華中科技大學

2018

计算机系统结构实验报告

题　　目：	计算机系统结构实验报告
专　　业：	计算机科学与技术
班　　级：	CS1503
学　　号：	U201514559
姓　　名：	周铭昊
电　　话：	15802740273
邮　　件：	630212894@qq.com
完成日期：	2018-05-03 周四晚上

计算机科学与技术学院

目 录

1 实验概述

1.1 实验目的

理解 cache 工作原理；

加深 Cache 缓存组成结构对 C 程序性能的影响的理解。

1.2 实验内容

第一部分：编写一个 200-300 行的 C 程序来模拟 Cache 缓存的行为；

第二部分：在参考 Cache 实现的基础上，优化一个矩阵转置函数，以最小化缓存不命中（cache miss）的数量。

1.3 实验环境

Linux 64-bit + valgrind 软件包（第二个实验需要），C 语言

1.4 实验要求

实验内容一：

- 任务：在 csim.c 提供的程序框架中，编写实现一个 Cache 模拟器：
  - 输入：内存访问轨迹
  - 操作：模拟缓存相对内存访问轨迹的命中/缺失行为
  - 输出：命中、缺失和（缓存行）淘汰/驱逐的总数
- 具体要求：完成的 csim.c 文件应能接受与参考缓存模拟器 csim-ref 相同的命令行参数并产生一致的输出结果。完成的 csim.c 文件应能接受与参考缓存模拟器 csim-ref 相同的命令行参数并产生一致的输出结果。
- 编程要求：

- 模拟器必须在输入参数 s、E、b 设置为任意值时均能正确工作——即需要使用 malloc 函数（而不是代码中固定大小的值）来为模拟器中数据结构分配存储空间。
- 由于实验仅关心数据 Cache 的性能，因此模拟器应忽略所有指令 cache 访问（即轨迹中“I”起始的行）
- 假设内存访问的地址总是正确对齐的，即一次内存访问从不跨越块的边界——因此可忽略访问轨迹中给出的访问请求大小
- main 函数最后必须调用 printSummary 函数输出结果，并如下传之以命中 hit、缺失 miss 和淘汰/驱逐 eviction 的总数作为参数：

printSummary(hit_count, miss_count, eviction_count);

实验内容二：

- 任务：在 trans.c 中编写实现一个矩阵转置函数 transpose_submit，要求其在参考 Cache 模拟器 csim-ref 上运行时对不同大小的矩阵均能最小化缓存缺失的数量

char transpose_submit_desc[] = "Transpose submission";

void transpose_submit(int M, int N, int A[N][M], int B[M][N]);

- 实现要求：
  - 限制对栈的引用——在转置函数中最多定义和使用 12 个 int 类型的局部变量，同时不能使用任何 long 类型的变量或其他位模式数据以在一个变量中存储多个值。
    - 原因：实验测试代码不能/不应计数栈的引用访问，而应将注意力集中在对源和目的矩阵的访问模式上
  - 不允许使用递归。如果定义和调用辅助函数，在任意时刻，从转置函数的栈帧到辅助函数的栈帧之间最多可以同时存在 12 个局部变

量。

- 例如，如果转置函数定义了 8 个局部变量，其中调用了一个使用 4 个局部变量的函数，而其进一步调用了一个使用 2 个局部变量的函数，则栈上总共将有 14 个变量，则违反了本规则。

- 转置函数不允许改变矩阵 A，但可以任意操作矩阵 B。
- 不允许在代码中定义任何矩阵或使用 malloc 及其变种。

2 实验方案设计

2.1 编写 Cache 模拟器

修改 cism.c 文件，首先理解每个变量的含义和每个函数的作用，采用 LRU 算法，用结构体数组来保存 cache 数据。

2.1.1 变量及结构体定义

表 2.1 变量及结构体定义

变量名	说明
Cache_line	Char valid：有效位，标识这块 cache 是否有效
	mem_addr_t tag ：标识位，与内存相对应
	unsigned long long int lru：LRU 计数位
cache_set_t	一组 cache，其中包含多行 cache_line
cache_t	所有的 cache，其中包含多组 cache_set_t
s	组索引位数
S	组数，等于 2 的 s 次幂
b	块内偏移位
B	块大小，等于 2 的 b 次幂
E	关联度，每组 cache 包含的行数
miss_count	未命中计数
hit_count	命中计数
eviction_count	LRU 替换计数
eviction_lru	被替换的 cache 的 lru 计数值
eviction_line	被替换的 cache 行号
isHit	命中标志位
isFull	Cache 已满标志位（一组中所有 cache 都有效）
lru_counter	Lru 计数器，每次访存加一

2.1.2 算法思想

Cache 模拟器的整个运行过程是：先读取命令行的内容，然后设定 s，E 和 b 的值，调用 initial 函数进行初始化。

Initial 函数用于 cache 的初始化。先判断 S 是否有效，若 s<0 则 cache 组数为 0，输入无效直接退出。然后初始化一个二维数组 cache[S][E]，其中包含 S 组，每组包含 E 行 cache。每个 cache 的 valid 初始化为'n'，表示无效，tag 和 lru 都初始化为 0，同时计算组索引掩码，在计算组号时要用到。初始化完成后进入读取函数，读取轨迹。

replayTrace 函数用于循环读取轨迹文件。先根据命令行输入的文件名 trace_fn，将其按行读取到 buf，每次读取过程判断具体操作是 S、L 还是 M，如果是前两者只需要访存一次，后者需要访存两次，即调用两次 accessData 函数。循环读取直到文件尾，完成后关闭文件。

AccessData 函数用于判断每次访存的 cache 命中情况。首先计算出被访问的 cache 组号和内存标识号，然后通过组号选定这一组 cache_set。在判断访存情况之前，先把 lru_counter 计数器+1，越大代表越新。随后判断是否命中，如果成功命中，则把这块 cache 的 lru 赋值为 lru_counter 的值，跳出。如果未命中，则还需要判断这组 cache 是否还有空位，如果不满，则直接找到某一行没有使用的 cache 进行载入，并把这行 cache 的 lru 赋值为 lru_counter；如果满了，则需要找到 lru 最小的 cache，即最久没有访问的 cache 进行替换，替换完成后，把这组 cache 的 lru 赋值为 lru_counter。所有的访存判断都结束以后，replayTrace 函数也结束读文件。

Freecache 函数用于释放在初始化时申请的 cache 空间。

printSummary 函数用于打印三个计数器（命中、未命中、替换）的值。

2.2 优化矩阵转置操作

2.2.1 局部变量定义

两个索引变量 i，j；用于控制 for 循环。

八个中间变量 tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7, tmp8；用于缓存数组值。

2.2.2 32×32 矩阵转置的优化策略

由于 cache 的规格是： s=5, 组索引 5 位, 共有 2^5=32 组

E=1, 每组包含 1 行

b=5, 块内地址 5 位, 块大小为 2^5=32 Bytes

一个 cache 块可以包含 8 个 int 型变量，由于数组是连续存储的，那么访问 A[0][0] 时，cache 中保存了 A[0][0] 及其相邻变量的值，其中可能包含 A[0][1],A[0][2]---A[0][7]。因此按照顺序访问，可以提高 cache 命中率。

读取时每次连续读取 8 个的相邻的 int，通过八个中间变量赋值给数组 B。

2.2.3 64×64 矩阵转置的优化策略

试用了 32*32 的转置策略，结果并不理想，可能是因为连续访存的 8 个 int 型变量分布存储在了不同的 cache 块中，所以改为每次访存 4 个 int，这样访问到的变量位于同一块 cache 的概率要大些。

2.2.4 61×67 矩阵转置的优化策略

与 32*32 矩阵的转置策略一样，都采用一次读取八个连续的数据进行转置，但是 61*67 的矩阵不能整除 8，所以剩下的不能分为 8 块的部分单独解决，采用最基本的矩阵转置，这部分无优化。

3 实验过程与调试

3.1 实验一测试

编译完成后，用 test-csim 测试，结果如图 3.1 所示，程序正确运行。

```
lumos@ubuntu:~/cachelab-handout$ ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     3 (1,1,1)       9       8       6       9       8       6  traces/yi2.trace
     3 (4,2,4)       4       5       2       4       5       2  traces/yi.trace
     3 (2,1,4)       2       3       1       2       3       1  traces/dave.trace
     3 (2,1,3)     167      71      67     167      71      67  traces/trans.trace
     3 (2,2,3)     201      37      29     201      37      29  traces/trans.trace
     3 (2,4,3)     212      26      10     212      26      10  traces/trans.trace
     3 (5,1,5)     231       7       0     231       7       0  traces/trans.trace
     6 (5,1,5)  265189   21775   21743  265189   21775   21743  traces/long.trace
    27

TEST_CSIM_RESULTS=27
```

图 3.1 实验一测试结果

3.2 实验二测试

3.2.1 32*32 矩阵测试

测试结果如图 3.2 所示，miss 计数为 287，得 8 分。

```
lumos@ubuntu:~/cachelab-handout$ ./test-trans -M 32 -N 32

Function 0 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:1766, misses:287, evictions:255

Function 1 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 1 (Simple row-wise scan transpose): hits:870, misses:1183, evictions:1151

Summary for official submission (func 0): correctness=1 misses=287

TEST_TRANS_RESULTS=1:287
```

图 3.2 32*32 矩阵测试

3.2.2 64*64 矩阵测试

测试结果如图 3.3 所示，miss 计数为 1651，得 4 分。

```
lumos@ubuntu:~/cachelab-handout$ ./test-trans -M 64 -N 64

Function 0 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:6546, misses:1651, evictions:1619

Function 1 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 1 (Simple row-wise scan transpose): hits:3474, misses:4723, evictions:4691

Summary for official submission (func 0): correctness=1 misses=1651

TEST_TRANS_RESULTS=1:1651
```

图 3.3 64*64 矩阵测试

3.2.3 61*67 矩阵测试

测试结果如图 3.4 所示，miss 计数为 2192，得 8 分。

```
lumos@ubuntu:~/cachelab-handout$ ./test-trans -M 61 -N 67

Function 0 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 0 (Transpose submission): hits:7455, misses:2192, evictions:2160

Function 1 (2 total)
Step 1: Validating and generating memory traces
Step 2: Evaluating performance (s=5, E=1, b=5)
func 1 (Simple row-wise scan transpose): hits:3756, misses:4423, evictions:4391

Summary for official submission (func 0): correctness=1 misses=2192

TEST_TRANS_RESULTS=1:2192
```

图 3.4 61*67 矩阵测试

4 实验总结与心得

本次实验主要设计实现了 cache 的模拟器以及三个矩阵转置的优化。经过本次课设，我对 cache 的理解更加深刻，对 cache 工作方式的了解更进了一步。Cache 对于程序的运行速度有着重大的影响，于是做完实验马上去了解了一下自己电脑的 cache 规格，只知道是 8m，采用了英特尔智能高速缓存技术，现阶段还不明白怎样的编程方式最适合这种 cache 结构。但在以后编程序时，都会考虑到尽量提高 cache 命中率，这是做完本次实验之后才的思想，也是本次实验最大的收获。

参考文献

[1] DAVID A.PATTERSON(美).计算机组成与设计硬件/软件接口(原书第4版).北京：机械工业出版社.

[2] David Money Harris(美).数字设计和计算机体系结构（第二版）. 机械工业出版社

[3] 秦磊华，吴非，莫正坤.计算机组成原理. 北京：清华大学出版社，2011年.

[4] 袁春风编著. 计算机组成与系统结构. 北京：清华大学出版社，2011年.

[5] 张晨曦，王志英. 计算机系统结构. 高等教育出版社，2008年.

一、原创性声明

本人郑重声明本报告内容，是由作者本人独立完成的。有关观点、方法、数据和文献等的引用已在文中指出。除文中已注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品成果，不存在剽窃、抄袭行为。

特此声明！

作者签字: 周铭昊