计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20150644
54(1):142-153,2017

一种基于分类策略的聚簇页级闪存转换层算法

姚英彪[1] 杜晨杰[1,2] 王发宽[1]
[1](杭州电子科技大学通信工程学院 杭州 310018)
[2](浙江万里学院科研部 浙江宁波 315100)
(yaoyb@hdu.edu.cn)

A Clustered Page-Level Flash Translation Layer Algorithm Based on Classification Strategy

Yao Yingbiao[1], Du Chenjie[1,2], and Wang Fakuan[1]
[1](*School of Communication Engineering, Hangzhou Dianzi University, Hangzhou* 310018)
[2](*Department of Science and Technology, Zhejiang Wanli University, Ningbo, Zhejiang* 315100)

Abstract This paper proposes a novel clustered page-level flash translation layer (CPFTL) algorithm which is based on classification strategy. Firstly, CPFTL divides RAM into hot cached mapping table (H-CMT), cold cached mapping table (C-CMT) and sequential cached mapping table (S-CMT), which are responsible for buffering map entries of requests with high temporal locality, low temporal locality and high spatial locality, respectively. Secondly, in order to benefit from the spatial locality of sequential requests, CPFTL prefetches multiple sequential map entries into S-CMT, and thus it can improve the response time of sequential requests. Finally, in order to reduce the read and write overhead of translation pages, CPFTL clusters the map entries which belong to the same translation page in C-CMT together, and manage these clusters by LRU (least recently used)strategy. When C-CMT is full, according to the map entry number and LRU of clusters, CPFTL chooses an appropriate cluster to evict into Flash. CPFTL has been extensively evaluated under various realistic workloads. Compared with the state-of-art FTL schemes such as classic DFTL and the latest SDFTL, our benchmark results show that CPFTL can improve cache hit ratio, operation counts of translation pages, response time and erase counts.

Key words solid state drive (SSD); flash translation layer (FTL); classification strategy; mapping table; locality

摘 要 提出一种基于分类策略的聚簇页级闪存转换层算法——CPFTL.1)CPFTL将地址映射缓存分为热映射表缓存、冷映射表缓存和连续映射表缓存,分别用来缓存访问频繁的请求的映射项、访问不频繁的请求的映射项和高空间本地性的连续请求的映射项,有效提升各类请求的处理能力;2)为利用连续请求的空间本地性,CPFTL的连续映射表缓存预取多个连续的映射项,提高它对连续请求的响应性能;3)为减少页级映射算法的转换页读写开销,CPFTL的冷映射表缓存采用聚簇策略,即将属于同一转换页中的映射项进行聚簇,按簇进行LRU管理,当冷映射表缓存满时,根据簇的映射项个数和LRU选取

收稿日期:2015-07-13;**修回日期**:2015-12-14
基金项目:国家自然科学基金项目(61100044);浙江省科技创新基金项目(2013TD03);浙江省科技计划资助项目(2013C31100)
This work was supported by the National Natural Science Foundation of China (61100044), Zhejiang Province Science and Technology Innovation Focused Team Foundation (2013TD03), and Zhejiang Province Science and Technology Project (2013C31100).

合适的簇剔除到闪存.实验结果显示,相比经典的页级 DFTL 算法和最新的 SDFTL 算法,CPFTL 的缓存命中率、平均响应时间、地址转换页操作次数和闪存块擦除次数都有显著提升.

关键词 固态硬盘;闪存转换层;分类策略;映射表;本地性

中图法分类号 TP333

近半个世纪来,随着计算机体系结构技术及芯片加工技术的不断进步,计算机系统的 CPU 性能与 IO 性能的差距不断扩大[1].计算机系统 IO 性能的瓶颈在于硬盘(hard disk drive, HDD).这些年虽然 HDD 容量有了很大的提升,但由于其存在机械旋转结构,访问速度提升有限,这使得基于 HDD 的存储系统成为计算机系统的性能瓶颈之一.相比于传统的硬盘,固态硬盘(solid state drive, SSD)呈现出许多优良的性能:低功耗、读写速度快、防震抗摔性好、无噪音、重量轻、体积小等,因此 SSD 在许多领域已经开始替代传统硬盘,它是当前存储领域的研究热点之一.

目前常见的 SSD 主要基于 NAND 闪存,而 NAND 闪存的结构与传统的磁存储介质不同,其主要特点有:1)闪存只提供读、写和擦除 3 种操作,且这 3 种操作性能不对称,读最快,写次之,擦除最慢.2)闪存是按页(page)、块(block)、平面(plane)的结构进行组织.页是读/写的最小单位,一般为 2 KB,4 KB,8 KB;块是擦除的最小单位,一个块一般包含 64 页、128 页.3)闪存擦除后只能写 1 次,即所谓的写前擦除(erase-before-write)[2],这造成闪存不能原地更新,否则会带来巨大的开销.4)闪存每个存储单元的编程/擦除(P/E)次数有限[3],通常 SLC 闪存的 P/E 次数是 10 万次左右,MLC 闪存的 P/E 次数是 1 万次左右,超过 P/E 次数后,闪存存储数据不再可靠.

本文研究 SSD 的闪存转换层(flash translation layer, FTL)中的地址映射算法,它对 SSD 的性能、寿命有至关重要的影响,也是当前 SSD 固件设计中的研究热点和难点[4].

1 相关工作

1.1 FTL 地址映射算法

FTL 是一个中间软件转换层,它隐藏了闪存的擦除操作,将 SSD 模拟成只有读写操作的传统硬盘的形式,以适应当前的文件系统.FTL 一般包括地址映射、磨损平衡和垃圾回收 3 个模块,其中地址映射是 FTL 最核心的功能,它负责将来自文件系统的逻辑地址转换为闪存中的物理地址.根据逻辑地址与物理地址映射粒度的大小,FTL 可以分为页映射[5-7]、块映射[8-10]以及混合映射[11-15].

页映射需要维护逻辑页和物理页之间的映射关系,即建立页映射表.传统的页映射将整个页映射表存储在 RAM 中,当 SSD 容量增大时,页级映射表需要大量的 RAM 空间,这造成传统的页级映射很难适应目前的大容量 SSD.块级映射表仅需维护逻辑块和物理块之间的映射关系,因而 RAM 开销大幅减少;但逻辑页只能映射到块中的固定页,从而降低了地址映射的灵活性[16].为克服以上 2 种映射方式的缺陷,混合映射机制被提出.混合映射机制将闪存分为数据块和日志块,日志块内使用页级映射表,数据块使用块级映射表,用日志块来记录更新.混合映射机制能够有效降低页映射占用空间过大和块映射频繁擦除的问题,但存在垃圾回收效率低和处理随机访问请求性能差的缺点.

针对混合映射机制出现的问题,Gupta 等人[6]重新设计页级映射机制,提出一种按需的页级 FTL (demand-based FTL, DFTL)算法.DFTL 采用基于页的映射机制,将整个映射表都存储在闪存中,并将闪存从逻辑上分为数据块区域和转换块区域,分别用于存储常规数据和映射表信息;然后根据实际请求动态加载部分映射表(cached mapping table, CMT)到 RAM 中,来处理访问频繁的请求;同时,DFTL 设置了一个全局转换目录(global translation directory, GTD)来记录转换页的变化.相对混合映射机制,DFTL 取得明显的响应时间的改善.

在 DFTL 思想的启发下,国内外的研究人员提出各种新型页级闪存转换层算法,代表有 SDFTL[16], S-FTL[17], HAT[18], OAFTL[19], CDFTL[20], WAPFTL[21]等.算法 SDFTL 提出在 RAM 中增设连续映射表缓存和二级映射表缓存;算法 S-FTL 提出通过识别 3 种空间本地性(顺序写、簇写和稀疏写)访问模式来减少按页映射的映射表大小;算法 HAT 提出将页映射表存储在 PCM 芯片,实现数据访问和页映射信息访问的并行;算法 OAFTL 提出根据访问操作的类型来组织映射表信息,并为映射

页保留日志信息来缓冲频繁修改的映射信息;算法 CDFTL 提出以"簇"的方式来装载/剔除映射项;算法 WAPFTL 提出基于页映射机制的自适应地址映射算法,能够在地址转换过程中预测负载读写特性并自适应地调整地址映射信息缓存的策略.

1.2 CPFTL 算法

本文提出的聚簇页级闪存转换层(clustered page-level flash translation layer, CPFTL)也是一种新型页级映射机制,与以上工作相比,本文创新主要体现在 3 点:

1) 采用分类策略思想,将地址映射缓存分为热映射表缓存(hot CMT, H-CMT)、冷映射表缓存(cold CMT, C-CMT)和连续映射表缓存(sequential CMT, S-CMT),分别用来缓存访问频繁的请求的映射项,访问不频繁的请求的映射项和高空间本地性请求(连续请求)的映射项;

2) 为利用连续请求的空间本地性,CPFTL 预取多个连续的映射项到 S-CMT 中,提高它对连续请求的处理能力;

3) 为减少新型页级映射算法的转换页读写开销,CPFTL 的 C-CMT 采用聚簇策略,即将属于同一转换页中的映射项进行聚簇,按簇进行最近最少使用(least recently used, LRU)管理;当 C-CMT 满时,根据簇内包含的映射项个数和簇的 LRU 来选取合适的簇剔除到闪存中去.

实验结果显示,平均而言,CPFTL 相比经典的 DFTL 算法,总体缓存命中率提高 50.59%,响应时间减少 24.43%,地址转换页操作次数减少 82.87%,闪存块擦除次数减少 29.35%;相比最新的 SDFTL 算法,总体缓存命中率提升 9.88%,响应时间减少 8.25%,地址转换页操作次数减少 50.62%,闪存块擦除次数减少 9.26%.

2 CPFTL 算法设计

2.1 总体构架

CPFTL 总体结构如图 1 所示,图 1 中 D_{LPN} 和 D_{PPN} 分别表示数据的逻辑页地址和物理页地址,T_{VPN} 和 T_{PPN} 分别表示转换块区域中页的虚拟地址和物理地址. 在 CPFTL 中,闪存被分为数据块区域和转换块区域,RAM 被分为 H-CMT, S-CMT, C-CMT, GTD 四个部分.

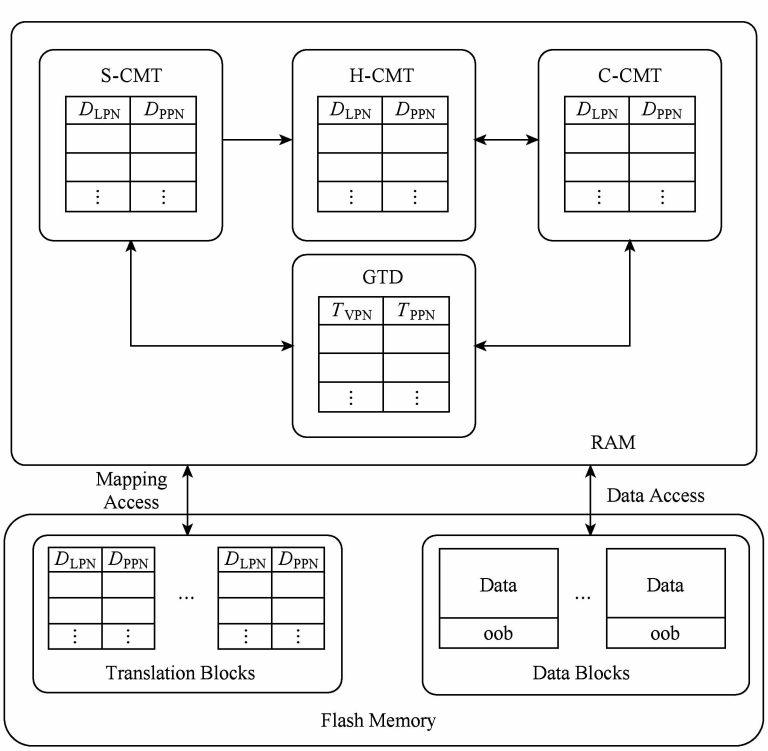


Fig. 1 The overall architecture of CPFTL

图 1 CPFTL 的总体构架

数据块区域用于实际的数据存储,转换块区域用于页级映射表的存储. H-CMT 用于缓存访问频率较高的请求的映射项;C-CMT 用于缓存访问频率低下的请求的映射项,以及当 H-CMT 满时从 H-CMT

剔除的、发生更新的映射项;S-CMT 用于缓存高空间本地性请求(连续请求)的映射项;GTD 的作用与 DFTL 的类似,用来记录转换块区域中每个页的地址映射项.

2.2 CPFTL 关键结构

1) H-CMT

H-CMT 主要是用来缓存访问频繁的请求的映射项. 当有请求到来且请求的映射项在 H-CMT 中时,就可立即得到服务. 当请求不在 H-CMT,则到 S-CMT 和 C-CMT 中查询. 再者,H-CMT 使用 LRU 策略对每个映射项进行管理. 当 H-CMT 满时,选择最近最少访问的请求的映射项进行剔除. 此时,若该映射项已更新,则将其剔除到 C-CMT 中;反之,则直接从缓存中剔除出去.

2) S-CMT

S-CMT 主要是用来缓存高空间本地性请求,也即连续请求的映射项. 当一个新的请求到来时,如果请求大小大于 2 KB,CPFTL 便认为该请求具有连续性,进而 1 次加载 1 组连续的映射项到 S-CMT 中,本文将在 3.2 节中研究 1 组包含多少个映射项的参数设置问题. 当请求在 S-CMT 再次命中后,则认为该请求为频繁访问请求,并把命中的映射项加载到 H-CMT 中. 再者,S-CMT 采用先进先出(first in first out,FIFO)的管理策略,即每次选择停留时间最长的 1 组映射项进行剔除或更新.

3) C-CMT

C-CMT 主要用来缓存访问频率低的请求的映射项. 一方面,大小小于或者等于 2 KB 的请求不在缓存中命中时,都被认为是访问频率低的请求,直接加载到 C-CMT 中;另一方面,C-CMT 还存储从 H-CMT 剔除的、发生更新的映射项. 此外,若 C-CMT 中的映射项被二次访问,同样认为该请求为频繁访问请求,需要将此映射项加载到 H-CMT 中. 再者,C-CMT 采用聚簇的思想,即将属于同一转换页中的映射项进行聚簇,并使用 LRU 策略对所有的簇进行管理,如图 2 所示:

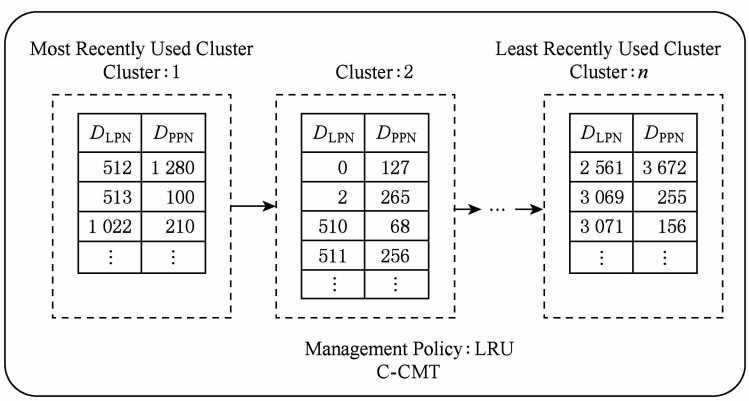


Fig. 2 Cluster strategy of C-CMT

图 2 C-CMT 的聚簇策略

当 C-CMT 满后,按簇为单位进行剔除. 选择剔除簇时,考虑到:一方面,若仅根据簇的映射项个数选择剔除簇,即选择映射项最多的簇进行剔除,则可能会使访问频率低下的簇对应的映射项长时间滞留在 C-CMT 中,浪费了宝贵的 C-CMT 缓存空间;另一方面,若仅根据 LRU 原则选择剔除簇,即选择最近最少访问的簇进行剔除,则可能造成每次剔除回收的缓存空间有限,从而造成经常的地址转换页访问,即增大地址转换页的读写开销. 实际上,本文 3.2 节的实验数据也证实了上述 2 点. 因此,CPFTL 在选择剔除簇时,综合考虑簇的映射项个数和簇在 LRU 队列中的位置. 具体来说,当 C-CMT 满时,先判断具有最多映射项的簇包含的映射项个数是否大于某个阈值,如果是,选择该簇进行剔除;否则,根据 LRU 原则选择最近最少访问簇进行剔除. 本文将在后续的实验中研究这个阈值设置问题.

2.3 CPFTL 处理流程

算法 1 给出了 CPFTL 算法的伪代码,输入的是请求的逻辑页号、请求的大小和请求类型,算法 2、算法 3 和算法 4 分别是 H-CMT,C-CMT,S-CMT 加载映射项时的伪代码.

算法 1. CPFTL 算法.

输入:请求 R 的逻辑页号 R_{LPN}、请求大小 R_{Size}、请求类型 R_{type};

输出:NULL.

① $Size=R_{Size}$;

② while ($Size \neq 0$)
③ if (R_{LPN}在 H-CMT 或 S-CMT 或 C-CMT 得到命中)
④ 服务这个请求; /* 执行算法行⑬～⑰ */
⑤ if (R_{LPN}在 S-CMT 或 C-CMT 二次命中)
⑥ 执行算法 2,加载 R_{LPN}的相关映射项到 H-CMT;
⑦ end if
⑧ else if ($Size \leqslant 2$ KB)
/* 加载映射项到 C-CMT */
⑨ 执行算法 3,加载 R_{LPN}的相关映射项到 C-CMT;
⑩ 服务这个请求; /* 执行算法行⑬～⑰ */
⑪ else /* 加载映射项到 S-CMT */
⑫ 执行算法 4,加载一组映射项到 S-CMT;
⑬ if (R_{type}是读请求)
⑭ 通过映射项读取物理页号;
⑮ else /* R_{type}是写请求 */
⑯ 用新的物理页号来服务这个请求;
⑰ end if
⑱ end if
⑲ R_{LPN}++;
⑳ $Size$--;
㉑ end while

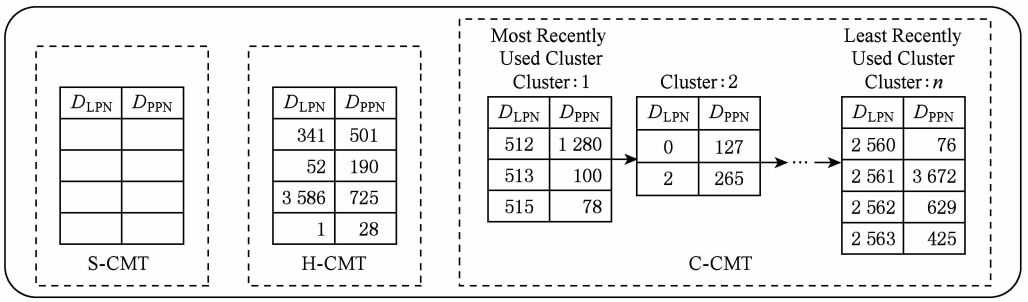
算法 2. H-CMT 更新策略.
输入: R_{LPN};
输出: NULL.
① if (H-CMT 已满)
② 根据 LRU 策略选择牺牲项;
③ if (牺牲项已更新)
④ 执行算法 3,加载牺牲项到 C-CMT;
⑤ else
⑥ 将牺牲映射项直接从 H-CMT 中剔除出去;
⑦ end if
⑧ end if
⑨ 加载 R_{LPN}的相关映射项到 H-CMT.

算法 3. C-CMT 更新策略.
输入: R_{LPN};
输出: NULL.
① if (C-CMT 已满)
② if (拥有最多映射项的簇的映射项数目大于阈值)
③ 从 C-CMT 中将这个簇剔除出去;
④ else
⑤ 从 C-CMT 中剔除最久未使用的簇;
⑥ end if
⑦ end if
⑧ 加载 R_{LPN}的相关映射项到 C-CMT.

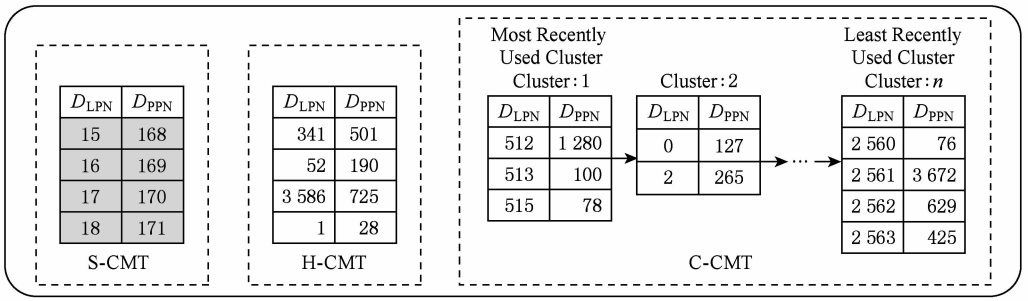
算法 4. S-CMT 更新策略.
输入: R_{LPN};
输出: NULL.
① if (S-CMT 已满)
② 根据 FIFO 策略,从 S-CMT 中剔除一组映射项;
③ end if
④ 加载 R_{LPN}的相关映射项和随后的一些映射项到 S-CMT.

下面给出一个示例,该示例详细描述了 CPFTL 处理请求的流程,为方便演示,我们弱化了缓存空间的大小. 假设请求到达的顺序如下:(15,4,R),(16,1,W),(3584,1,W),(2,1,R). 其中,括号内的 3 个参数分别表示请求的逻辑页号、请求大小和请求类型;R 表示读请求,W 表示写请求. 图 3 给出了 H-CMT, S-CMT,C-CMT 在各阶段的存储状态,假设逻辑页 3586,1 的映射项已经发生改变,且 H-CMT 已存满映射项,S-CMT 不存储映射项,如图 3(a)所示.

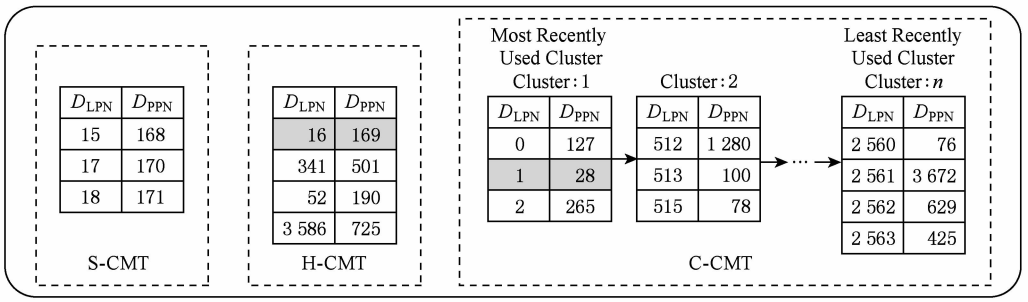
CPFTL 的处理流程为:当请求(15,4,R)到来时,由于请求的映射项不在缓存,且 CPFTL 判断出请求的连续性,则预取一组映射项到 S-CMT 中,使得该连续请求的后 3 个逻辑页 16～18 都能在 S-CMT 命中,如图 3(b)所示. 当请求(16,1,W)到来时,由于该请求的映射项在 S-CMT 中且被二次访问,需将其加载到 H-CMT 中,这时 H-CMT 已满,则剔除、更新逻辑页 1 的映射项到 C-CMT,然后将逻辑页 16 的映射项加载到 H-CMT 中,如图 3(c)所示. 当请求(3584,1,W)到来时,由于该请求的映射项不在缓存,则需将该请求的映射项加载到 C-CMT 中,这时 C-CMT 已满,假设具有最多映射项的簇包含的映射项个数小于阈值,则剔除 C-CMT 中最近最少访问的簇,然后加载逻辑页 3584 的映射项到 C-CMT,如图 3(d)所示. 当请求(2,1,R)到来时,由于该请求的映射项在 C-CMT 且被二次访问,则需要其加载到 H-CMT,这时 H-CMT 已满,则剔除、更新逻辑页 3586 的映射项到 C-CMT 中,然后加载逻辑页 2 的映射项到 H-CMT 中,如图 3(e)所示.



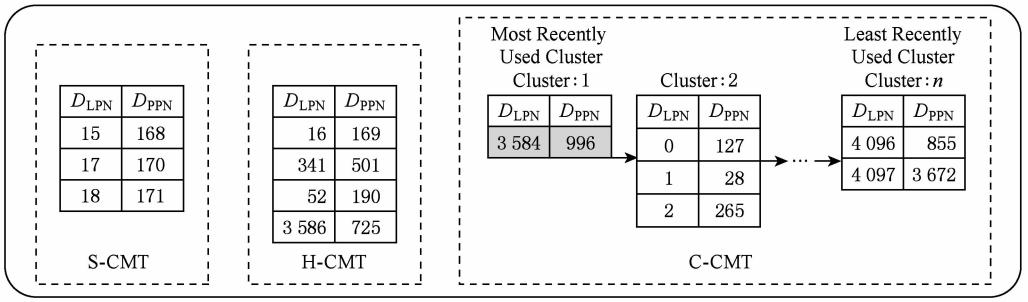
(a) Original state of mapping table



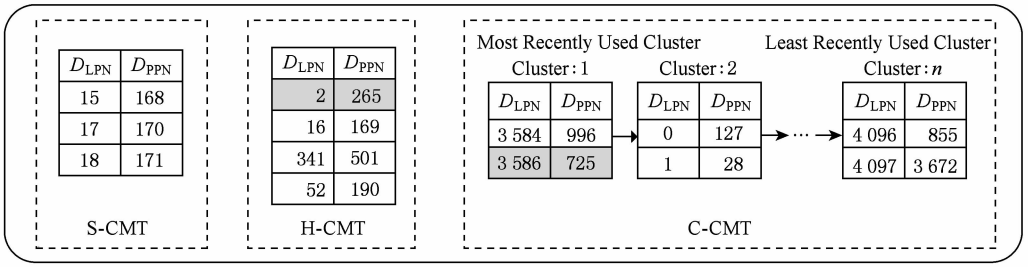
(b) Logical page number 15, 16, 17, 18 are served



(c) Logical page number 16 is served



(d) Logical page number 3 584 is served



(e) End state of mapping table

Old Data New Data

Fig. 3 An example of CPFTL process

图 3 CPFTL 处理请求的示例

3 性能评估

3.1 实验设置

在实验中,CPFTL 采用 FlashSim[22] 进行性能评估. FlashSim 是宾夕法尼亚州立大学开发的固态硬盘仿真器,它是对 DiskSim[23] 的扩展,即在 DiskSim 的基础上添加闪存仿真模块和闪存与上层的接口模块. 实验中,固态硬盘大小为 16 GB,地址映射缓存大小为 64 KB,闪存的关键参数如表 1 所示:

Table 1 Configuration Parameters of Flash Memory

表 1 闪存的关键参数

Parameters	Value
Page Size/KB	2
Block Size/KB	128
Page Read Time/μs	32.7
Page Write Time/μs	101.5
Block Erase Time/ms	1.5

在实验中,使用一系列真实的企业级负载、DiskSim 产生的合成负载以及用磁盘驱动数据跟踪器 DiskMon[24] 收集的负载,来仿真 3 种 FTL 算法的性能. Financial1 和 Financial2[25] 来自 OLTP 应用,是金融机构处理在线联机事务的 Trace. WebSearch1 和 WebSearch2[26] 是来自存储性能委员会(storage performance council, SPC),以读为主的网页搜索引擎 Trace. Systemdisk 是由专门的 Trace 收集工具 DiskMon 收集的 Trace,Test 是由 DiskSim 合成的 Trace. 表 2 给出了实验中使用的各负载的特性.

Table 2 The Trace Characteristics

表 2 实验中 Trace 的特性

Traces	Number of Requests	Write /%	Average Request Size/KB	Average Arriving Time/ms
Financial1	5 344 983	76.84	4.92	8.19
Financial2	3 699 194	17.65	3.89	11.08
Systemdisk	8 545 655	52.61	4.50	9.38
Test	322 875	90.51	4.38	133.51
WebSearch1	85 111	0.12	16.90	37.03
WebSearch2	348 439	0.14	17.13	44.18

实验中,我们选取经典的页级映射算法 DFTL[6] 和最新的页级映射算法 SDFTL[16] 作为比较算法. 我们选取 4 个性能指标——缓存命中率、地址转换页操作次数、平均响应时间和闪存块擦除次数作为评价标准,具体解释如下:

1) 缓存命中率指请求的映射项在映射缓存中的命中比率,即直接在映射缓存中得到服务的请求占总请求的比例.

2) 地址转换页操作包括转换页读操作和写操作,即当请求未在映射缓存中命中时引起的转换页读操作或者映射缓存满时,发生剔除引起的写操作.

3) 响应时间是指请求完成服务时间与到达时间的时间差,它是垃圾回收、地址映射的开销以及数据访问时间的综合,也是衡量 SSD 性能的关键指标.

4) 块擦除次数包括地址转换块的擦除次数和数据块的擦除次数,它直接决定了 SSD 的使用寿命.

3.2 CPFTL 参数设置研究

1) H-CMT,S-CMT,C-CMT 的空间配置

为获得最佳的 H-CMT,S-CMT,C-CMT 的缓存空间配置,我们进行了 36 组实验. 在实验中,S-CMT 与 C-CMT 的大小依次设置为 4 KB,8 KB,…,24 KB,总容量为 64 KB 不变. S-CMT 的映射项预取个数和 C-CMT 的阈值分别设为 4 和 100. 仿真 Trace 选择具有代表性的 Financial1 和 Financial2,性能指标选择平均响应时间(average response time),仿真结果如图 4 所示.

从图 4 可知,在 C-CMT 容量不变时,一方面,随着 S-CMT 的容量逐渐增大,Financial1 和 Financial2 的平均响应时间经历了由大变小再变大的过程,大多数情况下在 S-CMT 的大小为 16 KB 时有最小的平均响应时间;另一方面,这个变化幅度较小,这也说明 Financial1 和 Financial2 对 S-CMT 的容量并不是很敏感. 在 S-CMT 容量不变时,随着 C-CMT 的容量逐渐增大,Financial1 和 Financial2 的平均响应时间的变化趋势是逐渐减小,尤其是从 4 KB 增加到 16 KB 时,平均响应时间减小还是非常明显. 根据图 4 的实验结果,为兼顾各种类型负载,后续实验中我们取 H-CMT 的大小为 28 KB,S-CMT 的大小为 16 KB,C-CMT 的大小为 20 KB.

为详尽分析 CPFTL 的 H-CMT,S-CMT,C-CMT 的性能,我们统计了在 Financial1 负载下的命中率分布及平均响应时间,结果如表 3 和表 4 所示,其中 Hit ratio 为总的缓存命中率,H-CMT hit ratio 为 H-CMT 的命中率,S-CMT hit ratio 为 S-CMT 的命中率,C-CMT hit ratio 为 C-CMT 的命中率.

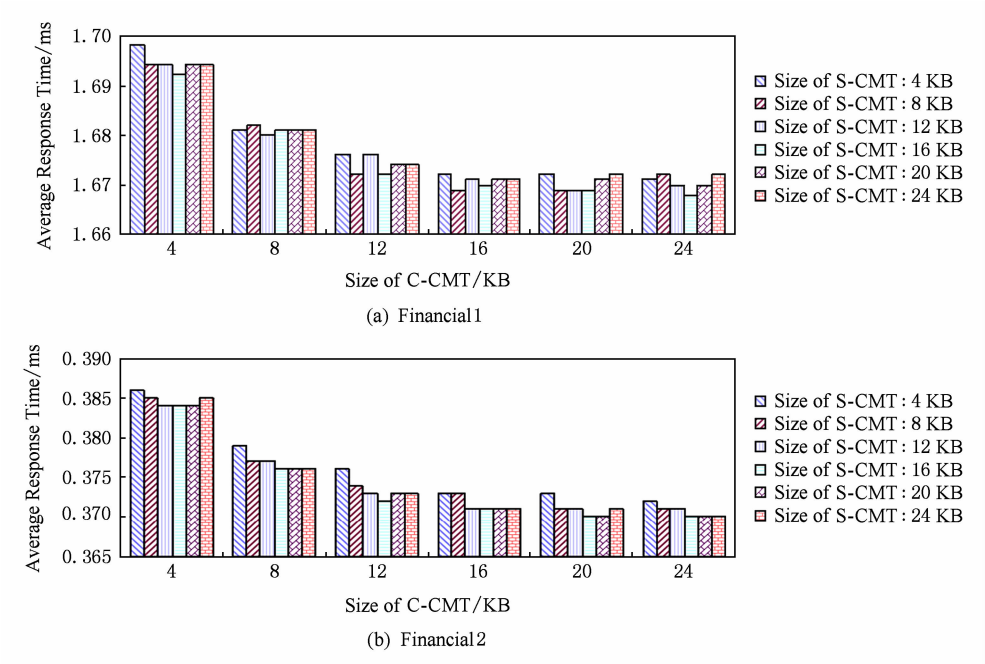


Fig. 4 Average response time under different size of S-CMT and C-CMT

图 4 不同 S-CMT 和 C-CMT 大小下的平均响应时间

Table 3 The Variation of Average Response Time and Hit Ratios of Financial1 (Size of C-CMT is 20 KB)

表 3 C-CMT 的大小为 20 KB 时 Financial1 的平均响应时间和各命中率

Size of S-CMT/KB	Average Response Time/ms	Hit Ratio/%	H-CMT Hit Ratio/%	S-CMT Hit Ratio/%	C-CMT Hit Ratio/%
4	1.672	84.88	48.92	30.21	5.75
8	1.670	85.05	48.69	30.60	5.76
12	1.669	85.11	48.40	30.94	5.77
16	1.669	85.19	48.14	31.22	5.82
20	1.671	85.18	47.75	31.53	5.90
24	1.672	85.10	47.23	31.90	5.97

Table 4 The Variation of Average Response Time and Hit Ratios of Financial1 (Size of S-CMT is 16 KB)

表 4 S-CMT 的大小为 16 KB 时 Financial1 的平均响应时间和各命中率

Size of C-CMT/KB	Average Response Time/ms	Hit Ratio/%	H-CMT Hit Ratio/%	S-CMT Hit Ratio/%	C-CMT Hit Ratio/%
4	1.692	84.77	49.81	30.13	4.84
8	1.681	84.98	49.40	30.47	5.11
12	1.672	85.12	48.93	30.82	5.37
16	1.670	85.17	48.55	31.04	5.59
20	1.669	85.19	48.14	31.22	5.82
24	1.668	85.13	47.69	31.39	6.05

从表 3 中数据可以看到,在 C-CMT 容量不变时,随着 S-CMT 容量的增大,S-CMT 与 C-CMT 的命中率都在提高,H-CMT 的命中率在下降,但变化幅度都不大.从命中率大小来看,H-CMT 的命中率最高,S-CMT 次之,C-CMT 最少,这说明 H-CMT 中存储访问频率较高的映射项,而 C-CMT 中存储的映射项访问频率较低.

从表 4 中数据可以看到,在 S-CMT 容量不变时,随着 C-CMT 容量的增大,总的缓存命中率先增大后减小,但变化并不明显.具体来说,H-CMT 的

命中率逐渐减小,S-CMT 与 C-CMT 的命中率都在提高.表 4 数据还表明,平均响应时间随着 C-CMT 的增大一直在减少,但后续减少基本可以忽略.

2) C-CMT 的剔除策略的阈值配置

在 H-CMT 为 28 KB,S-CMT 为 16 KB,C-CMT 为 20 KB 的缓存空间配置下,我们研究了 C-CMT 中的阈值大小配置问题.我们设计了一组仿真实验,在实验中阈值大小依次取 0,100,300,512.阈值取 0 表示 C-CMT 满时,选择包含最多映射项的簇进行剔除;阈值取 512 表示 C-CMT 满时,选择最近最少访问的簇进行剔除.仿真 Trace 选择具有代表性的 Financial1,仿真结果如图 5 所示.从图 5 可以看到,随着阈值的增大,Financial1 的平均响应时间先减少后增大,当阈值取 100 时整体性能达到最优.另外,其他 5 个 Trace 也有相似的结果,因此,后续实验中我们取 C-CMT 的阈值为 100.

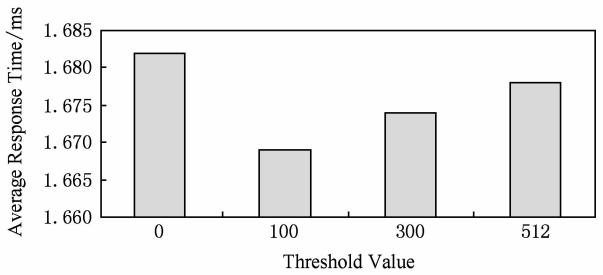


Fig. 5 Average response time of Financial1 under different threshold value of C-CMT

图 5 在 C-CMT 不同预阈值下 Financial1 的平均响应时间

3) S-CMT 的预取映射项个数

同样,在 H-CMT 为 28 KB、S-CMT 为 16 KB、C-CMT 为 20 KB 的缓存空间配置下,我们研究 S-CMT 的预取映射项个数设置问题.我们设计了一组仿真实验,在实验中映射项预取个数依次取 4,8,16,32,64,128,仿真 Trace 选择具有代表性的 Financial1,性能指标选择平均响应时间,仿真结果如图 6 所示.从图 6 可以看出,随着预取映射项个数的增加,平均响应时间先减少后增大,当预取映射项个数为 32 时,平均响应时间为最小.这说明,适当地增加预取映射项个数能有效降低 SSD 平均响应时间整体性能.此外,其他 5 个 Trace 也有相似的结果.因此,S-CMT 预取映射项个数设置为 32.

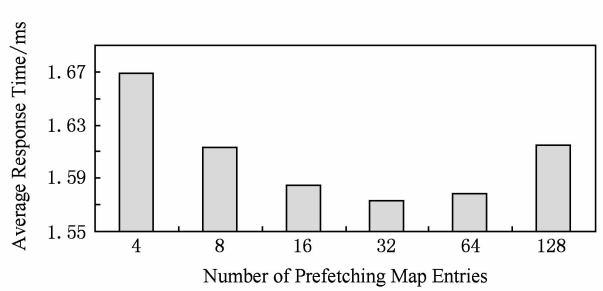


Fig. 6 Average response time of Financial1 under different numbers of prefetching map entries of S-CMT

图 6 在 S-CMT 不同预取映射项个数下 Financial1 的平均响应时间

3.3 CPFTL 与 SDFTL,DFTL 的性能对比分析

1) 总体缓存命中率比较

在不同 Trace 下,CPFTL,SDFTL,DFTL 的总体缓存命中率如图 7 所示.实验结果表明,对于这 6 个 Trace,CPFTL 的总体缓存命中率优于 SDFTL 和 DFTL,平均有 9.88%和 50.59%的改善.需要特别指出的是,WebSearch1 和 WebSearch2 都是以读请求为主,且读请求较大,但请求地址较为随机,重复性很小,故 DFTL 的总体缓存命中率很低;而 SDFTL 和 CPFTL 都采用预取技术,预取多个连续映射项到 S-CMT,使得大的连续请求的后若干个逻

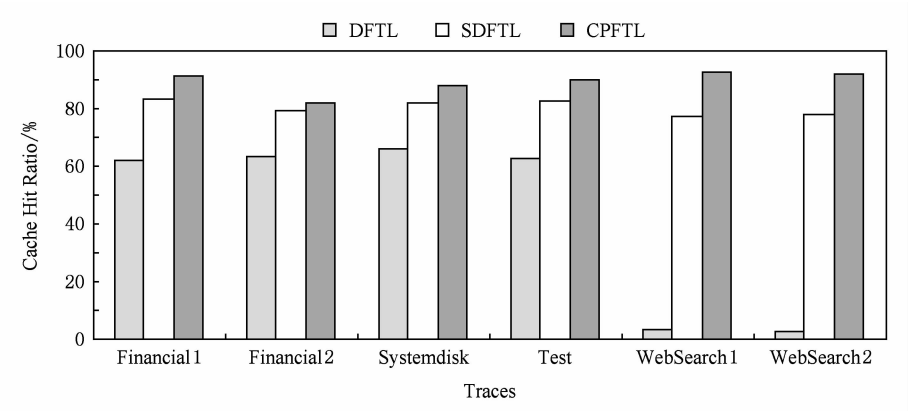


Fig. 7 Cache hit ratio of CPFTL, SDFTL and DFTL under different Traces

图 7 CPFTL,SDFTL,DFTL 在各 Trace 下总的缓存命中率

辑页都能在 S-CMT 命中,从而大幅度提高总体缓存命中率.

2) 地址转换页操作次数比较

CPFTL,SDFTL,DFTL 都采用页级地址映射机制,缓存未命中会造成额外的转换页读写操作开销,因此,地址转换页的操作次数能体现 CPFTL 缓存机制的效果. 在不同 Trace 下,图 8 给出的是 CPFTL,SDFTL,DFTL 的地址转换页操作次数比较. 为方便比较,将 CPFTL,SDFTL,DFTL 的地址转换页操作次数进行归一化处理. 实验结果表明,对于不同的负载类型,CPFTL 的地址转换页操作次数都优于 SDFTL 和 DFTL,平均有 50.62%和 82.87%的改善,显著地减少了地址转换页的读写操作次数.

3) 平均响应时间和块擦除次数比较

系统的平均响应时间和块擦除次数是衡量闪存存储性能的重要指标. 在不同 Trace 下,图 9 给出的

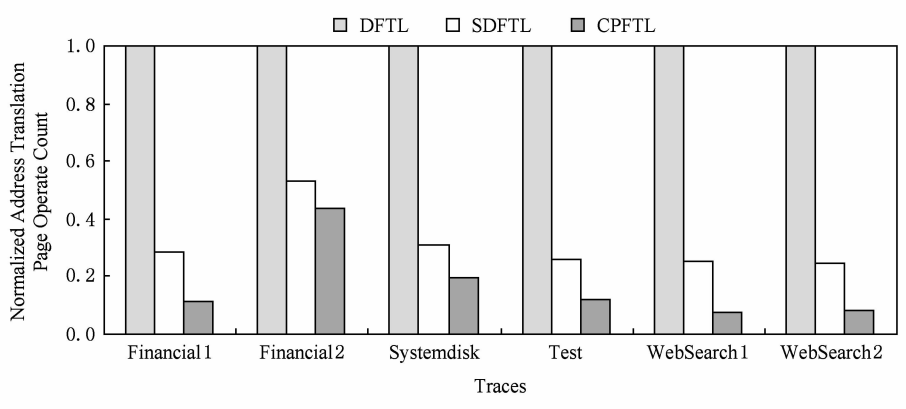


Fig. 8 Address translation page operate count of CPFTL, SDFTL and DFTL under different Traces

图 8 CPFTL,SDFTL,DFTL 在各 Trace 下地址转换页操作次数

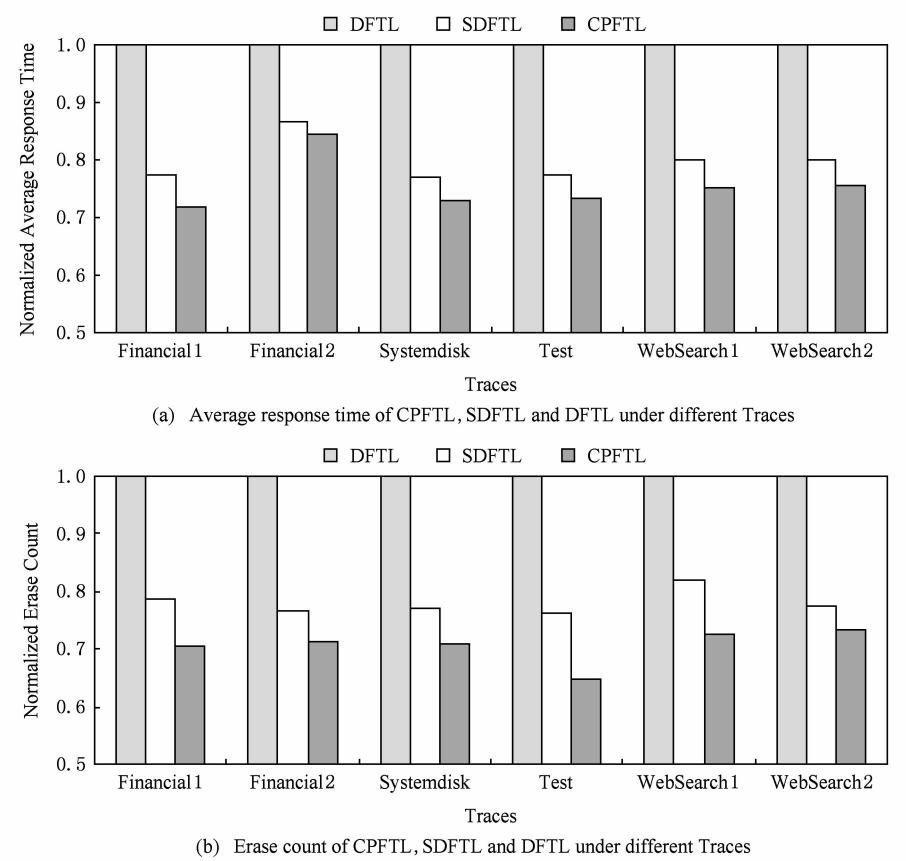


Fig. 9 Average response time and erase count of CPFTL, SDFTL and DFTL under different Traces

图 9 CPFTL,SDFTL,DFTL 在各 Trace 下平均响应时间和块擦除次数

是 CPFTL,SDFTL,DFTL 的平均响应时间和闪存擦除次数比较. 为方便比较,将 CPFTL,SDFTL,DFTL 的平均响应时间和块擦除次数进行归一化处理. 实验结果表明,对于不同的负载类型,CPFTL 的平均响应时间和闪存擦除次数都优于 SDFTL 和 DFTL,相比 SDFTL 分别有 8.25%和 9.26%的改善,相比 DFTL 分别有 24.43%和 29.35%的改善.

4 结束语

本文提出了一种基于分类策略思想的新型聚簇页级闪存转换层算法——CPFTL. 首先,它将地址映射缓存分为热映射表缓存、连续映射表缓存、冷映射表缓存,分别用来缓存访问频繁的映射项、连续请求的映射项和访问频率低下的映射项,从而有效地提升了对各类请求的处理能力,以及提高地址映射缓存的空间利用率;其次,为利用连续请求的空间本地性,CPFTL 预取 32 个连续的映射项到 S-CMT 中,提高它对连续请求的处理能力;最后,CPFTL 的 C-CMT 采用聚簇思想,即将属于同一转换页中的映射项进行聚簇,按簇进行 LRU 管理,当 C-CMT 满时,根据簇内包含的映射项个数和簇在 LRU 队列中的位置来选取合适的簇剔除到闪存中去. 实验中使用了一系列不同的负载对 CPFTL 的性能进行评估. 实验结果显示,平均而言,CPFTL 相比经典的 DFTL 算法,总体缓存命中率提高 50.59%,响应时间减少 24.43%,地址转换页操作次数减少 82.87%,闪存块擦除次数减少 29.35%;相比最新的 SDFTL 算法,总体缓存命中率提升 9.88%,响应时间减少 8.25%,地址转换页操作次数减少 50.62%,闪存块擦除次数减少 9.26%. 但 CPFTL 还有改进之处,例如,可以设计更加准确的请求类型识别方式,或者能够根据负载的特点自适应地调整其参数设置等,这些问题都留待下一阶段的研究工作进行解决.

参 考 文 献

[1] Lu Youyou, Shu Jiwu. Survey on flash-based storage systems [J]. Journal of Computer Research and Development, 2013, 50(1): 49-59 (in Chinese)
(陆游游, 舒继武. 闪存存储系统综述[J]. 计算机研究与发展, 2013, 50(1): 49-59)

[2] Wu Suzhen, Chen Xiaoxi, Mao Bo. GC-RAIS: Garbage collection aware and redundant array of independent SSDs [J]. Journal of Computer Research and Development, 2013, 50(1): 60-68 (in Chinese)
(吴素贞, 陈晓熹, 毛波. GC-RAIS: 一种基于垃圾回收感知的固态盘阵列[J]. 计算机研究与发展, 2013, 50(1): 60-68)

[3] Kim S, Kim T. QLRU: NCQ-aware write buffer management algorithm for SSDs [J]. Electronics Letters, 2013, 49(17): 1079-1081

[4] Wang Jiangtao, Lai Wenyu, Meng Xiaofeng. Flash-based database: Studies, techniques and forecasts [J]. Chinese Journal of Computers, 2013, 36(8): 1549-1567 (in Chinese)
(王江涛, 赖文豫, 孟小峰. 闪存数据库: 现状、技术与展望[J]. 计算机学报, 2013, 36(8): 1549-1567)

[5] Ma Dongzhe, Feng Jianhua, Li Guoliang. LazyFTL: A page-level flash translation layer optimized for NAND flash memory [C] //Proc of the SIGMOD 2011 and PODS 2011. New York: ACM, 2011: 1-12

[6] Gupta A, Kim Y, Urgaonkar B. DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings [C] //Proc of the 14th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2009: 229-240

[7] Qin Zhiwei, Wang Yi, Liu Duo, et al. A two-level caching mechanism for demand-based page-level address mapping in NAND flash memory storage system [C] //Proc of the 17th IEEE Real-Time and Embedded Technology and Applications Symp. Los Alamitos, CA: IEEE Computer Society, 2011: 157-166

[8] Choudhuri S, Givargis T. Performance improvement of block based NAND flash translation layer [C] //Proc of the 5th Int Conf on Hardware/Software Codesign and System Synthesis. New York: ACM, 2007: 257-262

[9] Choudhuri S, Givargis T. Deterministic service guarantees for NAND flash using partial block cleaning [C] //Proc of the 6th IEEE & ACM/IFIP Int Conf on Hardware/Software Codesign and System Synthesis. New York: ACM, 2008: 19-24

[10] Qin Zhiwei, Wang Yi, Liu Duo, et al. Demand-based block-level address mapping in large-scale NAND flash storage systems [C] //Proc of the 8th IEEE & ACM Int Conf on Hardware/Software Codesign and System Synthesis. Los Alamitos, CA: IEEE Computer Society, 2010: 173-182

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

[12] 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 Trans on Embedded Computing Systems, 2007, 6(3): 1-27

[13] Kang J U, Jo H, Kim J S, et al. A superblock-based flash translation layer for NAND flash memory [C] //Proc of the 6th ACM & IEEE Int Conf on Embedded Software. New York: ACM, 2006: 161-170

[14] Lee S, Shin D, Kim Y. LAST: Locality-aware sector translation for NAND flash memory-based storage systems [J]. ACM SIGOPS Operating Systems Review, 2008, 42 (6): 36-42

[15] Chen Jinzhong, Yao Nianmin, Cai Shaobin, et al. Page write-related scheme for flash translation layer [J]. Journal on Communications, 2013, 34(6): 76-84 (in Chinese)
(陈金忠, 姚念民, 蔡绍滨, 等. 基于页面写相关的闪存转换层策略[J]. 通信学报, 2013, 34(6): 76-84)

[16] Yao Yingbiao, Shen Zuobing. An improved DFTL algorithm based on sequential cache and second level cache [J]. Journal of Computer Research and Development, 2014, 51 (9): 2012-2021 (in Chinese)
(姚英彪, 沈佐兵. 基于连续缓存和二级缓存的 DFTL 改进算法[J]. 计算机研究与发展, 2014, 51 (9): 2012-2021)

[17] Jiang S, Zhang L, Yuan X, et al. S-FTL: An efficient address translation for flash memory by exploiting spatial locality [C] //Proc of the 27th IEEE Symp on Mass Storage Systems and Technologies (MSST). Los Alamitos, CA: IEEE Computer Society, 2011: 1-12

[18] Hu Yang, Jiang Hong, Feng Dan, et al. Achieving page-mapping FTL performance at block-mapping FTL cost by hiding address translation [C] //Proc of the 26th IEEE Symp on Massive Storage Systems and Technologies (MSST). Los Alamitos, CA: IEEE Computer Society, 2010: 1-12

[19] Zuan Xiaoying, Tang Xian, Liang Zhichao, et al. OAFTL: An efficient flash translation layer for enterprise application [J]. Journal of Computer Research and Development, 2011, 48(10): 1918-1926 (in Chinese)
(慕晓颖, 汤显, 梁智超, 等. OAFTL: 一种面向企业级应用的高效闪存转换层处理策略[J]. 计算机研究与发展, 2011, 48(10): 1918-1926)

[20] Lee Y, Jung T, Shin I. Demand-based flash translation layer considering spatial locality [C] //Proc of the 28th Annual ACM Symp on Applied Computing. New York: ACM, 2013: 1550-1551

[21] Xie Xuchao, Song Zhenlong, Li Qiong, et al. WAPFTL: A workload adaptive page-level flash translation layer with prediction [J]. Computer Engineering and Science, 2014, 36 (7): 1238-1243 (in Chinese)
(谢徐超, 宋振龙, 李琼, 等. WAPFTL: 支持预测机制的负载自适应闪存转换层算法[J]. 计算机工程与科学, 2014, 36 (7): 1238-1243)

[22] Kim Y, Tauras B, Gupta A, et al. FlashSim: A simulator for NAND flash-based solid-state drives [C] //Proc of the 1st Int Conf on Advances in System Simulation. Los Alamitos, CA: IEEE Computer Society, 2009: 125-131

[23] Bucy J S, Ganger G R. The DiskSim simulation environment version 3.0 reference manual, CMU-CS-03-102 [R]. Pittsburgh, PA: Carnegie Mellon University, 2003

[24] Zhang Qi, Wang Linzhang, Zhang Tian, et al. Optimized address translation method for flash memory [J]. Journal of Software, 2014, 25(2): 314-325 (in Chinese)
(张琦, 王林章, 张天, 等. 一种优化的闪存地址映射方法[J]. 软件学报, 2014, 25(2): 314-325)

[25] Liberate M, Shenoy P. OLTP trace [EB/OL]. UMass Trace Repository. [2015-07-06]. http://traces.cs.umass.edu/index.php/storage/storage

[26] Liberate M, Shenoy P. WebSearch trace [EB/OL]. UMass Trace Repository. [2015-07-06]. http://traces.cs.umass.edu/index.php/storage/storage

Yao Yingbiao, born in 1976. Associate professor of the School of Communication Engineering, Hangzhou Dianzi University. Received his PhD degree in communication and information system from Zhejiang University, Hangzhou, China, in 2006. Member of CCF. His main research interests include storage system design, wireless sensor networks, multimedia signal processing, etc.

Du Chenjie, born in 1990. Received his MSc degree in electronics and communication engineering from Hangzhou Dianzi University in 2016. Since 2016, he has been a research assistant of Zhejiang Wanli University. His main research interests include flash-based solid-state drive technique (du526292624@163.com).

Wang Fakuan, born in 1991. Received his BSc degree in electronic and information engineering from Jiaying University in 2014. Since 2014, he has been a MSc candidate in electronic and communication engineering of Hangzhou Dianzi University. His main research interests include SSD technique (469144949@qq.com).