第 33 卷 第 5 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.5 Computer Engineering March 2007
·工程应用技术与实现· 文章编号：1000—3428(2007)05—0214—03 文献标识码：A 中图分类号：TP334

基于 RAID50 的存储系统高速缓存设计

谭怀亮，贺再红
（湖南大学计算机与通信学院，长沙 410082）

摘 要：设计了一种多个控制卡分条的 RAID50 模型和主机虚拟卷地址到所属高级别阵列、低级别阵列分条的二级地址映射模式，以提高 I/O 访问的并发性。定义多个连续 I/O 块为扩展块并等同于 Cache 页大小，且作为 RAID50 传输粒度，进一步改善内存与存储设备之间的传输效率。设计了基于 Cache 描述符控制块的哈希链式查找算法和基于 Cache 页访问频率计数的二次机会置换算法，实现了一种主机数据接收与 RAID50 存储设备预读并发进行的策略。结果表明，该设计有效地提高了存储系统的 I/O 性能。

关键词：控制卡分条的分布式冗余校验磁盘阵列；高速缓存；高级别阵列；低级别阵列；控制卡分条

Design of Storage System Cache Based on RAID50

TAN Huailiang, HE Zaihong
(School of Computer and Communication, Hunan University, Changsha 410082)

【Abstract】The RAID50 model of multi-controller stripe and the two-level address mapped mode of the host virtual volume address to high array and low array stripe is designed for enhancing I/O access concurrency. The transmission efficiency between the Cache memory and storage media is improved by using extend block that consists of sequential I/O multi-block and equals with Cache page size as RAID50 transmission granularity. The hash link lookup algorithms on Cache descriptor block and the second chance replacement algorithms on Cache page access frequency count are designed. The concurrent policy of the host data receiving and storage device pre-reading is implemented. The results show that the design improves effectively the storage system I/O performance.

【Key words】RAID50; Cache; High array; Low array; Adapter stripe

近几年，CPU 的处理速度已达到几 GHz，磁盘存储密度也以每年 50%的速度增加，平均每 3 年提高 4 倍，但磁盘的存取时间却改善甚微，每 10 年仅减少 1/3，目前仍停留在毫秒级，与内存及 CPU 内部 Cache 的差距为 4～6 个数量级。根据木桶原则，计算机系统整体性能取决于系统中性能最差的关键部件，因此以磁盘作为主要外存储器的外存子系统就成为影响整个系统性能提高的瓶颈。独立冗余磁盘阵列(RAID)是改善磁盘存储性能的有效方法之一。它是将要存储的数据按一定规则分块，存放在多个盘上，利用多个盘的并行存取、并行传输来匹配带宽，提高存取速度，对主机来说它相当于一个快速大容量的磁盘。另外，RAID 还利用其提供的冗余数据(镜像数据或冗余校验信息)提高了系统的可靠性。目前的 RAID 分为 6 级：RAID0～RAID5，其中 RAID5 采用奇偶校验来提供数据恢复的能力，但它没有独立的校验盘，校验信息分布在各个磁盘驱动器上。RAID5 具有高的数据可靠性以及较快的存取速度。然而电子技术的发展进一步增大了主存和外存之间的速度差异。如何有效地利用 RAID 技术及其缓存机制增强外存储系统整体性能是本文研究的主要课题。

本文基于开放式的硬件平台与软件环境，结合 RAID0 和 RAID5 技术，设计了一种 RAID50 模型，并在该模型的虚拟化引擎中设计与实现了高速缓存 Cache 策略以改善存储系统 I/O 性能。

1 RAID50 系统模型

RAID50 是建立在 RAID5 和 RAID0 的基础上而形成的。同一个 RAID5 阵列卡上的 3 个或 3 个以上硬盘构成低级别阵列 La(Low array)，合并位于多块阵列卡上的低级别阵列构成控制卡的 RAID0 高级别阵列 Ha(High array)，在 Ha 内的每一个 La 必须有相同的磁盘数。Ha 内的数据通过逻辑块地址 LBA(Logical Block Address)寻址，通常每个块是 512B(即磁盘扇区大小)。Ha 内连续的多个逻辑块构成一个扩展块，Ha 按扩展块实现控制卡的 RAID0 分条。进一步，Ha 可根据主机服务器需求划分成多个容量各异的虚拟卷，卷对主机呈现为磁盘驱动器。每个卷用起始的 LBA 在 Ha 中的偏移和 512B 块数来表示卷的位置和容量。对于每一个卷必须以扩展块对界。扩展块的大小是扇区大小的倍数，也是控制卡的分条大小，即控制卡传输的最小单位，虚拟卷亦可通过扩展块地址来寻址。虚拟卷上的逻辑块与其在物理盘上的位置之间具有两级地址映射关系，即虚拟卷的逻辑块到 Ha 分条(即扩展块)和 La 分条(即物理磁盘)的地址转换。而对于应用程序来说，看到的虚拟卷是一个位于 Ha 内的连续的虚拟磁盘空间。RAID50 模型如图 1 所示。

RAID50 组成及两级地址映射如图 2 所示，La 的 RAID5 阵列由 N 块(block)磁盘组成，有效数据最终按分条分布存储在 N-1 个磁盘上，在同一物理盘上属于同一个分条的连续块块数称作 La 分条深度。每个磁盘上相同位置的块构成一个校验组，每个校验组里包含一个校验块，它是该组中其它数据

基金项目：中国网上教育平台试点工程资助项目“高等教育网上教育系统子项”(计高科[2000]2034)
作者简介：谭怀亮(1969－)，男，博士，主研方向：网络存储和计算机系统结构；贺再红，讲师
收稿日期：2006-11-02 **E-mail：**tanhuailiang@yahoo.com.cn

块的异或值。La 的数据容量是 N－1 个磁盘的总容量。高级别阵列 Ha 的 RAID0 按扩展块进行分条，其大小等于 La 分条深度 *(N-1)。图 2 中由 2 个 RAID5 阵列卡构成 RAID0 高级别阵列 Ha，每个阵列卡有 4 个硬盘组成 RAID5 的低级别阵列 La，La 的有效存储容量为 3 个硬盘，其分条深度为 2 个块，Ha 的分条深度为 6 个块，扩展块亦为 6 个块。因此，一次 I/O 操作可以并发访问 12 个块。即使两个物理磁盘发生故障(每个 La 阵列中一个)，也不会有数据丢失。

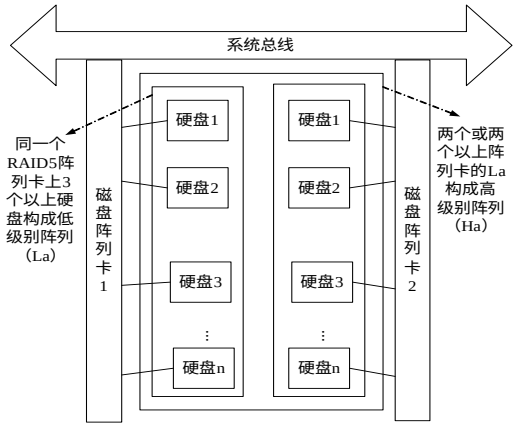

图 1 RAID50 模型

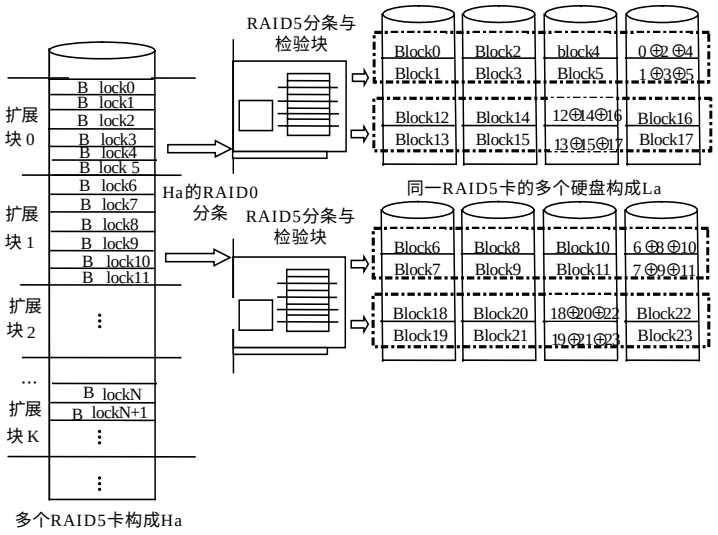

图 2 RAID50 组成及两级地址映射

2 高速缓存设计

2.1 高速缓存结构

基于 RAID50 的存储系统虚拟化引擎除了实现虚拟卷地址到实际物理磁盘地址两级映射外，还实现了高效的高速缓存(Cache)管理策略。在 RAID50 系统中的高速缓存是系统的部分内存(RAM)，它免除了磁盘频繁的寻道和旋转延迟，并且在写回策略时提高了系统的并行处理能力，进一步提高主机 I/O 性能。高速缓存从未使用的内存中分配。有些操作系统(如 Linux)通过设置某些内核启动参数，操作系统可以被限制在一小部分低端内存中。剩下的 RAM 将可以被存储系统虚拟化引擎访问。在 RAM 分配时，一小部分(约 10%左右)可用 RAM 分配给 Cache 描述符控制块，用于管理 Cache 和处理来自主机的事务。所有剩余的 RAM 被分成 Cache 页。Cache 页是保存数据的实体，它一方面暂存来自主机的写数据，以便延迟写入 RAID50 物理存储设备中，另一方面将送往主机的读数据或预读数据保存起来，便于主机的再次读。Cache 页大小等于 RAID50 高级别阵列 Ha 中的扩展块，亦即 Ha 的 RAID0 分条大小。将 Cache 按 Cache 页划分后，I/O 请求也按 Cache 页划分，RAID50 与 Cache 之间的数据交换以 Cache 页为单位进行。Cache 布局的控制区和实体区如图 3 所示，每个 cache 页有一个相应的叫做 Cache 描述符(Cache Discription)的控制块。描述符中包含队列指针(链式列表内的成员)、对应的 Cache 页号、指示 cache 页状态的标志和计数器以及指向 cache 页物理内存地址的指针。描述符从控制区域中分配。

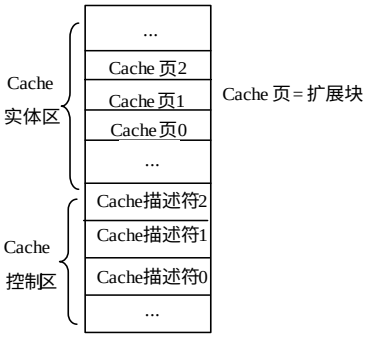

图 3 Cache 页布局

2.2 主机 LBA 到扩展块地址映射

映射给客户服务器主机的虚拟卷按扩展块编址，即按高速缓存页大小编址，来自客户服务器主机的 I/O 命令以<lba,op,blk_number>的形式访问虚拟卷，其中 lba 是请求数据的起始地址，op 是操作类型(如读操作或写操作等)，blk_number 是读/写请求的数据块数。Lba 地址按 Cache 页被转换成扩展块地址，blk_number 被转换成 Cache 页数(扩展块数)。即转化为<Cache_Page_No,op,Cache_Page_number>的形式。虚拟化引擎由此可以计算出主机传输事务在 RAID50 虚拟卷的起始和结束扩展块号以及请求的 LBA 在起始扩展块内的字节偏移。对于一个不够完整的 Cache 页数据传输，或者 Lba 起始地址或最后一块 Lba 地址可能位于 Cache 页中间，都得预取一个完整的 Cache 页，并加上 Cache 页内偏移。

2.3 高速缓存页查找算法

RAID50 的 Ha 中的每个虚拟卷维护一个 Cache 描述符(CDs)的哈希表。当接收到主机请求后，虚拟化引擎设置控制块并将主机请求对应的卷相应块地址翻译成 Ha 逻辑块地址，然后确定与该次请求有关的扩展块。下一步，虚拟化引擎根据扩展块地址及其相应的哈希值检查所属虚拟卷的哈希表以确定数据块的位置，哈希表的每一条目是一串指向 Cache 描述符链表的指针，该链表中的元素都是哈希到同一位置的 Cache 描述符控制块。哈希查找如图 4 所示，其算法工作如下：扩展块地址转换到哈希表中，用该地址与链表中的每一个 Cache 描述符元素的扩展块地址域比较。如果在链表中找到了匹配的元素，表示该扩展块在 Cache 中存在，即 Cache 击中，相应的描述符将被指派给主机事务。增加计数器值以指示出对应的描述符正在使用。如果遍历该链表未找到匹配的描述符元素，表示该扩展块在 Cache 中不存在，即 Cache 未命中，则需要获得一个新的描述符。获得描述符的情况有两种：按 FIFO 从自由 Cache 描述符链表中摘取；如果空闲描述符链表(队列)为空(所有的 Cache 块都在使用)，需要通过置换 Cache 页算法。在得到描述符后，设置其中的扩

展块地址域，并初始化计数器及标志位。按 FIFO 将该描述符插入到所属虚拟卷哈希表的相应链表中。

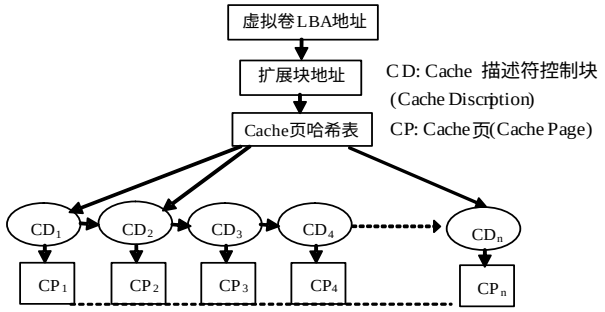

图 4 Cache 页的哈希查找

2.4 高速缓存页置换算法

当需要一个新的 Cache 页时，虚拟化引擎试图从空闲 Cache 描述符链表中得到一个 Cache 描述符。在初始化启动阶段，所有的描述符都在这个空闲链表中。由于来自多个主机的频繁传输事物，空闲链表的描述符有可能很快耗尽，在此情况下，虚拟化引擎需要使用 Cache 页置换算法以选择一个牺牲页。

RAID50 虚拟化引擎的 Cache 页置换算法是基于访问频率计数的二次机会算法，该算法综合了最近最少使用(LRU)和先进先出(FIFO)两种基本算法。当要选择一个 Cache 页时，检查其访问频率计数。如果少于访问频率阈值，那么就直接置换该 Cache 页。如果访问频率计数大于访问频率阈值，那么就给该 Cache 页第 2 次机会，并选择下一个 FIFO 页。其实现方法是采用循环队列，如图 5 所示。用一个全局索引表示下次置换算法的起始 Cache 页。当需要一个 Cache 页时，检查 Cache 描述符链表中当前索引指向的 Cache 描述符，对于未被保护加锁的、未处于更新状态的、未被修改过的 Cache 页作为第 1 次循环的候选者，索引向前移动直到找到一个访问频率计数少于访问频率阀值的 Cache 页。在最坏情况下，所有 Cache 描述符中的访问频率计数均大于访问频率阈值，索引会遍历整个循环队列，以便给每个 Cache 页第 2 次机会。如果是第 2 次循环，直接从链表中摘取索引指向的 Cache 描述符，二次机会置换就变成了 FIFO 置换。当全局索引到达其最大值时，即已配置的 Cache 描述符数，复位全局索引到 0，这样增加了使用 LRU 选择替换算法的公平性。对于脏 Cache 页(已经被主机活动更新，但还没有写回到 RAID50 物理存储设备)被定义为“在使用”状态，不可被置换，因为主机数据传输事务直到写到存储设备完成后(即使主机可能认为写磁盘早已经完成)才认为已经完成。这一点也意味着回收 Cache 页从不依赖对存储设备的写操作，因此回收 Cache 页可以以处理器速度完成。

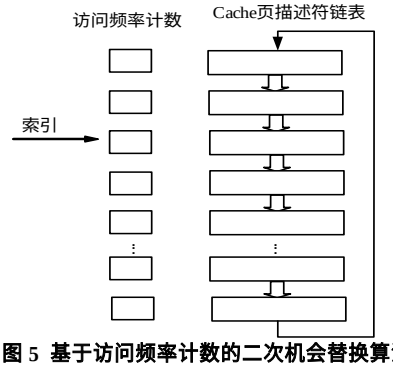

图 5 基于访问频率计数的二次机会替换算法

2.5 数据传输

对于来自主机的读请求，若在 Cache 中命中,可直接从 Cache 中读取扩展块，而不必再访问 RAID50 存储设备。否则本次读为不命中，还要再去访问存储设备。读存储设备结束后，要把这次读请求的扩展块数据调进 Cache。如果 Cache 已满，则要覆盖某些页面。如果要覆盖的页面被修改过，则须先写回到存储设备后才能被覆盖。

对于完整扩展块写不需要从 RAID50 存储设备中预读(所有数据将从主机获得)，而当处理部分扩展块(非完整扩展块)的写操作时，需要从 RAID50 存储设备预读完整的扩展块，并且需获得额外的 Cache 页容纳来自主机的数据。这一点允许主机数据的接收与 RAID50 存储设备的预读并发进行。当二者完成后，拷贝来自主机的数据到与该次写操作相关的扩展块对应的 Cache 页中，并同步 Cache。当所有与该次数据传输事务关联的 Cache 页存在于内存中并同步，虚拟化引擎发送确认消息给主机(Cache 模式应答)，对 RAID50 存储设备的写操作被发起。当回写操作完成后，即对 RAID50 存储设备的写操作确实提交到 RAID 控制卡，Cache 页被标记为干净页，主机被应答。

3 应用测试与结论

基于 RAID50 模型及其 Cache 实现方法笔者利用 x86 平台和 PCI-X/PCI-E 接口的 RAID5 控制卡开发了 RAID50 虚拟化引擎，实现了服务器主机存储空间的动态按需分配。并进行了相应的测试。

测试任务：RAID50 存储系统的 I/O 性能。

RAID50 存储系统的配置是：SUSE Linux/Infiniband Target + RAID50 虚拟化引擎/ Intel Xeon 3000MHz/1GB 内存/Infiniband TCA/RAID5 Adapter * 2/ Hard Disk Drive *16。

服务器主机平台是：Intel Pentium4 3.0GHz/1GB 内存/Infiniband HCA /Red Hat Linux10.0/Infiniband Initiator。

测试采用 Intel 公司开发的一个专门测试系统 I/O 速度的测试软件 Iometer，图 6 所示为 5 个服务器主机开 11 个读线程访问同一个虚拟卷，I/O 访问速度最大达到了 4 529Mbps，远远大于两个 RAID5 阵列卡所提供的最大带宽(约 600Mbps)。

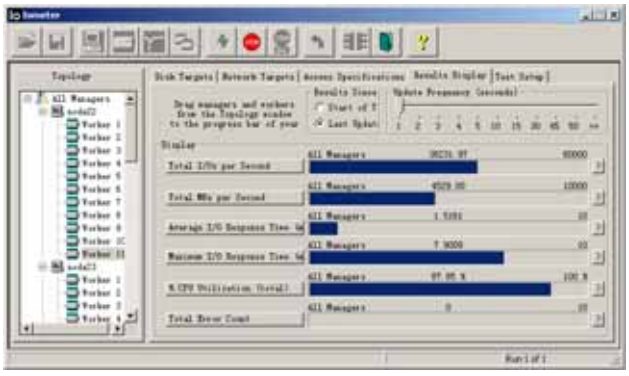

图 6 I/O 测试

由此可见，基于开放式的硬件平台与软件环境，结合 RAID0 和 RAID5 技术，设计的 RAID50 模型及其高速缓存 Cache 策略，将多个 RAID5 控制卡构成 RAID0 阵列，提高了 I/O 访问的并发性，以 Cache 页大小作为 RAID5 传输单位，进一步改善内存与存储设备的传输效率。实验证明，基于 RAID50 及其缓存机制实现的存储系统具有很好的 I/O 性能。

（下转第 242 页）