ICS-LAB6 Cachelab
高速缓冲器模拟

哈尔滨工业大学
计算机科学与技术学院

2021年5月28日, Friday

一、实验基本信息

- **实验类型：设计型实验**
- **实验目的**
  - 理解现代计算机系统存储器层级结构
  - 掌握Cache的功能结构与访问控制策略
  - 培养Linux下的性能测试方法与技巧
  - 深入理解Cache组成结构对C程序性能的影响
- **实验指导教师**
  - 任课教师：
  - 实验室教师：
- **人数与分组**
  - 一人一组

一、实验基本信息

- 实验学时：3
- 实验学分：3，本次实验按100分计算，折合成总成绩的3分。
- 实验地点：G712、G709
- 实验环境与工具：
  - X64 CPU；2GHz；2G RAM；256GHD Disk 以上
  - Windows7 64位以上；VirtualBox/Vmware 11以上；Ubuntu 16.04 LTS 64位/优麒麟 64位；
  - Visual Studio 2010 64位以上；TestStudio；Gprof;Valgrind等
- 学生实验准备：禁止准备不合格的学生做实验
  - 个人笔记本电脑
  - 实验环境与工具所列明软件
  - 参考手册: Linux环境下的命令；GCC手册；GDB手册
  - http://docs.huihoo.com/c/linux-c-programming/ C汇编Linux手册
  - http://csapp.cs.cmu.edu/3e/labs.html CMU的实验参考
  - http://www.linuxidc.com/ http://cn.ubuntu.com/ http://forum.ubuntu.org.cn/

二、实验要求

- 学生应穿鞋套进入实验室
- 进入实验室后在签到簿中签字
- 实验安全与注意事项
  - 禁止使用笔记本电脑以外的设备
  - 学行生不得自行开关空调、投影仪
  - 学生不得自打开窗户
  - 不得使用实验室内的其他实验箱、示波器、导线、工具、遥控器等
  - 认真阅读消防安全撤离路线
  - 突发事件处理：第一时间告知教师，同时关闭电源开关。
- 遵守学生实验守则，爱护实验设备，遵守操作规程，精心操作，注意安全，严禁乱拆乱动。
- 实验结束后要及时关掉电源，对所用实验设备进行整理，设备摆放和状态恢复到原始状态。
- 桌面整洁、椅子归位，经实验指导教师允许后方可离开

三、实验预习

- 上实验课前，必须认真预习实验指导书（PPT或PDF）
- 了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。
- 画出存储器的层级结构，标识其容量价格速度等指标变化
- 用CPUZ等查看你的计算机Cache各参数，写出Cache的基本结构与参数：缓存大小C、分组数量 S、关联度/组内行数 E、块大小 B，及对应的编码位数：组索引位数s、 e 、块内偏移位数b
- 写出Cache的各种读策略与写策略
- 掌握Valgrind、gprof的使用方法

四、实验内容与步骤

- 1.环境建立
  - Windows下Visual Studio 2010 64位
  - Ubuntu下gcc、gprof, valgrind
- 2.获得实验包
  - 从实验教师处获得下 cachelab-handout.tar
  - 也可以从课程QQ群下载，也可以从其他同学处获取。
  - HIT与CMU的不同
- 3.Windows下性能分析
  - 分析analyzer
- 4.Ubuntu下gprof的使用
  - gprof 教材上有
- 5. Ubuntu下valgrind的使用
  - valgrind **课程提供的虚拟机已经安装！**

四、实验内容与步骤

- 6. 实验包分析: $ tar vxf cachelab-handout.tar
  数据包中包含下面文件：
  - csim.c：实验中需要修改和提交的Cache模拟程序
  - trans.c：实验中需要修改和提交的矩阵转置程序
  - csim-ref：供参考的二进制可执行Cache模拟器（模拟一个具有任意大小、关联度和LRU（least-recently used）替换策略的Cache）
  - traces子目录：包含一组引用内存访问轨迹文件-reference trace files-由valgrind程序生成，用以评估Cache模拟器的正确性
  - test-csim：测试程序，用于验证Cache模拟器在上述引用内存访问轨迹上的正确性
  - test-trans.c：测试程序，用于测试矩阵转置函数实现的正确性、并评估程序性能

四、实验内容与步骤

- 内存访问轨迹文件
  - 位于traces子目录中，用以评估Cache模拟器的正确性
  - 记录了某一程序在运行过程中访问内存的序列及其参数（地址、大小等）
  - 每行记录1或2次内存访问的信息，格式：

    [0-1个空格] **operation** **address**, **size**

    operation（操作）：内存访问的类型。I-指令装载，L-数据装载， S-数据存储，M-数据修改（即数据装载后接数据存储）

    address：64-bit十六进制内存地址

    size：访问的内存字节数量

  - 示例：

    I 0400d7d4, 8
    M 0421c7f0, 4
    L 04f6b868, 8
    S 7ff005c8, 8

注意：I符号前没有空格，而每个M、L、S符号前总有一个空格，代表对应的数据访问是由指令（执行）引起的

7.编写Cache模拟器

- **任务**：在csim.c中添写代码，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(基于LRU算法)的总数
- **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。
- **csim-ref命令行格式：**
  - -h: 显**csim-ref [-hv] -s <s> -E <E> -b <b> -t <tracefile>**
  - 示帮助信息（可选）、-v: 显示轨迹信息（可选）
  - -s <s>: 组索引位数
  - -E <E>: 关联度（每组包含的缓存行数）
  - -b <b>: 缓存行的内存块内地址位数
  - -t <tracefile>: 内存访问轨迹文件名

7.编写Cache模拟器

- **csim-ref示例：**

```
$>./csim-ref -v -s 4 -E 1 -b 4 -t traces/yi.trace
L 10,1 miss
M 20,1 miss hit
L 22,1 hit
S 18,1 hit
L 110,1 miss eviction
L 210,1 miss eviction
M 12,1 miss eviction hit
hits:4 misses:5 evictions:3
```

7.编写Cache模拟器

- Cache模拟器编程要求
  - 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。
  - 由于本实验仅关心数据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
  - 必须在main函数最后调用printSummary函数，并按如下方式传送：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);

7.编写Cache模拟器

- csim.c代码框架:

```
#include "cachelab.h"
int hit_count = 0, miss_count = 0, eviction_count = 0;
int main() {        … …
    printSummary(hit_count, miss_count, eviction_count);
    return 0;
}
```

- 每一数据装载(L)或存储(S)操作可引发最多1次缓存缺失(miss)
- 数据修改操作(M)可认为是同一地址上1次装载后跟1次存储，因此可引发2次缓存命中(hit)，或1次缺失+1次命中+可能1次淘汰/驱逐(evict)

7.编写Cache模拟器

- Cache性能测试

8个测试用例——不同Cache参数和访问轨迹

```
linux> ./csim -s 1 -E 1 -b 1 -t traces/yi2.trace
linux> ./csim -s 4 -E 2 -b 4 -t traces/yi.trace
linux> ./csim -s 2 -E 1 -b 4 -t traces/dave.trace
linux> ./csim -s 2 -E 1 -b 3 -t traces/trans.trace
linux> ./csim -s 2 -E 2 -b 3 -t traces/trans.trace
linux> ./csim -s 2 -E 4 -b 3 -t traces/trans.trace
linux> ./csim -s 5 -E 1 -b 5 -t traces/trans.trace
linux> ./csim -s 5 -E 1 -b 5 -t traces/long.trace
```

访存轨迹文件示例

```
 L 10,1
 M 20,1
 L 22,1
 S 18,1
 L 110,1
 L 210,1
 M 12,1
```

```
I  0040056d,3
 L 7ff00038c,4
I  00400570,3
 S 00600a70,4
I  00400573,4
 M 7ff000388,4
I  00400577,4
 L 7ff000388,4
I  0040057b,2
I  0040053c,3
 L 7ff000384,4
```

8.Cache性能测试

- test-csim测试程序
- 依次使用前述每一个试用例对csim进行测试
  - 对每一测试轨迹文件，test-csim比较csim与csim-ref的三个指标是否一致：Cache的Hits（命中）、Misses（缺失）、Evicts（淘汰/驱逐）
    - 若csim与参考csim-ref模拟器输出指标相同则判为正确
  - 计算csim获得的分数：每个用例的每一指标5分（最后一个用例10）

test-csim测试程序

```
linux> ./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
```

9.优化矩阵转置操作

- 矩阵转置：设A为m×n阶矩阵（即m行n列），第i行j 列的元素是a(i,j)，即：A=a(i,j)
- B是A的转置：B为n×m阶矩阵，即 b (i,j)=a (j,i)（B的第i行第j列元素是A的第j行第i列元素），记A'=B。
- **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]);
```

矩阵转置实现要求

- 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、不能使用任何long类型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栈帧之间最多可以同时存在12个局部变量。
  - 例如，如转置函数定义了8个局部变量，并调用一个使用4个局部变量的函数，后者又调用了一个使用2个局部变量的函数，则栈上最多将有14个变量，则违反本规则！
- 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。

10.矩阵转置的性能测试

- 实验提供了名为test-trans.c的自动测试程序，该程序将调用trans.c中实现的registerFunctions()函数并测试其中注册的每一个转置函数，例如 transpose_submit:

```
registerTransFunction(transpose_submit,
transpose_submit_desc);
```

  - 最多可以向test-trans测试程序注册100个位于trans.c中的不同转置函数实现
  - 最终需选择注册函数实现中的一个，将其重命名/复制到函数transpose_submit中并作为实验结果提交

10.矩阵转置的性能测试

- **测试程序test-trans的原理**
  - 以矩阵大小作为命令行参数（-M、-N）
  - valgrind运行程序tracegen，为每个注册的转置函数生成访存轨迹文件。第i个转置函数的访存轨迹存储于文件trace.f[i]（例如trace.f0，trace.f1，...）中。**test-trans.c:72**
  - 在参考模拟器csim-ref模拟1KB 直接映射、块大小32Bytes的cache（参数s=5、E=1、b=5）上测试访存轨迹文件，将其输出作为评估转置函数的依据。
  - test-trans程序如下运行参考模拟器（**test-trans.c:140**）

./csim-ref -s 5 -E 1 -b 5 -t trace.f[i]

test-trans测试程序运行示例：

linux> make ← 编译和链接test-trans.c和trans.c，从而可访问后者中定义的转置函数

linux> ./test-trans -M 32 -N 32 ← 32×32大小矩阵

Step 1: Evaluating registered transpose funcs for correctness:

func 0 (Transpose submission): correctness: 1

func 1 (Simple row-wise scan transpose): correctness: 1

func 2 (column-wise scan transpose): correctness: 1

func 3 (using a zig-zag access pattern): correctness: 1

trans.c中注册的4个不同转置函数及其测试结果

Step 2: Generating memory traces for registered transpose funcs.

Step 3: Evaluating performance of registered transpose funcs (s=5, E=1, b=5)

func 0 (Transpose submission): hits:1766, misses:287, evictions:255

func 1 (Simple row-wise scan transpose): hits:870, misses:1183, evictions:1151

func 2 (column-wise scan transpose): hits:870, misses:1183, evictions:1151

func 3 (using a zig-zag access pattern): hits:1076, misses:977, evictions:945

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

矩阵转置的评分

- **test-trans程序在三个不同大小的矩阵上测试转置函数的正确性和性能:**
  - 32 × 32 (M = 32, N = 32)
  - 64 × 64 (M = 64, N = 64)
  - 61 × 67 (M = 61, N = 67)
- **针对每一矩阵大小，性能分数线性依赖于发生的Cache缺失总数m:**
  - 32×32：m<300得10分，m>600得0分，否则得(600-m)*10/300分。
  - 64×64：m<1300得10分，m>2000得0分，否则得(2000-m)*10/700分。
  - 61×67：m<2000得20分，m>3000得0分，否则得(3000-m)*20/1000分。

11.实验提交

- 修改完成两部分实验的结果文件csim.c和trans.c后，在实验数据的根目录中执行如下命令进行编译：
  linux> make clean
  linux> make
- 每次如上执行make命令时，相应Makefile将创建一个名为"-handin.tar"的文件，其中包含需要提交的csim.c和trans.c文件。
- 提交前应使用前述test-csim、test-trans测试程序（已在上述make过程中编译生成）对提交的正确性进行验证。
- 将该tar文件重命名为“学号+姓名.tar”后提交。

五、实验报告格式

- 按照实验报告模板所要求的格式与内容提交。
- 实验后 一周内 提交至课代表并打包给授课教师。
- 本次实验成绩按100分计
  - 按时上课，签到5分
  - 按时下课，不早退5分
  - 课堂表现：10分，不按操作规程、非法活动扣分。
  - 实验报告：80分。具体参见实验报告各环节的分值
- 学生提交1个压缩包即可，课代表提交1个包
- 在实验报告中，对你每一任务，按照要求用文字详细描述
- 杜绝抄袭！发现 全 0 分！