计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150910
53(9):1893-1905,2016

固态盘阵列构建方法研究综述

李祥楠[1,2] 张广艳[1] 李 强[2] 郑纬民[1]
[1](清华大学计算机科学与技术系 北京 100084)
[2](吉林大学计算机科学与技术学院 长春 130012)
(lixiangnan101@163.com)

A Survey on the Approaches of Building Solid State Disk Arrays

Li Xiangnan[1,2], Zhang Guangyan[1], Li Qiang[2], and Zheng Weimin[1]
[1](*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)
[2](*College of Computer Science and Technology, Jilin University, Changchun* 130012)

Abstract Flash-based solid state disks (SSDs) use flash memory chips as their storage media, which have the features of non-volatility, small size, light weight, high shock resistance, high performance, and low power consumption. Single SSDs have the drawbacks of poor random write performance and limited erase endurance. Organizing multiple SSDs with the RAID technology is promising in delivering high reliability, large capacity and high performance. However, researchers have demonstrated that applying RAID algorithms to SSDs directly does not work well and have proposed some SSD-aware RAID algorithms. In this paper, we first analyze the drawbacks of flash-based solid state disks and the RAID technology, and use performance, reliability and price as the evaluation criteria of the approaches to building SSD arrays, and choose random writes, small writes, garbage collection, load balance, erase/program cycles, wear leveling and redundancy levels as the analysis metrics. Then, we analyze and compare the advantages and disadvantages of two types of array building approaches on the disk level and the flash-chip level respectively. Finally, we summarize those different approaches and point out prospective research directions in the future.

Key words solid state disk (SSD); disk array; lifespan; performance; reliability

摘 要 单个固态盘存在随机写性能较差和擦写次数有限等缺陷,使用廉价盘冗余阵列(redundant arrays of inexpensive disks, RAID)技术将多块固态盘组织在一起有助于满足存储系统的高可靠、大容量、高性能的要求.然而,将RAID算法简单应用于固态盘阵列会遇到一些问题.首先分析了基于Flash的固态盘和RAID技术存在的缺陷,选取性能、可靠性和价格作为阵列构建方法的评价标准,并将随机写、小写、垃圾回收、负载均衡、擦除次数、磨损均衡、冗余度等问题作为分析重点;然后,从固态盘和Flash芯片2种构建粒度出发,分别论述了不同的构建方法并分析了各自的优缺点;最后,总结了不同构建方法并指出未来可能的研究方向.

关键词 固态盘;存储阵列;使用寿命;性能;可靠性

中图法分类号 TP333

收稿日期:2015-10-13;**修回日期**:2016-03-02

基金项目:国家"八六三"高技术研究发展计划基金项目(2013AA01A210);国家自然科学基金项目(61170008,61272055)
This work was supported by the National High Technology Research and Development Program of China (863 Program) (2013AA01A210) and the National Natural Science Foundation of China (61170008,61272055).

通信作者:张广艳(gyzh@tsinghua.edu.cn)

以磁盘为主的外部存储难以满足当前应用的存储需求,容易成为整个计算机系统的性能瓶颈. Flash 固态盘的出现为存储系统提供了新的选择. 然而,固态盘容量的提升是以降低可靠性为代价的,采用多层芯片技术使得多比特错误开始出现,从而使修复单比特错误的纠错码不再有效[1]. 除此之外,固态盘也存在着先天的介质损耗和写性能较差[2]问题,单块固态盘难以满足存储系统高性能、高可靠性的要求. 利用廉价盘冗余阵列(redundant arrays of inexpensive disks, RAID)[3]技术构建闪存存储系统可以在增大系统容量、提高系统性能的同时增强系统的可靠性,但是其也存在着小写性能差的缺陷. 因为固态盘的不同特性,并不能将 RAID 技术简单地应用到固态盘上,必须对固态盘阵列的数据布局进行优化,充分挖掘固态盘和 RAID 技术的优势,弥补相互的不足,以达到性能的最优[4].

已有多种利用固态盘构建阵列的方法,典型的有:Diff-RAID 和 WeLe-RAID 通过调整校验信息在阵列中的分布来实现吞吐量和可靠性之间的平衡;I-CASH 通过将 1 个固态盘(solid state disk, SSD)和 HDD 智能耦合,利用内容局部性将大多数 I/O 操作转化成对 SSD 的读操作来提高系统的吞吐量;部分校验延迟更新通过每次写入数据只计算部分校验数据,并将其存在部分校验缓存中以减少校验信息更新. 这些方法只在某一两方面优化了系统的性能,并没有全面综合地考虑 RAID 策略应用在固态盘时所造成的随机写、小写、垃圾回收、有限寿命等问题. 因此,如何构建大规模、高性能、高可靠的存储系统是我们需要进一步讨论的问题.

本文首先深入分析固态盘的特性和 RAID 技术,选取性能、可靠性、价格为构建 Flash 阵列的评价标准;然后根据构建粒度的不同,分别从固态盘和 Flash 芯片 2 种阵列构建模式深入分析了现有的固态盘阵列的关键技术,并指出了各种方法的优缺点;之后通过对比各个阵列构建方法提出了几点观察,阐述了阵列构建技术所面临的挑战,并指出未来可能的研究方向.

1 评价标准

通过分析 Flash 固态盘和 RAID 技术存在的技术缺陷,本节给出阵列构建方法的评价标准.

1.1 固态盘存在的问题

固态盘是由 Flash 芯片作为存储介质构建成的数据存储设备,提供与传统磁盘相同的 SATA 或 SAS 接口、功能特征及操作方式. 固态盘随着容量的提升和价格的降低,其相对于传统磁盘的高存取速度、低能耗、抗震性等优势越来越明显.

但是,固态盘还不能完全取代传统磁盘,因为它主要存在以下 4 个问题:

1) 随机写性能差. 固态盘采用异地更新策略,垃圾回收过程产生大量合并,导致固态盘随机写性能较差.

2) 擦除次数有限. Flash 本身有限的擦除次数,导致固态盘寿命有限.

3) 磨损不均衡. 闪存写入和擦除的不均衡[5-7]会使闪存的不同部位磨损程度各异,当个别部位提前达到磨损界限时会使数据可靠性下降.

4) 垃圾回收[8-10]频繁. 固态盘垃圾回收需要消耗较多 I/O 资源. 另外,当固态盘中某个芯片正在执行垃圾回收操作时,该芯片接收到的读写请求都将被挂起,直到垃圾回收完成[11]. 因此垃圾回收操作的效率直接影响固态盘存储系统的性能[12].

1.2 RAID 技术存在的问题

RAID 技术的基本思想是通过数据在多块硬盘之间进行条带化来提高数据存取速度,通过存储校验信息来提高数据可靠性. 大多数 RAID 技术研究集中在 RAID-4/5/6 上,其主要有以下 3 个问题:

1) 小写性能差. 对于 RAID-4/5,每执行 1 次小写请求会导致 2 次读和 2 次写操作,这将造成较大延迟;RAID-6 使用双维校验技术,使其小写性能更差.

2) 负载不均衡. RAID-4 结构中,校验信息集中在 1 块盘上,校验盘容易成为写性能瓶颈;RAID-5/6 虽然将校验信息分布到各盘,但是由于数据的局部性,如果有些负载对于某个条带读写很频繁,同样会导致某块盘成为性能瓶颈.

3) 冗余度固定. 对于传统的 RAID 技术,RAID 级别确定之后其冗余度就固定了. 随着系统使用时间的增加,盘出错的概率越来越大. 固定的冗余度很难使系统不同时期达到相同的可靠级别.

1.3 提出评价标准

如果简单地将 RAID 策略应用到固态盘上,校验信息的频繁更新会导致固态盘的介质损耗更加严重,降低固态盘的使用寿命. 加上固态盘的随机写性能较差,使得固态盘阵列的随机写、小写性能更差. 同时,垃圾回收的频率也会增加,从而影响性能.

随机写、小写、垃圾回收、负载均衡会影响系统的性能;磨损均衡、擦除次数会影响系统的使用寿命,不同的冗余度也会带来不同的可靠级别,从而影响系统的可靠性;构建固态盘阵列需要多块固态盘,固态盘存在诸多优于磁盘的优点,但是其价格却比磁盘高很多.

因此,我们选取性能、可靠性、价格这 3 点作为固态盘阵列构建方法的评价标准. 为了在降低价格的同时提高系统的可靠性、性能和使用寿命,大多数针对固态盘阵列的研究都集中在实现固态盘的磨损均衡,减少固态盘的擦除次数,不同时期提供不同的冗余度,优化固态盘随机写性能、垃圾回收性能和提高 RAID 技术的小写性能、实现其负载均衡上.

2 多固态盘构建 RAID 系统

SSD 提供磁盘访问接口,已有的面向磁盘开发的软件都可以直接应用到 SSD 上. 在 SSD 层面应用 RAID 机制来构建闪存存储系统就是将多个 SSD 组合成冗余阵列,用通常的 RAID 控制器和总线连接,直接把磁盘替换成 SSD[13].

2.1 校验数据差异分布阵列 Diff-RAID

由于每次数据的写操作都要更新对应的校验信息,所以一块固态盘上校验信息所占的比例越高,其老化速度也越快. Balakrishnan 等人[14]提出 Diff-RAID,主要通过在各盘之间进行校验数据的差异化分布,来实现不同固态盘之间不同的老化速度. 当某个固态盘接近擦除次数限制时,将其及时更换成新的固态盘. 此时,再按照阵列中各盘的老化程度重新分配校验信息,来重新实现不同老化程度.

Diff-RAID 通过调整校验信息在多块固态盘之间的分配比例来实现可靠性和性能之间的平衡,相当于 RAID-4 和 RAID-5 之间的折中. 它缓解了 RAID-4 系统中校验信息集中存储在单块固态盘上使之容易成为系统性能瓶颈的问题,降低了 RAID-5 系统中 1 块固态盘出现故障时第 2 个固态盘很快出现故障的概率,提高了系统可靠性.

但是,就整个固态盘阵列而言,Diff-RAID 并没有减少校验数据更新引起的总的固态盘擦除次数,而且,在使用期间会产生多次固态盘替换操作而降低阵列的性能. 此外,Diff-RAID 并没有解决固态盘阵列的随机写、小写性能较差等问题.

2.2 磨损均衡阵列 WeLe-RAID

RAID-5 系统主要靠平衡各个盘上校验块的个数来实现固态盘之间的磨损均衡. 一旦一些工作负载对某块盘上的校验信息所对应的数据块频繁存取时,该盘由于校验信息的频繁更新将比其他盘老化得更快.

Du 等人[15]提出磨损均衡阵列 WeLe-RAID. 正常情况下 RAID 控制器管理活跃设备组. 当整个系统达到寿命限度时,迁移控制器将数据从活跃设备组迁移到准备设备组. 从而,准备设备组成为新的活跃设备组,并且新的设备组被用作准备组. 此外,WeLe-RAID 用老化程度驱动的校验信息分布来优化数据布局.

与 Diff-RAID 相反,WeLe-RAID 以 SSD 的使用寿命为参数动态调整校验信息的分布,实现 RAID 范围内的磨损均衡. 使得固态盘组之间同速变老,然后同步替换,延长了阵列的寿命. WeLe-RAID 在 RAID 系统的整个生命周期内,替换操作出现频率很低,但是它并没有解决固态盘阵列的随机写、小写性能低的问题.

2.3 混合盘阵列 HPDA

由于校验信息的频繁更新,将 RAID 技术应用在固态盘上会加剧固态盘的损耗. 同时,固态盘随机写性能比较差,导致固态盘阵列的随机小写性能更差. Mao 等人[4]提出一种混合盘阵列 HPDA. 如图 1 所示,多个 SSD 和 1 个 HDD 组成 1 个 RAID-4 结构. 其中,SSD 存储数据信息,HDD 存储校验信息. 同时再增加 1 块磁盘,与校验盘的剩余空间组成 1 个 RAID-1 镜像,并以日志写方式组织,用作数据缓存空间.

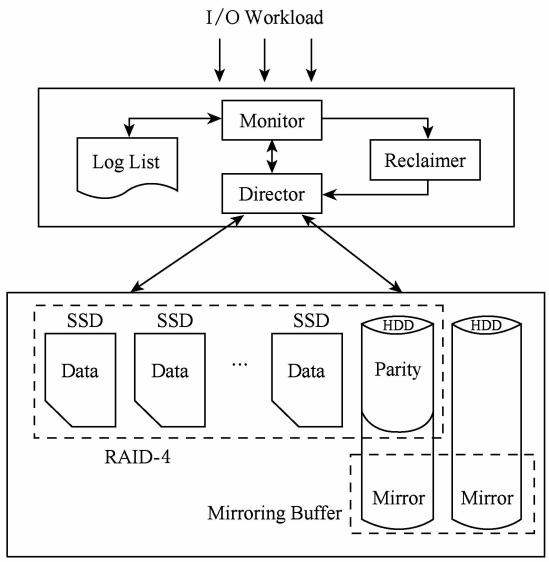

Fig. 1 The architecture of HPDA[4].

图 1 HPDA 的体系结构[4]

通过将磁盘用作校验盘,HPDA 消除了由于校验信息频繁更新导致的固态盘介质损耗问题,延长了阵列的使用寿命. 以日志写方式组织镜像缓存空间,用于缓存随机写或小写请求,优化了阵列的写性能. 但是,由于校验盘和镜像使用同一块磁盘,该磁盘上的数据访问容易出现上下抖动,从而导致读写性能下降. 另外,当进行在线恢复时,共享的磁盘需要同时响应用户的读写请求和恢复数据所需要的读请求,使得磁盘上下来回寻道,导致恢复时间和用户响应延迟明显变长.

2.4 支持连续数据保护的阵列 HerpRap

连续数据保护机制(continuous data protection, CDP)能够持续捕获写入操作的历史,所以能够恢复存储状态到任何一个之前保存的时间点. Zeng 等人[16]提出一种结合 RAID 和 CDP 技术的阵列 HerpRap. HerpRap 是 HPDA 的扩展,它增加了 2 个功能模块(如图 2 所示):CDP 管理模块和数据重建模块. CDP 管理模块负责管理 CDP 区域,维护数据块的访问历史信息. CDP 区域由多个 HDD 组成,但每次只有 1 个 HDD 是活跃的,目标是节省能耗. 数据重建模块包括 1 个用来恢复数据的恢复进程和 1 个用来追踪 CDP 区域块日志的追踪进程.

HerpRap 拥有 HPDA 的优点,同时具备以块级别追踪数据并且能够恢复数据到任意时间点. 但是,它同时也具备 HPDA 所有的缺点,存储校验信息的 HDD 会出现频繁的上下寻道抖动的情况. 另外,CDP 区域的 HDD 失效,数据将会丢失,导致系统可靠性下降.

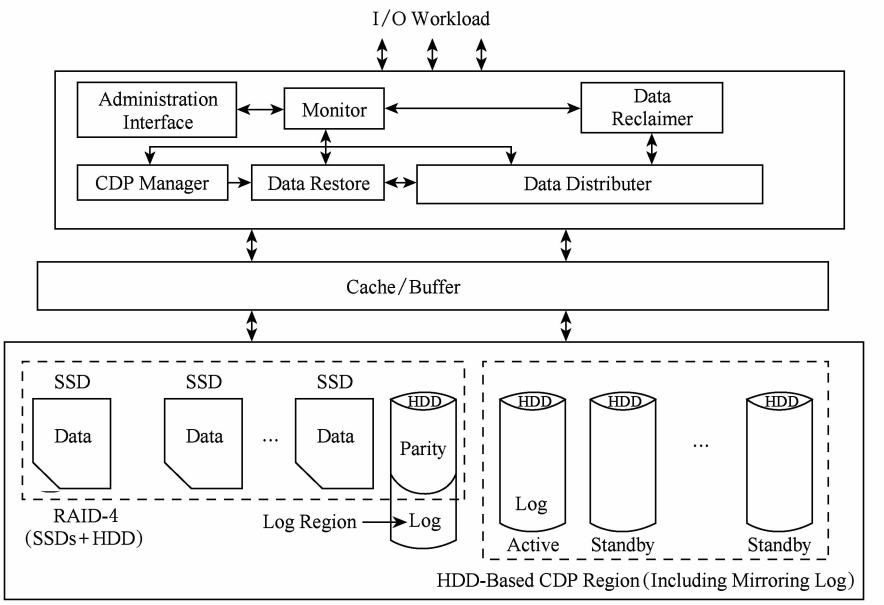

Fig. 2 The architecture of HerpRap[16].

图 2 HerpRap 的体系结构[16]

2.5 混合盘智能耦合阵列 I-CASH

数据存储中存在许多彼此相似的数据块,增量编码的存储方式可以消除数据块之间的冗余[17-18]. 另外,在许多数据密集型应用中,由于强大的内容局部性,在 1 次典型的数据块写操作中仅仅 5%～20% 的数据被改变[19-20],这被称为数据的内容局部性.

为了利用数据的内容局部性来最大化 I/O 性能,Yang 和 Ren[21]提出了一种智能耦合阵列 I-CASH. 如图 3(a)所示,I-CASH 将 1 个 SSD 和 1 个 HDD 混合. 其中,SSD 存储很少改变的大多为读的数据,称为参考块;HDD 存储数据修改增量 Δ,称为增量块. 当执行写操作时,I-CASH 首先识别出在 SSD 中的参考块,然后计算出要写入数据块的修改增量,并将增量写入 HDD 中如,图 3(b)所示. 当执行读操作时,在 SSD 中找出其参考块,在 HDD 中找出对应的增量,结合二者得到返回的数据块,如图 3(c)所示.

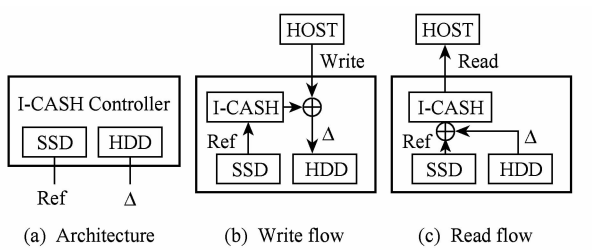

Fig. 3 The architecture of I-CASH [21].

图 3 I-CASH 的体系结构[21]

I-CASH 使得读操作主要发生在 SSD 上,写操作主要发生在 HDD 上.许多本应在 HDD 上的机械寻道-旋转-传输都转换成对 SSD 的读操作和 CPU 的计算操作,提高了性能.同时,也减少了对 SSD 的写入次数,延长了 SSD 的使用寿命.但是,I-CASH 没有数据冗余和校验机制,当数据发生错误或者丢失时,将没有办法恢复丢失的数据.

2.6 垃圾回收感知的阵列 GS-RAIS

当固态盘中某个芯片正在进行垃圾回收操作时,该芯片接收到的读写请求都将被挂起,直到垃圾回收完成[11].因此垃圾回收的执行效率直接影响整个固态盘性能.

吴素贞等人[12]提出一种垃圾回收感知的固态盘阵列 GC-RAIS.如图 4 所示,GC-RAIS 结构由若干个保存数据或校验的固态盘和 1 个热备固态盘组成.当某块固态盘正在执行垃圾回收时,发往该盘的写请求被重定向到热备盘上,并在映射表中修改相应的映射信息.对于发往该盘的读请求,首先检查映射表以判断读取数据的存储位置.如果在热备盘上,则由热备盘响应该读请求;否则,发送到该盘上的读请求被替换为发往系统中其他固态盘上的读请求,然后使用异或计算得到所需的数据并返回给用户.

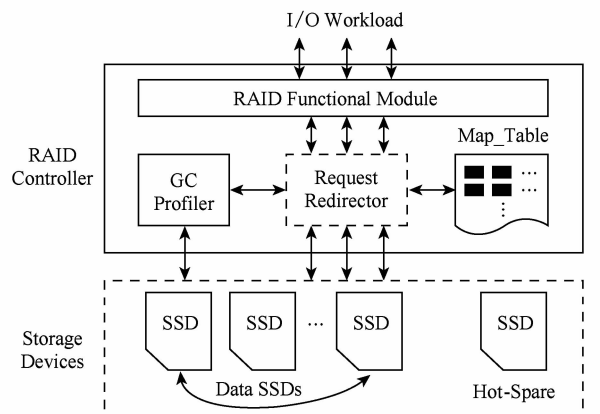

Fig. 4 The architecture of GC-RAIS[12].

图 4 GC-RAIS 的体系结构[12]

GC-RAIS 充分利用固态盘的高效随机读的特性,降低垃圾回收操作对固态盘阵列系统性能的负面影响.但是,并未对固态盘阵列固有的小写性能差、随机写性能差、有限寿命等问题进行改善.

2.7 图计算专用的存储阵列 FlashGraph

由于图数据的非结构化特征,图计算会产生很多随机读写操作.GraphChi,X-Stream 等单机图计算系统对图数据的存储进行划分和布局优化来提高图计算的性能,但是大量的磁盘 I/O 仍是其性能瓶颈.Zheng 等人[22]提出了 FlashGraph,使用 SSD 阵列来优化图计算的 I/O 性能.FlashGraph 使用 Semi-External Memory 的计算模式,计算时把所有顶点信息加载进 DRAM,而所有边信息则从 SSD 上动态读取.由于图计算只会对顶点信息进行修改而对边的信息只是读取,所以可以显著减少对 SSD 的磨损,还可以保持很好的扩展性.

为了充分利用 SSD 阵列的高带宽,FlashGraph 专门设计了用户态文件系统 SAFS,如图 5 所示.在图引擎和用户态文件系统层对 I/O 进行合并来减少 I/O 的数量,并增加 I/O 的顺序性.FlashGraph 在用户态实现简单的 Page Cache,以减小内容切换带来的内存拷贝.同时,因为缓存可以被应用感知,FlashGraph 根据任务上下文来提高缓存命中率.另外,FlashGraph 提供的编程模型使得每个计算任务可以将部分计算下放到用户空间文件系统中执行,完全利用用户态的 Page Cache,避免了用户空间的 2 次拷贝.

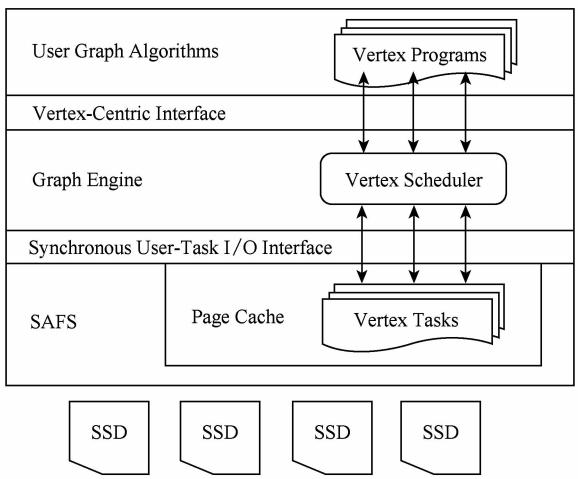

Fig. 5 The architecture of FlashGraph[22].

图 5 FlashGraph 的体系结构[22]

FlashGraph 可以根据具体图形算法来选择性地访问边数据以减少数据访问,通过合并 I/O 请求增加了 I/O 带宽,减少了 SSD 的磨损.FlashGraph 是利用 SSD 阵列来提高图计算的性能,为固态盘阵列的使用提供了有益参考.

3 多 Flash 芯片构建盘内 RAID 系统

多 Flash 芯片构建盘内 RAID 系统需要将 RAID 机制实现在闪存转换层(flash translation layer,FTL)[23-24].在 FTL 中,除了实现 RAID 功能外,还需实现地址映射、垃圾回收、磨损均衡等传统功能.

由于这些功能并非完全独立,因此在设计中需要综合考虑[13].

3.1 闪存感知的冗余阵列 FRA

基于 RAID 的方法均以写入性能的牺牲换取可靠性. 例如 RAID-5,每次小的随机写入操作均会带来校验数据的更新,造成写请求延迟的增加. Lee 等人[25]提出了闪存感知的冗余阵列(flash-aware redundancy array, FRA). 如图 6 所示,FRA 将每个芯片上 20%的块分配作为校验区,位于不同芯片上的用户数据和校验数据形成条带. FRA 利用 2 次盘存取间的空闲时间比盘本身的存取时间长的特点,将校验信息的更新延迟至空闲时间执行. 在执行写操作时,首先将用户数据写到闪存存储器中的 1 个日志块中,然后建立双页映射表,标记校验码是否更新. 在空闲时间内,FRA 查找日志映射表,并将校验信息写入冗余区域.

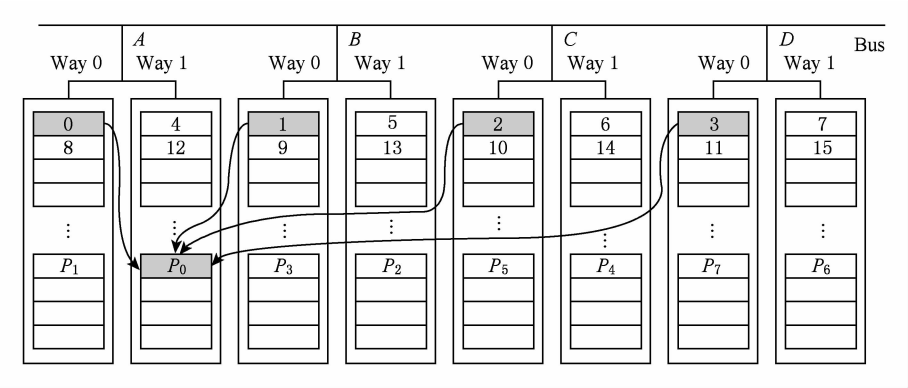

Fig. 6 The architecture of FRA[25].

图 6 FRA 的整体组织结构[25]

FRA 延迟校验更新的方式,优化了小写性能,保证校验信息的更新不会出现在读写关键路径上,写性能几乎等于没有采用冗余的方案,同时 FRA 减少了校验信息的写入次数,延长了固态盘的使用寿命. 但是,这是以牺牲数据可靠性为代价的,在新的校验信息提交之前,如果新的数据发生错误或丢失,FRA 将无法恢复. 此外,它并未优化固态盘阵列的随机写性能.

3.2 部分校验延迟更新技术

RAID-5 每次小写请求会导致 2 次读和 2 次写操作. 当 RAID-5 应用在固态盘时,由于闪存的写放大问题,将导致性能下降得更加明显. Im 和 Shin[1]

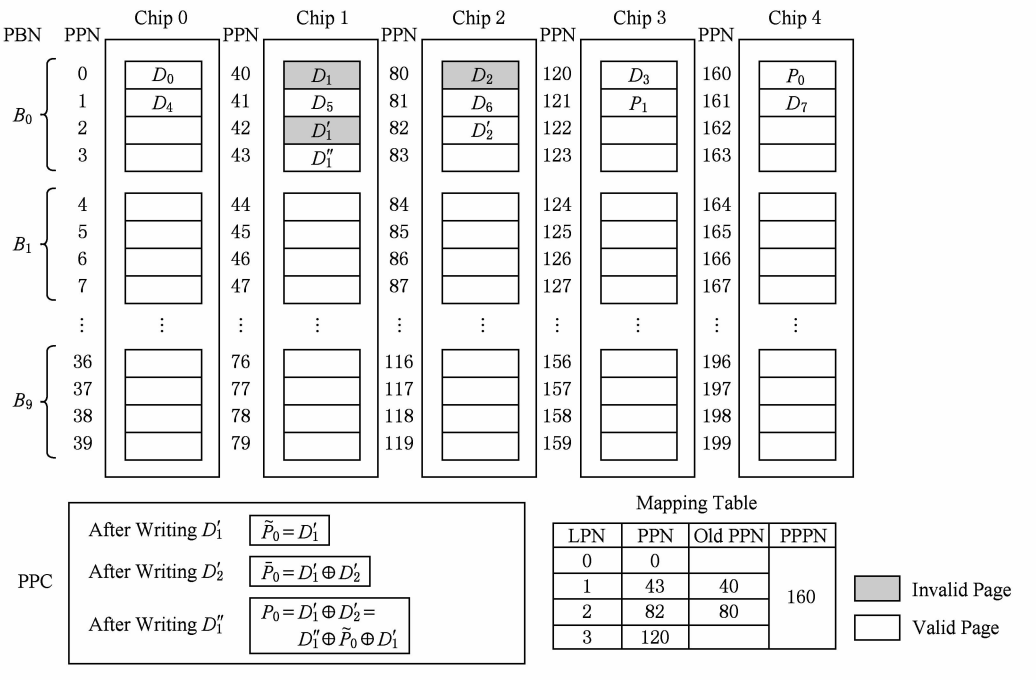

Fig. 7 The architecture of partial parity updating[1].

图 7 部分校验延迟更新技术的整体结构[1]

提出部分校验延迟更新技术.如图7所示,每次写入数据不会更新整个校验信息块,而只计算修改数据部分的校验并将其存储在NVRAM组成的部分校验缓存(partial parity cache, PPC)中.延迟更新校验信息直到PPC认为有必要提交或删除1个条目.数据的读取或写入请求到达控制器时,首先查询PPC,然后根据数据是否在PPC中执行其读写请求.

对于1次小写请求,如果数据是第1次写入或者还未被更新过,需要1次正常的写入操作;如果数据曾经被更新过,需要1次旧数据的读取操作和1次新数据的写入操作(不考虑在PPC中的读写操作).相对于原来的1次读和1次写操作,该技术减少了小写请求带来的附加操作,通过把校验信息存储在PPC中并且延迟更新,减少了校验信息的频繁更新对固态盘的写入次数,延长了固态盘的使用寿命,提高了可靠性.垃圾回收策略综合考虑数据迁移和提交代价,减少了垃圾回收的开销,但是并未对芯片阵列随机写、负载均衡等问题进行优化.

3.3 软件定义闪存SDF

在百度数据中心,仅仅40%或者更少的SSD原始带宽能被存储系统的上层应用所利用.另外,由于需要预留空间来容纳随机写入,一个SSD内只有50%~70%的空间能够被用来存储用户数据.如何最有效地利用SSD的带宽和容量是个很大的挑战.

Ouyang等人[26]提出了软件定义闪存(software-defined flash, SDF),如图8所示,SDF拥有与传统SSD完全不同的架构和设计,主要体现在以下3个方面:1)底层Flash通道暴露给上层软件,每个Flash通道具有1个独立的FTL控制器.2)SDF的软件栈消除了传统的Linux文件系统和I/O栈,只保留最底层的硬件驱动和轻量级用户态文件系统,降低了I/O请求延时.SDF把持久化写的粒度设置成NAND的擦除块大小,因此不需要预留冗余空间,也不需要垃圾回收,消除了写放大缺陷.3)因为上层存储系统已对数据进行了多副本或纠删码等冗余保护,SDF取消通道间的校验,把之前存放校验数据的通道用来存放数据,最大化利用数据存储空间.

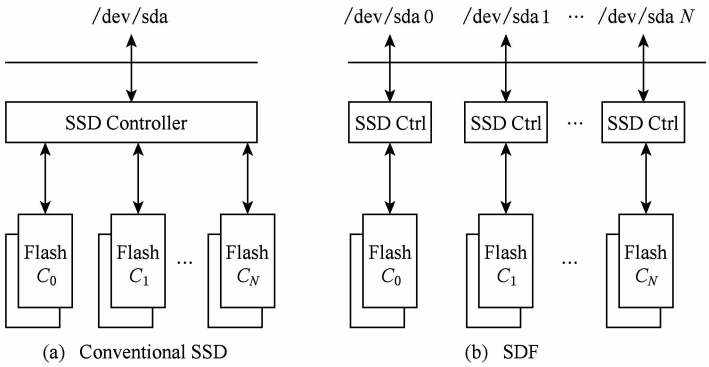

Fig. 8 The architectures of conventional SSD and SDF[26].

图8 传统SSD和SDF的结构[26]

SDF通过软硬件结合的方法保证了系统可靠性,并提高了空间利用率.通过聚合小写成大写提高了写性能,消除了写扩张从而延长了SSD的使用寿命.

3.4 寿命感知的可靠性方法

传统的RAID技术不能动态调整条带单元大小或者重新排列条带组.对于闪存来说,随着写入、擦除(P/E)次数的增加,位错误率将会增加[27-28].Lee等人[29]提出一种寿命感知的可靠性方法.如图9所示,所提方法的整体结构由多个闪存芯片组成RAID-5结构,分为数据区和日志区2部分.在SSD早期使用阶段,条带组的大小可能是芯片数目的4倍或者8倍.随着P/E周期地增加,条带组的大小可以减少到芯片数目大小或者更少.动态调整条带大小使得Flash芯片的整个生命周期有相同的可靠性级别.

对一个条带组的写请求,如果其校验块在数据区,则首先无效数据区的整个校验块,并在少量的存储级内存(storage class memory,SCM)中存储新的校验块(如图9中①),直到整个校验块都被更新或者SCM中没有空闲块时,将校验信息更新到数据区(如图9中②)).在日志区,当1个相应的日志块充满之后,将其校验页写入SCM中(如图9中③).

寿命感知的可靠性方法通过动态调整冗余度使固态盘在整个生命周期都有相同级别的可靠性,用SCM暂存校验信息提高了小写性能,通过减少对SSD的擦除次数延长了其使用寿命.虽然在SSD使用初期,位错误率很低,但当条带大小大于芯片数目

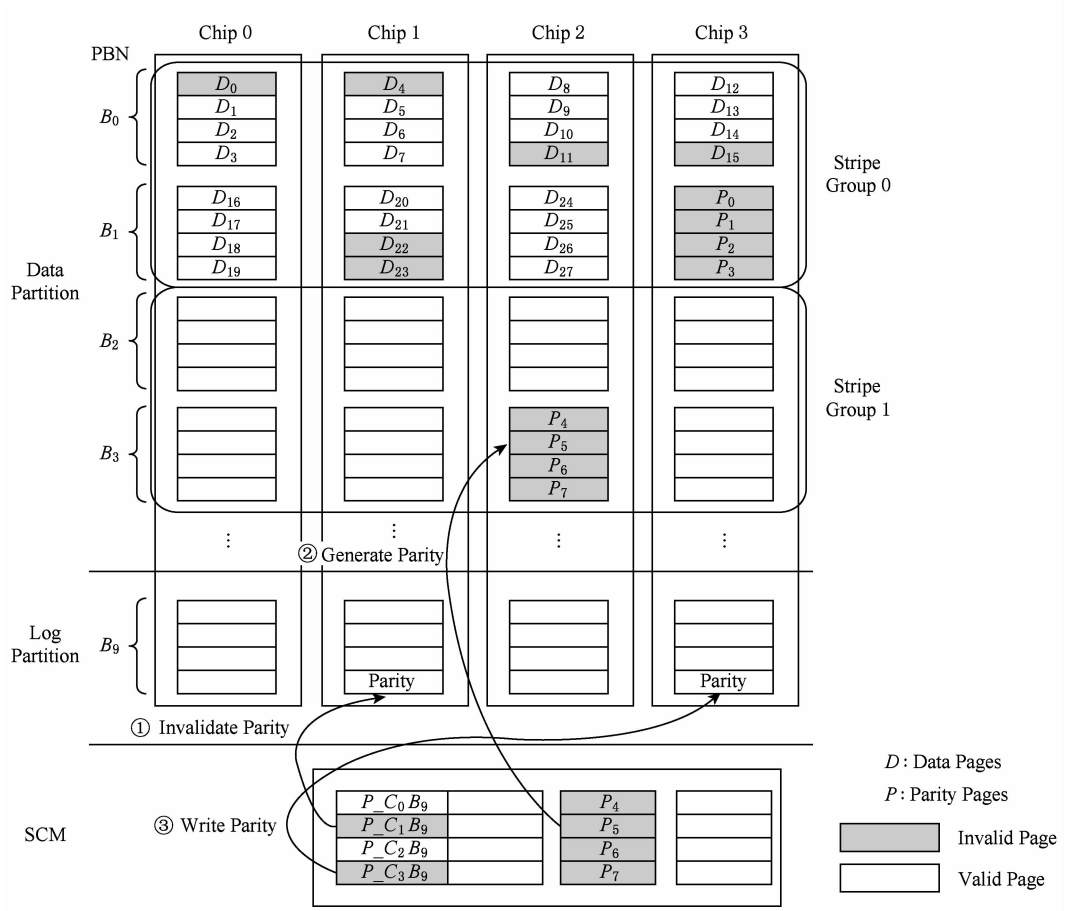

Fig. 9 The architecture of the life span-aware reliability scheme[29].

图 9 寿命感知可靠性方法的整体结构[29]

时,同一个条带上依然可能发生多个错误,这将导致数据无法恢复.该方法也未对固态盘阵列的随机写性能进行优化.

3.5 弹性条带化阵列 eSAP-RAID

传统的 RAID 结构在计算校验信息时需要读操作.特定的页被频繁更新会导致它和校验信息所在的芯片被频繁更新,使其相对于其他芯片磨损得更加严重.Kim 等人[30]提出弹性条带化阵列 eSAP-RAID,由多个闪存芯片组成 RAID-5 结构,相同条带中的每个单元有相同的物理块号.eSAP-RAID 解

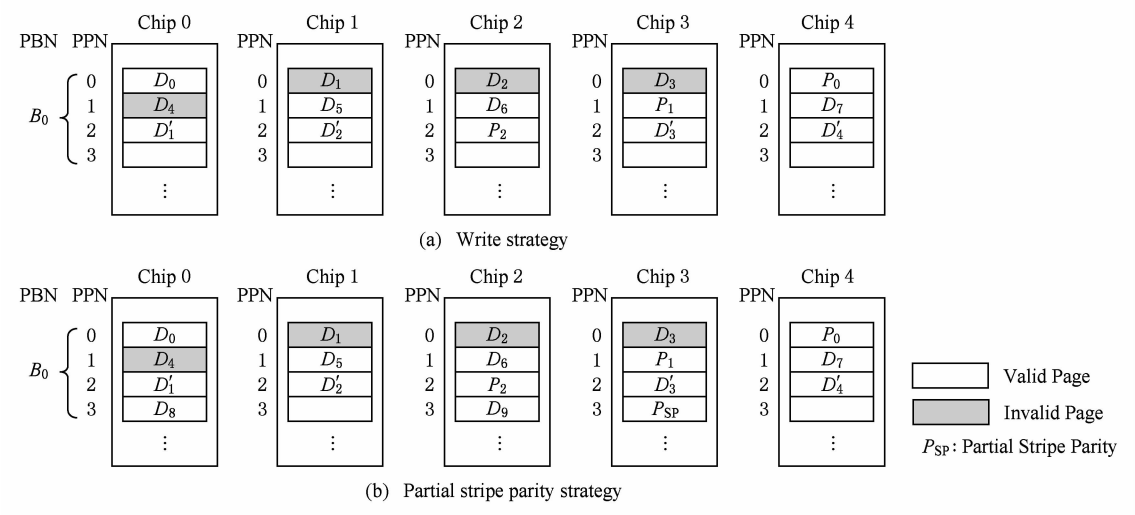

Fig. 10 Write scheme for eSAP-RAID[30].

图 10 eSAP-RAID 的写策略[30]

放传统 RAID-5 的条带数据的逻辑编址限制,1 个条带是由顺序到达的多个写请求的数据内容组成.

写入新的信息之后,控制器无效掉旧的数据而不需要读旧的页来计算校验信息,如图 10(a)所示. eSAP-RAID 采用可变条带模式为不是满条带的写请求建立部分条带,如图 10(b)所示. 在垃圾回收时,选择 1 个包含无效页最多的动态条带组(dynamic stripe group, DSG),将其中的有效页复制到新的 DSG 中,然后重新计算有效页的校验信息,最后擦除原来的 DSG 供以后使用.

eSAP-RAID 通过动态调整条带大小且忽略数据逻辑地址的方法,减少了计算校验信息时的读操作. 通过将随机写转换成顺序写,所有芯片被均衡写入,提高了系统性能并延长了芯片使用寿命. eSAP-RAID 的回收操作是以条带组为单位,由于擦除粒度较大,导致很多不经常改变的数据也被擦除,使芯片的擦除次数和擦除时间都有所增加. 另外,由于打乱了原有 RAID 的存储方式,需要维护 1 个映射表以记录数据的存储地址,带来了额外的维护开销.

3.6 层次化的 RAID 控制器 MuLe-RAID

对于基于 Flash 芯片的 RAID 结构,闪存控制器需要管理所有芯片. 随着芯片数量的增加,所有的访问都要通过控制器,所以控制器会成为整个系统的瓶颈. 杜溢墨等人[31]提出一种面向大容量高性能 SSD 的多层 RAID 结构 MuLe-RAID.

如图 11 所示,闪存芯片被分成若干个组(Part),每个 Part 由 1 个底层闪存控制器管理. 上层 RAID 控制器的管理单元是 Part,各 Part 类似于传统磁盘阵列系统中独立的磁盘. 上层 RAID 控制器采用 RAID 磨损均衡[32](RAID-Wear)算法,根据各个 Part 的损耗程度采用动态的校验数据分配策略,损耗越重的组分配越少的校验数据,损耗越轻的组分配越多的校验数据.

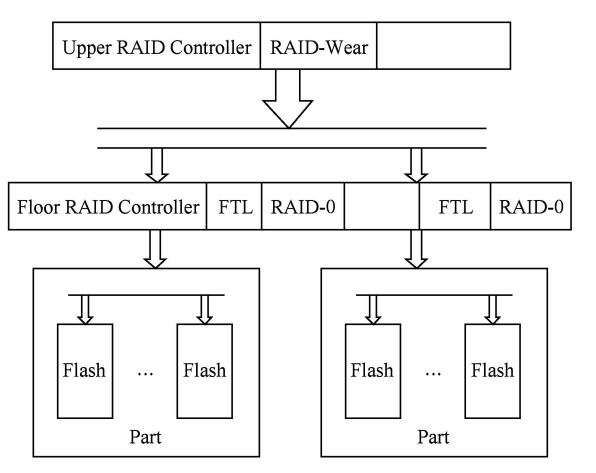

Fig. 11 The architecture of MuLe-RAID[31].

图 11 MuLe-RAID 的体系结构[31]

MuLe-RAID 采用多层次的控制器结构,增宽了数据通路从而有效提高了整个系统的性能. 分布式的磨损均衡算法保证了各个 Part 间的磨损均衡,提高了可靠性. 但是众多控制器之间需要协同,使得控制器的布局和时序控制变得很困难. 同时,为实现并行访问需要设置更多的数据通路和控制通路,实现起来更加困难. 此外,MuLe-RAID 并未对 RAID 策略中小写、随机写等问题进行改善.

3.7 SLC 和 MLC 混合的固态盘

闪存芯片可以分为单级单元(SLC)和多级单元(MLC)[33-34]两种. SLC 芯片可靠性高且存取速度快,但是容量低价格高;MLC 芯片容量大、价格低,但是结构复杂且出错率高[35-36]. Chang[37]提出一种 SLC 和 MLC 混合的固态盘.

如图 12 所示,混合固态盘由多个 MLC 芯片和 1 个 SLC 芯片组成. 热数据过滤器(hot-data filter)决定数据是否写入 SLC 芯片. 对于到来的写请求,如果是冷数据,则将其定向到 MLC 闪存;否则,将其交到利用率调节阀(utilization throttle)来平衡

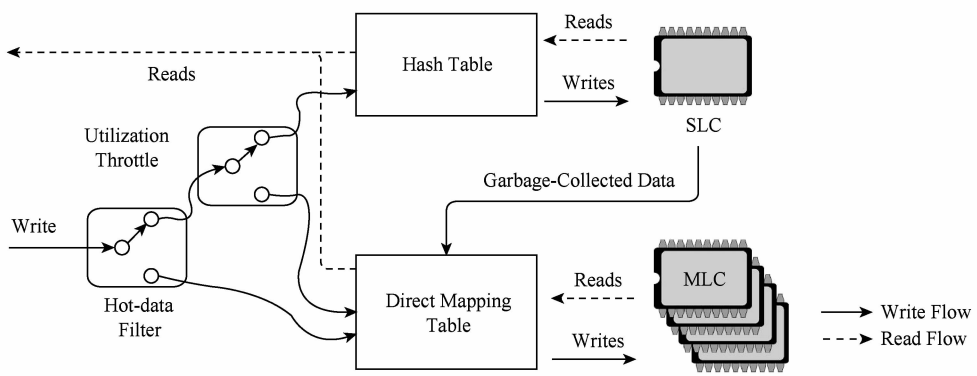

Fig. 12 The architecture of the hybrid SSD[37].

图 12 混合固态盘的体系结构[37]

SLC 芯片和 MLC 芯片之间的磨损次数. SLC 芯片的管理采用 1 个小的散列表代替传统的直接映射表,将物理上连续的 SLC 闪存块当作 1 个循环的日志空间. 当写入的数据块数目大于预定参数时,垃圾回收机制被触发.

混合固态盘兼顾数据写入效率和不同类型芯片之间的磨损均衡,最小化垃圾回收机制中的复制操作,延长了固态盘使用寿命. 混合固态盘实现了 MLC 芯片的容量和 SLC 芯片的响应速度. 针对冷热数据的不同,存储方案提高了随机写的性能. 但是在固态盘的内部没有保证数据可靠的冗余措施,一旦数据发生错误或丢失,将没有办法恢复数据.

3.8 干扰感知的阵列 DA-RAID-5

写干扰(program disturb)、读干扰(read disturb)和保持时间限制(retention time limit)是导致 NAND 闪存位错误的 3 个主要原因. Guo 等人[38]提出了干扰感知的阵列 DA-RAID-5 来提高企业级闪存存储系统的性能.

如图 13 所示,DA-RAID-5 将闪存存储系统的生命周期分成 3 个阶段:1)使用未绑定的干扰限制机制(unbound-disturb limiting, UDL)处理由于读干扰和保持时间限制引起的数据错误. 如果数据存在时间超过阈值 $T_{retention}$ 或者闪存块被读的次数大于阈值 T_{read},闪存块将被回收;如果块的 P/E 周期超过阈值 T_p,则系统进入第 2 阶段. 2)采用 P/E 感知的 RAID-5 方案,P/E 周期超过 T_p 的条带应用 RAID-5 机制,其他条带依然采用 UDL 机制. 当应用 RAID-5 机制的条带数量超过给定阈值 L 时,系统进入第 3 阶段. 3)对所有条带应用 RAID-5 机制,同时采用混合缓存策略以减少校验信息更新次数和开销.

DA-RAID-5 将闪存存储系统的生命周期分为 3 个阶段,不同阶段采用不同可靠性方案,减少了写干扰、读干扰和保持时间限制对 NAND 闪存引起的位错误. DA-RAID-5 减少了对校验信息的访问、提高了性能和可靠性,也延长了 NAND 闪存的使用寿命. 但是,没有优化 SSD 固态盘阵列的小写和随机写性能.

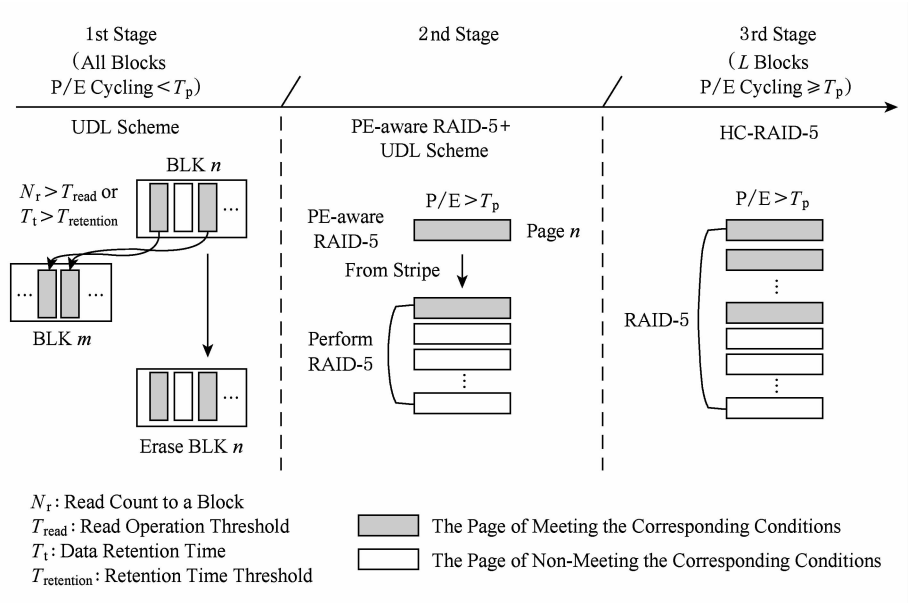

Fig. 13 The architecture of DA-RAID-5[38].

图 13 DA-RAID-5 的体系结构[38]

4 讨 论

以上各阵列构建方法只在某一两方面优化了系统的性能,并没有全面考虑 RAID 策略应用在固态盘时所造成的随机写、小写、磨损均衡、垃圾回收和可靠性等问题. 表 1 对比了各固态盘阵列构建方法符合评价标准的情况.

通过对各种阵列构建方法的对比分析,可以得到如下 4 点观察:

1) 没有一种构建方法完美地考虑了小写、随机写、垃圾回收、磨损均衡、负载均衡、擦除次数和冗余度等问题,因此如何构建大规模、高性能、高可靠的固态盘阵列系统是需要进一步探讨的问题.

2) 固态盘提供块设备访问接口,为上层应用提供了类似磁盘的透明访问,但同时也屏蔽了闪存的

Table 1 Comparison of Various SSD Arrays

表 1 各固态盘阵列的对比评价

Approach	Performance				Reliability			Cost
	Random Write	Small Write	Garbage Collection	Load Balance	Wear Leveling	P/E Cycles	Redundancy	
Diff-RAID					√			
WeLe-RAID				√	√			
HPDA	√	√				√		√
HerpRap	√	√				√		√
I-CASH	√							√
GS-RAIS			√					
FlashGraph	√			√	√			
FRA		√				√		
Flash-Aware RAID		√	√			√		
SDF	√	√	√		√			
Lifespan-Aware Scheme		√				√	√	
eSAP-RAID	√	√	√		√			
MuLe-RAID			√	√				
Hybrid SSD				√		√		√
DA-RAID-5					√		√	

部分优势.因此,基于固态盘的 RAID 系统不能有效地利用固态盘的内在特点,只能通过调整校验信息在各个盘上的分布、利用缓存随机写请求等方式粗粒度地解决固态盘存在的介质损耗和随机写性能差等问题.

3) 要想从根本上解决固态盘的缺陷,充分利用闪存存储的优势,就必须考虑固态盘内部的工作原理,通过直接对芯片的控制来优化性能.在闪存转换层不仅要设计出良好的地址映射、垃圾回收和磨损均衡算法,还要实现 RAID 机制保证数据的可靠性.

4) 基于芯片的 RAID 系统不能满足高容量、高可靠性、高性能的需求,基于固态盘的 RAID 系统不能有效地利用固态盘的特点,如何结合这 2 种构建方法以充分挖掘闪存芯片间和 SSD 设备间的并行性为下一步的研究方向.不仅解决固态盘自身所存在的随机写、介质损耗、RAID 阵列的小写等问题,同时实现系统的高容量、高性能、高可靠.

5 总结与展望

本文首先分析了基于 Flash 的固态盘和 RAID 技术所存在的问题,并且针对二者的特性选取性能、可靠性和价格作为阵列的评价标准.分别从固态盘和 Flash 芯片 2 个层面深入分析了构建 RAID 阵列的关键技术,并指出了各种方法的优缺点,最后总结了 2 种构建模式,并指出了未来可能的研究方向.

固态盘相对与传统磁盘的优势越来越明显,但是其与传统磁盘有着迥然不同的设备特性.为了优化基于固态盘的 RAID 系统的性能,我们需要设计一系列的策略来利用这样的差异,并在带宽、并行度等方面进行扩展以适应 SSD 更大的吞吐率.

参 考 文 献

[1] Im S, Shin D. Flash-aware RAID techniques for dependable and high-performance flash memory SSD[J]. IEEE Trans on Computers, 2011, 60(1): 80-92

[2] Agrawal N, Prabhakaran V, Wobber T, et al. Design tradeoffs for SSD performance [C] //Proc of the USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2008: 57-70

[3] Patterson D A, Gibson G, Katz R H. A case for redundant arrays of inexpensive disks (RAID)[C] //Proc of the 1988 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 1988: 109-116

[4] Mao Bo, Jiang Hong, Wu Suzhen, et al. HPDA: A hybrid parity-based disk array for enhanced performance and reliability[J]. ACM Trans on Storage, 2012, 8(1): 80-92

[5] Kim H, Lee S. A new flash memory management for flash storage system [C] //Proc of the 23rd Annual Int Computer Software and Applications Conf. Piscataway, NJ: IEEE, 1999: 284-289

[6] Chang L P, Huang L C. A low-cost wear-leveling algorithm for block-mapping solid-state disks [C] //Proc of the 2011 SIGPLAN/SIGBED Conf on Languages, Compilers and Tools for Embedded Systems. New York: ACM, 2011: 31-40

[7] Chang L P. On efficient wear leveling for large-scale flash-memory storage systems [C] //Proc of the 2007 ACM Symp on Applied Computing. New York: ACM, 2007: 1126-1130

[8] Chiang M L, Cheng C L, Wu C H. A new FTL-based flash memory management scheme with fast cleaning mechanism [C] //Proc of IEEE Int Conf on Embedded Software and Systems. Piscataway, NJ: IEEE, 2008: 205-214

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

[10] Ko S, Jun S, Kim K, et al. Study on garbage collection schemes for flash-based Linux swap system [C] //Proc of Advanced Software Engineering and Its Applications. Piscataway, NJ: IEEE, 2008: 13-16

[11] Lee J, Kim Y, Shipman G M, et al. A semi-preemptive garbage collector for solid state drives [C] //Proc of IEEE Int Symp on Performance Analysis of Systems and Software. Piscataway, NJ: IEEE, 2011: 12-21

[12] 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)

[13] Du Yimo. Research on RAID mechanism in storage system based on flash memory [D]. Changsha: National University of Defense Technology, 2010 (in Chinese)
(杜溢墨. 闪存存储系统中的 RAID 机制研究[D]. 长沙: 国防科学技术大学, 2010)

[14] Balakrishnan M, Kadav A, Prabhakaran V, et al. Differential RAID: Rethinking RAID for SSD reliability [J]. ACM Trans on Storage, 2010, 6(2): 1-22

[15] Du Yimo, Liu Fang, Chen Zhiguang, et al. WeLe-RAID: A SSD-based RAID for system endurance and performance [C] //Proc of the 8th IFIP Int Conf on Network and Parallel Computing. Berlin: Springer, 2011: 248-262

[16] Zeng Lingfang, Feng Dan, Mao Bo, et al. HerpRap: A hybrid array architecture providing any point-in-time data tracking for datacenter [C] //Proc of IEEE Int Conf on Cluster Computing. Piscataway, NJ: IEEE, 2012: 311-319

[17] Ajtai M, Burns R, Fagin R, et al. Compactly encoding unstructured inputs with differential compression [J]. Journal of the ACM, 2002, 49(3): 318-367

[18] Broder A Z. Identifying and filtering near-duplicate documents [C] //Proc of Combinatorial Pattern Matching. Berlin: Springer, 2000: 1-10

[19] Morrey III C B, Grunwald D. Peabody: The time travelling disk [C] //Proc of the 20th IEEE/the 11th NASA Goddard Conf on Mass Storage Systems and Technologies. Piscataway, NJ: IEEE, 2003: 241-253

[20] Yang Qing, Xiao Weijun. TRAP-Array: A disk array architecture providing timely recovery to any point-in-time [C] //Proc of the 33rd Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2006: 289-301

[21] Yang Qing, Ren Jin. 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

[22] Zheng D, Mhembere D, Burns R, et al. FlashGraph: Processing billion-node graphs on an array of commodity SSDs [C] //Proc of the 13th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2015: 45-58

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

[24] 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

[25] Lee Y, Jung S, Song Y H. FRA: A flash-aware redundancy array of flash storage devices [C] //Proc of the 7th IEEE/ACM Int Conf on Hardware/Software Codesign and System Synthesis. New York: ACM, 2009: 163-172

[26] Ouyang Jian, Lin Shiding, Jiang Song, et al. SDF: Software-defined flash for Web-scale Internet storage systems [C] //Proc of ACM SIGPLAN Notices. New York: ACM, 2014: 471-484

[27] Grupp L M, Caulfield A M, Coburn J, et al. Characterizing flash memory: Anomalies, observations, and applications [C] //Proc of the 42nd Annual IEEE/ACM Int Symp on Microarchitecture. New York: ACM, 2009: 24-33

[28] Mielke N, Marquart T, Wu N, et al. Bit error rate in NAND flash memories [C] //Proc of IEEE Int Reliability Physics Symp. Piscataway, NJ: IEEE, 2008: 9-19

[29] Lee S, Lee B, Koh K, et al. A lifespan-aware reliability scheme for RAID-based flash storage [C] //Proc of the 2011 ACM Symp on Applied Computing. New York: ACM, 2011: 374-379

[30] Kim J, Lee J, Choi J, et al. Improving SSD reliability with RAID via elastic striping and anywhere parity [C] //Proc of the 43rd Annual IEEE/IFIP Int Conf on Dependable Systems and Networks. Piscataway, NJ: IEEE, 2013: 1-12

[31] Du Yimo, Xiao Nong, Liu Fang, et al. MuLe-RAID: Constructing large scale and high performance SSD with multiple level RAID architecture [J]. Journal of Computer Research and Development, 2012, 49(Suppl Ⅰ): 111-117 (in Chinese)
(杜溢墨,肖侬,刘芳,等. MuLe-RAID:面向大容量高性能SSD的层次化RAID[J]. 计算机研究与发展,2012,49(增刊Ⅰ):111-117)

[32] Chang L P, Du C D. Design and implementation of an efficient wear-leveling algorithm for solid-state-disk microcontrollers [J]. ACM Trans on Design Automation of Electronic Systems, 2009, 15(1): 1-36

[33] LeventhaL A. Flash storage memory [J]. Communications of the ACM, 2008, 51(7): 47-51

[34] Roberts D, Kgil T, Mudge T. Integrating NAND flash devices onto servers [J]. Communications of the ACM, 2009, 52(4): 98-103

[35] Trinh C, Shibata N, Nakano T, et al. A 5.6 MB/s 64Gb 4b/cell NAND flash memory in 43nm CMOS [C] //Proc of IEEE Int Solid-State Circuits Conf. Piscataway, NJ: IEEE, 2009: 246-247

[36] Sun Guangyu, Dong Xiangyu, Xie Yuan, et al. A novel architecture of the 3D stacked MRAM L2 cache for CMPs [C] //Proc of the 15th IEEE Int Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2009: 239-249

[37] Chang L P. Hybrid solid-state disks: Combining heterogeneous NAND flash in large SSDs [C] //Proc of Design Automation Conf. Piscataway, NJ: IEEE, 2008: 428-433

[38] Guo Jie, Wen Wujie, Zhang Yaojun, et al. DA-RAID-5: A disturb aware data protection technique for NAND flash storage systems [C] //Proc of Design, Automation & Test in Europe Conf & Exhibition. Piscataway, NJ: IEEE, 2013: 380-385

Li Xiangnan, born in 1991. Master candidate. His main research interests include design and implementation of the storage system.

Zhang Guangyan, born in 1976. Associate professor. His main research interests include big data computing, network storage, and distributed systems.

Li Qiang, born in 1975. Associate professor. His main research interests include network security.

Zheng Weimin, born in 1946. Professor. His main research interests include big data computing, network storage, parallel compiler, and distributed systems.