一种能耗优化的RAID小写预读方法

孙志卓[1],黄 洪[2],张全新[3],谭毓安[3],李元章[3],张小松[4]
(1.德州学院计算机系,山东德州 253000;2.中国人民解放军61516部队,北京 100094;
3.北京理工大学计算机学院,北京 100081;4.唐山学院计算机科学与技术系,河北唐山 063000)

摘 要: 视频监控、备份、归档等应用产生海量存储数据,导致存储能耗急剧增加.S-RAID采用局部并行数据布局,可显著降低该类应用的存储能耗.为使更多磁盘待机节能,S-RAID通常执行"小写"操作,写操作时会额外引入等量的读操作,会显著降低性能.现有预读机制主要发生在文件级,无法感知RAID级小写引发的读旧数据、旧校验数据等读操作,因此不会也无法预读该类数据.为此,提出一种面向S-RAID的RAID级小写预读算法,由小写操作触发并在RAID级执行预读,根据S-RAID的数据布局方式,大粒度异步预读小写需要的旧数据、旧校验数据,有效减少I/O数和寻道数,提高磁盘的利用率.该方法可显著提高S-RAID的写性能,并且不依赖于任何额外硬件,具有更高的可用性.

关键词: 海量存储;节能;局部并行;盘阵;数据布局
中图分类号: TP333 **文献标识码:** A **文章编号:** 0372-2112 (2019)09-1987-07
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2019.09.024

A Small Write Prefetching Scheme for Engery-Optimized RAID Sub-systems

SUN Zhi-zhuo[1],HUANG Hong[2],ZHANG Quan-xin[3],TAN Yu-an[3],LI Yuan-zhang[3],ZHANG Xiao-song[4]
(1. *College of Computer Science and Technology,Dezhou University,Dezhou,Shandong* 253000,*China*;
2. *Chinese People's Liberation Army Unit* 61516,*Beijing* 100094,*China*;
3. *School of Computer Science,Beijing Institute of Technology,Beijing* 100081,*China*;
4. *Department of Computer Science and Technology,Tangshan University,Tangshan,Hebei* 063000,*China*)

Abstract: The applications,such as video surveillance,backup,and archiving,generate massive storage data,which make the energy consumption of storage devices increase rapidly. S-RAID can reduce the storage energy consumption of above applications significantly. In order to make more disks standby for energy saving,S-RAID prefers to small write. However,performing small write requires additional and equal amount of read operations for generating parity data,and the write performance of S-RAID is decreased. The existing prefetching mechanism mainly implements at the file system level,and cannot sense the read operations generated by small write at RAID level,such as reading old data and reading old parity. So it can not prefetch these old data and old parity data. Therefore,a RAID-level prefetching algorithm for small write of S-RAID is proposed,which is triggered by small write operations and performs at RAID level. It asynchronously prefetches the old data and old parity data required by small writes in large granularity according to the data layout of S-RAID. The write performance of S-RAID can be improved by about 40% without any additional hardware cost.

Key words: massive storage;energy conservation;partial parallelism;RAID;data layout

1 引言

视频监控、备份、归档等应用产生海量存储数据,导致存储能耗急剧增加[1~3].S-RAID采用局部并行数据布局,通过局部并行提供合适的性能,可避免全局并行带来的性能过剩,显著降低该类应用的存储能耗.在典型的视频监控存储实验中,可比传统RAID节能70%左右[1].

为保证磁盘待机节能,S-RAID主动采用"小写"

收稿日期:2018-08-30 修回日期:2018-12-31;责任编辑:梅志强
基金项目:国家自然科学基金(No.61370063);山东省自然科学基金(No.ZR2016FM37,No.ZR2017LC014,No.ZR2015FL006)

(small write)工作模式:写新数据时需要读取对应的旧数据、旧校验数据,与新数据一起计算新校验数据.小写额外引入了(与写数据)等量的读操作,降低了 S-RAID 的写性能和节能效率.与 RAID 5 等通用磁盘阵列相比,S-RAID 的小写问题独特,原因如下:S-RAID 的节能策略是把 I/O 请求集中在部分并行工作的磁盘上,以调度其它磁盘待机节能.重构写(reconstruction write)、完全写(full write)需要启动所有磁盘,使 S-RAID 丧失节能性,因此 S-RAID 是主动执行小写的.已有的面向通用 RAID 的小写优化方法,难以解决其小写问题.

操作系统具有数据缓存机制,数据缓存期间,磁盘一般会有较多空闲时间,可充分利用该时间进行有效预读.然而,现有预读机制主要在文件级执行,无法感知 RAID 级由小写引发的额外读操作,因此不会预读该类数据.为此,提出一种面向 S-RAID 的 RAID 级小写预读算法,由小写操作触发预读,根据 S-RAID 的数据布局方式,大粒度异步预读小写需要的旧数据、旧校验数据,有效减少 I/O 数,提高磁盘的利用率.小写预读优化后 S-RAID 的写性能可提高 40% 左右,并且不依赖于任何额外硬件,具有更高的便捷性和经济性,非常适合大规模部署.

本文的主要贡献如下:

(1)研究发现已有的文件级预读机制,无法预读 RAID 级小写操作需要的旧数据、旧校验数据,研究 RAID 级小写预读机制,对于以小写为主的 S-RAID,具有更加重要的意义;

(2)提出一种面向 S-RAID 的 RAID 级小写预读算法,由小写操作触发并在 RAID 级执行预读,大粒度异步预读小写需要的旧数据、旧校验数据,有效减少 I/O 数和寻道数,提高磁盘的利用率;

(3)对所提出的面向 S-RAID 的 RAID 级小写预读方法进行了较全面的实验测试,结果表明小写预读可显著提高 S-RAID 的写性能.

2 相关工作

视频监控、备份、归档是最大的大数据[4,5].仅以视频监控为例,全球每年产生的数据中约有 50% 是监控视频,未来该比例还会进一步提升.该类应用存储数据的快速增长,导致存储能耗急剧增加.很多研究工作关注存储能耗问题,提出了一些相应的解决方案[6,7].

该类应用需要海量存储空间,以顺序访问为主,对随机性能要求不高.针对上述存储特性,文献[8]提出了节能磁盘阵列 S-RAID.S-RAID 采用局部并行数据布局,局部并行既有利于调度部分磁盘运行而其余磁盘待机节能,又能够提供合适的性能.在典型视频监控存储实验中,S-RAID 的节能效果显著优于 PARAID[9]、Hibernator[10]、MAID[11]、eRAID[12] 等典型存储节能方法,非常适用于该类存储应用.

为使更多磁盘待机节能,S-RAID 主动执行小写操作.小写有利于 S-RAID 节能,但会额外引入读操作,降低 S-RAID 的写性能和节能效率[13,14].视频监控、备份、归档等应用以写操作为主,这使 S-RAID 的小写问题异常突出,因此对 S-RAID 其进行性能优化具有重要意义.已有的小写优化方法,如 Parity Logging[15]、RAID6L[16]、WOSS[17]等,主要面向通用磁盘阵列,难以有效解决 S-RAID 的小写问题.

李明强、舒继武提出的 DACO 磁盘架构[18],包括读、写、复合 3 种操作,复合操作采用流水技术实现块级数据的读改写,能够有效解决 S-RAID 的小写问题.DACO 仍处于研究阶段,或许视频监控的巨大存储需求,能进一步推动 DACO 的产业化进程.刘靖宇等[13]针对归档系统具有一次写入、极少修改的存储特性,基于 NILFS 日志文件系统(log files system)优化了 S-RAID 的写性能.该方法不适合需要多次写入、多次删除的视频监控存储系统.

Ripple-RAID 通过异地更新、渐进生成校验数据等方法优化了局部并行中的小写操作[14],需要引入额外的 SSD.通过增加日志盘,或者低功耗的 NVRAM、SSD 进行小写优化,会增加硬件成本,降低 S-RAID 大规模部署时的灵活性和经济性.研究不依赖于额外硬件的小写优化方法,对提高 S-RAID 的可用性具有重要意义.

本文将研究基于大粒度异步预读的 S-RAID 优化方法,不依赖于额外硬件,在提高 S-RAID 性能与节能效率的同时,保持 S-RAID 的灵活性和经济性.视频监控等应用需要海量存储空间,磁盘是首选存储设备,充分发挥磁盘性能优势、提高磁盘利用率是本文优化 S-RAID 的主要目标.

首先,增大 I/O 尺寸可有效提高磁盘利用率[19].磁盘执行一次 I/O 访问需要 T_a 和 T_d 两个时间,T_a 包括寻道时间及磁道内定位时间,T_d 为数据传输时间,是磁盘的有效利用时间,T_d 所占比重越大磁盘利用率越高.增大 I/O 尺寸可有效提高 T_d 比重,显著提高磁盘利用率,如图 1 所示.

第二,操作系统通常具有完善的缓存机制,数据首先在内存中缓存,直到缓存数据足够多后,再写回外部磁盘,目前缓存容量可达 10 至 100GB 量级.视频监控等应用以写操作为主,在写缓存期间,S-RAID 通常会有足够的空闲时间,可利用该时间异步预读小写操作需要的旧数据、旧校验数据,对上层应用隐藏由小写触发的读操作造成的磁盘 I/O 延迟.

第三,S-RAID 主要面向视频监控、备份、归档等应

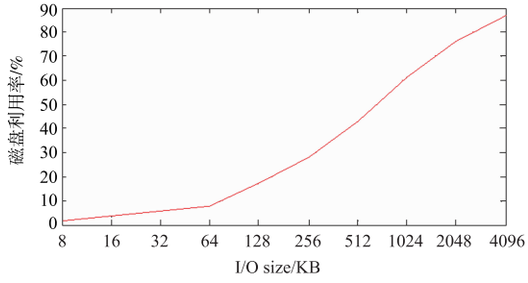

图1　大尺寸I/O可有效提高磁盘利用率[19]

用,与在线事务处理(OLTP)、搜索引擎等以随机访问为主的应用不同,该类应用以顺序访问为主,对于正常读操作、以及由小写触发的额外读操作,都适合进行大粒度异步预读,有效提高磁盘利用率.

根据数据访问的局部性原理,将当前读操作的相邻数据预读到内存,可避免应用程序由于数据缺失而被阻塞挂起,预读是现代操作系统的基本功能. 吴峰光等[19]提出了 Linux 按需预读算法,算法分为监控与执行两部分,其中监控部分简化了预读触发条件,执行部分则由独立的判决模块构成. 该预读算法简洁高效,已被 Linux 内核采用.

然而,操作系统预读发生在文件级,无法预读 S-RAID 中小写操作需要的旧数据、旧校验数据. S-RAID 的校验机制在文件系统下层的 RAID 级实现,对上层文件系统透明. 文件系统不能感知 RAID 级的额外读操作(读旧数据、旧校验数据),缺乏 RAID 级校验数据的组织信息,不会且无法预读该类数据. RAID 5 等通用磁盘阵列(或固态盘阵列)也不会预读该类数据,只是对旧数据、旧校验数据的预读需求没有 S-RAID 强烈,所以并未引起足够重视.

综上,本文提出一种面向 S-RAID 的 RAID 级小写预读算法,由小写操作触发并在 RAID 级执行,根据 S-RAID 的数据布局方式,大粒度异步预读小写需要的旧数据、旧校验数据,有效减少 I/O 数和寻道数,充分发挥磁盘的性能优势,提高磁盘利用率.

3　面向 S-RAID 的小写预读

S-RAID 的读操作分为两类:一类是应用程序发来的读操作,该类读操作在文件级有完善的预读机制,称为基本预读,本文不再讨论;另一类是 S-RAID 进行小写时,由小写触发的 RAID 级额外读操作,该类读操作对文件级透明,文件级预读机制不会执行该类预读,本文研究该类预读,记做小写预读.

3.1　小写预读的数据结构

基本预读由读操作触发,小写预读则由写操作触发. 小写预读需要维护两类状态变量:写记录和预读记录. 写记录用于识别随机写或是顺序写;预读记录是本次预读的决策结果,下次预读的决策依据.

为记录写记录和预读记录,小写预读算法采用图 2 所示的数据结构. 其中 start、count 构成一个预读窗口(readahead window),记录一次预读的起始位置和预读长度;flag 为预读标记块;weight 表示该数据结构的权重,记录预读窗口向前推进的次数,与写操作的顺序性紧密相关,其值越大,表明写操作的顺序性越强;time 记录该数据结构的建立时间.

```
struct block_ra_state {
    blkoff_t    start;      //预读起始块号
    int         count;      //预读块的数量
    blkoff_t    flag;       //预读标记块，上次预读中设置的，若当
                            //前写数据中包含该块，则开始预读
    int         weight      //预读窗口向前推进的次数
    int         time        //该数据结构建立的时间
};
```

图2　小写预读的数据结构

如果为每个“疑似顺序写”分配一个预读数据结构,由于每个写操作(包括随机写)都是“疑似顺序写”,会造成大量开销. 需要控制该数据结构的数量. 当其数量超过系统的规定值时,便撤销那些创建时间早、权重小的预读数据结构.

3.2　小写预读算法

为提高预读算法的鲁棒性和适应性,小写预读算法分为监控和执行两部分:监控部分检查 RAID 级写请求是否满足预读触发条件,如果满足则触发小写预读;执行部分根据不同的触发条件,进行初始预读或顺序预读,顺序预读进一步分为后继预读(1 个顺序写中夹杂若干随机写)、交织预读(多个顺序写交织)2 种情况.

3.3　小写预读触发条件

小写预读的预读触发条件是:①写数据块缓存缺失并且未在预读中,S-RAID 执行小写时,无论写部分块还是写全块,都要执行磁盘 I/O 加载该块(如果不是小写,写全块时不需加载该块),在加载该数据块的同时可进行初始预读,预读后续相邻的若干连续数据块;②预读标记块,该标记块是上一次预读中设置的. 假如当前写数据具有预读标记,说明进行下次预读的时机已到来,应启动预读程序进行异步预读. 为了避免重复触发预读,触发预读时应该清除本次预读触发标记.

Linux 采用的按需预读算法[19]面向各类应用,当发生页面缺失时,要进行严格的顺序性匹配,只有两次读操作是顺序读时才进行初始预读. 本文提出的小写预读算法做了针对性优化,只要写数据缓存缺失,且该数据没有在预读中,就启动初始预读,主要原因如下:①根据 S-RAID 的小写特性,此时必须加载该数据,而对于

磁盘来讲,加载 1 个数据块与加载几个连续数据块的代价差别很小,因此可以进行初始预读;②视频监控等应用中的写操作,以顺序访问为主,缺失数据块的相邻数据块一般马上被写到,因此有必要进行预读. 小写预读不检验顺序性,因此不需要记录最后一次写操作的位置.

预读标记用于识别顺序写和随机写,可保证一个顺序写的预读开始后,该预读序列不会被随机写或交织的顺序写打断. 为防止因缓存命中导致实际预读长度减小,预读中如果出现缓存命中情况,则跳过命中的数据块继续向前预读.

小写预读算法只对数据块作预读标记,根据标记数据块以及 S-RAID 的数据布局,可确定校验块所在磁盘及磁盘内的偏移量,并对其进行预读.

3.4 小写预读窗口设定

在视频监控、备份、归档等存储应用中,可以执行更加贪婪的预读策略(实验证明是可行的),可按以下规则设定预读窗口大小.

(1)初始预读时,预读窗口的初始大小可根据写请求大小设定:

$$size = write\_size \times scale$$

其中,scale 可以取 4 或 8.

(2)后继预读中逐次倍增预读窗口:

$$size = prev\_size \times 2$$

(3)限制最大预读量为 max_readahead:

$$size = \min(size, max\_readahead)$$

max_readahead 的取值应根据存储应用的基本写性能设定. 如果一个存储系统的写速率为 20MB/s,则 max_readahead 可取 40MB 至 200MB(2 至 10 倍写速率)即可.

3.5 小写预读示例

本节通过具体示例,说明小写预读算法在顺序写、以及随机写与多个顺序写交织时的执行过程,检测小写预读算法在复杂条件下的预读效果.

3.5.1 顺序写

以 3 个顺序写为例,说明小写预读在顺序写时的基本执行过程,如图 3 所示.

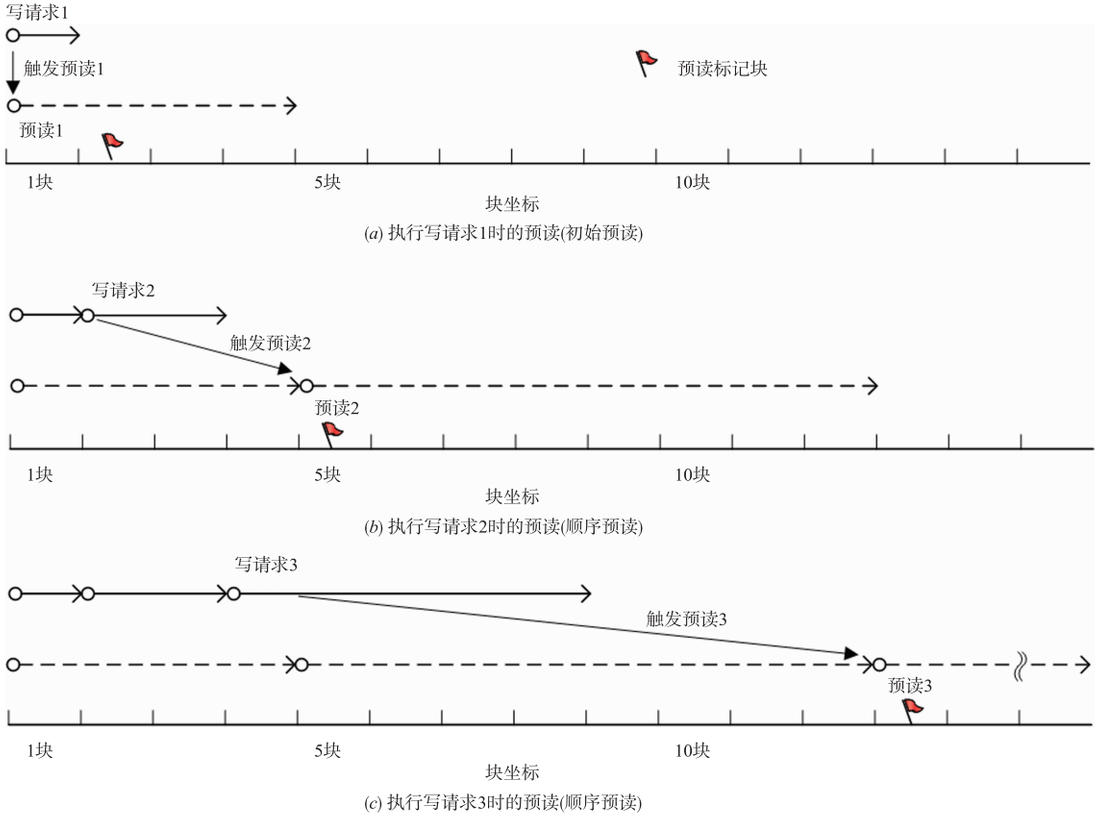

图3 顺序写时的小写预读

(1)如图 3(a)所示,执行写请求 1 时,假设写数据(第 1 块)没有被缓存,则需要从磁盘加载. 此时满足预读触发条件,因此触发预读 1(初始预读),从第 1 块开始,读取 4 个连续块(设写请求为 1 个块),并标记第 2 块为预读标记块;

(2)如图 3(b)所示,执行写请求 2 时,因为写数据块(第 2 块)为预读标记块,此时触发预读 2(顺序预读),从第 5 块开始,预读 4×2 个连续块,并标记第 5 块

为预读标记块;

(3)如图 3(*c*)所示,执行写请求 3 时,当写到第 5 块,由于第 5 块为预读标记块,会触发预读 3(顺序预读),从第 13 块开始,预读 4×2×2 个块,并标记第 13 块为预读标记块.

3.5.2　随机写与并发顺序写

视频监控、备份、归档等存储应用以顺序写为主,但也具有一定的复杂性[20]. 例如文件系统的元数据通常会与基本数据分离存储,基本顺序写中会掺杂少量随机写(如元数据更新);Ext4 等文件系统的日志区与数据区相分离[3],因此基本写操作与日志区写操作交织进行. 小写预读算法应具有较强鲁棒性,应该能够稳定、准确识别被随机写中断的顺序写及交织的顺序写.

下面通过写一组数据,验证小写预读算法的鲁棒性. 写数据块序列按时间顺序如下:{36,13,47,37,14,65,38,15,25,39,16,58,40,17,73,41,18,3,42,19,64,43,20,77,44,21,6,45,21,69,46,23,52},假设初始时各块均未缓存. 小写预读过程如下:

(1)写块 36 时,满足初始预读条件,读取块 36 ~ 39,标记块 37 为预读标记块;

(2)写 13 块时,满足初始预读条件,读取块 13 ~ 16,标记块 14 为预读标记块;

(3)写 47 块时,满足初始预读条件,读取块 47 ~ 50,标记块 48 为预读标记块;

(4)写块 37 时,命中预读标记,启动顺序预读,预读 40 ~ 47 块,标记块 40 为预读标记块;

(5)写块 14 时,命中预读标记,启动顺序预读,预读 17 ~ 24 块,标记块 17 为预读标记块;

(6)写块 65 时,满足初始预读条件,读取块 65 ~ 68,标记块 66 为预读标记块.

依此类推,小写预读算法能够成功分离出顺序序列{36,37,38,…,46}、{13,14,15,…,23},离散序列{47,65,25,…,52},如图 4 所示.

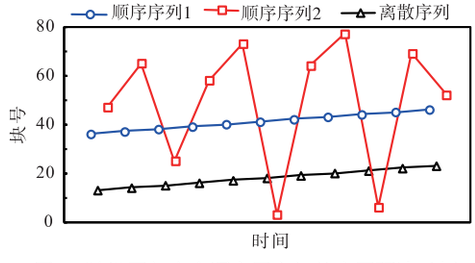

图4　随机写与多个顺序写交织的小写预读示例

通过以上示例可得,无论序列如何被中断,只要是顺序写序列,前进中一定会命中其预读标记块,从而触发新的预读,设置新的预读标记块,周而复始. 小写预读算法具有很强的鲁棒性,能够在较复杂环境下稳定、可靠地执行小写预读.

在该示例中,顺序写对应的预读窗口在向前推进中会逐渐增大,而随机写对应的预读窗口没有机会向前推进,因此也不会增大,随着时间的推移将逐渐老化,小写预读算法会根据预读数据结构中的 *weight*、*time* 参数进行回收.

4　实验结果

为验证上述小写预读优化方法,在 RAID 级对 S-RAID 进行了小写预读优化:预读窗口初始大小为 4MB,最大预取量为 128MB;"疑似写顺序流"的数量限定为 30,超过该值时回收创建时间早、权值小的预读数据结构. 存储服务器 CPU 为 Intel(R) Core(TM),内存 8GB,S-RAID 由 10 块希捷 ST3000DM001 磁盘构成,其 Strip(也称 Chunk)大小为 64 KB.

S-RAID 面向视频监控、备份、归档等存储应用,以顺序写操作为主. 当采用 NILFS 日志文件系统后,除包含少量读操作外,会完全执行顺序写. 因此利用 Linux 的 dd 命令,生成了一个包含 5% 随机读,95% 顺序写的测试负载,测试 S-RAID 在基本工作模式下的写性能. 小写预读优化后,S-RAID 的写性能可提高 40% 左右,如图 5 所示.

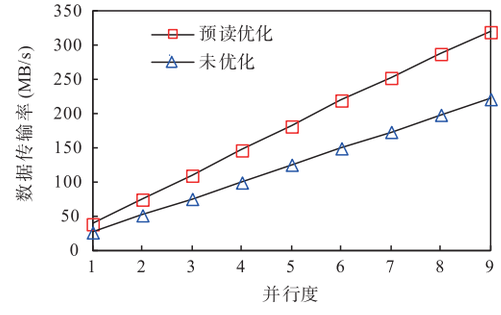

图5　少量随机读干扰下的写性能

为进一步验证小写预读优化方法,利用 dd 命令生成了 2 个写顺序流 *A* 和 *B*,各包含 1000 个 512KB 大小的顺序写请求,分别记作 A_1、A_2、…、A_{1000} 和 B_1、B_2、…、B_{1000}. 将 *A* 与 *B* 随机交织,但保持 *A*、*B* 内部各自的顺序性,形成 A_1、…、B_1、…,或 B_1、…、A_1、…形式的交织写序列,然后写入 S-RAID.

在上述交织写顺序流下 S-RAID 的写性能,优化后可比原来提高 50% 以上,如图 6 所示. 进一步比较图 5、图 6,可以得出随着交织写顺序流的增加,小写预读的优化效果会更加显著.

现代磁盘具有内置预读功能,上述实验是在磁盘内置预读功能打开情况下进行的. 磁盘内置预读功能与文件级预读,都缺乏 RAID 级数据布局的相关知识,无法感知 S-RAID 中的磁盘状态切换,其预读具有较大的盲目性. 此外,由于磁盘内处理器能力和缓存上午限制,磁盘内置预

读效果会随着读、写交织流的增加而显著下降[19].

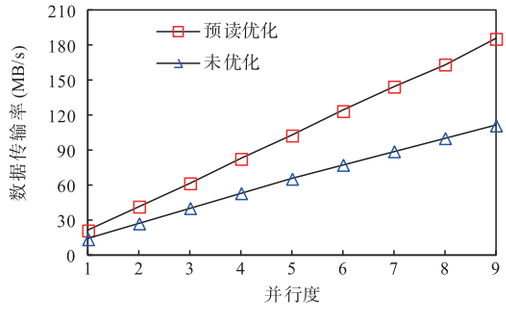

图6 2个写顺序流交织下的写性能

小写预读与基本预读具有较大差异,体现在预读目的、预读对象、预读执行层次、预读触发条件、预读可行性等方面,具体见表1.

表1 小写预读与基本预读比较

	小写预读	基本预读
预读目的	优化写性能	优化读性能
预读对象	小写中参与生成新校验数据的旧数据、旧校验数据	与当前读数据相邻的数据
执行层次	RAID级	文件系统级
触发条件	写操作	读操作
可行性	S-RAID面向以顺序写为主的存储应用,因此预读小写需要的旧数据、旧校验数据是可行的	数据访问具有空间局部性特征,因此预读是可行的
优化的基本依据	充分利用磁盘空闲时间	
	有效减少读操作次数,增大读操作尺寸	
	避免频繁的读、写操作切换;避免写操作因缺少旧数据、旧校验数据而等待	防止程序因读数据缺失而被阻塞挂起

5 总结

针对S-RAID存在的小写问题,提出了一种基于小写预读的性能优化方法,通过大粒度异步预读小写需要的旧数据、旧校验数据,有效减少读操作次数及等待时间,提高磁盘利用率.小写预读主要利用S-RAID中磁盘的空闲时进行,可显著提高S-RAID的写性能,而对读性能的影响可以忽略,.

该方法同样适用于广泛部署的RAID 5、RAID 6等通用磁盘阵列(或固态盘阵列).由于RAID 5、RAID 6面对的负载类型复杂多样,为了获得更佳的小写性能优化效果,还需要进一步深入研究,对该方法进行针对性的改进与提高.

参考文献

[1] Sun zhizhuo, Li Yuanzhang, Tan Yu-an. An energy-efficient storage for video surveillance[J]. Multimedia Tools and Applications, 2014, 73(1): 151 – 167.

[2] 王金铭,叶时平,徐振宇,陈超祥,蒋燕君. 半张量积低存储压缩感知方法研究[J],电子学报,2018,46(4):797 – 804.
Wang Jinming, Ye Shiping, Xu Zhenyu, Chen Chaoxiang, Jiang Yanjun. Low storage space of random measurement matrixfor compressed sensing with semi-tensor product[J]. Acta Electronica Sinica, 2018, 46(4): 797 – 804. (in Chinese)

[3] Xiao Yu, Yu-an Tan, Changyou Zhang, Chen Liang, AOURRA Khaled, Jun Zheng, Quanxin Zhang. Ahigh-performance hierarchical snapshot scheme for hybrid storage systems[J]. Chinese Journal of Electronics, 2018, 27(1): 76 – 85.

[4] Sun Zhizhuo, Zhang Quanxin, Li Yuanzhang, Tan Yu-an. DPPDL: a dynamic partial-parallel data layout for green video surveillance storage[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2018, 28(1): 193 – 205.

[5] Zhang Guangyan, Huang Zican, Ma Xiaosong, et al. RAID +: deterministic and balanced data distribution for large disk enclosures[A]. Proc of the 16th USENIX Conference on File and Storage Technologies (FAST)[C]. Oakland: USENIX Association, 2018. 279 – 294.

[6] Yan Fang, Tan Yu-an, Zhang Quanxin, Wu Fei, Cheng Zijing, Zheng Jun. An effective RAID data layout for object-based de-duplication backup system[J]. Chinese Journal of Electronics, 2016, 25(5): 832 – 840.

[7] 李元章,孙志卓,马忠梅,郑军,谭毓安. S-RAID 5:一种适用于顺序数据访问的节能磁盘阵列[J]. 计算机学报,2013,36(6):1290 – 1302.
Li Yuanzhang, Sun Zhizhuo, Ma Zhongmei, Zheng Jun, Tan Yu-an. S-RAID 5: An energy-saving raid for sequential access based application[J]. Chinese Journal of Computers, 2013, 36(6): 1290 – 1302. (in Chinese)

[8] Li X, Tan YA, Sun ZZ. Semi-RAID: a reliable energy-aware RAID data layout for sequential data access[A]. 27th Symposium on Mass Storage Systems and Technologies (MSST)[C]. Piscataway: IEEE, 2011. 1 – 11

[9] Weddle C, Oldham M, Qian J, et al. PARAID: a gear-shifting power-aware RAID[A]. 5th USENIX Conf on File and Storage Technologies (FAST)[C]. Berkeley: USENIX Association, 2007. 245 – 260.

[10] Zhu Q B, Chen Z F, Tan L, et al. Hibernator: helping disk arrays sleep through the winter[J]. Operating Systems Review, 2005, 39(5): 177 – 190.

[11] Colarelli D, Grunwald D. Massive arrays of idle disks for storage archives[A]. ACM/IEEE Conference on Supercomputing[C]. Los Alamitos: IEEE, 2002. 1 – 11.

[12] Wang J, Zhu H, Li D. eRAID: conserving energy in conventional disk-based RAID system[J]. IEEE Transactions

on Computers,2008,57(4):359 – 374.

[13] 刘靖宇,谭毓安,薛静锋,马忠梅,李元章,张全新. S-RAID 中基于连续数据特征的写优化策略[J]. 计算机学报,2014,37(3):722 – 734.
Liu Jingyu, Tan Yu-an, Xue Jingfeng, Ma Zhongmei, Li Yuanzhang, Zhang Quanxin. Write optimization strategy in s-raid based on sequential data characteristics[J]. Chinese Journal of Computers,2014,37(3):722 – 734. (in Chinese)

[14] 孙志卓,张全新,谭毓安,李元章. Ripple-RAID:一种面向连续数据存储的高效能盘阵[J]. 软件学报,2015,26(7):1824 – 1839.
Sun Zhizhuo, Zhang Quanxin, Tan Yu-an, Li Yuanzhang. Ripple-RAID: A high-performance and energy-efficient RAID for continuous data storage[J]. Journal of Software,2015,26(7):1824 – 1839. (in Chinese)

[15] Stodolsky D, Holland M, Courtright W V, et al. Parity-logging disk arrays[J]. ACM Transactions on Computer Systems,1994,12(3):206 – 235.

[16] Jin Chao, Feng Dan, Jiang Hong, et al. RAID6L: a log-assisted RAID6 storage architecture with improved write performance[A]. 27th Symptom on Mass Storage Systems and Technologies (MSST)[C]. Piscataway: IEEE, 2011. 1 – 6.

[17] 孙志卓,张全新,李元章,谭毓安,刘靖宇,马忠梅. 连续数据存储中面向 RAID 5 的写操作优化设计[J]. 计算机研究与发展,2013,50(8):1604 – 1612.
Sun Zhizhuo, Zhang Quanxin, Li Yuanzhang, Tan Yu-an, Liu Jingyu, Ma Zhongmei. Write optimization for RAID5 in sequential data storage[J]. Journal of Computer Research and Development,2013,50(8):1604 – 1612. (in Chinese)

[18] Li Mingqiang, Shu Jiwu. DACO: a high-performance disk architecture designed specially for large-scale erasure-coded storage systems[J]. IEEE Transactions on Computers,2010,59(10):1350 – 1362.

[19] 吴峰光,奚宏生,徐陈锋. 一种支持并发访问流的文件预取算法[J]. 软件学报,2010,21(8):1820 – 1833.
Wu Fengguang, Xi Hongsheng, Xu Chenfeng. File prefetching algorithm for concurrent streams[J]. Journal of Software,2010,21(8):1820 – 1833. (in Chinese)

[20] Yu Xiao, Zhang Changyou, Xue Yuan, et al. An extra-parity energy saving data layout for video surveillance[J]. Multimedia Tools and Applications,2018,77(4):4563 – 4583.

作者简介

孙志卓　男,1973 年出生,吉林伊通人,2013 年于北京理工大学获工学博士学位,现为德州学院计算机系副教授,研究方向为计算机体系结构与大规模数据存储.
E-mail:sunzhizhuo@126.com

黄　洪　男,1988 年出生,河南信阳人,2009 年于解放军国防科技大学获军事情报学士学位,现为战略支援部队信息通信基地软件工程师,研究方向为软件与大数据.
E-mail:huanghongbj@163.com

张全新　男,1974 年出生,山东德州人,2003 年于北京理工大学获工学博士学位,现为北京理工大学计算机学院讲师,研究方向为机器学习、移动计算.
E-mail:zhangqx@bit.edu.cn

谭毓安　男,1972 年出生,重庆巫溪人,北京理工大学计算机学院教授,博士生导师,北京理工大学计算机实验教学中心主任,中国计算机学会信息存储技术专委会委员,研究方向为信息安全、网络存储、嵌入式系统.
E-mail:victortan@yeah.net

李元章　男,1978 年出生,江苏滨海人,2015 年于北京理工大学获工学博士学位,现为北京理工大学计算机学院讲师,研究方向为信息安全、嵌入式系统.
E-mail:popular@bit.edu.cn

张小松(通信作者)　男,1979 年出生,河北唐山人,2018 年于北京理工大学获工学博士学位,现为唐山学院计算机科学与技术系副教授,研究方向为网络与信息安全.
E-mail:zxs0224@163.com