Cachelab 实验报告

李梓童 2017202121

一、实验目的

深入理解 Cache 及优化方法

二、实验内容

1）编写一个 Cache Simulator

2）优化矩阵转置操作

三、实验过程

PART A. Writing a Cache Simulator

（1）问题描述

在 csim.c 文件中编写一个 cache 模拟器，它将会读取一个 valgrind 主存作为输入，在此基础上模拟一个命中/不命中的 cache 操作，最后输出命中、不命中和驱逐行的总数。

（2）限制条件

1.cache 模拟器需要在任意 s、E、b 下都能正常工作，因此需要使用 malloc 函数分配相应的存储空间。

（3）解题思路

1.如何从用户输入的指令中解析出有效的参数

根据提示，可以使用 getopt()函数对用户输入的字符进行解析，详见 csim.c 中的 int main()函数部分。

2.如何进行 cache 的存储结构

带写代码时，定义以下两种数据结构。

typedef struct Block {

```
    char valid;             // 判断是否有效
    unsigned long tag;   // tag 位
    int data;               // 数据，实际用于 LRU 策略的判断
} Block;
```

以及

```
typedef struct cache {
    int s;                  //组数（原始值，即指数）
    int b;                  //块大小（原始值即指数）
    int E;                  //行数
    int block_num;          //块的数目
    int size;           //每个 cache 的大小
    Block *cacheBlock;   //每个 cache 的块的集合
} Cache;
```

3.如何模拟 cache 的操作

简单来说，cache 的功能可以概括为：解析读取的地址，根据 s 和 tag 确定 cache 中有无命中；如果没有命中，检查有没有空行，如果有空行，则将数据放入空行；如果没有空行，则根据 LRU 原则进行替换。

参考资料：

《Linux 下 getopt()函数的简单使用》：

http://www.cnblogs.com/qingergege/p/5914218.html

PART B. Optimizing Matrix Transpose

（1）问题描述

在 trans.c 文件中编写矩阵转置功能代码，尽可能地减少缓存不命中的次数。

另，cache 的参数为(s,E,b)=(5,1,5).

（2）限制条件

1.每个转置函数中最多包含 12 个变量。

2.不能使用 long 型数据、位运算等。

3.不能使用递归。

4.程序运行的栈里不能同时存在超过 12 个变量。

5.不能定义新的数组、使用 malloc 函数。

（3）解题思路

1.block 为 32 字节，可以放下 8 个 int；cache 为 32*32 字节。在矩阵转置过程中，冲突不命中主要是在访问同一个块中的两个元素的时候，由于中间访问了其它的块，导致已经加载的块被驱逐，进而第二次访问时不命中。因此，通过一次性访问同一个块中的多个元素、访问完以后不再访问这个块，可以大大地减少冲突不命中的数目。

2.在 32*32 矩阵中，一行有 32 个 int（4 个 block），cache 可以存 8 行，所以只要两个整型数之间相差 8 行的整数倍，那么读取这两个元素所在的 block 就会发生替换。采用 8*8 的数据块，可以尽可能大地避免冲突。而遇到对角线上元素的时候，映射到同一个 block，可以另外开辟一个局部变量来单独处理。

3.在 64*64 矩阵中，一行有 64 个 int（8 个 block），cache 可以存 4 行，所以只要两个整型数之间相差 4 行的整数倍，那么读取这两个元素所在的 block 就会发生替换。如果用和 32*32 矩阵一样进行 8*8 分块，那么在读取数据的时候就会产生冲突。如果进行 4*4 分块，因为存在浪费，还是达不到 1300 的要求（1795misses），所以得进一步优化。

把 8*8 块分成 4 个 4*4 块，按照左上角、右上角、左下角、右下角的顺序进行访问，则在数组 B 在访问其前 4 行前 4 列时，读取的数据可以在进行右上角转置时继续使用；访问后四行前 4 列时，读取的数据可以在进行右下角转置时继续使用。

4.在 61*67 矩阵中，由于矩阵不规则，不好根据 cache 的尺寸来判断，所以延续前两个矩阵优化的方法，在分块策略上进行枚举。

2*2：3111 misses

10*10：2070 misses

14*14：1989 misses

14*14 的分块已经达到<2000 misses 的要求，另外经尝试，16*16 分块也可以。

四、实验心得

通过编写 cache 模拟器和矩阵优化转置，我对 cache 的工作原理有了更深的理解，实验对课堂老师教授的内容起到了很好的巩固作用。

（实验源代码与注释见附件 trans.c、csim.c）