

一种倒排索引缓存替代算法的研究与实现

时亚南, 束文杰

(新疆维吾尔自治区特种设备检验研究院, 新疆 乌鲁木齐 830011)

摘要:为提高倒排索引的构建速度和检索效率,设计与实现了一套专门的缓存系统。整个缓存系统包含一个用于跟踪每个缓存帧状态的缓存帧描述器 BufDesc 和一张用于将文件及页号映射到缓存池帧号的动态哈希表 BufHashTable。缓存帧描述器记录该缓存页是否被修改过、该缓存页是否可用以及该缓存页是否为有效页等信息,它通过双向链表将所有 BufDesc 类的实例链接在一起。缓存替代策略使用爱憎算法,即采用给帧加 Love/Hate 标记的方式选择被替代出去的页,它是对传统 LRU 和 MRU 算法的改进,能显著提升倒排索引的性能。

关键词:倒排索引;缓存替代策略;爱憎算法;LRU 和 MRU 算法

中图分类号:TP301.6

文献标识码:A

文章编号:1673-629X(2015)05-0060-04

doi:10.3969/j.issn.1673-629X.2015.05.015

Research and Implementation of an Inverted Index Cache Replacement Algorithm

SHI Ya-nan, SHU Wen-jie

(Xinjiang Uygur Autonomous Region Inspection Institute of Special Equipment,
Urumqi 830011, China)

Abstract: In order to improve the construction speed and retrieval efficiency of the inverted index, design and implement a set of specialized caching system. The entire cache system includes a frame buffer descriptor BufDesc which is used to track the status of each cache frame and one for the file and page number that is mapped to a frame number of the dynamic buffer pool hash table BufHashTable. Frame buffer descriptor records whether the cached page has been modified, the cached page is available and whether the information is valid cached pages and so on, it adopts a doubly linked list of all the instances of the BufDesc class together. Love and hate algorithm is used as an alternative strategy that uses to frame plus Love/Hate marker selection that is an alternative way out of the page, which improves the traditional LRU and MRU algorithm, and can significantly improve the performance of the inverted index.

Key words: inverted index; cache replacement strategy; love and hate algorithm; LRU and MRU algorithm

0 引言

倒排索引是搜索引擎的核心技术,因而研究如何加快倒排索引的构建速度、更新速度与检索效率具有十分重要的意义。通常情况下,人们引入缓存技术来解决上述问题。

文献[1-5]主要从缓存算法选择、缓存区大小设置、缓存命中率以及缓存替代策略等几个方面对磁盘文件缓存技术展开相关研究,阐述了缓存技术在改善计算机 I/O 性能方面所起的关键作用。文献[6-20]主要从常用的页面替换算法入手,对缓存替代算法在搜索引擎、文件索引、数据库、普适计算、Linux 操作系

统以及嵌入式系统等领域的应用效果进行综合评估。文中借鉴以上专家学者的研究成果,对常用缓存替代算法 LRU 和 MRU 加以改进,提出了一种类 LRU/MRU 算法,即爱憎算法,并将其应用到倒排索引文件中。经实验测试,该算法表现出较好的性能,可显著提升索引查询性能。

1 缓存管理器设计

数据库和文件都会将数据持久化存储在第二级存储器——磁盘中,而数据本身又必须加载到内存中才能进行各种有用的操作。通常情况下,磁盘容量是内存

容量的几百倍甚至上千倍,这样就会产生一个问题,那就是:无法将磁盘中的所有数据都搬到内存中去执行。为解决内存不足的问题,操作系统引入了虚拟存储器技术,该技术的基本思想是:在内存耗尽时,将部分数据暂时存储在磁盘中,将大容量的磁盘充当内存,必要时再与内存进行数据交换。众所周知,内存的运行速度要比磁盘的运行速度快好几个数量级,程序在内存与磁盘之间的数据交换无疑会带来较多的 I/O 操作。因此,上述方式虽在一定程度上缓解了内存容量紧张的问题,但却无法从根本上解决内存不足时程序运行慢的问题。为减少系统 I/O 操作,操作系统、数据库管理系统、搜索引擎等通常会采用缓冲区管理技术。

1.1 缓存管理器介绍

缓存管理器的基本思想是:首先将内存中的地址空间划分成一片可用的缓存区,然后采用一定的缓存替代算法来决定哪些页面留在内存中,哪些页面需要写入磁盘中,这样当用户执行一个数据请求时,程序会首先检查该数据是否在缓存区。如果在缓存区,那么就将缓存区中的数据返回给用户,这种情况下系统就避免了读磁盘操作,从而大大加快了用户查询速度;如果用户请求的数据不在缓存区,那么就需要通过相应读磁盘操作,将磁盘中的数据读入缓存区,这样当下次用户读取同样的数据时就会避免每次请求数据都执行读磁盘操作,从而也在一定程度上加快了检索速度。从上述描述可以看出,缓存替代算法的命中率越高,程序的执行效率越高,反之,程序执行效率越低。因此,命中率可以在一定程度上去衡量替代算法本身的优劣。一个设计良好的缓存替代算法可以极大地提高缓存命中率,进而减少大量磁盘读写操作,最终实现倒排索引检索性能提高的目的。

1.2 缓存替代算法选择

在操作系统和数据库管理系统中,常见的替代策略有以下几种:FIFO(先进先出置换策略)、LRU(最近最少使用置换策略)、MRU(最近最多使用置换策略)、LFU(最少使用频率置换策略)、MFU(最多使用频率置换策略)、Clock(时钟置换策略)和 RND(随机置换策略)。经查阅大量文献得知,Clock 算法的综合性能优于其他六种算法。

在本项目中,采用给帧加 Love/Hate 标记的算法,该算法能取得与 Clock 算法相近的检索速度,它是对 LRU 算法和 MRU 算法的折衷与改进。

1.3 缓存管理器结构

缓存管理器结构主要由 BufMgr 和 BufDesc 两部分构成,下面分别对两者进行详细介绍。

1.3.1 缓存帧管理器 BufMgr

BufMgr 类是缓存管理器的核心,其数据结构定

义为:

```
public class BufferMgr {
    private int pageNum;
    private Page[] bufPool;
    private BufDesc[] bufTable;//缓存描述器
    private HashMap<PagePtr, Integer> hashMap;    private BufDesc rearUPL;
    private BufDesc frontUPL;
}
```

pageNum 表示页的个数,bufPool 表示缓存池,用于存储内存中所有的缓存帧,bufTable 用于跟踪内存中所有缓存帧的状态,hashMap 用于实现文件名加页号到帧号的映射,rearUPL 表示指向未钉住的帧列表的尾部节点的指针,frontUPL 表示指向未钉住的帧列表的指针前面节点的指针。

BufMgr 类提供了一个桥梁,将内外存联系起来。在创建类实例时,需要提供缓存池的初始大小,也就是缓存池 bufPool 中页的个数 pageNum,它由缓存区大小 BUFFER_SIZE 及缓冲区中帧的大小 PAGE_SIZE 相除即可得到,二者均为固定值。在配置文件由用户指定(大小必须为 512 的整数倍),而 bufTable 会为每个页创建一个描述段,rearUPL 与 frontUPL 将这些描述段用双向链表的方式连接起来。

在缓存管理器创建过程中,将创建一个动态散列表 BufHashTable 用来保存缓存池 bufPool 中正在被使用的或者还没有被释放的 frame 的使用情况。BufHashTable 使用 pageNo 与 frameNo 将 BufPool 中对应 frame 的使用记录下来,记录文件的每一页在缓存池中的使用情况,实现对缓存页的快速查找,从而提高了缓存中页的管理效率。同时创建的 BufDesc 数组将保存空闲 frame 的使用信息,该数组将以双向链表的形式连接起来,双向链表上将记录空闲 frame 的使用情况。

1.3.2 缓存帧描述器 BufDesc

BufDesc 被用来描述这些页的使用情况,记录对应页所属文件,以及在文件中的位置。其数据结构定义如下:

```
public class BufDesc {
    private FileMgr file;//文件
    private int pageNo;//文件中的页号
    private int frameNo;//内存中对应的帧号
    private int pinCnt;//页被钉住的次数
    private boolean dirty;//脏为 true 否则为 false
    private boolean valid;//是否为有效页
    private BufDesc next;
    private BufDesc prior;
}
```

为了能够方便地删除和插入元素,一般不采用顺序存取结构,而采用链表进行存取。该设计采用双向

链表加以实现。其中, next 指向双向链表的后一个节点, prior 指向双向链表的前一个节点。初始状态, 需要对缓存帧进行初始化, 然后在内存中分配缓存状态描述器, 初始化帧编号, 并利用标志节点将可替代缓存

描述节点连接成双向环形链表。先设置正向链接, 然后再设置反向链接, 设置完反向链接, 缓存帧描述器在内存中的状态见图 1。

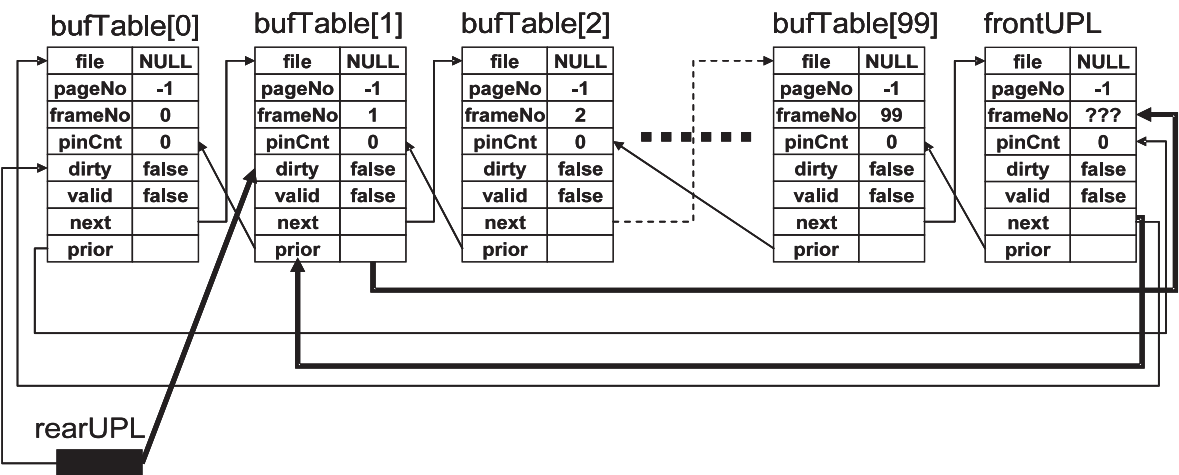


图 1 设置完反向链接

2 缓存替代算法实现

Love/Hate 算法是为每个 frame 的 BufDesc 段设置一个标志位, 用来描述当前 frame 的使用情况。每个 frame 的 BufDesc 将组成一个双向链表。当 bufMgr 首次启动时, BufDesc 的 dirty 段将初始化为 false。置换算法在选择 frame 淘汰时, 将需要检查 valid 位与 dirty 位。在 valid 为 true 的前提下, 如果选择 loveIt 标志, 当 dirty 为 false 时, 对应的 frame 将保存在双向链表的头部, 当 dirty 为 true 时, 对应的 frame 将保存在双向链表的尾部; 而如果选择 hateIt 标志, 情况则恰好相反。

爱憎算法的优点就是简单, 仅仅做一个标记, 该标记是一个布尔型变量, 它与双向链表联系起来。而在缓冲区管理模块中, 其核心是实现分配可用缓存帧, 下面介绍一下爱憎算法分配可用缓存帧部分的详细实现过程:

- ①如果缓冲池中的所有缓存帧都被加了摁钉, 则直接返回-1, 否则, 进行步骤②;
- ②从未钉住帧的双向环形链表取出由 rearUPL 指向的节点, 并从双向环形链表剥离由 rearUPL 指向的节点;
- ③如果剥离下的这一帧曾经被使用过, 则进行步骤④, 否则, 跳转至步骤⑥;
- ④如果剥离下的这一帧曾经被修改过, 则将该页直接写回磁盘, 否则, 执行步骤⑤;
- ⑤移除散列表中原先的文件加页号到对应帧号的映射;
- ⑥重新初始化被替掉的缓存帧描述器, 并返回可用帧号。

从上面可以看出, 其实分配帧的过程就是从环链上剥离一个节点的过程。需要说明的是: 当内存中的帧与磁盘对应的页的内容不一致时, 就会出现所谓的脏页。为保证数据的一致性, 这时必须将数据从内存写回磁盘。

- 该缓冲区管理器主要提供了以下公共接口:
- (1) 分配页 allocatepage (FileMgr file);
 - (2) 读页 readPage (FileMgr file, int pageNo);
 - (3) 释放文件中由 pageNo 指定的页 disposePage (FileMgr file, int pageNo);
 - (4) 对页进行解钉 unPinPage (FileMgr file, int pageNo, boolean dirty, boolean loveIt);
 - (5) 强制写磁盘 flushFile (FileMgr file)。

allocatepage 函数主要实现为指定的文件分配页, 返回分配的页本身。它的实现过程主要有三步: 首先在内存中分配一帧; 然后在磁盘上分配一页; 最后在 hashMap 中插入 fileName + pageNo 到 frameNo 的映射关系。

readPage 函数主要实现根据指定的页号和文件读取指定的页的功能, 返回读取的页本身。其实现主要有两步: 首先看请求的页是否在缓冲区, 如果要读的页在缓冲区, 则直接从缓冲区读取指定的页, 否则从磁盘上根据指定的文件和页号读取指定的页即可。

disposePage 函数主要实现释放文件中由 pageNo 指定的页功能。被释放的页可能在缓存池中, 也可能不在, 如果在缓存池中, 则包含该页内容的帧必须清理。

unPinPage 函数主要实现对页进行解钉操作。其具体实现是: 如果该帧摁钉数变为 0, 则该帧对应的

bufTable 节点必须加到未钉住的缓存池帧的双向链表中,如果 loveIt 为 true,则将其加入到未钉住列表的头部(由 frontUPL 指向的位置),类似于 LRU 算法的效果,否则,将其加入到未钉住列表的尾部(由 rearUPL 指向的位置),它将第一个被分配出去,故类似于 MRU 算法。

flushFile 函数主要实现强制写磁盘操作。当一个文件的所有实例都已经关闭,此时该文件中的所有的页都应该是“未钉住”的页,扫描该文件中的所有的页,对于脏页必须写回磁盘,文件中的所有页,不管脏或不脏,valid 都必须设置为 false。

3 性能测试

文中测试环境为:Centos 6.4 操作系统,2 G 内存,ext3 文件系统。文中在上述实验条件下将爱憎算法与常见缓存替代算法 LRU、Clock、FIFO 及 MRU 进行了比较测试。

结果表明,在写入块数相同的情况下,爱憎算法和 Clock 在各种算法中,读操作耗时最少,FIFO 和 MRU 读操作耗时最高,LRU 读操作耗时处于中间水平。但是,随着文件记录的增大,同一页在较短时间间隔内频繁被读写会导致页面抖动问题,这时,爱憎算法和 Clock 算法的优势会明显减少,与其他几种算法的变化趋势相近。

由此可见,没有任何一种算法会在任何条件下均处于绝对优势地位,爱憎算法的开销相对较小,综合性能较优,可明显提升倒排索引检索性能。

4 结束语

文中介绍了国内专家学者对缓存区的研究现状,提出了一种改进的缓冲区替代算法—爱憎算法,详细介绍了实现该算法的数据结构及定义,并对该算法的核心思想进行了深入阐述,最后对算法进行了性能测试。尽管该算法能显著提升系统文件的 I/O 操作,但仍有一些地方需要进一步改进和优化,比如缓冲区空白页垃圾回收机制尚不完善,页面抖动问题的优化等,在下一步的工作中,笔者会加强这方面的研究。

参考文献:

[1] 朱平,吴碧伟.磁盘缓存管理机制研究[J].计算机工程

与应用,2004,40(20):47-49.

- [2] 刘少伟,王永海,文中领.基于磁盘块保护的优化缓存管理机制[J].计算机研究与发展,2011,48(S1):371-374.
- [3] 郭传鹏,汤光明,孙怡峰.磁盘缓存工作机制研究[J].微计算机信息,2005,22(24):81-83.
- [4] 谢健聪,肖依,褚瑞.一种分布式磁盘缓存的设计与实现[J].微电子学与计算机,2007,24(9):169-170.
- [5] 王强,花嵘,安效国.机群文件系统的缓存机制分析与研究[J].山东科技大学学报:自然科学版,2005,24(3):75-77.
- [6] 杨进才,侯卫红,庞敏.移动环境下自适应缓存管理机制[J].计算机工程与设计,2009,30(2):487-489.
- [7] 魏文国,赵慧民,庄林凯,等.一种基于时钟自适应的改进缓存替换算法[J].中山大学学报:自然科学版,2012,51(6):54-58.
- [8] 杨晓波.倒排文件索引缓存机制的优化[J].计算机系统应用,2012,21(5):96-99.
- [9] 薛煜阳,张太红,张晓明,等.农业搜索引擎倒排索引缓冲机制研究[J].新疆农业大学学报,2011,34(2):161-164.
- [10] 陈婵颖,薛贺,王良家,等.缓存技术在移动数据库中的应用研究[J].计算机工程与设计,2006,27(9):1615-1617.
- [11] 陈慕冰,赵季中,郝旻,等.普适计算中基于上下文信息的缓存管理算法[J].小型微型计算机系统,2007,28(10):1793-1798.
- [12] 姜国松.一种高效、可扩展细粒度缓存管理混合存储研究[J].计算机科学,2013,40(8):79-82.
- [13] 张学亮,左小翠.Linux 页面缓存机制分析及其对磁盘 I/O 性能影响[J].计算机与现代化,2010,26(2):183-187.
- [14] 钟锐,方文楷.嵌入式系统的高速缓存管理[J].电脑知识与技术,2008(12):393-397.
- [15] 刘小珠,孙莎,曾承,等.基于缓存的倒排索引机制研究[C]//第二十四届中国数据库学术会议.海口:中国计算机学会数据库专业委员会,2007.
- [16] 郑榕增,林世平.基于 Lucene 的中文倒排索引技术的研究[J].计算机技术与发展,2010,20(3):80-83.
- [17] 陈玉鹏,陈玮,石晶,等.机械手存储库缓存替换算法研究及应用[J].计算机工程与应用,2003,39(36):5-8.
- [18] 刘瑞虹,曹东启.用自适应机制改进 Web 信息缓存管理的性能[J].计算机研究与发展,2000,37(5):589-594.
- [19] 国志宏,王宏安,王强.实时数据库缓冲区管理算法的设计和实现[J].计算机应用研究,2005,22(2):121-124.
- [20] 汤显,孟小峰.FClock:一种面向 SSD 的自适应缓冲区管理算法[J].计算机学报,2010,33(8):1460-1471.

一种倒排索引缓存替代算法的研究与实现

作者：[时亚南](#)，[束文杰](#)，[SHI Ya-nan](#)，[SHU Wen-jie](#)
作者单位：[新疆维吾尔自治区特种设备检验研究院, 新疆 乌鲁木齐, 830011](#)
刊名：[计算机技术与发展](#)[ISTIC](#)
英文刊名：[Computer Technology and Development](#)
年，卷(期)：2015(5)

引用本文格式：[时亚南](#). [束文杰](#). [SHI Ya-nan](#). [SHU Wen-jie](#) 一种倒排索引缓存替代算法的研究与实现[期刊论文]-[计算机技术与发展](#) 2015(5)