计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
50(1): 49-59, 2013

闪存存储系统综述

陆游游[1]　舒继武[1,2]
[1](清华大学计算机科学与技术系　北京　100084)
[2](清华信息科学与技术国家实验室(筹)　北京　100084)
(luyy09@mails.tsinghua.edu.cn)

Survey on Flash-Based Storage Systems

Lu Youyou[1] and Shu Jiwu[1,2]
[1](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)
[2](*Tsinghua National Laboratory for Information Science and Technology (TNLIST), Beijing* 100084)

Abstract　Flash memory has gained great popularity for its properties of low latency, high parallelism, energy efficient, and small volume. But the direct substitution of flash-based solid state drives for hard disk drives has hidden these properties from the operating system, preventing the operating system from optimizations. In this paper, comparison and analysis are made on the storage systems built on the raw flash, including flash accelerating cards, flash arrays, and flash-based clusters, which have achieved the goals of low latency, high reliability and energy efficiency through hardware interface change, software or controller module refinement, or computation and I/O capacities matching. Based on the comparison of these recent raw flash based storage systems, challenges and design issues are discussed on three aspects. Firstly, performance optimizations with I/O stack redesign are surveyed from the hardware interface, notification mechanisms, the system software refinement or redesign, and new storage interfaces for rich semantics. Secondly, the reliability approaches from the system level are described. Finally, the energy efficiency and the volume gains from the flash based storage are presented. After the discussion, the research works are summarized and the possible research directions are pointed out.

Key words　flash memory; storage system; I/O stack; reliability; energy efficient

摘　要　闪存因其低延迟、高并发、低能耗、体积小等特点受到了广泛关注.首先讨论了简单利用闪存固态盘替换传统磁盘的方式隐藏了闪存特性,限制了软件系统对闪存特性充分利用的不足.然后,分析并比较了现有包括闪存加速卡、闪存阵列、基于闪存的分布式集群系统等基于存储介质直接构建的闪存存储系统的特点,归纳了其通过改变硬件接口、调整软件或控制器管理模块、匹配处理器与I/O处理能力等方式实现系统低延迟、高可靠、低能耗等特性的优化方法.然后重点讨论了闪存存储系统3个方面的关键技术:基于I/O栈调整与重构的存储性能优化、系统级可靠性、体积与能耗.最后总结了闪存存储系统的现状与特点,并指出未来可能的研究方向.

关键词　闪存;存储系统;I/O栈;可靠性;节能

中图法分类号　TP302.1

收稿日期:2012-08-20;**修回日期**:2012-09-12;**网络发表日期**:2012-12-05

基金项目:国家杰出青年科学基金项目(60925006);国家"八六三"高技术研究发展计划基金项目(2012AA011003)

近 30 年来,计算机系统的 I/O 性能与 CPU 处理性能的差距迅速扩大,而且随着多核技术的快速发展,这种差距还将越来越大. 与此同时,社交网络、物联网、移动互联网等多种新型应用的出现,以及高性能计算趋于精细的处理要求,导致数据处理规模的急剧增长. 频繁的 I/O 请求制约了计算机系统所提供的应用处理能力,因而 I/O 能力正成为计算机系统中急需解决的问题.

传统磁盘存储系统为此作出了诸多努力. 磁盘系统通过 RAID 技术发挥多磁盘的并行存取,I/O 请求被分发到多个磁盘设备,通过多设备的并行工作提供聚合带宽. 在设备内部,磁盘系统通过数据分布以及 I/O 调度等多种手段顺序化数据的访问,减少寻道时间,以提高磁盘的吞吐率.

尽管磁盘存储系统有效提升了带宽,但访问延迟难以得到有效解决,依然不能满足延迟要求较高的应用. 同时,由于需要提供多磁盘设备的并行访问的性能,存储系统往往部署了远多于实际容量需求的磁盘数目,系统规模也随之扩大. 系统规模的扩大一方面提高了系统的复杂性,另一方面也提升了能耗.

近年来,随着闪存芯片容量的增加和价格的下降,闪存芯片的部署正从移动设备中走到个人计算机以及大规模数据中心中去. 闪存的低延迟、低能耗、体积小、重量轻、抗震等特点使得闪存在数据中心的应用正受到极大的关注.

1 闪存存储现状

闪存设备主要由闪存介质、闪存控制器两部分构成,如图 1 所示. 闪存操作以闪存页(page)为读写单元,以闪存块(block)为擦除单元,设备内部的介质访问提供了通道(channel)、颗粒封装(package)、颗粒(die)、闪存片(plane)等多个级别的并行[1-2]. 设备内部以多通道方式组织闪存颗粒封装,每个通道上可连接多个颗粒封装,多颗粒封装之间共享传输通道,但可独立执行指令. 每个颗粒封装内部包含两个或多个闪存颗粒,每个颗粒可被独立选中执行指令. 颗粒内部可分为多个闪存片,每个闪存片含有一个闪存页大小的寄存器缓存,用于暂存读写数据. 通过多个级别的指令并行执行,闪存设备充分利用介质的存取性能. 闪存控制器主要负责地址映射、垃圾回收和磨损均衡[3]. 闪存通过异地更新的方式缓解闪存单元写前擦除的延迟. 由于闪存单元的擦除次数有限,闪存控制器通过磨损均衡算法平衡设备内部闪存块之间的擦除次数,同时垃圾回收算法尽量选取擦除效率较高的块回收以控制写放大(write amplification). 闪存控制器既要维护高效的数据地址映射查询,也要均衡设备内部磨损,控制写放大,以提高设备的整体寿命.

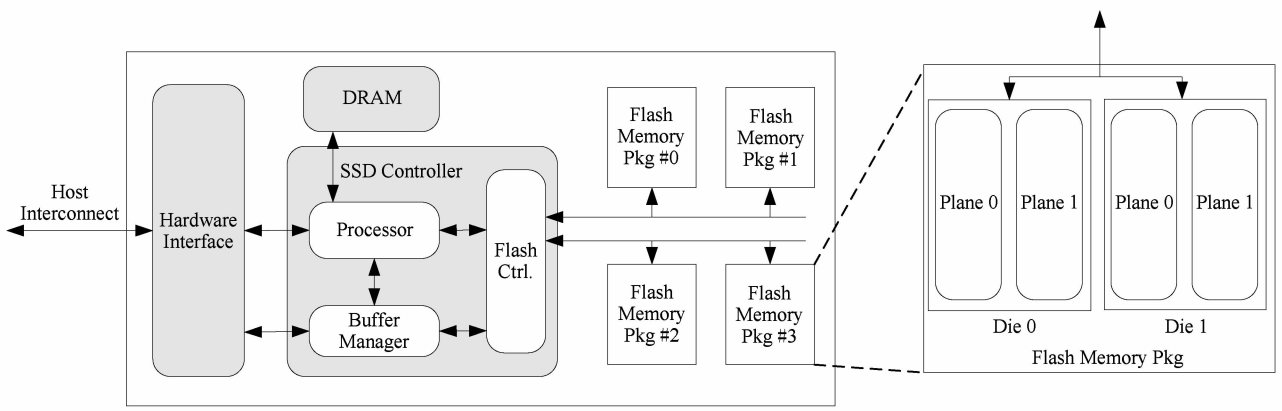


Fig. 1 An illustration of SSD archtecture[1].

图 1 闪存内部结构图[1]

闪存存储主要以固态盘(solid state drive, SSD)以及裸闪存两种形式存在. 固态盘的方式是在设备内部通过闪存转换层(flash translation layer, FTL)的转换可处理 SATA 命令,其外部的使用接口与传统磁盘没有差别. 固态盘与传统磁盘在外部使用上差异极小,固态盘可以简单地替换磁盘,因而固态盘的形式为当前闪存存储的主要形式,包括个人笔记本、服务器、存储阵列等.

然而,固态盘的形式限制了闪存优势的有效发挥. 在硬件接口上,由于闪存内部并发可以有效聚合访问带宽,SATA 接口的标准已经远不能满足要求,硬件接口成为闪存存储系统的瓶颈. 在存储子系

统上,以文件系统为代表的系统软件在存储管理上大多以磁盘为假设进行优化,较少考虑闪存特性,闪存优势难以得到充分利用.现有在固态盘上构建的软件系统存在的不足主要体现在以下几个方面:

1) 冗余工作

文件系统对文件的管理既包括文件目录树的维护信息,也包括对于文件逻辑块到存储设备物理块的映射,同时也包括存储设备空间管理.闪存设备内部需要提供地址映射以实现数据块的异地更新.文件系统中从文件逻辑块到设备物理块的映射,与闪存设备转换层中逻辑地址到物理地址的映射,构成了闪存系统的双层映射[4-5].双层映射的出现既增加了元数据的管理与存储开销,也可能造成双层的优化之间存在冲突.

2) 语义缺失

闪存设备转换层封装了对闪存介质的操作,向上层提供了统一的块设备接口.设备内部接收到以页面为单位的数据,却不能理解数据页面之间的关系,也就难以优化数据页面的读写顺序与分布[5].除此之外,闪存设备也难以感知文件系统的操作,不能及时处理,以至于影响下次关联操作[6].例如,文件系统对于文件的擦除操作仅修改对应元数据,而数据的擦除操作直到下次写操作才被闪存设备感知,影响闪存设备的垃圾回收效率.

3) 特性错失

闪存设备提供了异地更新的特性,数据的更新会写入新分配的闪存页,旧闪存页一直保留到垃圾回收的时刻.而文件系统、数据库管理系统或者其他上层应用为提供操作的原子性,通常使用 WAL (write ahead logging)的方式,先申请新的存储空间记录数据更新作为 redo 日志,然后在原地进行数据的更新.这样的原子性以两次写操作的代价保证,既增加了延迟,也损害了闪存寿命[7-9].同样,文件系统为容错而采用多版本信息也不能直接索引到旧的闪存页[10].

4) 分工欠妥

闪存设备的容量不断增加,设备内部的处理能力以及内存的需求也在不断增长.这既提升了存储设备的制造成本,也提高了设备的能耗,影响闪存的优势.而主机在等待闪存设备读写时,主机 CPU 与内存有不少空闲.另外,多层的处理与缓存也造成访问延迟与数据可靠性的问题.主机与闪存设备在数据处理与缓存以及数据存储方面应寻求最佳的职能分工,以达到整体系统的最优.

简单地用固态盘替换磁盘的存储系统不能充分发挥闪存的性能,因而闪存存储系统倾向于直接管理闪存介质,重新设计或调整软硬件结构,优化存取路径,丰富存储语义,增强特性功能,以提升闪存存储系统的整体性能[11-12].

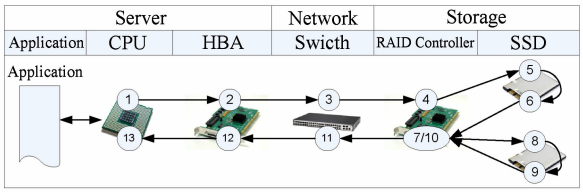
2 闪存存储系统

基于闪存介质直接构建闪存存储系统是当前的研究热点,构建形式多样,包括闪存加速卡[13]、闪存阵列[14-16]以及近年来提出的基于闪存的分布式集群系统[17-21].

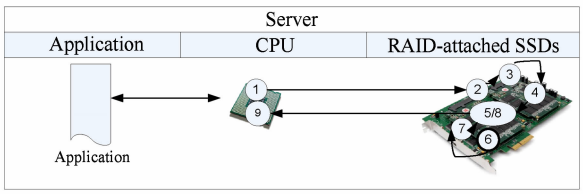
2.1 闪存加速卡

闪存加速卡用于服务器中,主要用于本地数据的缓存处理.相比于传统的 PCIe 固态盘 RAID 卡,以 FusionIO 为代表的厂商推出了 PCIe 闪存卡,有效地降低了存储访问延迟,扩大了访存带宽.

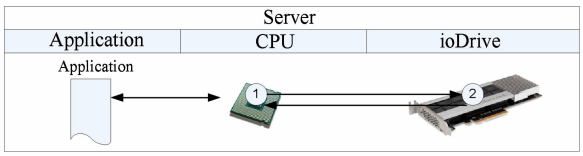
磁盘阵列的访问需要经过主机 HBA、网络交换、阵列控制器等访问模块,如图 2(a)所示;PCIe 固态盘 RAID 卡也需要经过 RAID 控制器、元数据处理模块后才能访问数据,如图 2(b)所示;FusionIO 采用裸闪存 PCIe 卡,主机直接访问闪存芯片,缩减了 I/O 传输路径,提供了极低的访问延迟[13].



(a) I/O path of the disk array



(b) I/O path of the PCIe SSD



(c) I/O path of the FusionIO ioDrive

Fig. 2 I/O path comparison of the disk array, the PCIe SSD and FusionIO ioDrive[13].

图 2 磁盘阵列、PCIe 固态 RAID 卡及 FusioIO ioDrive 的 I/O 路径比较[13]

FusionIO 在主机端实现了闪存转换层,称为 VSL(virtualized storage layer)[22]. VSL 实现了闪存转换层所需要的映射机制、磨损均衡、可靠性等多种功能. 相比于传统闪存固态盘在设备内部采用嵌入式处理器的处理方式相比,VSL 能充分利用主机的冗余计算与缓存能力,大大提高了处理性能. 此外,利用闪存异地更新的特性,依据 VSL 中映射转换的日志机制,VSL 实现了多闪存页的原子写(atomic write)操作,从而避免了上层应用为提供多数据页更新所花费的重复写,整体上延长闪存设备寿命[7]. 利用 VSL 中提供的空间管理和原子写特性,DFS[4]基于 VSL 重新设计闪存文件系统,提供了很好的性能,降低了文件系统的复杂性.

基于闪存介质构建的闪存加速卡在软件层实现闪存介质管理,并与软件系统结合,提供灵活的机制,大大提升了单机存储子系统的效率. 这类加速卡引起业界关注,Intel 及众多存储厂商制定了 NVMe 标准[23],以提供统一接口标准支持.

2.2 闪存阵列

闪存加速卡难以实现存储容量的扩展及存储共享. 类似于传统的磁盘阵列设计,多家存储厂商推出了闪存阵列,其中包括采用固态盘与传统阵列控制器的演进式设计,以及采用闪存芯片与全新阵列控制器的革新式设计.

Violin Memory 与 Toshiba 合作设计闪存阵列,通过 Violin 闪存阵列控制器对闪存介质颗粒级别进行优化[14]. Pure Storage 根据闪存特性完全重新设计控制器控制逻辑,充分发挥闪存的并行性,采用阵列内全局的磨损均衡算法,以提供全局更好的性能与寿命[15]. 刚被 EMC 收购的以色列闪存阵列公司 XtremeIO 认为传统阵列控制器主要考虑磁盘特性的优化,例如顺序 I/O 等. 针对这一问题,XtremeIO 设计了新型闪存阵列控制器,发挥闪存较好的随机访问性能,并利用闪存异地更新特性在阵列内实现了删冗、快照/镜像、精简配置等多项功能[16].

2.3 基于闪存的分布式集群系统

2.3.1 FAWN[17]

FAWN(a fast array of wimpy nodes)是卡内基梅隆大学(Carnegie Mellon University,CMU)基于闪存介质构建的可扩展、低能耗、高性能的集群系统. 与闪存加速卡和闪存阵列仅关注于 I/O 子系统的性能和可靠性的设计不同,FAWN 从集群整体设计的角度考虑闪存与处理器的匹配以降低系统整体能耗.

在数据密集型计算环境下,I/O 速度远不及 CPU 处理速度. CPU 很多时候处于空闲等待状态,同时 CPU 的能耗随着频率的提高呈现超线性增长. FAWN 采用低频低能耗 CPU 与闪存存储相匹配,提供数据密集型计算的集群系统,提高系统各组成部分的利用率,降低能耗. FAWN 实现了键值存储系统,以日志的方式进行更新,实现闪存的异地更新. FAWN 可提供每焦耳高达 364 次查询请求,相比于桌面硬盘系统的每焦耳 1.96 次查询性能提升数百倍[17].

2.3.2 Gordon[18]和 Moneta[19]/Onyx[20]

Gordon[18]系统是加州大学圣地亚哥分校(University of California, San Diego,UCSD)设计的,与 FAWN 类似的采用低频处理器以构建低能耗数据中心的集群系统. 与 FAWN 关注键值存储系统的高性能与低能耗不同,Gordon 主要工作在于设计闪存转换层与匹配处理器与闪存芯片的性能和能耗. Gordon 闪存转换层在发挥闪存芯片间的并发特性上采取了多种措施,包括地址动态映射、多物理页组合成大物理页、以及同时采用并发与流水机制. Gordon 在单板上集成了 256 GB 的闪存和 2.5 GB 的 DRAM,一个 16 个单板的封装节点可提供 4TB 的闪存存储和高达 14.4 GBps 聚集带宽[18]. Gordon 系统已经被用于 San Diego 超级计算中心,主要应用在天体物理、基因组测序等数据密集型计算领域,并将其部署在 TeraGrid 中. 当完全配置和部署时,Gordon 将拥有 300TB 闪存存储容量,由 1 024 块英特尔 710 系列高性能固态存储组成,将拥有 16 384 个处理器内核,理论性能峰值达到 340Tfps[24].

Moneta[19]和 Onyx[20]是 UCSD 基于 PCM 构建的存储系统. Moneta 认为软件延迟占低延迟存储介质整体访问延迟的比例越来越大,因此软件系统需要重新考虑以降低整体访问延迟. Moneta 采用 DRAM 模拟 PCM,该系统绕开了操作系统中 IO 调度等操作过程,通过 spin 的等待而不是中断调用等对系统的修改,能够减少 60%的访问延迟,提升 18 倍的带宽[25]. Moneta 是采用 PCIe 接口的模拟 PCM 存储系统,Onyx 是对 Moneta 改进的第 2 代原型. Onyx 替换 PCIe 接口为 DIMM 接口,并采用了定制的 PCM 模块,进一步提升了系统性能.

2.3.3 RAMCloud[21]

在采用非易失性存储器件构建存储系统之外,加州大学伯克利分校(University of California, Berkeley,UC Berkeley)提出采用 DRAM 构建分布式存储系统 RAMCloud[21]. RAMCloud 利用服务器集群中服务器 DRAM 构建统一命名空间的存储系

统,所有数据存储在 DRAM 中,磁盘仅用作备份.相比于传统的磁盘存储系统网络访问远端磁盘 5~10 ms 的延迟,RAMCloud 的访问延迟可达到 5μs,提升 100~1 000 倍的性能[21].尽管闪存存储系统"FlashCloud"可提供略低于 RAMCloud,但在查询吞吐量要求越高的情况下,RAMCloud 总体拥有成本越低.

持久性与可用性是 RAMCloud 最为关心的问题[26].RAMCloud 采用磁盘作为备份存储,在内存中维护日志记录,记录每次更新,通过聚集写(bulk write)刷入磁盘,以提供持久性.在可用性上,RAMCloud 在不同的服务器上放置 3 个副本,且不同的数据副本分散到不同的服务器.在出现故障时,可以并行地从其他节点恢复,实验原型可在 1~2 s 的时间内恢复 64 GB 的数据.

能耗是 RAMCloud 另一个引起争论的问题.RAMCloud 认为在以每操作焦耳数为衡量指标时,RAMCloud 因为性能的优势可以提供高吞吐量应用下的低能耗[27].

2.3.4 Mangix

清华大学的 Mangix 是基于固态存储设计的分布式存储系统.Mangix 分离了文件系统的目录树维护与空闲空间管理功能,将空闲空间管理与管理闪存的闪存转换层相结合,重新优化软件层次,以减少冗余工作.

闪存转换层在软件层实现,通过感知文件系统的文件语义优化对闪存的数据分布与垃圾回收策略.闪存转换层直接管理闪存介质,减少 I/O 路径冗余,通过数据分布发挥闪存的高效并发.

Mangix 采用分布式对象文件系统管理,数据及元数据以对象形式存储于集群中的闪存介质上.闪存转换层以远程过程调用的方式相互通信,实现分布式原子性操作以及多节点均衡等特性.

2.4 小 结

上述系统分别在低延迟、低能耗、多设备磨损均衡等多方面考虑对软件重构或调整,基于存储介质构建新型的闪存存储系统,其特点如表 1 所示.闪存加速卡、闪存阵列摒弃传统固态盘的方式,根据闪存特性调整闪存管理模块,提升了存储子系统的效率,但扩展性不够理想.FAWN 与 Gordon 通过匹配I/O与 CPU 计算能力,从集群的角度构建分布式节能系统,其在多节点间闪存特性协作上有进一步研究的可能.基于 PCM 的 Moneta/Onyx 同样以重构软件系统的方式实现了极低延迟的 I/O 访问.RAMCloud 基于 DRAM 构建了分布式存储系统,降低存储系统的 I/O 延迟.Mangix 通过软件直接管理闪存介质,通过软硬件协作构建基于固态存储的分布式集群存储系统.

Table 1 Comparison of Flash-Based Storage Systems

表 1 闪存存储系统比较

Storage System	Storage Media	Distributed	I/O Stack Redesign	Latency	Energy	Semantic Aware	Global Wear Leveling	Scalability
FusionIO	Flash	No	Yes	Low	Medium	No	No	Low
FlashArray	Flash	No	Yes	Medium		No	Yes	Medium
FAWN	Flash	Yes	Yes	Medium	Low	No	No	High
Gordon	Flash	Yes	Yes	Medium	Low	No	No	High
Moneta/Onyx	PCM	No	Yes	Extremely Low	Low	No	No	Low
RAMCloud	DRAM	Yes	Yes	Low		No	No	High
Mangix	Flash	Yes	Yes	Medium	Medium	Yes	Yes	High

Notes: The latency of distributed systems includes the network latency. And the energy of FusionIO includes the host energy.

总之,闪存的出现给原有针对磁盘优化的软件系统提出了挑战.为了充分发挥闪存低延迟、高并发、异地更新等特性,多数闪存系统重新设计或者修改了阵列控制器或软件模块,以充分发挥闪存优势.

3 闪存存储系统关键技术

闪存介质访问呈现低延迟、读写不对称的特点.与磁盘介质访问相比,随机性能提升很多.传统针对磁盘优化设计的软件系统直接用于闪存系统时,一方面带来了软件系统不必要冗余功能,另一方面隐藏了闪存可能带来优势.闪存的可靠性与传统磁盘设备不同,闪存单元的擦除次数有限,可靠性随着擦除次数的增加而下降,闪存的可靠性与闪存设备负载相关.此外,闪存体积小、能耗低,给低能耗的系统设计带来了机遇.

针对闪存的特性,当前的闪存存储系统的研究主要集中在以下3个方面.

3.1 基于I/O栈调整与重构的存储性能优化

在新型存储系统中,存储介质的访问延迟越来越低,相对应的软件开销所占比例越来越高.报告[28]指出,传统磁盘存储系统中软件开销占0.3%,PCIe闪存卡系统中软件开销占比21.9%,而随着非易失性器件的发展,预计软件开销所占比率将高达94.09%,如图3所示:

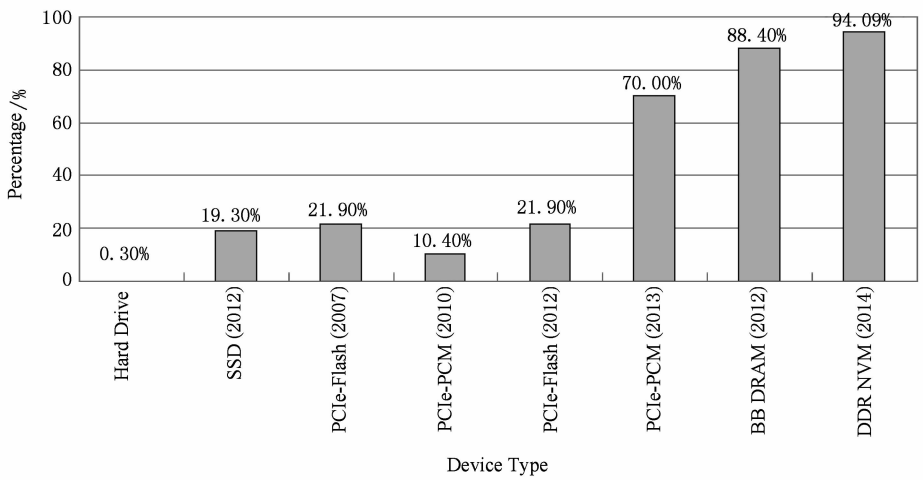


Fig. 3 Trend of software contribution to latency[28].

图3 软件开销占比趋势[28]

软件开销一方面因软件抽象的层次化设计所致,层次化设计中I/O存储路径较长[12];另一方面针对磁盘寻道所作的顺序化的优化设计也带来了不必要的软件开销[29].

除了软件开销导致的时延之外,利用闪存特性提供特定的功能[7-9],通过感知上层文件语义更好地布局闪存数据等都是当前研究的热点问题[6].

3.1.1 硬件接口与通知机制

硬件接口的带宽限制影响了闪存设备的性能.磁盘的从设备缓存到磁介质的数据读写需要经过磁头的寻道与定位.机械部件的旋转速度尽管可达15 000转,但读写带宽仍不能显著提升.如表2所示,目前希捷企业级磁盘15K.3的读写带宽为202 MBps,其使用SAS 6.0 Gbps的硬件接口,可见磁盘内部的读写是带宽瓶颈.在闪存设备中,闪存并发性能可达到极高的数值,硬件接口成为瓶颈.闪存单芯片可提供330 MBps读带宽、33 MBps写带宽.闪存设备内部通过多通道并发,以及芯片间流水指令执行,可提供外部极高的并行访问带宽.如表2所示,Intel X25-E的读带宽达到250 MBps,SATA3 Gbps接口成为瓶颈.FusionIO采用PCIe的硬件接口,ioDrive Octal可达到6 GBps的读带宽和4.4 GBps的写带宽.

Table 2 Hardware Interface and R/W Performance of Enterprise Drives

表2 几种主流高端存储设备的硬件接口与读写性能

Storage Disk Model	HW Interface	Read Bandwidth/MBps	Write Bandwidth/MBps	Read Latency/ms	Write Latency/ms
Seagate Savvio 15K.3[30]	SAS 6 Gbps	202	202	2	2
Intel X25-E[31]	SATA 3 Gbps	250	170	0.075	0.085
FusionIO ioDrive Octal[32]	PCIe x16 Gen2.0	6 000	4 400	0.030	0.030

除带宽之外,访问延迟也需要新型高速的硬件接口.FusionIO通过PCIe接口直接使用DMA在闪存与内存之间传输数据,从而避免SAS/SATA HBA与RAID等模块的处理延迟.如图1所示,FusionIO可减少2~8个模块的处理延迟[13].UCSD在PCM原型系统中升级Moneta的PCIe接口到Onyx的DIMM接口,进一步降低了访问延迟[20].

新存储器件访问延迟的降低使得传统基于中断的软硬件通知机制开销增大.在传统方式中,数据读写命令发送给设备之后,可由DMA进行数据传输,

主机可继续执行指令,而不必等待数据的完成.设备完成数据传输之后,通过中断指令通知主机 CPU 以进行后续处理.在新存储器件出现后,低延迟访问会造成频繁的中断,上下文切换的代价已超出循环等待的代价.文献[33]认为在新型存储系统中 I/O 完成采用轮询方式会优于中断方式,I/O 请求可采用同步完成的机制.UCSD 的 Moneta 原型系统也采用了 spin 等待的方式检查 I/O 请求是否完成,以避免频繁中断带来高昂的上下文切换代价[19].

3.1.2 软件存取路径

在 I/O 路径上,因软件的模块化需求,软件系统对多层进行了抽象,包括虚拟文件系统层、块设备层以及 SCSI 驱动层.在软件的各层次中,由具体的文件系统、设备、驱动进行注册以实现正确的数据传输.软件层次屏蔽了多层异构特性,但也导致了不必要的处理与转换.

在设备驱动上,FusionIO 直接使用 DMA 通过 PCIe 接口在闪存和内存之间传输数据[13],从而避免了传统 SCSI 的 3 层传输模型.SCSI 的 3 层传输模型命令之间的转换及处理既带来了延迟开销,也限制了命令集的扩展.

在块设备层中,现有的 IO 调度策略都是基于磁盘存取顺序性等问题.因而在固态盘的使用中,不少人提出使用 noop 的调度策略,即不进行 I/O 的调度.闪存访问也有独特的特征,读写性能不对称会导致读操作会被写操作引入很大的延迟,而且读写请求不能充分利用闪存设备内部的并发特性将导致带宽浪费.文献[29]提出闪存设备上的 FIOS 调度算法,该算法通过选取合适的时间片,并分离读写之间的干扰,既利用了闪存设备的并发特性,也保证了访问请求的公平性.

在文件系统层,Princeton 研究人员与 FusionIO 合作,在 FusionIO 设备上实现了新型闪存文件系统,该文件系统利用了 FusionIO 的 VSL 管理闪存存储空间的功能,从而避免了文件系统空间管理与闪存设备 FTL 管理的冗余管理开销[4].Wisconsin 大学的研究人员提出了 nameless writes 的做法,通过文件系统与闪存设备间新的软件程序接口的方式,由闪存设备自主选择数据写的物理地址,然后通知文件系统进行地址的更新记录[5].nameless writes 的做法减少了重复的映射管理开销,也提供了设备内部进行垃圾回收、磨损均衡等操作的灵活性.

在应用层,UCSD 基于 PCM 原型系统 Moneta 实现了在用户态直接进行数据存取的系统 Moneta-D[34].Moneta-D 通过硬件实现了权限验证,以及从设备通知应用程序机制的支持,实现了用户程序与存储设备间的直接 IO 请求,从而避免了用户态与内核态切换开销以及文件系统权限验证的开销.

3.1.3 软件程序接口

系统软件与存储设备的软件程序接口仅有两个基本操作接口:READ 和 WRITE.闪存介质的状态更新不可逆,更新的物理页必须在擦除完成之后才能进行,因而多数采用异地更新.异地更新后数据的旧版本通过垃圾回收进行空间的回收.垃圾回收时选择擦除的单元是闪存块,因为需要将闪存块中有效闪存页移动到新的闪存块后才能进行闪存块的擦除.垃圾回收时有效闪存页的移动是写放大最主要的来源.在文件删除时,传统文件系统通过修改元数据的方式删除文件,并不对数据页进行标识.闪存设备直到该逻辑页被写入新数据才能感知该闪存页被置为无效,导致设备保持的有效页的数量远大于实际系统的有效页(很多已删除的数据页在闪存设备内仍然被错误地认为有效),进而导致写放大,闪存性能和寿命均受到影响.TRIM 接口的引入提供了数据显式删除的语义.文件系统通过 TRIM 命令通知闪存设备已经删除的数据页的范围,使得设备可以及时置无效位.除 TRIM 命令外,文献[6]提出 PTRIM 和 EXISTS 的接口命令.由于 TRIM 命令仅在设备缓存中提示数据页的无效,在掉电情况下显式删除命令不能确保完成.PTRIM 采用持久性删除的语义,解决 TRIM 在掉电情况下失效的问题.EXISTS 命令用于检查数据页的存在性.通过 PTRIM 与 EXISTS 命令,文件系统可以通过 FTL 来管理存储空间.

新的软件程序接口不仅用于文件系统对闪存设备的信息通知,也用于闪存设备对上层系统提供新的增强功能,主要集中在用闪存支持原子写的研究上.Atomic write FTL 利用 log block 记录多个物理页的更新,在每个更新页的元数据(spare area)区域记录事务号,通过 *AtomicWriteStart*()和 *AtomicWriteCommit*()的调用实现原子操作的起始与提交操作[35].Atomic write FTL 主要用于移动设备环境下,设备以裸 Flash 的形式接入系统,由闪存文件系统直接管理,也就是闪存文件系统实现了 FTL 的功能.OSU 的研究人员与 FusionIO 合作,提出了基于 FusionIO VSL 的 atomic-write 操作接口[7].FusionIO atomic-write 利用 VSL 中映射页以日志形式追加的特点记录映射页的状态改变,通过映射页中标记

位标记事务的完成. TxFlash[8]提出了较为通用的原子写实现机制,在原子写操作内各个物理页的元数据区域记录前继页面的页地址,从而使原子写内的多个物理页构成环,并通过环结构的检测判断该原子操作是否完整. Flag Commit[9]与 TxFlash 类似,改进了 TxFlash 的环结构表示.

3.2 系统可靠性

闪存单元的擦除会削弱闪存单元保存电子的能力,导致存储的数据易被干扰,从而降低可靠性. 闪存单元的擦除次数在 10 000～100 000 次之间,随着闪存密度的增加,闪存单元可靠性变得更低[36]. 为延长闪存设备的使用寿命,FTL 内实现了磨损均衡策略[3]. 与磁盘不同,上层系统的负载与闪存的寿命密切相关,减少上层应用的写数据量也是延迟闪存寿命的一个重要途径. 因而,从系统角度实现负载优化与磨损均衡对闪存系统的可靠性有着重要的意义.

3.2.1 通道内可靠性

在单系统内部,通过 I/O 路径内部上层应用与底层设备的优化提高通道内可靠性. 通道内可靠性主要包括两个方面:系统数据的删冗与压缩、设备内部多版本机制.

CAFTL[37]在 FTL 上通过双层映射检测冗余更新数据,对其进行重复数据删除和合并,减少固态盘写入数据量. CA-SSD[38]将基于内容寻址(content addressable storage, CAS)的思想引入到固态盘中,数值局部性的存在减少了数据写入量,延长了固态盘的寿命. 在数据库系统中,记录更新的大小远小于闪存物理页的大小. 频繁的记录更新会导致大量的闪存物理页的更新,造成闪存寿命问题. 针对这一问题,文献[39]提出了 in-page logging(IPL)的机制,通过在闪存块中保留 8 KB 大小的 log 区域,将记录的更新追加到 log 区域,然后通过延迟合并机制更新数据,减少了闪存页的整体更新. 同一逻辑地址的页面写多数情况下是对原有数据的更新,因而重复率比较高. Delta FTL[40]通过对同一逻辑地址的页面的写进行比较压缩,记录压缩后的数据,合并多个写操作的压缩数据记录到 delta log 区域,减少闪存页的更新. 同时,通过维护映射表的 delta mapping table 以提供最新数据的查询. 闪存阵列厂商 Pure Storage 和 XtremeIO 均通过重复数据删除技术减少阵列的数据写入量[15-16].

从设备内部多版本数据提供数据容错从另一个角度提高了闪存系统的可靠性. 闪存的异地更新未立刻擦除旧版本,文献[10]等提出在闪存文件系统中维护单个文件两个一致性的版本,当最新的版本出错时通过回退到上一个一致性版本的方式提供文件的容错.

3.2.2 跨设备可靠性

闪存系统中设备整体故障出错率虽不及闪存单元磨损出错率,但出于设备在线替换及数据可用性的考虑,在多设备间维护冗余数据以提高可靠性仍有必要. 然而传统 RAID 容错的假设是磁盘之间出错相互独立,该条件在闪存存储系统中不再成立. 由于 RAID 尽可能均衡负载,这导致 RAID 各闪存设备以近似的速度磨损. 当多个闪存设备同时接近寿命极限时,各设备出错概率上升,也造成整体出错概率的上升. Diff-RAID[41]调整校验值(parity)在不同闪存设备间的分布,差异化各设备的磨损速度,从而降低设备同时出错的概率,也就提高了固态盘 RAID 的可靠性. 存储阵列厂商 Pure Storage 阵列内部实现了全局的磨损均衡,以提高阵列的整体寿命.

多设备的可靠性也体现在错误恢复的速度上. RAMCloud 采用了多副本的技术,将数据的副本分布于不同的服务器上,并且不同数据的数据副本均匀分布于集群中的所有服务器,这样在出错恢复时可以通过并发恢复降低错误恢复的时间开销[26].

3.2.3 混合系统可靠性

写入数据量对磁盘可靠性影响微乎其微,在磁盘与固态盘的混合系统中将写数据尽可能分布于磁盘介质上,可以有效减少闪存介质所承受的数据写入量. 文献[42]把磁盘用作固态盘的缓存,通过以追加写的形式顺序地在磁盘上写入更新的数据. 由于数据的时间局部性,有很大一部分的数据存在覆盖写. 通过延迟将数据更新到固态盘的做法可以减少大量的数据写. I-CASH 是利用固态盘作为磁盘缓存的混合存储系统,其将读操作频繁但写操作很少发生的数据缓存在固态盘中,而将对固态盘中缓存数据的更新变化以 delta 的形式追加写到磁盘中,以减少对固态盘的写操作[43].

3.3 体积与能耗

闪存的体积小、能耗低,为构建低能耗计算机系统或数据中心提供了机会. EXCES 利用固态盘作为磁盘缓存,将磁盘的数据预取、缓存、缓冲在固态盘,从而减少磁盘访问,以达到节能的目的[44].

在数据中心设计中,FAWN[17]和 Gordon[18]都提出了匹配集群节点的计算处理能力与 I/O 能力,通过使用低频率的 CPU 和低延迟的闪存芯片

以减少计算处理与 I/O 之间的差距. CPU 的能耗随频率超线性增长,相对于现有服务器的高频 CPU,低频 CPU 所需能耗很低. 由于闪存芯片体积小,可在单块板子上集成大量的闪存芯片,去除服务器不需要的冗余功能,可实现单板的集群服务器,从而大大减少数据中心的占地面积. 此外,在服务器中使用闪存盘或闪存卡也可降低系统对传统磁盘阵列或服务器数目的需要,从而降低系统规模,以节省能耗.

3.4 小 结

在以低延迟访问介质构建的存储系统中,软件系统的开销逐渐占据较大的比例. 以 FusionIO 为代表的闪存加速卡以及 UCSD 基于 PCM 的 Moneta/Onyx 在 I/O 栈上减少了软件处理模块,以降低软件开销[19-20, 22]. 当前不少研究工作在设备驱动、缓存调度、文件系统上分别提出了针对闪存介质的优化[4-5, 13, 29, 34]. 在软件程序接口方面,TRIM 指令已广泛应用于固态盘系统. 但其他指令多为针对特定场景提出,如 PTRIM/EXISTS 指令用于文件系统借用闪存转换层进行空间管理[6];Nameless write 的多种接口实现将空间分配功能交由闪存转换层完成[5];Atomic write 用于闪存转换层向上层系统导出原子写功能[7-9]. 在硬件接口与通信机制上,SATA 向 PCIe 接口过渡已得到大多数人的认可[13],硬件通信上频繁中断的效率不及轮询机制也得到了关注[30]. I/O 栈的调整或重构对提升闪存存储系统性能相对明显,然而现有研究工作大多基于纯软件闪存转换层的方式,或者纯硬件的闪存转换层的方式,对于软硬件分工讨论较少. 而且现有研究仅针对闪存介质优化,而忽略对传统磁盘或未来其他新型存储介质的支持. 因而,如何进行软硬件(系统软件与设备内硬件控制)合理分工,并实现有效的接口语义,以及如何重构软件系统以提供多种访问介质的存储系统的管理,都值得进一步研究.

在闪存存储系统的可靠性上,存储可靠性不仅取决于设备本身,而且与外部负载密切相关. 现有的研究分别在系统通过对写操作的删冗压缩[15-16, 37-40]、多设备间的磨损与恢复[21, 41]、混合存储系统优劣势互补[42-43]上开展工作. 从系统缓存、文件系统组织方面对写负载影响的研究工作还比较少.

在利用闪存构建低能耗计算机系统或数据中心方面,通过闪存的高性能以减少存储规模,利用闪存缓存数据以较少磁盘使用[44]或者直接利用闪存重构节能数据中心[17-18]等几个方面都有所研究. 直接利用闪存重构节能数据中心的方式对未来数据中心的构建较有启发,但针对不同负载情形下计算与 I/O 比例不同,以及对网络互联与软件系统等产生的影响,还需要进一步的研究.

4 总结与展望

本文介绍了当前集中典型的闪存存储系统,并从 I/O 栈调整与重构、闪存系统可靠性、体积与能耗 3 个角度对闪存系统构建中的关键技术和热点问题进行了探讨. 闪存系统呈现与传统磁盘系统迥然相异的特性,简单的设备替换限制了闪存性能的发挥,难以克服闪存如磨损等劣势,也不利于利用闪存提供原子写等新功能.

如何充分利用闪存特性并通过修改或重构软件系统以提供友好支持,是构建闪存系统的研究热点,也是需要进一步研究的方向. 首先,在软件系统的 I/O 栈上,如何减少软件冗余,以及缓存调度 I/O 请求以发挥闪存低延迟与并发特性仍是闪存存储系统构建中的一大挑战. 其次,闪存可靠性是与系统负载密切相关的,如何从负载及系统的角度设计闪存的可靠性也是值得研究的问题. 闪存与传统磁盘的可靠性模型发生变化,现有系统构建较少考虑闪存寿命,从系统构建的角度减少数据写以提高闪存寿命也是系统构建时需要考虑的因素. 最后,闪存系统的设计也应当考虑新型存储器件可能对系统带来的影响,以提高软件系统在多种存储器件下的普适性. 现有文件系统多数优化是基于磁盘特性的,而随着闪存存储以及 PCM 等新型存储器件的出现,系统与存储硬件之间的分工协作需要权衡,并且系统应尽可能提供多种存储设备的管理.

闪存部署已较为广泛,其可靠性与成本问题逐渐被人们接受. 闪存存储系统的构建还处在起步阶段,但通过系统级提供对闪存的友好支持,可充分发挥闪存优势,也将会给计算机系统带来重要的改变.

参 考 文 献

[1] Chen F, Lee R, Zhang X. Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing [C] //Proc of the 17th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2011: 266-277

[2] Hu Y, Jiang H, Feng D, et al. Performance impact and interplay of SSD parallelism through advanced commands, allocation strategy and data granularity [C] //Proc of the Int Conf on Supercomputing. New York: ACM, 2011: 96-107

[3] Zheng Wenjing, Li Mingqiang, Shu Jiwu. Flash storage technology [J]. Journal of Computer Research and Development, 2010, 47(4): 716-726 (in Chinese)
(郑文静, 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)

[4] Josephson W K, Bongo L A, Flynn D, et al. DFS: A file system for virtualized flash storage [C] //Proc of the 8th USENIX Conf on File and Storage Technologies. Berkeley: USENIX Association, 2010: 85-99

[5] Zhang Y, Arulraj L P, Arpaci-Dusseau A C, et al. De-indirection for flash-based SSDs with nameless writes [C] //Proc of the 10th USENIX Conf on File and Storage Technologies. Berkeley: USENIX Association, 2012: 1-16

[6] Nellans D, Zappe M, Axboe J, et al. ptrim()+exists(): Exposing new FTL primitives to applications [C/OL]. 2011 [2012-09-01]. http://david. nellans. org/files/NVMW-2011. pdf

[7] Ouyang X, Nellans D, Wipfel R, et al. Beyond block I/O: Rethinking traditional storage primitives [C] //Proc of the 17th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2011: 301-311

[8] Prabhakaran V, Rodeheffer T L, Zhou L. Transactional flash [C] //Proc of the 8th USENIX Conf on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2008: 147-160

[9] On S, Xu J, Choi B, et al. Flag commit: Supporting efficient transaction recovery on flash-based DBMSs [J]. IEEE Trans on Knowledge and Data Engineering, 2011, 24(9): 1624-1639

[10] Hsu P, Chang Y, Huang P, et al. A version-based strategy for reliability enhancement of flash file systems [C] //Proc of the 48th Design Automation Conf. New York: ACM, 2011: 29-34

[11] Ranganathan P, Chang J. (Re) Designing data-centric data centers [J]. IEEE Micro, 2012, 32(1): 66-70

[12] Caulfield A M, Coburn J, Mollov T, et al. Understanding the impact of emerging non-volatile memories on high-performance, IO-intensive computing [C] //Proc of the 2010 ACM/IEEE Int Conf for High Performance Computing, Networking, Storage and Analysis. Los Alamitos, CA: IEEE Computer Society, 2010: 1-11

[13] Fusion-io. The fusion-io difference [OL]. [2012-09-01]. http://www. fusionio. com/load/-media-/1qaz4e/docsLibrary/FIO_SSD_Differentiator_Overview. pdf

[14] Voilin memory [OL]. [2012-09-01]. http://www. violin-memory. com

[15] Pure Storage. How flash changes everything [OL]. [2012-09-01]. http://www. purestorage. com/pdf/Pure _ Storage _ Whitepaper_How_Flash_Changes_Everything. pdf

[16] XtremIO. Flash implications in enterprise storage array designs [OL]. [2012-09-01]. http://www. xtremio. com resources/xtremio-white-paper-flash-implications-in-enterprise-storage-array-designs/

[17] Andersen D G, Franklin J, Kaminsky M, et al. FAWN: A fast array of wimpy nodes [C] //Proc of the 22nd ACM SIGOPS Symp on Operating Systems Principles. New York: ACM, 2009: 1-14

[18] Caulfield A M, Grupp L M, Swanson S. Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications [C] //Proc of the 14th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2009: 217-228

[19] Caulfield A M, De A, Coburn J, et al. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories [C] //Proc of the 43rd Annual IEEE/ACM Int Symp on Microarchitecture. Los Alamitos, CA: IEEE Computer Society, 2010: 385-395

[20] Akel A, Caulfield A M, Mollov T I, et al. Onyx: A prototype phase change memory storage array [C] //Proc of the Workshop on Hot Topics in Storage and File Systems. Berkeley: USENIX Association, 2011: 74-78

[21] Ousterhout J, Agrawal P, Erickson D, et al. The case for RAMClouds: Scalable high-performance storage entirely in DRAM [J]. ACM SIGOPS Operating Systems Review, 2010, 43(4): 92-105

[22] Fusion-io. FusionIO ioMemory virtual storage layer [OL]. [2012-09-01]. http://www. fusionio. com/load/-media-/1qaz4e/docsLibrary/VSL_Technical_Overview. pdf

[23] NVM Express. NVMe standard [OL]. [2012-09-01]. http://www. nvmexpress. org/

[24] San Diego Supercomputer Center, UC San Diego. Gordon: Data-intensive supercomputing [OL]. [2012-09-01]. http://www. sdsc. edu/supercomputing/gordon/

[25] Anthes G. Revamping storage performance [J]. Communications of the ACM, 2012, 55(1): 20-22

[26] Ongaro D, Rumble S M, Stutsman R, et al. Fast crash recovery in RAMCloud [C] //Proc of the 23rd ACM Symp on Operating Systems Principles. New York: ACM, 2011: 29-41

[27] Ousterhout J, Agrawal P, Erickson D, et al. The case for RAMCloud [J]. Communications of the ACM, 2011, 54(7): 121-130

[28] Swason S. Redrawing the boundary between software and storage for fast non-volatile memories [OL]. [2012-09-01]. http://nvsl. ucsd. edu/assets/talks/2012-05-21-DaMoN-release. pdf

[29] Park S, Shen K. FIOS: A fair, efficient flash I/O scheduler [C] //Proc of the 10th USENIX Conf on File and Storage Technologies. Berkeley: USENIX Association, 2012: 155-169

[30] Seagate. Savvio? 15K. 3 Data Sheet [OL]. [2012-09-01]. http://www. seagate. com/files/www-content/product-content/savvio-fam/savvio-15k/savvio-15k-3/en-us/docs/savvio-15k-3-data-sheet-ds1732-5-1201us. pdf

[31] Intel. Intel X25-E Data Sheet [OL]. [2012-09-01]. http://download. intel. com/design/flash/nand/extreme/319984. pdf

[32] Fusion-io. FusionIO ioDrive Octal Data Sheet [OL]. [2012-09-01]. http://www. fusionio. com/load/-media-/1shm1l/docsLibrary/FIO_DS_Octal. pdf

[33] Yang J, Minturn D B, Hady F. When poll is better than interrupt [C] //Proc of the 10th USENIX Conf on File and Storage Technologies. Berkeley: USENIX Association, 2012: 25-31

[34] Caulfield A M, Mollov T I, Eisner L A, et al. Providing safe, user space access to fast, solid state disks [C] //Proc of the 17th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2012: 387-400

[35] Park S, Yu J H, Ohm S Y. Atomic write FTL for robust flash file system [C] //Proc of the 9th Int Symp on Consumer Electronics. Piscataway, NJ: IEEE, 2005: 155-160

[36] Grupp L M, Davis J D, Swanson S. The bleak future of NAND flash memory [C] //Proc of the 10th USENIX Conf on File and Storage Technologies. Berkeley: USENIX Association, 2012: 17-24

[37] Chen F, Luo T, Zhang X. CAFTL: A content-aware flash translation layer enhancing the lifespan of flash memory based solid state drives [C] //Proc of the 9th USENIX Conf on File and Storage Technologies. Berkeley: USENIX Association, 2011: 77-90

[38] Gupta A, Pisolkar R, Urgaonkar B, et al. Leveraging value locality in optimizing NAND flash-based SSDs [C] //Proc of the 9th USENIX Conf on File and Storage Technologies. Berkeley: USENIX Association, 2011: 91-103

[39] Lee S W, Moon B. Design of flash-based DBMS: An in-page logging approach [C] //Proc of the ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2007: 55-66

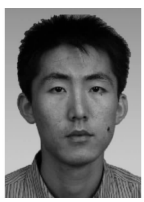
[40] Wu G, He X. Delta-FTL: Improving SSD lifetime via exploiting content locality [C] //Proc of the 7th ACM European Conf on Computer Systems. New York: ACM, 2012: 253-266

[41] Balakrishnan M, Kadav A, Prabhakaran V, et al. Differential RAID: Rethinking RAID for SSD reliability [C] //Proc of the 5th European Conf on Computer Systems. New York: ACM, 2010: 15-26

[42] Soundararajan G, Prabhakaran V, Balakrishnan M, et al. Extending SSD lifetimes with disk-based write caches [C] //Proc of the 8th USENIX Conf on File and storage technologies. Berkeley: USENIX Association, 2010: 101-114

[43] Yang Q, Ren J. I-CASH: Intelligently coupled array of SSD and HDD [C] //Proc of the 17th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2011: 278-289

[44] Useche L, Guerra J, Bhadkamkar M, et al. EXCES: External caching in energy saving storage systems [C] //Proc of the 14th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2008: 89-100



Lu Youyou, born in 1987. PhD candidate. His research interest includes flash-based storage and file systems.

Shu Jiwu, born in 1968. PhD, professor and PhD supervisor. Senior member of China Computer Federation. His main research interests include network storage and cloud storage, storage security and reliability, and parallel process technologies (shujw @ tsinghua. edu. cn).