一种提高磁盘阵列重建效率的缓存替换算法

毛 波[1] 吴素贞[2] 冯 丹[1]

(1 华中科技大学计算机科学与技术学院,湖北 武汉 430074;
2 厦门大学信息科学与技术学院,福建 厦门 361005)

摘要 磁盘的高故障率导致磁盘阵列重建成为数据处理中心一个经常性发生的事件.为了提高磁盘阵列的重建效率,提出了一种新的磁盘阵列缓存替换策略,即分支刷新算法,从缓存管理的角度加快磁盘阵列的重建过程.分支刷新算法淘汰脏数据到后备磁盘阵列时,优先淘汰重建区域附近的脏数据块,从而减少了磁头在重建区域和淘汰区域之间的移动开销,尽可能地保证重建过程的顺序性.仿真实验结果表明:对比传统的最近最少使用缓存淘汰算法,分支刷新算法显著地提高了磁盘阵列的重建效率.通过对4种负载的回放测试,分支刷新算法平均减少了41.6%的磁盘阵列重建时间和16.1%磁盘阵列重建过程的平均用户响应时间.

关键词 存储系统;磁盘阵列;数据重建;缓存管理;替换算法
中图分类号 TP393 **文献标志码** A **文章编号** 1671-4512(2011)06-0054-04

Cache replacement algorithm for improving RAID reconstruction efficiency

Mao Bo[1] *Wu Suzhen*[2] *Feng Dan*[1]

(1 College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China; 2 School of Information Science and Technology, Xiamen University, Xiamen 361005, Fujian China)

Abstract High failure rate of disks often leads to RAID (redundant array of independent disks) reconstruction a common case in large-scale data centers. To improve the efficiency of RAID reconstruction process from the perspective of the cache management, a new cache replacement algorithm, i.e., branch destage algorithm, was proposed. The branch destage algorithm updated the dirty data that was close to the reconstruction region to the disks firstly, thus reducing the moving overhead of the disk head between the reconstruction region and the region that services the user request, and making the reconstruction process more sequential in disks. The simulation results show that the branch destage algorithm outperforms LRU (least recently used) significantly. By four real traces evaluations, the branch destage shortens the RAID reconstruction time by 41.6% and the average user response time during reconstruction by 16.1%, respectively.

Key words storage system; RAID; data reconstruction; cache management; replacement algorithm

磁盘是一种机械装置,容易发生故障.当磁盘阵列中某个磁盘发生故障时,磁盘阵列将自动恢复故障盘上的数据,并写入替换盘或热备份盘中,该过程称为磁盘阵列的数据重建过程[1].在磁盘阵列重建期间,内部重建请求和外部用户I/O请求共享磁盘带宽,所以用户对磁盘阵列的访问请求将受到重建请求的影响,同时磁盘阵列的重建时间也将受到外部用户请求强度的影响.由于故障恢复时间(MTTR)直接影响到磁盘阵列存储系统的可靠性[2],因此如何在系统的可靠性和用户感知的系统性能这2方面达到一个平衡点或者双赢是一个挑战性的问题.

任何到达磁盘阵列存储系统的用户I/O请求都将经过缓存的过滤,因此缓存的替换策略对

收稿日期 2010-11-10.
作者简介 毛 波(1983-),男,博士;吴素贞(通信作者),讲师,E-mail:suzhen@xmu.edu.cn.
基金项目 国家高技术研究发展计划资助项目(2009AA01A402);国家重点基础研究发展计划资助项目(2011CB302300).

存储系统的性能影响较大[3-6]. 现有的缓存替换算法多关注提高存储系统的性能和能耗，而没有考虑其对磁盘阵列重建效率的优化效果. 针对该问题，本研究提出了一种提高磁盘阵列重建效率的缓存替换算法，即分支刷新算法. 当分支刷新算法淘汰脏数据到后备磁盘阵列时，优先淘汰重建区域附近的脏数据块，从而减少磁头在重建区域和淘汰区域之间的移动开销. 仿真实验结果验证了对于写请求比较多的应用，分支刷新算法明显减少了重建时间和平均用户响应时间，提高了磁盘阵列存储系统的可靠性和性能.

1　磁盘阵列的重建过程

以 Linux 2.6.11 中软磁盘阵列的数据重建流程为例说明磁盘阵列重建的过程，如图 1 所示. 默认的系统配置参数中，重建带宽的上下阈值设置分别是 200 和 1 MB/s. 该设置表明系统优先响应用户请求(用户应用请求)的优先级大于重建请求. 用户可根据需求通过/proc/sys/dev/raid/*接口动态地调整重建带宽的上下阈值，对重建速度进行在线调整.

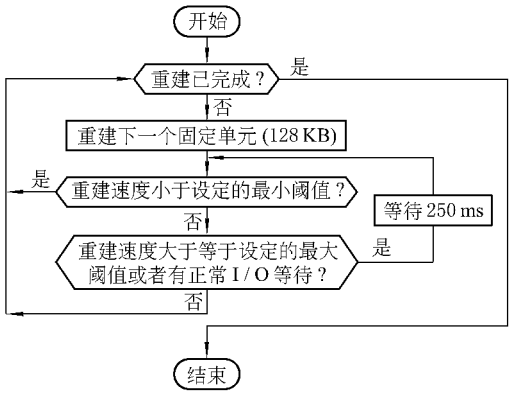

图 1　Linux 2.6.11 操作系统中软磁盘阵列的重建流程

重建过程以块为单位向前推移，每重建一个数据块(默认 128 KB)后，系统将检测重建速度是否在设定的范围内. 若重建速度小于设定的最小重建带宽，则重建将继续进行，此时不管有没有用户 I/O 请求等待，都将优先执行重建请求；否则检查重建速度是否大于等于设定的最大重建带宽或者是否有用户 I/O 请求在等待，若是，则重建线程挂起等待 250 ms，然后再进入循环检测中，直到故障磁盘上的数据都被恢复到替换盘上时，重建过程完成.

以 4 个磁盘组成的磁盘阵列级别 5 为例说明重建过程中具体的 I/O 请求处理流程，如图 2 所示. 图中磁盘 2 发生故障，此时要恢复磁盘 2 上的数据块 6，重建线程向磁盘 1、磁盘 3 和磁盘 4 上对应的条带发送读请求，读取数据块 5，校验块 P_1 和数据块 4，然后对读取的数据进行异或计算得到数据块 6，再将计算得到的数据块 6 写入替换盘中对应位置，该条带的数据恢复完成[7]. 按照相同的原理恢复下一个数据块，直到磁盘 2 上所有的数据都恢复到替换盘中，重建过程完成.

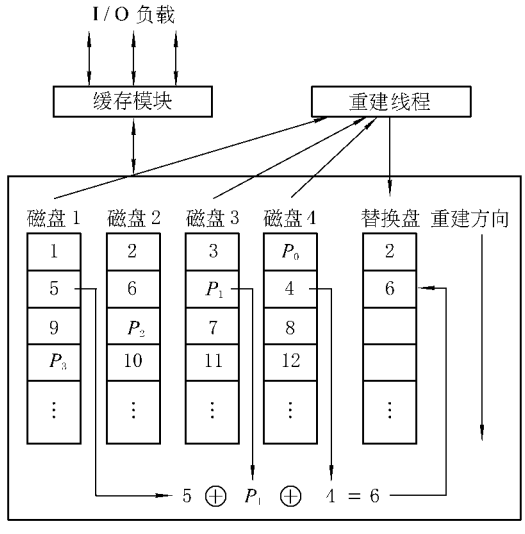

图 2　磁盘阵列级别 5 的重建过程示意图

对于在线重建，重建请求被来自用户的应用请求不断地打断，导致磁头在磁盘上来回移动，磁盘的服务效率大大降低，既延长了重建时间，又影响了用户请求的响应速度；因此，应该让用户 I/O 请求感知重建的存在，进而合理地调度用户 I/O 请求以最大化地利用磁盘资源. 图 2 同时也显示了所有用户 I/O 请求都要经过缓存过滤，然后才到达磁盘阵列设备. 现有的缓存管理算法仅仅考虑了对系统性能[8-10]或者系统能耗[11]的优化，而没有考虑对系统可靠性的优化和提高. 由于缓存管理算法对用户请求到达磁盘阵列的序列影响很大，因此影响了磁盘阵列的重建效率. 针对该问题，本研究文从磁盘阵列缓存替换算法的角度对磁盘阵列的重建效率进行优化，即提出分支刷新算法.

2　分支刷新算法

将缓存中的脏数据从高一级存储设备移动到低一级存储设备的过程称为刷新操作. 在磁盘阵列存储系统中，刷新操作指的是当磁盘阵列控制器缓存中的脏数据超过设置的缓存空间上限后，从磁盘阵列缓存中将脏数据刷入到后备磁盘阵列的过程. 磁盘的机械特性保证重建请求在磁盘上

的顺序性,可以大大提高磁盘阵列的重建性能.刷新操作的效率由2个方面的因素决定:a. 刷新的速度,因为刷新过程中磁盘资源被刷新操作占据,用户请求将被阻塞,因此影响了用户感知的系统性能;b. 刷新的请求序列,一次刷新过程将进行多次磁盘写操作,若利用磁盘的顺序写性能优势将这些写请求按照地址先后顺序写入磁盘,则可大大提高磁盘的利用率.

综合考虑以上2个因素,分支刷新算法在每次刷新脏数据时同时考虑当前重建位置因素,只刷新当前重建位置附近的数据块,这就减少了刷新操作与重建位置之间的磁盘寻道开销;并且在每次刷新操作中,刷入磁盘的数据块按照地址先后顺序写入磁盘,大大提高了分支刷新操作的效率并减少了对应刷新操作的时间,进而减少了对系统性能的影响.如图3所示:在所有的数据块中选择要刷入后备磁盘的脏数据块时,由于当前重建位置在D_9附近,因此优先刷新数据块D_4,D_9,D_{10}和D_{11},并按照这样的顺序写入磁盘,而将脏数据块D_{x+1},D_{x+2}和D_{y+2}暂时保存在缓存中,直到重建位置到达其附近时再将其刷入磁盘阵列中.

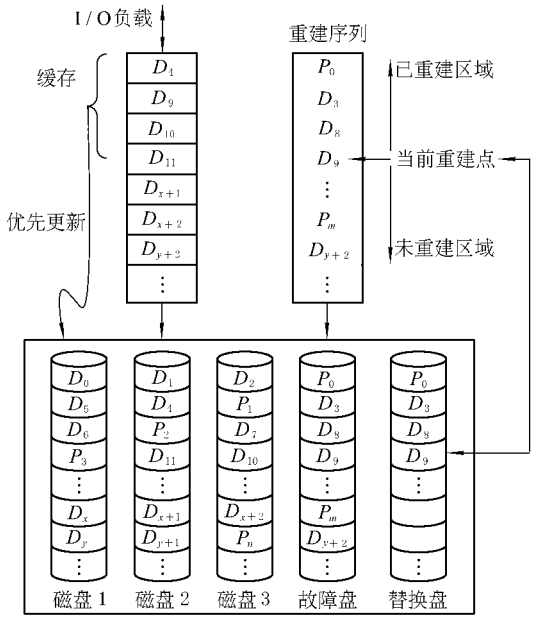

图3 分支刷新示意图

分支刷新算法的具体实现过程如下:

a. 当缓存中的数据块超过了预先设定的阈值时,唤醒刷新线程,转步骤b;

b. 刷新线程首先检查系统是否处于重建状态,若是则获取磁盘阵列数据正在重建区域的地址(即recovery_offset的值,在用户态下可通过系统提供的接口/sys/block/md0/md/sync_completed获得),转步骤c,否则直接按照最近最少使用(least recently used, LRU)策略淘汰数据,转步骤d;

c. 从LRU列表的尾部向前搜索并获取与当前重建地址相近的数据块(如当前重建地址前后100 MB的区域),作为备选淘汰数据块,同时暂停发起重建请求,转步骤d;

d. 将缓存中要淘汰的数据块按地址偏移重新排序,并依次写入磁盘阵列中,腾出的空闲缓存空间能够继续服务用户I/O请求,此时若系统处于重建状态,则转步骤e,否则转步骤a;

e. 检查重建是否完成,是则退出,否则发起新的重建请求,直到新用户请求到达,且缓存的使用空间超过预先设定的阈值,转步骤a.

从上述刷新过程可以看出,分支刷新算法并不会影响正常状态下的磁盘阵列性能.只有当磁盘阵列处于重建过程时,分支刷新算法才会发挥作用,其用途在于提高磁盘阵列的重建效率.

3 测试结果

系统性能测试在一台服务器架构的磁盘阵列中进行,该系统采用英特尔志强3.0 GHz的处理器和1 GB的内存,3Ware 9650的磁盘适配器连接西部数据WD2500YD的SATA磁盘.实验中采用的真实负载是在线事务处理的负载和惠普实验室收集的应用负载Cello99,在线事物处理负载是2个商业应用环境下的应用负载(Financial1和Financial2),应用请求的回放通过RAIDmeter工具[12]发送和接收负载的用户请求和响应.实验中用动态随机存储内存来代替非易失性随机存储内存使用,因为在存储控制器中的非易失性随机存储内存大多采用电池备份的随机存储内存,所以性能是一样的.由于大容量磁盘的同步时间和修复时间很长,因此实验中对磁盘均采用分区的方式,分区大小为10 GB,所有测试中磁盘阵列均以相同的容量进行对比测试[13].缓存管理策略内嵌在RAIDmeter中,用于和分支刷新算法对比的缓存管理策略是LRU算法,这也是应用最广泛的算法之一[14].为了得到准确的实验结果,测试运行一段时间后再触发重建进程,避免了缓存大小对重建的影响,所以在测试中缓存的大小对结果的影响较小.

表1是LRU和分支刷新2种缓存替换算法对重建性能影响的对比结果.从表中可以看出:不管是重建过程的平均用户响应时间还是磁盘阵列的重建时间,分支刷新算法都优于传统的LRU

算法. 特别地，分支刷新算法对平均用户响应时间的优化更为明显，最高达到 76.7%，而对磁盘阵列的重建时间最多只优化了 27.7%，平均优化效果分别为 41.6%和 16.1%. 通过对以前的研究结果[13]和本文的实验结果进行对比，发现 Linux 2.6.18 及以上的内核版本中的软磁盘阵列重建模块更加偏向于重建进程，导致优化的重建算法对磁盘阵列的重建时间提升的幅度比较有限，而将优化效果转向了用户的平均响应时间. 通过常态下的用户平均响应时间和重建过程中用户响应时间的对比也可以发现，用户平均响应时间明显增大. 而若在低版本的内核下测试(例如 Linux 2.6.11)，则用户的平均响应时间和常态下的平均响应时间差距不会太大.

表 1　LRU 和分支刷新 2 种缓存替换算法对重建性能影响的对比结果

负载	平均响应时间/ms		重建时间/s	
	LRU	分支刷新	LRU	分支刷新
Financial1	301.4	268.9	220	180
Financial2	405.4	272.7	300	270
Cello99-a	1 730.0	928.2	204	187
Cello99-b	3 531.0	822.6	286	207

分支刷新算法可以有效地减少刷新过程的写延迟，进而可以更快地腾出有效的缓存空间接受上层应用的写数据，减少了用户写请求的等待时间. 同时，由于腾空缓存的时间减少，并且腾空到磁盘的缓存数据都处于重建区域附近，有效地减少了磁头在重建区域和刷新数据区域之间移动的延迟开销，从而加快了磁盘阵列的重建进程，提高了磁盘阵列的重建效率. 分支刷新算法目前只考虑了对于写请求的处理，没有对读请求进行优化处理，在将来的研究工作中将从理论和实验中分析分支刷新算法对于读请求的优化策略.

参　考　文　献

[1] 田磊，冯丹. RAID 在线数据重建方法仿真器设计[J]. 华中科技大学学报：自然科学版，2010，38(5)：1-4.

[2] Holland M. On-line data reconstruction in redundant disk arrays[D]. Pittsburgh：Department of Electrical and Computer Engineering，Carnegie Mellon University，1994.

[3] 罗益辉，谢长生. 对象存储系统的合作缓存方案[J]. 华中科技大学学报：自然科学版，2008，36(11)：67-70.

[4] 廖小飞，殷江培，程斌. 基于 P2P 的 VOD 系统中数据缓存策略研究[J]. 华中科技大学学报：自然科学版，2007，35(8)：67-79，73.

[5] 罗益辉，谢长生，张成峰. 存储系统的集中式 Cache 替换算法[J]. 华中科技大学学报：自然科学版，2006，34(11)：41-43.

[6] 冯丹，张江陵. 不同负载分布下磁盘阵列响应时间分析[J]. 计算机研究与发展，2001，38(9)：1144-1148.

[7] Lee J Y B，Lui J C S. Automatic recovery from disk failure in continuous media servers[J]. IEEE Transactions on Parallel and Distributed Systems，2002，13(5)：499-515.

[8] Lee D，Choi J，Kim J S，et al. LRFU：a spectrum of policies that subsumes the least recently used and least frequently used policies[J]. IEEE Transaction on Computers，2001，50(12)：1352-1361.

[9] Bansal S，Modha D S. CAR：clock with adaptive replacement[C]// Proceedings of the 3rd USENIX Conference on File and Storage Technologies. Berkeley：USENIX，2004：187-200.

[10] Varma A，Jacobson Q. Destage algorithms for disk arrays with nonvolatile caches[J]. IEEE Transaction on Computers，1998，47(2)：228-235.

[11] Zhu Q，Zhou Y. Power aware storage cache management[J]. IEEE Transactions on Computers，2005，54(5)：587-602.

[12] Tian L，Feng D，Jiang H，et al. PRO：a popularity-based multi-threaded reconstruction optimization for RAID-structured storage systems[C]// Proceedings of the 5th USENIX Conference on File and Storage Technologies. Berkeley：USENIX，2007：277-290.

[13] Wu S，Jiang H，Feng D，et al. Workout：I/O workload outsourcing for boosting RAID reconstruction performance[C]// Proceedings of the 7th USENIX Conference on File and Storage Technologies. Berkeley：USENIX，2009：239-252.

[14] Megiddo N，Modha D S. ARC：a self-tuning，low overhead replacement cache[C]// Proceedings of the 2nd USENIX Conference on File and Storage Technologies. Berkeley：USENIX，2003：115-130.