

闪存数据库系统中一种高效的自适应存储模式

王立¹,王跃清¹,王翰虎^{1,2},陈梅¹

(1. 贵州大学 计算机科学与信息学院, 贵阳 550025; 2. 贵州星辰科技发展有限公司, 贵阳 550025)

(love-diablo@163.com)

摘要:使用闪存作为存储介质成为提高数据库系统性能的一条新途径,为了解决闪存数据库系统存储管理技术中基于日志的更新策略存在查询效率低、日志区空间分配不合理、索引更新代价高等问题,提出了基于 Bloom Filter 的最新版本预测算法,引入记录定位器结构,提出日志概要结构和基于闪存更新查询代价评估模型的自适应机制。实验证明,该方法能够自适应地划分合理的日志区空间,有效提高查询性能,减少各种非聚集索引的更新代价。

关键词:闪存;数据库;存储管理;自适应机制;代价评估

中图分类号: TP311.13 **文献标志码:** A

Efficient and self-adaptive storage schema in flash-based database system

WANG Li¹, WANG Yue-qing¹, WANG Han-hu^{1,2}, CHEN Mei¹

(1. School of Computer Science and Information, Guizhou University, Guiyang Guizhou 550025, China;

2. Guizhou XingChen Technology Development Company Limited, Guiyang Guizhou 550025, China)

Abstract: It has been a new approach that flash memory is used as storage media to improve database system performance. In the current flash database system, to solve the problems of low search performance, improper log region allocation, and high index update cost in storage management, a forecasting algorithm based on the latest version of Bloom Filter was proposed, record locator and log summary structure were introduced, and a self-adaptive mechanism based on flash search and update cost estimate model were given. The experimental results prove that it can make a proper log region allocation, efficiently improve search performance, and reduce non-clustered index update cost.

Key words: flash; database; storage management; self-adaptive mechanism; cost estimate

0 引言

闪存具有读写速率快、节能、抗震等特性。近年来有研究者开始尝试使用闪存来提高数据库系统性能。与磁盘相比,闪存读比写快,无覆盖写,块擦除粒度、擦除代价很高,这些特性使磁盘的存储管理技术不适用于闪存。

闪存专用存储管理技术的关键在于提高数据的更新性能。Intel 提出了闪存转换层(Flash Translation Layer, FTL)方法^[1],近几年研究者提出了改进的 FTL 方法,如 NFTL^[2]、FAST^[3]等。这几类 FTL 方法通过将更新的页写入新页的方式避免了擦除操作,但对数据库中面向记录的细粒度的随机更新效率很低。日志更新方法将对记录的更新转化为日志记录,从而以写代替擦除操作。此类方法有 IPL 方法^[4]、动态日志法^[5]、OPL 方法^[6]等。这几类方法存在读取开销大、更新性能低、空间划分不合理,以及不能适应更新频率的变化等问题。

本文从提高日志更新方法的数据查询性能、动态合理地划分块内日志区的角度出发,提出了基于 Bloom Filter^[7] 的最新版本预测算法,在数据区中引入了记录定位器结构,在日志区提出了日志概要结构,并提出了基于闪存更新查询代价评估模型的自适应机制。实验证明本文提出的存储模式能够提高查询效率,自适应地、合理地划分日志区空间,提高不同环

境下的更新性能,减少多种非聚集索引的更新代价。

1 闪存的特点及相关工作

闪存的基本读写单位是页,典型的读写延迟分别为 40 μ s 和 200 μ s,而磁盘平均读写延迟为 10 ms ~ 30 ms。闪存页中一旦写入数据就变成只读页,除非进行擦除操作,否则其中的数据无法更改。擦除的基本单位是块,通常由 64 或 128 个页组成,块的擦除时间很长,通常需要 1.5 ms,每个块的擦除次数有限,大约为 1 万次^[8]。综上所述闪存具有无替换更新、块擦除粒度、擦除代价高的缺陷。

为此有研究者提出了 FTL 方法^[1-3],该方法在更新页时,不进行擦除操作,而是将该页更新后的数据写入新页,并更新页地址映射表,从而屏蔽了闪存无替换更新的特性。然而数据库中频繁进行着细粒度级别的更新操作,这些操作通过 FTL 会频繁且大量地消耗空闲页、产生废弃页,更新性能十分有限,空间回收过程复杂而且代价很高。

日志更新策略^[4-5]是对 FTL 的改进,其主要思想是将闪存块划分为数据区和日志区。当对记录执行更新操作时,并不对该记录进行替换写,而是向该块的日志区内写入更新日志,从而将替换更新转化为写操作。在查询时,获得目标记录后,还需要访问日志区以确认该记录是否存在更新日志;如果不存在,则该记录就是最新版本;如果存在,则需要根据更新

收稿日期:2010-11-23;修回日期:2011-01-20。

基金项目:贵阳市科技攻关项目([2010]筑科工合同字第 28 号);贵州大学 2010 年研究生创新基金资助项目(校研理工[2010033])。

作者简介:王立(1985-),男,陕西西安人,硕士研究生,CCF 会员,主要研究方向:闪存数据库系统;王跃清(1985-),男,内蒙古呼和浩特人,硕士研究生,CCF 会员,主要研究方向:闪存数据库系统;王翰虎(1946-),男,贵州遵义人,教授,CCF 高级会员,主要研究方向:分布式系统、数据库系统;陈梅(1964-),女,贵州贵阳人,教授,主要研究方向:数据库、软件工程。

日志来计算该记录的最新版本。随着更新操作的执行,日志区会不断地写入更新日志,当日志区满时,对该块进行合并操作:将所有记录和对应的更新日志合并,得到最新版本的记录,再将原块擦除,将数据写入。

然而现有的日志更新策略依然有许多需要改进的地方,本文从3个方面出发改进现有的更新策略:

1) 现有的策略在访问日志区前,缺少判断记录是否存在更新日志的机制,从而导致不存在更新日志的记录也需要访问日志区,为解决此缺陷本文提出记录最新版本预测(Record Latest Version Forecasting, RLVF)算法。

2) 在日志区的扫描过程中,现有的方法必须扫描整个日志区,当日志区较大时,读取代价明显增大,因此本文提出了记录定位器和日志概要结构以降低日志区较大时的扫描代价。

3) 现有方法日志区只有一个页或者缺少指导日志区划分的策略,在更新频率较高和不断变化时效率很低,为此本文提出了闪存性能评估模型来指导日志区自适应的划分。

2 高读写性能的存储模式

2.1 动态日志更新策略

本文采用日志更新策略与 IPL 方法日志区只有一页不同,本方法日志区的划分是动态的,如图 1(a) 所示。逻辑映射结构如图 1(b) 所示,块逻辑地址映射表实现了块逻辑地址到块物理地址的映射,使得合并操作后记录的逻辑块地址不变。

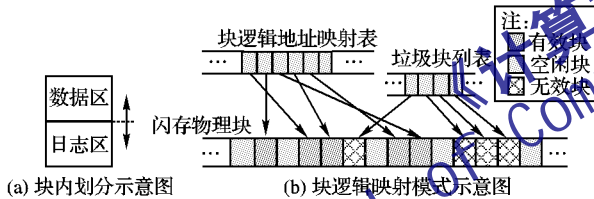


图1 动态日志更新策略总体结构

与 IPL 相比,动态日志策略更灵活,可以适应不同的更新操作频率。与 OPL 相比,该方法在块中划分日志区,从而限定记录对应的日志区范围,避免了 OPL 方法统一管理日志的复杂度,减少了查找日志记录时的搜索范围。

2.2 记录最新版本预测方法

为避免 IPL、OPL 等方法读取记录时不必要的日志区扫描操作,提高查询效率,本文提出了基于 Bloom Filter^[7] 的 RLVF 算法,可以在扫描日志区之前预测记录是否为最新版本。存在相应更新日志的记录被称为“脏”记录,不存在相应更新日志的记录被称为最新版本记录。RLVF 算法的核心数据结构是为每一个块在内存中划分一个长度为 m 的二进制序列 T 和 k 个相互独立的哈希函数 h_0, h_1, \dots, h_{k-1} , 每个哈希函数都能够将记录的主键均匀映射到 $[0, m-1]$ 。对于任何一条记录 x , 都对应 k 个二进制位, 分别为: $T[h_0(x)], T[h_1(x)], \dots, T[h_{k-1}(x)]$ 。在日志区为空的时候, T 全为 0。当更新记录 x 时, 向日志区写入更新日志的同时, 将 x 所对应的 k 个二进制位置 1。当读取记录 x 时, 查找 x 所对应的 k 个二进制位, 如果都为 1, 则认为 x 是“脏”记录, 进而搜索日志区以获得最新版本; 如果不全为 1, 则认为 x 是最新版本, 无需搜索日志区。

显然, RLVF 无误报, 即不会将“脏”记录误认为最新版本

的记录。RLVF 算法存在一定的漏报概率, 设已经更新了 n 条记录, 则 T 中第 i 位为 0 的概率为 P_{Ti} , RLVF 将最新版本误认为“脏”记录的概率(漏报概率)为 $P_{fn} \circ P_{Ti}$ 和 P_{fn} 的公式如下:

$$P_{Ti} = (1 - 1/m)^{kn}$$

$$P_{fn} = (1 - P_{Ti})^k = (1 - (1 - 1/m)^{kn})^k \approx (1 - e^{-kn/m})^k$$

其中: m/n 的值越大, P_{fn} 越小, 内存消耗越大。 m/n 超过一定值后, 再增大对 P_{fn} 影响很小。 k 值太大计算复杂度太高, 值太小影响预测精度。本文认为在保证 $P_{fn} > 0.99$ 的情况下, 兼顾预测精度、空间消耗和计算复杂度的一组的 m/n 和 k 的取值分别为 10 和 5。

利用 RLVF 算法的查询过程如下: 找到记录时, 利用 RLVF 算法预测该记录是否为最新版本, 如果是最新版本, 则不必访问日志区; 如果预测为“脏”数据, 则访问日志区以获得最新版本。由于 RLVF 算法漏报概率为 0.01, 在查询中 99% 的最新版本都会被 RLVF 算法正确预测而免去对日志区的访问, 使得查询效率大幅度提高。

2.3 记录定位器与日志概要结构

RLVF 算法提高了最新版本记录的读取效率, 但对于“脏”记录和 RLVF 算法漏报的记录来说, 依然需要扫描整个日志区以获得最新版本。为了减少扫描日志区过程中的页访问量, 本文引入了记录定位器, 提出了日志概要结构。

闪存无替换更新以及 NAND 不支持块内的随机页写入^[8]的特性使得基于块的磁盘记录定位器结构^[9]不能在闪存中使用。为此本文提出了以页为单位的定位器结构, 如图 2(a) 所示。页中的每一条数据记录都有一个记录定位器与之对应, 记录定位器中存储该记录的页内偏移地址和该定位器的块内唯一编号, 如图 2(b) 所示。

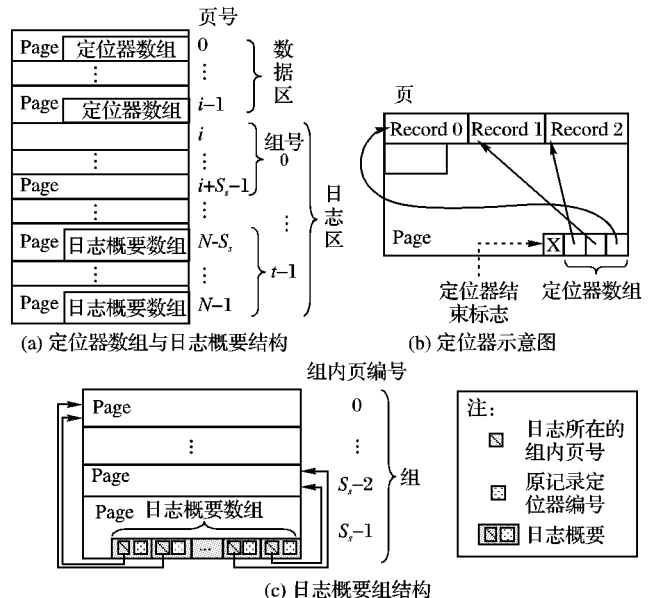


图2 记录定位器和日志概要数组结构示意图

日志概要结构将由 L_p 个页组成的日志区每 S_p 个页划分为一个概要组, 总共可以分 $t = L_p/S_p$ 个概要组。如图 2(c) 所示, 在每个概要组的最后一个页的尾部开辟用来存储日志概要数组的空间, 每条日志概要对应应该组中的一条更新日志。日志概要由所对应的更新日志所在的组内页号和更新日志对应的原数据记录的定位器编号组成。

日志概要的维护过程很简单: 写入更新日志的同时向该

组日志概要中添加与之对应的日志概要,直至该块进行合并操作时随更新日志一起删除。

通过日志概要结构获取记录最新版本的过程如下:读取日志区的所有日志概要,从中找到对应的日志概要。如果不存在,则说明没有相应的日志,该记录就是最新版本;如果存在,则根据该日志概要中包含的更新日志组内页号对该页进行访问,得到该记录的最新版本。日志概要结构使得读取所有日志概要后最多再需要一次页访问就可得到目标更新日志。这样通过日志概要结构获取记录最新版本而进行的页访问上限为 $IO'_{Read}(L_p) = \|L_p/S_i\| + 1$ (对 x 上取整),与不采取日志概要的 L_p 次页访问量相比明显下降。

2.4 降低索引更新代价

记录定位器编号在块内唯一,索引使用定位器编号和块逻辑地址作为引用记录的唯一标识和存取路径,从而使得记录更新导致的块内偏移地址的改变对索引透明,降低了索引的更新代价。

3 闪存更新查询代价评估模型及自适应机制

为了兼顾查询和更新效率,合理划分日志区大小,本文提出闪存的更新查询代价(Update and Search Cost, USC)模型,并说明它如何实现自适应机制。所要用到的符号及含义如表1所示。

表1 符号及含义

符号	含义	符号	含义
r	读操作频率	a	页读取时间代价
u	更新操作频率	b	页写入时间代价
S_L	日志区空间(单位:页)	N	一个内存块中的页数
N_{LR}	页的更新日志容量	β	空间代价权重
S_s	日志区组大小(单位:页)	η	宽容度系数

更新查询时间代价(Update & Search Time Cost, USTC)分为查询时间代价和更新时间代价两个部分,两者按照出现的频率加权求和得到 USTC:

$$USTC = \frac{r}{r+u} \cdot \left[\frac{S_L}{N-S_L} \left(\left\| \frac{S_L}{S_s} \right\| + 2 \right) \cdot a + \frac{N-2S_L}{N-S_L} \cdot a \right] + \frac{u}{r+u} \cdot \left[a + b + \frac{MTC}{S_L \cdot N_{LR}} \right]$$

由于 $P_{fn} < 0.01$, 可以近似认为所有的最新版本记录都被正确预测,因此前半部分公式包括“脏”记录的 $\|L_p/S_i\| + 2$ 次页访问时间代价和最新版本记录的1次页访问时间代价,二者按照出现的频率加权求和。后半部分公式包括读取要更新的记录的1次页读取操作、写入更新日志的一次页写入操作,以及按照一条更新日志占日志区长度比率加权的合并操作时间代价(Merge Time Cost, MTC)。MTC为:

$$MTC = N \cdot a + (N - S_L) \cdot b$$

由读取整块所有页时间代价、进行合并计算的时间代价以及在新块中写入数据区的写入代价,合并计算在内存中完成,可忽略不计,在新块中写入避免了擦除时间代价。

USC 综合考虑了查询更新时间代价和日志区空间代价,其公式为:

$$USC = USTC + LSC = USTC + (\beta \cdot USTC) \cdot S_L$$

S_L 按照 USTC 的 β 倍加权得到 LSC。增大 β 可以提高空间利用率,但牺牲了一定的读写性能,本文认为 β 合理的取值区

间是 $[0.005, 0.02]$ 。

计算 USC 最优值的过程中遇到一元四次方程求解,计算复杂度很高。为了解决此问题本文提出了算法1,其策略是:计算相邻的几个 S_L 取值下的 USC 值,从中选择更为合理的 S_L 取值,从而避免一元四次方程的求解。此外,日志区大小的调整造成的记录移入和移出会带来额外消耗,为此算法1包含宽容度系数 η ,如果 $USC(S_L')$ 比起 $USC(S_L)$ 提升 η 以上,才认为 S_L 需要调整, η 一般取 0.02。

算法1 最优 USC 求解算法。

输入 USC 中需要的各种参数、 S_L 、 η 。

输出 USC 模型认为合理的 S_L' 值。

```

1)  $S_L' = S_L$ 
2) while (  $USC(S_L' - 1) < USC(S_L')$  ) //向左搜索最优值
   {  $S_L' = S_L' - 1$ ; }
3) while (  $USC(S_L' + 1) < USC(S_L')$  ) //向右搜索最优值
   {  $S_L' = S_L' + 1$ ; }
4) If (  $USC(S_L) - USC(S_L') > \eta * USC(S_L)$  )
   //判断是否应该进行调整
   return  $S_L'$ ; else return  $S_L$ ;

```

利用 USC 模型实现日志区自适应划分的过程如下:在创建新块到日志区满的过程中统计该块的更新操作频率 u 和读操作的频率 r , 利用算法1 得出新的 S_L 值,如果 S_L 值改变则进行日志区大小调整。调整策略如下:如果需要减小日志区,该块进行合并操作后进入允许插入记录的状态,即允许一定量的来自其他块或新产生的记录以更新日志的方式写入该块,这些更新日志将在下次合并操作后成为数据区的记录,达到增加数据区、减少日志区的目的;如果需要增大日志区,若有其他块处于允许插入记录状态,则将需要移出的记录插入到这些块中,如果依然有需要移出的记录或者不存在允许插入记录状态的块,则创建新块,将这些数据记录移入。

4 实验结果与分析

本章通过实验验证提出的存储模式的性能。实验中的操作系统为 Windows XP, 编程环境是 VC 2008。使用的 PC 硬件配置为: Intel 2.8 GHz CPU, 2 GB 内存, 硬盘为 Intel X25-V SSD, 使用 SATA II 接口。SSD 中页的大小为 4 KB, 块由 128 个页组成, 芯片的读、写、擦除延迟分别为 40 μ s、200 μ s 和 1500 μ s。测试中的数据源来自 TPC-C 基准测试, 它模拟了 OLTP 应用环境, 既有对记录分散的读写, 又有对记录集中的读写, 可以考察存储模式的自适应性。通过改变 TPC-C 的参数以获得不同的查询/更新频率的 I/O 操作队列作为实验的数据源, 平均更新日志长度为 100 B。

在 S_L 取值不同的情况下分别进行了 50 万次记录读写操作, 跟踪查询过程中 RLVF 算法的漏报频率, 实验结果见图 3(a)。这是大量数据测试的结果, 从而验证了 RLVF 算法漏报概率 P_{fn} 在 1% 左右。

以 IPL 和 FTL 作为对照, 在设定了不同的更新比率的情况下进行了 50 万次读写操作, 跟踪其更新性能, 实验结果见图 3(b)。实验结果表明在各种更新频率情况下 FTL、IPL 平均更新时间代价是分别为本文方法的 3 倍和 5 倍。这是因为在更新过程中本文采用的动态日志区的划分策略选择了更为合理的日志区大小, 减小了合并操作频率, 提高了更新性能。

以 20% 的更新频率在不同的 S_L 取值下分别进行 50 万次

读写操作,并跟踪读取性能,实验结果见图3(c),表明FTL不存在日志区,只需要读取一个页,因而读取代价最高;IPL方法每次读取记录必须额外访问一次日志页,所以平均读取代价约为两个读页时间;动态日志方法引入RLVF算法后避免了最新版本记录多余的日志区扫描,读取代价在 S_L 较小时明显低于IPL,但RLVF算法不能减少读取“脏”记录的页访问量,随着 S_L 的增大,读取代价增长加速;采用记录定位器和日志概要结构,“脏”数据的页读取量降为 $\|S_L/S_s\| + 2$,从而使得读取代价始终略高于FTL。

图3(d)显示在不同的更新频率下分别进行50万次读写操作的平均时间代价。实验结果表明随着更新频率的增长,FTL由于对更新操作处理不当导致平均时间代价严重增长,IPL方法的平均读写代价虽比FTL明显减小,但随更新频率

的增加平均时间代价增大;引入RLVF算法的IPL在更新频率较低时性能改善明显,但代价依然随着更新比率的增加而增长明显;本文提出的方法可以有效地减少平均读写代价,其优势在更新频率比较高的情况下更为明显,这直接得益于RLVF算法、记录定位器与日志概要结构对读取性能的帮助以及USC模型对日志区大小的合理划分。

图3(e)显示了在不同日志区划分的情况下分别进行50万次记录读写操作的平均时间代价,实验中分别设定不同的 S_L 以不同的更新比率做了3组测试,以反映 S_L/β 的选取对平均读写时间代价的影响。实验结果表明USC模型的理论最优值与实际最优值十分接近,同时考虑了空间消耗。在对整体性能影响很小的原则下,USC模型倾向于节约空间的划分策略,参数 β 决定了对节省空间消耗策略的倾向度。

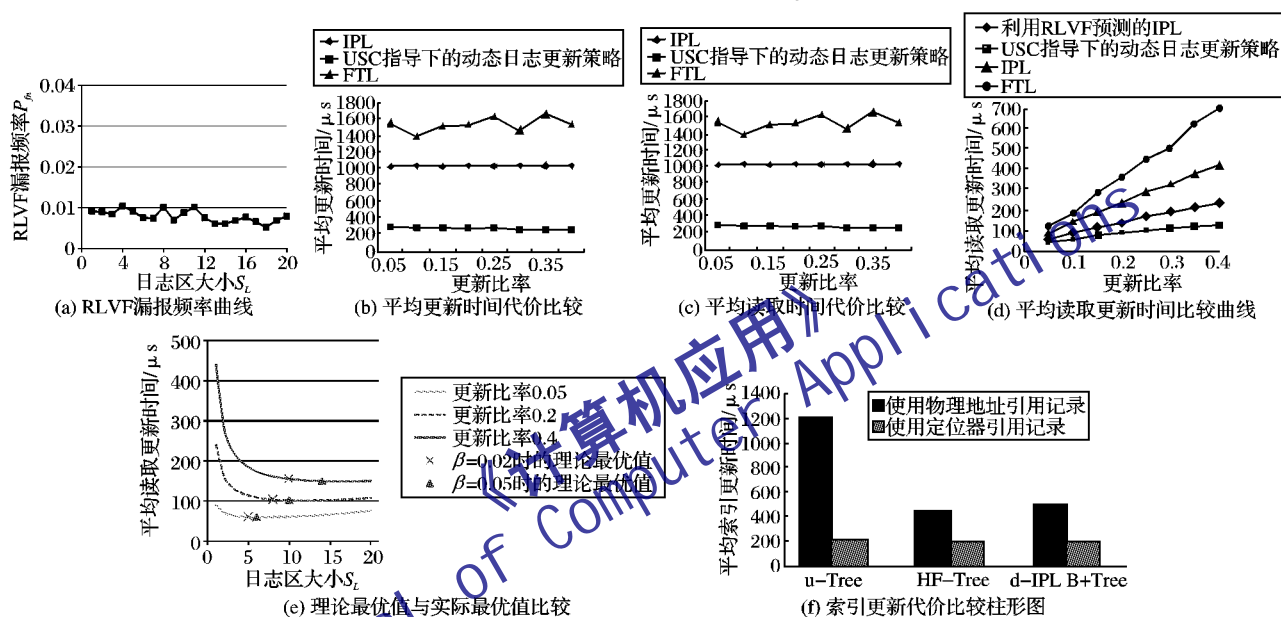


图3 实验结果

本文通过编程实现了u-Tree、HF-Tree和d-IPL B+Tree共3种索引,并跟踪3种索引分别进行50次记录读写操作过程中的索引更新代价,实验结果如图3(f)所示,表明采用记录定位器作为引用记录的指针可以大幅度减少索引的更新代价。这是因为记录的块内更新不会更改定位器编号,使得块内的异地更新对索引透明。

5 结语

闪存比起磁盘虽有众多优势,然而不合理回避闪存的缺陷,就无法高效利用闪存。本文提出的自适应存储模式中包含RLVF算法、记录定位器和日志概要结构以及基于闪存更新查询代价模型的自适应机制。实验证明该存储模式大幅度提高了日志区较大时的读取性能,有效减少高更新频率下的更新代价,并自适应地动态划分日志区大小。下一步工作将对闪存更新查询代价模型进行完善,进一步提高其在各种情况下的准确性。

参考文献:

- [1] Intel. Understanding the Flash Translation Layer (FTL) specification [EB/OL]. [2010-08-21]. http://www.cse.usf.edu/~yjrobin/reading_list.html.
- [2] KIM J, KIM J M, NOH S H. A space-efficient flash translation lay-

er for CompactFlash systems[J]. IEEE Transactions on Consumer Electronics, 2002, 48(2): 366-375.

- [3] LEE S W, PARK D J, CHUNG T S. A log buffer-based flash translation layer using fully-associative sector translation [J]. ACM Transactions on Embedded Computing System, 2007, 6(3): 18.
- [4] LEE S-W, MOON B. Design of flash-based DBMS: An in-page logging approach [C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. New York: ACM, 2007: 55-66.
- [5] DU MING, ZHAO YAN, LE JIAJIN. A dynamic logging approach on flash-based DBMS [C]// International Seminar on Business and Information Management. Piscataway: IEEE Computer Society, 2008: 37-40.
- [6] 岳丽华, 向小岩, 金培权, 等. 基于分离日志的闪存数据库系统存储管理方法[J]. 中国科学技术大学学报, 2010, 40(5): 526-532.
- [7] MITZENMACHER M. Compressed bloom filters [J]. IEEE/ACM Transactions on Networking, 2002, 10(5): 604-612.
- [8] ELECTRONICS S. K9F8G08B0M. 2G x 8 Bit NAND Flash Memory [EB/OL]. [2010-06-28]. http://www.datasheet4u.net/html/K/9/F/K9F8G08UXM_SamsungElectronics.pdf.
- [9] JOHNSON L J. 数据库: 模型、语言与设计[M]. 李天柱, 译. 北京: 电子工业出版社, 2004: 391-394.