第 35 卷 第 4 期 计 算 机 工 程 2009 年 2 月
Vol.35 No.4 Computer Engineering February 2009
·软件技术与数据库· 文章编号：1000—3428(2009)04—0053—03 文献标识码：A 中图分类号：TP302

一种 SAN 上的数据迁移算法设计与实现

曾雷杰，张延园，李战怀，赵晓南
(西北工业大学计算机学院，西安 710072)

摘 要：业务执行繁忙和资源分配不均可能降低个别磁盘池的 I/O 响应速度，影响请求的正常响应。该文通过在不同磁盘池之间进行逻辑磁盘(LD)移动处理，利用空闲磁盘池中的容量做成一个预约 LD，在空闲时间后台并行进行数据迁移，使存储系统的 IOPS 提高了 8%左右。通过交换 LD 元数据信息，可以避免外部 Mount 点的变化。

关键词：性能瓶颈；磁盘池；逻辑磁盘移动

Design and Implementation of Data Migration Algorithm on SAN

ZENG Lei-jie, ZHANG Yan-yuan, LI Zhan-huai, ZHAO Xiao-nan
(School of Computer, Northwestern Polytechnical University, Xi'an 710072)

【Abstract】 Because of the busy ratio of operation and uneven distribution of resources, the individual disk pool may be led to decrease the I/O speed of response and the normal response to the request is affected. By the Logical Disk(LD) migration processing between disk pools, a reserved LD can created by using the spare capacity. By parallel data migration on idle time background, the IOPS of storage system can be around increased by 8%. Through the exchange of LD metadata information, it is possible to avoid an external point of the Mount.

【Key words】 performance bottleneck; Disk Pool(DP); Logical Disk(LD) move

1 概述

近年来，FC-SAN 磁盘阵列的物理磁盘容量不断扩大，已出现 300 GB 的磁盘。为了使磁盘系统在个别磁盘发生故障时能继续进行访问，研究者采用 RAID 技术[1]和 Spare 技术，按 RAID 的类型将物理磁盘(Physical Disk, PD)组成磁盘池(Disk Pool, DP)，并从中构筑逻辑磁盘(Logical Disk, LD)，供多个服务使用，从而提高数据可用性和可靠性。当多个业务服务器共享存储在 LD 上的数据时，可能使特定 PD 上的 I/O 过分集中，形成瓶颈，导致 I/O 响应速度降低。

针对上述情况，应进行数据迁移以调整数据组织方式，使更新的数据均匀分布，提高数据并行访问能力，从而使 I/O 走向 I/O 不太集中的 PD 上。由于归档或备份等原因，在早期存储管理中，数据迁移已经是一项重要任务。为了保障迁移数据的一致性，管理员通常需要暂停应用程序对 LD 的访问来完成数据迁移任务。但现有存储服务要求常年不间断连续执行，为实现数据迁移而暂停服务的执行对使用者来说是不可接受的。因此，多数现有存储管理软件都能在数据不间断的情况下，实现在数据迁移过程中不间断的服务，例如 HP 的 LVM、Veritas 的 VxVM 等。能提供连续性存储访问的数据迁移称为在线迁移，需要暂停服务执行的数据迁移称为离线迁移。在线迁移技术的基本思想如下：使用镜像技术在迁移的目标位置上创建镜像数据，保证迁移过程中迁移数据的一致性，当数据保持一致后，断开它们之间的镜像关系，让镜像 LD 代替数据 LD 提供存储服务。

对上述在线迁移而言，由于镜像 LD 代替数据 LD 提供服务，使文件系统的访问点发生变化，即外部接口发生变化，因此要求外部执行程序的配合。

本文根据所监视磁盘数据性能分析的结果，设置一定阈值，在空闲时保持外部接口的不变性并完成数据的在线自动迁移，从而实现性能优化和负载均衡，提高了存储管理的自动化能力，减少了存储管理人员的介入且降低了管理成本[2]。

2 存储系统的相关概念

在以磁盘为介质的大型存储系统中，为了满足高性能、高可靠和高可用要求，通常采用 RAID 管理多个 PD，形成一个大容量 DP，作为构成 LD 的基础。LD 是操作系统等可以通过 Mount 访问的存储介质对象，它依据用户要求从 DP 中构筑。PD 和 LD 间的映射关系由 DP 管理，需要事先保留部分物理磁盘空间来存储上述关系。PD-DP-LD 的关系如图 1 所示。

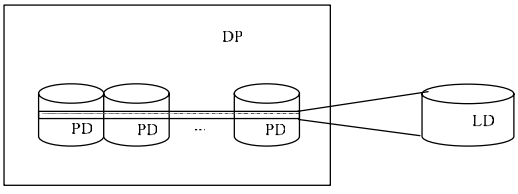

图 1 PD-DP-LD 的关系

性能优化过程通常是对当前工作状态下，可能成为瓶颈的各个因素进行监视，并根据采集的数据信息进行具体分析，找到瓶颈点后采取数据迁移等策略加以改善。因此，必须预先进行性能监视，表 1 列出了具有代表性的性能参数。

基金项目：国家自然科学基金资助项目“基于模式的高可扩展性 P2P 数据管理技术的研究”(60573096)

作者简介：曾雷杰(1977－)，男，博士研究生，主研方向：海量数据存储，软件工程；张延园、李战怀，教授；赵晓南，博士研究生

收稿日期：2008-07-10 **E-mail：**zenglj@nwpu.edu.cn

表 1　性能参数与相应的原始数据

性能参数名称	参数构成对象	原始数据
外部 I/O 密度	LD	读/写数据次数；时间
外部传送速率		读/写数据长度；时间
外部平均传送长		读/写数据长度及次数
外部平均应答时间		读/写应答时间及次数
内部 I/O 密度	PD/DP	读/写数据次数；时间
内部传送速率		读/写数据长度；时间
内部平均传送长		读/写数据长度及次数
内部平均应答时间		读/写应答时间及次数
繁忙率	PD/LD/DP	繁忙时间；监视时间 读/写应答时间(对 LD)

3　数据迁移算法的设计与实现

本文提出的数据迁移算法如下：当存储管理软件监视到某个 DP 中的 I/O 繁忙(称为忙 DP)，而其他某个 DP 空闲(称为闲 DP)时，分析并计算监视的数据，找出导致 DP 繁忙的关键对象 LD，并进行“LD 移动”，从而把忙 LD 的数据迁移到闲 DP 中的预约 LD 中(若预约 LD 没有被构筑，则可以调用 API 动态生成)。在数据完全迁移后，交换上述 2 个 LD 的元数据信息，包括 LD 的名字、LD 番号、LD-DP-PD 的映射关系等。此时，虽然数据的映射关系等元数据信息已发生变化，但从应用程序的角度来看，操作对象 LD 的接口没有任何变化。实现了无须暂停应用程序的执行，并消减 I/O 瓶颈的目的。

需要对 PD, DP 和 LD 对象进行监视和数据分析。监视的数据主要包括 PD 繁忙率、DP 繁忙率和 LD 繁忙率。

PD 繁忙率，即 PD 上 I/O 动作时间与监视时间的比率为

$$pd\_busyrate = \frac{pd\_busytime}{mon\_time} \tag{1}$$

其中，*mon_time* 为监视时间。

DP 繁忙率为

$$pool\_busyrate = \frac{1}{pd\_num}\sum_{i=0}^{pd\_num-1} pd\_busyrate[i] \tag{2}$$

其中，*pd_num* 是 DP 中的 PD 个数。

LD 繁忙率为

$$ld\_busyrate[i] = pool\_busyrate \times \frac{rw\_restime[i]}{\sum_{j=0}^{ld\_num-1} rw\_restime[j]} \tag{3}$$

其中，*rw_restime* 表示读写应答时间。

由式(1)~式(3)可知

$$ld\_busyrate[i] = \frac{rw\_restime[i]}{mon\_time} \times \frac{\sum_{k=0}^{pd\_num-1} pd\_busytime[k]}{\sum_{j=0}^{ld\_num-1} rw\_restime[j]} \tag{4}$$

由式(4)可以看出，若 DP 中 LD 的读写操作频繁，则 DP 读写时间较长，将导致构成它的各个 PD 的繁忙率较大。因此，对成为性能瓶颈的 LD 进行数据迁移，可以消除 PD 产生的性能瓶颈，本文算法以此为理论基础。

为了有效进行数据迁移，必须根据操作对象的容量和实际业务要求进行资源预约，从而在数据迁移时节省资源的分配和初始化等时间。资源预约是指从 DP 中构筑一个特殊逻辑磁盘 LD，赋予特定属性，使它不能直接被文件系统访问，即用户不可直接访问。本文称上述 LD 为预约 LD。采用静态事先预约的方法会占用一定容量空间，但可以在进行数据迁移时减少总体迁移时间。本系统提供了一个 API 实现预约 LD 的动态生成，实现操作自动化，不再借助管理者手动操作。

定义(LD 移动) 把对象 LD 的数据迁移到等容量且 RAID 类型一致但不在同一个 DP 的预约 LD 中，并在数据迁移完毕后，交换 2 个 LD 的元数据信息。

由定义可知，LD 移动分为 2 个阶段：(1)把对象 LD 的数据全部迁移到预约 LD 中，在此期间该操作是可以停止的；(2)对上述 2 个 LD 的元数据信息进行交换(无须修改用户接口)，实现了数据的迁移和 LD 的交换。元数据信息主要包括 LD 的名字、编号、LD-DP-PD 映射关系等。LD 元数据信息交换过程是不可中断的，否则会导致 LD 元数据信息的不一致，即该阶段的操作必须是原子的。

在 LD 移动的第 1 个阶段，数据的迁移时间可能较长，而暂停应用程序的执行对用户来说是不可接受的，为了继续处理业务的执行，一般采用如下 2 种解决方式：

(1)用足够量的 RAM 缓存 LD 迁移过程中更新的数据，待 LD 移动完毕后再更新到对应的 LD 上。此方式通过增加内存容量来减缓数据迁移造成的性能降低，需要考虑 2 个问题：1)系统中的 RAM 容量有限，在业务较繁忙时，RAM 会很快被用完，导致 I/O 响应停止；2)突然断电等情况会导致 DRAM 中保存的本来更新成功的数据发生丢失，造成数据不一致。由于 DRAM 价格不断下降，通过配备足够大的内存解决问题 1)是可行的。为了解决问题 2)，一般采用 NVRAM 替换 DRAM，以避免数据的损失和数据不一致，但 NVRAM 比 DRAM 贵很多。

(2)采用事务处理的思想进行在线更新。对于在 LD 移动过程中发送来的 I/O 请求，按如下控制方式进行处理，不需要很多 RAM，但会影响正常 I/O 的响应速率：

1)在 LD 移动开始时，维持一个对象 LD 块迁移的位图信息，全部初始化为 0。每个块的位图信息有 2 个值：0 代表该块数据没有迁移到预约 LD 中，1 代表该块数据已迁移到预约 LD 对应的块上。

2)在 LD 移动过程中根据块号的次序依次进行数据拷贝。若在移动过程中出现 I/O 写请求，则采用如下算法：

①查看指定块的迁移位图信息，若为 0，则说明该块数据没有迁移，只要直接写入对象 LD 中即可。

②若指定块的迁移位图信息为 1，就说明该块数据已迁移到预约 LD 中。此时，先把该块的数据写入预约 LD 中，再写入对象 LD 中，若 2 个写操作都成功，则表示整个写操作成功，否则写操作失败。写操作失败表示 LD 迁移处于 LD 移动故障状态，无须继续迁移。

I/O 读操作的实现方式是从源 LD 中直接取，其性能基本不会受到影响。

本系统采用第(2)种方式实现数据更新和一致性维护。

为了有效地管理 LD 移动的状况，定义了若干 LD 移动的状态如未执行、移动中、交换中、移动中断和移动故障等，通过状态来反映 LD 移动的过程。LD 移动的状态如表 2 所示。

表 2　LD 移动的状态

状态	描述
未执行	LD 移动开始前的状态/迁移完毕状态
移动中	LD 移动的第 1 个阶段进行中
交换中	LD 移动的第 2 个阶段进行中
停止	在移动中执行停止后的状态
停止(故障)	出现故障而导致 LD 移动停止的状态

本文提供了 5 个 API 和命令，具体如下：

(1)LD 移动的开始(Start LD Move)，实现从对象 LD 到预约 LD 的 LD 移动过程。

(2)LD 移动的停止(Stop LD Move)，停止 LD 移动的处理。该 API 只能在 LD 移动中数据的第 1 阶段进行，在数据拷贝完后，即 Meta 属性信息的交换阶段是不能停止的。对曾被停

止过的 LD，若再次执行 LD 移动，就会把前次已拷贝的数据变为无效，需要重新进行全拷贝。

(3)LD 移动的速度变更(Change LD Move Mode)，在 LD 移动过程中改变 LD 移动的模式，主要是为了修改数据拷贝的速度，减少 LD 移动的时间。系统会根据当前负荷的繁忙状况，以优先响应 I/O 执行为第 1 要务，适当并自动地调整数据拷贝速度，保证业务性能没有太大降低，并充分利用空闲时间尽快完成 LD 移动。

(4)LD 移动的强制终了(Terminate LD Move)，当 LD 移动出现故障，导致 LD 移动状态处于停止(故障)时，通过该 API 使 LD 移动状态迁移到未执行状态，恢复到 LD 移动前的状态。

(5)LD 移动状态的取得(Sense LD Move Status)，取得 LD 移动的状态(完了/中断/故障/实行中)，拷贝速度、已拷贝的数据量等。

通过上述 API 和命令可以实现 LD 移动和状态监视，LD 移动的状态迁移如图 2 所示。

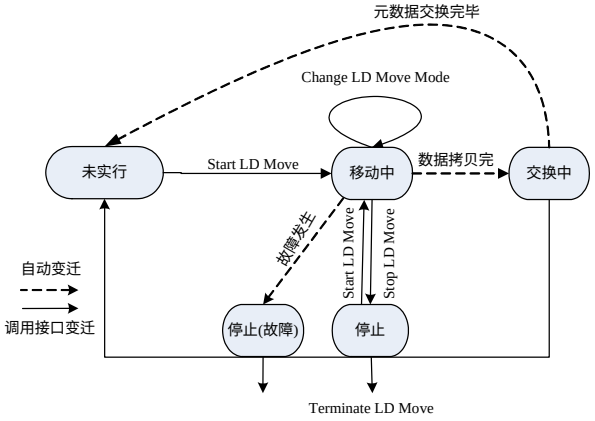

图 2　LD 移动的状态迁移

对于 LD 移动的状态监视，可以通过定期对系统发送 API 来取得当前 LD 移动的状况，进而在 LD 移动完毕或出现故障时，及时通知用户进行 LD 的构成变化或进入故障状态。但如果发行 API 的密度太大，就会影响正常 I/O 响应，如果发行 API 的密度太小，则起不到应有的状态监视功能，对于本系统而言，定期监视的间隔缺省为 10 s。

4　实验和评估

本文测试采用 NEC S2100 中的 2 个物理磁盘，按 RAID0 配置成一个 DP，称为源 DP，构筑一个容量为 1 GB 的逻辑磁盘 LD，包括 LD1 和 LD2。用 2 个物理磁盘按 RAID0 配置成另外一个 DP，称为目的 DP，构筑 2 个容量为 1 GB 的逻辑磁盘 LD，包括 LD3 和 LD4，其中，LD4 作为预约 LD。实验中的存储负荷由 Iometer[3]产生，LD1 和 LD2 的访问模式为 64 KB 大小的纯随机访问(包括 67%的读操作和 33%的写操作)。在 Iometer 中，设置 LD 未完成访问请求的最大数目为 5，并用参数 *delay time*(相邻 2 个访问请求的间隔时间)来调节存储负荷的强度。

在正常条件和 LD 移动条件下分别进行测试，结果如表 3 所示。可以发现，当 LD1 和 LD2 高负荷工作时，LD4 低负荷；当 LD4 作为 LD1 数据迁移时，LD1 的 IOPS 下降 3%。但在 LD 移动结束后，LD2 的 IOPS 提高了 8.7%。

表 3　正常条件和 LD 移动条件下的测试结果

条件	逻辑磁盘	*delay time*/ms	IOPS/次	平均响应时间/ms	迁移时间/ms	IOPS 的变化比例/(%)
正常	LD1	15	51.56	21.15	-	-
	LD2	20	30.34	29.39		
移动	LD1	15	50.03	24.72	1 169	-3.0
	LD2	20	28.74	59.27		+8.7

5　结束语

本文利用空闲容量达到分散 I/O 的目的，提高了系统整体性能，但没有实现自动执行对 I/O 瓶颈的预测。下一步工作将研究自动预测模型，以提高 I/O 性能。

参考文献

[1] Chen P M, Lee E K, Gibson G A, et al. RAID: High-performance, Reliable Secondary Storage[J]. ACM Computing Surveys, 1994, 26(2): 145-185.

[2] Gartner Group. Total Cost of storage Ownership——A User Oriented Approach[Z]. 2000.

[3] Intel Corporation. Iometer Project[Z]. (2008-09-26). http://www.iometer.org/.

(上接第 52 页)

CVE 系统的一致性、响应性和并发性得到了较好的平衡[10]。

4　结束语

本文对协同虚拟环境中的并发控制机制进行了概括和分析，比较了悲观并发控制机制和乐观并发控制机制的优缺点。在此基础上，提出一种基于客户端的并发控制模型，并予以设计和实现。由于一致性是 CVE 系统中保证协同的基础和关键，下一步工作是研究共享物体的状态信息在服务器和协作客户端之间的同步更新策略及算法，从而减少不必要的操作撤消，使用户在 CVE 系统中能更加自然、方便和高效地开展协作。

参考文献

[1] 陈　红. 协同虚拟环境一致性研究[D]. 杭州: 浙江大学, 2005.

[2] 潘志庚, 姜晓红, 张明敏, 等. 分布式虚拟环境综述[J]. 软件学报, 2000, 11(4): 461-467.

[3] 宋　渝. 协同虚拟环境的研究和设计[D]. 重庆: 重庆大学, 2006.

[4] Sung Un-Jae, Yang Jae-Heon, Wohn Kwang-Yun. Concurrency Control in CIAO[C]//Proc. of IEEE Virtual Reality Conference. Houston, USA: IEEE Computer Society, 1999.

[5] 余春艳, 庄越挺, 潘云鹤. 基于角色的协同虚拟环境并发控制投机策略[J]. 浙江大学学报: 工学版, 2004, 38(6): 658-664.

[6] Roberts D, Wolff R. Controlling Consistency Within Collaborative Virtual Environments[C]//Proceeding of the 8th IEEE International Symposium on Distributed Simulation and Real-time Application. Budapest, Hungarypp: [s. n.], 2004.

[7] Lam K W, Lam K Y. Optimistic Concurrency Control Protocol for Real-time Databases[J]. Journal of Systems and Software, 1997, 38(2): 119-131.

[8] Hu Xiaorong, Hua Ming. Role-based Concurrency Control Policy in CSCW[J]. Journal of Computer Era, 2004, 10(3): 30-32.

[9] Lin Qingping, Low C P. Multiuser Collaborative Work in Virtual Environment Based CASE Tool[J]. Journal of Information and Software Technology, 2003, 45(5): 253-267.

[10] Song Xiaohui, Liu J W S. Performance of Multiversion Concurrency Control Algorithms in Maintaining Temporal Consistency[C]//Proc. of the 14th IEEE Ann. Int'l Computer Software & Application Conference. Chicago, USA: [s. n.], 1990.