一种有效减小最大擦除次数差的损耗均衡设计

王伟能,罗志坤,彭 潇,陈福胜,欧朝龙
(国网湖南省电力公司电力科学研究院,湖南长沙 410007)

摘 要: 损耗均衡影响智能电能表中闪存固态存储系统的寿命和性能.为了提高系统寿命和性能,本文提出了一种有效减小最大擦除次数差的损耗均衡设计.该设计根据位置指针所指物理块的相对擦除次数差来决定损耗均衡是否需要启动和损耗均衡需要的物理块地址.实验结果表明,该设计相比于已有的损耗均衡方法,能有效减小最大擦除次数差,减小擦除次数方差,有效降低损耗不均衡程度.本文还对影响损耗均衡程度的触发阈值进行了分析与讨论,得出了在选择触发阈值时,需要综合考虑损耗均衡情况和时间开销的结论.

关键词: 固态存储系统; 智能电能表; 损耗均衡; 位置指针; 触发阈值
中图分类号: TP333 **文献标识码:** A **文章编号:** 0372-2112 (2015)07-1344-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2015.07.014

A Wear-Leveling Design for Effectively Reducing the Maximum Erase-Cycle Difference Between Blocks

WANG Wei-neng, LUO Zhi-kun, PENG Xiao, CHEN Fu-sheng, OU Chao-long
(*Electric Power Research Institute, State Grid Hunan Electric Power Corporation, Changsha, Hunan* 410007, *China*)

Abstract: The wear-leveling algorithm significantly impacts the lifetime and performance of solid-state storage systems in smart meters. To prolong the lifespan, this paper proposes a wear leveling design which can reduce the maximum erase-cycle difference. According to the erase-cycle difference between blocks pointed by the position pointers, our proposed design actives the wear-leveling algorithm and picks the data block needed to be replaced for wear leveling. Experimental results show that our design reduces the maximum erase-cycle difference and standard deviation, and effectively levels the wear even, which is compared with previous methods. The trigger threshold impacting wear leveling is also discussed, and the conclusion that the balance between wear leveling and time cost need to be considered is obtained in the choice of trigger threshold.

Key words: solid-state storage systems; smart meter; wear leveling; position pointer; trigger threshold

1 引言

闪存存储系统由于具有高数据传输率、低功耗、高可靠性、封装尺寸小、质量轻等优点,被认为是最有潜力的存储系统,并有望替代磁盘存储[1~3].

但是,闪存具有有限的擦除次数:对于 SLC 型闪存,最大擦除次数一般为 100000;而对于 MLC 型闪存,最大擦除次数仅为 10000[4].由于闪存的上述特性,不均匀分布的负载将导致闪存存储系统损耗不均衡,缩短系统使用寿命.

为了提高系统寿命,已有文献提出了许多不同的动态损耗均衡方法和静态损耗均衡方法[5~15].虽然动态损耗均衡方法能有效提高均衡效果,但是由于它侧重于算法的快速性和低时间成本,所以只能均衡热数据块(更新频率高的数据块).为了获得更好的损耗均衡,静态损耗均衡被引入系统中.Ban 和 Hasbaron 提出了一定擦除次数后再随机擦除的算法[5].虽然 Ban 和 Hasbaron 的算法被证明有效,但是它不能很好的区分静态数据和动态数据.Yuan-Hao Chang 和 Jen-Wei Hsieh 为此提出了在一个周期内随机挑选未被擦除块来进行损耗均衡的算法[8].虽然 Yuan-Hao Chang 和 Jen-Wei Hsieh 的算法能提高均衡效果,但是对于大容量闪存存储系统,可能会出现数据块损坏而静态损耗进程未启动的情况.Surafel Teshome 和 Tae-Sun Chung 提出了基于平均值挑选物理块的方法[13].虽然它有效的克服了 Yuan-Hao Chang 和 Jen-Wei Hsieh 算法的不足,但是需要较大的存储空间.

收稿日期:2013-07-08;修回日期:2014-12-17;责任编辑:孙瑶

本文提出了一种基于位置指针挑选数据块的损耗均衡设计.该设计根据位置指针所指物理块擦除次数的差值来决定损耗均衡是否需要启动和挑选的数据块地址.本文还对影响该不均衡程度的触发阈值进行了分析与讨论.

2　系统结构

本文提出的损耗均衡设计所基于的系统硬件架构如图 1 所示.

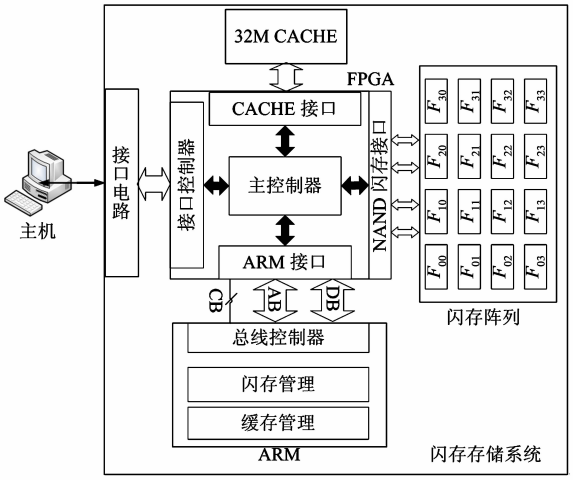

图1　系统硬件架构

系统主要分为五个模块:FPGA 控制模块,接口电路模块,微处理器模块,数据缓存模块和闪存阵列模块.其中,FPGA 控制模块起主控制器作用,为其他模块提供数据传输接口.接口电路模块为闪存存储系统提供外围接口.微处理器模块(采用 ARM)控制多通道闪存阵列及缓存的读写操作.数据缓存模块(采用 Cache)暂存读写数据.闪存阵列模块为系统提供存储介质.

3　一种有效减小最大擦除次数差的损耗均衡设计

3.1　概述

为了提高系统寿命,减小最大擦除次数差值,在增加有限系统开销情况下,本文提出了一种静态损耗均衡算法.图 2 表示所提算法的流程图.为了方便描述,文中把它简称为 SWL(Static Wear-Leveling Leveler).当 SWL 启动时,它或者在更新位置指针信息,或者在替换数据块.本文所提算法能被作为一个带触发条件的进程或者线程添加进固态存储系统中.

本文提出的损耗均衡包含位置指针和两个执行程序:SWL-Run 和 SWL-Update.文中所用符号表示的物理意义如表 1 所示.

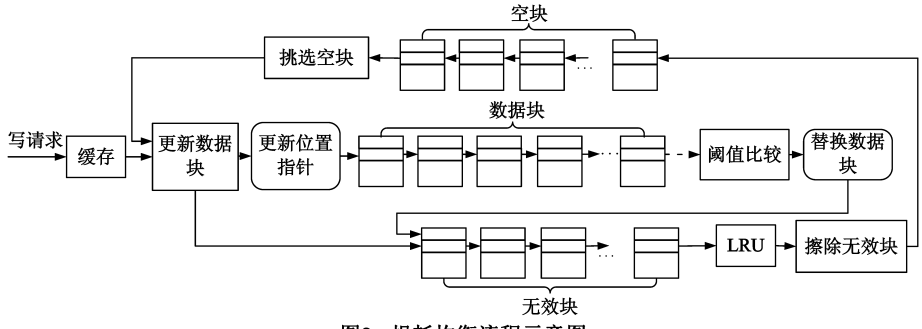

图2　损耗均衡流程示意图

表 1　系统参数符号

符号	定义
TH	触发阈值
p_A	指向块 A 的位置指针
p_B	指向块 B 的位置指针
n_A	块 A 的相对擦除次数
n_B	块 B 的相对擦除次数
n_{ec}	单一数据块擦除次数

3.2　位置指针

位置指针包括 p_A 和 p_B 两个指针,分别指向具有最大擦除次数的数据块(称之为块 A)和具有最小擦除次数的数据块(称之为块 B).当系统刚上电的时候,寻找块 A 和块 B 并初始化位置指针;当系统更新数据块时,同时更新位置指针;当块 A 和块 B 擦除次数差值大于或等于触发阈值 TH 时,即 $n_A - n_B \geqslant TH$ 时,用空块替换指针 p_B 所指数据块.例如,如图 3 所示,假设 $TH = 200$.因为指针 p_A 指向的物理块地址 469(擦除次数 200)与指针 p_B 指向的物理块地址 4(擦除次数 0)的擦除次数差值等于 TH(200),所以静态损耗启动.挑选空块(物理块地址 69,擦除次数 32)并替换指针 p_B 指向的数据块(物理块地址 4,擦除次数 0).最后查找新的块 B 并更新指针 p_B(指向物理块地址 285).

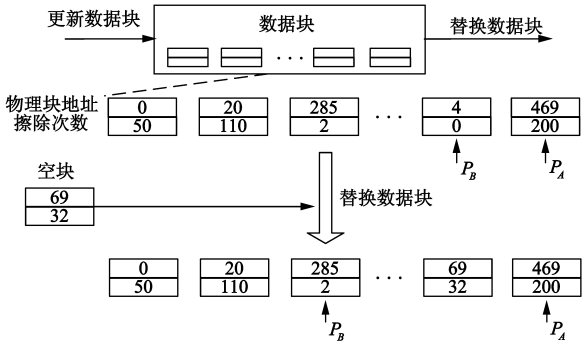

图3 一个替换数据块位置指针示例

3.3 执行程序

本文的损耗均衡包含两个执行程序:SWL-Run 和 SWL-Update(见程序 1 和 2).

程序 1 SWL-Run

```
输入:n_A, n_B, P_A, P_B, TH
输出:n_A, n_B, P_A, P_B
1.  if n_A = n_B then
2.     Reset(void);/* 全部块的擦除次数复位为 0 */
3.  end
4.  if (n_A - n_B) < TH then return;
5.  AddInvalidBlock(P_B);  /* 添加到无效块 */
6.  SearchMin(n_B, P_B)  /* 更新指针 P_B */
7.  PBA = PickEmptyBlock(void);/* 挑选空块 */
8.  if Erase[PBA] > n_A then  /* 擦除次数大于 n_A */
9.     n_A←Erase[PBA];
10.    * P_A←PBA;
11.  end
12.  if Erase[PBA] < n_B then  /* 擦除次数小于 n_B */
13.     n_B←Erase[PBA];
14.     * P_B←PBA;
15.  end
```

程序 2 SWL-Update

```
输入:n_A, n_B, P_A, P_B
输出:n_A, n_B, P_A, P_B
1.  PBA = FlushCache(void);/* 数据块 PBA 需要更新 */
2.  CopyValidData(PBA);
3.  AddInvalidBlock(PBA);/* 添加到无效块 */
4.  if * P_A = PBA then
5.     SearchMax(n_A, P_A)
6.  end
7.  if * P_B = PBA then
8.     SearchMin(n_B, P_B)
9.  end
10.  PBA = PickEmptyBlock(void);/* 挑选空块 */
11.  if Erase[PBA] > n_A then  /* 更新位置指针_A */
12.     n_A←Erase[PBA];
13.     * P_A←PBA;
14.  end
15.  if Erase[PBA] < n_B then
16.     n_B←Erase[PBA];
17.     * P_B←PBA;
18.  end
```

SWL-Run 程序启动的条件是 $n_A - n_B \geqslant TH$.程序 1 表示 SWL-Run 的流程:如果 $n_A = n_B$,则把全部块的擦除次数重置为 0(第 1 步到第 3 步).如果 $n_A - n_B < TH$,则直接返回(第 4 步).否则启动 SWL(第 5 步到第 17 步).先拷贝指针 P_B所指数据块数据并把它添加进无效块(第 5 步),然后查找新的块 B 并把 P_B指向它(第 6 步),最后挑选空块(第 7 步)和更新位置指针(第 8 步到第 15 步).程序 1 中 PBA 表示物理块地址.

SWL-Update 程序启动的条件是数据块需要更新.程序 2 表示 SWL-Update 的流程:先确定需要更新的数据块地址 PBA(第 1 步),然后拷贝数据块中有效数据(第 2 步)和添加块 PBA 到无效块(第 3 步),再更新指针 P_A和 P_B(第 4 步到第 9 步),最后挑选空块(第 10 步)和更新位置指针(第 11 步到第 18 步).

4 实验结果

本节对 Surafel Teshome 和 Tae-Sun Chung 算法和本文提出的算法在所需存储空间大小和损耗均衡两个方面进行了对比.实验是在采用相同的动态损耗均衡算法基础上进行的[14].

实验采用两种测试向量.第一种是从 OLTP 应用中截获的 I/O 命令[16],称之为 OLTP 应用;第二种是从带 Windows XP 操作系统的个人电脑日常使用中收集的 I/O 命令,称之为 PC 日常使用.OLTP 应用和 PC 日常使用包含不同长度的随机写和连续写.表 2 包括了两种测试向量的详细情况.

表 2 测试向量详细情况

写长度(扇区)	1~8	9~64	>64
OLTP 应用	86.58%	12.41%	1.01%
PC 日常使用	56.22%	6.89%	36.89%

实验所采用的系统参数如表 3 所示.页的大小,每块包含的页数目和最大擦除次数都是基于 NAND 闪存规格说明书.

4.1 所需存储空间大小

表 4 表示对于容量 80GB 的闪存固态存储系统,两种算法对应的所需存储空间大小.由表 4 可以看出,本文提出的算法所需存储空间较小.这是因为 Surafel Teshome 和 Tae-Sun Chung 算法需要 16 位数据记录每块的绝对擦除次数,而本文提出的算法只需要 8 位数据记录每块的相对擦

除次数.

表 3　系统仿真参数

参数	值
用户存储容量	80GB
片数	24
每片的块数	16384
每块的页数	64
空间使用率	0.84
每页大小	4kB
缓存容量	32MB
最大擦除次数	10K
触发阈值	32

表 4　所需存储空间大小

算法	大小(KB)
本文提出的算法	384
Surafel Teshome 和 Tae-Sun Chung 算法	768

4.2　损耗均衡

图 4 表示在两种不同测试向量情况下,擦除次数分布情况的对比.相比于 Surafel Teshome 和 Tae-Sun Chung 算法,本文提出的算法具有小的擦除均值和方差,同时能够减小最大擦除次数的差值.

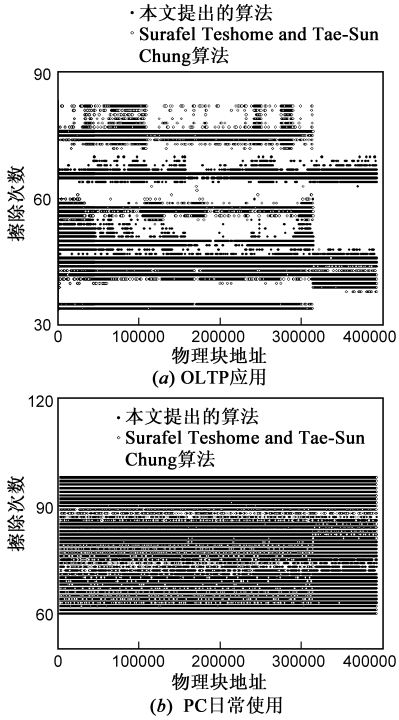

图4　擦除次数分布情况对比

图 4 的统计结果如表 5 所示.从表 5 同样可以得出上述结论.

表 5　统计结果

		本文算法	Surafel Teshome 和 Tae-Sun Chung 算法
OLTP 应用	擦除次数均值	50.11	50.45
	擦除次数方差	8.29	14.77
	最大擦除次数	77	82
	最小擦除次数	44	34
PC 日常使用	擦除次数均值	68.10	74.57
	擦除次数方差	8.05	14.94
	最大擦除次数	94	98
	最小擦除次数	61	60

5　触发阈值对系统性能的影响

触发阈值 *TH* 是本算法中静态损耗均衡启动的判断依据,决定静态损耗的频率,影响损耗均衡效果.

图 5 给出了在两种测试向量情况下,擦除次数均值和方差随 *TH* 的变化情况.从图 5 可以得出,*TH* 值越大,均值则越小,而方差则越大.小的 *TH* 对应大的平均擦除次数和较好的损耗均衡效果.但是越小的 *TH*,其静态损耗启动频率越高,时间成本就越大.

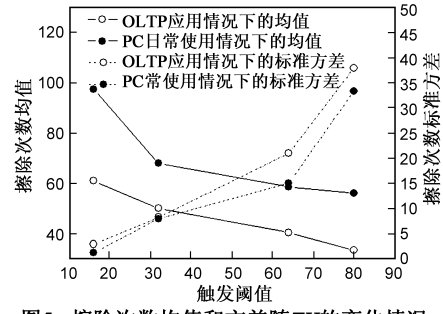

图5　擦除次数均值和方差随*TH*的变化情况

图 6 表示在两种测试向量情况下,归一化运行时间随 *TH* 的变化情况.归一化运行时间是指两种测试向量情况下不同触发阈值运行算法的时间分别除以 *TH* = 32 情况下的运行时间得到.由图 6 得出,触发阈值越小,运行时间越长.

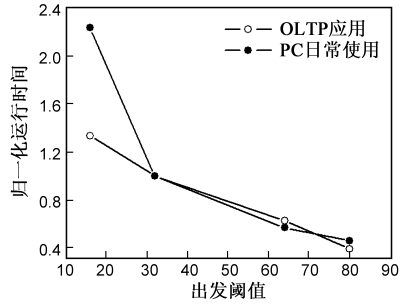

图6　归一化运行时间随*TH*的变化情况

综合从图 5 和图 6 得出的结论,因此触发阈值 *TH* 的选择需要综合考虑损耗均衡的效果和系统运行的时间成本.

6 结论

为了提高系统寿命,减小最大擦除次数差值,在增加有限系统开销情况下,本文提出了一种静态损耗均衡算法.该算法根据位置指针所指物理块的相对擦除次数差来决定损耗均衡是否需要启动和损耗均衡需要的物理块地址.实验结果表明,该设计相比于已有的损耗均衡方法,能有效减小最大擦除次数差,减小擦除次数方差,有效降低损耗不均衡程度,同时减少系统所需内存开销.本文还对影响损耗均衡程度的触发阈值进行了分析与讨论,得出了在选择触发阈值时,需要综合考虑损耗均衡情况和系统运行成本的结论.

参考文献

[1] L P Chang. Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSD[A]. The 13th IEEE/ACM Asia and South Pacific Design Automation Conference[C]. Seoul, Korea: IEEE Press, 2008. 428 – 433.

[2] Chin-Hsien Wu. A flash translation layer for huge-capacity flash memory storage systems[A]. Computer Systems and Applications[C]. Doha, Qatar: IEEE Press, 2008. 100 – 107.

[3] R Bez, E Camerlenghi, A Modelli, A Viscont. Introduction to flash memory[J]. Proceedings of the IEEE, 2003, 91(4): 489 – 502.

[4] Rino Micheloni, Massimiliano Picca, Stefano Amato, et al. Non-volatile memories for removable media[J]. Proceedings of the IEEE, 2009, 97(1): 148 – 160.

[5] A Ban, R Hasbaron. Wear Leveling of Static Areas in Flash Memory[P]. USA: 6732221, 2004 – 05.

[6] L P Chang, T W Kuo. An adaptive striping architecture for flash memory storage systems of embedded systems[A]. Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium[C]. St Louis, Missouri: IEEE Press, 2002. 187 – 196.

[7] Spivak M, Toledo S. Storing a persistent transactional object heap on flash memory[A]. Proceedings of ACM SIGPLAN/SIGBED Conference on Language, Compilers, and Tool Support for Embedded Systems (LCTES'06)[C]. Ottawa: ACM press, 2006. 22 – 33.

[8] Chang Yuanhao, Hsieh Jenwei, Kuo Teiwei. Improving flash wear-leveling by proactively moving static data[J]. IEEE Transactions on Computers, 2010, 59(1): 53 – 65.

[9] Jin X, Jung S, Song YH. Write-aware buffer management policy for performance and durability enhancement in NAND flash memory[J]. IEEE Transactions on Consumer Electronics, 2010, 56(4): 2393 – 2399.

[10] E Gal, S Toledo. Algorithms and data structures for flash memories[J]. ACM Computing Surveys, 2005, 37(2): 138 – 163.

[11] L P Chang, T W Kuo. Efficient management for large-scale flash-memory storage systems with resource conservation[J]. ACM Transactions on Storage, 2005, 1(4): 381 – 418.

[12] Li-Pin Chang, Li-Chun Huang. A low-cost wear-leveling algorithm for block-mapping solid-state disks[A]. ACM Conference on Languages, Compilers, Tools and Theory for Embedded Systems (ACM LCTES)[C]. Chicago, Illinois: ACM press, 2011. 112 – 168.

[13] Surafel Teshome, Tae-Sun Chung. A tri-pool dynamic wear-leveling algorithm for large scale flash memory storage systems[A]. Information Science and Applications (ICISA)[C]. Jeju Island, Korea: IEEE press, 2011. 1 – 6.

[14] Wei-Neng Wang, Kai Ni, Jian-She Ma, et al. An efficient dynamic wear leveling for huge-capacity flash storage systems with cache[J]. Journal of Circuits, Systems, and Computers, 2012, 21(4): 2553 – 2558.

[15] 时正,纪金松,陈香兰,等.一种基于差分进化的 Flash 文件系统垃圾回收算法[J].电子学报,2011,39(2):280 – 284.
Shi Zheng, Ji Jin-song, Chen Xiang-lan, et al. A garbage collection algorithm for flash file system based on differential evolution[J]. Acta Electronica Sinica, 2011, 39(2): 280 – 284. (in Chinese)

[16] Gerhard Weikum. UMass Trace Repository[OL]. USA: IBM, http://traces.cs.umass.edu/index.php/Storage, 2010.

作者简介

王伟能 男,1983 年 9 月出生,湖南益阳人.2012 年博士毕业于清华大学仪器科学与技术系,其后在国网湖南省电力公司电力科学研究院从事电能表、采集终端相关方面的研究工作.
E-mail: wangwn@126.com