基于分组—循环擦写的闪存磨损均衡算法

王 浩，邵高平，胡泽明
（解放军信息工程大学 信息系统工程学院，郑州 450002）

摘 要：为延长作为终端外部存储设备的闪存介质的使用寿命，采用分组—循环擦写的思想设计了一种闪存磨损均衡算法。该算法将闪存物理块分成若干组，在满足态势信息快速存取要求的前提下，使不同组中同一用户的信息存储区域循环地参与擦写。仿真实验结果表明该算法能够使得擦写操作较为均匀地发生在闪存上。

关键词：移动终端；战场态势信息；磨损均衡；闪存；分组
中图分类号：TP303 **文献标志码**：A **文章编号**：1001-3695(2013)05-1353-04
doi:10.3969/j.issn.1001-3695.2013.05.018

Flash memory wear-leveling algorithm based on group-erase circularly

WANG Hao, SHAO Gao-ping, HU Ze-ming
(*Institute of Information System Engineering, PLA Information Engineering University, Zhengzhou* 450002, *China*)

Abstract: In order to extend the lifetime of flash memory, this paper proposed a flash memory wear-leveling algorithm based on group-erase circularly. The major idea of this algorithm was grouping physical blocks to erase domains in turn. Simulation results show that the algorithm can introduce evenly erase operation on flash.

Key words: mobile terminal; battlefield situation information; wear-leveling; flash memory; group

0 引言

与传统的硬盘存储相比，闪存具有抗震动、低能耗、随机访问和良好的散热性能等优势[1]。因此，在当前军用及航空航天领域，普遍采用闪存作为移动终端信息存储介质。尽管闪存存储容量越来越大，仍无法完全存储海量的战场态势信息，过时的信息需要被擦除并将新接收的态势信息写入。然而，闪存的每个存储单元（物理块）的写/擦除操作次数有限，超过一定的次数后数据的可靠性将得不到保证。所以需要运用磨损均衡（wear-leveling）算法使擦除操作均匀地发生在各个单元上。

现有的磨损均衡算法分为动态和静态算法两大类[2]。动态磨损均衡算法在垃圾回收过程中综合考虑磨损程度来选择被擦除的块。而静态算法按照处理过程中是否带有随机性又可以分为随机性和确定性磨损均衡算法[3]。随机性磨损均衡策略每隔一定数目的擦除操作，随机选择一个物理块进行擦除；确定性磨损均衡机制依据物理块的擦除次数之差是否超过给定的阈值来决定是否启动磨损均衡，实施冷、热数据交换。存储战场态势信息的移动终端要满足作战人员快速存取态势信息的需求，其闪存上的信息存储必须以快速检索为原则，现有磨损均衡研究中的块上数据交换[4~7]将会增大检索复杂度；闪存容量有限，需要不断地将存储时间最长、最老的信息整区域擦除，这不同于现有磨损均衡研究中的存储单元内的部分更新。因此需要针对这一具体应用来设计相应的闪存磨损均衡算法。

本文在简要介绍闪存特性的基础上，根据战场态势信息存储终端的特点，提出了一种磨损均衡算法。该算法结合组（群）[8~10]的思想将闪存物理块按照物理位置分成若干组，组内依据用户个数划分为若干区，利用所定义的数据结构使得同一用户的各个信息存储区依次循环地参与擦除操作。最后通过实验验证了算法的有效性。

1 闪存及战场态势信息存储

1.1 闪存特性

通常，闪存由若干个物理块（physical block）阵列组成，而每个物理块又分为若干物理页（physical page）。闪存在写之前必须以物理块为单位进行擦除，而读/写则是以物理页为单位进行的。当前占据市场主流的闪存类型主要有 NOR 和 NAND 两种。NOR flash 以并行的方式连接存储单元，有足够的地址引脚来寻址，故其读取速度较快，但写操作和擦除操作的耗时较长，且容量低、价格高，一般多用于代码存储。NAND flash 使用复杂的 I/O 端口来串行地存取数据，故其读取速度较慢，但其写操作和擦除操作相比 NOR flash 较快，且容量大、价格较低。同时，NAND flash 每个块的最大擦写次数是一百万次，而 NOR flash 的擦写次数是十万次。故在大容量数据存储设备中广泛采用 NAND flash。本文以下的讨论是基于 NAND flash 进行的。

闪存所具有的上述特点使得对闪存的存储管理变得十分重要。通过闪存转换层（flash translation layers）为闪存存储器提供块设备接口，使得已有的基于磁盘存储的文件系统无须修改就能运用。基于 NAND 型闪存的嵌入式存储系统的体系结

收稿日期：2012-09-13；**修回日期**：2012-10-19

作者简介：王浩（1987-），男，河南罗山人，硕士研究生，主要研究方向为嵌入式系统与智能信息处理（wanghao9701@163.com）；邵高平（1964-），男，浙江桐庐人，教授，硕导，主要研究方向为信号检测、分析与处理；胡泽明（1977-），男，河南新县人，讲师，博士，主要研究方向为嵌入式 GIS 系统、战场态势信息高效存储.

构如图 1 所示[11]。

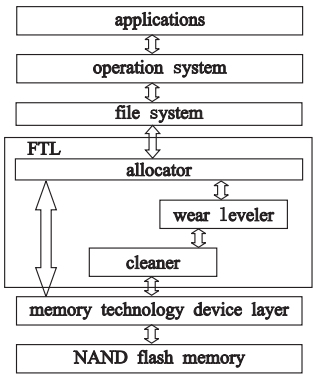

图1 NAND型闪存系统结构

FTL 使用常驻内存的索引结构将逻辑扇区映射到物理地址,其中 allocator 维护着逻辑地址与物理地址对应关系的映射表,根据映射表的粒度大小,可分为基于页、基于块和混合式的地址映射[12]。Cleaner 负责垃圾回收管理,而 wear leveler 用于均衡各物理块的擦除次数,避免对某区域反复擦除导致闪存数据存储性能下降。不同应用背景下的磨损均衡机制是不同的,因此,战场态势信息存储终端的磨损均衡应结合其存储特点进行设计。

1.2 基于 NAND flash 的战场态势信息存储

战场环境下,作战单元通过其所携带的相关终端设备获取战场综合态势信息,这些态势信息主要有定位信息、情报信息、标绘信息和代码指挥信息等。每个终端上既存储有本机信息也有其他终端信息。移动终端态势信息交互示意图如图 2 所示。

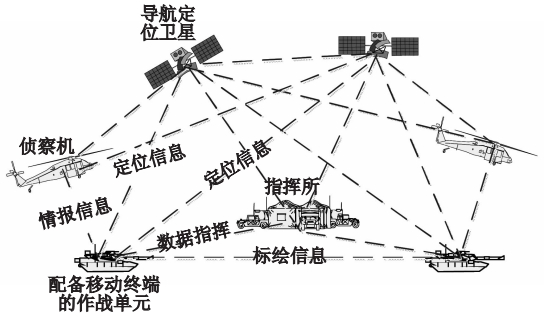

图2 终端态势信息交互示意图

作战单元通过检索终端所存储的历史态势信息,实现战场态势回演,辅助决策进而提升部队协同作战能力。由于战场态势信息数据量庞大、用户从几十个到上百个不等,若将各个用户的态势信息集中混合存储势必会增加检索设计复杂度,而嵌入式系统资源有限,这将使得查询耗时延长。为满足用户的快速查询需求,NAND flash 上的态势信息存储通常借鉴磁盘分区的思想,将属于不同用户的态势信息分区存储,而在分区内,将不同类型的态势信息再次分区存储,如图 3 所示。其中,每个分区是一组连续的物理块,如此,通过建立较为简单的索引就可以较为便捷地查找到所需信息。

战场态势信息在短时间内可能会出现爆炸式增长,信息终端数据输入速率会很高。尽管当前 NAND 型闪存的主流容量已达到 64 GB,也无法存放整个作战时间(一般要求为 7 × 24 h)内的所有态势信息。因此,需要不断地将过时的信息擦除,而频繁的擦除将导致分区物理块的提前损坏,数据存储将变得不可靠。针对该存储模式,需要设计相应的磨损均衡机制。

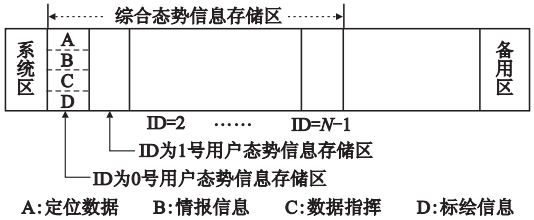

图3 基于NAND型闪存的态势信息分区存储示意图

2 磨损均衡算法的设计

图 3 所示存储方案中,所有用户共享整个态势信息存储区,当不断接收到战场态势信息时,在 NAND flash 中靠前的态势信息存储区将不断地被擦写,其使用频率将大大超过靠后的部分,造成磨损的不均衡,导致该物理扇区在较短的时间内被损坏并丢失全部存储数据。同时,由于分配给每个用户的信息存储空间较大,进行信息读/写的寻址将会耗费较长时间。为此,本文设计了一种基于分组一循环擦写的磨损均衡算法,在保证态势信息快速存取的同时实现闪存的磨损均衡。

2.1 基于组的循环擦除

为充分合理利用闪存上的有效存储区域并在一定程度上实现磨损均衡,将系统区和备用区中间的区域分成$2^k(k \in N^+)$个组(group),组内存储所有的 $m(m \in N^+)$个用户的态势信息,如图 4 所示,其中每个组的每个用户占有连续的 2^n 个物理块。

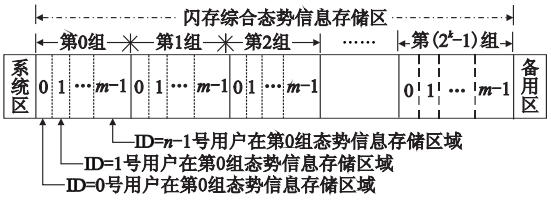

图4 态势信息存储区分组示意图

当某一时刻各个组中 ID = i 的所有存储区域均已写满时,若再接收到与该用户相关的态势信息,将 2^k 个组中的存储 ID 为 i 的、存放时间最长的区域(即该组中该用户占有的连续 2^n 个物理块)擦除、重写新数据。随着时间的推移、态势信息的不断更新,各个组所存储部分或所有用户的态势信息将循环地成为过时的信息,即所有组依次循环地参与擦除,从而实现闪存综合态势信息存储区的磨损均衡。

2.2 算法的数据结构和符号定义

为准确定位循环擦写的区域,该算法需要利用如下两个数据结构:

a)组内各 ID 区域写次数映射表(GWCT)。该表记录了 2^k 个组中各个用户信息存储区域被写的次数,由于闪存物理块在写之前要进行擦除操作,因此写次数也反映了区域擦除次数。由于用户信息存储区域(如第 0 组 ID 为 1 的态势信息存储区域)是连续 2^n 个物理块的集合,某一次操作可能只写了该区域的部分物理块,此时区域写次数保持不变,下次再写之前就不能进行擦除操作,而是顺序写该区域剩下的空闲物理块。为此,还需要使用一种数据结构来标志各个用户信息存储区域是否写满。

b)区域写满标志表(DFMT)。该表是一个 bit 数组,每个 bit 位代表一个块的写满状态,当且仅当区域所有的 2^n 个物理块全部写满而不存在空闲块时,区域所对应的 bit 为 1,否则为 0。

如图 5 中 GWCT 所示的某一时刻各个组内所有用户信息

存储区域的写次数情况,DFMT 为该时刻各区域写满标志。此时,第 0 组的 ID 为 0 号用户区域写次数为 3 次且已写满,ID 为 1 号用户区域写次数为 2 次且已写满;而第 1 组的 ID 为 0 号用户区域写次数为两次且未写满,故若再接收到 ID 为 0 号用户的态势信息时将其写入到第一组对应位置中,若再接收到 ID 为 1 号用户的态势信息时,由于所有组中 ID = 1 的区域写次数均为 2 次且都已写满,则需要将第 0 组该用户对应存储区域擦除再写。如此,将磨损均衡从擦除的角度转换到写的层面上进行分析。

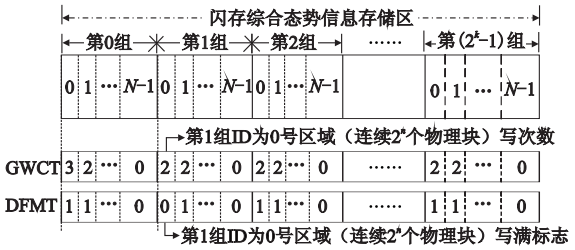

图5　GWCT和DFMT表映射示意图

GWCT 和 DFMT 采用双备份机制存储:在固定时间周期内交替存储(回写)于系统区和备用区,终端上电时将其读入到内存中。该机制不仅能够避免因设备突然断电造成算法失效,而且在一定程度上实现了系统区和备用区的磨损均衡。

算法中所用到的符号定义如下:

DFMT(ID = i)表示各个组内 ID 为 i 的用户信息存储区域满标志映射表;

Domain 表示任一组内任何一个用户所占用的连续的 2^n 个物理块;

g 表示一个组编号,其值从 0 到 $2^k - 1$;

GWCT(ID = i)表示各个组内 ID = i 的信息存储区域写次数映射表;

Domain{group = g, ID = i}表示第 g 组内 ID 为 i 的 domain;

Timer 表示计时器的值,其初始值为 0;

T 表示算法使用的两个数据结构回写的时间周期值。

2.3　算法流程

对某一区域执行完擦除和写的操作后,需要修改 GWCT 和 DFMT,并将这些结构作为参数传入到算法中。算法的伪代码如下:

输入:GWCT,DFMT。

输出:GWCT,DFMT。

Timer←0;

while 接收到 ID = i 的态势信息 do

遍历 GWCT 得到 GWCT(ID = i),判断 GWCT(ID = i)中各元素值是否相等;

if 各元素值相等

遍历 DFMT 得到 DFMT(ID = i),判断 DFMT(ID = i)中是否存在零元素;

if DFMT(ID = i)中存在零元素,其组号为 g

WriteFunction(domain{group = g, ID = i});

else

EraseFunction(domain{group = 0, ID = i});

WriteFunction(domain{group = 0, ID = i});

else

在表 GWCT(ID = i)中寻找写次数最多的组 g;

遍历 DFMT 得到 DFMT(ID = i),判断 DFMT(ID = i)中组号为 g 所对应的元素是否为零;

if 该元素为零

WriteFunction(domain{group = g, ID = i});

else

WriteFunction(Domain{group = g + 1, ID = i});

更新 GWCT,DFMT;

if 计数器 Timer > = T;

将 GWCT,DFMT 交替写入系统区和备用区;

Timer←0;

End while

代码中 EraseFunction 函数将其参数所指定的 domain 擦除,而函数 WriteFunction 执行将态势信息写入指定 domain。

3　仿真分析

下面通过实验来验证该磨损均衡算法的有效性。其实验比较对象为图 3 所示的不考虑均衡磨损的态势信息存储,称为参照组;而使用该磨损均衡算法的信息存储称为实验组。由于该存储方案下各个用户的态势信息接收均衡程度会对实验结果产生较大影响,因此,将从极端情况下(每次只能接收到本机用户信息)和理想状况下(每次均能接收到各个用户的等量的态势信息)进行仿真。

3.1　仿真设置

为将实验组和参照组进行对比,搭建了仿真实验平台。其中 NAND 型闪存仿真模型的容量为 4 MB(不包括系统区和备用区),每个物理页大小为 2 KB,每个物理块包含 64 个物理页,则总共有 32 个物理块,每个块的可擦除上限为 10 万次,只要有块的擦除次数达到其上限,则认为闪存达到其使用寿命。同时假设移动终端有 4 个(即有 4 个态势信息用户)。对于实验组,设定 2 个物理块为 1 个 domain,即 1 个 domain 的容量大小为 256 KB,整个态势信息存储区共划分为 2^2 个组,故一个组包含 4 个 domain(即 8 个物理块)。对于参照组,为与实验组进行比较,设定其态势信息存储区域容量为实验组的一个组的大小。

为降低仿真复杂度,考虑到战场综合态势信息中定位信息所具有的特点(如 GPS 系统可达到 1 s 定位一次,每条定位信息数据量一般为十到几十个字节),故将终端所接收到的态势信息进行如下设置:极端情况下(简称情况Ⅰ),每一秒只接收一条本机用户的定位信息;理想情况下(简称情况Ⅱ),每秒接收到每一个用户的一条定位信息。同时,本机用户编号为 ID = 0。为减少写次数,定位信息采用缓存存储,即将用户的定位信息缓存在内存中,当达到一个物理块的容量时一次性写入。

3.2　实验结果比较与分析

磨损均衡算法的目的是使擦除操作较为均匀地发生在各个物理块上,对于本文所提的应用场景而言,参与的块数越多,擦除操作分布得就越均匀。为此,提出一个评价该算法的指标:参与块数百分比(percent of blocks involved, PBI),即参与擦写的物理块数占可用存储区物理块数的百分比。截止闪存到达其使用寿命,在情况Ⅰ下,参照组和实验组各个块的擦除情况如图 6 所示,纵轴表示擦除次数(单位:万次),横轴表示物理块编号(从 0 到 31);在情况Ⅱ时两组的擦除情况如图 7 所示。

由图 6 可知,无论在哪种情况下,使用本文所提出的磨损均衡算法使得被擦写的物理块分布得较为均匀,这是由于算法基于分组循环擦写的思想,使得各个组中各个用户所对应的 do-

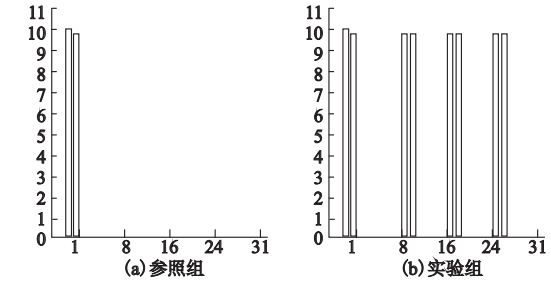

图6 Ⅰ两组闪存擦除情况

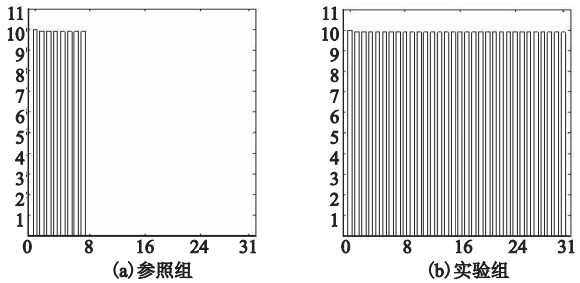

图7 Ⅱ两组闪存擦除情况

main 能被等概率擦写。同时,两种情况下的 PBI 比较见表 1。

表 1 两种情况下 PBI 比较

	Ⅰ参照组	Ⅰ实验组	Ⅱ参照组	Ⅱ实验组
PBI	12.5%	25%	25%	99.9%

显然,无论在哪种情况下,在满足态势信息快速存取的前提下该算法使得参与擦写操作的物理块更多、闪存的磨损均衡效果更好一些。

4 结束语

应用于战场环境下的移动终端要求态势信息能够快速存取,而有限的存储空间需要不断地将过时的信息擦除。结合该具体应用背景,本文设计了一种闪存磨损均衡算法。该算法基于分组循环擦除的思想,在满足态势信息能够快速存取、擦除最过时信息的前提下使得更多的空间循环参与擦写。仿真实验结果表明该算法是有效的。下一步的工作是在某型号终端上验证并完善该算法。

参考文献:

[1] CHANG Li-pin, HUANG Lu-chun. A low-cost wear-leveling algorithm for block-mapping solid-state disks[C]//Proc of Conference on Languages, Compilers and Tools for Embedded Systems. New York: ACM Press, 2011: 31-40.

[2] GAL E, TOLEDO S. Algorithms and data structures for flash memories [J]. ACM Computing Surveys, 2005, 37(2): 138-163.

[3] 黄德才,邢春波,吕莲. 闪存磨损均衡算法综述[J]. 浙江工业大学学报, 2009, 37(1): 73-78.

[4] LOFGREN K, NORMAN R. Wear leveling techniques for flash E^{2}PROM systems: USA, US007353325B2[P]. 2008-04-01.

[5] CHANG Li-pin. On efficient wear leveling for large-scale flash-memory storage systems[C]//Proc of ACM Symposium on Applied Computing. [S. l.]: ACM Press, 2007: 1126-1130.

[6] JUNG D, CHAE Y, JO H, *et al*. A group-based wear-leveling algorithm for large-capacity flash memory storage systems[C]//Proc of the 2007 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. New York: ACM Press, 2007: 160-164.

[7] 邢春波, 杨良怀, 龚卫华, 等. 一种有效的混合式闪存磨损均衡算法[J]. 小型微型计算机系统, 2009, 30(9): 1903-1906.

[8] CHANG Y, HSIEH J, KUO T. Endurance enhancement of flash memory storage systems: an efficient static wear leveling design [C]//Proc of the 44th Annual Conference on Design Automation. New York: ACM Press, 2007: 212-217.

[9] LEE C C. System and method for managing blocks in flash memory: US, 0204187[P]. 2005-09-15.

[10] 张骏, 樊晓桠, 刘松鹤. 一种 flash 存储器静态负载平衡策略[J]. 计算机应用, 2006, 26(5): 1205-1207.

[11] 邢春波. 闪存磨损均衡算法研究[D]. 杭州:浙江工业大学, 2009.

[12] 孙文静, 李明强, 舒继武. Flash 存储技术[J]. 计算机研究与发展, 2010, 47(4): 716-726.

(上接第 1322 页)

4 结束语

本文对蝙蝠算法进行了理论分析,该算法吸收了自然界中蝙蝠通过超声波搜索、捕食猎物的行为特性,利用进化的方式来实现智能体的行为以达到优化的目的。算法实现简单,具有本质并行性。通过选取的标准算例进行仿真测试,表明了蝙蝠算法在连续空间和离散空间优化的可行性和有效性,具有良好的应用前景。由于蝙蝠算法优化理论和应用研究还处于初始阶段,许多问题还有待于人们不断地探索和解决,如算法中涉及的参数设置的理论依据、算法的收敛性分析以及与其他群智能优化算法的有机融合等,这些都是进一步要做的研究工作。

参考文献:

[1] 李晓磊,邵之江,钱积新. 一种基于动物自治体的寻优模式: 鱼群算法[J]. 系统工程理论与实践, 2002, 22(11): 32-38.

[2] KARABOGA D. An idea based on honey bee swarm for numerical optimization, Technical Report-TR06[R]. Kayseri: Erciyes University, 2005.

[3] KARABOGA D, AKAY B. A survey: algorithms simulating bee swarm intelligence[J]. Artificial Intelligence Review, 2009, 31(1-4): 61-85.

[4] KRISHNANAND K N, GHOSE D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics[C]//Proc of IEEE Swarm Intelligence Symposium. Piscataway: IEEE Computer Society, 2005: 84-91.

[5] YANG X S. nATURE Inspired metaheuristic algorithms[M]. Frome. UK: Luniver Press. 2008: 83-96.

[6] YANG Xin-she. Nature-inspired metaheuristic algorithms[M]. 2nd ed. Frome, UK: Luniver Press, 2010: 97-104.

[7] YANG Xin-she. A new metaheuristic bat-inspired algorithm[M]// GONZÁLEZ J R, PELTA D A. Nature Inspired Cooperative Strategies for Optimization. Berlin: Springer, 2010: 65-74.

[8] SHI Yu-hui, EBERHART R. A modified particle swarm optimizer [C]//Proc of IEEE International Conference on Evolutionary Computation. Anchorage: IEEE Press, 1998: 69-73.

[9] 王凌, 刘波. 微粒群优化与调度算法[M]. 北京: 清华大学出版社, 2008: 125-126.

[10] TASGETREN M F, SEVKLI M, LIANG Y C, *et al*. Particle swarm optimization algorithm for permutation flowshop sequencing problem [C]//Lecture Notes in Computer Science, vol 3172. 2004: 382-389.