网络出版时间：2015-03-11 11:23
网络出版地址：http://www.cnki.net/kcms/detail/42.1658.N.20150311.1123.023.html

DOI：10.13245/j.hust.150323

基于 MWM 的闪存数据库缓冲区置换算法

崔金华[1] 伍卫国[1] 王寅峰[2]
(1 西安交通大学电子与信息工程学院，陕西 西安 710049；
2 深圳信息职业技术学院软件学院，广东 深圳 518172)

摘要 针对现有闪存数据库缓冲区置换算法无法充分发挥闪存存储器性能的现状，提出了一种基于最小权重矩阵(MWM)的高效缓冲区置换算法.该算法基于缓冲区代价置换算法思想，使用 MWM 来组织管理缓冲区的数据块，将数据访问频度映射为权重参数，用来参与缓冲区中数据块的置换.在闪存感知平台 Flash-DBSim 上针对几种典型测试类型数据集进行对比实验，结果表明：本文算法充分利用数据访问特征，特别适用于数据库中数据使用频率高的应用场景.

关键词 闪存固态硬盘；缓冲存储器；置换策略；闪存数据库；最小权重矩阵
中图分类号 O121.8 **文献标志码** A **文章编号** 1671-4512(2015)03-0118-05

Buffer replacement algorithm for flash database systems based on MWM

Cui Jinhua[1] *Wu Weiguo*[1] *Wang Yinfeng*[2]
(1 School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China; 2 Department of Software Engineering, Shenzhen Institute of Information Technology, Shenzhen 518172, Guangdong China)

Abstract A buffer replacement algorithm based on minimum weighted matrix (MWM) was proposed to facilitate the buffer replacement policies to enable flash-based database take advantages of flash materials. The cost-based least recently used (CBLRU) approach was drilled down and a more flash-aware policy was offered, in which MWM managed both the data pages and the data access frequencies to evaluate the weight values. Furthermore the performance improvement was estimated by the standard simulation platform called Flash-DBSim. Experiments on trace data and comparisons with the classical flash buffer replacement algorithms demonstrate that this algorithm is more powerful and more flexible, in particular with respect to the scenario of ultra-high data access frequency in main memory database domain.

Key words flash-based solid-state disks (SSDs); buffer storage; replacement schedule; flash database; minimum weighted matrix

近年来随着闪存存储容量的日渐攀升以及制造工艺的日臻完善，闪存所具有的延迟低、能耗低、移动性强及抗震性高等优势越来越突出.闪存存储器不支持数据的原地更新，且具有读写不对称的特性，因而面向闪存介质的数据库结构须要进行重新设计，适当减少写操作可以从整体上提升闪存数据库系统的性能[1].

闪存数据库缓存置换算法的研究是其中一个重要的研究方向.传统的面向磁盘的缓存置换算法是基于读写延迟相同的情况设计的，例如

收稿日期 2014-05-15.
作者简介 崔金华(1989-)，女，博士研究生，E-mail：cjhnicole@gmail.com.
基金项目 国家自然科学基金资助项目(91330117，61202041)；国家高技术研究发展计划资助项目(2012AA01A306)；深圳基础研究项目(JCYJ20120615101127404)；深圳科技攻关项目(JSGG20140519141854753).

LRU-K[2]，2Q[3]，LRFU[4] 和 ARC[5] 等. CFLRU[6] 是第一个针对闪存特性设计的缓冲区置换算法，当缓冲区数据置换发生时它优先置换干净页，减少对闪存中数据的更新次数，提升系统整体性能. 改进算法 CFLRU/C，CFLRU/E 和 DL-CFLRU/E[7] 能减少对闪存的擦除操作，并提高闪存的磨损均衡程度. 但因为上层的应用程序无法获知闪存内部的闪存块的擦除次数，所以不适用于闪存数据库管理系统. LRU-WSR[8] 对脏页进行冷热区分，而 CCF-LRU[9] 区分干净和脏页的频度特征. AD-LRU[10] 根据冷热区比例来确定置换发生区域，在读写局部性较低的工作负载下也有较好的性能.

CBLRU[11] 则将数据页的所有特征因素转化为权值，在干净页和脏页之间公平选择被置换数据页. 其比基于既定先后置换顺序的算法具有更广泛的适用性. 但是该算法没有将数据访问频度特征纳入考虑. 例如在证券交易所的 level-2 行情高频数据相关的高频数据库系统中，将数据访问频度纳入考虑一定程度上会增加缓冲区的更新负载，但是却能减少高频数据的频繁置换所带来的代价，因而系统整体性能会更优；因此，在此算法基础上提出了一种新型的缓冲区置换算法，其充分利用数据访问特征，特别适用于数据库中数据使用频率高的应用场景.

1　新型的缓冲区置换算法

1.1　最小权重矩阵

提出的缓冲区置换算法基于最小权重矩阵(MWM)实现，MWM 定义为

$$\boldsymbol{W}_{\min}=\begin{bmatrix} p_1 & o_1 & n_1 & d_1 & w_1 \\ p_2 & o_2 & n_2 & d_2 & w_2 \\ \cdots & & & & \\ p_i & o_i & n_i & d_i & w_i \\ \cdots & & & & \\ p_N & o_N & n_N & d_N & w_N \end{bmatrix},$$

式中：第 i 行中的 p_i 表示第 i 行的数据指针，$i=1,2,\cdots,N$；o_i 表示对该数据的处理类型，若只进行过读操作，则 o_i 为读，若进行更新修改数据的操作，则 o_i 为写；n_i 为对数据访问的次数；d_i 为该数据的向后距离[11]；w_i 为该数据的综合权重值，$w_i=(o_i' n_i)/d_i$，其中 o_i' 表示根据 o_i 确定的数据页需要付出的平均 I/O 代价，当 o_i 为读时，o_i' 为从闪存读入一个数据页的平均代价，当 o_i 为写时，o_i' 为将一个数据页写回闪存的平均代价.

定义 1（向后距离）　假定 r 是缓冲区中的数据页，r 的向后距离表示从 r 最后一次被访问到目前为止系统访问数据页的次数.

由于数据块进入缓存后只要不被淘汰，数据块的访问次数将不断增大，因此设置一个过期时间变量，若 n_i/d_i 超过阈值，则 n_i 将减半，防止某段时间的高频数据一定时间后不再被访问的情况下对缓冲区的污染. MWM 中存在关系：$w_1\leqslant w_2\leqslant\cdots\leqslant w_N$.

由此可知：MWM 中权重值从小到大有序排列，MWM 中一行代表了一个数据页信息，并且缓冲区包含 N 个数据页的存储系统的 MWM 是一个 N 行 5 列的排序矩阵.

1.2　MWM 更新

当前请求页到达时，若该数据页已经在 MWM 中，则其后向距离的改变也须全部更新；若首次进入 MWM 中无须进行置换，则由于后向距离的改变，原 MWM 的每个数据页的权重值也将发生更新；若首次进入 MWM 中须要发生置换操作，则选择权重值最小的数据页进行置换，然后对数据页的权重值进行更新，由此可见 MWM 的更新是缓冲区置换算法中常用的操作. 文献[11]中已经提出并理论证明缓冲区中两个数据页信息的前后关系稳定性，因而在此基础上设计了针对闪存缓冲区的权重变化特色的动态 MWM 内容更新算法. 假设 MWM 中有效的最大权重值为 $w_m(1\leqslant m\leqslant N)$. 算法具体过程如下所示.

步骤 1　判断数据页 r 是否在缓冲区中. 若该数据页不在缓冲区命中则执行步骤 2，否则判断数据页干净与脏属性；若数据页依旧为干净页则执行步骤 3，否则执行步骤 4.

步骤 2　设置该数据页 o_r 为读，$d_r=1$，计算 w_r，并更新其他数据页权重值. 在权重值顺序排列的 MWM 中，从数据页 1 到数据页 m 二分查找到数据页 r 的位置.

步骤 3　更新该数据页在 MWM 中对应的 o_r 为读，$d_r=1$，计算 w_r，并更新其他数据页权重值. 在最小权重矩中从数据页 $r+1$ 到数据页 m 二分查找到数据页 r 的新位置.

步骤 4　更新该数据页 o_r 为写，$d_r=1$，计算 w_r，并更新其他数据页权重值. 在 MWM 中从数据页 1 到数据页 m 二分查找到数据页 r 的新位置.

由于矩阵中行向量间无相关性，因此行向量采用并行地同步更新，时间复杂度为 $O(1)$. 二分查找新位置的步骤时间复杂度为 $O(\log(N))$，因

此矩阵更新算法的整体时间复杂度为 $O(\log(N))$.

1.3 本文算法

本文算法将缓冲区的数据页信息有序地组织在一个 MWM 中,其中数据页的权重值综合考虑了时间的访问特性、数据访问频率特性以及数据干净与否的特征. 当 CPU 处理闪存数据库系统上的数据 I/O 操作请求时,若该数据在缓冲区命中或者未命中但缓冲区未满,则更新 MWM 然后返回数据页的位置;当该数据不在缓冲区中且缓冲区已满时,剔除 MWM 第一行的数据页后再更新 MWM,算法具体过程如下所示:

步骤 1 首先判断数据页在缓冲区的情况. 若不在缓冲区中且缓冲区已满,则执行步骤 2,否则执行步骤 3;

步骤 2 移除缓冲区中的 MWM 首行对应的数据页,若该数据页为脏页,则写回存储器中然后执行步骤 3,否则直接执行步骤 3;

步骤 3 更新 MWM.

本文算法须要维护一个数据页对应的 MWM 表,每次更新 MWM 中使用二分查找的思想找到某数据页的位置,其搜索时间复杂度为 $o(\log N)$. 当算法置换策略发生时,移除缓冲区中对应的 MWM 中第一行数据,其时间复杂度为 $o(1)$,因此算法的平均时间复杂度为 $o(\log N)$.

2 实验及分析

2.1 实验设置

实验选取了面向硬盘设计的经典 LRU 算法、基础的面向闪存的 CFLRU 算法、典型的代价置换算法 CBLRU,以及经典的 LRU-WSR,CCF-LRU 和 AD-LRU 来对比本文算法.

实验为了获取精确的实验测试时闪存读写操作的次数,在 Flash-DBSim[12] 平台模拟闪存存储系统上实现 7 种置换算法. Flash-DBSim 可以根据实际需求配置模拟出不同性质的固态盘,从而方便地提供不同的测试环境.

实验中模拟一块 NAND 闪存固态盘,该固态盘所采用的闪存数据页大小为 2 048 B,每个数据块包含 64 个数据页,其中读数据页的代价为每页 25 μs,写数据的代价为每页 200 μs,擦除数据页代价为每块 1.5 ms,并且每个数据块的擦除限制次数为 1×10^5,这些参数都取自对应文献实验中所设置的数值.

各个缓冲区管理算法的性能与算法自身的参数设置密切相关. CFLRU 算法置换区的窗口大小设为缓冲区大小的 75%. 而 AD-LRU 算法的 min_lc 设置为 0.1.

为了尽可能真实地模拟出实际数据库系统运行时的数据页访问模式,采用 CCF-LRU[9] 中的测试方式进行测试. 四种测试数据集($T_1\sim T_4$)分别代表符合齐普夫定律类型数据、读写均匀类型数据、写密集型数据(例如联机事务处理系统 OLTP)和读密集型数据(例如决策支持系统),如表 1 所示,表中:请求次数表示这个测试实例中总共数据请求次数;若某种测试数据中所有请求 x 为读操作,y 为写操作,则读/写比例为 x/y;若某种测试数据中在 y_1 的页上有 x_1 的操作,则集中度为 x_1/y_1. 这里采用命中率和物理写操作次数来评价缓冲区置换算法的性能.

表 1 测试数据集的相应信息

数据集	请求次数/10^5	x/y	x_1/y_1
T_1	3	90/10	80/20
T_2	3	50/50	40/60
T_3	3	20/80	90/10
T_4	3	90/10	50/50

2.2 实验结果及分析

通过在测试平台上运行不同的置换算法,然后统计各个算法的命中率,物理读写操作次数等详细情况. 图 1 所示为本文算法与其他算法物理写操作次数(C)的对比. 实验数据表明:每种算法随着缓冲区大小(S)的增加,物理写操作次数逐渐减低. 从实验数据还可以看出基于闪存的算法所涉及的物理写操作次数少于基于磁盘设计的 LRU 算法,因为 LRU 只考虑了数据的时间局部性特性,没有考虑闪存的读写不对称性,所以导致最多的物理写操作次数,而针对闪存的置换算法将降低物理写操作次数纳入考虑中,虽然这样也许在某些场景下使得命中率比针对磁盘设计的 LRU 算法获得的命中率有所降低,但是物理写操作次数相应地减少,因此整体上闪存介质获得了更好的效果.

CFLRU 是第一个针对闪存特性设计的缓存置换算法,开始考虑要减少对脏页的写回次数,但是其没有考虑干净页的操作频率就直接进行置换,导致在这里物理写操作次数仅次于 LRU. LRU-WSR 对脏数据页进行冷热区分,进一步减低了数据在闪存上的物理写操作次数. CCF-LRU 则更进一步对所有数据进行冷热区分,使得性能比 LRU-WSR 更好. 而 AD-LRU 则对 CCF-LRU 进行改进,通过判定冷区所占大小来选择在冷区

还是热区进行数据页的缓冲置换操作. 另外 CBLRU 也详细地考虑了时间局部性及数据页干净与脏属性，性能与 AD-LRU 基本相当. 其不再无条件优先置换某种数据页，而是通过数据页权值的比较，在各数据页之间进行公平的选择，但是没有考虑数据高频命中的情况，因而在处理具有高访问频率特性的数据时性能比本文算法差. 整体上本文算法在四种类型的测试数据上都获得了比较好的性能，因为其全面考虑了数据本身所具有的最近被访问时间、干净与脏属性对应的被置换代价及数据被访问频率的属性特征. 当然由于考虑的数据特征更全面，本文算法所需要的代价也越大，但是也进一步降低了缓冲置换在闪存上的物理写操作所带来的影响，还是能提升整体系统性能.

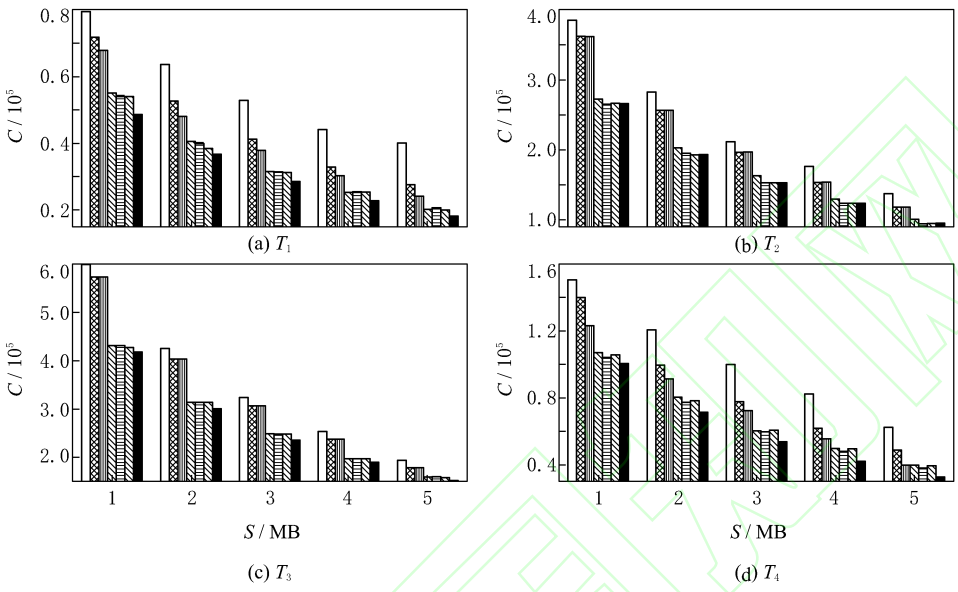

图 1　$T_1 \sim T_4$ 数据集不同算法物理写操作次数对比

此外，由图 1(c)可见：T_3 测试数据集中的局部性特征最高，90％的数据操作集中在 10％数据页上，它的数据具有高频访问的特性. 由测试结果可以看出：相比于其他测试数据集，T_3 测试结果中本文算法比其他算法的数据物理写操作次数降低地更明显，表明数据访问频度高的场景下本文算法更能体现出良好的性能优势.

图 2 所示为本文算法与其他算法命中率(α)的对比. 实验数据表明：每种算法随着缓冲区大小的增加，命中率普遍都获得提升，因为更大的缓冲区可以缓存更多的数据页信息，使得命中率越高. 可以看出与 LRU 相比，其他几种算法在多数场景下都有更好的命中率，这是因为LRU仅简单

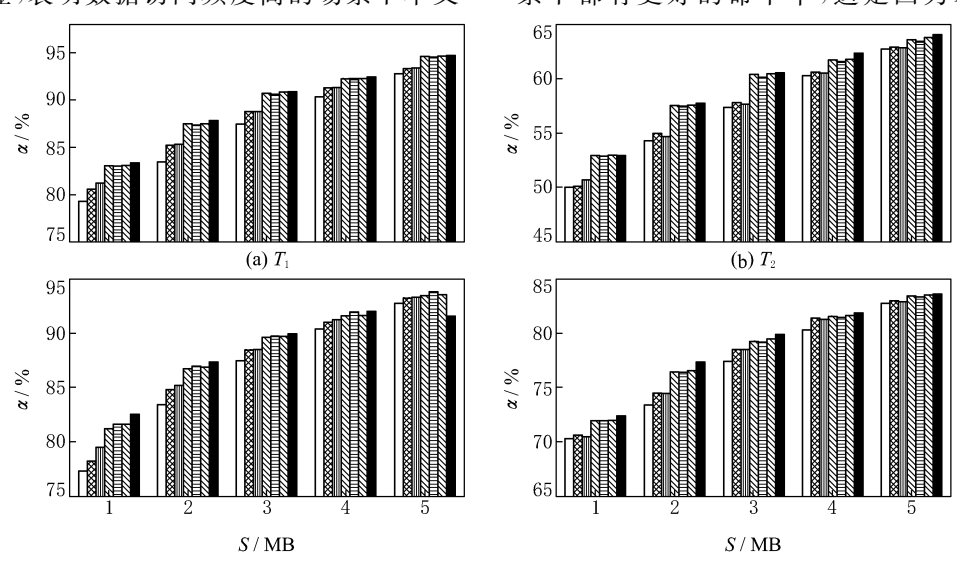

图 2　$T_1 \sim T_4$ 数据集不同算法命中率对比

地考虑了数据最近被访问的时间因素,而 CFLRU 既考虑了最近被访问的时间因素,又同时考虑了干净页置换的代价小于脏页置换代价的影响,获得了七种算法中第二低的命中率. LRU-WSR,CCF-LRU 和 AD-LRU 则更详细地考虑数据页干净与脏属性,使得命中率进一步提升. 而 CBLRU 也考虑时间和干净与脏属性的不同置换代价,当缓冲区中数据页已满,须要剔除部分数据页时,使用权值来对比选取被置换页,使得剔除操作公平地在干净与脏页之间选择. 本文算法则在 CBLRU 基础上更进一步将干净与脏属性的不同命中次数的频率这一因素加入代价值中,挑选最小代价值的数据进行置换操作,因而获得了较好的命中率性能.

3 结论

针对现有基于闪存的数据库缓冲区管理须要考虑闪存读写代价不对称性的巨大差异性的问题,提出一种基于闪存数据库缓冲区置换算法. 其基于 CBLRU 算法,更详细地考虑了数据信息等因素,即同样为每个数据页附加一个权值来综合衡量数据页的置换代价和其驻留内存的影响,使用 MWM 来管理数据页信息,当须要选择置换页时,首先置换具有最小权值的数据页. 实验结果验证了本文算法在闪存固态硬盘上的性能优势,与已有典型缓存置换算法相比更适合高频数据访问的场景.

参 考 文 献

[1] Ahn S, Hyun S, Kim T, et al. A compressed file system manager for flash memory based consumer electronics devices[J]. IEEE Trans on Consumer Electronics, 2013, 59(3): 544-549.

[2] O'neil E J, O'neil P E, Weikum G. The LRU-K page replacement algorithm for database disk buffering [C]// ACM SIGMOD Record. Fort Collins: ACM, 1993: 297-306.

[3] Johnson T, Shasha D. A low overhead high performance buffer management replacement algorithm[C]// Proc of International Conference on Very Large Data Bases. San Mateo: Morgan Kaufmann, 1994: 439-450.

[4] Lee D, Choi J, Kim J, et al. LRFU: a spectrum of policies that subsumes the LRU and LFU policies [J]. IEEE Trans on Computers, 2001, 50(12): 1352-1361.

[5] Megiddo N, Modha D S. ARC: a self-tuning, low overhead replacement cache[C]// Proc of USENIX Conference on File and Storage Technologies. San Francisco: USENIX Association, 2003: 115-130.

[6] Park S, Jung D, Kang J, et al. CFLRU: a replacement algorithm for flash memory[C]// Proc of International Conference on Compilers, Architecture and Synthesis for Embedded Systems. New York: ACM, 2006: 234-241.

[7] Yoo Y S, Lee H, Ryu Y, et al. Page replacement algorithms for NAND flash memory storages[C]// Proc of International Conference on Computational Science and its Applications. Heidelberg: Springer-Verlag, 2007: 201-212.

[8] Jung H, Shim H, Park S, et al. LRU-WSR: integration of LRU and writes sequence reordering for flash memory[J]. IEEE Trans on Consumer Electronics, 2008, 54(3): 1215-1223.

[9] Li Z, Jin P Q, Su X, et al. CCF-LRU: a new buffer replacement algorithm for flash memory[J]. IEEE Trans on Consumer Electronics, 2009, 55(3): 1351-1359.

[10] Jin P Q, Ou Y, Harder T, et al. AD-LRU: an efficient buffer replacement algorithm for flash-based databases[J]. Data & Knowledge Engineering, 2012, 72: 83-102.

[11] 汤显,孟小峰,梁智超,等. 基于代价的闪存数据库缓冲区置换算法[J]. 软件学报, 2011, 22(12): 2951-2964.

[12] Su X, Jin P Q, Xiang X Y, et al. Flash-DBSim: a simulation tool for evaluating flash-based database algorithm[C]// Proc of IEEE International Conference on Computer Science and Information Technology. Piscataway: IEEE, 2009: 185-189.