网络磁盘阵列中基于任务排队的多用户请求动态调度策略

李洁琼 冯 丹

(华中科技大学计算机学院 武汉 430074)

摘 要 在存储系统中,底层I/O调度策略十分重要,它决定了整个存储系统的效率。一个良好的调度策略可以有效地提高系统的性能。结合网络磁盘阵列的工作特点,提出了一种基于任务排队的动态调度算法,其基本思想是充分利用多个网络用户的请求数据在网络磁盘阵列上的空间连续性,最大限度地减小磁头寻道延迟和旋转延迟,从而降低系统的响应时间。

关键词 网络磁盘阵列,动态调度算法,I/O响应时间

Queue-based Dynamic Schedule Algorithm for Multi-user Request in Network-attached Disk Array

LI Jie-qiong FNEG Dan

(Department of Computer, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract In the storage system, the bottom layer I/O schedule strategy is very important which decides the efficiency of the system. A good schedule policy could improve system performance obviously. This paper proposed an innovational solution that traces net users' requests and queues them according to their logical addresses based on the characteristics of network attached disk array. Theoretical analysis and test results show that the policy makes the access arm of the disks move regularly, decreases the disk's I/O time maximally, and reduces the average I/O response time of the system.

Keywords Net-RAID, Dynamic schedule policy, I/O response time

1 动态调度策略的提出

磁盘阵列是将多个物理磁盘虚拟成一个大的逻辑盘,当服务器对该逻辑盘进行操作时,由磁盘阵列的控制软件利用并行技术将服务器请求的I/O数据按不同的算法进行分块并交叉存放在不同的物理磁盘上,通过提高多个磁盘操作的并行性来提高I/O系统的数传率和响应时间。图1为RAID0中逻辑盘上的数据交叉存放在4个物理盘上。对于常规磁盘阵列而言(以RAID0的读命令为例),按SCSI协议,当服务器对磁盘阵列进行命令接收、命令分解、磁盘I/O、数据回送和状态回送之后,服务器才会发出下一条I/O指令,即在常规磁盘阵列的命令队列里只会有一条SCSI命令。对逻辑盘而言,每一条SCSI命令所要操作的都是一段连续的数据。从图1可以看出,这一段连续的数据分解到每一个物理盘上也必然是连续的。因此,尽管常规磁盘阵列的命令队列里的一条SCSI命令分解到每个物理磁盘上的子命令可能有多条,但由于地址连续,经过命令合并之后,也只有一条子命令,RAID5除外[1]。

基于网络磁盘阵列[2,3](以下称为Net-RAID)的存储系统在处理网络用户的文件请求时,首先由文件服务器获取文件数据空间分布的地址表信息,然后在一定的管理权限下,由网络用户按此地址表信息直接在存储设备上存取文件数据。此时,存储设备通过网络接口直接看到多个完整的文件I/O任务队列,系统的请求并发度由同时发出文件请求的网络用户数决定。

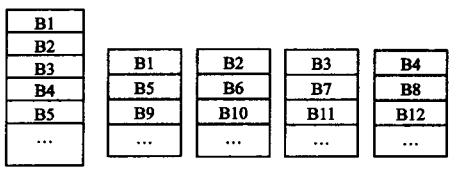

图1 RAID0逻辑盘到物理盘的地址映射

由于Net-RAID可以接收多个网络用户的请求,因此其命令队列里会有多条虚拟SCSI命令。多个网络用户请求的数据在逻辑盘上是随机分布的,那么多个用户请求的数据分解到每一个物理盘上更加没有规律。实验表明,即使对于一个连续的大的媒体文件,由于受文件服务器上文件系统的管理,在网络磁盘阵列上的分布也不是完全连续的,整个文件的地址信息往往由几十个甚至上百个命令描述,每个命令链之间的数据间隔又是随机的[4]。

文献[5]分析了网络存储的I/O负载特征,提出了完全利用请求顺序性的算法(Maximized Sequence Algorithm, MS)及综合考虑存储设备利用率和用户性能的算法(Less Sequence Algorithm, LS)[6,7]。MS算法表示存储系统按照用户

到稿日期:2008-05-14 本课题得到国家九七三重点基础研究发展规划项目基金(2004CB318300)和国家自然科学基金(60303032)资助。

李洁琼(1978-),女,博士研究生,研究方向为网络存储技术,E-mail:Jieqiong_li@yahoo.com.cn;冯 丹 女,教授,博士生导师,研究方向为网络信息存储和系统结构。

请求到达的顺序，连续完成一个用户请求所对应的所有 I/O 任务后，再服务其它用户请求所对应的 I/O 任务。MS 算法从系统管理员角度进行用户请求处理，它以设备利用率的最大化和最短时间完成尽可能多的任务量为目标。基本思路是利用用户请求所对应 I/O 任务的内在空间顺序性，顺序执行用户请求派生出的 I/O 任务，最大限度地减少磁头寻道延迟和旋转延迟，使服务于用户请求的时间最少，存储设备的执行效率最高。但由于每一个网络用户请求的数据在磁盘阵列上并不是完全连续的(尽管可能是顺序的)，那么多个用户请求的数据则有可能在磁盘阵列上交叉分布，因此该算法没有充分考虑网络磁盘阵列可以同时接收多个网络用户请求的特点，而且其对单一用户的实时响应性能也较差。LS 算法是在满足一定用户交互性能的同时，以提高设备利用率为出发点，调度各用户请求接受存储服务。磁盘阵列根据系统资源数量设定最大允许用户数，当进入系统的用户请求数到达此限额时，超限的用户请求被拒绝。对于进入系统的各用户请求，磁盘阵列轮流服务，但每次只服务用户请求对应 I/O 任务的一部分。该算法没有充分利用各网络用户请求数据在磁盘阵列上的空间位置信息，会导致磁盘的磁头臂无规则移动，因此其整体性能也不是很好。

如何利用多个网络用户请求的数据在磁盘阵列上的空间位置信息，提高存储系统的服务效率，并保证一定的用户响应性能，是网络存储设备中用户请求调度算法应权衡的问题。

2 动态调度系统的实现

由图 1 可知，在逻辑盘上按顺序分布的数据分解到单个物理盘上也是顺序的(对于多个请求不一定连续)。所以，如果能将多个网络用户的 I/O 请求进行排队，使排队后各 I/O 请求的数据在逻辑盘上按一定的顺序分布，则排队后进行磁盘 I/O，可以使磁盘的磁头臂按一定的规律移动，使移动量达到最小，降低磁盘的 I/O 时间。虽然个别的请求可能延迟，但网络磁盘阵列的整体 I/O 性能将会得到很大的提高。

磁盘中各块数据在磁盘上的物理分布顺序对系统的性能有一定的影响。假设磁盘阵列上各连续数据块所对应的盘操作任务为 $Task_1$，$Task_2$，$Task_n$，这 n 个任务有 $N!$ 种排列顺序。理论上，理想的调度策略应该是选择最佳排列顺序，使完成所有操作所花费的时间最少，即 $T=\min\{P_1,P_2,\cdots,P_{N!}\}$，其中 P_i 为几个任务的一种排列。但在实际应用中，这种调度策略是无法实现的，在磁盘的操作过程中，新的用户请求的到达有可能使事先选择的操作顺序不再是最佳的选择。动态调度策略则按各任务在磁盘阵列上的起始位置来排序，操作时依次进行，这种调度策略简洁易行，效果也很好。

动态调度算法具体描述如下。

(1) Net-RAID 的请求队列是按请求数据的每一段连续空间的第一个块号进行排队的优先队列。当向队列中添加新的请求时，会按块号顺序将该请求插入到请求队列中并根据块号的连续性将该请求与相邻请求进行合并。如果新请求的块号在当前的执行请求之前，则把该请求挂到请求队列的最后。

(2) 请求并不是严格按照块号顺序排队。为了防止请求密集时某个大块号的请求难以得到执行，每个请求设置有一个“潜伏期”变量。每次添加新的请求时，队列中各个请求项的潜伏期值不断递减。当它小于零时，后继请求就不能再插入到它的前面了。

(3) 根据系统资源的情况，请求队列中最多请求项的数量是一定的，而且是预分配的。在队列满的情况下，只有当一个请求被执行，才会接收下一个请求。执行某个请求时，不是从队列头开始，而是以当前磁头所在的位置为基准，动态选择相对移动距离最短的任务来执行。

在网络磁盘阵列系统中，网络用户根据文件服务器转发过来的数据所在的 Net-RAID 的 IP 地址、接收端口以及数据的逻辑地址链和数据长度等信息同 Net-RAID 进行交互。所涉及到的信息包括头信息数据结构 Header 和地址表数据结构 Cmd，如图 2 所示。

```
type struct{
ulong len;//文件数据分布的地址表个数，一段连续存储空间用一个 cmd 结构表示，
          //len 即为连续存储空间个数
ulong ip;//目标网络磁盘阵列的 IP 地址，通过 IP 地址区分不同的网络磁盘阵列
int port;//目标网络磁盘阵列的服务端口，
         //凭此(IP,PORT)信息用户建立与网络磁盘阵列的连接
int filelen;//若文件服务器接受网络用户的读写请示，则返回文件长度信息，
            //否则返回错误原因代码
int magic;//返回给用户的权限字
}Header;
type struct{
unsigned int firstblock;//连续存储空间的起始地址
int count;//连续存储空间的大小，以 Kbyte 为单位
}Cmd;
```

图 2 用户与网络磁盘阵列交互信息的数据结构

网络用户通过 Header 数据结构判断请求是否已被文件服务器接受，并根据其成员域 len 分配文件数据分布地址的缓冲区。文件服务器发送的地址表信息为一组 Cmd 数据结构，每个 Cmd 数据结构表示文件在磁盘上占用的一段连续空间。不同的 Cmd 数据结构所表示的磁盘空间是不相交的。

由于 Net-RAID 的网络接口模块主要处理和网络用户的通讯，动态调度算法的基本思想就是在网络接口模块中跟踪网络用户的请求，并按网络用户请求的数据的逻辑地址进行排队。由上面的分析可知，我们仅需要跟踪 Cmd 结构的 firstblock 成员域即可实现该算法。

表 1 是简化的动态调度策略前后系统状态的比较。系统中共有 10 个 I/O 任务，每个任务表示磁盘阵列上的一段连续空间，它们互不相交。表 1 的左边表示未采用动态调度策略之前，系统采用先来先服务策略，可以看到每一个 I/O 任务的数据的起始地址(Cmd->firstblock)在磁盘阵列上的逻辑空间是不连续的，必然导致各磁盘磁头不规则移动。表 1 的右边表示采用动态调度策略后，以任务 1 的磁头所在位置为基准，各 I/O 任务的起始地址按升序排列，从而使磁盘的磁头按一定的规则移动，降低了磁头的定位时间。同时，由于任务 1 和任务 8、任务 2 和任务 7、任务 5 和任务 10 经过重新排序后，其请求的数据块在磁盘阵列上是相邻的，因此对它们进行了合并，使 I/O 任务数从 10 个减少为 7 个，减少了磁头的定位次数。由于磁头定位时间和定位次数的减少，必然会导致系统性能的提高。

表 1　简化的动态调度策略前后系统状态比较

动态调度之前				动态调度之后		
任务序号	Cmd->firstblock	Cmd->count		任务序号	Cmd->firstblock	Cmd->count
10	18	3		6	22	5
9	4	1		10	17	4
8	2	1	排序	5		
7	7	1	→	4	14	1
6	22	5		3	9	3
5	17	1		7	6	2
4	14	1		2		
3	9	3		9	4	1
2	6	1		8	1	2
1	1	1		1		

3　动态调度系统的性能分析

假设同时有 n 个不同的网络用户对 n 个不同文件有读写请求，这些请求都要经过文件服务器的预处理，形成对网络磁盘阵列的读写请求。每个请求都占用一小块连续的磁盘空间，可以用一个 Cmd 数据结构描述。

第 i 次 I/O 任务的执行时间均可表示为 $t_i = \mathrm{Seek}_i + \mathrm{Rotate}_i + \mathrm{Transfer}_i$，$\mathrm{Seek}_i$ 和 Rotate_i 分别为每次 I/O 任务相应的寻道和旋转延迟，$\mathrm{Transfer}_i$ 则为一次 I/O 任务的数据传输时延。

通常情况下，总共的 I/O 服务时间 $T_{\mathrm{normal}} = \sum_{i=1}^{n}(\mathrm{Seek}_i + \mathrm{Rotate}_i + \mathrm{Transfer}_i)$，其中第一项为总共的寻道时间，第二项为总共的旋转等待时间，第三项为总共的数据传输时间[8]。若系统采用动态调度策略，总的 I/O 服务时间为 $T_{\mathrm{dynamic}} = \sum_{j=1}^{n}(\mathrm{Seek}_j + \mathrm{Rotate}_j + \mathrm{Transfer}_j)$。

如果相邻的请求之间存在较大的寻道和旋转延时，通常情况下，会导致磁头臂无规则移动。而动态调度策略以当前磁头所在的位置为基准，通过任务排队，动态选择相对移动距离最短的任务来执行。由于各个 I/O 任务请求数据的空间连续性，可使磁头寻道时间和旋转延时减到最少，I/O 执行效率最大，有：$\sum_{i=1}^{n}(\mathrm{Seek}_i + \mathrm{Rotate}_i) >> \sum_{j=1}^{n}(\mathrm{Seek}_j + \mathrm{Rotate}_j)$，由于 $\sum_{i=1}^{n}\mathrm{Transfer}_i = \sum_{j=1}^{n}\mathrm{Transfer}_j$，因此有：$T_{\mathrm{normal}} >> T_{\mathrm{dynamic}}$。

显然，网络用户请求的数目越多（即 n 越大），请求的数据在网络磁盘阵列上分布得越离散，排队后的 I/O 性能提高越显著。

对于磁盘 I/O 而言，磁头定位时间（磁头寻道时间和旋转时间）相对较长，减少磁头的定位时间可以显著改善性能。由于磁头定位时间与磁头当前位置和请求访问的磁盘位置有关，动态调度策略充分利用了网络磁盘阵列可以同时接收多个网络用户请求的特点，跟踪各个网络用户请求的子 I/O 任务的地址信息，并按一定的策略排队，从而极大地降低了磁头定位时间，提高了系统的整体性能。而且，该策略还增大了对地址连续的请求进行合并的概率，减少了磁头定位次数，进一步提高了系统性能。

4　性能测试

表 2 是在相同测试环境下，Net-RAID 在未采用动态调度策略和采用动态调度策略两种情况下的性能对比。Net-RAID 设为 0 级。测试读时，N 个进程分别读取离散分布在网络磁盘阵列上的 N 个大小为 57Mbyte 的文件；测试写时，N 个进程分在网络磁盘阵列上 N 个离散区域写进大小为 57Mbyte 的文件。

表 2　改进策略和传统策略的性能测试比较

操作类型	文件大小	用户数	未采用动态调度算法		采用动态调度算法	
			服务时间 (s)	集合带宽 (MB/s)	服务时间 (s)	集合带宽 (MB/s)
写	57MB	3	33.7	3.27	30.2	3.51
		6	76.5	4.15	65.7	4.48
		9	90.3	4.76	72.7	5.14
		12	104.8	5.82	81.6	6.61
读	57MB	3	23.3	3.21	21.1	3.78
		6	43.7	4.30	35.1	4.98
		9	65.2	5.22	50.1	5.73
		12	84.9	6.36	62.3	7.28

由数据可以看到，采用动态调度策略之后系统的 I/O 服务时间和集合带宽明显提高，而且提高的幅度随网络用户数目的增加而增加。这是因为网络用户的数目越多，用户请求的数据在磁盘阵列上分布的随机性越大，排队后的效果越明显。该结论显然和前面理论分析得出的结果相一致。

结束语　网络磁盘阵列中基于任务排队的用户请求动态调度策略通过对 Net-RAID 命令队列中多个请求的数据按逻辑地址排队，并且对地址连续的请求进行合并，使得磁盘磁头臂按规律移动，最大限度地减少了移动量，降低了磁盘的 I/O 时间。虽然个别用户请求可能会延迟，但系统的整体 I/O 性能得到了很大的提高。

参 考 文 献

[1]　陈琼，张江陵，冯丹. 一种提高磁盘阵列 I/O 性能的策略[J]. 小型微型计算机系统，2000(1)：13-15

[2]　李洁琼，冯丹. 一种基于网络磁盘阵列的高性能海量存储系统[J]. 小型微型计算机系统，2006，27(12)

[3]　王芳. 网络磁盘阵列系统的研究[D]. 武汉：华中科技大学图书馆，2001

[4]　Zhang Jiangling, Deng Yuhui, Feng Dan, et al. Design and Implementation of a Mass Storage System with Scalable Bandwidth and Capacity // Proceeding of 7th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2003). Orlando, Florida, U. S. A

[5]　Collins B, McLarty T. Mass storage systems reference model system management // Proc. of 9th IEEE Symposium on Mass Storage Systems. Digest of Papers, 1998: 29-34

[6]　Worthington B L, Ganger G R. Scheduling Algorithms for Modern Disk Drives // Proc. of the ACM Sigmetrics Conference. May 1994: 241-251

[7]　Jacobson D M, Wilks J. Disk Scheduling algorithms Based on Rotational Position. Hewlett-Packed Technical Report, HPL-CSP-91-7. Feb. 1991, 26

[8]　Chen Qiong, Zhang Jiangling. Analyse for the I/O serving time of the high performance disk array. Chinese Computer Systems, 2000, 21(3): 235-237