第32卷 第16期 计 算 机 工 程 2006年8月
Vol.32 № 16 Computer Engineering August 2006
·工程应用技术与实现· 文章编号：1000—3428(2006)16—0237—03 文献标识码：A 中图分类号：TP393.08

基于磁盘异或引擎的 RAID-5 小写性能优化

谭毓安[1]，王婉星[1]，于 强[1]，朱立谷[2]，张雪兰[1]
(1. 北京理工大学计算机科学与工程系，北京 100081；2. 华中科技大学国家外存储系统专业实验室，武汉 430074)

摘 要：SCSI 标准中已扩充了新的 SCSI 命令(XDWRITE，XDREAD，XPWRITE 等)，用以实现高效 RAID-5 写操作。对“小写”操作，传统的方法需要主机或 RAID 控制器读入原有的校验块，通过异或计算来构造新的校验块。采用这些新的 SCSI 命令实现“小写”操作，不再需要读入校验块，由磁盘来进行异或运算构造出校验块。利用磁盘的异或引擎，提高了 RAID-5 的吞吐率，缩短了平均响应时间。
关键词：RAID；小写；异或引擎；磁盘阵列

Optimization for RAID-5 Small-write Performance Using Disk-based XOR Engine

TAN Yu'an[1], WANG Wanxing[1], YU Qiang[1], ZHU Ligu[2], ZHANG Xuelan[1]
(1. Dept. of Computer Science and Engineering, Beijing Institute of Technology, Beijing 100081;
2. National Lab for Peripheral Storage System, Huazhong University of Science & Technology, Wuhan 430074)

【Abstract】 The SCSI commands (XDWRITE, XDREAD and XPWRITE) are already on fibre channel drives to involve the drives participating in performing RAID 5 XOR functions. For "small writes", traditional RAID-5 implementation requires a host or a RAID controller to read the original parity block, then construct the new parity block by computing the exclusive-or (XOR) of the original parity block and data blocks. the new SCSI commands have been adopted to implement the "small writes" without reading the parity block, and cause the drive to construct and update the parity block itself. By utilizing the disk's XOR engine, the throughput is increased and the average latency is also reduced.
【Key words】 RAID; Small write; Exclusive-or engine; Disk array

在 RAID 5 磁盘阵列中，由主机或 RAID 控制器将数据和校验块分布在多个物理磁盘上，在读取数据时，通过多个磁盘并行执行来提升读性能。执行写操作时，除写入数据块外，还需要根据数据块的内容重新构造和写入校验块。特别是对“小写”操作(要写入数据块的磁盘数目小于 RAID 内磁盘数目的一半)，首先要读入数据块和校验块的内容，计算出校验块的内容后，再写入数据块和校验块，效率较低。而数据库、文件系统等应用程序访问的模式包含了大量的“小写”操作，因此“小写”的性能高低对整个存储系统具有重要影响。

DCD(Disk Cache Disk)使用了智能CACHE思想，可以将多个小写请求合并成大写请求，从而有效减少磁盘阵列对磁盘的实际读写次数，提升了RAID性能[1]，然而该方法的局限是需要系统配置高速磁盘或硬件Cache。文献[2]提出的分块镜像式RAID结合了RAID 0 和RAID 1 的优点，提高了“小写”性能，但数据不再按RAID-5 格式存放，存储空间利用率比RAID-5 要低。部分SCSI及光纤磁盘实现了异或引擎，支持新的SCSI命令(XDWRITE, XDREAD, XPWRITE, XDWRITE-EXT等)，可用于构造校验块及异或数据计算[3]。文献[4,5]利用XDWRITE-EXT命令降低了“小写”操作所需的SCSI数据传输的次数，提升RAID-5“小写”性能。然而，XDWRITE-EXT命令要求校验块和数据块必须在同一SCSI或FC总线上，需要解决死锁问题，而且目前许多具有异或引擎的磁盘都没有实现XDWRITE-EXT命令。

本文采取已被广泛实现的 XDWRITE, XDREAD, XPWRITE 命令来实现“小写”操作，主机和磁盘都参与校验块的计算和写入。在提升性能的同时，还适应于各种拓扑结构配置，不存在死锁问题，能够在目前的主流磁盘上实现。

1 相关研究

1.1 “小写”操作的 RMW 实现

传统的 RAID-5 中，校验块是由主机或 RAID 控制器(以下简称主机)构造和维护的。由 m 个磁盘构成一个 RAID，向该 RAID 中写入 n 个分块时($n<m/2$)，主机需要进行以下操作：

(1)读入这 n 个分块的数据，计算新旧数据的异或值；
(2)读入校验块的数据，与(1)的结果进行异或；
(3)将 n 个分块的新数据写入，将(2)的结果写入校验块。

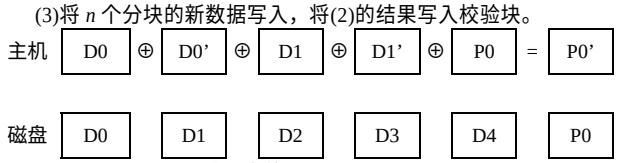

图 1 “小写”过程中的 Read-Modify-Write(RMW)

这个过程也被称作读-修改-写(read-modify-write, RMW)。图 1 表示了一个 n=2，m=5 的“小写”操作。分块中的数据为 D0~D4，校验块为 P0。写入新数据 D0'和 D1'前，主机需要先读入 D0 和 D1，计算出(D0⊕D0')⊕(D1⊕D1')，再读入 P0，计算出 P0'=(D0⊕D0')⊕(D1⊕D1')⊕P0，最后再

基金项目：国家自然科学基金资助项目(60273073)
作者简介：谭毓安(1972—)，男，博士、副教授，主研方向：存储区域网络和计算机安全；王婉星、于 强，硕士生；朱立谷，博士后、副教授；张雪兰，教授
收稿日期：2005-11-17 **E-mail：**victortan@yeah.net

写入 D0’、D1’和 P0’。

采用 RMW 方法实现“小写”，主机需要执行(n+1)次读操作，(n+1)次写操作，2n 次异或操作。某些 RAID 控制器设计了专用集成电路来执行异或操作，而一些 CPU(如 Intel i960)也提供了异或加速器以降低异或操作所需的开销。

1.2 XDWRITE-EXT 方法

在文献[4,5]中，提出了利用 XDWRITE-EXT 命令提高“小写”的效率。主机需要更新数据 D0’时，直接向磁盘发出 XDWRITE-EXT 命令，命令中包括 P0 所在的磁盘号。收到命令后，磁盘读入 D0 并利用其异或引擎计算出(D0⊕D0’)，写入 D0’。接着，磁盘再向存放 P0 的磁盘发出 XPWRITE 命令和(D0⊕D0’)。P0 磁盘收到命令后，读入 P0，与(D0⊕D0’)异或后计算出 P0’，再写入 P0’。写入 D1’时，重复上述过程。这样，写入一个数据块时，主机只需要发出一次 XDWRITE-EXT 写操作，由 2 个磁盘分别完成异或操作并更新数据块和校验块，如图 2 所示。

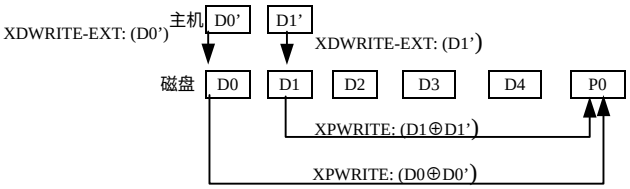

图 2 基于磁盘的 XDWRITE-EXT 方法

XDWRITE-EXT 实现“小写”时，主机只需要执行 n 次写操作，不需要执行读操作和异或操作，和 RMW 方法相比，降低了主机读入数据块以及计算、更新校验块的开销。然而，这种方法存在以下几个局限：

(1)当多个“小写”操作同时进行时，校验块分布在不同的磁盘上，这些磁盘相互之间可能形成死锁。文献[4,5]提出了解决死锁问题的方案，但降低了效率，方案复杂。

(2)RAID-5 的成员磁盘必须连接到同一个 SCSI 或 FC 总线上，存在拓扑结构的限制。

(3)XDWRITE-EXT 命令要求磁盘具有 SCSI 发起端(Initiator)的能力，向另一磁盘发出 XPWRITE 命令。许多磁盘(如 Hitachi 的 UltraStar 系列，Quantum Altas 10K 等)并没有提供对该命令的支持，不能使用该方法来发挥其异或引擎的功能。

需要特别指出，XDWRITE在SCSI标准及各种磁盘的技术手册中被正式命名为XDWRITE-EXT[4,5]。

2 基于磁盘异或引擎的均衡方法

XDWRITE-EXT 方法虽然能利用磁盘的异或引擎，除必需的数据块写入操作外，主机的负担被全部转移到磁盘上，存放校验块的磁盘的开销太大，容易形成死锁。本文将计算和更新校验块的任务均衡地分配在主机和磁盘上，克服了 XDWRITE-EXT 方法的局限。

2.1 均衡方法

目前具备异或引擎的磁盘均支持 3 个 SCSI 命令：XDWRITE，XDREAD，XPWRITE。如图 3 所示，采取下面的步骤来实现“小写”操作：

(1)主机发出 XDWRITE 命令写入新数据块到数据磁盘，由磁盘计算出新旧两个数据块的异或值，保存在磁盘的缓冲区中。

(2)主机发出 XDREAD 命令从磁盘缓冲区读入 XDWRITE 的计算结果。

(3)主机重复(1)、(2)两个步骤将所有 n 个数据块写入，对(2)读出的结果进行异或。如果 n=1 则不需此步骤。

(4)主机发出 XPWRITE 命令将(3)的内容写入校验磁盘，由磁盘读入校验块的内容后，与之进行异或后，将结果写入校验块。

使用这种方法，主机只需要执行 n 次 XDWRITE 操作，n 次 XDREAD 操作，1 次 XPWRITE 操作，n-1 次异或操作，而磁盘执行了 n+1 次异或操作，降低主机的负担。

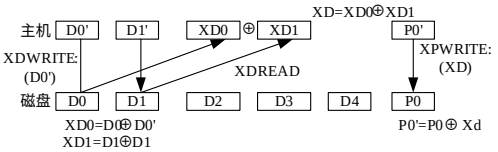

图 3 基于磁盘的均衡方法

2.2 3 种方法的比较

实现“小写”操作时，RMW 方法对主机的开销最大，但可以支持任何类型的磁盘；XDWRITE-EXT 方法最大限度地降低了主机的开销，磁盘必须具有异或引擎并实现 XDWRITE-EXT 命令；而均衡方法中主机的开销位于二者之间，磁盘必须具有异或引擎但不需实现 XDWRITE-EXT 命令。RAID-5 的成员磁盘数为 m，写入分块数为 n 时，3 种方法的比较如表 1 所示。

表 1 3 种“小写”操作实现方法的比较

实现方法	RMW 方法	XDWRITE-EXT 方法	均衡方法
主机发出的读写操作	n+1 次读、n+1 次写	n 次 XDWRITE-EXT 写	n 次 XDWRITE 写 n 次 XDREAD 读 1 次 XPWRITE 写
主机读写操作次数	2n+2	n	2n+1
磁盘发出的写操作次数	0	n 次 XPWRITE 写	0
总线上的数据传输次数	2n+2	2n	2n+1
磁盘执行的写入动作	n 次数据块 1 次校验块	n 次数据块 n 次校验块	n 次数据块 1 次校验块
磁盘写入次数	n+1	2n	n+1
主机执行的异或操作次数	2n	0	n-1
磁盘执行的异或操作次数	0	2n	n+1
磁盘类型	任何磁盘	需支持 XDWRITE-EXT 及 XPXPWRITE 命令	需支持 XDWRITE、XDREAD、XPXPWRITE

XDWRITE-EXT 方法只需在 SCSI/FC 总线上传输 2n 次，开销比 RMW 和均衡方法小。然而，它需要磁盘执行的写入次数为 2n，在 n>1 时写入磁盘所需开销最大(2n>n+1)，而存放校验块的磁盘需执行 n 次异或运算和 n 次写入，成为“小写”操作的瓶颈。

如果 $n \geq m/2$，传统的 RAID-5 实现只需要 m+1 次数据传输，低于 XDWRITE-EXT 方法和均衡方法所需传输次数，因此后 2 种方法一般只适用于 RAID-5 的“小写”操作。

3 实验结果

我们未检索到支持 XDWRITE-EXT 命令的光纤磁盘，因此仅比较了 RMW 和均衡方法。采用了 SUN StorEdge 3510 JBOD，安装了 12 个 Hitachi 的 UltraStar 15K73 光纤磁盘。主机的配置为 2.4GHz Pentium 4 CPU，内存为 1GB，安装 QLA 2342 光纤卡，卡上的 2 个端口分别连接到 JBOD 的 2 个主机通道上，运行于 2GHz。

在 Windows 中分别编写了实现 RMW 和均衡方法的程序，未设置数据块 Cache。分块的大小选择为 16kB，写入 RAID-5 分块时，构造各种 CDB(command descriptor block)命令，包括 READ(28h)、WRITE(2Ah)、XDWRITE(50h)、XDREAD(51h)、XPWRITE(52h)，使用 SCSI 直通接口(pass

through)接口向 JBOD 内的磁盘发送 CDB 命令及数据。设置了 40 个线程并行访问磁盘。

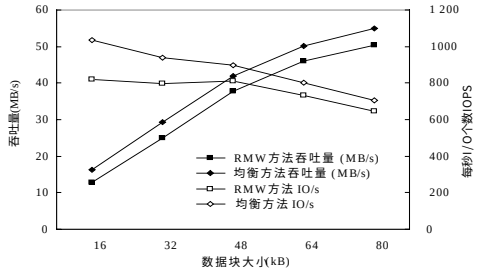

图 4 吞吐量对比

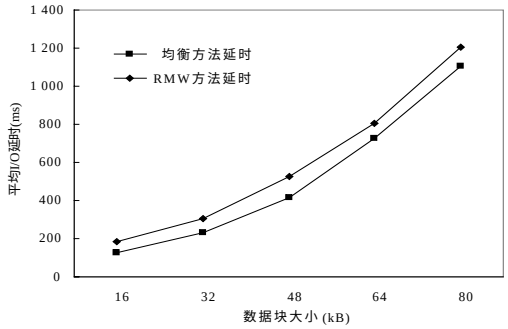

图 5 I/O 延时对比

由于 m=11，取 $1 \le n \le 5$。对每一个 n 值构造大小为 16kB～80kB 的数据写操作，写操作的块号及 n 个磁盘随机选择。程序连续运行 1 000s，根据该时间段完成写操作的次数，计算出每秒平均 I/O 数和吞吐量。记录每个写操作的完成时间，取平均值后得到 I/O 的平均延时。图 4 和图 5 显示了 RMW 和均衡方法的对比。由于均衡方法较 RWM 方法降低了总线上的数据传输次数，每一次 I/O 操作所需的传输从 $2n$+2 次降低到 $2n$+1 次，因此吞吐量有所提高，I/O 响应时间随之降低。

本文并没有修改主机操作系统或 RAID 控制器中的“小写”RAID-5 模块来实现均衡方法，实验结果应能近似地模拟该方法在实际 RAID-5 模块的作用。

4 结论

本文利用磁盘异或引擎及其扩充 SCSI 命令实现了 RAID-5“小写”操作的性能优化。和传统的读－修改－写方法相比，执行 n 个数据块写入操作时主机的读写操作从 $2n$+2 次减少为 $2n$+1 次，异或操作从 $2n$ 次降低为 n-1 次，主机的部分计算负载被转移到磁盘上。可以进一步研究该方案，使之适用于 RAID-6 双盘容错。

参考文献

1 He Xubin, Yang Qing. A DCD Filter Driver for Windows NT 4[C]. Proc. of the 12th International Conference on Computer Applications in Industry and Engineering, Atlanta, USA, 1999-11.

2 李洁琼, 冯　丹, 张　熳. 提高磁盘阵列小写性能的方法研究[J]. 计算机工程, 2003, 29(18): 57-59.

3 Houlder G, Elrod J, Miller M. XOR Commands on SCSI Disk Drives[EB/OL]. http:// www.t10.org/ftp/t10/document.94/94-111r9.pdf Accessed, 2005-06-25.

4 Chang Taisheng, David H C D. Efficient RAID Disk Scheduling on Smart Disks[C]. Proc. of the 19th IEEE Symposium on Mass Storage Systems, Maryland, USA, 2002-04.

5 Jayaram H, Torng E, Chen Y, et al. The Impact of Smart Disks and Spatial Reuse Property on RAID-5 Storage Systems[C]. Proc. of the 1998 Workshop on Architectural and OS Support for Multimedia Applications, Minnesota, 1998-08.

(上接第 206 页)

m与n之中（最小和最大项）的，因此有O(m+n)的时间复杂性。对于邻接，一个块的南方和东方向上的邻接象素的代码可以在O(n)内找到。在北方和西方向上在O(m) 内。为了搜索图像中的一块，在最坏情况下，需要$\log_2(mn)$交操作，因此，最坏时间复杂性为$O(NB \cdot \log_2(mn))$。

对于存储要求来说，当图像大小为 $2^m \times 2^m$时，只要区域的位置（x，y）有 x mod 2=y mod 2=0，即属于区域的偶数坐标，是最好的情况。很明显，在m=1 时，即对于一个 2×2 的方块，不论其位置都可获得最好的情况。

在区域位置（x，y）有 x mod 2=y mod 2=1，即在奇数坐标情况下，需要四方块的数目为：在 x和y方向上有 2^m 个像素，则

$$2^m=1+(2^m-1)=1+(1+2^1+2^2+\ldots+2^{m-2}+2^{m-1})$$

对应上面数列的每一项，都有一块大小为 1×1，1×1，2×2，…，$2^{m-2}\times 2^{m-2}$，$2^{m-1}\times 2^{m-1}$块，则有(m+1)块。

另外，还有大小为 $2^{m-1}\times 2^{m-2}$，$2^{m-1}\times 2^{m-3}$，…，即一次取 2 个项目的组合，有m(m+1)/2 个这样的组合，因此有m(m+1)个这样的块。

由此，表示一个 2m×2m 方块所需要的块的总数，在最坏情况下是(m+1)+m(m+1)=(m+1)2。相对比而言，在文献[1]方法中最坏情况下需要块的数目为 9×2m-1-2m-3。而文献[2]的 4 叉树则要求 3(2m+1-m)-5 个结点。

因此，本文所提出的方法对块的数目的要求在 m>1 时大大减少了。表 1 给出了相应的结果与对比。

表 1 一幅 256×256 大小的图像的对比

图像 256×256	黑色像素数目	需求块的数目			有效率	
	26 391	LQ	IBB	所提出的方法	超 LQ	超 IBB
		804	531	327	59.33	38.42

从表 1 可以看出，本方法对于时间复杂性和表示一幅图像的空间上（编码）有了很大的改善，这对于图像处理及其算法是重要的。

参考文献

1 Gargantini. An Effective Way to Represent Quadtrees[J]. Communications of ACM, 1982, 25(12).

2 Ouksel M A, Yagoub A. The Interpolation-based Bintree and Encoding of Binary Image[J]. CVGIP: Graphical Models and Image Processing, 1992, 54(1).

3 Samst H. The Quadtree and Related Hierarchical Data Structures[J]. ACM Computing Surveys, 1984, 16(2).

4 Bryant R E. Graph-based Algorithm for Boolean Functions Manipulation[J]. IEEE Trans. on Comput., 1986, 35(11).

5 Harmzaoui R, Saupe D. Distortion Minimization with Fast Local Search for Fractal Image Compression[J]. Journal of Visual Communication and Image Representation, 2001, 12(4).

6 范　策. 一种无前缀编码[J]. 计算机学报, 2002, 25(2).