计53 李诗懿

2015011310

系统结构实验一

实验报告

一、理解学习 LRU 及其它已经提出的 Cache 替换策略

Cache替换算法的目的是在出现缺失时，选择Cache中已有的一个内容，删掉，并放入刚才访问的内容。

（1）Random：随机删除一页。随机替换没有用到任何历史的信息，所有元素都是公平的被替换掉，一方面没有考虑到时间局部性和空间局部性，所以优化理论上不会很大，但是另一方面，随机替换很难被构造数据卡掉。

（2）FIFO：删除存入时间最早的一页。 也就是先进先出， 这个主要是利用了我们程序实现上的时间局部性，刚被访问的内容，大概率还会被再次访问。

（3）LRU：删除最近修改时间最早的一页，本质上也是利用了时间局部性，只是和FIFO不同，FIFO只考虑了其进入队列的时间，而LRU则记录了其最近一次被使用的时间。

二、我使用的替换策略的核心思想和算法

**DirtyLRU方法**： 同时考虑dirty位，和在LRU栈上的位置。

**核心思想**： 我们知道将Cache中一个dirty的内容替换，需要访存，而访存的代价是高昂的，我们应当尽量避免替换dirty的内容，希望能优化CPI，但同时我们也要兼顾时间局部性，所以在dirty情况相同的时候，我们用LRU来比较。

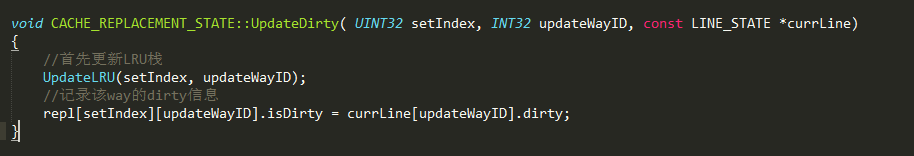
**算法**： 当需要替换的时候，我们将如下代价**最小**的内容替换：

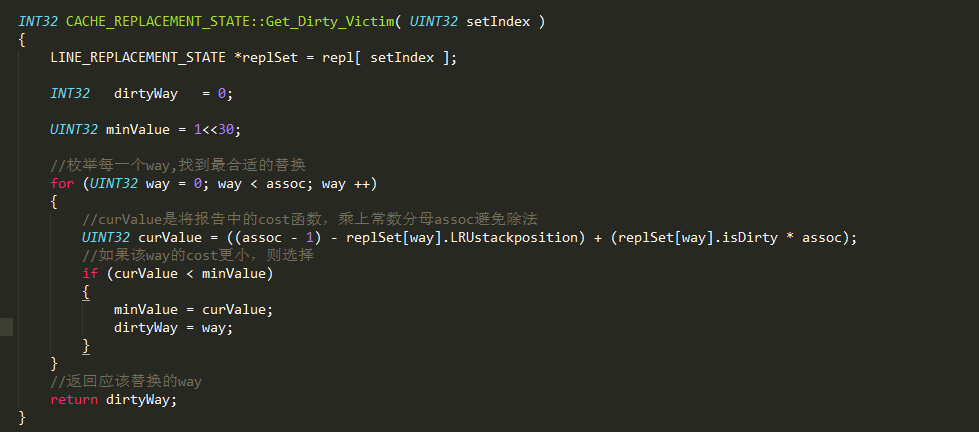
Cost(way) = isDirty(way) + ((assoc - LRUStackPosition(way)) / assoc)

如果way 是dirty的,isDirty(way) = 1，否则等于0，LRUStackPosition(way)表示way在LRU栈上的位置，如果way是最新一次访问的，则等于0，最久一次访问的，则等于assoc, assoc表示组相连数， 由于isDirty取值0到1，所以后面的LRUStackPosition也需要归一化。 所以如果一个way又不是dirty，又很久没有访问过了，它就应该被替换掉。

具体代码实现：

主要实现两个函数，Get\_Dirty\_Victim和UpdateDirty





三、对不同的 Cache 替换策略进行测试的方法

使用模拟器的指令对trace进行测试

四、不同替换策略下，程序的运行时间、Cache 命中率受到的影响

1. MISS RATE: 如下是LRU，Random, 以及我的策略DirtyLRU的MissRate实验结果，横坐标是测试程序的编号。

2. CPI : 如下是LRU，Random, 以及我的策略DirtyLRU的CPI实验结果，横坐标是测试程序的编号。

五、分析与总结

可以发现大部分情况下我使用的算法会更优于LRU算法，说明我的算法更好的衡量了未来访问的趋势。但是也可以发现，在某些数据中还是LRU算法的数据更好，说明我的算法还有改进的空间，可以考虑更多地删除最近一段没有访问页。

通过本次实验，我学习了页替换机制，了解了一些传统的和比较新的页替换算法，让我对系统存储管理有了更深刻的理解和认识。