计算机系统结构课程实验

实验 1　Cache 替换策略设计与分析

一、实验目的

- 深入理解各种不同的 Cache 替换策略
- 理解学习不同替换策略对程序运行性能的影响
- 动手实现自己的 Cache 替换策略

二、实验要求

- 理解学习 LRU 及其它已经提出的 Cache 替换策略
- 在提供的模拟器上实现自己设计的 Cache 替换策略
- 通过 benchmark 测试比较不同的 Cache 替换策略
- 在实验报告中简要说明不同 Cache 替换策略的核心思想和算法
- 在实验报告中说明自己是怎样对不同的 Cache 替换策略进行测试的
- 在实验报告中分析不同替换策略下，程序的运行时间、Cache 命中率受到的影响

三、提交文件

- 模拟器 CRC/src/LLCsim 目录下的两个文件：replacement_state.cpp 和 replacement_state.h
- 实验报告（含“实验要求”的各项内容及实验数据）

四、实验步骤

实验内容部分来自“Cache Replacement Championship”（https://www.jilp.org/jwac-1/）。

（一）前期准备

参考所提供模拟器的 README 中的“Setting Up the Simulation Infrastructure”部分。

运行模拟器需要较老版本的 OS 内核。目前已知模拟器可以在 Ubuntu 10.04 运行，不能在 Ubuntu 14.04 运行，其它系统对模拟器的支持情况还不清楚。

实验提供两个文件：CRC_VAGRANT.tar.gz 和 Vagrantfile，前者为模拟器，后者为虚拟器配置文件。准备用 Ubuntu 10.04 + Virtualbox + Vagrant 作实验环境的同学可以直接使用提供的 Vagrantfile 启动虚拟机（会自动从网上下载 Ubuntu 10.04 镜像，因此消耗一些流量），Vagrant 的使用参考官方主页（https://www.vagrantup.com/）。虚拟机启动后在 home 目录下解压 CRC_VAGRANT.tar.gz，然后参考 README 配置、编译模拟器即可。

使用其它实验环境的同学只需在配置好实验环境后，解压 CRC_VAGRANT.tar.gz。然后参考 README 配置、编译模拟器即可。

模拟器的运行依赖 zlib。Ubuntu 下 zlib 的安装可参考提供的 Vagrantfile。

（二）Benchmark

参考所提供模拟器的 README 中的“Generating Traces to Simulate”部分。

Trace 是对运行过的测试程序（即 benchmark）的记录，本次实验的模拟器以 trace 而不是真实的程序为输入，来测试 Cache 替换策略在不同 benchmark 下的效果。

任何能够在你的机器上运行的程序都可以用来做 benchmark，README 说明了将可执行程序制作成 trace 的方法。例如，在 64 位系统上制作 ls 程序的 trace，可通过以下方法：

```
pinkit/pin-2.7-31933-gcc.3.4.6-ia32_intel64-linux/pin \
    -t ./bin/CMPsim.gentrace.64 \
    -threads 1 \
    -o traces/ls.out \
    -- /bin/ls
```

本次实验只要求使用单线程程序制作的 trace。由于 Cache 替换策略的运行效果与程序的局部性密切相关，我们后续会提供统一的 trace 用于测试。同学们可以先使用自己生成的 trace 作初步测试。

（三）运行模拟器

参考所提供模拟器的 README 中的“Running the Simulator”部分。

README 说明了读入 trace 运行模拟器的方法。例如，在 64 位系统上使用之前制作的 ls 程序的 trace 运行模拟器，可通过以下方法：

```
../bin/CMPsim.usetrace.64 \
    -threads 1 \
    -t ../traces/ls.out.trace.gz \
    -o ls.stats \
    -cache UL3:1024:64:16 \
    -LLCrepl 0
```

“-cache UL3:1024:64:16”中的“L3”表示实验在第 3 级 Cache 上进行，“U”表示统一存储指令和数据，“1024”表示 Cache 大小为 1024KB，“64”表示 Cache 行大小为 64B，“16”表示 16 路组相连。“-LLCrepl 0”的取值与 Cache 替换算法实现的位置有关，参考 README。**本次实验只要求测试 Cache 替换策略在单线程 benchmark 下的性能。所有实验参数和实验结果均记录在输出文件 ls.stats.gz 中，请在实验报告中记录这些信息。**

（四）Cache 替换策略

1、已经提出的 Cache 替换策略

模拟器实现了 LRU 替换策略；“Cache Replacement Championship”主页的“Workshop Program”部分（https://www.jilp.org/jwac-1/JWAC-1%20Program.htm）介绍了多种基于本次实验的模拟器实现的 Cache 替换策略，同学们也可以搜索其它关于 Cache 替换策略的论文。请同学们选择自己感兴趣的 Cache 替换策略深入了解，为设计实现自己的 Cache 替换策略提供基础。

2、设计实现自己的 Cache 替换策略

设计实现自己的 Cache 替换策略，只需修改模拟器 CRC/src/LLCsim 目录下的 replacement_state.cpp 和 replacement_state.h 两个文件，其它文件请不要改动（参考所提供模拟器的 README 中的“Writing Your Own Replacement Algorithm”部分）。

设计自己的 Cache 替换策略时，鼓励参考前人已经做出的工作。如果参考了别人的工作，请在实验报告中说明。另外，也请在实验报告中突出自己的设计思路，解释为什么自己设计的替换策略是合理的。