计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(3):156−161 [doi: 10.15888/j.cnki.csa.006254]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 ARC 的闪存数据库缓冲区算法①

梁　鑫[1], 林铭炜[1,2], 姚志强[1,2]

[1](福建师范大学 数学与信息学院, 福州 350108)
[2](福建省公共服务大数据挖掘与应用工程技术研究中心, 福州 350108)
通讯作者: 林铭炜, E-mail: linmwcs@163.com

摘　要: 闪存是一种纯电子设备, 具备体积小、数据读取速度快、能耗低、抗震性强等优点, 被用来部分替代机械硬盘从而提升存储系统的性能. 但是, 现有的缓冲区置换算法都是针对机械硬盘的物理特性进行设计和优化, 因此有必要针对闪存的物理特性重新设计缓冲区置换算法. 提出一种新的面向闪存数据库的缓冲区替换算法 CF-ARC. 算法设计了一种新的页替换机制, 即在替换干净页或者脏页的时候考虑其访问频度的大小, 优先将访问频度少的干净页替换出缓冲区, 使得热页继续留在缓冲区提高命中率, 从而获得更好的性能, 通过对实验结果的对比分析发现 CF-ARC 在多数情况下具有比其它置换算法更高的性能.

关键词: 闪存数据库; 缓冲区置换算法; 替换机制法; 传统机械式硬盘; ARC 算法

引用格式: 梁鑫,林铭炜,姚志强.基于 ARC 的闪存数据库缓冲区算法.计算机系统应用,2018,27(3):156–161. http://www.c-s-a.org.cn/1003-3254/6254.html

Buffer Replacement Algorithm for Flash-Based Databases Based on ARC

LIANG Xin[1], LIN Ming-Wei[1,2], YAO Zhi-Qiang[1,2]

[1](College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350108, China)
[2](Fujian Engineering Research Center of Public Service Big Data Mining and Application, Fuzhou 350108, China)

Abstract: Flash memory is a pure electronic equipment and has the advantages of smaller volume, faster reading speed, lower power consumption and strong vibration resistance, so it is used to partly replace the disk to improve the performance of storage system. But the existing design and optimization of buffer replacement algorithms are based on the physical characteristics of mechanical hard disk. Therefore, it is necessary to redesign a new buffer replacement algorithm which contrapose the physical characteristics of flash memory. This study presents a new buffer replacement algorithm named CF-ARC. A new type of page mechanism replacement is designed, which means the access frequency should be considered when the clean or dirty pages are replaced. The clean pages less visit should replace the buffer to improve the hit rate in hotspot and achieve a better performance. The experimental results show that CF-ARC has better performance than other buffer replacement algorithms in most cases.

Key words: flash databases; buffer replacement algorithm; replacement menchanism; traditional mechanical disk; ARC algorithm

随着技术的高速发展以及成本价格的不断降低, 闪存相对于传统磁盘的优势越来越大, 基于闪存的存储介质被认为具有很大的潜力取代传统的机械式硬盘. 基于闪存的设备由于其体积小、质量轻、能耗低、更高的读取速度等优良特性[1], 它不仅深受 IT 行业所钟爱, 在其它行业中也有着广泛的应用并发挥着巨大的

① 基金项目: 国家自然科学基金 (61502102, 61402109, 61370078)
收稿时间: 2017-06-21; 修改时间: 2017-07-12; 采用时间: 2017-07-17; csa 在线出版时间: 2018-02-09

优势, 如医药业、航天航空、金融市场等.

闪存尽管比磁盘具有更好的 I/O 访问性能, 但仍然比内存的速度低两个数量级. 在闪存与内存之间设置高速缓冲区不仅可以减少代价高昂的磁盘访问开销, 还可以提高数据库的性能. 作为数据库系统的核心组件, 缓冲区利用局部性特性将访问密度高的数据页存储在内存上, 以快速响应中央处理器的读写请求[2]. 当请求访问数据页的时, 如果所请求的页正好在缓冲区中, 则不需要到外部存储中读取该数据页, 由于缓冲区的存取速度比二次存储快得多, 从而可以提高整体的 I/O 性能. 对于那些使用闪存存储作为外部存储的系统, 当缓冲区已满并且需要释放缓存页来释放空间时, 应考虑到闪存读写不对称的特性[3], 同时尽可能减少代价较大的写操作和缓冲区数据页脱靶的次数. 此外, 闪存比磁盘快一个数量级的访问速度, 这种低的访问延迟, 不允许采用复杂的、具有高计算代价的数据结构与算法, 也就是说缓冲区驱逐页的算法应尽可能简单且具备低的时间复杂度和空间复杂度, 避免抵消由闪存低访问延迟带来的 I/O 收益.

1 相关工作

闪存缓冲区替换代价主要为两方面: 1) 是将页面从辅助存储读取到缓冲区的代价; 2) 为将从缓冲区中被驱逐的页写入到辅助存储中的代价. 一种代价的减少往往以另一种代价的增加为前提. 因而, 一个优秀的缓冲区替换算法必须在保持较高命中率的同时降低闪存的写操作和擦除操作次数, 从而获得整体性能的提升.

目前关于闪存缓冲区置换算法的研究主要包括: LRU、CF-LRU、LRU-WSR、CCF-LRU、PB-LRU 和 AD-LRU、F-LRU[4]等算法.

LRU (Least Recently Used) 策略将缓冲区中所有缓存数据页存放在一个链表中, LRU 端保存最近最不常使用的数据页, MRU (Most Recently Used) 端保存最近频繁使用的数据页. 当访问页面 P 时, 首先从链表的 LRU 端开始往 MRU 端查找, 若命中 (链表中找到)P, 则将 P 从当前位置移除并将其移至链表的 MRU 端, 若脱靶 (链表中不存在所需页) 且缓冲区还有额外空间时, 则把 P 页放入 MRU 端, 如过发生脱靶且缓冲区空间已满, 则需要进行页的驱逐替换, 把 LRU 端的页作为首选替换页驱逐出缓冲区, 腾出空间后再将 P 页放在链表的 MRU 端. LRU 算法具有较高的命中率, 但也存在以下缺陷: (1) 没有记录缓冲区页的访问频度; (2) 算法替换的是链表的 LRU 端, 而在实际应用中脏页往往集中在这, 替换这些脏页会带来大量的写和擦除操作, 恶化了闪存数据库的性能.

CFLRU(Clean-First LRU)[5]算法, 不仅考虑到缓冲区的命中率还考虑了驱逐脏页的的替换代价[6], 该算法能有效地减少写和擦除操作的次数, 进而提高闪存数据库的读写性能. 但 CFLRU 替换区的大小 w 并不是一个容易确定的值且算法没有考虑缓冲区页的访问频度, 容易保留较老的脏页而驱逐热干净页从而降低命中率, 另外, 寻找干净页的开销比较大, 有时甚至要遍历整个 LRU 链表.

LRU-WSR 替换策略克服了 CFLRU 策略的缺陷. 使用“冷探测”技术来判断一个页是不是冷页, 增加了选择驱逐页前脏页的频度判断. 在实验中可以看出使用这种策略, 虽然命中率比普通 LRU 算法低, 但能在有效减少写操作和擦除操作的次数的同时不导致命中率的严重恶化. 然而 LRU-WSR 在替换页面时仍然没有考虑到干净页的访问频度, 很有可能被驱逐替换的是热干净页. CCF-LRU 针对以上缺陷进行改进, 将链表分为两个, ML 链表 (Mixed LRU List) 和 CCL 链表 (Cold Clean LRU List). 替换策略优先替换 CCL 链表中的数据页, 当 CCL 链表为空时才按照 LRU-WSR 策略替换 ML 链表中的数据页. CCF-LRU 获取了干净页和脏页的访问频度, 一定程度上优化了存储性能. 但存在一个干净页刚进入缓冲区还未变成热页便被驱逐的情况.

AD-LRU 将闪存缓冲区分为冷区和热区两个区域且区域大小可动态调整, 冷区容量设定下限 min_lc, 当冷区容量大于最小值时, 替换操作发生在冷区, 否则替换操作发生在热区. 当访问脱靶需要驱逐页面时总是优先替换干净页, 若冷区中没有干净页, 则按照 LRU 算法替换脏页, 若热区没有干净页, 则使用二次机会策略替换脏页. 该算法虽获得了更好的性能, 但依然没有彻底解决冷区总是无条件替换干净页最后会导致脏页完全占领缓冲区, 导致一个干净页刚进入到缓冲区便被驱逐的情况, 同时也很难找到一个 min_lc 值, 使它能在于不同数据类型下都获得良好性能.

2 干净页优先的自适应缓冲区置换算法

传统的 ARC 算法[7]在磁头盘片式的机械磁盘上可

以获得不错的性能, 但在闪存数据库系统中会把很多的脏页替换出去, 而替换脏页会带来大量开销大的写操作.

为了解决 ARC 算法运用在闪存数据库上的不足, 本文提出一种新的面向闪存的缓冲区置算法 CF-ARC (Clean First Adaptive Replacement Cache), 给缓冲区的每个数据页定义一个参数 ref, ref 初始为 0, 每当缓冲区的页命中则该页的 ref 加 1, 用来记录缓冲区热页的命中次数, 替换时优先替换 ref 最小的干净页, 若缓冲区不存在干净页再替换 ref 最小的脏页, 采用这种置换策略的优势是: (1) 优先替换干净页, 尽管增加了读操作的次数, 但减少了写操作的次数. (2) 能够有效保证驱逐的页不会是最热页, 从而也保证了缓冲区较高的命中率.

2.1 基本思想

CF-ARC 采用 LRU-WSR 算法类似的“冷探测”技术, CF-ARC 算法将缓冲区划分为冷区 (T1) 和热区 (T2), 冷、热区中都包含干净页和脏页, 缓冲区的数据页都带有冷、脏标识, 当冷标识位为 1 则代表该页为冷页, 否则为热数据页, 当脏标识位为 1 则代表该页为脏页, 否则为干净页, 首次进入缓冲区的页默认为冷页, 当该页再次被访问时则将其冷标识位设为 0, 同时定义两个链表 B1 用来存储替换 T1 时的页号, B2 用来存储替换 T2 时的页号, B1 和 B2 的数据不存入到缓冲区中, 定义参数 P, L1, L2. 且 L1、L2 链表长度满足如下等式:

Len[L1] = Len[T1] + Len[B1]

Len[L2] = Len[T2] + Len[B2]

Len[]函数定义为计算链表的长度算法主要思想如下:

(1) 将缓冲区分为冷区 (T1) 和热区 (T2), 冷区保存仅访问过一次的数据页, 热区保存访问过两次及以上的数据页.

(2) 首次进入缓冲区的 Page 进入冷区的 MRU 端, 当缓冲区的页被命中后则将该命中页移至热区 MRU 端.

(3) 当缓冲区已满且 Page 不在缓冲区中, 如果在 B1 中找到 Page 的记录则更新 P=min{p+X, C}, 其中如果 B1 的长度大于或等于 B2 的长度则 X=1, 否则 X=(B2 的长度)/(B1 的长度), 常数 C 等于缓冲区的大小. 并执行替换策略后, 将 Page 从 B1 中移除并添加到 T2 的 MRU 端, 放入缓冲区.

(4) 当缓冲区已满并且 Page 不在缓冲区中, 如果在 B2 中找到 Page 的记录则更新 P=max{p–X, C}, 其中如果 B2 的长度大于或等于 B1 的长度则 X=1, 否则 X=(B1 的长度)/(B2 的长度), 常数 C 等于缓冲区的大小. 并执行替换策略后, 将 Page 从 B2 中移除并添加到 T2 的 MRU 端, 放入缓冲区.

(5) 当缓冲区已满且在 T1、T2、B1、B2 中都找不到 Page 的记录时, 如果 L1(T1 加 B1 的长度) 等于缓冲区的大小并且如果 T1 的小于缓冲区大小则删除 B1 的 LRU 项, 执行替换策略, 同时 P=min{p+X, C}, 如果 T1 不小于缓冲区的大小则删 T1 的 LRU 端, 从缓冲区删除并将删除的页号添加到 B1 的 MRU 端, 并将页从缓冲区删除. 执行替换策略, 将 Page 添加到 T1 的 MRU 端并放入缓冲区; L1 小于缓冲区的大小并且 L1+L2 大于或等于缓冲区大小, 如果 L1+L2 等于两倍的缓冲区大小则删除 B2 的 LRU 项. 执行替换操作, 将 Page 添加到 T1 的 MRU 端并放入缓冲区;

本算法的替换策略是, 如果满足 (T1>=1)&&(((Page in B2)&&(T1 = P))||(T1>p)) 的条件则优先删除 T1 中的干净页, 然后替换 T1 中的脏页. 否则优先替换 T2 中 ref 最小的干净页, 如果 T2 中没有干净页则从 LRU 端开始寻找 ref=0 的替换页, 若当前页 ref>0 则将其 ref-2 并移至 MRU 端后重新寻找直到找到 ref=0 的冷脏页作为替换页.

2.2 算法设计

CF-ARC 算法细节如算法 1 所示.

算法 1. CF-ARC 算法的伪代码

输入: 请求页 x1, x2, x3,…
输出: p

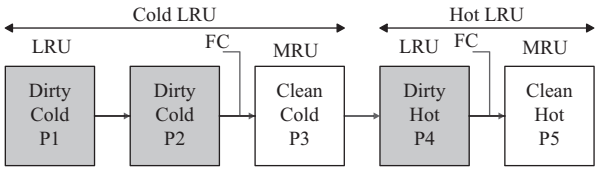
```
1   IF x is in T1 or T2;
2   move x to the MRU position of T2;
3   ELSE IF x is not in T1 or T2 but B1;
4     p = min{p+Y, C};
5     If |B1|>=|B2| then Y=1;
6     else Y=|B2|/|B1|;
8   ELSE IF x is not in T1 or T2 but B2;
9     UPDATE p = max{p-Y, 0};
10    If |B1|>=|B2| then Y=1;
11    else Y=|B2|/|B1|;
12  ELSE IF x is not in T1∪T2∪B1∪B2;
13  Case A: L1 = T1∪B1 = DEBUFFSIZE;
14    If(|T1| < DEBUFFSIZE);
15    DELETE LRU page of B1;
16  Case B: L1 = T1∪b1 < DEBUFFSIZE;
```

```
17    If(|T1|+|T2|+|B1|+|B2|>=DEBUFFSIZE);
18    DELETE LRU page;
19    if(|T1|+|T2|+|B1|+|B2|=2*DEBUFFSIZE) ;
20  RETURN p
```

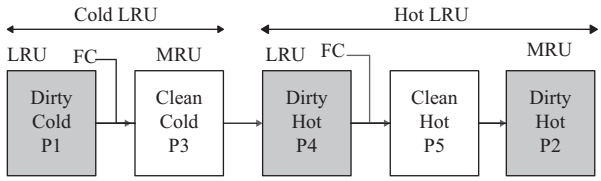
2.3 CF-ARC 算法与 AD-LRU 算法实例分析

通过实验可知: 多数情况下 AD-LRU 算法总是可以获得比 LRU、CF-LRU、CCF-LRU、LRU-WSR 算法更高的性能[8], 下面通过一组实例来论证 CF-LRU 算法相对 AD-LRU 算法的性能优势.

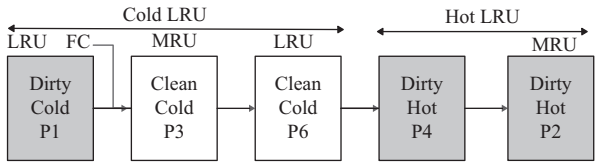
假设闪存缓冲区的大小只能容纳 5 个数据页. 初始情况如图 1(a) 、图 2(a) 所示, 假设 AD-LRU 算法的下界 min_lc 为 3, 指针 FC 用来指向最不常使用干净页. 缓冲区内存在 P1、P2、P3、P4、P5 五个数据页. 当一个对 P2 页的读操作到达时, CF-ARC 算法和 AD-LRU 算法都会发现 P2 存在缓冲区当中, 这时就直接从缓冲区中读取数据页 P2, 访问完之后将 P2 移至 Hot(T2)LRU 链表的 MRU 端并把冷标记位设置为热. 如图 1(b) 、图 2(b) 所示.



(a) AD-LRU 初始状态



(b) AD-LRU 读取 P2

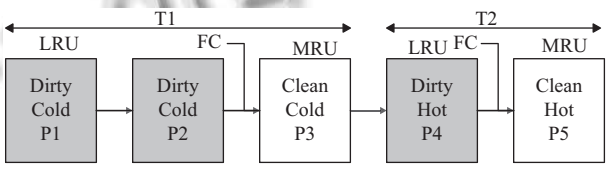


(c) AD-LRU 读取 P6

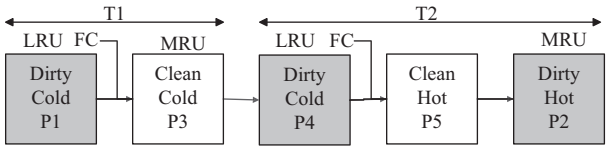
图 1 AD-LRU 链表

然而假设一个针对 P6 的读操作到达, AD-LUR 算法检查发现 P6 不在缓冲区中, 且此时缓冲区已满且冷区容量为 2 小于 min_lc, 按照 AD-LRU 算法可知此时将会选取热区上的 P5 作为首选驱逐页, 最终效果如图 1(c) 所示.

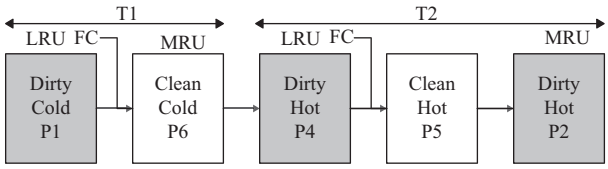
而针对 P6 操作采用 CF-ARC 算法将会出现更优的性能, 根据算法描述替换最后发生在冷区还是热区由多种参数共同决定, 如果替换发生在 T2 则执行跟 AD-LRU 算法一样的操作, 本例中假设通过参数计算后发现替换操作发生在 T1, 即替换 FC 指向的页 P3, 然后把 P6 放入冷区的 MRU 位置, 如图 2(c) 所示, 此时 CF-ARC 替换的是冷区中的干净页, 而 AD-LRU 替换的是热区的干净页, 因为替换热区的数据页 P5 所带来的开销大于替换冷区中的数据页 P3, 且将热页驱逐出缓冲区会降低命中率, 因此在这种情况下 CF-ARC 算法具有更优的性能.



(a) CF-ARC初始状态



(b) CF-ARC读取P2



(c) CF-ARC读取P6

图 2 CF-ARC 链表

通过实例分析, CF-ARC 算法的性能至少与 AD-LRU 算法保持一致, 在某些情形下优于 AD-LRU 算法, 下面将通过实验对实验结果进行验证.

3 实验分析

本节首先介绍实验设计及参数配置 (3.1 节); 进而通过大数据抽样选取 4 种符合 Zipf 分布的数据集来测试算法, 通过对 4 种数据的大量重复实验, 得出实验结果并进一步对比分析, 因每种数据集所得出的结论大致相同, 特选取 T8282 数据集来详细说明 CF-ARC 具有良好的性能优势 (3.2~3.3 节).

3.1 实验设计

实验使用 Flash-DBSim 平台[9]模拟闪存存储系统, 一个用于进行闪存数据库研究的仿真工具, 能够对闪

存技术研究中出现的各种实验环境进行尽可能准确的模拟, 尽可能的减少了接口数量并且很容易就能对整个 Flash-DBSim 的环境进行配置, 使之适应当前的闪存研究实验[4]. 本实验模拟一个 128 G 的闪存固态盘, 配置参数如表 1 所示.

表 1 闪存的参数

参数	数值
块容量	64 pages
页容量	2048 B
写代价	200 μs/page
顺序读代价	25 μs/page
随机读代价	0 μs/page
擦除代价	1500 μs/page
擦除次数	100000

在本次性能测试中, AD-LRU 算法的 min_lc 取值为 0.1 倍的缓冲区大小, CF-LRU 算法的 w 取值为 0.5, FlashDBsim 通过统计闪存的命中率、读次数、写次数来实现算法之间的差异性比较[2].

为了尽量真实的模拟数据库系统运行时的数据访问方式, 我们采用 4 种符合 Zipf 分布的数据集来评判缓冲区替换算法的性能[10], 如表 2. 其中“读写比 x%/y%”表示待测试的数据集当中读操作占 x%, 写操作占 y%, “局部性 x%/y%”表示数据集当中 x% 的操作集中在 y% 的数据中. 通过评判缓冲区的命中率、读操作的次数和写操作的次数来评判算法的整体优越性[11]. 通过多次重复实验并采用单因素方差分析方法进一步分析实验结果.

表 2 测试数据集的详细信息

数据集	请求次数	读写比例	局部性
T8282	300000	80%/20%	80%/20%
T1982	300000	10%/90%	80%/20%
T3773	300000	30%/70%	70%/30%
T7373	300000	70%/30%	70%/30%

3.2 命中率

图 3 展示了各个缓冲区置换算法在不同的缓冲区大小的形况下运行 T8282 数据的命中率情况. 从图中可以看出: 那些采用了“冷判断”机制并且充分考虑了替换干净页和替换脏页代价差异特性的算法, 在多数情况下命中率明显高于其他算法, 而 CF-ARC 算法在大多数的情况下都比 AD-LRU 和 CCF-LRU 命中率高, 这充分说明 CF-ARC 算法是一个具有较高命中率的闪存缓冲区置换算法. 例如, 在缓冲区容量在 1–4 M 的时候, CF-ARC 都能取得最佳的命中率, 就算是在缓冲区容量在 5 M 的时候也与 AD-LRU 性能相当. 通过实验对比发现在另外 3 中测试数据集 T1982, T3773, T7373 中, CF-ARC 依然具有更高的命中率, 因 4 种数据集的测试结论具有较高的一致性, 本文省略另外 3 中测试数据集的详细介绍.

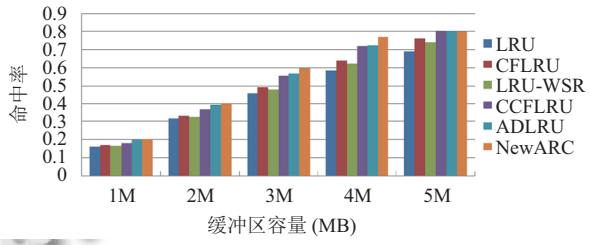


图 3 运行 T8282 数据的命中率

3.3 物理读、写操作次数

图 4、图 5 展示了各个缓冲区替换算法在不同的缓冲区大小下运行 T8282 测试数据集的情况. 从图中可以看出, LRU 算法没有考虑页的访问频度等问题, 在物理读和写方面总是表现为最差. 由于充分考虑了数据页的访问频度以及在优先替换访问频度小的干净页的策略, CF-ARC 算法物理读的次数在多数情况下总是低于其他算法, 在缓冲区大小为 1–4 M 时候, 算法写操作次数也低于其它算法, 甚至在 5 M 的时候, 也仅次于 AD-LRU 算法. 更少的物理写操作一方面降低了开销, 另一方面也延长了闪存的使用寿命.

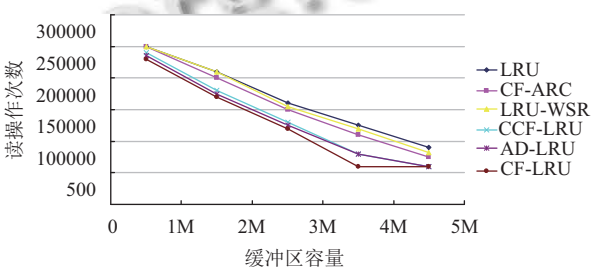


图 4 物理读操作次数

3.4 本章小结

本章首先介绍实验平台的相关信息, 以及对参与实验的不同测试数据的分析与解释. 通过模拟在正常使用中产生的数据页对几种面向闪存缓冲区置换算法进行性能比较, 分别从命中率、读操作次数和写操作次数综合判断 LRU、CFLRU、LRU-WSR、CCF-LRU、AD-LRU 和 CF-ARC 算法的性能, 并在后面给出了具体测试数据, 实验结果充分表明 CF-ARC 算法

具有以下良好的特性: (1) 优先替换最不常使用干净页, 降低替换页的代价; (2) 替换脏页时按照页的访问频度优先替换不常使用页, 能够保证最热页始终在缓冲区内, 保证了命中率; (3) 本算法替换冷区数据页的时候考虑到冷区的大小, 因此不会出现随着时间的推移最终冷区为空的情况.

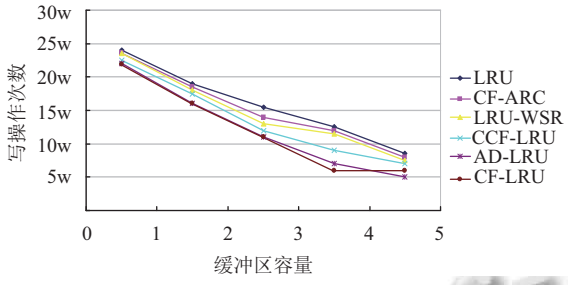


图 5 物理写操作次数

4 结论与展望

随着科技的发展、闪存存储技术的日益完善, 基于闪存数据库的应用也逐渐普及, 现有的缓冲区置换算法大多面向磁盘设计, 没有充分考虑闪存独特的特性, 无法取得较好的性能. 一个高效的缓冲区置换算法不仅能优化 PC 性能, 也能进一步促进社会生产力的提高. CF-ARC 算法充分考虑闪存的写前擦除、异地更新、读写速度不对称的特性, 通过优先替换访问频度小的冷干净页, 不但可以最小化驱逐页的代价, 而且能避免一个干净页刚进入缓冲区便被驱逐的情况, 同时替换访问频度最小的脏页能够有效的保证最热脏页始终保存在缓冲区中保证命中率的同时减少了写操作的次数, 从而提高闪存数据库的性能.

参考文献

1 Chiang ML, Lee PCH, Chang RC. Managing flash memory in personal communication devices. Proceedings of the 1997 IEEE International Symposium on Consumer Electronics. Singapore. 1997. 177–182.

2 王江涛, 赖文豫, 孟小峰. 闪存数据库: 现状、技术与展望. 计算机学报, 2013, 36(8): 1549–1567.

3 Effelsberg W, Haerder T. Principles of database buffer management. ACM Transactions on Database Systems, 1984, 9(4): 560–595. [doi: 10.1145/1994.2022]

4 Lin MW, Yao ZQ, Huang TQ. F-LRU: An efficient buffer replacement algorithm for NAND flash-based databases. Optik-International Journal for Light and Electron Optics, 2016, 127(2): 663–667. [doi: 10.1016/j.ijleo.2015.10.155]

5 Park SY, Jung D, Kang JU, *et al*. CFLRU: A replacement algorithm for flash memory. Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems. Seoul, Korea. 2006. 234–241.

6 Koltsidas I, Viglas SD. Flashing up the storage layer. Proceedings of the VLDB Endowment, 2008, 1(1): 514–525. [doi: 10.14778/1453856]

7 Megiddo N, Modha DS. Outperforming LRU with an adaptive replacement cache algorithm. Computer, 2004, 37(4): 58–65. [doi: 10.1109/MC.2004.1297303]

8 李志. 面向闪存的缓冲区管理算法研究[硕士学位论文]. 合肥: 中国科学技术大学, 2010.

9 Su X, Jin PQ, Xiang XY, *et al*. Flash-DBsim: A simulation tool for evaluating flash-based database algorithms. 2nd IEEE International Conference on Computer Science and Information Technology. Beijing, China. 2009. 185–189.

10 Johnson T, Shasha D. 2Q: A low overhead high performance buffer management replacement algorithm. Proceedings of the 20th International Conference on Very Large Data Bases. San Francisco, CA, USA. 1994. 439–450.

11 Jiang S, Zhang X. Making LRU friendly to weak locality workloads: A novel replacement algorithm to improve buffer cache performance. IEEE Transactions on Computers, 2005, 54(8): 939–952. [doi: 10.1109/TC.2005.130]