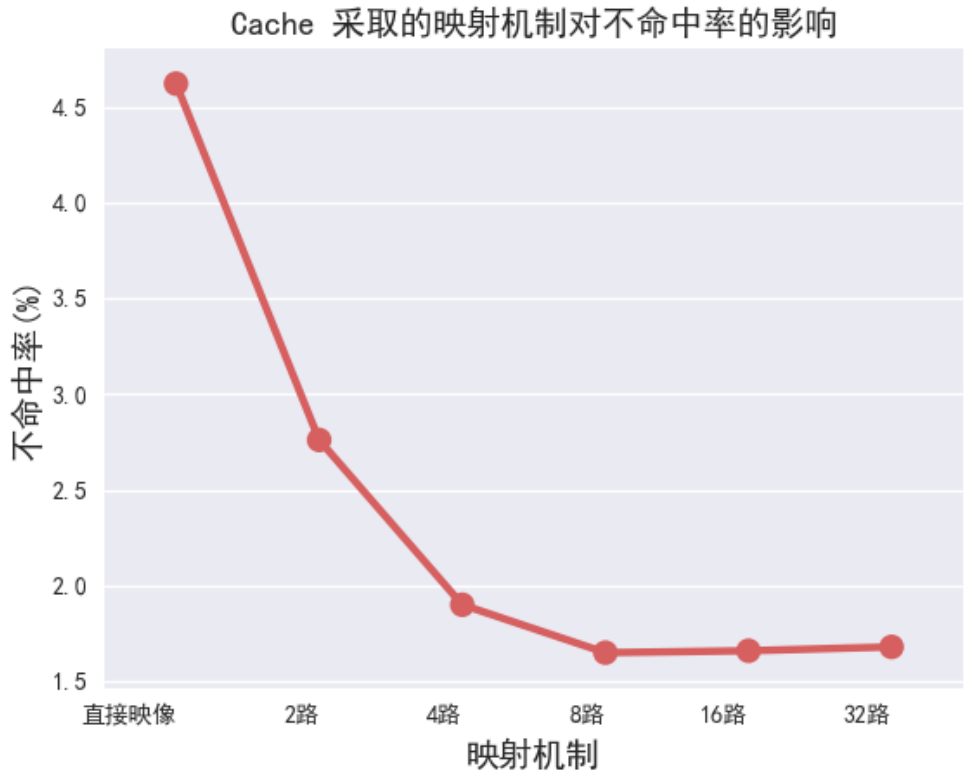
Cache 的不命中率随它的容量的增加而降低，它们之间的关系曲线如图所示。在 Cache 容量比较小的时候，命中率提高得非常快，但根据边际效应递减原理随着 Cache 容量的增加，命中率提高的速度逐渐降低。当 Cache 的容量增加到无穷大时，命中率可望达到 100%，但是这在实际是做不到的。当 Cache 的容量达到一定值之后，再增加 Cache 容量，命中率的提高很少。

（2）Cache 采取的映射机制对不命中率的影响

设定 Cache 大小为 8KB，块大小为 32B，替换策略为 LRU，预取策略为不预取，写策略为写回法，写不命中调块策略为按写分配，在此条件下采用不同的映射机制进行实验，得到：

映射机制	直接映射	2 路组相联	4 路组相联	8 路组相联	16 路组相联	32 路组相联
不命中率	4.62%	2.76%	1.9%	1.65%	1.66%	1.68%

绘制曲线得到：



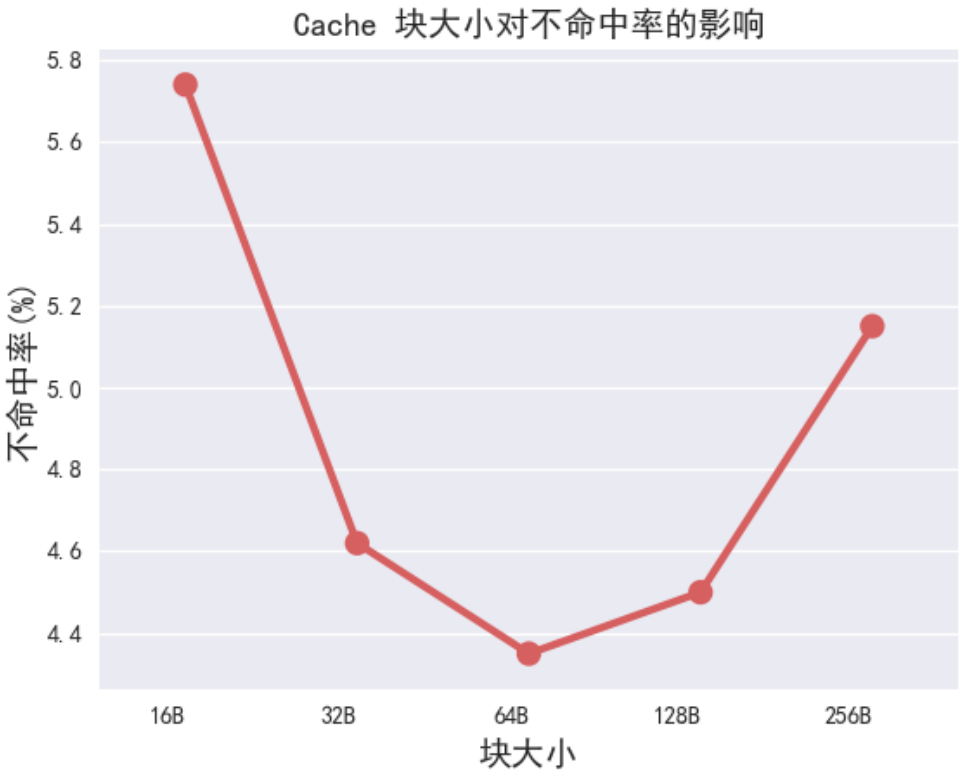
根据曲线图可知，随着映射机制中组相联组中的块数增加其不命中率降低，到达 8 路组相联之后，其不命中率趋于不变，影响其不命中的因素来源于其他的 cache 因素。应用直接相联的方式地址映象方式简单，数据访问时，只需检查区号是否相等即可，但替换操作频繁，命中率比较低。

（3）Cache 块大小对不命中率的影响

设定 Cache 大小为 8KB，相联度为直接映射，替换策略为 LRU，预取策略为不预取，写策略为写回法，写不命中调块策略为按写分配，在此条件下改变 Cache 的块大小进行实验，得到：

块大小	16B	32B	64B	128B	256B
不命中率	5.74%	4.62%	4.35%	4.50%	5.15%

绘制曲线得到：



由曲线可知， 对于给定的 Cache 容量，当块大小增加时，命中率开始时处于上升趋势，后来反而会下降。

当 Cache 的块容量很小，组的数目就多，主存中的某一块可以映象到 Cache 中的块数就少，所以此时，Cache 的命中率低。 随着块大小的增加，由于程序的空间局部性起主要作用，同一块中数据的利用率比较高。因此，Cache 的命中率开始升高。

但如果块变得过大的话，会减少装入 Cache 的总行数，而且，也会使得离所访问的位置较远的块被再次使用的概率变小。因此，这种增加趋势在某一个“最佳块大小”处使 Cache 命中率达到最大值。在这一点以后，命中率随着块大小的增加反而减小。

（4）Cache 替换算法对不命中率的影响

设定替换策略为 LRU，预取策略为不预取，写策略为写回法，写不命中调块策略为按写分配，分别在三种 Cache 大小、块大小和映射机制条件下采用三种替换策略进行实验，得到：

替换策略	LRU	FIFO	RAND
不命中率（条件 1）	3.05%	3.37%	11.99%
不命中率（条件 2）	0.55%	0.63%	5.06%
不命中率（条件 3）	2.83%	3.16%	16.29%

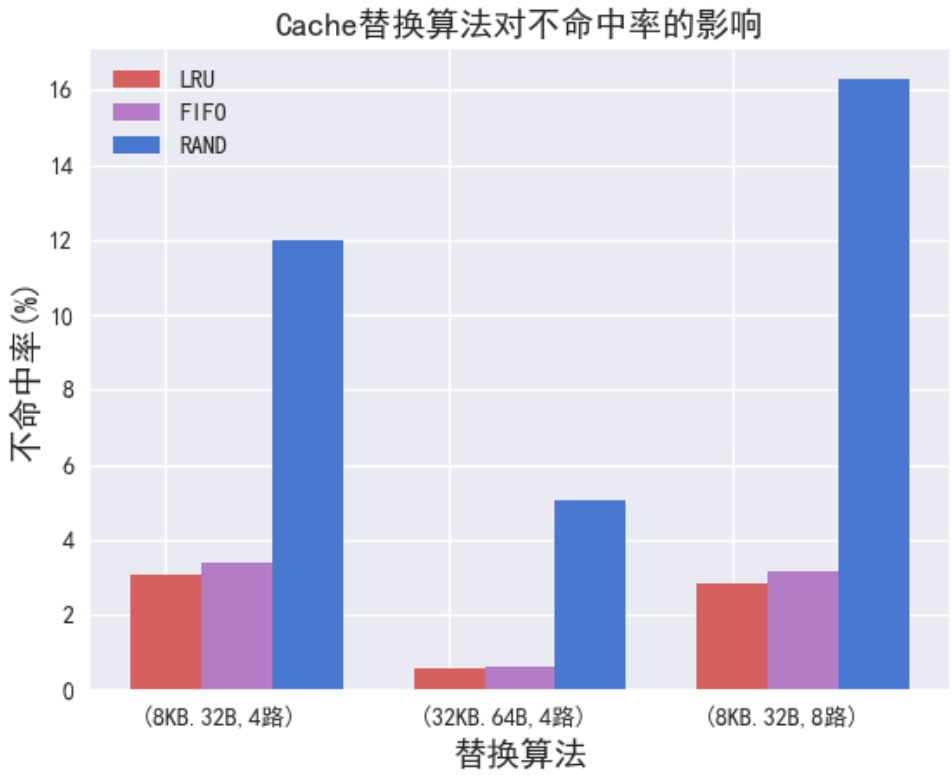
其中：

条件 1 为 Cache 大小为 8KB，块大小为 32B，相联度为 4 路组相联

条件 2 为 Cache 大小为 32KB，块大小为 64B，相联度为 4 路组相联

条件 3 为 Cache 大小为 8KB，块大小为 32B，相联度为 8 路组相联

绘制曲线可得：



根据曲线可知最近最少使用(LRU)和先进先出(FIFO)策略的效果较好，且 LRU 略微优于 FIFO。

五、实验总结

通过本次 cache 模拟器的设计，对于在体系结构课程中的 Cache 模块有了更加深入的理解，真正取实现一个 cache 模拟器需要从指令的读取加载开始到替换策略、预取策略的算法等 cache 的具体策略，这样对 cache 的整个工作原理和优化作用有了整体的认识。此外，还进行了对 cache 不命中率影响因素的分析，通过设定不同的参数来分析不同的因素对 cache 性能的作用，这对于如何选择一个发挥最佳性能的 cache 有重要的指导意义，但在该实验中仅分析了其命中率的影响因素，在实际中对于 cache 要考虑的还有时间性能。