文章编号：2095-2163(2021)12-0201-08 中图分类号：TP391.41 文献标志码：A

闪存数据管理研究综述

陈玉标，李建中，高 宏
（哈尔滨工业大学 海量数据计算研究中心，哈尔滨 150001）

摘 要：闪存固态盘逐渐取代磁盘成为主流存储，由于存储性质相异，因而基于磁盘的数据管理技术在闪存设备上很难发挥闪存设备的特性。因而需要对数据管理的各个子模块，针对闪存环境设计专门的优化算法。本文对目前闪存数据管理的各个研究领域进行整理和总结，让读者了解目前基于闪存特点的优化算法主要思路和方法，以及存在的不足缺点。最终的目的是让读者对整个研究领域的主要脉络有比较精准的把握。

关键词：闪存；固态盘；磁盘；数据管理

A survey of research on flash memory data management

CHEN Yubiao, LI Jianzhong, Gao Hong
(Massive Data Computing Research Center, Harbin Institute of Technology, Harbin 150001, China)

【Abstract】 Flash-based solid state drive is gradually replacing disk as mainstream storage. Due to different storage properties compared with magnetic disk, it is difficult for data management technologies optimized for magnetic disk to work well on the flash devices. Therefore, it is necessary to design specific optimization algorithms for each sub module of data management systems on flash memory. In this paper, various research fields are listed out and summarized, so that readers can understand the main ideas and methods of the optimization algorithms working on flash memory, as well as the existing shortcomings. The final purpose is that readers can have a more accurate grasp of the main context of the whole research field.

【Key words】 flash; solid state drives; magnetic disk; data management

0 引 言

近些年，处理器和传统磁盘之间的性能代沟越来越大，外存储器的性能成为很多外存索引和算法的瓶颈。加之大数据时代的到来，使得该瓶颈缺点愈加明显。随着闪存和固态盘的出现，大大缩减处理器和外存存储器之间的代沟。由于闪存和固态盘的存储机理完全不同于磁盘，很多基于磁盘的优化研究不再有效。因而，近年来闪存存储设备已经引起大家强烈的兴趣和广泛研究，基于闪存和固态盘的数据管理也成为一个热门的研究领域。

早些年，数据管理系统中的索引和算法主要是针对磁盘的 I/O 特性进行设计和优化。由于闪存介质和磁盘存储的原理完全不同，因而拥有完全不同的 I/O 特性。磁盘又叫机械硬盘，其以磁作为存储介质，因而包含读写磁介质的磁头和驱动磁头的转动装置。当读写数据时，需要先将磁头指向读写的目标地址，这个过程叫做寻道，即通过转动盘片和移动磁头来定位数据在磁盘中的位置。然后，读取该位置的数据或将数据写进该位置。根据上述磁盘读写的过程很容易看出，磁盘读写对于顺序读写性能会比较好。当遇到随机读写时，大量的时间花费在寻道过程中，从而导致随机读写带宽极剧下降。闪存作为一种半导体电子设备，其内部不包含任何机械部件，完全靠电路控制来进行数据读写操作，避免了磁盘的缺点。因而，之前基于磁盘 I/O 特点做的优化工作不再有效，在闪存固态盘上需要被重新设计和研究。相对于磁盘，闪存拥有以下特点：

(1)读写延迟低：闪存是纯粹的电子器件，内部所有操作都是由电路实现。因而闪存读写延迟很低，且读写延迟和寻址的位置无关。因而，对于闪存没有缓存的芯片，顺序读写和随机读写延迟相同。

(2)读写单元页：页是闪存芯片操作的最小物理读写单元。所有上层的大块读写最终在闪存芯片层面都会分解成页物理单元进行操作。闪存内一个块中包含很多页，但页的写并不是随意的。已经写过的页是不能进行写的，写前必须进行擦除操作；有些 MLC、TLC、QLC 多层结构的闪存芯片，要求上层

基金项目：国家自然科学基金(61832003)。

作者简介：陈玉标(1991-)，男，博士研究生，主要研究方向：闪存数据库管理；李建中(1950-)，男，教授，博士生导师，主要研究方向：大数据计算理论和管理；高 宏(1966-)，女，博士，教授，博士生导师，主要研究方向：传感网和大数据管理。

收稿日期：2021-04-06

页写过之后,才能写下层页;而另外一些闪存芯片写块中,必须前序的页都写过才能进行下一个位置页的写入等。当然,由于真实的闪存设备上层都会配置一个 FTL(闪存翻译层),通过异步读写,重新映射的方法,可以完全对用户隐藏这些限制。

(3)读写不对称:闪存设备的存储单元的写操作采用 ISPP 技术,不断对存储单元进行加电压来进行充电,直到达到指定电位。因而,该过程很费时。而读过程,只需要获取存储单元的电位大小即可,因而很迅速。因而,闪存的读操作和写操作之间的延迟相差会比较大,这就是闪存的读写不对称。

(4)写前擦除机制:由于闪存存储单元的写操作是一个充电过程,且充的是电子。因而,即将被写的存储单元为了确保写入成功,在此之前必须释放所有电子。闪存引入一个批量的放电操作,叫做擦除。擦除的延迟很高,如果按照磁盘的更新方式,对所有数据进行“就地更新”,那么就会对已经写过的数据页进行写。由于写前擦除机制,延迟会非常大。为了优化写操作,闪存引入“异地更新”策略。即每次对数据页进行修改重写,都会寻找一个新的空白页进行写入操作。其会在闪存上设计一个闪存翻译层(FTL),实时保持虚拟地址到物理地址的映射。这样看来,对于上层用户来说,写的还是同一虚拟地址页,但实际写的是不同的物理地址。

(5)擦除粒度大:闪存为擦除的单元称为块,块要比页大很多。如图 1 所示,1 块(block)包含 256 页(page)。即这里擦除一块就会对块内 256 页进行擦除。擦除之后的页都是待写页。如果块内已经被写页,又有写操作需求,就还需要重新对整个块进行擦除。每个块都有自己的寿命,擦除次数是有限制的。超过次数限制的块,块内数据就会降低安全性。

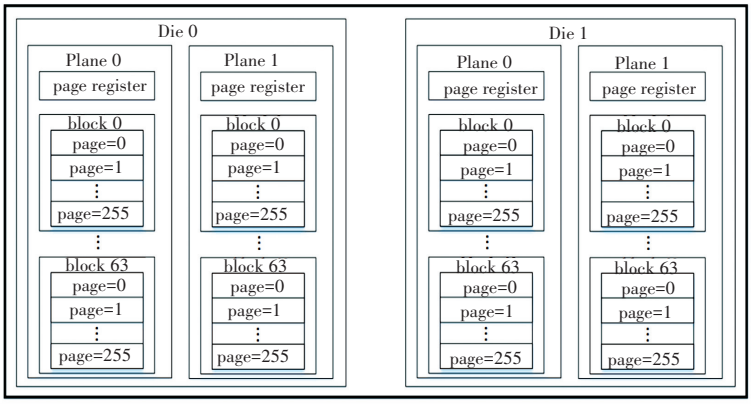

图 1 闪存包结构图

Fig. 1 The architecture of flash package

闪存的特点中有些是优点,有些是缺点和限制。目前,基于闪存的数据管理技术已经取得了不错的进展,其整体的思路都是想办法去发挥闪存的优势(如随机读),尽量避免其缺点(如时间成本高和寿命缩减的擦除操作)。为此本文将按照闪存上数据管理各个研究方向分别进行归纳整理。

1 闪存存储系统

大数据时代到来,对于存储系统的要求越来越高。对于这些存储系统,性能和耗能这两项指标都至关重要。由于闪存和固态盘性能高且耗能低,因而被考虑应用于存储系统平台。FAWN[1] 是一个 key-value 存储系统,其用一些低能耗的处理器和低能耗的闪存设备搭配,来平衡计算和 I/O 能力,使得整体分布式数据处理平台耗能变低。内存很珍贵,且成本耗能非常高,于是在 Google Fusion Tables[2] 系统中,动态配置固态盘作为内存的拓展。为了更充分地开发闪存存储的性能,Gordon[3] 实现了一种新的 flash 翻译层,其能针对数据密集型工作负载和大型闪存存储数组,充分发挥闪存性能。这些存储系统平台都是直接利用闪存低能耗和高读写带宽的特点直接对其进行优化。

2 闪存文件系统

文件系统作为上层应用和底层存储之间的关键环节,对于其优化具有重要意义。早前,有放弃索引结构的极端方式。如,Pilot[4] 直接通过一个 64 位的 UID 来进行数据访问和定位,其明显的缺点就是对于代价更高的写操作表现会非常差。

2.1 日志结构

主流文件系统,从磁盘开始主要采用日志结构。日志结构的文件系统主要分为两种类型:一种是日

记方式[5-8]，另外一种是日志方式[9]。日志结构的文件系统对于随机写性能表现良好，但随机写在闪存存储器上依然是个痛点，因而基于日志结构的闪存文件系统也应运而生。其中基于日记方式的文件系统有 JFTL[10] 和 JFFS[11]。JFFS 直接在闪存上根据闪存特性，建立最简单的日记系统；而 JFTL 将日记系统建立在 FTL 上，为保证数据一致性，将会同时修改在闪存日记区域和数据区域记录。由于闪存块的寿命有限，坏块对于闪存的性能和可靠性影响比较大，因而如何去保证闪存块内的磨损均衡至关重要。SFS[12] 通过记录块的使用情况，并采用对冷热数据分组的办法来实现磨损均衡的目的。F2FS[13] 虽也采用冷热数据分组的方式，但其会根据文件和数据的类型来进行分组，以避免粒度小、频繁而代价高昂的元数据写操作。

2.2　对象存储和其它

除此之外，考虑数据特点，为了解决和处理数据的离散单元问题，基于对象存储的闪存文件系统的 OFSS[14] 应运而生，其主要考虑将存储数据的结构和闪存存储器内部结构相适应。考虑到闪存读写不对称特点，ReconFS[15] 尝试去设计持久性的目录树来平衡读写的性能。针对固态盘内部并行性，ParaFS[16] 在固态盘内部各个环节，利用加速内部并行性来提高文件系统性能。

2.3　开源项目

目前，最主流的开源项目是 Yaffs[17] 和 JFFS[11]，被广泛应用和采纳的文件系统，都已被闪嵌入到 Linux 内核中，对于闪存文件系统的开发学习具有重要推动作用。

3　闪存转换层

闪存翻译层（FTL）是固态盘内部最核心的组件，主要负责逻辑地址到物理地址的映射、垃圾回收、磨损均衡、断电恢复等。通过实现上述功能，其隐藏了固态盘内部的结构，让用户可以像磁盘一样使用固态盘。由于 FTL 的表现直接影响着整个固态盘的数据和性能，因此 FTL 作为固态盘企业的核心资产，属于商业机密，但在学术界对 FTL 进行了详细的研究。

3.1　同质固态盘 FT

按照映射单元，可以将 FTL 分为：页级别 FTL、块级别 FTL、混合 FTL 和变长映射 FTL 4 种类型。页级别 FTL 能将请求的逻辑页面映射到闪存空间中的任何物理页面。因此，该映射机制很灵活，具有很高的闪存页面利用率。其代表性工作在文献[18-20]中有所体现。块级别 FTL 是将请求的逻辑块映射到闪存空间中的任意物理块。页寻址是块地址加上偏移量来获得，这样做的优点是映射表会小很多，占用更少的内存，减少固态盘的制作成本；缺点是会造成闪存空间的浪费，以及垃圾回收代价增加。其代表性工作在文献[21-22]中有所体现。混合的 FTL，结合页级别映射和块级别映射的优点，使得 FTL 即具备页级别映射灵活性的特点，又具备块级别映射表小的优点。实现方式是将固态盘内的块分为数据块和日志块两类，数据块采用块映射保存数据，日志块采用页映射保存更新信息。由于日志块占整体固态盘的比例很低，因而页映射表很小。LAST[23]、BAST[24]、A-SAST[25]、SAST[26]、SuperBlock-FTL[27] 均采用混合 FTL 设计。

无论页级别映射或块级别映射，均是定长映射。而对于多媒体场景，大部分访问是大批量顺序写操作，定长映射会带来映射表的冗余。因而文献[28]提出一种变长的映射机制，来实现映射粒度的动态调整。这种映射的缺点，就是 FTL 的逻辑地址到物理地址翻译速度会很慢。

3.2　混合固态盘 FTL

一般情况下，默认固态盘内部闪存介质是单一的。例如：闪存均采用 MLC 固态盘，面向同类型闪存的 FTL 统称为同质固态盘 FTL。SLC 与 MLC、TLC、QLC 相比，优点是读写速度快、能耗低，缺点是数据密度低和单价高。为了合理的利用其特点，对应多类型闪存介质的 FTL 称为混合固态盘 FTL。实际上 MLC、TLC 和 QLC 可以通过更改设置让其当作 SLC 使用。因而，混合固态盘可以分为两种类型，一种称为硬划分固态盘，即固态盘内部同时包含 SLC 和其它多层闪存介质；另外一种称为软划分固态盘，即虽然固态盘内部只有一种闪存介质，但是可以通过更改设置，让固态盘内部的多层闪存的一部分当作 SLC 使用。硬化分的 FTL 算法有 CFTL[29]、ComboFTL[30] 和 DPAFTL[31] 等。CFTL、ComboFTL 和 DPAFTL 将冷热数据分别存储在 MLC 和 SLC 上，用来优化整体性能。软划分固态盘有 TLC-FTL[32]、HFTL[33]。冷热数据一般是按照访问频数来进行分类，而 TLC-FTL 和 HFTL 是采用访问数据的大小进行分类，其在固态盘内部维护一个 SLC 的环形缓冲区，当有较小的写到来时，就将其放进改写区域，通过维护一个热数据识别方法来控制 SLC 到非 SLC 区域数据的迁移。

3.3 双模式 FTL

固态盘的 FTL 要同时兼顾读写操作的性能,但一些应用只有读操作或者写操作。文献[34]首次提出双模式 FTL 的概念。双模式 FTL 一般包括两种模式:一是支持随机读操作、顺序读操作,禁止随机写操作但支持顺序写操作,能为随机读、顺序读和顺序写提供最优的性能保证;二是支持随机写操作,但不能保证提供最优的性能。根据不同的应用场景可以设置 FTL 模式,来实现固态盘性能的最大化。

3.4 内部并行性加速

为了利用固态盘内部并行性来加速固态盘内部的读写操作,基于 DFTL 改进的 Parallel-DFT[35] 被提出。其提出了一个创新的 IO 调度策略,与 DFTL 一起工作以打破 DFTL 数据访问地址转换操作的耦合,并行地安排 DFTL 地址转换和数据访问操作,允许固态盘使其闪存访问通道资源,两种类型的操作都是完全并行的。

4 闪存数据库日志

闪存数据库的管理通常采用页管理,有时候也会采用日志管理的方式[36]。由于闪存的更新都是异步更新,因而如果能跳过 FTL 在固态盘内部进行优化,便可以将所有更新过程中的历史物理页直接当作日志记录。这种方式对于写操作性能优秀,但是读成本代价很高。大部分情况下,闪存数据库的日志主要用于系统恢复[37]。但也有一些直接在闪存上通过日志的方式建立索引结构,比如:B 树[38-39]。

5 闪存数据库缓冲区

在内存中设置缓冲区,将经常被访问或最有可能即将被访问的数据放进缓冲区中,可以明显地减少 I/O,提升系统整体的性能表现。因而对于闪存缓冲区的优化具有重要意义。

5.1 经典缓冲策略

早期的缓冲区研究工作,主要考虑闪存读写不对称的特点进行优化[40-42]。缓冲区维护能减少 I/O 的基本原理,是操作系统中数据访问的空间局域性和时间局部性,其是超越存储介质特性的存在,因而很多闪存固态盘上的缓冲区维护算法继承了磁盘缓冲区的策略。如:CFLRU[43]、LRU-WSR[44] 和 CCF-LRU[45] 就是继承经典的 LRU 缓冲区策略改进的算法。而 LIRS[46] 不仅考虑 LRU 策略的时间临近性,还加入了访问频数来加入缓冲区的维护策略。LIRS-WSR[47] 想着重优化减少写操作,维护脏页频数,并对频数进行排序,通过延迟频数高的逐出策略尽可能少的进行写操作。

5.2 自适应缓冲区

上述这些缓冲区算法都是基于写代价比读代价高的特点进行设计,但不同闪存设备的读写代价比往往不同。因而,能够自动适应不同类型闪存的缓冲区维护算法就变的非常必要。CASE[48] 通过动态维护干净页链表和脏页链表长度,来自适应获取不同读写代价比闪存设备的最佳配置。ACR[49] 和 CRAW-C[50] 也依据该思路进行优化。不同的是,ACR 不仅能自适应不同的闪存环境,还能自适应不同的读写访问模式,而 CRAW-C 是针对压缩的文件系统环境设计。

5.3 写合并缓冲区

频繁细粒度的写除了延迟代价高之外,还会造成闪存固态盘内部的垃圾回收。垃圾回收成本很高,会造成大量的块合并和擦除操作。为了减少块擦除操作的次数,FAB[51] 和 BPLRU[52] 采用块作为缓冲区的替换单元来达到此目的。另外,基于已有的 CFLRU,加入写聚集的策略,CFLRU 的优化版本 CFDC[53] 被提出。

6 闪存数据索引

索引是数据管理系统中最重要的组织工具,优秀的索引能大大提升数据访问的性能。下面将介绍闪存上具有代表性的几个重要索引结构的优化研究。

6.1 B-树优化

B 树和 B+树在更新结构时,会涉及大量的更新。一般情况下该索引存储在文件中,和闪存介质之间隔着一个 FTL 层,其结果便是更新的写放大极其严重。因而,文献[54-56]直接在闪存介质上基于日志的思想建立一个 B 树,这样就能让 B 树很好的支持频繁且细粒度的更新操作。B 树能够有效的支持随机查询,但是对于顺序查询效果较差。因而,实际数据管理工具采用的核心数据结构往往是 B+树。对于闪存 B+树的优化,文献[57]主要考虑很好的利用固态盘内部的内存资源,一部分内存缓存 B+树页子结点,另外一部分缓存更新请求,对更新操作采用延迟满足的策略来实现 B+树更新过程较少的写操作。

6.2 μ 树优化

传统的 B 树和 B+树每个结点的扇出都是相同

的，而 μ 树[58]采用一种从根结点往下，每一层扇出以 2 为倍数递增式增加的方式，将一棵树存储到闪存的一页上。当树内有更改时，直接再存储一页。这样做的情况下，在小数据规模维护时，任何一次更新最多对于索引只更新 1 页。优点是能够减少更新时写的页数量，缺点是同样的数据需要的索引存储空间会增大。相比之下，在更新比较频繁的情况下，该索引表现会非常优秀。

6.3 FD 树优化

基于 B+树，FD 树[59]采用和 LSM 树类似的思想，将其作为头部索引，在其下方加了两层有序段。其每一层都有一定的容量，容量阶梯式增加，当上一层满时与下一层合并。这样能有效减少随机写带来的写放大。LA-tree[60]采取了类似的惰性更新的思想，区别是其通过一个级联的缓冲区，在缓冲区内优化合并更新操作，来减少更新带来的写代价。

6.4 哈希索引优化

哈希索引也是数据管理系统中一个重要的索引结构，很多基本操作都需要基于该索引完成。但是，哈希索引对于闪存介质而言具有天然的不适应性。因为其更新位置均是随机的，且哈希更新细粒度的操作居多，因而会带来巨大的写放大效果。对于插入更新操作，SAL-HASH[61]采用和 LSM 树类似的思路，通过多层合并，尽量减少随机写带来的写放大。MicroHash[62]主要的优化思路是消除时间和能耗代价高昂的随机删除操作，通过索引和数据的调整，让删除以块的形式进行，以达到对哈希索引的性能和能耗进行的优化。

7 闪存数据库查询处理

查询过程的优化和存储模式是直接相关的，因而这里将先介绍闪存数据库的页存储模式，然后介绍基于该存储模式的经典的连接研究工作。

7.1 页存储模式

闪存页数据的存储主要有 3 种模式：NSM（行式数据存储）、DSM（列式数据存储）、PAX（混合数据存储）。PAX 实际上就是在 NSM 页内部采用类似于 DSM 的组织方式。连接操作往往只需要读取、输出和连接列相关的属性值，不需要处理所有的列属性值。连接操作是树查询中的核心算法，闪存连接算法 RARE-join[63]基于 PAX 存储模式，其只选择和查询结果相关的列属性，减少中间结果的生成来加速整个连接过程。缺点是相对于 NSM 和 DSM，PAX 维护成本较高。

7.2 连接算法优化

传统的连接算法，基于磁盘主要考虑的是减少代价很高的随机读写，增加顺序读写的比例。但是这种优化在支持快速随机读写的闪存上并没有明显的优化效果。因而，基于 PAX 存储模式的 Digest-Join[64]尝试发挥闪存的快速随机读写，来加速连接操作执行。Digest-Join 分为两个阶段，第一阶段生成摘要表，第二阶段通过随机读来连接。难点是第二阶段最小化页的读数据量的 NP 复杂度“页抓取问题”。Digest-Join 采取启发式的方法来解决该问题。SubJoin[65]根据连接列进行排序形成连接子表，然后进行连接。SubJoin 对于每个子表连接中的属性进行 DSM 列存储，需要连接的属性数据均从原始数据中获取。这样做，就是用顺序写替代随机读操作，从固态盘的性能角度看，性能上有所提升。

8 混合存储系统

固态盘虽然具有很好的性能优势，但其价格依然相对比较高，且固态盘寿命有限。数据访问往往具有时间局部性和空间局部性，因而一种思路就是将固态盘作为内存和磁盘之间缓存。固态盘能有效的支持随机读写，且相对于磁盘读写带宽高很多，磁盘顺序读写带宽表现良好，因而另外一种思路就是结合两者的优点，将数据合理地分配在两种存储介质上，以实现存储系统的提升。

文献[66-67]尝试将固态盘作为内存和磁盘之间的扩展缓存，来改善存储系统整体的 I/O 性能表现。文献[68]以延长固态盘寿命为目的，结合磁盘顺序写性能表现良好的特点，将磁盘作为固态盘的写缓存，这样就能对写操作进行进一步的处理合并，减少数据最终写到固态盘的写放大，减少写带来的擦除操作，达到延长固态盘寿命的目的。实际数据管理中，不同数据访问的模式存在不同特点，有些数据会被经常访问，有些访问频数比较少，有些数据是更新密集型，有些是读密集型等等。如果能识别出这些数据，能够合理的分配到固态盘和磁盘上，就能让存储系统的整体性能接近纯固态盘的存储性能。I-CASH[69]尝试识别出写密集的数据，并将其修改成以日志的方式存储在磁盘上，以减少固态盘的写操作。对于磁盘固态盘混合存储的环境，Hystor[70]设法寻找出影响整体性能关键的数据，将其放置到性能更优异的固态盘上，以提高整体存储系统的性能。

9 结束语

本文首先对闪存的特性进行描述,然后分别对闪存数据管理各领域的研究情况进行了详细的介绍,分别指出基于闪存各类算法的优化思路。概括来看,这些优化思想主要分为 4 大类:第一类,使用闪存的优良特点直接进行优化;第二类,用成本更低的读操作代替成本更高的写操作;第三类,通过日志的策略,将写操作顺序化,以减少写放大;第四类,通过懒惰执行合并操作,以减少对闪存的写放大。总的来说,目前几乎所有的研究工作都主要是通过上述 4 种思路,对闪存上的数据结构和算法进行优化。

参考文献

[1] ANDERSEN D G, FRANKLIN J, KAMINSKY M, et al. FAWN: a fastarray of wimpy nodes. [C]//Proceedings of the 22nd symposium on Operating systems principles. New York: ACM Press, 2009: 1.

[2] BU Y, HALIM F, KIM C, et al. Using SSDs to scale up Google Fusion Tables, a database-in-the-cloud [C]//2016 IEEE 32nd International Conference on Data Engineering. IEEE, 2016: 1263-1274.

[3] CAULFIELD A M, GRUPP L M, SWANSON S. Gordon. Using flash memory to build fast, power-efficient clusters for data-intensive applications [C]//Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems. Washington, DC: ACM, 2009: 217-228.

[4] REDELL D D, DALAL Y K, HORSLEY T R, et al. Pilot: An operating system for a personal computer [C]//Proceedings of the 7th ACM Symposium on Operating Systems Principles (SOSP). 1979: 106-107.

[5] LEE S W, MOON B, PARK C. Advances in flash memory SSD technology for enterprise database applications [C]//Proceedings of the 35th SIGMOD international conference on Management of data - SIGMOD '09. New York, USA: ACM Press, 2009: 863.

[6] Hans Reiser. ReiserFS [EB/OL]. [2021-03-01]. https://reiser4.wiki.kernel.org/index.php/Main_Page.

[7] TWEEDIE S C. Journaling the Linux ext2fs filesystem [C]//The Fourth Annual Linux Expo. Durham, North Carolina, 1998:1-8.

[8] TWEEDIE S. Ext3, journaling filesystem [C]//Ottawa Linux Symposium. 2000:1-7.

[9] ROSENBLUM M, OUSTERHOUT J K. The design and implementation of a log-structured file system [J]. ACM Transactions on Computer Systems (TOCS), ACM New York, NY, USA, 1992, 10(1): 26-52.

[10] CHOI H J, LIM S H, PARK K H. JFTL. A flash translation layer based on a journal remapping for flash memory [J]. ACM Transactions on Storage (TOS), ACM New York, NY, USA, 2009, 4(4): 1-22.

[11] WOODHOUSE D. JFFS: The journalling flash file system [C]// Ottawa linux symposium. 2001, 2001.

[12] MIN C, KIM K, CHO H, et al. SFS: random write considered harmful in solid state drives [C]//FAST. 2012: 1-16.

[13] LEE C, SIM D, HWANG J, et al. F2FS: Anew file system for flash storage [C]//13th USENIX Conference on File and Storage Technologies (FAST 15). 2015: 273-286.

[14] LU Y, SHU J, ZHENG W. Extending the lifetime of flash-based storage through reducing write amplification from file systems [C]//11th USENIX Conference on File and Storage Technologies (FAST 13). 2013: 257-270.

[15] LU Y, SHU J, WANG W. ReconFS: A reconstructable file system on flash storage [C]//12th USENIX Conference on File and Storage Technologies (FAST 14). 2014: 75-88.

[16] ZHANG J, SHU J, LU Y. ParaFS: A log-structured file system to exploit the internal parallelism of flash devices [C]//2016 USENIX Annual Technical Conference (USENIX ATC 16). 2016: 87-100.

[17] Aleph One Ltd.2007. Yaffs [EB/OL]. [2020-05-01]. https://yaffs.net/.

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

[19] MA D, FENG J, LI G .LazyFTL: A page-level flash translation layer optimized for NAND flash memory [C]//Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. 2011: 1-12.

[20] GUPTA A, KIM Y, URGAONKAR B. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings [J]. Acm Sigplan Notices, ACM New York, NY, USA, 2009, 44(3): 229-240.

[21] YEO D B, PAIK J Y, CHUNG T S. Request-Size Aware Flash Translation Layer Based on Page-Level Mapping [C]//2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES). IEEE, 2016: 68-71.

[22] CHOUDHURI S, GIVARGIS T. Performance improvement of block based NAND flash translation layer [C]//Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis. 2007: 257-262.

[23] CHEN R, QIN Z, WANG Y, et al. On-demand block-level address mapping in large-scale NAND flash storage systems [J]. IEEE Transactions on Computers, IEEE, 2014, 64(6): 1729-1741.

[24] LEE S, SHIN D, KIM Y J, et al. LAST: locality-aware sector translation for NAND flash memory-based storage systems [J]. ACM SIGOPS Operating Systems Review, ACM New York, NY, USA, 2008, 42(6): 36-42.

[25] KOO D, SHIN D. Adaptive log block mapping scheme for log bufferbased FTL (flash translation layer) [C]//IWSSPS 2009: International Workshop on Software Support for Portable Storage. 2009:1-6.

[26] LEE S W, PARK D J, CHUNG T S, et al. A log buffer-based flash translation layer using fully-associative sector translation [J]. ACM Transactions on Embedded Computing Systems (TECS), ACM New York, NY, USA, 2007, 6(3): 18-es.

[27] JUNG D, KANG J U, JO H, et al. Superblock FTL: A superblock-based flash translation layer with a hybrid address translation scheme [J]. ACM Transactions on Embedded Computing Systems (TECS), ACM New York, NY, USA,

2010，9(4)：1-41.
[28]CHANG L P，KUO T W. An efficient management scheme for large-scale flash-memory storage systems[C]//Proceedings of the 2004 ACM symposium on Applied computing. 2004：862-868.
[29]CHANG L P. A hybrid approach to NAND-flash-based solid-state disks[J]. IEEE Transactions on Computers，IEEE，2010，59(10)：1337-1349.
[30]IM S，SHIN D. ComboFTL：Improving performance and lifespan of MLC flash memory using SLC flash buffer[J]. Journal of Systems Architecture，Elsevier，2010，56(12)：641-653.
[31]KWON S J，CHUNG T S. Data pattern aware FTL for SLC+MLC hybrid SSD[J]. Design Automation for Embedded Systems，Springer，2015，19(1)：101-127.
[32]YAO L，LIU D，ZHONG K，et al. TLC-FTL：Workload-aware flash translation layer for tlc/slc dual-mode flash memory in embedded systems[C]//2015 IEEE 17th International Conference on High Performance Computing and Communications，2015 IEEE 7th International Symposium on Cyberspace Safety and Security，and 2015 IEEE 12th International Conference on Embedded Software and Systems. IEEE，2015：831-834.
[33]YANG M C，CHANG Y H，TSAO C W，et al. Utilization-aware self-tuning design for TLC flash storage devices[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems，IEEE，2016，24(10)：3132-3144.
[34]BONNET P，BOUGANIM L. Flash device support for database management[C]//CIDR. 2011：1-8.
[35]XIE W，CHEN Y，ROTH P C. Parallel-DFTL：Aflash translation layer that exploits internal parallelism in solid state drives[C]//2016 IEEE International Conference on Networking，Architecture and Storage (NAS). IEEE，2016：1-10.
[36]LEE S W，MOON B. Design of flash-based DBMS：an in-page logging approach[C]//Proceedings of the 2007 ACM SIGMOD international conference on Management of data. 2007：55-66.
[37]FANG H W，YEH M Y，KUO T W. MLC-Flash-Friendly logging and recovery for databases[C]//Proceedings of the 28th Annual ACM Symposium on Applied Computing. 2013：1541-1546.
[38]KIM B，LEE D H. LSB-Tree：a log-structured B-Tree index structure for NAND flash SSDs[J]. Design Automation for Embedded Systems，Springer，2015，19(1)：77-100.
[39]JIN R，CHO H J，CHUNG T S. A GroupBased In-Block Logging for Flash Based Systems[C]//2013 International Conference on IT Convergence and Security (ICITCS). IEEE，2013：1-3.
[40]ROBINSON J T，DEVARAKONDA M V. Data cache management using frequency-based replacement[C]//Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems. 1990：134-142.
[41]LEE D，CHOI J，KIM J H，et al. LRFU：Aspectrum of policies that subsumes the least recently used and least frequently used policies[J]. IEEE Computer Architecture Letters，IEEE Computer Society，2001，50(12)：1352-1361.
[42]LEE R，DING X，CHEN F，et al. MCC-DB：Minimizing cache conflicts in multi-core processors for databases[J]. Proceedings of the VLDB Endowment，VLDB Endowment，2009，2(1)：373-384.
[43]PARK S，JUNG D，KANG J，et al. CFLRU：a replacement algorithm for flash memory[C]//Proceedings of the 2006 international conference on Compilers，architecture and synthesis for embedded systems. 2006：234-241.
[44]JUNG H，SHIM H，PARK S，et al. LRU-WSR：integration of LRU and writes sequence reordering for flash memory[J]. IEEE Transactions on Consumer Electronics，IEEE，2008，54(3)：1215-1223.
[45]LI Z，JIN P，SU X，et al. CCF-LRU：A new buffer replacement algorithm for flash memory[J]. IEEE Transactions on Consumer Electronics，IEEE，2009，55(3)：1351-1359.
[46]JIANG S，ZHANG X. LIRS：An efficient low inter-reference recency set replacement policy to improve buffer cache performance[J]. ACM SIGMETRICS Performance Evaluation Review，ACM New York，NY，USA，2002，30(1)：31-42.
[47]JUNG H，YOON K，SHIM H，et al. LIRS-WSR：Integration of LIRS and writes sequence reordering for flash memory[C]//International Conference on Computational Science and Its Applications. Springer，2007：224-237.
[48]OU Y，HÄRDER T. Issues of flash-aware buffer management for database systems[C]//British National Conference on Databases. Springer，2010：127-130.
[49]TANG X，MENG X. ACR：an adaptive cost-aware buffer replacement algorithm for flash storage devices[C]//2010 Eleventh International Conference on Mobile Data Management. IEEE，2010：33-42.
[50]PARK J，LEE H，HYUN S，et al. A cost-aware page replacement algorithm for nand flash based mobile embedded systems[C]//Proceedings of the seventh ACM international conference on Embedded software. 2009：315-324.
[51]JO H，KANG J U，PARK S Y，et al. FAB：Flash-aware buffer management policy for portable media players[J]. IEEE Transactions on Consumer Electronics，IEEE，2006，52(2)：485-493.
[52]KIM H，AHN S. BPLRU：A Buffer Management Scheme for Improving Random Writes in Flash Storage.[C]//FAST. 2008，8：1-14.
[53]SEO D，SHIN D. Recently-evicted-first buffer replacement policy for flash storage devices[J]. IEEE Transactions on Consumer Electronics，IEEE，2008，54(3)：1228-1235.
[54]WU C H，CHANG L P，KUO T W. An efficient B-tree layer for flash-memory storage systems[C]//International Conference on Real-Time and Embedded Computer Systems and Applications. Springer，2003：409-430.
[55]LEE H S，PARK S，SONG H J，et al. An efficient buffer management scheme for implementing a B-tree on NAND flash memory[C]//International Conference on Embedded Software and Systems. Springer，2007：181-192.
[56]NA G J，MOON B，LEE S W. In-page logging b-tree for flash memory[C]//International Conference on Database Systems for Advanced Applications. Springer，2009：755-758.
[57]ON S T，HU H，LI Y，et al. Lazy-update b+-tree for flash devices[C]//2009 Tenth International Conference on Mobile Data Management：Systems，Services and Middleware. IEEE，2009：323-328.
[58]KANG D，JUNG D，KANG J U，et al. μ-tree：An ordered index structure for NAND flash memory[C]//Proceedings of the 7th ACM & IEEE international conference on Embedded software. 2007：144-153.
[59]LI Y，HE B，LUO Q，et al. Tree indexing on flash disks[C]//2009 IEEE 25th International Conference on Data Engineering. IEEE，2009：1303-1306.