DOI:10.7544/issn1000-1239.2019.20170832
56(4):755-766,2019

基于工作负载感知的固态硬盘阵列系统的架构设计与研究

张 强[1] 梁 杰[1] 许胤龙[1,2] 李永坤[1,2]
[1](中国科学技术大学计算机科学与技术学院 合肥 230026)
[2](安徽省高性能计算重点实验室(中国科学技术大学) 合肥 230026)
(zhgqiang@mail.ustc.edu.cn)

Research of SSD Array Architecture Based on Workload Awareness

Zhang Qiang[1], Liang Jie[1], Xu Yinlong[1,2], and Li Yongkun[1,2]
[1](*School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230026)
[2](*Anhui Province Key Laboratory of High Performance Computing* (*University of Science and Technology of China*), *Hefei* 230026)

Abstract The fixed data layout of traditional array system and the locality of workloads cause the partial disks of the array system to become hot disks, which affects the reliability and the overall concurrency performance of the array system. This paper proposes a new RAID architecture for SSD array systems, HA-RAID, which leverages hot/cold data separation and sliding window techniques. The main idea is that HA-RAID divides the disk array into hot disks and ordinary disks, which stores hot data on hot disks and cold data on ordinary disks, and it changes the role of each disk dynamically by moving a fixed-length sliding window. So, each disk has the opportunity to become a hot disk and stores hot data which achieves the purpose of storing hot data evenly on each disk. Experiments under real-world workloads on a RAID-0 array system composed of eight commercial SSDs show that HA-RAID can achieve an even distribution of hot data across all disks and reduce the percentage of hot disks appearing in the array to almost zero. This implies that HA-RAID achieves load balance and wear balance at the device level. In terms of performance, HA-RAID reduces the average response time by 12.01%~41.06% which achieves the I/O performance enhancement, compared with traditional RAID-0 array.

Key words SSD array architecture design; cold and hot data identification; RAID-0; load balance; response time

摘 要 在RAID-0阵列中设计了一种基于冷热数据分离存储的固态硬盘阵列系统架构HA-RAID,并结合滑动窗口技术进行优化.其主要思想是,利用固定长度的滑动窗口将阵列系统中的盘划分为普通盘和热点盘,热点盘专门存放热数据,普通盘存放普通数据,且阵列中各盘的角色随着滑动窗口的移动而不断变化.在8块商用固态硬盘组成的RAID-0阵列系统上对HA-RAID予以实验分析.实验结果表明,相比于未引入冷热数据分离机制的原始RAID-0做法,HA-RAID可以将热数据相对均匀地存储到各个盘上,很好地实现了阵列系统级的负载和磨损均衡,从而将阵列中热点盘出现的比例降低到几乎为0.在真实的企业级工作负载下,相比原始RAID-0,HA-RAID减少12.01%~41.06%的平均响应时间,很好地实现了阵列系统级的I/O性能提升.

收稿日期:2017-11-03;**修回日期**:2018-03-05
通信作者:许胤龙(ylxu@ustc.edu.cn)

关键词 SSD 阵列架构设计;冷热数据区分;RAID-0;负载均衡;响应时间

中图法分类号 TP333

大数据时代,应对海量数据的存储分析,基于固态硬盘的容错阵列系统能够很好地解决海量数据的可靠存储和高效访问的困难.原因有 2 方面:1)容错阵列系统既能通过多盘 I/O 提供数据的高并发性,又能通过存储冗余信息保障数据的可靠性;2)固态硬盘相比机械硬盘,因其具有更高访问性能、更低能耗和更强抗震性等特点,进一步提升容错阵列系统的性能.

多块固态硬盘组成的容错阵列系统,往往按照轮转(round-robin)的顺序,并行地在各个固态硬盘上读写数据(即每块盘的地位相同).由于数据本身具有时间局部性及空间局部性的特点,易造成局部盘成为热点盘,进而会造成整个固态硬盘阵列的性能出现瓶颈(极端情况下会出现单盘热点),又由于固态盘有寿命限制,对某些盘的过多写易造成固态盘故障等问题,也会对整个阵列可靠性造成影响.本文在由 8 个固态硬盘(solid state drive, SSD)构成的 RAID-0(redundant array of independent disks 0)阵列系统中测试不同企业级工作负载,均发现热点盘的现象.

如何保证阵列系统级的负载和磨损均衡以及如何区分各个固态硬盘在阵列中的角色,设计合理的固态硬盘阵列系统架构,对提升固态硬盘阵列系统整体性能及可靠性都有很重要的意义.

本文的主要贡献有 2 个方面:

1) 利用工作负载感知特性,在 RAID-0 阵列中设计了一种基于冷热数据分离存储的固态硬盘阵列系统架构 HA-RAID(hotness aware RAID),并结合滑动窗口技术进行性能优化.实验结果表明,HA-RAID 可将热数据相对均匀地存储到各个盘上,很好地实现了阵列系统级的负载和磨损均衡,从而将阵列中热点盘出现的比例降低到几乎为 0.

2) 在真实的企业级工作负载下,相比传统 RAID-0,HA-RAID 可减少 12.01%~41.06%的平均响应时间,很好地实现了阵列系统级的 I/O 性能提升.

1 相关研究工作

近年来,国内外开展了大量针对固态硬盘的相关研究.其中,许多工作关注固态硬盘本身的内部结构,进而分析并设计高效的垃圾回收机制,以提升固态硬盘的读写性能、延长使用寿命;另外,还有不少工作关注多块固态硬盘构成的存储阵列系统,利用数据冗余提高数据的可靠性、延长固态硬盘阵列系统寿命并提升阵列系统性能,这些工作可以分别被概括为 2 方面:

1) 对于单个固态硬盘的研究.关注固态硬盘的内部结构,如文献[1]探索了固态硬盘的内部组成以及并行化等,同时实现相应的固态硬盘模拟器.关注固态硬盘逻辑地址到物理地址的映射方式,如文献[2-3]通过探索在不同的应用场景采用不同的地址映射方式来优化固态硬盘的性能.此外,还有关注固态硬盘的垃圾回收过程,如文献[4]通过建立分析模型探究了不同垃圾回收算法在垃圾回收开销和擦写均衡性之间的权衡关系,提出了可以根据需求调整的 d-choice 垃圾回收算法;文献[5-6]通过建立理论模型分析了固态硬盘垃圾回收开销与工作负载之间的关系,证明了将冷热数据分离存储在固态硬盘中能够大大改善垃圾回收性能;文献[7-8]提出了不同的热数据识别方法,以部署到固态硬盘中改善其读写性能、减少其垃圾回收开销等.

2) 对于固态硬盘阵列系统的研究.文献[9]研究了由多块固态硬盘构成的阵列系统,通过在多块固态硬盘之间引入冗余数据,进一步提高了系统的可靠性;文献[10]针对于固态硬盘的特点,构建了新型的适用于固态硬盘的文件系统;文献[11-12]通过构建可靠性分析模型对基于固态硬盘阵列系统进行了可靠性评估;文献[13]通过引入弹性条带机制解决了传统磁盘校验块更新方式不适用于固态硬盘阵列的问题等.

综合海量存储的背景和国内外研究现状可知,将固态硬盘应用于大规模存储系统(例如阵列系统)已经成为了当前的趋势和热点.尽管已经有很多针对于固态硬盘及其阵列系统的性能优化研究,但是其中仍然存在着很多的不足,特别是从感知工作负载的角度来优化固态硬盘及其阵列系统的性能方面,仍有着进一步的改善空间.如在固态硬盘中部署冷热数据识别方法用于冷热数据分离存储,以减少热点盘出现的概率,实现阵列级负载和磨损均衡,已有的热数据方法并不能很好地适应工作负载的变化.

2　阵列系统原始架构

在 Linux 内核的 MD 模块,固态硬盘阵列系统以条带的形式对阵列的存储空间进行统一编址.每个请求数据的逻辑地址根据阵列系统的编址方式,映射到某块盘上的固定逻辑地址(固态盘内部闪存转换层会进一步对该逻辑地址映射为固态盘上真实物理地址,该映射方式限定在单个固态盘内部映射,并不影响数据在阵列级别的分布).由 4 块固态盘组成 RAID-0 阵列系统如图 1 所示.图 1 中数字 0…7,8…15,分别代表扇区(sector,大小为 512 B)0 到 7 以及扇区 8 到 15;1 个页(page,大小为 4 KB)包含 8 个扇区,如图 1 中,page 0 包含扇区 0 到 7;1 个块(chunk)包含 2 个页,图 1 中的扇区 0 到 15 构成 1 个块;1 个条带(stripe)包含 4 个块,条带大小可表示为

$$stripe\_size=4\times chunk\_size=4\times 2\ \text{page}=4\times 2\times 8\ \text{sector}=64\ \text{sector}.$$

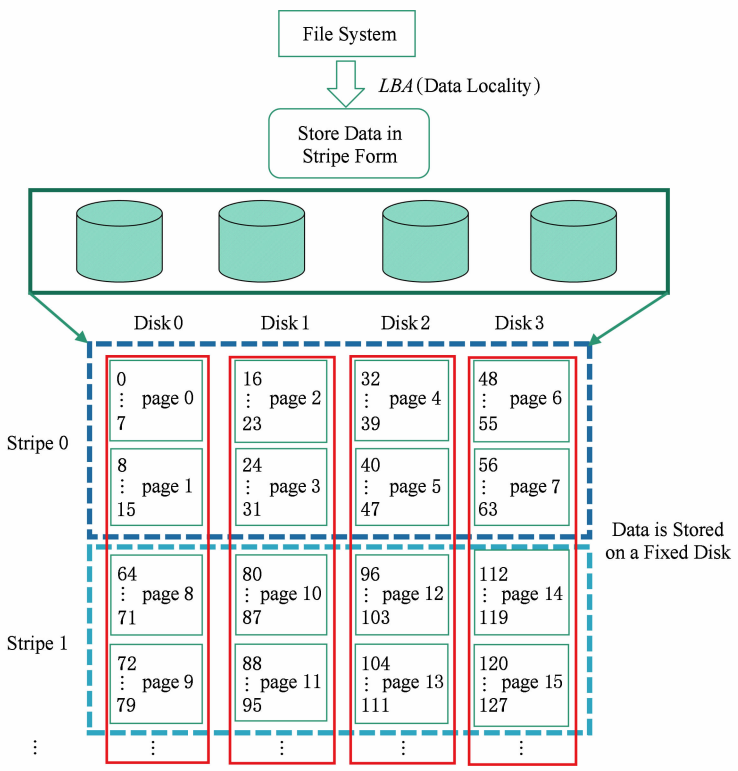


Fig. 1　The original architecture diagram of SSD array

图 1　固态硬盘阵列系统原始架构图

如果请求的数据(扇区编号为 32～39)为热数据,根据阵列存储空间的编址方式,计算得到扇区编号为 32～39 的数据存放的物理地址:

$$stripe\_num=32/stripe\_size=32/64=0,$$

$$chunk\_num=32/chunk\_size=32/16=2,$$

$$dd\_idx=chunk\_num\%disk\_num=2\%4=2,$$

$$chunk\_offset=32\%chunk\_size=32\%16=0,$$

$$sector\_num=stripe\_num\times chunk\_size+chunk\_offset=0\times 16+0=0.$$

因此,扇区编号为 32～39 的热数据存放在 2 号盘上,并且盘上的起始物理扇区号为 0.各应用系统请求的数据具有时间及空间局部性原理,对某些数据的过多访问会造成对部分盘的过多读写,从而导致热点盘的出现.例如,对扇区编号 32～39 的热数据频繁访问,将导致对 2 号盘的过度读写,从而使 2 号盘变为热点盘.如果大部分 I/O 请求集中在 2 号盘,则不能充分发挥 RAID 阵列高并发的优势,进而造成整个阵列系统的 I/O 性能出现瓶颈.固态盘有擦写次数寿命限制,对某些盘的过多写,也会对整个阵列可靠性造成影响.

3　数据冷热区分算法

为避免第 2 节中提到的热点盘出现的问题,需对数据进行冷热区分,并对区分过的数据进行分别处理.设计冷热数据区分算法,首先要求其识别结果的

有效性,否则不能达到冷热数据分离存储对存储系统性能的优化;其次要求算法识别过程的计算开销要尽可能小,以免降低存储系统的请求处理吞吐量.

先前的研究者们已经提出了很多有效的冷热数据区分算法,其中比较典型的冷热数据区分算法分别是:基于访问频次的算法 DAMS[14]、基于缓存替换的算法 Two-level LRU list[15]、基于访问频次和 LRU 分组的算法 GLRU[16]、基于访问频率和最近信息的算法 MBF[14-17]、基于采样技术的算法 HotDataTrap[17-18].

DAMS 假定可用的内存空间无限大,采用直接地址写次数统计方法,为每个逻辑块地址(logical block address, LBA)设立一个计数器,记录其写统计次数,图 2 为其算法流程. 一个 LBA 请求页被识别为热数据,当且仅当其写统计次数大于等于预定义的阈值.

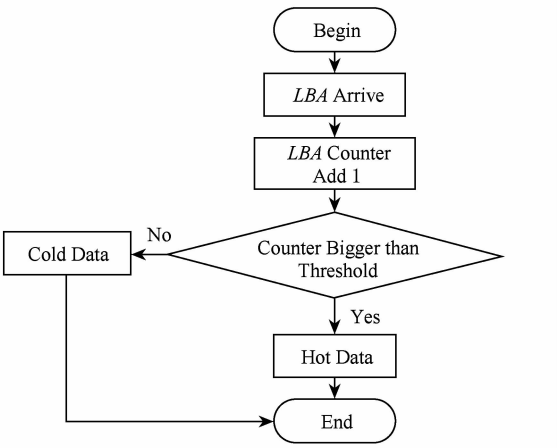


Fig. 2 Flow chart of DAMS algorithm

图 2 DAMS 算法流程图

不同的工作负载一般表现出不同的访问特征,为了能够精确地捕获各种工作负载的访问特征,保证冷热数据划分的合理性,采取以下方法:统计最近访问数据的修改频率来动态调整热度阈值,并对每个逻辑地址的计数器值进行定期减半的衰退操作以反映工作负载的动态性. 实验结果证明该方法能准确地识别出热数据,从而很好地实现了阵列系统的负载均衡.

考虑到计算开销对存储系统的请求响应时间的影响,因此本文在设计和实现固态硬盘阵列系统架构时采用最理想的 DAMS 作为冷热数据区分算法,DAMS 只需很小的计算开销,但需要较大的内存开销来记录访问频次,4.3 节会进一步讨论如何对内存开销进行优化.

4 基于热度感知的阵列系统架构设计研究

本节我们利用工作负载感知特性,在 RAID-0 阵列中设计了一种基于冷热数据分离存储和滑动窗口的固态硬盘阵列系统架构,以此来减少热点盘出现的概率,使阵列系统达到负载和磨损均衡.

4.1 架构设计

利用第 3 节介绍的冷热数据识别方案,我们设计了基于冷热数据分离存储和滑动窗口的 HA-RAID 阵列系统架构,如图 3 所示.

基于冷热数据分离存储和滑动窗口的 HA-RAID 阵列系统架构由 4 个模块组成:

1) 冷热数据识别模块. 用于对到来的请求数据进行冷热识别,以用于在固态硬盘阵列系统中进行冷热数据的分离存储. 该模块用第 3 节介绍的典型的冷热数据区分算法来实现.

2) 滑动窗口模块. 该模块用来变换各个盘的角色,使每个盘都有机会成为热点盘,从而达到将热数据均匀放置在每个盘上的目的,实现阵列水平的磨损均衡. 如图 3 所示,设定滑动窗口大小为 4,初始窗口位置为 1~4 号盘. 当处理一定数量的连续请求后(本文设置的滑动阈值是 1 000),窗口以轮转的方式向右滑动 1 个位置,移出窗口的盘由普通盘变为热点盘,移入窗口的盘由热点盘变为普通盘. 对于识别为热的数据,将其存储到热点盘,普通数据存储到普通盘.

3) 映射表模块. 该模块用来记录请求的逻辑地址到阵列中物理地址的映射关系,由于固态硬盘读写的最小单位为页(4 KB),并且本文所用工作负载全部 4 KB 对齐,因此映射表中只需记录数据起始地址除以 8 后的值即可. 本文用数据结构(LBA, $SSDN$, PSN)来实现该映射表,其中 LBA 表示请求数据块的起始逻辑地址(起始逻辑扇区号)除以 8 后的值,$SSDN$ 表示请求数据块在 SSD 阵列中被存储的盘号,PSN 表示请求数据块被存储的起始物理地址(起始物理扇区号) 除以 8 后的值.

4) 编址模块. 不同于原始架构以条带的形式对阵列进行地址空间的编排,我们设计的架构中根据冷热数据识别结果对请求数据的逻辑地址在阵列中乱序编址,数据在每块盘上顺序存放,当有数据需要存储到盘上时,以顺序的方式为其分配物理地址,并更新映射表模块对应的表项. 如图 3 所示,下半部分的编址空间中,扇区 0…7,存储于 Disk3 的 2 号物理位置,而右上角处的映射表模块的表项 0,

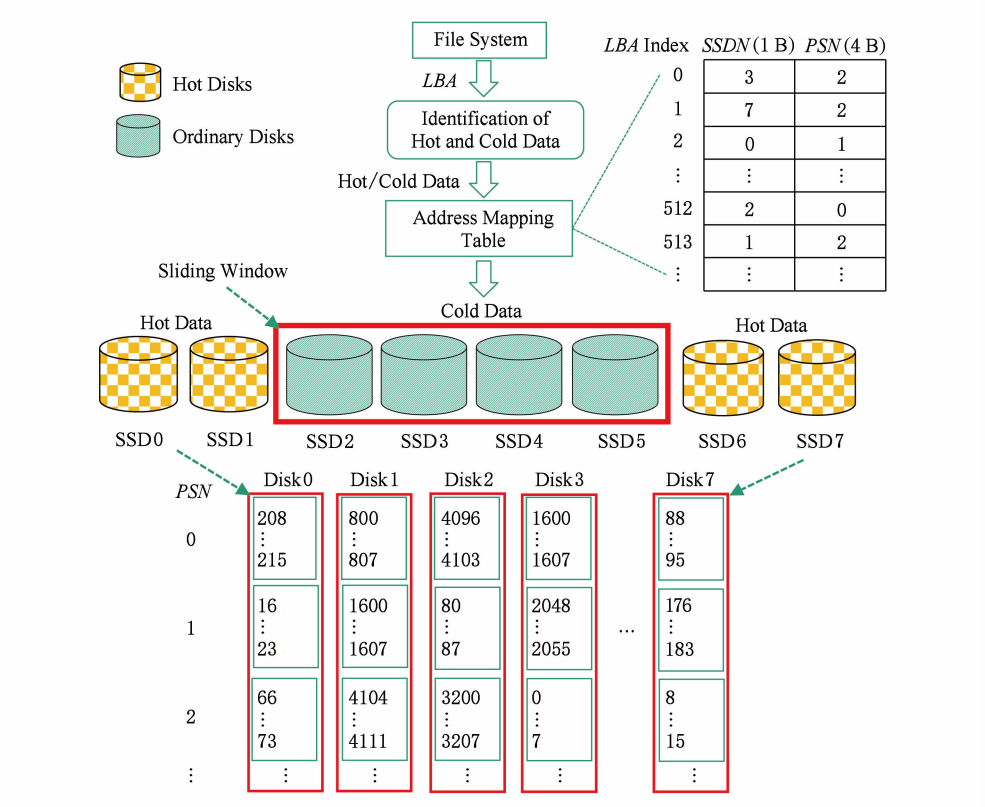


Fig. 3　HA-RAID architecture based on cold/hot data separation and sliding window

图 3　基于冷热数据区分和滑动窗口的 HA-RAID 系统架构

即 page 0(对应扇区 0…7),记录 $SSDN$ 为 3 号盘,且 $PSN=2$.

4.2　处理流程

基于冷热数据分离存储和滑动窗口的 HA-RAID

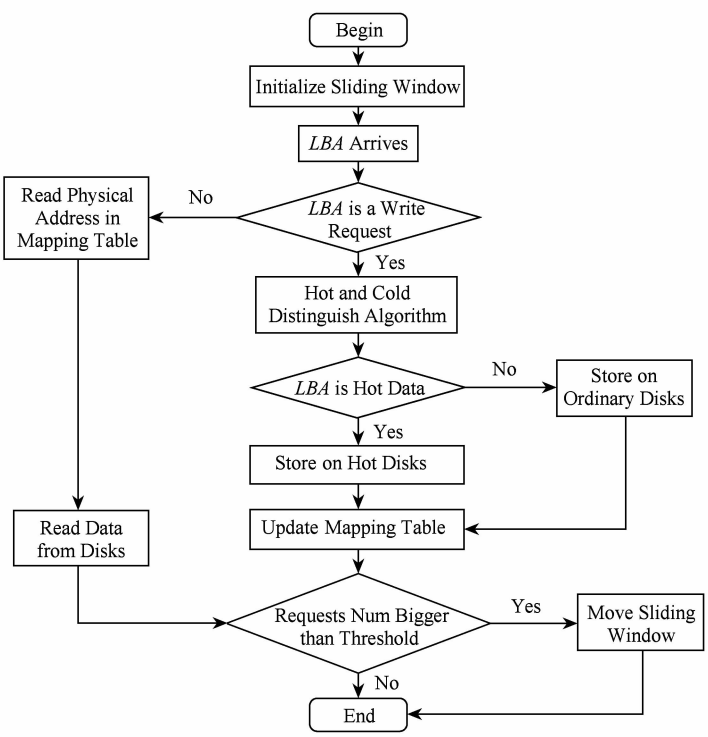


Fig. 4　The I/O flow chat of HA-RAID

图 4　HA-RAID 的 I/O 处理流程

阵列系统架构,对 I/O 请求的基本处理流程如图 4 所示. 首先对阵列中的滑动窗口进行初始化(如滑动窗口大小、滑动窗口初始位置、滑动窗口滑动周期等),然后开始处理文件系统层下发的请求.

对于文件系统层下发的请求 LBA,首先判断它是读请求还是写请求,如果为读请求,则根据维护的地址映射表得到其物理地址(盘号和扇区号),然后在底层盘上读取数据,完成此次 I/O 请求;如果为写请求,则利用冷热数据识别模块对其进行冷热识别:

1) 如果该请求 LBA 被识别为热数据,则将其以轮转的方式存储到热点盘中(数据在盘中根据请求先后顺序,顺序编址并存放),然后更新地址映射表中对应的物理地址,完成此次 I/O 请求.

2) 如果该请求 LBA 被识别为冷数据,则将其以轮转的方式存储到普通盘中(数据在盘中根据请求先后顺序,顺序编址并存放),然后更新地址映射表中对应的物理地址,完成此次 I/O 请求.

最后判断连续请求的数量是否超过滑动窗口的滑动阈值,如果超过,则将滑动窗口向右滑动 1 个盘的距离,继续处理下一个 I/O 请求;否则,结束此次 I/O 请求并开始处理下一个 I/O 请求.

4.3 实现细节

本文在 Linux 内核 MD 模块下的 RAID-0 中用 C 语言实现 SSD 阵列系统架构,这里将详细介绍 SSD 阵列系统架构实现过程中的一些实现细节,如 Linux 内核 MD 模块的层次结构图,实现逻辑地址到物理地址转换的映射表的构造与管理,处理请求过程中的数据对齐.

1) MD 模块层次结构. Linux 内核响应块设备读写请求的层次结构,主要由系统调用层、文件系统层、块 I/O 子系统、SCSI(small computer system interface)/SATA(serial advanced technology attachment)子系统和底层存储设备组成. 本文通过修改图 5 中块设备驱动层的 MD 驱动下的 RAID-0 源码来实现冷热数据区分算法和基于冷热数据分离存储的 SSD 阵列系统架构,以实现底层设备阵列中各盘的负载和磨损均衡.

2) 映射表的构造与管理. 由于在本文设计的系统架构中,同一个写请求不是固定写到一块盘的固定物理地址上,而是每次都写到不同盘的不同物理地址上,以实现负载和磨损均衡. 因此需要构造与管理一张映射表来实时记录请求数据块的逻辑地址到 SSD 盘号和该盘上物理地址的映射关系. 用数据结构 $(LBA, SSDN, PSN)$ 来实现该映射表,如图 6 所示,其中 LBA 表示请求数据块的起始逻辑地址(起始逻辑扇区号)除以 8 后的值,$SSDN$ 表示请求数据块在 SSD 阵列中被存储的盘号,PSN 表示请求数据块被存储的起始物理地址(起始物理扇区号)除以 8 后的值. 使用双数组进行阵列系统中映射表的管理.

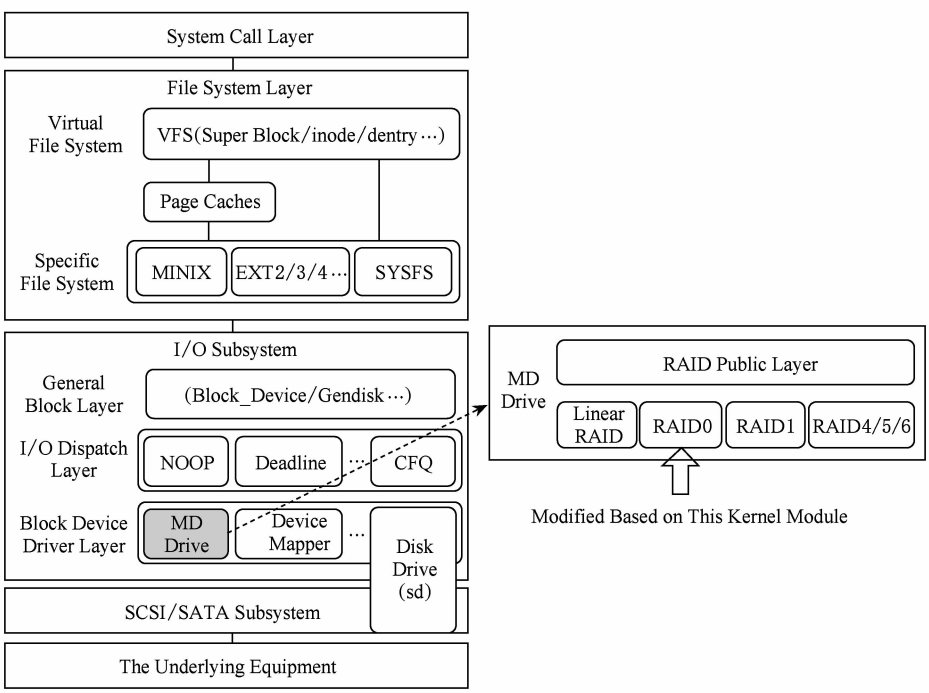


Fig. 5 The hierarchy diagram of Linux kernel block devices for read and write requests

图 5 Linux 内核块设备读写请求的层次结构图

LBA Index	*SSDN*(1 B)	*PSN*(4 B)
1	6	512
2	8	2 048
3	1	32
⋮	⋮	⋮
N	5	128

Fig. 6　The mapping table

图 6　映射表示意图

LBA 用于双数组中的索引,无需存储;*SSDN* 使用 1 B 存储,能够存储最多 256 块盘;*PSN* 使用 4 B 存储,能够存储的扇区号最多为 4 294 967 296,满足 120 GB 固态硬盘的容量要求. 这样仅用 5 B 就足够记录 1 个数据块的映射关系,对于固态硬盘阵列 1 TB 的数据空间仅仅需要的映射表空间为 1.25 GB.

3) 数据对齐. 在系统架构实现中,设定系统的读写单位为一个特定大小 4 KB 的数据块 chunk. 为了实现读写请求的数据对齐,当一个到来的请求不能与设定的 chunk 对齐,若小于 chunk 大小,则将其补充到 chunk 大小;若大于 chunk 大小,则将其补充到 chunk 的整数倍. 这样做是因为固态硬盘的读写基本单位为 4 KB,并且实验中设置 chunk 大小为 4 KB,为了让一次 chunk 大小的请求集中在 1 个块,而不是操作多个块(比如未对齐时,1 个 chunk 大小的请求可能会操作 2 个块,写入数据时需要读-擦-写,造成性能下降;对齐时,1 个 chunk 大小的请求只会操作 1 个块,从而提升固态硬盘性能).

5　实验与结果

为了进行评估,本文将设计的系统架构部署到由多个商用固态硬盘组成的阵列系统上,并在系统中使用了 4 个真实的企业级工作负载. 由于本文的方案是在原始 RAID-0 阵列系统上进行的改进,故将原始 RAID-0 用作本文的实验基准,并从 2 个方面对 RAID-0 和 HA-RAID 进行对比,即固态硬盘阵列系统的负载均衡以及请求的平均响应时间.

5.1　实验设置

1) 工作负载. 实验中使用了企业数据中心的块级 traces 工作负载,分别为写占主导的 *hm*_0,*src*2_0,*usr*_0 和 *stg*_0. 其中,*hm*_0 代表硬件监控服务器上的负载,*src*2_0 代表源控制服务器上的负载,*usr*_0 代表用户主目录服务器上的负载,*stg*_0 代表 Web 分段服务器上的负载[19].

我们在实验中对工作负载进行了预先处理,即数据 4 KB 对齐. 所有工作负载的工作集总大小在 10 GB 到 20 GB 之间;与此同时,所有的工作负载具有很强的访问倾斜性和很高的访问局部性. 例如在后 3 个工作负载中,大部分写请求集中在 2%～4% 的地址空间,而第 1 个工作负载中,大部分的写请求集中在 10%的地址空间. 各工作负载的统计数据如表 1 所示.

2) 实验环境. 本文将 HA-RAID 阵列系统架构在物理机器上加以实现. 实验硬件配置为:戴尔 PowerEdge T620 服务器(配有 4 个 2.40 GHz Intel Xeon CPU 和 8 GB 物理内存空间),1 个 Intel 530 系列固态硬盘用作系统盘,8 个 Intel 530 系列固态硬盘用于构建一个固态硬盘阵列,每个盘的容量 120 GB,SATA 3 接口,MLC(multi-level commission)类型. 阵列设置数据块 chunk 的大小为 4 KB. 使用 FIO 工具测试了固态硬盘的随机读写延时(50% 读,50%写,I/O 大小 4 KB,队列深度设置为 1,单线程,并且绕过磁盘缓存),测试结果如表 2 所示.

Table 1　Statistical Data of Different Workloads

表 1　不同工作负载的统计数据

Workload	Total Request	Proportion of Write/%	Different Write Ratio/%	Different Read Ratio/%	Working Set/GB	Different Write Percentage of Work Set/%	Different Read Percentage of Work Set/%
*hm*_0	3993316	64.50	7.15	16.95	13.94	11.73	13.36
*src*2_0	1557732	88.66	5.05	28.86	15.63	3.20	2.56
*usr*_0	2238009	59.58	4.86	5.98	15.9	4.07	13.30
*stg*_0	2030985	84.81	2.50	83.08	10.82	3.64	56.75

3) 冷热区分算法参数设置. 由于本文重点研究阵列系统的磨损均衡和性能提升,并不对冷热数据区分算法做深入研究,留在以后再探究. 因此在实验中使用计算时间开销最少的 DAMS 作为本文设计的阵列系统架构中的冷热数据区分算法. 不同工作负载下 DAMS 算法的参数设置如表 3 所示.

Table 2 Performance Indicators of Intel 530 Series SSD

表 2 Intel 530 系列固态硬盘实际性能指标 μs

Device	R/W Delay
sdb	161
sdc	172
sdd	186
sde	182
sdf	183
sdg	187
sdh	138
sdi	174

Table 3 Parameter Setting of DAMS Algorithm Under Different Workloads

表 3 不同工作负载下 DAMS 算法的参数设置

Workload	Hot Threshold	Decay Interval
hm_0	68	100 000
src2_0	58	50 000
usr_0	110	80 000
stg_0	16	80 000

Note: Under various workloads, the parameters of DAMS algorithm are experimentally tested and the best value is taken.

5.2 系统请求平均响应时间评估

实验中,通过记录每个请求的响应时间和系统处理请求的总个数,计算得到每个请求的平均响应时间,实验结果展示在图 7 中.

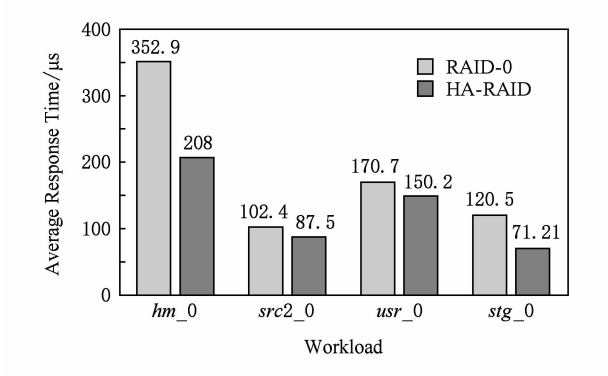


Fig. 7 The average response time of I/O requests

图 7 不同工作负载下系统请求的平均响应时间

通过图 7 可知,采用阵列系统架构 HA-RAID,相比于原始架构 RAID-0,部署在实际系统上时,极大地减少了请求的平均响应时间. 实验结果验证了区分数据冷热的有效性,因为原始架构会因为冷热数据的混合存储而导致热点盘的出现,从而不能发挥 RAID 阵列高并发的优势,进而会一定程度上延迟用户的访问请求,造成整个阵列系统的 I/O 性能出现瓶颈. 对比原始架构 RAID-0,本文设计的阵列

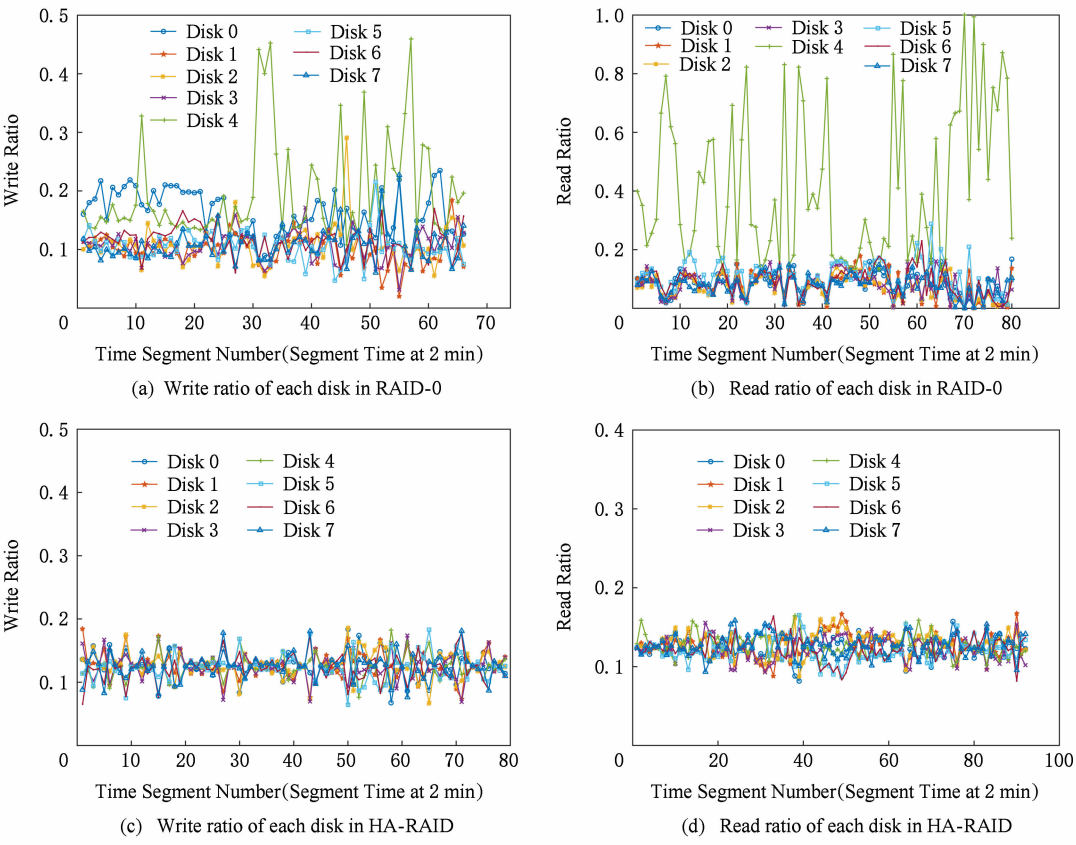


Fig. 8 Read/write ratio of each disk in RAID-0 and HA-RAID under *hm_0*

图 8 负载 *hm_0* 下 RAID-0 与 HA-RAID 阵列中各盘上的读写比例

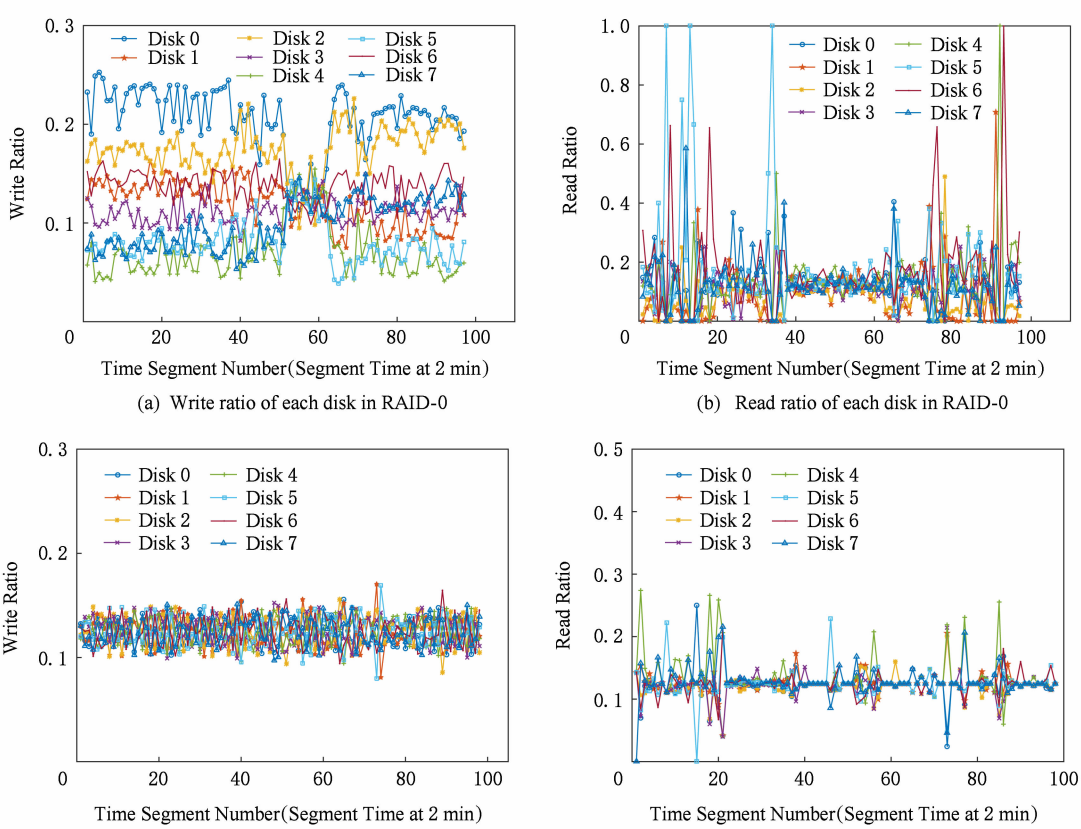
系统架构 HA-RAID 部署在系统上时,对请求的平均响应时间减少了 12.01%～41.06%,有效提高固态硬盘阵列系统的 I/O 性能.

5.3　系统负载均衡评估

固态硬盘容错阵列系统的负载和磨损不均衡,都容易降低系统的可靠性,造成数据的遗失.因此,本文实验统计了各种工作负载下阵列系统中各个盘的负载和磨损情况,统计结果如图 8～11 所示.

由图 8 所示可知,工作负载为 *hm*_0 时,对于原始架构 RAID-0,各盘上的读写比例在每个时间点都有较大差异,且波动幅度较大;对于热度感知架构 HA-RAID,各盘上的读写比例在每个时间点都差异不大,主要集中在 0.1～0.15,且波动幅度很小,很好地实现了阵列系统级的负载均衡.

由图 9 所示可知,工作负载为 *src*2_0 时,对于原始架构 RAID-0,各盘上的读写比例在每个时间点都有较大差异,且波动幅度较大;对于热度感知架构 HA-RAID,各盘上的读写比例在每个时间点都差异不大,主要集中在 0.1～0.15,且波动幅度很小,很好地实现了阵列系统级的负载均衡.

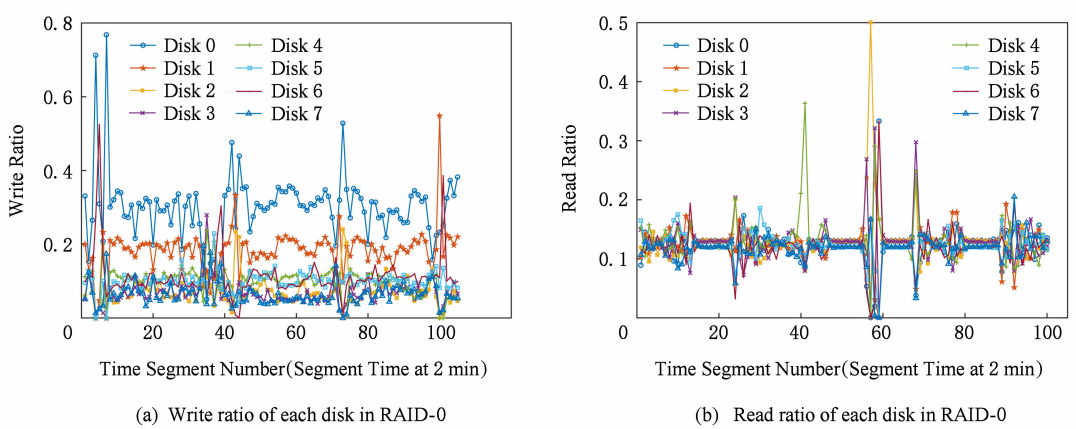


(a) Write ratio of each disk in RAID-0　(b) Read ratio of each disk in RAID-0

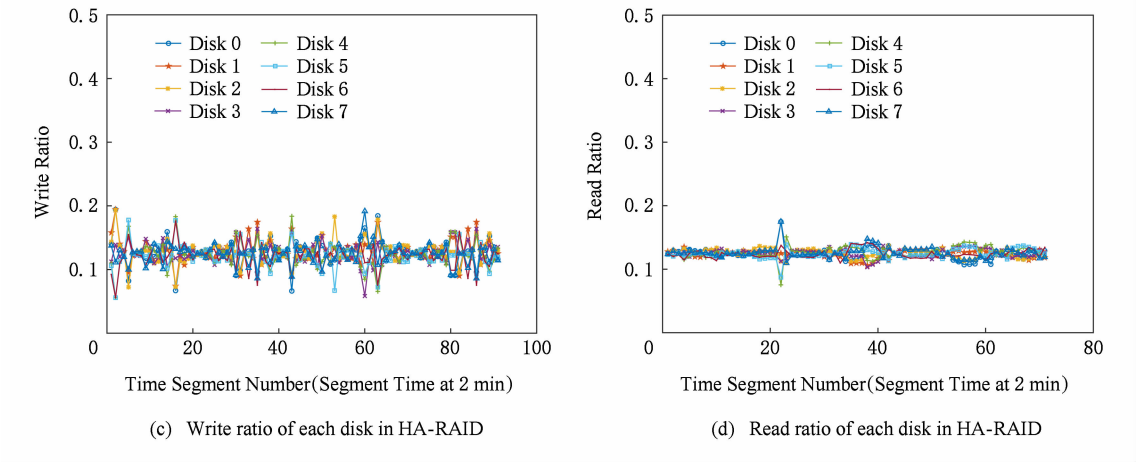
(c) Write ratio of each disk in HA-RAID　(d) Read ratio of each disk in HA-RAID

Fig. 9　Read/write ratio of each disk in RAID-0 and HA-RAID under *src*2_0

图 9　负载 *src*2_0 下 RAID-0 与 HA-RAID 阵列中各盘上的读写比例



(a) Write ratio of each disk in RAID-0　(b) Read ratio of each disk in RAID-0



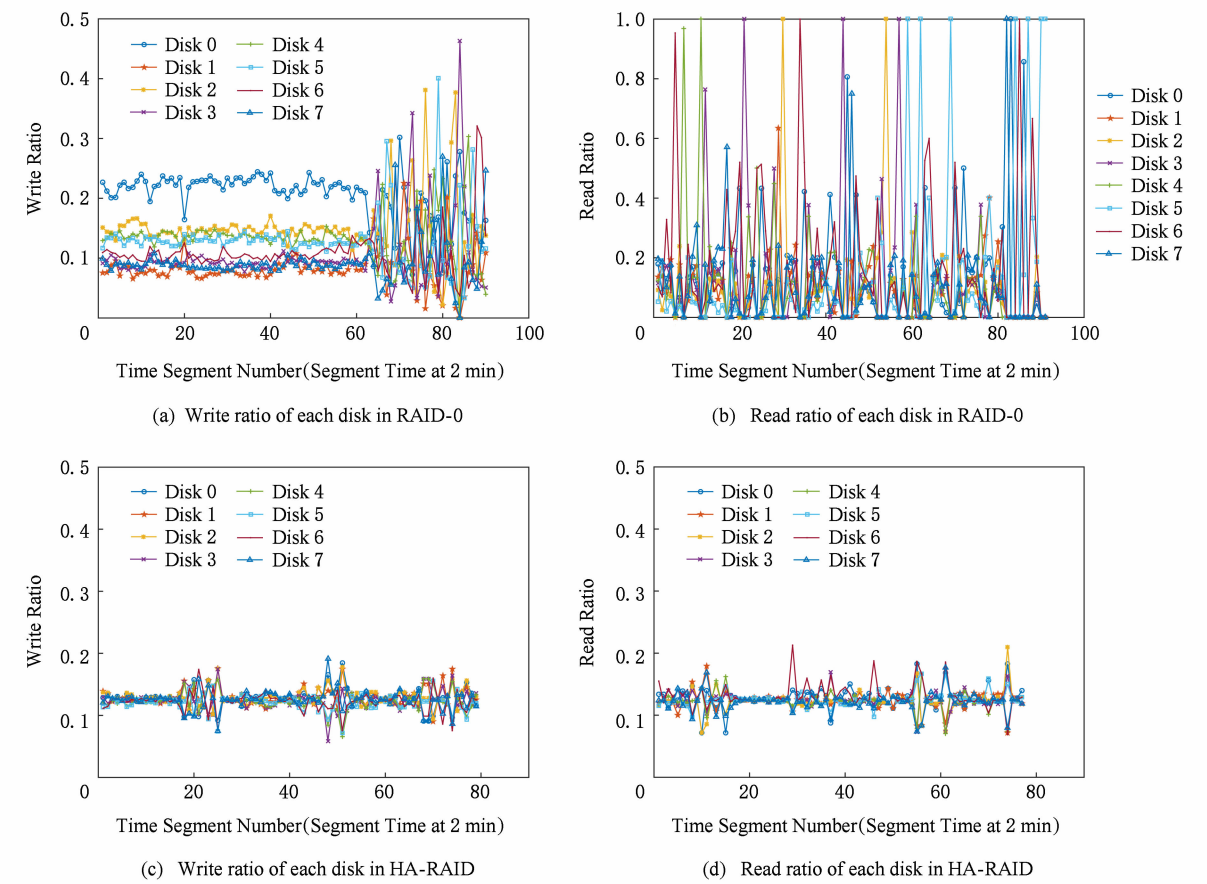
(c) Write ratio of each disk in HA-RAID (d) Read ratio of each disk in HA-RAID

Fig. 10 Read/write ratio of each disk in RAID-0 and HA-RAID under *usr*_0

图 10 负载 *usr*_0 下 RAID-0 与 HA-RAID 阵列中各盘上的读写比例

由图 10 所示可知,工作负载为 *usr*_0 时,对于原始架构 RAID-0,各盘上的读写比例在每个时间点都有较大差异,且波动幅度较大;对于热度感知架构 HA-RAID,各盘上的读写比例在每个时间点都差异不大,主要集中在 0.1～0.15,且波动幅度很小,很好地实现了阵列系统级的负载均衡.

由图 11 所示可知,工作负载为 *stg*_0 时,对于原始架构 RAID-0,各盘上的读写比例在每个时间点都有较大差异,且波动幅度较大;对于热度感知架构 HA-RAID,各盘上的读写比例在每个时间点都差异不大,主要集中在 0.1～0.15,且波动幅度很小,很好地实现了阵列系统级的负载均衡.



(a) Write ratio of each disk in RAID-0 (b) Read ratio of each disk in RAID-0

(c) Write ratio of each disk in HA-RAID (d) Read ratio of each disk in HA-RAID

Fig. 11 Read/write ratio of each disk in RAID-0 and HA-RAID under *stg*_0

图 11 负载 *stg*_0 下 RAID-0 与 HA-RAID 阵列中各盘上的读写比例

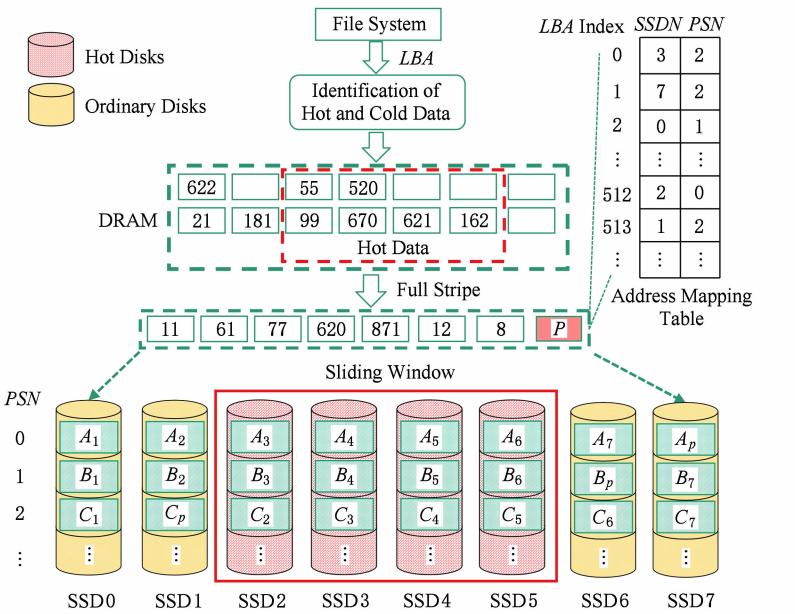


Fig. 12　HA-RAID architecture extended to RAID-5 array

图 12　扩展至 RAID-5 阵列上的 HA-RAID 系统架构

6　HA-RAID 扩展至 RAID-5/6

基于 RAID-0 设计的系统架构 HA-RAID 很好地实现了阵列系统级的负载均衡和磨损均衡,从而延长整个阵列的使用寿命并提升 I/O 性能. 但 RAID-0 阵列没有引入任何冗余数据,无法保证数据的可靠性. 而数据可靠性保证在实际应用中又非常重要,企业的核心数据一旦丢失,损失将是无法估量的. 因此,我们将基于 RAID-0 设计的 HA-RAID 系统架构扩展至 RAID-5 阵列,主要设计思路如图 12 所示.

对基于 RAID-0 的 HA-RAID 系统架构,将其扩展至 RAID-5 阵列,需要做 3 个方面的修改:

1) RAID-5 阵列需要对写入内存中的数据以条带形式组织,并使用与阵列盘上对应的滑动窗口将条带划分为热区域和冷区域,条带中的热区域存放识别出的热数据,冷区域存放普通数据.

2) 对于 *LBA* 读请求,根据地址映射表得到其物理地址(盘号和条带号),然后在底层盘上读数据.

3) 对于 *LBA* 写请求,在内存以条带的组织形式对其进行缓存:①如果该 *LBA* 被识别为热数据,则将其存储到条带的热区域(滑动窗口内);②如果该 *LBA* 被识别为冷数据,则将其存储到条带的冷区域(滑动窗口外). 当一个条带上数据存满,则计算产生该条带的校验信息 P,然后将条带数据和校验数据以条带形式追加写入阵列(校验数据以轮转的方式存储到普通盘上),并更新地址映射表中对应的物理地址.

RAID-6 阵列上的实现方法与 RAID-5 阵列类似,只是增加了计算开销和额外校验数据的存储,数据可靠性更高,这里不再赘述.

7　总　结

本文在 RAID-0 阵列中设计了一种基于冷热数据分离存储的固态硬盘阵列系统架构 HA-RAID,并结合滑动窗口技术进行优化. 相比原始 RAID-0 阵列架构,HA-RAID 可以将热数据相对均匀地存储到各个盘上并减少 12.01%～41.06%的平均响应时间,很好地实现了阵列系统级的负载和磨损均衡及阵列系统的 I/O 性能提升.

参　考　文　献

[1] Agrawal N, Prabhakaran V, Wobber T, et al. Design tradeoffs for SSD performance [C] //Proc of the USENIX'08 Annual Technical Conf. Berkeley, CA: USENIX Association, 2008: 57-70

[2] 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

[3] 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/IFIP Int Conf on Hardware/Software Codesign and System Synthesis. New York: ACM, 2010: 173-182

[4] Li Yongkun, Lee P, Lui J. Stochastic modeling of large-scale solid-state storage systems: Analysis, design tradeoffs and optimization [J]. ACM SIGMETRICS Performance Evaluation Review, 2013, 41(1): 179-190

[5] Li Yongkun, Lee P, Lui J, et al. Impact of data locality on garbage collection in SSDs: A general analytical study [C] // Proc of the 6th ACM/SPEC Int Conf on Performance Engineering. New York: ACM, 2015: 305-315

[6] Yang Yue, Zhu Jianwen. Analytical modeling of garbage collection algorithms in hotness aware flash-based solid-state drives [C] //Proc of the 30th IEEE Symp on Mass Storage Systems and Technologies(MSST). Piscataway, NJ: IEEE, 2014: 1-10

[7] Hsieh J W, Kuo Teiwei, Chang Lipin. Efficient identification of hot data for flash memory storage systems [J]. ACM Transactions on Storage, 2006, 2(1): 22-40

[8] Zhang Yufang, Yang Jihong, Xiong Zhongyang, et al. Method about identifying hot data in solid state memory based on queue count [J]. Journal of Application Research of Computers, 2011, 28(8): 2886-2888 (in Chinese)
(张玉芳, 阳佶宏, 熊忠阳, 等. 基于队列计数的固态存储器热数据识别方法[J]. 计算机应用研究, 2011, 28(8): 2886-2888)

[9] Zeng Lingfang, Feng Dan, Chen Jianxi, et al. HRAID6ML: A hybrid RAID6 storage architecture with mirrored logging [C] //Proc of the 28th IEEE Symp on Mass Storage Systems and Technologies (MSST). Piscataway, NJ: IEEE, 2012: 1-6

[10] Lu Youyou, Shu Jiwu, Wang Wei. ReconFS: A reconstructable file system on flash storage [C] //Proc of the 12th USENIX Conf on File and Storage Technologies (FAST). Berkeley, CA: USENIX Association, 2014: 75-88

[11] Kadav A, Balakrishnan M, Prabhakaran V, et al. Differential raid: Rethinking RAID for SSD reliability [J]. ACM SIGOPS Operating Systems Review, 2010, 44(1): 55-59

[12] Li Yongkun, Lee P, Lui J. Stochastic analysis on RAID reliability for solid-state drives [C] //Proc of the 32nd IEEE Int Symp on Reliable Distributed Systems (SRDS). Piscataway, NJ: IEEE, 2013: 71-80

[13] 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 (DSN). Piscataway, NJ: IEEE, 2013: 1-12

[14] Park D, Du D. Hot data identification for flash-based storage systems using multiple Bloom filters [C] //Proc of the 27th IEEE Symp on Mass Storage Systems and Technologies (MSST). Piscataway, NJ: IEEE, 2011: 1-11

[15] Jung S, Lee Y, Song Y. A process-aware hot/cold identification scheme for flash memory storage systems [J]. IEEE Transactions on Consumer Electronics, 2010, 56(2): 339-347

[16] Shen Biaobiao, Li Yongkun, Xu Yinlong, et al. A light-weight hot data identification scheme via grouping-based LRU lists [G] //LNCS 9531: Proc of the 15th Int Conf on Algorithms and Architectures for Parallel Processing. Berlin: Springer, 2015: 88-103

[17] Park D. Hot and cold data identification: Applications to storage devices and systems [D]. Minnesota: Computer Science, University of Minnesota, 2012

[18] Park D, Nam Y J, Debnath B, et al. An on-line hot data identification for flash-based storage using sampling mechanism [J]. ACM SIGAPP Applied Computing Review, 2013, 13(1): 51-64

[19] SNIA. IOTTA Repository [OL]. [2016-12-06]. http://iotta.snia.org/

Zhang Qiang, born in 1994. MSc in computer software and theory from USTC. His main research interests include RAID and SSD technique.

Liang Jie, born in 1988. PhD. His main research interests include storage systems, especially NAND flash SSDs and PCM based storage systems, distributed file systems.

Xu Yinlong, born in 1963. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include storage systems, data processing, and high performance computing.

Li Yongkun, born in 1986. PhD, associate professor. His main research interests include storage systems, especially NAND flash SSDs and PCM based storage systems, distributed storage systems, and storage support for graph analysis and big data analysis.