

## RAID 控制器的 Cache 管理策略研究

陈云霄,王丽芳,蒋泽军

(西北工业大学 计算机学院, 陕西 西安 710129)

**摘 要:** 为进一步缩小外存与 CPU 间的速度差异,满足日益增高的 I/O 请求率,在磁盘阵列设计中引入 Cache,实现一种适用于 RAID 控制器的 Cache 管理策略。此方法重点研究了 Cache 的组织与管理,利用优化的最近最少使用算法(LRU)提高 Cache 命中率,通过树形结构的转换减少磁盘 I/O 次数,以提高系统的整体性能。从该策略在 RAID 控制器原型的实验结果分析,证明该策略对减少写磁盘 I/O 确有显著效果。

**关键词:** Cache;LRU 算法优化;命中率;I/O 写延迟

**中图分类号:** TP391.9

**文献标识码:** A

**文章编号:** 1000—7180(2014)07—0172—05

## The Cache Management Strategy Research on RAID Controller System

CHEN Yun-xiao, WANG Li-fang, JIANG Ze-jun

(School of Computer, Northwestern Polytechnical University, Xi'an 710129, China)

**Abstract:** To reduce further the speed difference between the external memory and the CPU, and to meet more and more I/O request rate, Cache is introduced in the disk array. The strategy focuses on the organization and management of Cache. Taking use of the Least Recently Used algorithm optimization (LRU) improves Cache hit rate and taking use of the conversion between tree structures reduce disk I/O times in order to improve the overall system performance. The experimental results prove that the strategy has really a significant effect to reduce disk write I/O.

**Key words:** Cache; LRU algorithm optimization; hit rate; I/O write delay

### 1 引言

RAID 技术在近二十年里得到快速发展,作为直接存储设备,RAID 具有读写速度快,存储容量大、容错性能好等特点。目前 RAID 已经越来越广泛地用于高性能存储系统。RAID 技术在早期是基于主机的,称为软 RAID,但是由于需要占用许多宝贵的系统资源,对计算机运行的性能影响很大。为了解决这个问题,硬 RAID 控制器的设计方案被提了出来,由这个控制器完成磁盘组的管理,实现各种级别的 RAID,主机不再干涉 RAID 本身实现的细节。

RAID 以普通内存作为 Cache,在命中情况下,免除了磁盘频繁的寻道和旋转延迟,并且在采用写回策略时能提高系统的并行处理能力。因此,如何合

理地设计、管理和调度 Cache,已成为提升磁盘阵列整体 I/O 性能的重要手段。

### 2 RAID 控制器 Cache 组件

为屏蔽不同类型接口目标驱动程序的差异,在开发 RAID (Redundant Arrays of Inexpensive Disks)过程中引入了 SCST<sup>[1]</sup>(generic SCSI target subsystem for Linux 的简称)技术,它为各种类型的 SCSI 目标端驱动程序提供统一的接口,降低了驱动程序的开发难度。其中为数据分配缓冲区的接口提供两种分配缓冲区的方式:SCST 分配和设计者自主分配。为了控制缓存的分配,提高 Cache 命中率和系统性能,在设计系统时选择了自主分配管理缓冲区。

**收稿日期:** 2013—09—19; **修回日期:** 2013—11—08

**基金项目:** 航空科学基金项目(2012ZC53040);西北工业大学研究生创业种子基金项目(Z2013131)

不同的目标端驱动程序通过 SCST 来确保自身对 Linux 内核以及 SCSI 中间层的一致性,从而实现目标端驱动程序和存储后台各种设备的连接. SCST 的接口与目标端驱动程序的连接主要由三个部分组成<sup>[2]</sup>:目标端驱动程序(target driver)、SCST 目标端中间层(STML)和设备句柄层(Dev\_handlers)如图1所示.目标端驱动程序负责接收主机命令,维护启动端和目标端的传输通道,并根据命令处

理状态向主机返回信息. STML 层负责对目标端驱动程序提交的命令进行预处理,包括数据缓冲区的分配和状态切换,提取相关的命令信息,提交命令至设备句柄层,控制整个 I/O 流程向前推进. 设备句柄层包含众多命令处理的回调函数,这些函数已经被注册到 SCSI 目标端中间层,接收来自 STML 传递的命令参数,在合适的时候被 STML 调用,并将 SCSI 命令向底层提交,完成命令的执行.

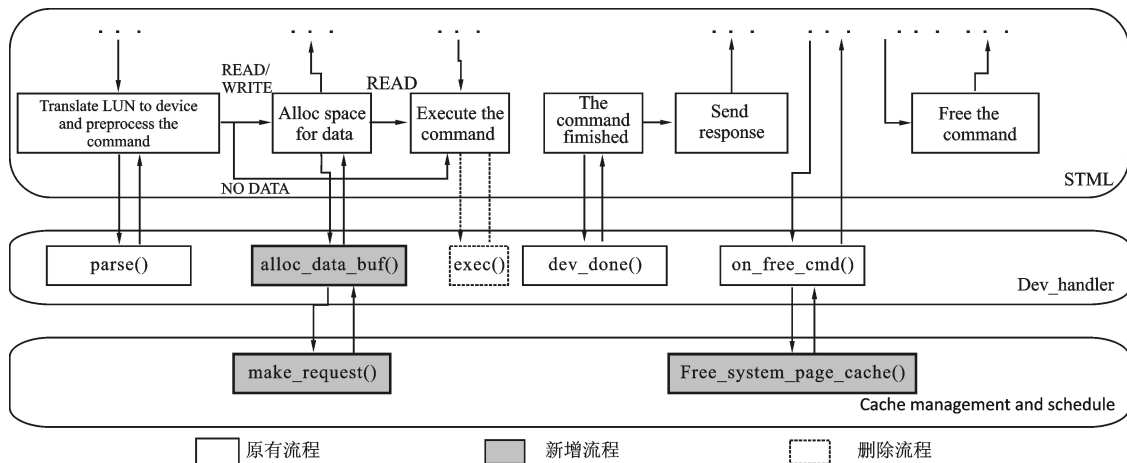


图1 SCST 命令处理流程

在这个过程中 STML 为每次请求都分配数据缓冲区,但真正的读写操作是在对应的设备句柄层完成的,如图1中虚线模块所示,由 Dev\_handlers 层的 exec() 执行. 完成后将执行结果拷贝到 STML 层分配的缓冲区中,该缓冲区仅仅缓存数据,不具备 Cache 的管理功能,额外增加了数据拷贝次数.

引入控制器 Cache 后,将设备句柄层作为统一接口,称作 Dev\_handler 层,读写请求缓冲区的分配统一交由控制器 Cache manager and schedule 层执行. 如图1粗线条阴影模块所示,在 Dev\_handler 层分配数据缓冲区接口将缓冲区的分配接管到 Cache manager and schedule 层. 在这一层通过发送请求接口向下执行缓冲区的分配、管理和调度.

### 3 管理策略

与文献[3,5]在 RAID 控制器设计中引入的 Cache 不同,该 Cache 是控制器中独立存在的模块,与 SCST 没有隶属关系. 控制器 Cache 的设计中不区分读 Cache 和写 Cache,通过状态标识和锁机制来控制读写操作和资源互斥,提高了缓存利用率.

#### 3.1 Cache 设计

缓存页有两种状态:已使用和未使用. 已使用的缓存页是被锁定的,其他请求在该缓存页被释放前

不能使用它,未使用的缓存页则可以被任意请求占用. 鉴于此,系统设计将整个 Cache 按页划分到两个双向链表中,页空闲队列和页锁定队列,分别管理当前空闲和已被占用的内存空间,页锁定队列中不再使用的缓存页,将释放回页空闲队列,以便后续请求使用.

其中,页锁定队列的设计与页空闲队列略有不同. 在申请空闲缓存时从页空闲队列的队头依次进行,只要有空闲缓存,一定可以申请到,时间复杂度为  $O(1)$ . 但检查是否命中时需要依次遍历整个页锁定队列,时间复杂度为  $O(N)$ ,当 Cache 容量较大时  $N$  的值会很大,耗费大量系统时间. 因此为了提高查找效率,在设计页锁定队列时,采用分组映射<sup>[6]</sup>的思想,按照索引分类将页锁定队列设计成由若干子队列构成的链表数组,以 10 000 个子队列为例,每次查找仅需查找 1/10 000 的页锁定队列,大大缩减了查找长度,有助于提高 Cache 命中率,提高系统性能.

#### 3.2 Cache 实现

2.1 节中的 Cache 设计中提到将页锁定队列中不再使用的缓存页释放,以便后续请求使用. 在 Cache 调度实现中,具体采用最近最少使用算法 (LRU),选择 Cache 中最长时间没有被使用的数据

页进行释放,但若每次释放都直接完全释放回页空闲队列,当再次请求该缓存页时将无法命中.为了提高 Cache 命中率,减少申请频繁度,在设计中对 LRU 算法做了如下优化.

页锁定队列中的缓存页释放回页空闲队列时,并不执行真正的释放操作,而是将该缓存页指针挂接到页空闲队列的队头,但页锁定队列依旧保留着该缓存页的指针和数据.根据数据访问的局部性原

理,当新的请求到达时,相对于从未申请过的数据页,该页被再次请求的概率很大.假设仍是对该缓存页的操作,由于页锁定队列还保留有该缓存页指针,此时仅需将其从空闲缓存队列中释放即可,无需重新填充数据页信息.具体如下:

(1)如果是读请求,若命中,命中的 Cache 块直接调整到 LRU 链表即页锁定队列的头结点之后,如图 2 所示.

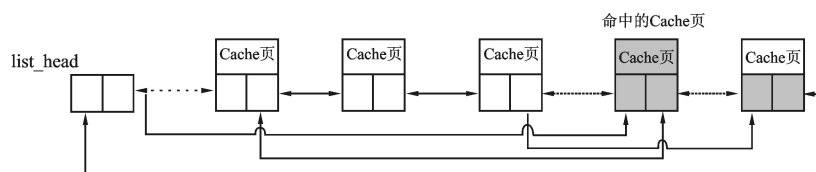


图 2 读请求命中调换位置

(2)如果是写请求,若命中且命中的 Cache 块仅在页锁定队列中,情况和上面一致;若命中的 Cache 块在空闲缓存队列页空闲队列中,则需要将其从页空闲队列中删除,并将其调整到页锁定队列的头结点之后.图 3 中阴影的 Cache 页是当前已不再使用

的缓存页,此时并没有真正地释放,而是将其同时挂接到页空闲队列的队头.这样设计不仅可以提高 Cache 命中率,而且降低了系统开销,最终使得系统的整体性能得以大幅提升.

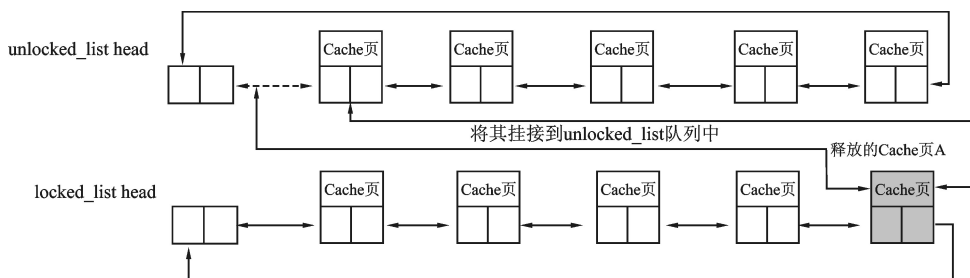


图 3 写请求命中已释放缓存页

#### 4 I/O 延迟

Cache 需要频繁的向磁盘读或写数据,产生大量的 I/O 操作,而 I/O 操作对系统性能有着很大影响.因此尽可能地减少 I/O 次数是提升系统性能不可忽视的方向.高效的预取算法可以减少磁盘读 I/O,而写 I/O 的减少需要合理地设置写延迟.如何有效地控制写延迟是本文研究的另一个重点.

Cache 的磁盘策略<sup>[7]</sup>总体上分为写回和写穿.写穿是信息在写入 Cache 的同时立即刷新到磁盘上,虽然保证了数据的安全性,但是 I/O 次数过多,严重影响系统性能.写回策略是将数据滞留在 Cache 中,一段时间后统一刷新到磁盘上,这就可以减少大量的 I/O 次数,因此系统首选写回策略.为了配合电梯算法,尽可能减少磁头寻道次数,系统采用树结构组织写请求完成的 Cache 页.树能够保证数据稳定有序,满足磁盘驱动程序的电梯调度算法.

为此系统为每个物理磁盘都维护一棵页同步树和页写树.对于写回操作结束后释放的缓存页,并不立即刷新到磁盘,而是将其挂接到页同步树中.在这个过程中若有新的写操作,便可以将两次写操作的数据在一次 I/O 操作中完成,其过程如图 4 上半部分所示.同理,只要 Cache 空间足够,可以合并多次写操作作为一次 I/O,在大数据请求的情况下可以减少大量写 I/O 次数,对系统性能的提升有很显著的效果.

与 RAID0、1 不一样的是,RAID5、6 等需要校验操作,写入磁盘中的数据不仅是有修改的数据页,还需要校验信息,以便在故障时能够恢复原始数据,那么就要有不同的处理方法.为此,系统专门设计了一棵条带写树和条带同步队列.用户请求的数据完成后,并不直接将其挂接到页写树中.首先在条带同步队列中根据条带划分和校验算法完成校验信息的填充或修改.与无校验类型的 RAID 类似,设置条带

写树是为了延迟数据刷新到磁盘的过程,以减少 I/O 次数,具体过程如图 4 下半部分所示。

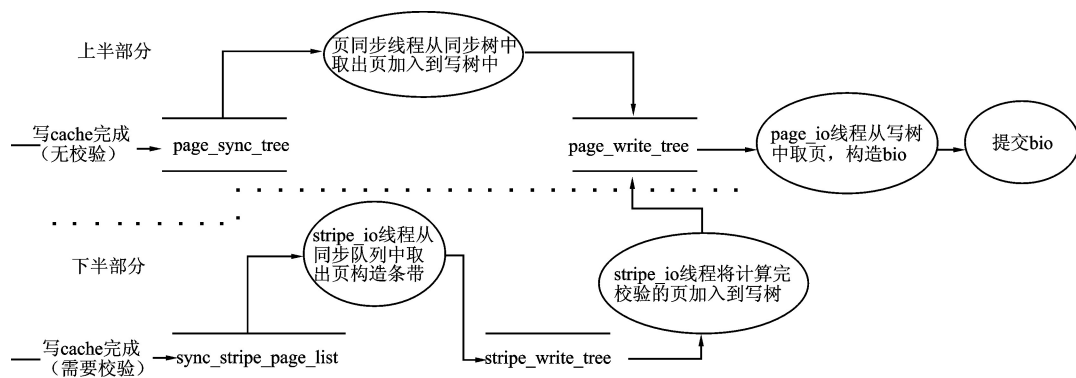


图4 写操作数据流程图

## 5 分析与评估

### 5.1 试验环境与评价指标

该 RAID 控制器原型集成了 Linux-SCSI 子系统、SCSI 启动器模块和 QLA2562 光纤通道驱动器模块,其硬件配置如表 1 所示。

表 1 控制器原型测试硬件

| 配置硬件配置     | 目标端                   | 发起端                  |
|------------|-----------------------|----------------------|
| 机器的型号      | Intel 开发板             | HP Proliant ML150G6  |
| 操作系统       | CentOS 6.0            | windows server       |
| 处理器        | Linux 2.6.34 内核       | 2003 Enterprise 32 位 |
| 内存         | Intel LC5528,4 核 8 线程 | Intel E5504 4 核 4 线程 |
| 网卡及光纤卡配置情况 | 2 GB                  | 8 GB                 |
|            | 光纤卡 Qlogic2562 双口     | 光纤卡 Qlogic2560 单口    |

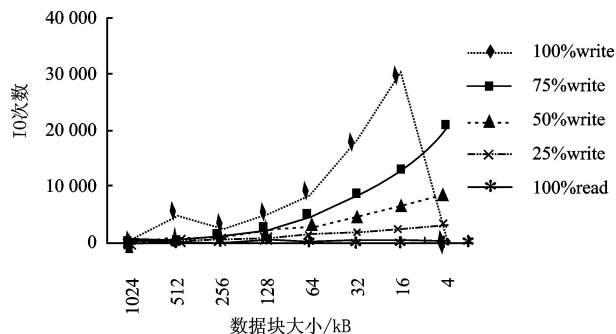
评价磁盘阵列性能最常用的指标是数据的读写速度。本文使用存储领域测试读写速度的专用软件 Iometer。测试过程中 Cache 的总容量为 640 MB,测试以 RAID5 为例。

### 5.2 性能测试与分析

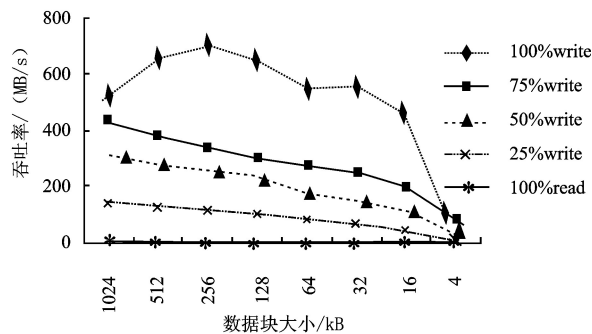
对 RAID 控制器性能测试的结果与负载特性密切相关。本文重点测试了在请求数据块大小不同的情况下,原型机顺序写和随机写的性能。

图 5 所示为顺序写情况下的测试结果。由于 256 kB 最接近 RAID 条带大小,因此在数据块请求大小为 256 kB 时 IO 请求次数最少,达到 2 583 次 MB,此时的数据传输率也最大,吞吐率几乎达到 700 MB/s,已很接近 8 GB 光纤下的理论上限值。在相同测试条件下,取消 Cache 模块中的写延迟策略,IO 性能则显著下降。

图 6 所示为随机写情况下的测试结果。可以看出在随机写的情况下,请求数据块越大,写 IO 性能越好。但是在随机写情况下的数据传输率远没达到理想效果。根据测试结果可得出如下结论:



(a) 不同比例下顺序写IO次数



(b) 不同比例下顺序写吞吐率情况

图5 不同数据块请求下各比例顺序写的写数据传输率

(1)原型机在顺序写的情况下,数据传输率随着请求数据块与条带大小差异越小,数据传输率越高,峰值数据传输率较理想,几近 700 MB/s。

(2)随机写 IO 性能还不太理想,虽然有数据延迟,但是不同数据读写模式应区别处理。因此对数据请求读写模式的识别和策略优化还有待改进。

## 6 结束语

本文研究基于 RAID 控制器的 Cache 管理策略,并重点介绍了如何通过特定的算法,利用数据访问的局部性来提高 Cache 命中率,以及通过构建特定的树结构,利用多棵树之间在不同的情况下切换来延迟写操作,减少磁盘 I/O 次数。通过在原型系

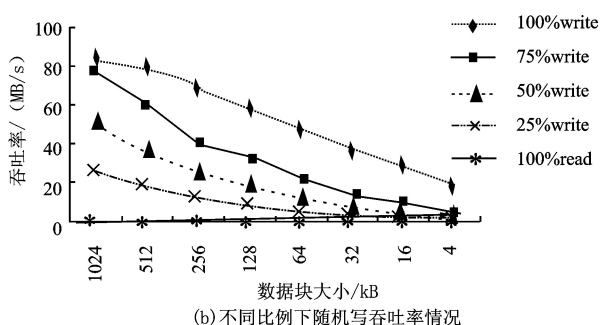
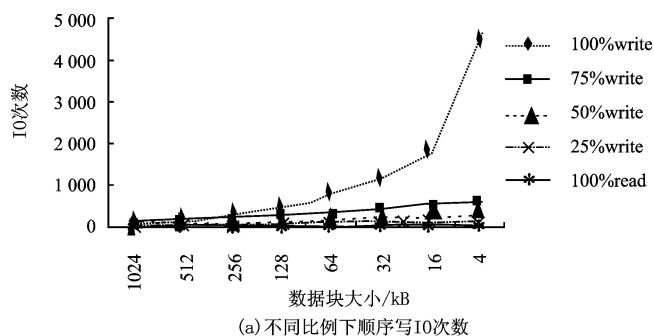


图6 不同数据块请求下各比例随机写的写数据传输率

统上的试验,证实了该策略的可行性和高效性.在8G光纤下I/O吞吐量几乎可达到700 MB/s,已很接近理论值.为了进一步提高控制器的性能和稳定性,对控制器Cache的替换和预读策略的优化以及读写模式识别和区别处理是下一步的研究方向.

#### 参考文献:

- [1] SCST SGV cache description[EB/OL]. [http://scst.sourceforge.net/sgv\\_cache.html](http://scst.sourceforge.net/sgv_cache.html).
- [2] Generic SCSI target mid-level for Linux(SCST)[EB/OL]. [2008-09-19]. [https://git.ipxe.org/mirror/scst/.git/blob\\_plain?\\_f=scst/README](https://git.ipxe.org/mirror/scst/.git/blob_plain?_f=scst/README).
- [3] 杨巍.采用基树的磁盘阵列Cache技术研究[D].武汉:华中科技大学,2009.
- [4] 刘志强,王丽芳,王新民,等.基于存储处理器的Raid控制器研究[J].西北工业大学学报,2010,28(4):633-636.
- [5] 胡泉波.基于FC的磁盘阵列的研究与设计[D].成都:电子科技大学,2011.
- [6] 魏沁祺,谢长生,刘瑞芳,等.基于结构化模型的高性能RAID系统的设计与实现[J].小型微型计算机系统,2007,28(2):376-380.
- [7] QinQi Wei, ChangSheng Xie, Xu Li. Stripe-Cache: An Efficient Cache Scheme for Building Multimedia Oriented RAID System[C]//Human-computer interaction, Beijing, China:IEEE,2007.

#### 作者简介:

陈云霄 女,(1988-),硕士研究生.研究方向为网络与信息安全、分布式网络存储.

王丽芳 女,(1964-),教授.研究方向为电子商务安全、分布式网络存储、云存储.

蒋泽军 男,(1964-),教授.研究方向为云存储、嵌入式系统.

(上接第171页)

[9] 李鹏,孙未蒙,李文强,等.一种改进积分滑模面在飞控中的应用[J].控制工程,2010,17(3):269-289.

[10] 杨海霞.基于改进数据过滤算法的异构数据库优化挖掘技术[J].科技通报,2013,29(6):125-127.

#### 作者简介:

武俊生 男,(1965-),硕士,副教授.研究方向为软件开发及

网络应用.

王伟 男,(1980-),硕士,讲师.研究方向为计算机网络技术、通信技术.

任志勇 男,(1971-),硕士,副教授.研究方向为自适应信号处理、宽带接入技术.