一种面向固态硬盘的Cache置换算法

李 博 谢长生 王 芬 赵小刚

(华中科技大学计算机学院 武汉光电国家实验室 武汉 430074)

摘 要 固体硬盘的出现改变了现有存储子系统的框架,也逐渐成为了嵌入式应用的主流存储设备。但是其自身的写机制由于需要先擦后写,成为了影响SSD性能的瓶颈。针对此问题,出于减少写操作次数这一思路提出了LRU-AB算法,力图在基于Cache数据的访问频度上改进SSD置换算法。同时对现有算法进行了分析。

关键词 固体硬盘,flash转换层,缓冲,置换算法

New Cache Replacement Algorithm for Solid-state Drive

LI Bo XIE Chang-sheng WANG Fen ZHAO Xiao-gang

(Wuhan National Laboratory for Optoelectronics, College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract The appearance of solid-state drive(SSD) incurs exciting changes in the architecture of computer storage subsystems and SSD has become the main storage devices for embedded application, gradually. But the bottle neck problem of "erase before write" in NAND-based SSD and its lifetime are the most important issues in today's SSD design. This paper presented the algorithm, LRU-AB(access-based) by considering the access frequency to improve the performance of write operations. Meantime, we also discussed some existing algorithms for this area.

Keywords Solid-state drive, FTL, Cache, Replacement algorithm

近年来,SSD(Solid State Disk)越来越受到业界的广泛重视,它是通过固态存储单元构建的一个数据存储设备[1]。当前大多数SSD都采用NAND flash模式[2],其中有一些也采用SRAM或DRAM而非flash。相较磁盘,SSD有较少的机械故障,同时耗能和发热量也较低。除此之外,大多数SSD产品由于没有机械部件,比硬盘更为安静、更稳固,这一特点使它更适合应用在诸如MP3、智能手机存储和笔记本等领域。由于SSD的本身特性,它的seeking time接近于0。所有这些优点使得SSD能够提供更好的带宽和I/O性能。虽然就目前而言SSD单位存储的价格过高,但是其前景是喜人的。

1 SSD简介

在众多存储技术中,flash受人亲睐的关键一点是它不需要额外的电力来维持存储的数据。表1给出了这样一个比较。

表1 多种存储介质的性能比较

Feature	Disk	DRAM	Low Power SRAM	Flash
Read Access	8.3ms	60ns	85ns	85ns
Write Access	8.3ms	60ns	85ns	4~10ms
Cost/MB	$1.00	$35.00	$120.00	$30.00
DataRetention Current/GB	0A	1A	2mA	0A

在表1中,DRAM,SRAM和flash在读写性能上都优于传统的磁盘。世上没有完美的事物,SRAM功耗低但本身价格昂贵,同时需要额外的电力来维持数据;DRAM有卓越的读写能力,其价格也是可以接受的,但同样需要额外的电力来维持数据。虽然我们可以通过DRAM或SRAM同磁盘复合,把其上的数据最终转移至磁盘,但这种设计必然增加实现的复杂性,减小系统的灵活性[3]。最后可以看出flash在综合诸多因素的考虑下成为我们最好的选择。同时也可以看出,flash存储介质瓶颈也就在于写性能的提高。其写操作性能低下,归咎于flash不能即位(in-place)更新,必须擦除需要的空间来满足写。

SSD的内部,一般包含一个外部接口逻辑(Host interface Logic)、SSD控制器、片内RAM和flash packages阵列。图1是SSD一个大致的框图[4]。

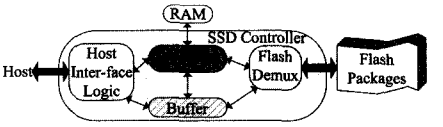


图1 SSD的结构框图

从图中可以知道,外部接口逻辑(Host Interface Logic)主要负责向主机提供可用的连接接口,如USB、光纤接口、PCI Express或SATA等。处于SSD中央部分的处理器(Processor)主要用来执行主机提交的指令,同时在其上可实现多种优化来适应不同的负载需求。同时,SSD加载片内RAM,对实际的存储进行缓冲,这里也是我们提出的算法所应用的场所。Flash阵列是SSD实际存储的物理部件,它通

到稿日期:2010-03-17 返修日期:2010-05-27 本文受国家自然科学基金项目(60933002),863课题(2009AA01A402)和留学生基金委资助。

李 博(1979—),男,博士生,主要研究方向为存储系统、存储系统负载和SSD等,E-mail:libo.ch@gmail.com。

过多路复用设备(Flash Demux)发送给处理器和缓冲管理器(Buffer Manager),SSD 最终的数据也存放于此。处理器、RAM 和多路复用通常构成了 SSD 的控制器。

1.1 课题来源

SSD 在写方面不同于传统存储设备的原因是,其数据的更新不能及时完成,flash 存储介质要求写操作前要进行擦除,但是擦除操作时效远远缓慢于读写操作。另外,NAND 的 flash 擦除是按 block 进行处理的,但 block 不是 SSD 中最小的单位,这就意味着一个擦除操作有可能清除比一个写操作需求还要大的空间[4,5]。为此研究人员定义了一个中间模块——FTL。通过 FTL,应用层用户可以透明地看到整个文件系统,flash 存储单元可以看成是一个 Flash 阵列[6,7]。上层文件系统每次提交一个写请求,FTL 都会把它重定向到一个空的区域。通过这个操作就避免了立即擦除,其代价是必须执行额外的操作和耗费额外的空间。在这一过程中,我们发现 SSD 在进行清除过程(清除包含两个过程数据转移和擦除)中存在不平衡性[2],它对上层提交的数据没有感知性。而我们知道文件通常因为类型不同,其使用的频度是不一样的,例如系统文件、声音和视频文件等。它们通常是读操作过多而写操作过少,反映在 SSD 的块中必定是某些块更新频繁。因此,我们试图合理优化清理过程,尽量通过 Cache 缓冲更新多的块,以减少对 SSD 实质的写入。

1.2 相关研究现状

如何合理地利用片内 Cache,成为问题的关键。因此 Cache 的置换算法成为了大家研究的着力点。从传统磁盘概念上讲,Cache 置换算法主要关注访问的命中率。但是对于 SSD 而言,Cache 置换算法不仅要考虑命中率,而且要尽量减少对 Flash 阵列实际的写操作。为此,文献[8]中 Park Seon-yeong 提出了他的置换算法 Clean-First LRU。该算法考虑了在 Cache 置换过程中读操作和写操作的不均衡性。接着 Jung Hoyoung 在文献[9]中提出了 LRU-WSR(Writes Sequence Reordering)。

1.2.1 Clean-First LRU

CFLRU[8]是针对 Flash 存储器建立的置换法则,充分考虑了 Flash 存储介质的特性。由于写操作通常比读操作花费更多的时间,因此算法规定 Cache 中应该考虑有些需要更新的页面,即带有"脏"位的页面。算法把原有的 LRU 链表分成了两个部分:Working 区和 Clean-First 区,如图 2 所示。

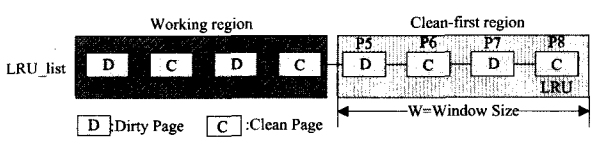


图 2 CFLRU 置换过程

Working 区通常包含最近使用的页面,而 Clean-First 区主要作为置换出的优先选择区。在这种情况下,SSD 控制器会有些置换出 Clean-First 区内的页面。如果该区没有页面供选择,控制器才会考虑 Working 区。例如图中给出的例子,在 CFLRU 作用下其置换的顺序为 P7,P5,P8 和 P6。同时在算法中,作者通过添加 Clean-First 区的窗口大小变量 W,使得算法可动态地进行自我优化。当 W 值过小,意味着过多的"脏"页需要换出,增加了写操作的次数;但是若 W 值过大,反而会降低页的访问命中率。总体而言,通过这一个算法 SSD 可以有效地减少不必要的写操作,提高整体性能。

1.2.2 LRU-WSR

LRU-WSR[9]算法的独特之处在于,它将所有的页面分为两类:访问高的页面——"热"页和访问低的页面——"冷"页。它认为访问低的"脏"页也应该被优先丢弃。算法如图 3 所示。

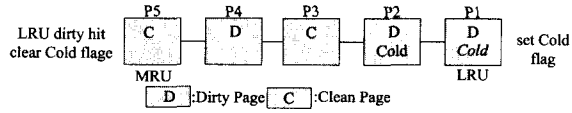


图 3 LRU-WSR 置换过程

作者在页链表的每个页中增加了一个"冷热"标识。当"脏"页 #1 即将被置换出时,控制器会首先查看其"冷热"标识。如果标识被标记,那么该页被置换出;如果该页没有被标记,那么该页应首先标记其"冷热"标识,然后将该页移至链表 MRU 处。一旦在此过程中被标记的"脏"页得到访问,其"冷热"标识置空。

但是在随后的仿真实验中发现,当写的局部性偏低的情况下,LRU-WSR 并不理想。这个问题准备在后面仿真实验结果的分析阶段给予解释。

1.2.3 WOLF

文献[10]最近提出了一个叫做 WOLF 的算法,其核心是通过重序写操作来提高 SSD 对小文件读写的性能。对待小文件算法会在写 Cache 中划分多个缓冲段,将多个小写合并放入相应的缓冲段行程 log 中,如图 4 所示。

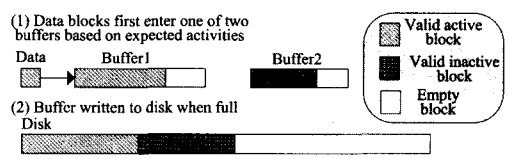


图 4 WOLF 置换过程

通过这一个过程不难想到,算法由小写变大写可以对 SSD 进行批量写入,从而可以有效地解决原来写小擦除大的矛盾。

2 我们的算法——LRU-AB

我们的算法是吸取上述算法的思想提出的、一种基于页面访问频度的置换算法。在算法中,我们对所有的页面分成两大类:"活跃"页面和"非活跃"页面。然后将"活跃"和"非活跃"页面进行合并,分别放置,对 SSD 进行优化。

2.1 "活跃"度的衡量

为了衡量页面本身带有的"活跃"度,需要系统为页面增加一个访问的计数器,由它来记录页面在 Cache 中访问的次数。这里,我们提出了一个界定"活跃"和"非活跃"的指标:如果当前页的访问次数$>\frac{\text{链表中总的访问次数}}{\text{链表中页面的个数}}$,那么就认为此页为"活跃";反之,为"非活跃"。

例如图 5 描述的,LRU 链表中总的访问次数为 11,其平均访问次数为 2.2。由算法的衡量指标,可以得出页 5、页 3 和页 2 是"非活跃"的,而页 4 和页 1 为"活跃"页面。

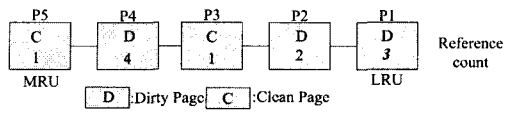


图 5 LRU-AB 算法实例

很显然，这样的操作给原有的算法增加了一定的开销。

2.2 LRU-AB 的二项分布性

通过之前对页“活跃”和“非活跃”的界定，对所有的页面进行分组。我们简单将“活跃”的页面放在一起形成一个 log，将“非活跃”的页面放在一起形成一个 log，然后回写入 SSD。图 6 给出了这一过程。我们发现，对于通常的写机制而言，当放置在 SSD 一段时间之后，其 log 呈现由于数据失效而形成的“孔洞”。而相对于我们的算法写机制，其形成的数据分布具有二项性，即在页面集中 log 内呈现要么“孔洞”过大要么“孔洞”过小。这一现象从结果上方便了我们在 SSD 进行垃圾回收的操作。

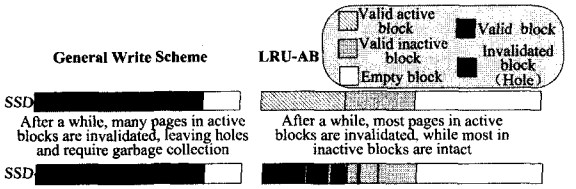


图 6 LRU-AB 写后的二项分布性

3 仿真实验和结果分析

我们的仿真实验在 SSDsim[2] 的基础上进行了修改，它继承了原来 Disksim[11,12] 的主体框架和构件。

3.1 具体实验环境

实验的仿真器由 4 部分构成：Device driver 层、Controller 层、Bus 层和 SSD 层。仿真环境采用 trace 文件作为仿真实验的输入，采用三星 K9XXG08UXM 产品参数构建 SSD 部件。表 2 给出了 4GB 的三星 SSD 设备的具体参数。

表 2 三星 SSD 设备的具体参数

Page Read to Register	25μs
Page Program (Write) from Register	200μs
Block Erase	1.5ms
Serial Access to Register (Data bμs)	100μs
Die Size	2 GB
Block Size	256 kB
Page Size	4 kB
Data Register	4 kB
Planes per die	4
Dies per package (2GB/4GB/8GB)	1,2 or 4
Program/Erase Cycles	100 k

所有的 SSD 事件通过函数 ssd_event_arrive 进入仿真器。上层文件系统通过发送 IO_access_arrive 来设置 SSD 的具体参数。然后 Cache 控制器将区分“活跃”请求和“非活跃”，把它们分别加入不同的队列。当 SSD 总线传输完毕之后，SSD 将响应在队列中请求。如果请求满足，ssd_active_element将按页的大小进行写或读。流程如图 7 所示。

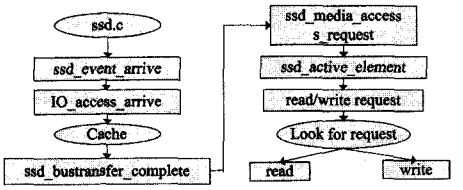


图 7 LRU-AB 算法的 I/O 流程图

3.2 仿真负载模型

在仿真实验室中采用了 3 类负载模型，即 Iozone[12]，Postmark[13] 和 Financial trace。Iozone 和 Postmark 是标准的文件系统测试指标，而 Financial trace 来自于 UMass 的 OLTP 应用。表 3 给出了这 3 类负载模型的具体细节。

表 3 仿真负载说明

Workloads	Average request size	Write (%)	Seq (%)	Avg. req. Inter-arrival time(ms)	Simulated time (s)
Financial	4.38 KB	91%	1.7%	53.50	38769.257
Postmark	0.87 KB	63%	10.2%	8.005	498.398
Iozone	1.4 KB	100%	50.0%	9.417	941.714

3.3 仿真结果与分析

在这个章节我们主要分析了仿真实验的结果和产生的现象。比较了 LRU，CFLRU，LRU-WSR 和 LRU-AB。取得的数据采样是 Cache 大小从 0 到 10MB 的实验结果。这里我们主要参考了两个指标：平均响应时间和清除效率。

3.3.1 清除效率的建立

SSD 在不断地读写过程中需要满足写请求，要不断提高可以使用的页。这就需要 SSD 具有垃圾回收的能力，不断地将用过的页面回收。这就涉及到两类页面：失效页面和没有失效页面。失效页面处理简单，只需擦除原有位置即可。但是对于没有失效的页面，其擦除过程需要将页内数据进行迁移，然后再进行擦除。因此清除过程不仅包含擦除过程，还包含一定的数据迁移过程。由此 LRU-AB 建立了衡量清除效率的公式：

$$100\%-\frac{\text{清除过程中需要迁移的页面大小}}{\text{需要擦除的页面数}\times\text{页面大小}}$$

如果该值过高，那么 LRU-AB 反而会降低 SSD 的整体性能。这是因为区分和分组页面的开销大于算法本身产生的优化。

3.3.2 平均响应时间的比较

图 8 给出了 Iozone 的仿真结果。从它我们不难看出，LRU-AB 比 LRU 能够显著地降低 SSD 的平均响应时间。在绝大多数情况下，LRU-WSR 和 LRU-AB 有相同的表现，但是当 Cache 大小为 2MB 时，LRU-AB 落后于 LRU-WSR。这是因为当 Cache 大小为 2MB 时，Iozone 产生的负载在仿真实验室的清除效率过高，影响了整体性能。

针对 Postmark 负载也有类似的形象，只是这种情况发生在 Cache 大小为 8MB 或 10MB 的情况下，LRU-AB 性能才有所下降。图 9 反映了这个情况。

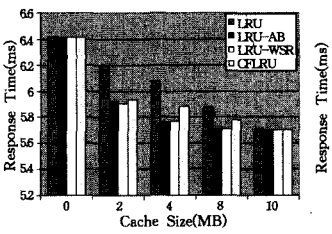


图 8 Iozone 平均响应时间

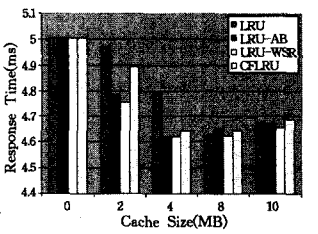
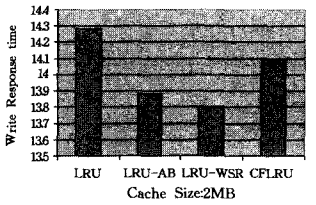


图 9 Postmark 平均响应时间

图 10 和图 11 分别给出了针对 Postmark 负载的写和读平均响应时间，而图 12 描述了 Financial 负载的平均响应时间。从图 10 和图 11 可以清楚地看到 LRU-AB 在读写操作上的差异。显然，LRU-AB 带来的开销主要体现在写这一环节。但从图 12 总体上看 LRU-AB 还是优于其他算法的。



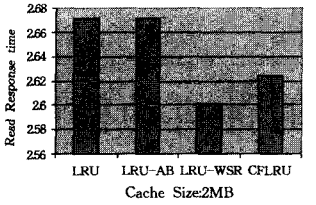


图 10 Postmark 写平均响应时间 图 11 Postmark 读平均响应时间

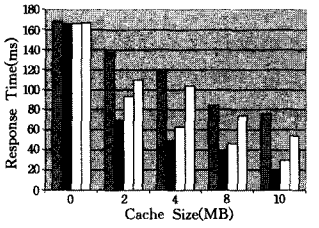


图 12 Financial 平均响应时间

3.3.3 清除效率的比较

图 13 给出了 3 类负载的清除效率。可以看出，对于算法 LRU-AB 而言，效率最低的是 Financial。这也再次验证了图 12 的结果。在 Cache 大小的所有变化情况下，LRU-AB 始终最优。而在 Iozone 和 Postmark 负载下，有技不如人的情况出现。

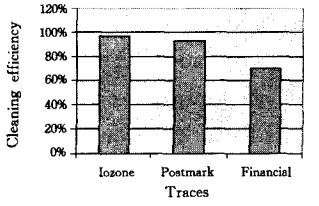


图 13 3 类负载的清除效率

结束语 本文详细介绍了我们提出的 SSD 置换算法 LRU-AB。它通过考虑数据访问的频度，参考 WOLF 算法进行有效的数据聚类，实现了对 SSD 读写性能的优化。通过仿真实验验证了该算法的可行性，同时建立了衡量清除效率的指标。通过对指标的仿真实验，得出结论如果负载在 SSD 中该指标数值偏高，那么 LRU-AB 会在某些情况下性能有所削弱。

参 考 文 献

[1] Wikipedia. Solid-state drive[EB/OL]. http://en. wikipedia. org/wiki/Flash_drive

[2] Agrawal N, Prabhakaran V, Wobber T, et al. Design Tradeoffs for SSD Performance[C]// Proceedings of the USENIX Technical Conference. June 2008

[3] Wu M, Zwaenepoel W. eNVY: A Non-Volatile, Main Memory Storage System[C]// Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems. 1994:86-97

[4] CompactFlash association. Information about CompactFlash [EB/OL]. http://www. ssfdc. or. jp/

[5] Kim Jesung, Kim Jong Min, Noh S H, et al. A Space Efficient Flash Translation Layer for Compact Flash Systems[J]. IEEE Transactions on Consumer Electronics, 2002, 48:366-375

[6] Intel corporation. Understanding the flash translation layer (FTL) specification[EB/OL]. http://developer. intel. com/

[7] MTD. Memory Technology Device (MTD) subsystem for Linux [EB/OL]. http://www. linux-mtd. infradead. org

[8] Park Seon-Yeong, Jung Dawoon, Kang Jeong-Uk, et al. CFLRU: a Replacement Algorithm for Flash Memory[C]// Proceedings of the 2006 International Conference on Compilers, Architecture and Synthesis for Embedded Systems, CASES'06. 2006:234-241

[9] Wang Jun, Hu Yiming. WOLF-A Novel Reordering Write Buffer to Boost the Performance of Log-Structured File[C]// Proceedings of the FAST 2002 Conference on File and Storage Technologies. January 2002

[10] Ganger G, Worthington B, Patt Y. The disksim simulation environment version 4. 0 reference manual[R]. CMU-PDL-08-101, Carnegie Mellon University, May 2008

[11] IOzone. org. IOzone Filesystem Benchmark[EB/OL]. http://www. iozone. org

[13] Katcher J. PostMark: a New File System Benchmark[R]. TR 3022. Network Appliance, October 1997

(上接第 279 页)

[7] Liu C, Wechsler H. Enhanced fisher linear discriminant models for face recognition[C]// Proceedings of Fourteenth International Conference on Pattern Recognition. Brisbane, Qld., Australia, 1998, 2:368-1372

[8] Yambor W S. Analysis of PCA-based and fisher discriminant-based image recognition algorithms[R]. CS-00-103, Computer Science, Colorado State University, 2001

[9] 肖明. 民族形象与解剖结构[M]. China Academic Journal Electronic Publishing Housing, 1994:86-93

[10] Daugman J G. Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression[J]. IEEE Transactions on Acoustics, Speech, and Signal Proces-sing, 1988, 36(7):1169-1179

[11] Dunn D, HigginsW E. Op timal Gabor filters for texture segmentation[J]. IEEE Transactions on Image Processing, 1995, 4(7):947-964

[12] Jain A, Healey G. A multiscale rep resentation including opponent color features for texture recognition[J]. IEEE Transactions on Image Processing, 1998, 7(1):124-128

[13] Teknomo K. What is K Nearest Neighbors Algorithm? [EB/OL]. http://people. revoledu. com/kardi/tutorial/KNN/Contents. ht m

[14] D'Amato C, Malerba D, Esposito F, et al. Extending the K-nearest Neighbour classification algorithm to symbolic objects[C]// Convegno Scientifico Intermedio SIS. Universitàdegli Studi di Napoli "Federico II", 9-11Giugno2003

[15] Friedl M A, Brodley C E. Decision tree classification of land cover from remotely sensed data[J]. Remote Sensing of Environment, 1997, 61(3):399-409

[16] Yang C-C, Prasher S O, Enright P, et al. Application of decision tree technology for image classification using remote sensing data[J]. Agricultural Systems, 2003, 76(3):1101-1117

[17] Han Jiawei, Kamber M. 数据挖掘概念与技术[M]. 范明，孟小峰，等译. 北京：机械工业出版社，2001:185-195

[18] Campadelli P, Lanzarotti R. Automatic Facial Feature Extraction for Face Recognition [M]. Face Recognition, Kresimir Delac and Mislav Grgic, 2007:558

[19] Brunelli R, Poggio T. Feature recognition: features versus template[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993, 15(10):1042-1052

[20] 龚雯，陈丽华，沈建国. 基于几何特征的人脸正面图像特征提取[J]. 现代计算机，2005(9):61-63