FIFO LRU LFU 三种策略的缺页比较

前言

用不同策略 (FIFO LRU LFU) 去模拟内存页面置换算法。我们采用虚拟页面大小为4KB，虚拟内存为4GB,物理内存为1GB，一级页表模拟内存。2^{20} 个page_table 存储每个虚拟页面的信息，根据不同策略，用队列或者链表去维护所有在当前内存的虚拟页面。其中虚拟页面数量page_size=2^{20},物理内存页面数量real_size=2^{18}.

FIFO 介绍

通过维护一个链表结构(记做L) 去存储当前调入的页面。将最先进入的页面维护在链表的最前，最后进入的页面维护在链表的最后。并对所有虚拟页面设置R位，R=0表示最近没读写过，R=1表示最近读写过。当发生内存缺页时：此时如果L表头的页面R位为0，表示这个页面不仅老而且很久没使用，将页面淘汰出内存；如果R为1，将R位设置为0，将页面弹出表头，插入表尾，重复操作，直到找到表头页面R位为0的页面。新加入的页面放入L尾部，并将R为设置为1。

伪代码：

```
FIFO()
初始化虚拟内存页表page_table 初始化链表L
while 内存地址发生器(memory) 不为空
   从 memory 获取页面 p
   对应p的R位设置为1
   if p 不在物理内存中
     从L中获取合适的页面q，淘汰出去，并将p放入L
```

复杂度分析：
非缺页处理： O(1)
缺页处理 一般：O(1) 最坏： O(real_size)

LRU介绍

LRU算法是最近最久未被使用的一种置换算法。也就是说LRU是向前查看。在进行页面置换的时候，查找到当前最近最久未被使用的那个页面，将其剔除在内存中，并将新来的页面加载进来。

伪代码

```
LRU()
初始化虚拟内存页表page_table
初始化链表L
while 内存地址发生器(memory) 不为空
   从 memory 获取页面 p
   if p 在物理内存中
      将页面P从L取出，插入到L表头
   else 从L中淘汰尾部页面q，并将p放入L表头
```

复杂度分析：
非缺页处理： O(1)
缺页处理 :O(1)

LFU介绍

LFU是最近最少未被使用。也就是说当页面满时，需要进行页面置换的时候，所采取的措施是在缓存队列中找到最近使用次数最少的页面，将其剔除出去。将新的页面加载到页面缓存队列中。也就是说在LFU中，需要记录每个页面被访问的次数。

伪代码

```
LFU()
初始化虚拟内存页表page_table
初始化哈希表
哈希表Key为访问次数，Value为所有访问次数为Key的页面组成的链表L
while 内存地址发生器(memory) 不为空
   从 memory 获取页面 p
   if p 在物理内存中
      哈希表弹出 p
      将 页面 p的访问次数+1
      页面重新加入哈希表
   else
     从哈希表中淘汰次数最小最久未被访问的页面
     页面p的访问次数+1
     页面p加入哈希表
```

复杂度分析：
非缺页处理： O(1)
缺页处理 :O(?)

模拟结果

模拟程序用 C++ 实现
操作系统： macOS 13 ventura Beta 7

CPU: 2.3 GHz 双核Intel Core i5

内存：8 GB 2133 MHz LPDDR3

图形卡：Intel Iris Plus Graphics 640 1536 MB

C++编译器:Apple clang version 14.0.0 (clang-1400.0.29.102)

C++标准：C++17

编译选项：-Ofast -march=native -Wall -Wextra

编译环境：终端makefile

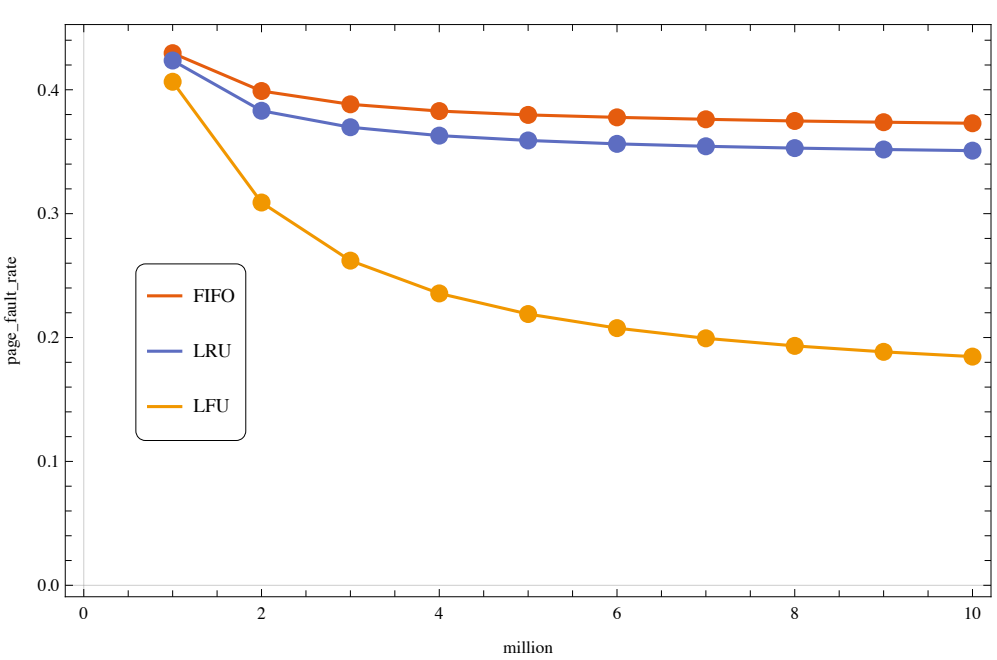
运行结果：

Run_Loops:1000000 FIFO_fault_num:429548 LRU_fault_num:423623 LFU_fault_num:406471
Run_Loops:2000000 FIFO_fault_num:797993 LRU_fault_num:766105 LFU_fault_num:617968

Run_Loops:3000000 FIFO_fault_num:1164772 LRU_fault_num:1109283 LFU_fault-_num:786226
Run_Loops:4000000 FIFO_fault_num:1531227 LRU_fault_num:1451965 LFU_fault-_num:942433
Run_Loops:5000000 FIFO_fault_num:1898487 LRU_fault_num:1795404 LFU_fault-_num:1094847
Run_Loops:6000000 FIFO_fault_num:2266374 LRU_fault_num:2137998 LFU_fault-_num:1245593
Run_Loops:7000000 FIFO_fault_num:2633144 LRU_fault_num:2480662 LFU_fault-_num:1395532
Run_Loops:8000000 FIFO_fault_num:2998492 LRU_fault_num:2823319 LFU_fault-_num:1546087
Run_Loops:9000000 FIFO_fault_num:3364356 LRU_fault_num:3166100 LFU_fault-_num:1696029
Run_Loops:10000000 FIFO_fault_num:3729695 LRU_fault_num:3508487 LFU_fault-_num:1846050
Run time:1565ms

不同策略的表现

Out[ ]=



横坐标代表模拟次数（每百万次），纵坐标代表缺页率（越低越好）

内存地址发生器采用随机数生成，将地址 $[0,2^{20}]$ 分成4个等距区间，每个区间距离刚好为一个物理内存，每区间被选到的概率分别为 (0.05,0.05,0.85,0.05) ,区间内地址均匀分布。

从图中可以看出三种策略的缺页率比较，LFU>LRU>FIFO。