文章编号:1001 - 9081(2010)11 - 2962 - 03

混合结构数据库中基于页迁移的存储分层算法

王跃清[1],黄　烨[1],王翰虎[1,2],陈　梅[1]
(1. 贵州大学 计算机科学与信息学院,贵阳 550025; 2. 贵州星辰科技开发有限公司,贵阳 550001)
(kof_wyq@163.com)

摘　要:为了有效地利用固态盘读速快以及磁盘低存储成本的特点,在磁盘和固态盘共存的混合存储结构模型下,设计并实现了一种基于页迁移思想的存储分层算法(SZA)。不同于 NUMA 的迁移代价计算方法,该算法按照迁移代价选择相应的存储介质,并且对不同工作负载的数据进行迁移。实验结果显示,算法有效地提升了数据库系统的 I/O 性能,同时大幅度地减少了对闪存的擦写次数。

关键词:数据库存储;固态盘;混合存储结构;页迁移
中图分类号: TP311.13　　**文献标志码**:A

Storage zoning algorithm based on page migration for database with hybrid architecture

WANG Yue-qing[1], HUANG Ye[1], WANG Han-hu[1,2], CHEN Mei[1]
(1. *College of Computer Science and Information, Guizhou University, Guiyang Guizhou* 550025, *China*;
2. *Guizhou Xingchen Science and Technology Development Company Limited, Guiyang Guizhou* 550001, *China*)

Abstract: To effectively take advantage of high read speed of Solid-State Disk (SSD)'s and low cost of disk storage, under a mixed structure which is the coexistence of disk and SSD, a storage zoning algorithm based on page migration named SZA was proposed. A different calculating method of migration cost from NUMA was presented. The SZA algorithm was explained to choose corresponding storage medium based on the cost of migration. Migrating activity was processed on different workload of data. The simulation results show that the algorithm improves I/O performance effectively and the erasing times of flash have been reduced significantly.

Key words: database storage; Solid State Disk (SSD); hybrid storage architecture; page migration

0　引言

过去的十几年里,磁盘的容量在不断增加,每吉字节的存储成本也在快速下降,但是它的性能却仍然相对持平。低速的磁盘已经成为大型数据库,尤其是以 I/O 密集型应用为主数据库性能的主要瓶颈。而同时随着固态盘(闪存)的容量增长和价格的不断下降,它已经进入了人们的视野。由于闪存本身具有的"写前擦"、读写速度不对称(读速快,写速与磁盘在一个数量级)以及存储单元擦写次数有限等重要特性,所以在设计闪存数据库管理系统或者与闪存存储管理有关应用程序时必须要考虑到这些因素。

为了迎接闪存物理特性给数据存储与管理带来的挑战,国内外已经进行了大量的研究工作。索引设计方面,文献[1]提出页内日志(In-Page Logging,IPL)对于一个数据页的改变不直接写入闪存,而是写到与之相关联的日志记录中。仿真实验表明,它数量级地提高了闪存随机写的性能。LGeDBMS[2]也是基于 IPL 的一个闪存数据库管理系统。磨损均衡方面,文献[3]提出了冷热数据交换的方法实现 wear leveling,后来的一些磨损均衡算法很多都是在其之上的改进版本。闪存查询优化方面,文献[4]提出一种按列存储的方式来提高连接的速度。这些思想和算法主要根据闪存读写不对称性和擦除次数有限等特点提出。可是由于闪存固有的特性,所以针对存储在闪存介质上的数据而提出的思想以及算法,它们的提高是有限的,并且固态盘在短期内不能完全取代磁盘的事实不可否认。

本文从数据库存储结构角度出发,提出了基于页迁移思想的存储分层算法——SZA。它不仅考虑了数据对存储介质的选择,而且根据数据集工作负载类型的不同进行数据的迁移,从而高效地利用闪存的特性提高了系统的 I/O 性能。

1　混合存储结构

EMC、Samsung 等公司提出了一种金字塔[5]存储模型,将昂贵、高性能但是容量较小的介质放在模型的顶端,用来处理密集的 I/O 需求,相当于整个系统的 Cache,而把较便宜高容量的磁盘放在底端来存放不经常使用和归档的数据。这种存储结构顶端是 SLC(Single Level Cell)芯片组成的 SSD(Solid State Disk),成本高昂,并且由于大量处理 I/O 请求,使得系统可靠性、稳定性维护困难。为了解决上述问题,本文提出了更为常用的 MLC(Multi Level Cell)芯片组成的 SSD 和磁盘共存的混合存储结构模型。

如图 1,磁盘和固态盘在同一存储层次上,混合存储结构模型采用传统的数据库系统缓冲区替换策略(Least Recently Unused, LRU)。存储分层算法应用于系统的存储管理器中,

收稿日期:2010 - 04 - 27;**修回日期**:2010 - 06 - 28。　**基金项目**:贵州省科技计划工业攻关基金资助项目(黔科合 GY 字[2008]3035);贵阳市 2010 年科技攻关项目([2010]筑科工合同字第 28 号);贵州大学 2010 年研究生创新基金资助项目(校研理工[2010033])。

作者简介:王跃清(1984 -),男,内蒙古呼和浩特人,硕士研究生,主要研究方向:数据库、软件工程;　黄烨(1983 -),男,贵州安顺人,硕士研究生,主要研究方向:数据库、软件工程;　王翰虎(1946 -),男,贵州贵阳人,教授,主要研究方向:数据库系统、分布式系统、面向对象方法;陈梅(1964 -),女,贵州贵阳人,副教授,主要研究方向:数据库、软件工程。

算法决定数据存放位置的同时,会将存放在固态盘和磁盘上的数据,根据代价进行相互的迁移操作。决策动态地根据每页的情况做出,主要由数据页的历史访问情况决定,并且独立于其他页。存储管理器跟踪每个数据页存放的位置,从而可以找到相应的数据页,并对其工作负载进行统计。在混合存储结构中,数据页从一种介质迁移到另一个种介质上会产生很大的I/O代价,对数据页工作负载的预测必须尽可能的准确。因为在介质之间迁移操作发生的情况下,不能达到可接受的准确度,意味着 I/O 代价将会急剧上升。

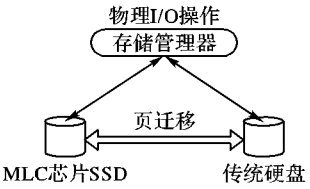

图 1　混合存储结构抽象示意图

2　存储分层算法

2.1　页迁移设计思想

存储分层算法基于页迁移[6]的思想。在分布式内存存取(Non-Uniform Memory Access, NUMA)模式[6]中每个处理器访问本节点内的存储器所需要的时间,比访问某些远程节点内的存储器所花的时间要少得多。NUMA 模式中页迁移的执行触发条件是找出网络中最优的节点去存储一个数据页,从而达到相比其他网络中节点最小化[7]的数据页请求代价。基于页迁移的思想,将数据存储在不同介质上实质上类似于在一个判定树上的数据迁移问题,关键的不同是,这种情况下的页迁移代价由迁移的方向决定而并非距离决定。

图2模拟了数据页状态转换示意图。图中f和m分别代表数据页被存放在固态盘上和磁盘上。在磁盘上读取一个随机页的代价用 r_m 表示,把数据页写到磁盘的另一个随机位置的代价用 w_m 表示;相应的,r_f 和 w_f 分别表示在闪存盘上的代价。数据页从一种存储状态转换为另一种状态的代价等于对这个数据页进行物理写操作的代价。

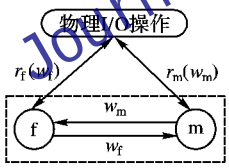

图 2　存储分层算法中数据页状态转换

2.2　基于页迁移的存储分层算法

算法及符号定义:

由于逻辑操作和物理操作之间的影响还不能计算,系统的实际 I/O 代价由物理操作决定,应用层的 I/O 请求只是逻辑的,必须对逻辑 I/O 操作和物理的 I/O 操作作出区分。

算法对每个数据页要维护 4 个计数器。从最后一次迁移开始,l_r 和 l_w 分别用来表示对逻辑读和逻辑写的计数;相应的 p_r 和 p_w 用来表示对物理读和物理写的计数。逻辑操作最终以物理操作实现的概率和缓冲区的大小成比例。令 n 表示一个文件的页数,b 是缓冲区管理器分配的页面数。那么,数据页上的逻辑操作在主存中执行的概率就是 b/n。在数据页上的逻辑操作影响总 I/O 代价的概率和逻辑操作导致物理操作的概率相同,即:n 大小数据页的文件,其逻辑操作影响总 I/O 代价的比例是 $1 - b/n$。

SZA 伪代码如下:

输入　pg。

输出　Null。

```
1) if (pg is a new page) pg.state←m
     pg.lr← 0,pg.lw←0,pg.pr←0,pg.pw←0
   end if                                    //初始化计数器
2) pg.lr← pg.lr + 1
   //在首次物理操作之后开始计数;逻辑读操作后计数
3) pg.pr← pg.pr + 1                          //物理读操作后计数
4) pg.lw← pg.lw + 1                          //逻辑写操作后计数
5) if(pg.dirtybit = 1)且执行弹出时
   pg.pw←pg.pw + 1                           //物理写操作后计数
   end if
6) q←1 - b/n
7) C_f←(pg.lr * q + pg.pr) * r_f + (pg.lw * q + pg.pw)
   * w_f                                     //数据页闪存的计数代价
8) C_m←(pg.lr * q + pg.pr) * r_m + (pg.lw * q + pg.pw)
   * w_m                                     //数据页的磁盘计数代价
9) If (((C_f - C_m) > (w_f + w_m) and pg.state = m) or
     ((C_m - C_f) > (w_f + w_m) and pg.state = f))
   pg.state← other state                     //改变状态位
        //根据数据页在闪存上的计数代价与在磁盘
        //上的计数代价决定是否执行迁移操作
   pg.lr←0,pg.lw←0,pg.pr←0,pg.pw←0
   pg.dirtybit←1
   end if        //清零计数器并写脏页标记准备迁移
```

算法第1)~5)行是新页计数过程。最新创建的数据页写入磁盘,直到执行过一次物理操作才开始计数。这是因为新创建大多数页时,将被逻辑写多次,例如,在 B + 树节点分裂时的情况。对于此页来说,这些逻辑写操作并没有正常地反映工作负载类型,所以并不被记录。第7)~8)行计算两种状态下的逻辑操作和物理操作的总代价。其中,逻辑操作的代价按照 $1 - b/n$ 计算。第9)行判断是否进行迁移操作,若执行迁移,则标记并清零原来的计数器并且将脏页标记写 1。算法尽量从逻辑代价方面减少物理 I/O 次数。确切地说,就是它等到逻辑访问模式已经转换为物理模式才执行。

3　实验与结果分析

3.1　实验环境

本实验使用的计算机配置为:Intel Core 2 P4 Duo CPU 2.83 GHz,2 GB 内存。操作系统是 Windows XP SP3,算法是在 VS2005 平台下使用 C#语言实现。仿真实验配有两个 160 GB 磁盘[8]和一个 25 GB 固态盘[9]。算法程序和操作系统运行在一个磁盘上,另一个磁盘和固态盘用来存储测试数据。

3.2　实验结果分析

仿真实验使用3种比例的查询集,其中数据集中每行数据代表执行计数的数据页,对比3种不同的存储结构下的I/O性能:闪存、磁盘、混合结构。结果显示如图3,M 表示磁盘、M/F 表示混合结构、F 表示闪存。算法的每次运行完一次都要清空系统的缓冲区,一共执行了 12 次。

第一个数据集的工作负载是 50 000 次的读查询集,目标是 B + 树的所有叶子节点。由图 3 数据集执行了 7 次之后,结果显示,M 下性能最差,M/F 下数据集的执行时间(35.1 s)是 F 下(34.4 s)的 1.02 倍,并且随着执行次数的增加,M/F 与 F 的性能比值趋于 1。这是因为部分被访问的数据页在第 4 次执行查询集时迁移到了闪存盘,剩余大量的读密集型页在 7 次查询后迁移到了闪存盘。从读查询集执行时间来看,在使用 SZA 的 M/F 下,I/O 性能与闪存结构系统基本相同(在数据集迁移完成之后)。

第二组数据集是读操作和写操作混合的查询集。数据集

大小不变,读、写比例为 1∶1。从图 4 看出,写工作负载的加入增加了 F 的执行时间,M 执行时间基本保持不变,M/F 的性能仍逼近 F。为了客观全面地验证算法,按照读写、比例为 1∶3的查询集进行实验,因为这种比例在实际的数据库工作负载中很少见到。由图 5 可以看出,随着数据集中写工作负载的比例提高,M/F 与 F 结构下的 I/O 性能都受到了影响,显然 F 结构下 I/O 性能相对于 M/F 下受到的影响更大。

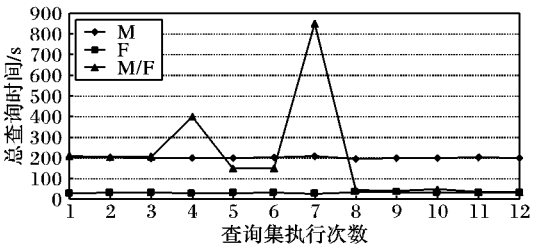

图 3 不同结构下读查询集性能比较

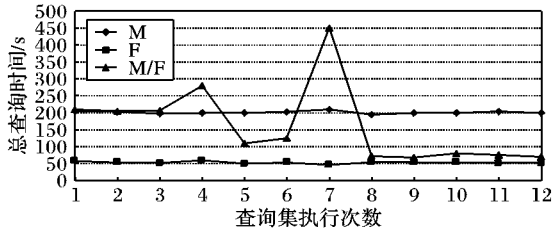

图 4 不同结构下混合查询集性能比较(50% 读,50% 写)

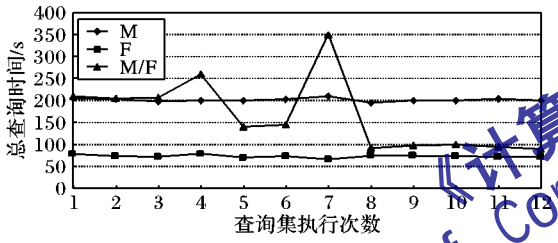

图 5 不同结构下混合查询集性能比较(25% 读,75% 写)

由图 3 ~5 可以看出,M/F 的初始性能和 M 是相同的,因为所有的数据开始都是存放在磁盘 C 的。观察 3 种不同比例的查询集在第 4 和第 7 次的执行时间,可以看到 M/F 均超过了 M 下的执行时间,主要是由于磁盘迁移到闪存所用的时间,由于迁移操作等同于写入闪存的耗费时间,致使执行时间急剧上升。此外,从图 3 可以看出当查询集全部为读查询集时,影响最为明显,因为最终所有的数据除了响应 I/O 请求外,同时全部要写入闪存。此外,3 种查询集下,M/F 中第 7 次查询时间最长,这是由于在缓冲区大小一定的前提下第 4 次已经有部分数据写入了闪存,使得缓冲区效率提高,剩下相对更多的迁移数据在第 7 次执行时发生。表 1 是对 3 种查询集下的总执行时间统计,显然 F 结构下的执行时间最短,结合图 3 ~5 可以发现,此结构下执行时间与工作负载类型比例关系密切,而 M/F 下相对稳定。经过迁移调整过后,如果继续执行查询,M/F 的时间总和接近于 F,接近程度与读工作负载占数据集比例大小成正比,例如第一种数据集下,M/F 结构下的执行时间就越来越接近 F 结构下的时间。随着数据集中写工作负载的增加,不进行迁移的数据增加,迁移所产生的代价就会变小。此外,如果缓冲区的大小足够容纳所有数据的内部节点,迁移操作就会更早完成。

闪存擦除次数统计如图 6 所示。同样使用读写比例为 1∶3的查询集,大小设置为 10 000 次,F 结构下将数据集初始化存储在固态盘上。

由图 6 可以看出,擦除次数和前面 M/F 下 I/O 性能的变化趋势是一致的。F 下对闪存的擦除在 3 500 次左右。M/F 下大量的擦除操作产生在第 6 次数据集的执行过程中,主要是迁移操作引起的,擦除次数是 F 下的 68.21%。随着 SZA 的继续执行,当所有数据页的迁移完成之后,M/F 中的擦除操作基本不再发生。数据结果显示,算法有效地将数据按照工作负载迁移存储,从而显著降低了闪存的擦除次数。

表 1 不同结构下 12 次查询执行总时间统计 s

查询集	M	F	M/F
读查询集	2418	356	2378
1∶1 读写查询集	2408	636	1871
1∶3 读写查询集	2411	876	2086

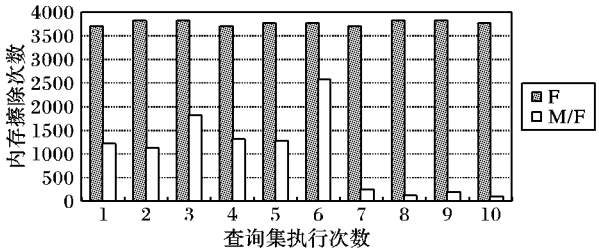

图 6 闪存擦除次数比较(25% 读;75% 写)

4 结语

随着闪存的不断发展,固态盘进入计算机存储体系的步伐变得越来越快。本文提出了基于页迁移的存储分层算法(SZA)。该算法充分利用了闪存和磁盘的特性,提高了数据库系统的总 I/O 性能并显著减少了闪存的擦除次数。算法动态地将读密集型工作负载的数据存储到固态盘上,写密集型工作负载的数据存放到磁盘上,同时可以适应不同工作负载的混合数据集。以后的研究,将侧重于混合结构下缓冲区管理器的页替换策略。

参考文献:

[1] LEE S, MOON B. Design of flash-based DBMS: In-page loggings approach [C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM, 2007: 55 - 66.

[2] KIM G, BAEK S, LEE H, *et al*. LGeDBMS: A small DBMS for embedded system with flash memory [C]// Proceedings of the 32nd International Conference on VLDB. [S. l.]: VLDB Endowment, 2006: 1255 - 1258.

[3] KIM H J, LEE S G. An effective flash memory manager for reliable flash memory space management [J]. IEICE Transactions on Information and System, 2002, E85-D(6): 950 - 964.

[4] SHAH M A, HARIZOPOULOS S, WIENER J L, *et al*. Fast scans and joins using flash drives [C]// Proceedings of 4th Workshop on Data Management on New Hardware. New York: ACM, 2008: 17 - 24.

[5] Celerra FAST technical [EB/OL]. [2009 - 12 - 13]. http://china. emc. com/collateral/hardware/solution-overview/h3479-celerra-fast-so. pdf.

[6] KOLTSIDAS I, VIGLAS S D. Flashing up the storage layer [C]// Proceedings of the 34th Conference on VLDB. New York: ACM, 2008: 514 - 525.

[7] BORODIN A, LINIAL N, SAKS M E. An optimal on-line algorithm for metrical task systems [J]. Journal of the ACM, 1992, 39(4): 746 - 762.

[8] Seagate Barracuda 160GB ST3160318AS [EB/OL]. [2009 - 12 - 08]. http://www. seagate. com/docs/pdf/datasheet/disc/ds_barracuda_7200_12. pdf.

[9] SAMSUNG Serial ATA NSSD 2.5" MCBQE25G5MPQ-0VA03 [EB/OL]. [2009 - 12 - 08]. http://www. samsung. com/global/business/semiconductor/products/flash/ssd/2008/down/ss41x_25_inch. pdf.