一种基于差分进化的 Flash 文件系统垃圾回收算法

时 正,纪金松,陈香兰,龚育昌
(中国科学技术大学计算机系,安徽合肥 230027)

摘 要: 垃圾回收算法对于 Flash 文件系统具有十分重要的意义.本文针对已有垃圾回收算法在存储容量剩余较小时垃圾回收性能急剧下降进而影响写入速率的问题,采用最优化方法,提出并实现了一种基于差分进化的垃圾回收算法.该算法能够使得垃圾回收的代价均匀化,在保证性能和损耗均衡的前提下,减少擦除次数,延长 Flash 寿命.实验结果验证了该算法的有效性.

关键词: 差分进化算法; 垃圾回收; 损耗均衡; Flash 文件系统
中图分类号: TP315 **文献标识码:** A **文章编号:** 0372-2112 (2011) 02-0280-05

A Garbage Collection Algorithm for Flash File System Based on Differential Evolution

SHI Zheng, JI Jin-song, CHEN Xiang-lan, GONG Yu-chang
(*Department of Computer Science, University of Science and Technology of China, Hefei, Anhui* 230027, *China*)

Abstract: Flash memory based storage systems are becoming increasingly prevalent. Garbage collection plays an important role in such systems. This paper analyzes popular file system overload, and points out the problems of existing GC algorithms: the performance of Garbage Collection decreases dramatically under high capacity utilization, which has a great impact on write speed. Then the paper proposes a new flash memory garbage collection mechanism based on differential evolution algorithm (DEbGC), which considers the speed and wear leveling in garbage collection. The experimental results show that DEbGC could amortize the overhead of garbage collection, decrease erase count and number of page copies while satisfying overall performance and wear leveling.

Key words: differential evolution algorithm; garbage collection; wear leveling; flash file system

1 引言

在嵌入式系统中,由于 Flash 的写操作必须在擦除操作之后进行,而擦除单元(块)的大小远大于写单元(块),所以面向 Flash 的各种更新算法都采取的是异地更新(out-of-place update)策略[1];更新之后原来的页面就变成了无效页面,所以基于 Flash 的存储系统都需要垃圾回收(Garbage Collection, GC).垃圾回收的过程一般是先选中 1 个(或多个)擦除块作为目标块,把其中的有效页面拷贝到其他块上的空闲空间,再擦除这些目标块,使目标块还原为空闲块.目标块中有效页面数越多,无效页面越少,垃圾回收的时间越长.同时因为 Flash 存在擦除次数限制,垃圾回收算法还需要考虑损耗均衡,即保持各个擦除块的均匀使用,使得块之间的擦除次数差距尽可能的小.

GC问题已被证明是个 NP-完全问题[2],已经有若干 Flash 文件系统的研究工作围绕 GC 展开.NAMU[3]根据不同触发时机采用不同的垃圾回收策略:在写请求导致的垃圾回收中,采取贪心策略;在空闲时刻的垃圾回收中则采用记录策略.Lee[4]等则提出了一种以"迁移"代替"合并"的思想,在一定程度上减少了垃圾回收的代价,提高了整体性能.Bennett[5]等通过调度损耗均衡和垃圾回收等操作,降低了写操作的延迟,提高了系统性能.Seungjae 等[6]指出 GC 开销与擦除块存储内容一致性具有非常大的关系.DAC[7]将冷热数据分开存放,并在 GC 的时候综合考虑数据冷热属性和有效页面个数,提高了 GC 效率.Peng 等[8]采用类似思想,为生存周期不同的数据分配不同的擦除块集合,提高了 GC 效率,并通过轮换使用不同的擦除块集合达到损耗均衡.这些工作在进行 GC 优化时,或者考虑数据的分配策略,

收稿日期:2009-10-21;修回日期:2010-01-27

或者考虑不同的 GC 目标块选取标准、触发时机,但算法都集中在于 GC 的局部优化,而忽略了全局的效果.实验指出随着 Flash 空间利用率的上升,GC 效率会急剧上升,引发整体性能的强烈波动.针对上述问题,本文首先对垃圾回收问题进行了抽象和定义,并提出了一种基于差分进化的回收算法,在兼顾 GC 效率和损耗均衡的前提下,减少 Flash 芯片的擦除次数;然后给出了算法复杂度分析以及实验结果分析.

2　垃圾回收问题的描述

Flash 文件系统垃圾回收问题主要可以分为候选块的选择和目标块的选择两个子问题.

2.1　问题描述

定义 1　令第 i 次垃圾回收时 Flash 中的脏块为 $B_{i.1},\cdots,B_{i.n_i}$,每个脏块中包含的无效页面数分别为 $p_{i.1},\cdots,p_{i.n_i}$,每个块的擦除次数记为 $e_{i.1},\cdots,e_{i.n_i}$.令 D_i 是垃圾回收的时间约束(写响应时间),N 为回收的块约束(最多从 N 个块中回收页面).D_p 为拷贝单个页面的时间.W 为损耗均衡系数,即最热块与最冷块的擦除次数差值的上限.二进制变量 $s_{i.j}$ 表示是否选中该脏块进行垃圾回收($s_{i.j}$ 为 1,则表示在第 i 次垃圾回收时选择了 $B_{i.j}$,为 0 则表示没有选择);损耗均衡的垃圾回收算法中候选块的选择问题可以抽象为如下多约束的最优化问题:

$$\min:\sum_{i=1}^{\infty}\sum_{j=1}^{n_i}(s_{i,j}\times e_{i,j}) \tag{1}$$

$$\text{s.t.}\quad \forall i,\ 0<\sum_{j=1}^{n_i}s_{i.j}\leqslant N;$$

$$\sum_{j=1}^{n_i}(s_{i,j}\times p_{i,j}\times D_p)\leqslant D_i;$$

$$\forall 0<j,k<n_i,\ |e_{i,j}-e_{i,k}|\leqslant W.$$

也就是在满足响应时间约束和损坏均衡约束的前提下,尽可能的减少擦除次数,延长 Flash 寿命.

定义 2　令第 i 次垃圾回收时 Flash 中的目标候选块为 $B'_{i.1},\cdots,B'_{i.n'_i}$,这些块的擦除次数为 $e'_{i.1},\cdots,e'_{i.n'_i}$,待写入页面所在块的活跃程度为 $a_{i,1},\cdots,a_{i,k}$(即因页面数据被重写到新的页面而导致该页面无效的概率),回收的页面数分别为 $p'_{i,1},\cdots,p'_{i,k}$,其中 k 为回收的块数.二进制变量 $s'_{i.j}$ 表示是否选中该目标候选块.则损耗均衡的垃圾回收算法中目标块的选择问题可以抽象为如下最优化问题:

$$\min:\sum_{i=1}^{\infty}\sum_{j=1}^{n'_i}\Big(s'_{i,j}\times e'_{i,j}\sum_{j=1}^{k}(a'_{i,j}\times p'_{i,j})\Big) \tag{2}$$

$$\text{s.t.}\quad \forall i,\ \sum_{j=1}^{n_i}s'_{i,j}=1$$

也就是利用页面活跃程度预测来尽可能地保证物理块的损耗均衡.

2.2　一个简单的示例

如图 1 所示,候选块为 $B_{i.1},\cdots,B_{i.4_i}$,图中以颜色深浅表示擦除次数,以有底纹的表示脏数据块.其中假设每个块总页面数为 16,$n_i=4$,$N=1$,$W=5$,$D_i/D_p=8$,即在 4 个候选块中选择 1 个块进行回收,回收的时间不能超过 8 个页面拷贝时间,且保证所有块的擦除次数相差不超过 5.

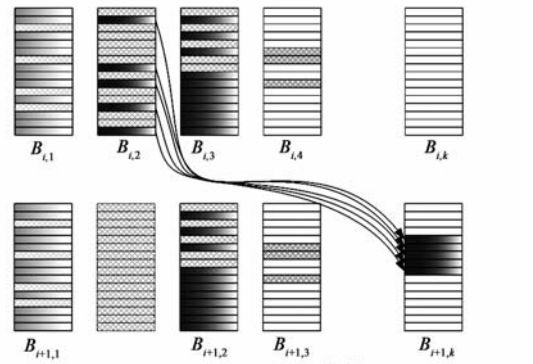

图1　垃圾回收候选块选择问题简单示例

从图中我们可以看到:

$$(e_{i.1},e_{i.2},e_{i.3},e_{i.4})=(22,23,25,20)$$

$$(p_{i.1},p_{i.2},p_{i.3},p_{i.4})=(12,5,11,13)$$

图中的选择为 $(s_{i.1},s_{i.2},s_{i.3},s_{i.4})=(0,1,0,0)$,这样满足了所有的约束:

$$0<\sum_{j=1}^{4}s_{i.j}=1\leqslant 1;$$

$$\sum_{j=1}^{4}(s_{i,j}\times p_{i,j}\times D_p)=5D_p\leqslant D_i=8D_p;$$

$$\forall 0<j,\ k<4,\quad |e_{i,j}-e_{i,k}|\leqslant 5.$$

3　基于差分进化的 Flash 文件系统垃圾回收算法(DEbGC)

目前主要的 GC 算法都采用贪心策略,在 Flash 空间使用率较高时会发生性能大幅下滑,下面以 YAFFS[9] 为例进行具体说明.YAFFS 的垃圾回收在创建或修改文件时发生,而且分为两种模式:当 Flash 空闲空间比较多的时候采用 Passive 模式,反之采用 Aggressive 模式.两种模式都采用贪心算法,从无效页面数不超过某阈值的擦除单元中选择无效页面数最多的(最脏的)擦除块,但是 Passive 模式的阈值要远低于 Aggressive 模式的阈值.这样虽然能保证当次 GC 代价最小(局部最优),但是那些含有较少无效页面的擦除单元无法在 Passive 模式中得到回收,等到 Aggressive 模式时才会被迫回收,而这些块的回收代价又很大,从而引发 GC 代价的急剧上升.一个典型的回收过程如图 2 所示,其中横坐标表

示 GC 的序列号,纵坐标表示每次 GC 中的页面拷贝次数.从图中我们可以看到,在空闲空间较小的情况下,垃圾回收会产生大量的页面拷贝,如图中横坐标 50000 左右.而且此时的垃圾回收操作每次需要复制的页面上也是递增的,甚至会出现需要复制目标块中所有页面的情况,这样的垃圾回收显然会导致写操作的性能急剧下降.

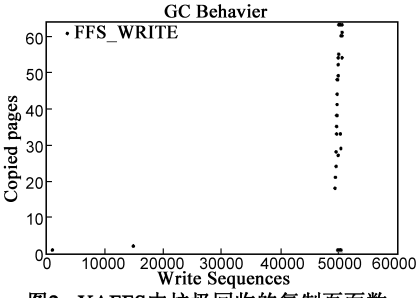

图2 YAFFS中垃圾回收的复制页面数

针对上述问题,结合最优化方法,我们引入基于群体搜索的差分进化算法提出了 DEbGC 算法,在整个搜索空间里进行迭代的启发式搜索,使在满足性能要求和损耗均衡的前提下,最大程度地减少擦除次数,从而延长介质的寿命.

3.1 编码机制与适应度评价

由于待回收的候选基本块数目不会太大,我们采用比较直观的编码方式,采用二进制 N 维解向量 S 作为解空间参数的遗传编码,$S_i=1$ 表示选中该物理块,$S_i=0$ 表示不选中该物理块.

使用指令实例编码时,对应的适应度评价函数如公式(1)所示.由于存在约束,经过交叉、变异的新个体,并不一定是包含在可行的解空间里.所以,在进行适应度评价时,我们需要根据约束来对新个体进行修复,使之能满足约束.如果采取简单的随机修复方式,算法会在整个搜索空间里面进行搜索,但是收敛比较慢,一般无法满足需求.为了加快收敛速度,我们采取贪心地启发式修复:先使用启发函数对所有候选块进行评估,优先修复高优先级候选块所对应的指令实例的染色体.为了减小修复时的复杂度,加快适应度评估,我们在编码之前就先对候选块进行启发式函数的评估,然后根据候选块所对应的权值从高到低顺序编码.这样处理之后,在适应度评估时仅需要按顺序从头开始对每一个染色体进行修复就能应用贪心的选择策略了.综上,个体评估算法的伪代码描述如算法 1 所示.

算法1 候选块个体适应度评价算法

```
ind.y=0;unmBIKs=0;
maseindexs=get_max_erased_index_set();// 初始化各变量
tmpid=ind;
//拷贝个体用于修复,不改变原有个体以保证群体多们性
for i =0 to nbin do
    if numBUcs≥ N then
        tmpib.x[i..nbin]=0;
        brek;// 回收块数目约束
    end
    if tmpind.x[i]= =0 then
        continue;// 不选中物理块
    end
    if i ∈ maxeinders then
        tmpind.x[i]=0;
        contiue;// 损耗均衡约束
    end
    if Candidates[i].p> D_l/D_p then
        tmpind.x[i]=0;
        continue;// 回收时间约束
    end
    D_t-= Candidates[i].p×D_p;// 选中物理块
    numBIks++;
    ind.y+=Candidates[i].e;// 累计擦除次数值
end
if ind.y<y-best then
    y_best =ind.y;
    bestind=tmpind;
    // 设成群体最优值和最优个体
end
```

3.2 算法描述

设编码长度为 N,群体规模为 M,差分进化垃圾回收算法 DEbGC 的形式化描述如算法 2 所示.其中的算法终止条件设置为个体适应度值连续几代不变或达到遗传代数限制.而遗传代数则是根据当前垃圾回收的触发时机来自适应的进行设置:如当前介质操作相对比较少的时候,遗传代数设置相应的增加;如果比较多的时候,则相应的减少.一种极端的情况就是,遗传代数退化为 1.此时,DEbGC 算法退化为保证性能和损坏均衡的随机选择算法.

算法 2 DEbGC 算法

步骤 1 初始化并评估群体 P_0,$t=0$.

步骤 2 离散差分产生下一代 Q_t,对于第 i 个个体 $Q_{t,i}$,产生方式具体如下:

$$Q_{t,i}(j)=\begin{cases}P_{t,k}(j)\ , & u\leqslant 0.5, k=rnd(1,M)\\ P_{t,i}(j)\ , & u>0.5\end{cases}$$

其中 u 是区间[0,1]上的均匀随机数,$P_{t,k}(j)$以概率 $5/N$ 进行变异,$1\leqslant j\leqslant N$.

步骤 3 评估群体 Q_t,计算个体的适应度值.

步骤 4 选择,取 P_t 和 Q_t 中对应位置适应度值小的个体进入下一代 P_{t+1}.

步骤 5 判断终止条件是否满足,若满足则结束.否则,$t=t+1$,转步骤 2.

3.3 算法复杂度分析

设候选块数目为 N,种群规模为 M.T_{GCDE}主要包括初始编码时间 $T_{DE-init}$和种群迭代进化时间 $T_{DE-iter}$.$T_{DE-init}$中主要进行了候选块权值计算,复杂度为 $O(N)$.假设差分进化算法进化了 I_{DE}代,种群中每个个体完成一次遗传操作所需的时间为 T_{DE-op},适应度计算时间 T_{DE-fit},那么种群迭代进化时间就是 $T_{DE-iter}\approx I_{DE}\times M\times(T_{DE-op}+T_{DE-fit})$.其中 T_{DE-op}的复杂度为 $O(N)$,个体适应度评价是时间复杂度为 $O(N)$.故 $T_{ISDE}\approx O(I_{DE}\times M\times N)$.其中 M 为常数,而 I_{DE}和算法收敛快慢已经当前应用特征有关,一般被限制在比较

小的范围内,此时的 T_{GCDE}时间复杂度约为$O(N)$.

4 实验结果

4.1 实验环境

为验证 DEbGC 算法的有效性和性能,本文对 YAFFS 垃圾回收算法进行了修改,并针对 3 种常用文件应用负载[10,11]进行测试.各种负载的特性参见表 1.实验的硬件平台是 Intel Core 2 Duo 2.3GHz,2G DDR2 400MHz SDRAM.Flash 芯片采用 Linux NANDSIM 进行模拟[12].

表 1　各种应用负载特性

负载名	描述	文件大小
DC (Digital Camera)	以写入、删除部分的递增方式填满 Flash,再全部删除	1MB ~ 2MB
MP3	填满 Flash,删除 50% 的文件,再填满,每次删除和填满过程为一次事务,执行指定个数次事务	4MB ~ 5MB
大小文件混合型	同 MP3	在 DC 和 MP3 中随机选择

4.2 结果分析

我们对 3 种不同的负载,使用不同的垃圾回收进行了实验,并记录了发生垃圾回收的时间以及垃圾回收时的写页面次数.为了方便比较,我们根据 NAND flash 的特性将时间约束转换成页面拷贝次数约束,所以 DEbGC 可配置的参数组合可以记作三元组 $<M, I, C>$,其中 M 为群体规模,I 为遗传代数,C 为页面拷贝次数约束.下文中的 DEbGC 配置参数皆为 $<25,10,32>$.

图 3 中给出了原有的贪心策略 MP3-Greedy, Mix-Greedy, Media-Greedy 以及使用了 DEbGC 算法的 MP3-DEbGC, Mix-DEbGC, Media-DEbGC 发生垃圾回收时的写页面次数分布.其中圆形的半径为页面写次数,圆周坐标为垃圾回收的序号(即第几次发生垃圾回收).从图中我们可以很清楚的看到,基于 DEbGC 算法的几条曲线的半径明显小于基于贪心策略的算法的曲线半径,也就是说基于 DEbGC 算法是垃圾回收发生时的写次数明显少于贪心策略的.同时,我们还可以看到,基于 DEbGC 算法的曲线中的点数明显少于贪心策略的,这就表明了使用 DEbGC 算法之后,系统发生的垃圾回收次数明显减少了.综上,从图 3 我们可以看出 DEbGC 算法明显的减少了对 Flash 介质的写入次数,能较好的延长系统的寿命.我们选取 Media 负载来进一步分析 DEbGC 算法与 Greedy 算法区别.如图 4 所示,4 条不同曲线代表不同算法、不同垃圾回收触发条件下,每次垃圾回收中的页面拷贝次数.gr.unlink(右下角)和 gr.write 曲线分别描述了 Greedy 算法下删除文件和写文件引发的垃圾回收情况,32.unlink(中间偏右下)和 32.write 曲线分别描述了 DEbGC 算法下删除文件和写文件引发的垃圾回收情况.图 4 中横坐标表示垃圾回收的写序号,即在上层应用对 Flash 文件系统第几次写时发生垃圾回收.

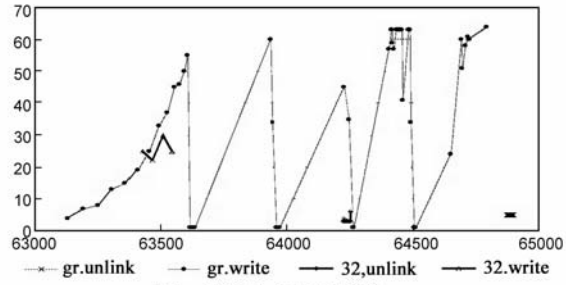

图4　算法选择结果比较

从图 4 中可以看出 DEbGC 中的垃圾回收次数以及页面拷贝个数都明显少于 Greedy 算法.而另一方面,gr.write 曲线中点数明显多于 gr.unlink 曲线,32.unlink 曲线中点数多余于 32.write 曲线;这说明 Greedy 算法中大部分垃圾回收是在写文件时发生,而 DEbGC 算法中则更多的是在文件删除时发生.由于文件删除操作相比写文件操作可以产生更多的垃圾擦除块,而且这些块

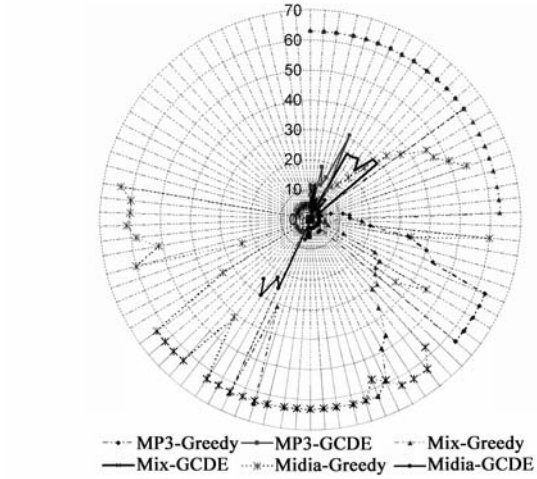

图3　GC的写次数分布

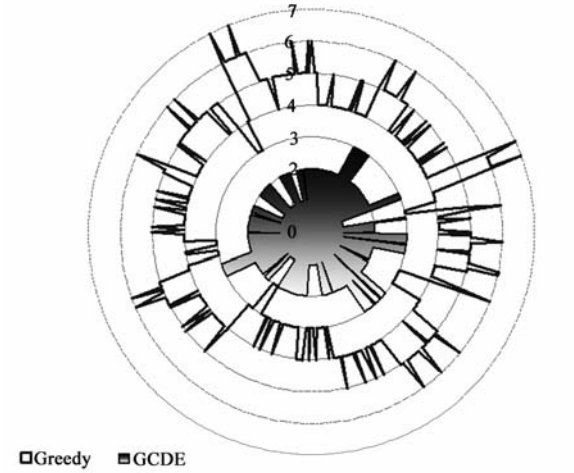

图5　Flash擦除次数比较

中的无效页面也会比较多;而垃圾回收时,需要拷贝的仅仅是有效页面,所以 DEbGC 算法在垃圾回收时产生的页面拷贝次数(即写页面次数)也就会比 Greedy 算法少.图 5 给出了算法执行结束时,Flash 介质上各基本块的擦除次数分布曲线.其中圆周表示基本块的编号,圆半径为该基本块的擦除次数.从图 5 中我们可以看到 Greedy 算法的曲线半径明显比 DEbGC 算法的曲线半径大,而且 Greedy 算法的曲线的毛刺(即擦除次数差别)也明显比 DEbGC 算法的曲线大.这说明,DEbGC 算法有效的减少了 Flash 介质的擦除次数,延长了寿命,而且使各基本块的擦除次数比较平均,也就是达到了更好的损耗均衡.

5 总结与展望

本文在对垃圾回收中候选块和目标块选择问题进行了合理的建模之后,引入了最优化方法中的差分进化算法,提出了基于差分进化算法的垃圾回收算法 DEbGC.实验数据表明,虽然算法引入了一定的时间开销,但是通过设置算法的参数,DEbGC 算法能够使得垃圾回收的代价均匀化,在给定的时间约束内完成垃圾回收,在保证性能和损耗均衡的前提下,有效的减少对 Flash 介质的擦除次数,延长 Flash 介质的寿命.该算法特别适用于常见的数码应用上,通过设置合适的参数,DEbGC 能够在保证性能,满足用户体验的前提下,更好的延长 Flash 介质的寿命.

参考文献:

[1] G Eran, T Sivan. Algorithms and data structures for flash memories[J]. ACM Computing Surveys, 2005, 37(2): 138 – 163.

[2] C Li-Pin, K Tei-Wei. Efficient management for large-scale flash-memory storage systems with resource conservation[J]. ACM Transactions on Storage, 2005, 1(4): 381 – 418.

[3] Sang Oh Park, S J Kim. An efficient multimedia file system for NAND flash memory storage[A]. International Conference on Consumer Electronics[C]. Las Vegas, NV, USA, IEEE Computer Society, 2009. 1 – 2.

[4] L Jongmin, K Sunghoon, K Hunki, H Choulseung, A Seongjun, C Jongmoo, L Donghee, H N Sam. Block recycling schemes and their cost-based optimization in nand flash memory based storage system[A]. Proceedings of the 7th ACM & IEEE international conference on Embedded software[C]. Salzburg, Austria, ACM, 2007. 174 – 182.

[5] A D Bennett, S A Gorobets, A Tomlin, C Schroter. Scheduling of housekeeping operations in flash memory systems[P]. US. 7565478, SanDisk Corporation (Milpitas, CA, US) 2009-7-21.

[6] B Seungjae, A Seongjun, C Jongmoo, L Donghee, H N Sam. Uniformity improving page allocation for flash memory file systems[A]. Proceedings of the 7th ACM & IEEE international conference on Embedded software[C]. Salzburg, Austria, ACM, 2007. 154 – 163.

[7] C Mei-Ling, C Chen-Lon, W Chun-Hung. A New FTL-based Flash Memory Management Scheme with Fast Cleaning Mechanism[A]. International Conference on Embedded Software and Systems[C]. Washington, DC, USA, IEEE Computer Society, 2008. 205 – 214.

[8] W Peng, Y Lihua, L Zhanzhan, X Xiaoyan. Flash memory management based on predicted data expiry-time in embedded real-time systems[A]. Proceedings of the 2008 ACM symposium on Applied computing[C]. Fortaleza, Ceara, Brazil, ACM, 2008. 1477 – 1481.

[9] Aleph1, YAFFS: Yet Another Flash Filing System[OL]. http://www.yaffs.net/, 2008

[10] H Jo, J U Kang, S Y Park, J S Kim, J Lee. FAB: Flash-aware buffer management policy for portable media players[J]. IEEE Trans. on Consumer Electronics, 2006, 52(2): 485 – 493.

[11] K Dongwon, J Dawoon, K Jeong-Uk, K Jin-Soo. μ-tree: an ordered index structure for NAND flash memory[A]. Proceedings of the 7th ACM & IEEE international conference on Embedded software[C]. Salzburg, Austria, ACM, 2007. 144 – 153.

[12] S Electronics. Samsung Semiconductor-Products-Flash-NAND Flash[OL]. http://www.samsung.com/global/business/semiconductor/products/flash/Products_NANDFlash.html, 2008.

作者简介:

时 正 男,1982 年出生,中国科学技术大学计算机科学与技术学院博士研究生,主要研究方向为嵌入式操作系统.
E-mail: shizheng@mail.ustc.edu.cn

纪金松 男,1981 年出生,中国科学技术大学计算机科学与技术学院博士研究生,主要研究方向为指令集扩展、可重构计算.
E-mail: jsji@mail.ustc.edu.cn

陈香兰(通信作者) 女,1977 年出生,中国科学技术大学计算机科学与技术学院博士研究生,讲师,主要研究方向为操作系统、异构计算. E-mail: xlanchen@ustc.edu.cn