PA 3-1 实验报告

221220085 时昌军

一、实验目的

利用程序访问地址的局部性来高效的缓存数据：

1. 利用时间局部性原理，Cache将缓存从主存中读出的数据，这样下次再访问的时候就不需要再次访存，而只需从Cache中读取即可；
2. 利用空间局部性原理，每次Cache缓存数据的时候并不是CPU要多少就缓存多少，而是多读一点。Cache和主存之间交换数据的基本单元在主存中称为块（block），而在Cache中则称为行（line）或槽（slot）。

二、要解决的问题

1. 主存中的块与Cache中的槽如何对应？在这里就要考虑到查找的效率和Cache存储空间使用效率的权衡。于是就产生了直接映射、全相联映射和组相联映射这三种方式；
2. 当新访问的主存块映射到Cache中已经被占用的槽时怎么办？于是便产生了不同的替换策略如先进先出、最近最少用、最不经常用和随机替换算法等；
3. 当Cache槽中的数据和主存对应块的数据产生不一致时怎么办？这种不一致只会由对Cache的写操作引起，于是针对写操作的不同处理方法就形成了全写法和回写法两类方法。
4. 写操作时Cache缺失时怎么处理？根据是否将内存块调入Cache就形成了包括写分配法和非写分配法两种基本的策略。

三、实验过程及要求

在`include/config.h`中定义宏`CACHE_ENABLED`并`make clean`；

在NEMU中实现一个cache，它的性质如下:

1. cache block存储空间的大小为64B
2. cache存储空间的大小为64KB
3. 8-way set associative
4. 标志位只需要valid bit即可
5. 替换算法采用随机方式
6. write through
7. not write allocate

还需要在`nemu/src/memory/memory.c`的`init_mem()`函数中对cache进行初始化，将所有valid bit置为无效即可。实现后，修改`memory.c`中的`paddr_read()`和`paddr_write()`函数，让它们读写cache，当缺失时由cache负责调用`hw_mem_read()`和`hw_mem_write()`读写DRAM。

四、实验心得

pa3-1 的难度适中，主要难度在理解cache 的原理以及随机替换算法的应用和写回的策略。学习计算机内cache 的访存机制，局部性原理；掌握cache 与主存块的映射关系，学习替换方法和写回方法。