计算机组织与体系结构实习 Lab 3.2

2017/11/22

该lab是在lab 3.1的基础上，进行层次化cache管理策略优化的探索。

cache管理策略优化

1. 从网盘目录lab 3.2中下载实习相关文件：https://pan.baidu.com/s/1o8wXJHg 密码: sann

2. 该lab是在lab 3.1的基础上，通过cache策略优化，减少总的存储访问延时。各级cache的默认配置如下：

Core i7-900 Desktop Processor

Level	Capacity	Associativity	Line size(Bytes)	WriteUp Polity	Bus Latency
L1	32 KB	8 ways	64	write Back	0 cpu cycle
L2	256 KB	8 ways	64	write Back	6 cpu cycle

说明：

- **Bus latency** 指的是上层请求传输到本层+本层返回数据给上层过程中数据传输所占时间。对于本层cache的访问时间= Hit Latency + Bus Latency。
- **Hit Latency** 指在本层cache查找数据的时间。不论命中与否，都需要将其加入到本层的访问时间上去。除非采用了bypass策略。
- 每一层Cache的Hit Latency都需要通过运行cacti模型程序获得（向上取整），并请转成时钟周期数。具体见cacti65.tgz。内有README，请仔细阅读后使用。工艺节点固定为0.090（um）工艺，cache type为 “cache”。
- 主存中默认包含了所有的数据，主存访问平均延时设置为 100 个时钟周期， CPU 和主存频率均默认为 2GHz。
- 每层Cache的替换策略默认为 LRU

3. 根据平均访存延迟计算公式：平均访存延迟AMAT(Avarage Memory Access Time）= Hit Latency + Miss Rate * Miss Penalty，对默认配置的Cache进行性能优化。需优化或者添加的管理策略如下：

- 1）使用其他替换算法替代LRU
- 2）添加数据预取策略
- 3）使用cache旁路策略

说明:

- 以上三种优化策略，每种至少应用于其中一级cache
- 需要估算出优化或者添加新的管理策略带来的额外存储开销
- 对于数据预取，预取时机为访存发生miss时。只需要累加获取当前目标数据的访存时间。预取数据的获得可以认为是在cpu忙时自动加载的，不产生额外访问开销。每次预取的数据不应超过4个cache line（包括目标数据本身）。

检查要求

使用附件中给定的 trace 进行cache优化前后的对比测试。具体见trace.tar.gz。

1. 给出策略优化前后cache的平均访存延迟的数据，并比较，说明原因。
2. 提供的trace是测试样例，实际用来测试的样例不公开，推荐大家根据常见访存 pattern 自己构造一些 trace，考虑各种不同的优化。
3. 最后成绩=管理优化策略实现+测试结果排名。各种策略不要过于简单，否则将会影响得分。

说明：

1. cacti65.tgz 为存储访问模型，使用方法为：

```
make
 ./cacti -infile cache.cfg
```

通过在cache.cfg中修改cache的配置，来获得不同配置下的各种参数。*注：如果使用64位系统，一般需要安装gcc-multilib和g++-multilib，才能编译通过。*

2. 2.trace 是 Cache 管理策略优化探索的样例 trace。请注意， 这些样例 trace 并不是最终用于测试的 trace，请不要过度拟合。为了更加直观地看出这些 trace 潜在的规律，本次 trace 中的地址全部采用 16 进制表示，请在读取时注意。
3. 实习报告参见pdf和md模板。

提交要求

每人需单独提交：

1. 实验报告1份。具体内容参照模板。
2. 对应的实验代码。提交内容中还应包含README文档，简要描述Simulator的使用方法；主程序可以以命令行参数的形式指定trace文件，正确地加载并输出统计结果。