Project 1: Cache Analysis

Due: 2020.4.9 THU 00:00 (UTC +8)

（一）实验简介

本实验要求同学们使用 C/C++编写一个 Cache 替换策略模拟器。输入为存储器访问 trace，输出为在不同的条件下（如 Cache 块大小，组织方式替换策略，不同的写策略等），在给定 trace 上的缺失率，以及访问 log（记录命中和缺失的结果）。希望同学们能够深入理解不同的 Cache 替换策略，动手实现不同的 Cache 替换策略，同时观测不同的替换策略对于程序运行性能的影响。

（二）实验流程

我们要求同学们实现的 Cache 模拟器能够实现：

1、读入 trace 文件。

本实验的第一步要求同学们写程序解析 trace 文件，作为 Cache 模拟器的输入。

trace 文件记录了一个程序访问存储器的行为（包括具体是读还是写，具体的访问地址等）。我们假设所有的地址都是按照字节编码的，即每一个地址对应于一个字节，这里我们不需要关注具体的读写内容。我们为大家准备了 10 个不同的 trace，他们的基本情况如下：

	大小（MB）	行数	格式
Astar.trace	11	501468	r/w+64 位地址
Bodytrack_1m.trace	3.1	291187	64 位地址
Bzip2.trace	11	544514	r/w+64 位地址
Canneal.uniq.trace	5.7	556784	64 位地址
Gcc.trace	6.4	515683	l/s+64 位地址
Mcf.trace	11	507700	r/w+64 位地址
Perlbench.trace	11	507441	r/w+64 位地址
Streamcluster.trace	9.3	1054082	64 位地址（有空行）
Swim.trace	3.8	303193	l/s+64 位地址
Twolf.tarce	6.0	482824	l/s+64 位地址

trace 文件中标红的为重点 trace，其他为扩展 trace，重点 trace 的格式是固定的（r/w（读/写）+16 进制表示的 64 位地址），扩展 trace 格式有两种（l/s（Load/Store）+16 进制表示的 64 位地址；只记录地址），有些 trace 地址中间还会有空行。

2、设计实现 Cache 模拟器，根据输入，按照一定的替换算法给出命中或者缺失的情况。

同学们需要设计实现一个 128KB 的 Cache，接受 trace 文件作为输入，能够按照一定的替换策略记录每一次存储器访问的命中和缺失情况，统计整体的缺失率。要求同学们至少测试所有重点 trace。

3、在固定替换策略（LRU），固定写策略（写分配+写回）的前提下，尝试不同的 Cache 布局：
固定替换策略为 LRU，要求尝试的布局包括：不同的块大小（8B， 32B，64B），不同的组织方法（直接映射，全关联，4-way 组关联，8-way 组关联），报告所有 12 种不同组合下在不同 trace 上的缺失率（至少测试所有重点 trace），以文字描述，图表等形式分析不同的块大小和组合方法的特点（要求至少分析元数据开销空间，对缺失率的影响等）。

4、在固定 Cache 布局（块大小 8B，8-way 组关联），固定写策略（写分配+写回）的前提下，尝试不同的 Cache 替换策略：
固定 Cache 布局（块大小 8B，采用 8 路组关联），尝试不同的 Cache 替换策略（至少要求尝试 LRU，随机替换，二叉树替换算法三种替换策略），报告不同 trace 上采用不同替换策略的缺失率（至少测试所有重点 trace），以文字描述，图表等形式分析不同替换策略的特点（至少分析它们占用的空间大小，替换时所执行的动作的差异，对于缺失率的最终影响等）。

5、在固定 Cache 布局（块大小 8B，8-way 组关联），固定替换策略（LRU）的前提下，尝试不同的写策略：
固定 Cache 布局（块大小 8B，8-way 组关联），固定替换策略（LRU），尝试不同的 Cache 写策略（写不分配+写直达，写分配+写回，写不分配+写回，写分配+写直达），报告不同的 trace 上采用不同写策略的缺失率（至少测试所有重点 trace），以图表等形式分析不同写策略的特点，及其在不同 trace 上的表现。

注意：
1、为了尽可能准确地使用软件来模拟硬件的动作，要求同学们使用尽可能少的位数来存储需要的元数据（例如对于 8-way 组关联，写回，块大小为 8B，128KB 的 Cache。需要 11 位来记录 Index，因此 Tag 位需要 50 位，此外还需要 1 位来记录 Dirty，1 位来记录 Valid，这样就一共需要 52 位的元数据，希望同学实际实现的时候使用 7 个 Char（合计 56 位）来记录一行所需要的元数据，在实际进行操作时请使用 C++的位操作来实现 Tag，Valid，Dirty 等位的修改）

2、对于 LRU 算法，要求使用课上介绍的堆栈法来实现。即在有 n 个路时，每一路都需要 $\log_2 n$ 个位来记录当前路在这一组中的顺序，例如对于 8-way 组关联的情况，一共需要 $8*$

$\log_2 8 = 24$ 位来记录 LRU 状态，因此希望同学们实际实现时使用 3 个 Char（合计 24 位）来记录 LRU 状态，请使用 C++的位操作来实现 LRU 状态的修改和维护操作。

3、对于二叉树替换算法，要求使用课上介绍的方法来实现。即在有 n 个路时，一共需要 n 位来记录当前的树状态（n-1 个位来维护树的状态，1 个位来维护是否所有路都 Valid），例如对于 8-way 组关联的情况，一共需要 8 位来记录二叉树的状态，因此希望同学们实际实现时使用 1 个 Char（合计 8 位）来记录二叉树状态，请使用 C++的位操作来修改每一次访问后的二叉树状态。

4、考虑到同学们实际情况的差异，如果实在无法做到上述的存储要求，也可以考虑其他的实现方法。例如存储 55 位元数据，可以使用一个 long long（64 位）来存储。但如果其他方法浪费的空间比较多，和实际硬件存储差异较大，可能会对于分数有影响。请大家在实验报告中如实说明自己的具体实现方法，助教会核对代码与实验报告中的实现是否一致，如果发现不一致，按作弊处理，请大家秉承诚实守信的学风。

扩展加分项：
（1）尝试其他 Cache 替换方法，特别是近年来新提出的 Cahce 替换方法（相关参考文献参见附录，也鼓励大家自己调研，请标明参考文献），分析他们的特点以及在 trace 上的表现。
（2）尝试对于 trace 做更为细致的分析，例如使用一些静态分析的方法来衡量不同 trace 的局部性强弱（例如可以定义一些统计量来描述局部性），分析局部性的差异对于缺失率的影响。（有一些 trace 中有一些提示信息，例如 Streamcluster 中访问 trace 聚成簇的形式，不同簇之间用空行隔开）。
（3）尝试模拟 Cache 优化课程上所介绍的各类 Cache 优化方案（比如可以尝试模拟多级 Cache，路预测算法等）。

（三）需要提交的文件：
1、实验源代码及编译方法（如给出 Makefile 等）
2、访问 Log：固定 Cache 布局（块大小 8B，采用 8-way 组关联），固定 LRU 替换策略，固定写策略（写分配+写回）时，给出每一个重点 trace 的访问 Log（我们规定 Log 的格式为每一行一个 Hit or Miss 以记录一次访问的结果）
3、实验报告：给出要求报告的各种情况下的缺失率，给出自己的分析（请认真完成实验报告，过于简略将可能影响分数）

有问题请联系：wei-jy19@mails.tsinghua.edu.cn

参考文献和网站

ISCA Cache Replacement Championship：https://www.jilp.org/jwac-1/

Sreedharan S, Asokan S. A cache replacement policy based on re-reference count[C]//2017 International Conference on Inventive Communication and Computational Technologies (ICICCT). IEEE, 2017: 129-134.

Duong N, Cammarota R, Zhao D, et al. SCORE: A score-based memory cache replacement policy[C]. 2010.

Peress Y, Finlayson I, Tyson G, et al. CRC: Protected lru algorithm[C]. 2010.

Qureshi M K, Jaleel A, Patt Y N, et al. Adaptive insertion policies for high performance caching[J]. ACM SIGARCH Computer Architecture News, 2007, 35(2): 381-391.