基于随机游走的大容量固态硬盘磨损均衡算法

赵 鹏 白 石
(清华大学计算机科学与技术系 北京 100084)

摘 要 基于闪存的大容量固态硬盘(SSD)能够在未来取代磁盘.它有很多优点,包括非易失性、低能耗、抗震性强等.然而,基于 NAND 闪存的存储块自身存在有限的擦除重写次数的问题一直影响着它的广泛应用.当闪存芯片达到擦除重写的限制次数后,存储块上的数据就会变得不可靠.目前研究者们已经提出了一些磨损均衡算法来解决这个问题.但当固态硬盘的存储容量不断增大后,这些算法需要越来越多的内存容量来保证运行.文中提出一种基于随机游走的磨损均衡算法来应用在大容量的固态硬盘上,该算法能够很大程度地减少内存消耗.实验表明所需内存容量仅为 BET 算法的 15.6%,与此同时磨损均衡的性能并没有降低.

关键词 闪存;固态硬盘;磨损均衡;随机游走
中图法分类号 TP302 **DOI 号**: 10.3724/SP.J.1016.2012.00972

A Random-Walk Based Wear-Leveling Algorithm for Large-Capacity SSDs

ZHAO Peng BAI Shi
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract Large-capacity flash based SSD (Solid-state disk) has the potential to become the storage system alternative in the future. It has many advantages: non-volatility, low-power consumption and shock resistance and so on. However, reliability is still a critical problem when using NAND flash memory. Every block of NAND flash memory has a limited number of write/erase cycles, after the limited number, the data that keep in the block become unreliable. Many wear-leveling algorithms have been proposed to solve the reliability problem. But with the capacity of SSD increases, these algorithms always need more and more DRAM capacity. We propose a random-walk based wear-leveling algorithm for large-capacity SSD that can obviously reduce memory consumption (only 15.6% DRAM space compared to BET algorithm) for storing wearing information and achieve the same performance compare with other algorithms.

Keywords flash memory; SSD; wear leveling; random walk

1 引 言

闪存是一种非易失的存储介质,具有体积小、能耗低、抗震性强等特点,被广泛应用于 MP3 播放器、智能手机、掌上电脑、数码相机等电子消费类设备.随着闪存容量的不断增大,基于闪存的固态硬盘随之产生.目前,固态硬盘的接口是模拟通用的硬盘接口,但其内部的存储介质是闪存,主机端提供和磁盘相同的读写接口.它已经广泛应用在笔记本电脑上,如苹果公司的 Mac Air 系列笔记本全部采用了固态硬盘.固态硬盘取代磁盘存储,能够弥补磁盘的高能耗和抗震性能差等缺点.在企业级应用领域固态硬盘也逐步替代磁盘作为主要的存储介质.固态硬盘

收稿日期:2011-08-30;最终修改稿收到日期:2011-12-08.本课题得到国家"八六三"高技术研究发展计划项目基金(2011AA01A203)和国家自然科学基金(61103020)资助.**赵 鹏**,男,1985 年生,博士研究生,中国计算机学会(CCF)学生会员,主要研究方向为新型内存体系结构、操作系统、系统安全.E-mail: p-zhao07@mails.tsinghua.edu.cn. **白 石**,男,1982 年生,博士研究生,主要研究方向为存储系统、操作系统.

正在慢慢成为主流存储设备.

本文提出了一种应用于大容量的固态硬盘的基于随机游走的磨损均衡算法.利用该算法能够很大程度地减少控制器的内存消耗,与此同时又不会降低磨损均衡的性能.文章首先分析闪存及固态硬盘的发展趋势,随后在第 2 节着重介绍闪存存储芯片的内部结构以及存在的磨损均衡问题;第 3 节介绍关于磨损均衡方面的相关工作;在第 4 节中引入随机游走的概念和模型,并将其设计到我们提出的算法中,还介绍存储面所维护的元数据,最后给出了算法的整体流程;第 5 节给出本文算法的一些实验结果并进行比较分析;第 6 节对工作做出总结和展望.

2　固态硬盘的内部结构

近几年来,英特尔、三星等电子制造厂商设计实现了多种基于闪存的固态硬盘产品.厂商通常使用 MLC(多层存储单元)来降低固态硬盘的制造成本.然而 MLC 的单元磨损上限值远远低于 SLC(单层存储单元).

一个闪存芯片由多个存储面(plane)组成,每个存储面由多个存储块(block)组成,而每个存储块又可以分成一定数量的存储页(page).存储块是擦除操作的最小单元,而读和写操作是以存储页为单位的.当一个存储页被写入数据后,它只有在进行了擦除操作后才能再次写入新的数据.这个特点称为写前擦除.因此,利用非本地更新数据的方法是将要更新的数据写到空闲存储区域,从而替代老版本的数据.最新版本的数据是有效的,而老版本的数据是无效的.在闪存中可能同时存在同一逻辑地址数据的多个不同版本.存储有效数据的存储页称为有效页,存储无效数据的存储页称为无效页.虽然 NAND 闪存有不少优点,但是它还有一些缺点,比如,由写前擦除引起的高 I/O 延迟和由擦除重写次数限制带来的不可靠性.

当达到擦除重写的限制次数后,数据就会变得不可靠.当前 SLC NAND 闪存的擦除重写次数限制为 100 000 次,而 MLC 的仅为 10 000 次.如果数据总是被写到同一个闪存地址,频繁的更新数据会使存储块失效.为了延长闪存的使用寿命,基本的方法是数据不在同一个地址被更新并且各个存储块均匀地被更新,这就是磨损均衡.也就是说需要将数据写入操作均匀地分布到各个存储块上,从而达到磨损均衡的目的.

本文选用了常见的三星 K9XXG08UXM[1-2] NAND 闪存进行分析讨论.图 1 给出了 16 GB MLC NAND 闪存芯片的内部结构.一个 16 GB 的闪存封装单元包含 8 个 2 GB 的闪存芯片.每个闪存芯片包含 8192 个存储块,由 4 个包含 2048 个存储块的存储面组成.闪存芯片可以独立操作,每一个操作包含对一个或两个存储面的操作.每个存储块包含 64 个 4 KB 大小的存储页.每个存储页有 4096＋128 B.在每个存储页中可以划分为用户数据区(4096 B)和冗余数据区(128 B).冗余数据区用于存放数据错误检测信息和其它元数据,用户数据区则用于存放通用的数据.

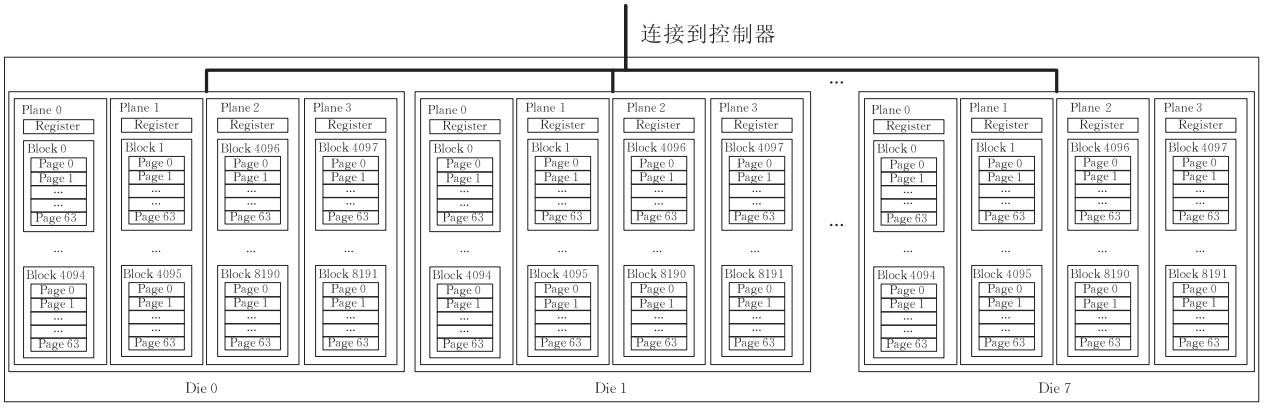

图 1　16 GB 闪存内部结构图

如图 2 所示,基于闪存的固态硬盘通常包含一个控制器和一系列的 NAND 闪存芯片.它给主机系统提供并行 ATA、串行 ATA、PCI-E 和 USB 等逻辑磁盘接口.控制器中包含一个闪存转换层(FTL).闪存转换层负责实现地址映射、磨损均衡、差错控制和垃圾回收等功能.由于垃圾回收的单位是数据块,当一个数据块要被擦除时,数据块中的有效数据必须要复制到其它空闲数据页来保证数据不会丢失.

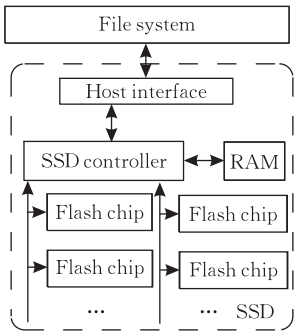

图 2 基于闪存的固态硬盘结构图

3 相关工作

在闪存中,数据更新程度是不一样的.更新频繁的数据称为热数据,而很少更改的数据称为冷数据.如果热数据在某一个存储块上被频繁地更新,则这个存储块可能比其它块更快失效.目前研究人员已经提出了很多有效的闪存磨损均衡算法[3].磨损均衡算法的目标是通过均匀地将擦除操作分布到整个存储空间来延长闪存的使用寿命.随着闪存技术的发展,闪存的容量不断增大,而支持磨损均衡的内存空间需求也变得很大.相比于闪存,人们对固态硬盘的磨损均衡算法研究较少.现有的磨损均衡算法可以被分成两类:动态磨损均衡算法和静态磨损均衡算法.

3.1 动态磨损均衡算法

动态磨损均衡算法主要是防止热数据写到同一个数据块上,因此不存在某个数据块比其它数据块更容易磨损失效.Ban[4]提出要在一定的数据擦除操作次数后随机选择一个数据块进行擦除.Woodhouse[5]提出,在100次擦除操作后要随机选择一个全部数据都是有效的数据块进行擦除.然而,动态磨损均衡算法的问题在于存储冷数据的块的擦除次数远小于那些存储热数据的数据块.这将导致冷热数据块之间的不均衡,热数据块很容易失效.

3.2 静态磨损均衡算法

为了防止冷数据块保持不变,研究人员又提出了很多静态磨损均衡算法.静态磨损均衡算法试图将冷数据移动到被擦过多次的块中从而保证更加均衡的磨损.Dual-Pool[6]方法维护了两个存储池,分别存放热数据块和冷数据块,通过数据计算来对两个存储池中的数据块进行交换,但会增加大量不必要的交换开销.BET[7]方法则采用一个"擦除位图表"来维护数据块的擦除状态.如果数据块的个数很多,则"擦除位图表"也将变得很大.

还有很多算法是通过维护整个闪存上数据块的信息来保证擦除操作的均衡性,但是当闪存容量变大的时候,它们需要更多的内存空间来存储每个数据块和数据页的信息以完成策略的运算.

4 随机游走磨损均衡算法

算法的主要目标是减少控制器的内存开销,降低能耗,其本身具有极强的容量可扩展性,即使固态硬盘的容量达到数百GB甚至上万GB也不会消耗太多内存容量.

由于闪存芯片的物理结构,考虑以一个包含2048个块的存储面作为一个分组.只为每个分组(一个存储面)维护少量的元数据信息,而不是维护每个数据块的信息.根据每个存储面的元数据信息,可以选择一个合适的存储面来进一步决定哪个块被擦除.然后采用有偏向的随机游走算法来最终决定需要擦除的目标数据块.

4.1 随机游走模型

随机游走[8-9]是一种有趣的理论,其基本含义是一系列运动中的每一步的方向和长度都是随机决定的,如同一个人随机地行走.一个最基本的随机游走模型是,在规则格子上随机游走的每一步都根据某个概率分布移动到另一个格子交叉点上.在最简单的随机游走中,每一步只能跨越一个方格而到达相邻的方格交叉点上.在有限方格的简单均衡随机游走中,每一步到达相邻的格子交叉点的概率是相同的.给出一个在N维空间整数格上随机游走的实例.首先,考虑在数轴上的一维随机游走,每一步移动都可能到达左边或右边的概率是

$$P(x)=\begin{cases}\frac{1}{2}, & x=\pm 1\\ 0, & \text{其它}\end{cases} \tag{1}$$

那么一维随机游走就是从原点出发,然后在一维数轴上不断重复随机游走过程.

根据图1,闪存芯片的内部可以看成是二维方格,那么一个最基本的想法就是将随机游走映射到闪存的层次化结构上.图3给出了在闪存上进行随机游走的情况.在行走过程的每一步都有4种移动方向的选择,X轴表示存储面号,Y轴代表数据块号.为了能使数据块均匀地被擦除,我们采用有偏向

的随机游走来控制每步移动的概率.对每个数据块要给出方向选择的权重作为参数.比如得到向左侧移动的概率是：

$$P(x)=\frac{w_{\text{left}}}{w_{\text{left}}+w_{\text{right}}} \tag{2}$$

w_{left}和w_{right}分别表示向左移动和向右移动的权重参数.

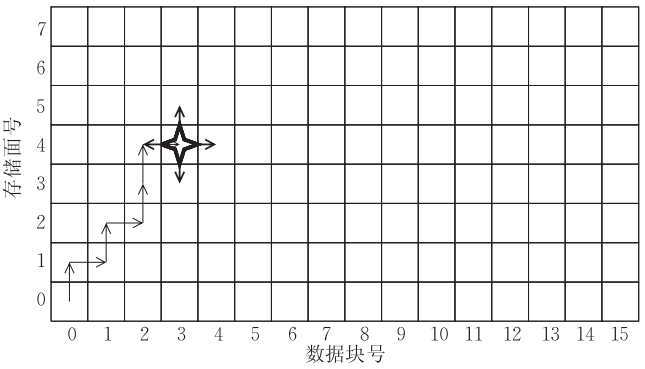

图 3　在闪存方格上的游走

4.2　闪存存储面上的元数据

动态随机存储器(DRAM)相比于闪存价格更高,而且 DRAM 内存容量更大时需要消耗更多的能量,会造成很大的电力损耗.通常,如图 2 所示,大容量固态硬盘的控制器会采用 64 MB 或更大的 DRAM 内存容量来存储控制磨损均衡的数据表.我们设计算法的主要目的是减少维护磨损均衡数据的内存开销,并且达到和其它算法相当的性能.我们提出的算法只在内存中维护每个闪存存储面的元数据,而不需要在内存中维护每个数据块的元数据.每个闪存存储面可以被认为是闪存数据块组.采用闪存存储面而不是任意大小的数据块组有更大的好处,闪存芯片支持一些新的操作,例如三星的闪存芯片支持内部的写回优化,允许数据在闪存存储面内部从一个数据块传输到另一块上不需要通过外围的引脚,从而加快传输速度.

在每个闪存存储面的元数据的维护中,我们选择了两种统计数据值,期望值 E 和方差 Var.在这种情况下,期望值是闪存存储面上的所有数据块的平均擦除次数.方差则表示相同闪存存储面上的所有数据块的各个擦除次数距离期望值的变化大小.较小的方差能够表明在某个存储面内的每个块的擦除次数趋向于平均擦除次数,而较大的方差则能够表明擦除次数的差异性很大.

在固态硬盘的控制器初始化时,应当读入每个存储面上每个块的擦除次数来计算相应的 E 和 Var.当一个块被擦除后,统计值需要被重新计算.例如,ec 用来表示一个数据块的擦除次数,如果通过一次擦除操作,其中一个数据块的擦除次数从 ec 更新到 ec',则存储面上的期望值更新为

$$E'=E+(ec'-ec)/N \tag{3}$$

N 表示一个闪存存储面上的数据块个数.

那么当一次擦除操作需要擦除同一存储面上的 $n(n\geq 1)$ 个数据块时,则存储面上的期望值更新为

$$E'=E+n/N \tag{4}$$

由于方差的计算依赖于期望值,同样条件下方差的计算公式如下所示：

$$Var'=Var+\frac{2n(ec-E)}{N}+\frac{(N-1)n^2}{N^2} \tag{5}$$

4.3　算法流程

算法基于大容量固态盘上广泛应用的块映射 FTL[10].块映射 FTL 相比页映射 FTL 需要更少的内存.FTL 是固态硬盘中的一个重要组成部分,主要负责地址转换,它同样需要在 DRAM 内存中维护一个地址转换表,基于块粒度的转换表显然比基于页粒度的转换表节省内存空间.在块映射 FTL 中,会将上层操作系统的逻辑块地址(LBA)请求转换为物理块地址(PBA).每个逻辑块地址会根据地址转换表被分成逻辑数据块号和偏移地址.利用物理块地址可以访问具体的存储面上的某数据块中的数据.

图 4 给出了随机游走磨损均衡算法的基本流程.在算法初始阶段,首先需要计算每个存储面的 E 和 Var.为了节省计算时间,每个存储面的元数据会事先存储到固态硬盘的特定区域中.系统开机后,直接将需要维护的元数据进行加载.若固态硬盘中不存在元数据表,则首先需要在固态硬盘控制器的内存中创建元数据表,创建元数据表需要读取每个存储面上的所有物理块的元数据中的擦除次数,然后计算每个存储面的数学期望E和方差Var.面内位

算法.基于随机游走的磨损均衡算法流程.

1. 利用式(4)和(5)为每个闪存存储面 i 计算 E_i 和 Var_i 初始化面内位置指针 p.
2. 根据 E_i 对存储面进行排序.
3. 利用方差 Var 在数学期望 E 最小的 M 个存储面组中选择方差最大的存储面为目标存储面.
4. 随机游走,利用数据块中的擦写次数来计算移动方向的概率.
5. 利用左右移动的概率来决定移动方向.
6. 将指针移动到目标数据块,更新面内位置指针 p.
7. 如果目标数据块块中含有有效数据,则将数据拷贝到合适的更新块中,并修改地址映射表.
8. 擦除目标数据块,更新目标块的擦除次数和相关的 E_i 和 Var_i.

图 4　算法流程

置指针初始化为 0，也就是指向存储面上的 0 号数据块. 然后按照 E 值的大小对存储面进行排序. 利用相应方差 Var 在数学期望 E 最小的 M 个存储面中选择方差最大的存储面作为目标存储面. 这样就找到了平均擦除次数少而擦除分布不均匀的存储面.

接下来，在目标存储面上进行随机游走的过程. 从目标存储面中维护的面内位置指针所指的位置作为起始位置，根据预定的步长向左或向右移动面内位置指针并利用当前位置相邻的数据块的擦除次数来影响左右移动的概率，计算向左移动的概率公式如式(6)所示：

$$P_{\mathrm{L}}=\frac{ec_{\mathrm{L}}}{ec_{\mathrm{L}}+ec_{\mathrm{R}}} \tag{6}$$

P_{L} 为向左移动的概率，ec_{L} 为左相邻数据块的擦除次数，ec_{R} 为右相邻数据块的擦除次数. 每步的随机游走均通过伪随机函数生成一个 0 到 1 区间的小数，若该小数在 0 到 P_{L} 之间，则向左移动；否则向右移动.

按照随机游走的结果移动面内的位置指针，从原来的位置基础上向左或向右移动步长个数据块. 面内位置指针移动到目标数据块后，更新面内位置指针 p 到新位置. 将一个存储面上的数据块按照块号构成一个循环，如果指针到达存储面的最后一块上，则右侧的块是 0 号块. 如果指针在 0 号块，则左侧是最后的数据块. 最终我们得到了需要擦除的那个目标数据块. 如果目标数据块中存在有效数据，则需要将有效数据拷贝到更新块，同时需要更新地址转换表，然后才能擦除目标数据块. 最后，更新目标存储面中维护的元数据的 E 和 Var 值.

为了防止固态硬盘的突然掉电，从而导致 DRAM 内存中的元数据丢失，设计了周期性的保存元数据的机制. 在指定的时间周期内，控制器会将内存中的元数据表写回到特定的存储区域中. 为了防止重复写回，增加一个全局写入计数器，当写入计数器不为零时，才需要将元数据信息进行写回操作. 每个周期结束后，计数器将会清零.

5 实验结果

5.1 实验配置

为了评测我们提出的基于随机游走的磨损均衡算法，我们采用 trace 驱动的软件模拟器 DiskSim SSD 扩展版[2]来进行评测. 主要的实验手段是对模拟器进行相应的代码修改，加入所提出的磨损均衡算法. 为了对其性能进行比较，还同时实现了 JFFS2[5]和 BET 算法[7].

我们选择了写操作密集的工作负载序列来评测算法的性能. 采用的工作负载序列来自企业级实际应用. UMASS 金融工作负载序列①是在线收集的金融单位的实际工作负载数据. 我们将其分别命名为 Financial1 和 Financial2. MSR 剑桥工作负载序列②是从微软剑桥研究院的企业服务器上采集一周的数据. 我们将其命名为 MSR.

对实验结果采用了 3 个测量基准来评价算法的磨损均衡的性能. 其分别是平均擦写次数 EX，擦写次数的标准差 $VarX$ 和最大擦写次数 $MaxX$，其计算方法如下：

$$EX=\frac{\sum_{i=1}^{N}ec_i}{N} \tag{7}$$

$$VarX=\sqrt{\frac{\sum_{i=1}^{N}(ec_i-EX)^2}{N-1}} \tag{8}$$

$$MaxX=Max\{ec_i\} \tag{9}$$

其中，ec_i 为各个数据块的擦除次数，N 为数据块个数.

5.2 性能比较

我们选择了 4 种方法来做性能对比实验，分别是：无磨损均衡、JFFS2 中的磨损均衡算法、BET 算法和我们提出的算法，分别将其标记为 No WL、JFFS2、BET、Our.

图 5 中显示了最大擦写次数，No WL 在每种测试数据中都有最大的擦写次数. 结合图 6 可以看出，在无磨损均衡的情况下，最大擦除次数与其相应的平均擦除次数差别较大，也就是说明磨损不是均衡的；JFFS2 中增加了磨损均衡的考虑，所以它的最大擦除次数和平均擦除次数差别小一些；BET 和我们提出的算法的性能差不多，在最大擦写次数和平均擦写次数的差别很小，擦除较为均衡.

最后，图 7 显示了擦写次数的标准差，这也验证了上述说明，在无磨损均衡的情况下，擦写次数的标准差非常高；JFFS2 方法的标准差有所降低；BET 和我们提出的算法的标准差都小于 JFFS2 且数值较小，这就意味着数据块的擦写次数相对更加均匀. 通过实验数据可以看出，我们提出的方法在磨损均衡的性能方面与 BET 相当.

① University of Massachusetts Amhesrst Storage Traces. http://traces.cs.umass.edu/index.php/Storage/Storage

② SNIA IOTTA Repository: MSR Cambridge Block I/O Trace. http://iotta.snia.org/traces/list/BlockIO

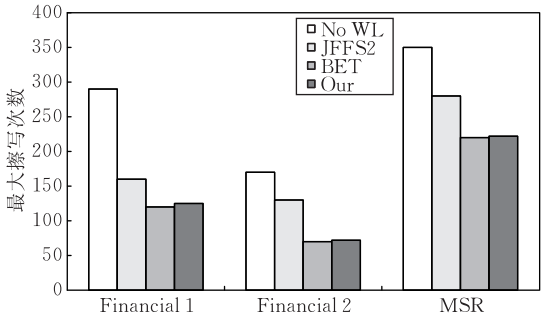

图 5　最大擦写次数

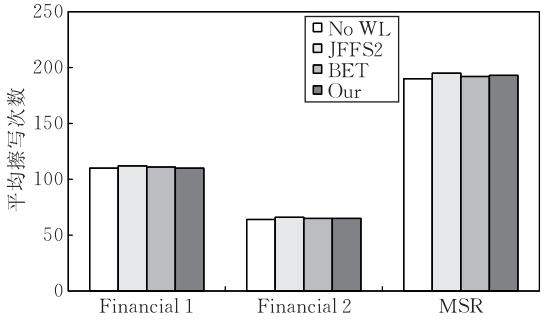

图 6　平均擦写次数

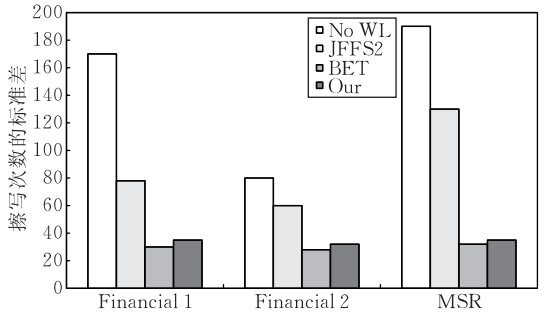

图 7　擦写次数的标准差

5.3　内存消耗

算法中主要维护了两个统计量：存储面上各数据块的擦除次数的期望 E 和方差 Var，为了维护这些数据，需要为每个数据在内存中分配 32 位的存储空间. 此外，由于每个存储面有 2048 个数据块. 那么面内位置指针 p 只需要 16 位来保存. 总共每个存储面需要 10 个字节的内存消耗. 在 BET 算法中，其使用位图来标示数据块的擦除状态，每个数据块用 1 位来表示. 对于 64 GB 的固态硬盘，BET 算法需要 8192 字节的空间(512×16＝8192 字节)来保证算法的正常运行. 对于我们提出的算法，总共元数据需要 1280 字节. 和 BET 相比，我们提出的算法只需要其占用空间的 15.6%. 图 8 给出了在不同容量下，我们所提的算法和 BET 算法所占用的内存容量对比. 可以看到即使固态硬盘容量达到 10 TB，我们的算法所需要的内存容量也只有 200 KB，而 BET 算法则需要 1.25 MB 的内存容量.

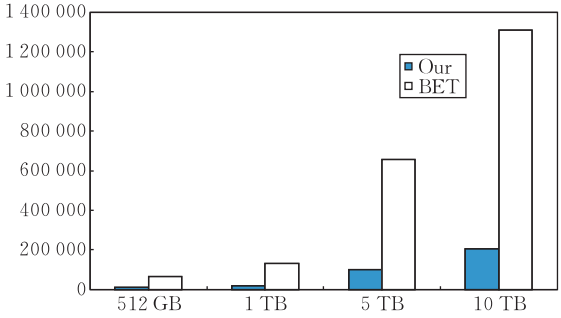

图 8　内存消耗容量对比

6　总　结

本文提出了一种应用于大容量固态硬盘的基于随机游走的磨损均衡算法. 算法中主要维护了两个统计量：期望 E 和方差 Var，还有每个存储面上的面内位置指针 p. 算法主要采用了随机游走的模型来选择需要擦出的目标块.

我们的实验结果表明，该算法在运行时只需要很少的内存空间就可以得到和其它算法大致相同的磨损均衡性能. 自身具有很好的可扩展性，可以更好地应用在更大容量的固态硬盘上.

致　谢　本文得到了清华大学计算机科学与技术系操作系统实验室的老师与同学们的许多帮助和有益建议，在此表示感谢！

参　考　文　献

[1] Samsung Corporation. K9XXG08XXM Flash Memory Specification. 2007

[2] Agrawal N, Prabhakaran V, Wobber T et al. Design tradeoffs for SSD performance//Proceedings of the USENIX 2008 Annual Technical Conference. Boston, USA, 2008: 57-70

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

[4] Ban A. Wear leveling of static areas in flash memory. US Patent App. 09/870, 315, Jun. 1 2001

[5] Woodhouse D. JFFS: The journalling flash file system//Proceedings of the Ottawa Linux Symposium. Ottawa, Canada, 2001

[6] Chang L. On efficient wear leveling for large-scale flash memory storage systems//Proceedings of the 2007 ACM symposium on Applied computing. Seoul, Korea, 2007: 1130

[7] Chang Y H, Hsieh J W, Kuo T W. Endurance enhancement of flash-memory storage systems: An efficient static wear leveling design//Proceedings of the 44th Annual Design Automation Conference. Berkeley, USA, 2007: 217

[8] Spitzer F. Principles of Random Walk. Berlin: Springer Verlag, 2001

[9] Boyd J N, Raychowdhury P N. Biased Random Walks. Virginia Journal of Science, 1995, 46(1): 35

[10] Kim J M, Noh S H et al. A space efficient flash translation layer for compact flash systems. IEEE Transactions on Consumer Electronics, 2002, 48(2): 366-375

ZHAO Peng, born in 1985, Ph. D. candidate. His research interests include emerging memory, memory management in operating system, computer security.

BAI Shi, born in 1982, Ph. D. candidate. His research interests include storage system, operating system

Background

The flash is a non-volatile storage medium, it has small volume, low energy consumption, low strong vibration resistance etc, it is widely used in MP3 players, mobile phones, handheld computers and other electronic consumer equipments. With flash memory capacity increases, Solid-state disk appears. The current solid-state disk interface is general hard disk interface simulation, but its storage medium is flash memory, it has been widely used in notebook computers, like Apple Mac Air serial. It used in place of disk storage, to make up for the high energy consumption and disk performance etc. Solid-state disk is becoming mainstream storage devices.

When the flash memory chip achieves the limit number of erase, data become less reliable. The current SLC NAND flash memory erasing limit is 100 000 times, and only 10 000 times for the MLC. If the data is always wrote the same flash memory address, frequent updates data storage blocks will make the failure. In order to prolong the service life of flash memory, and the basic methods is that the data is not updated in the same address and various storage blocks uniform is updated, this is a balanced wear-leveling. That is, the erase operation will be evenly distributed on all storage blocks. Now researchers have put forward some wear-leveling algorithm to solve the problem. But when the solid-state disk storage capacity increases, the algorithm needs more and more memory capacity to ensure operation.