动态盘阵 D/H 分布与基于控制理论的在线重构*

刘 军+, 杨学军, 王俊伟, 唐玉华

(国防科学技术大学 计算机学院,湖南 长沙 410073)

D/H Placement and On-Line Data Reorganization Based on Control Theory in Dynamic Disk Array

LIU Jun+, YANG Xue-Jun, WANG Jun-Wei, TANG Yu-Hua

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573679, E-mail: muplj@tom.com, http://www.nudt.edu.cn

Received 2004-01-29; Accepted 2004-05-08

Liu J, Yang XJ, Wang JW, Tang YH. D/H placement and on-line data reorganization based on control theory in dynamic disk array. *Journal of Software*, 2005,16(5):1028 - 1038. DOI: 10.1360/jos161028

Abstract: Disk Array is adopted widely because of its high performance I/O. To adapt the need of applications' changeable I/O performance, I/O storage subsystem should be highly scalable. So DDA (dynamic disk array), which can scale adaptively, is an ideal system. The key technology of DDA is its data placement algorithm and online data reorganization algorithm. The main contribution of the paper is: first, a detailed study on DDA data placement is conducted and a new placement method, D/H, is presented. In D/H placement, the space in DDA is balanced after scale, and the reorganization cost is minimized; then, an online data reorganization algorithm based on control feedback theory is provided. With this strategy, the reorganization in DDA does little impression to the system QoS, and under this condition, data reorganization can be accomplished as quickly as possible; finally, simulation results show that Online Data Reorganization based on Control Theory is useful.

Key words: disk array; scaling dynamically; dynamic disk array; data placement; algorithm; on-line reorganization; control theory

摘 要: 由于能够提供高性能 I/O,盘阵被广泛采用.但以往的盘阵扩展性不足.而用户或应用程序对外存容量和I/O性能需求是变化的,盘阵系统本身必须有很强的扩展性,以适应系统的I/O需求.因此,由于既具有盘阵的高性能I/O,又能通过增加或减去设备后进行数据重构实现性能的扩展,动态盘阵具有广泛的前景.动态盘阵的技术热点是数据分布算法和在线自适应数据重构技术,使得盘阵的性能和容量能够随着系统的扩展而伸缩,同时使得盘阵动态扩展时的数据重构对系统的影响非常小.主要工作是:第 1,对动态盘阵的数据分布展开研究,并提出一种新的数据分布算法(D/H 分布).在 D/H 分布中,盘阵扩展时始终保持各设备上空间和负载的平衡性,同时扩展时重构的数据最少;第 2,针对 D/H 分布,提出基于控制理论的数据重构技术,使得盘阵动态扩展时的在线数据

* Supported by the National Science Fund for Distinguished Young Scholars under Grant No.69825104 (国家杰出青年科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA1Z2101 (国家高技术研究发展计划(863))

作者简介: 刘军(1971－),男,湖南安化人,博士,主要研究领域为并行 IO;杨学军(1963－),男,教授,博士生导师,主要研究领域为高性能计算;王俊伟(1975－),男,博士生,主要研究领域为并行计算;唐玉华(1962－),女,研究员,主要研究领域为计算机网络与通信.

重构对请求 QoS 的影响非常小,同时使得数据重构能够尽快完成;第 3,研究中针对 Sprite trace 和合成负载进行了大量模拟实验,结果表明,提出的基于控制理论的数据重构技术行之有效.

关键词: 盘阵;动态扩展;动态盘阵;数据分布;算法;在线重构;控制理论

中图法分类号: TP333 **文献标识码:** A

随着 CPU 和内存等微电子部件性能的迅猛提高,磁盘 I/O 成为计算机系统的主要瓶颈之一.因此作为一种高性能的 I/O 子系统,盘阵被广泛应用.

但是以往的盘阵大都是一种静态的,也就是说,其规模和容量、性能在系统构建之初就已经确定,不能随需求的变化而动态扩展.而在实际的需求中,一方面,不同的应用,I/O 需求不同.比如有的科学计算应用 I/O 需求巨量[11],而有的应用 I/O 需求低一些;有的情况下,即使同种类型的应用程序,在不同的时间段运行,其 I/O 需求差别达到一个数量级;另一方面,有的盘阵系统本身规模也是变化的.比如在集群分布式系统中,在将各结点的外设构成虚拟盘阵的 I/O 子系统时,由于分布式环境是一种变化的系统,经常有结点的加入或退出[2,3],从而导致虚拟盘阵规模的变化,因此,为了满足这些应用程序的需要或适应系统环境的变化,盘阵应该是可动态扩展的系统,也就是动态盘阵[4].

动态盘阵是由磁盘外设构成的并发、可靠、可自适应扩展的 I/O 子系统.其并发性是指,一个 I/O 能够通过多个设备并发完成;可靠性是指,采用信息冗余技术进行纠错,保障数据的可用性和可靠性;动态扩展性是指,盘阵能够随着系统 I/O 需求的变化,通过加载或减去存储设备,自适应扩展(扩大或缩小)存储系统的容量和性能.其中,动态扩展性是动态盘阵中很重要的一个特征.

在技术上,一方面,对于动态盘阵,当由最初的 N_0 个磁盘设备扩张到 N_1 个设备时,只有选择很好的数据分布算法,并根据该算法进行数据重构,使得扩展后各设备上负载保持一种平衡性,才能保持盘阵的性能随着盘阵的扩展而扩展.同时,数据分布算法必须使动态数据重构的移动负载量最小.因此,动态盘阵的数据分布算法是一个研究热点.近年来在这些方面有大量的研究,比如,round-robin 分布[5]、SCADDAR 算法[6]和行置换数据重构 RPDR[3]等,但这些分布都不能使盘阵设备间的空间平衡性和重构开销达到最优.本文研究中发现,如果设备的扩展方式是成倍或减半(Double/Halve)的 D/H 方式,则可以找到很理想的分布算法——D/H 算法.

另一方面,提高盘阵数据重构性能是一个很重要的问题.所谓重构是指,根据数据分布算法,将盘阵原来的数据块在扩展后的盘阵上重新分布.只有通过重构,动态盘阵才能使各设备上空间保持一种平衡,才能利用动态扩展的设备性能.动态盘阵数据重构的尽快完成对系统性能有很大的意义.

数据重构可以有离线(off-line)与在线(on-line)两种方式[5].所谓离线重构是指,数据重构时占有所有系统资源,直到重构完成.离线方式中,数据重构的过程中系统不能服务任何其他的客户请求;而在线重构是指,在数据重构的过程中还能服务系统的其他 I/O 请求.

离线重构会引起业务中断,因此许多应用中在线重构非常重要.商业领域中的许多应用,如航空售票系统、视频点播和移动梦网企业服务器等一般与人们生活、生命财产和国民经济密切相关,应用中数据库存储系统服务能力直接关系到系统的服务质量 QoS(quality of service)、人们的生活质量与公司企业的经济收入;而有的应用,如医疗卫生信息系统[5],进行紧急抢救时,许多医生在不同的医院都需要查看系统信息,此时动态扩展时,不能将系统离线,终止当前服务而进行数据重构.

许多研究都已经意识到了在线数据重构的重要性[5,7],文献[7]还专门分析了在线数据迁移与 QoS 问题,这些算法并不是针对动态盘阵展开的.虽然文献[5]针对盘阵扩张进行了研究,但该研究没有针对请求的 QoS 展开.我们认为,很好的在线数据重构算法,不仅能使数据重构尽快完成,而且数据重构对当前 I/O 请求的 QoS 影响非常小;并且作为动态盘阵的重构来说,重构的速度是不断加快的.这是因为,随着重构的进行,扩展增加的设备能够承担一部分负载,原来设备上的负载会越来越轻,从而设备能用来进行重构的资源也越多.本研究将服务器的 QoS 和设备上负载情况、在线重构速度等多方面同时考虑,提出了基于控制理论的方法来研究动态盘阵的在线数据重构问题,将当前重构与 QoS 情况作为反馈,不断控制和调整重构速度.

本文研究的主要工作和创新是:

(1) 提出了动态盘阵 D/H 数据分布算法.动态盘阵中,优化的数据分布算法,必须使数据重构的数据量最小、数据分布始终保持很好的平衡性和负载平衡性,而 D/H 数据分布算法是较为理想的选择;

(2) 提出了基于控制理论的在线重构技术.动态盘阵的在线重构技术,必须考虑系统服务的 QoS 与重构时间之间的平衡.基于控制理论的技术能够根据当前的 QoS 和系统资源情况的变化,不断调整重构请求的输入速率,找到较理想的重构方式;

(3) 研究中进行了大量模拟实验.实验针对动态盘阵的 D/H 和基于控制理论的重构展开,并与文献[5]中 eager 在线重构模式比较,实验结果表明,本研究基于控制理论的重构是系统 QoS 与重构速度之间的较好折衷.

本文第 1 节是相关工作,从动态盘阵的研究、在线迁移技术和控制理论的广泛应用,进行详细阐述.第 2 节是动态盘阵的 D/H 分布.第 3 节是基于控制理论的在线数据重构技术.第 4 节是模拟实验.第 5 节是结论.

1 相关工作

盘阵通过优化数据分布,让多个设备并发工作提高 I/O 性能,已经得到广泛应用.但对于一个配置确定的盘阵来说,其服务能力已被限定,为了满足对 I/O 性能的不断增长的需要,盘阵必须有很强的动态扩展能力.

Ghandeharizadeh 和 Kim 等人[5]最先对动态盘阵进行研究,但他们对盘阵一直采用 round-robin 分布.虽然 round-robin 分布能够保持请求很好的并发性、设备上空间的平衡性,但是每次扩展,几乎要移动所有的数据,数据重构开销太大;Goel 等人[6]采用动态盘阵构建连续媒体服务器时,研究提出的 SCADDAR 分布算法,能够获取最小的重构开销,但扩展到几次后,该算法不能保证盘阵设备间工作负载平衡性[3].Round-robin 与 SCADDRA 分布是空间平衡性与重构开销的两个极端.Ho 和 Lee 等人的研究[3]提出了在负载平衡性与重构开销之间的折衷数据分布算法——行排列数据重构算法 RPDR(row-permutated data reorganization),但是该算法不能使空间平衡性与重构开销同时达到优化;并且 RPDR 中数据分布的映射不能像 Round-robin 和 SCADDRA 那样通过简单的映射函数计算出数据所处的物理位置,而是要通过映射表.表映射方式的缺点是:当盘阵的容量很大时,映射表浪费的存储空间巨大,一致性问题突出;且由于每次访问都要通过映射表,则该映射表容易成为瓶颈.

经研究,如果盘阵每次扩展少量设备就重构一次,扩展较为频繁时,数据重构的开销太大[3],以至于必须从扩展获取的性能与扩展开销的角度重新考虑扩展重构的意义.因此,更有意义的方式是,当盘阵扩展到一定规模后再进行一次数据重构[3].为此,我们提出动态盘阵扩展与收缩重构方式是 Double/Halve 方式,也就是说,盘阵的扩张是设备数目增加一倍,而收缩是指设备数目减半.针对 D/H 方式,本研究提出了一种新的数据分布方式——D/H 分布,使得盘阵在动态扩展时,空间平衡性与重构开销同时达到优化.

QoS 一直是许多系统(特别是 I/O 子系统)中的一个很重要的指标[8,9].盘阵的在线重构更应该考虑 QoS.文献[5]针对 round-robin 分布的在线重构讨论了 lazy 和 eager 两种算法.所谓 lazy 算法[5]是指,只有当数据被客户端的 I/O 请求到,并且该数据是必须进行重构迁移的时候,才将该数据块进行迁移.Lazy 算法中数据重构负载不会更多地增加系统的额外 I/O 负载,因而不会影响系统原来的 QoS,但其缺点是:① 数据重构时间是不确定的,且会很长;② 由于重构时间太长,动态盘阵扩展的性能不能很快地对系统的 QoS 带来明显改善.所谓 eager 算法[5]是指,每个时间片都给重构分配固定的一些资源.这种方法使得数据重构时间得到了控制,但是由于采用分配固定资源的方式,当系统的 I/O 负载很重时,eager 算法会影响系统的 QoS,而如果系统负载轻时,又不能充分利用系统资源使重构尽快完成.

在线数据迁移已经有了大量的研究.LVM[10]和 VxVM 通过镜像技术,数据迁移的过程中保证数据的连续访问;Aqueduct[7]采用通过检测系统空闲时才发出数据迁移请求等等.但是动态盘阵的在线重构与系统的数据迁移不完全相同.比如:① 传统的数据迁移中,一般不考虑迁移目的地的负载情况,也就是说,数据迁移只需符合源设备的 QoS-guarantee,而不考虑目的设备的 QoS;② 一般认为,迁移过程中源设备的负载变化不大.但在动态盘阵的重构过程中,随着重构的进行,源设备与目的设备同时承担系统的客户端应用 I/O 负载,而且承担的负载是不断变化的,各设备由于 QoS-guarantee 影响的设备请求输入率不完全相同.

近年来,控制理论在计算机领域内被广泛应用[11].比如分布式系统中基于反馈控制来保障网络报文率[12]、

采用控制理论来分析 IP 路由的拥塞控制算法、基于反馈控制的实时调度技术[13]、基于反馈控制的 Web 服务中 cache 管理、Aqueduct[7]中采用了基于控制理论的在线数据迁移技术等.这些研究中控制理论应用不是针对动态盘阵 I/O 子系统展开的.本研究第一次采用控制理论的思想,将动态盘阵的 QoS 控制和系统当前负载信息等综合考虑,进行反馈控制,在不影响系统 QoS 的情况下,尽可能地提高重构的速度,减少数据重构时间.

2 D/H 分布

动态盘阵与静态盘阵的区别是动态盘阵的数据分布必须考虑性能动态扩展的需要.也就是说,每次动态扩展时,根据数据分布技术,经过适量的数据重构,就能保持设备上所使用空间的平衡性,同时还要保持盘阵的性能与容量同时随着盘阵的扩展而扩展.

2.1 动态盘阵数据分布要求

动态虚拟盘阵是一种盘阵,其数据分布除了满足盘阵数据分布要求[14]以外,主要必须满足以下动态可扩展方面的要求(为了表述方便,假定最初是 N_0 个磁盘,在此基础上扩展第 j 次后,设备总数为 N_j):

I. 空间平衡性:存储系统每次动态变化,都会引起数据迁移.动态负载需均匀分布在各设备上,保证每次扩张后各设备所使用的空间保持一种平衡性,每个设备上重构迁移的数据块数目基本相同.如果令盘阵的设备编号为 0,1,…,N_j-1,$E[k]$为系统在盘阵的设备 k 上的数据量,其中($0 \le k \le N_j-1$),那么 $E[0] \ge E[1] \ge \cdots \ge E[N_j-1]$;

II. 动态负载最小(本文中动态负载和重构负载,都是指动态扩展时为了均衡而引起迁移的数据块):动态重构时,数据分布算法要使迁移数据最小.假定扩展前每个盘的已使用的空间为 S,第 j 次扩展时,需移动的数据量为 $S_{j\text{-}move}$: $S_{j\text{-}move} = \dfrac{|N_j - N_{j-1}|}{\max(N_j, N_{j-1})}(N_{j-1} \times S)$;

III. 负载平衡性:设备空间的平衡性并不能保证一个大请求能由多个设备最大可能地并发完成.这是因为,可能出现一个很大请求的所有数据块在少数的几个设备上.为此,盘阵的每次动态变化,如果原来请求是最大可能并行完成,则根据数据分布算法,动态重构后,请求的并发性不能减少.这要求动态扩展后的设备间负载均衡.

2.2 具体算法

如果既要使盘阵的设备数目随意扩展,又要满足动态盘阵的数据分布要求的分布算法是很困难的,并且是不现实的[3].针对盘阵的动态扩张时每次是设备数加倍,而收缩则减半(Double 或 Halve)的方式,克服以往方法的缺点,我们提出了类似于二分法的分布算法,称为 D/H 算法.

所谓 D/H 是指,如果盘阵最终需要扩展 k 次,对于初始盘阵,每个设备被分成 2^k 个虚设备,以 2^k 取模取整的 round-robin 的方式将系统数据存放在虚设备上.这样每次动态变化只需移动一半的数据.

以镜像为例说明数据分布,如图 1 所示.图 1 是仅动态变化两次的情况.盘阵最初由两个盘构成,每个盘有 16 块数据,扩张 1 次为 4 个盘,第 2 次扩张为 8 个盘.图中 *Bi* 是要存放的数据块,而 *Mi* 是 *Bi* 的镜像块.

最初,两个盘每盘分割成 4 个虚盘(2^2),编号 disk0~disk7.数据按 8 个虚盘以 round-robin 存储(以 8 取整得到物理盘号,以 8 取模得到数据块存放在设备上的物理块号).扩张一次,设备数加倍,将原来分割出的 disk2,disk3,disk6,disk7 数据移动到增加的两个盘上.动态第 2 次时,将 disk4,disk5,disk6,disk7 数据移动到新增的 4 个盘上.收缩则相反.

从分布特性来看,D/H 算法是比较理想的,它符合动态虚拟盘阵数据分布的要求,且映射 hash 函数简单,符合 round-robin 特性,是所有算法中分布性能最好的.虽然在 D/H 分布中,最初配置就确定和限制了盘阵的扩展不能超过 k 次,但是最初可以确定 k 为一个很大的值,使得实际的盘阵系统不可能包含那么多设备.比如最初,盘阵的设备数目为 2(最小值),确定 k 为 20,则要使盘阵不能再扩展,盘阵的扩展超过 20 次,此时盘阵必须包含 1M 个设备,实际系统中,这种盘阵是不可能的.

很容易看到,D/H 算法的缺点是,设备扩张的数目要求太苛刻.比如现在盘阵由 100 个设备构成,如果要扩张一次,则必须增加 100 个设备,只增加了 99 个都还不行.

但是,如果扩展的设备数目不符合要求,我们可以通过一些方法,尽可能地变通使用,比如暂时只进行部分

数据的重构,减轻部分设备的 I/O 负载;或者通过一些自适应技术减轻瓶颈设备的负载等.

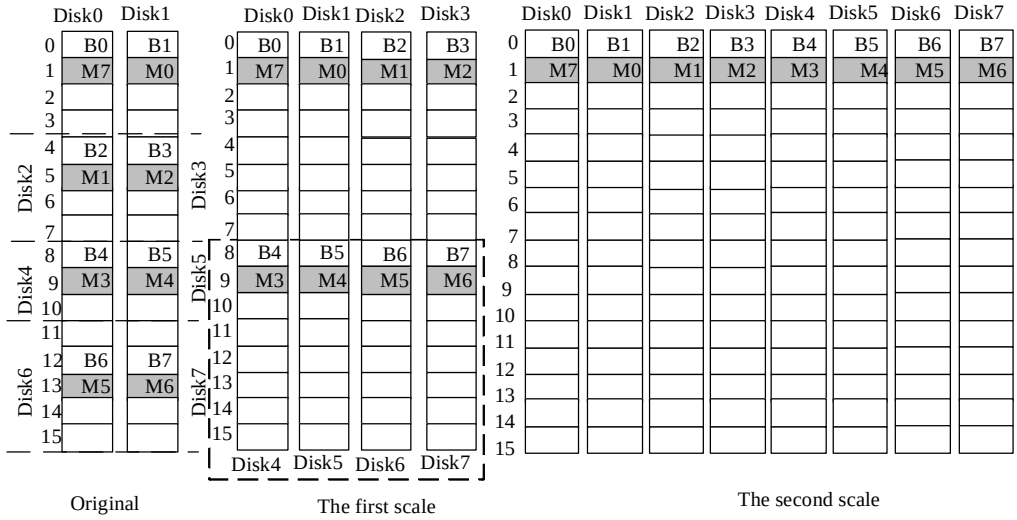


Fig.1 Data placement in D/H while scaling for two times

图 1 D/H 算法中动态扩张两次的数据分布图

3 基于控制理论的在线重构

盘阵在线扩展要求系统一边服务客户端应用程序的 I/O 请求,一边进行数据的重构.其中满足客户应用程序的 QoS 需求是主要的,是 I/O 性能优化的标准,也是确定数据重构序列的一个出发点[7].

但是,由于 I/O 系统服务请求所需的时间很难确定(请求需要排队、磁头定位和数据传送等),且负载本身有一定的波动性[15],系统很难保证对每个请求的绝对 QoS.一般情况是,通过统计方式确定系统是否满足 QoS 要求.动态盘阵的在线重构要求重构不影响系统服务的 QoS,在尽可能短的时间内完成重构.

3.1 QoS问题

一般情况下,特定系统的存储服务器有专门的用途,一般面向一种客户端应用,因而 QoS 要求一般是相同的.对 I/O 子系统来说,客户端发出的 I/O 请求的 QoS 一般用所限定的响应时间 RT(response time)来评价.令存储服务器服务一个请求的时间为 ST(service time),如果 $ST \leqslant RT$,则该 I/O 请求的 QoS 是有保障的(QoS-guarantee);否则,该请求失效(miss deadline).而整体 QoS 是统计请求失效率 PM(probability of miss-deadline)是否小于一个约定阈值 $\boldsymbol{s}$.

当然,不同系统的应用程序,其 QoS 中请求的 RT 和 PM 阈值 $\boldsymbol{s}$ 限定标准是不同的,这与具体的系统运行情况、资源情况和缓冲区大小等多种因素有关,是从系统的具体测量得出的.

在动态盘阵的重构过程中,为了保障系统的 QoS,将重构过程分成 M 个时间段 SW(sample window).定义第 i 个时间段 SW_i 中的请求失效率为 PM_i,失效率偏差为 VPM_i(variation of probability of miss-deadline),$VPM_i=PM_i-\boldsymbol{s}$.要使得系统对 QoS 有保障,则必须使 $PM_i \leqslant \boldsymbol{s}$,其中 $1 \leqslant i \leqslant M$.

而实际给定的系统,其对 I/O 请求的服务率 $\boldsymbol{l}$ 是一定的.一般情况下,系统 QoS 的保障是由当前的请求输入率 Rate 与系统服务率 $\boldsymbol{l}$ 的关系决定.如果 $Rate > \boldsymbol{q}\,\boldsymbol{l}$,则系统的 QoS 是不能得到保障的;否则,系统的 QoS 可以得到保障的,其中 $\boldsymbol{q}$ 为 0~1 之间一个给定常数,它与系统配置、ST、RT 等限定有关.因此,我们采用请求输入率来控制系统 QoS 情况.

3.2 控制理论方法

由于本研究的动态盘阵采用 D/H 分布,数据重构过程是指,将数据由原来的设备迁移到新的设备.在迁移过程中,源设备必须服务当前系统客户端应用程序的请求,又要服务迁移请求;而随着重构的进行,目的设备也会

承担部分系统客户端请求,从而源设备上承担的系统客户端负载越来越轻,源设备上可用来重构的 I/O 资源越来越多,盘阵在线重构的速度越来越快,而不是 eager 算法中固定不变[5];而在一般的负载迁移算法中,认为系统客户端对某个设备的请求率也是不变的,因此原来一般的数据迁移算法在动态盘阵的重构过程中也不会有优化的效果.

面对这种自适应情况,基于控制理论的方法是一种很好的选择.因为这些方法是根据当前系统情况,作为反馈进行自适应调整,确定下一步重构速度,获取优化的结果.

3.2.1 基于反馈控制的在线重构方法

基于反馈控制的在线重构如图 2 所示.图中,动态盘阵原来有两个磁盘,动态扩展时增加两个盘,由于数据分布是 D/H 算法,所以数据重构是将第 1 和第 2 个磁盘上的一半数据分别迁移到第 3 和第 4 个磁盘.盘阵的输入包括客户端的请求和重构请求,客户端的请求是不能控制的,而重构请求可以根据实际情况进行调整.盘阵控制器监控各磁盘设备上客户端请求的 QoS 情况作为反馈,确定在下一个时间段的重构请求输入率.

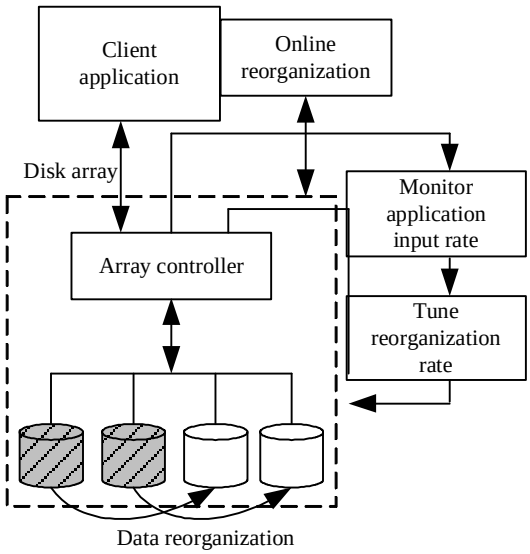


Fig.2 Feedback control in DDA data reorganization

图 2 动态盘阵的数据重构反馈控制

磁盘控制器针对每个采样时间段监控.通过监控,如果源设备和目的设备都是 QoS guarantee,则增加重构的输入率,否则减少重构请求的输入率.具体的操作控制方式如下:

(a) 监控测试出磁盘 l,QoS guarantee 时的 q 等值;

(b) 在重构过程中的第 k 个 SW,测试出客户端的请求对每个磁盘的输入率 $r^k=(r^k_i)$.用向量 $R^k=(R^k_i)$表示重构对所有磁盘的请求输入率.这些输入情况可以分成两部分:$R^k_0,\ldots, R^k_{N_0-1}$, $R^k_{N_0},\ldots, R^k_{2N_0-1}$,其中 $R^k_0,\ldots, R^k_{N_0-1}$ 为盘阵初始第 0~第 N_0-1 设备的输入率, $R^k_{N_0},\ldots,R^k_{2N_0-1}$ 为扩展新增设备输入率;

(c) 估计第 $k+1$ 个 SW 时客户端应用对各设备的请求输入率.对于任意设备 j: $r_j^{k+1}=b_0+b_1 r_j^k$,其中 b_0, b_1 是系统给定的配置参数;计算出重构请求对各设备的输入率: $R_j^{k+1}=\begin{cases}0, & r_j^{k+1}\ge q\ l\\ q\ l-r_j^{k+1}, & r_j^{k+1}<q\ l\end{cases}$.但是,在 D/H 分布中,重构是将数据从磁盘 j 迁移到磁盘(N_0+j)上,$0\le j<N_0$,实际上的磁盘 j 重构请求输入率为 $R_j^{k+1}=\min(R_j^{k+1}, R_{j+N_0}^{k+1})$;

(d) 根据新计算出的请求重构输入率 R^{k+1},作为第 $k+1$ 个 SW 时盘阵重构输入率,进行重构.

因为请求输入率服从一定规律的随机分布,所以实际的输入率是波动的.如果客户端负载的输入率小于

QoS guarantee 的磁盘服务率 $\boldsymbol{q}\ \boldsymbol{l}$,那么可用一部分资源进行重构.然而,下一阶段的负载情况是不知道的,这需要进行预测.事实上,人们可以根据历史信息,预测客户端应用的负载[17].在这些预测方法中,线性回归是一种常用的模型[16].因此我们采用一元简单线性回归的方式去拟合输入变化情况,确定 $\boldsymbol{b}_0, \boldsymbol{b}_1$ 后,再估计出下一步输入;而参数 $\boldsymbol{q}$ 必须选择适当,使得满足输入率 Rate 小于系统服务率 $\boldsymbol{l}$ 时,系统的 QoS 一般也都是能满足的.

3.2.2 参数确定

对系统参数的确定是非常复杂的过程.参数 $\boldsymbol{b}_0$ 和 $\boldsymbol{b}_1$ 的确定与请求输入服从的随机分布规律、盘阵重构的过程中随客户端向某特定设备的请求率影响等许多因素有关;而参数 $\boldsymbol{q}$ 的选择与系统配置、设备特性和 QoS 限定等许多因素相关.

确定参数 $\boldsymbol{q}$ 和设备的服务率 $\boldsymbol{l}$,我们可以在静态环境下进行(不考虑动态迁移的情况).而估计 $\boldsymbol{b}_0$ 和 $\boldsymbol{b}_1$ 时,我们测试许多个不同采样时间段(SW)的输入率,然后动态估计出这两个值.比如,如果我们知道前 m+1 个 SW 的请求输入率为 $x_0,x_1,x_2,\ldots,x_m$,则:

$$\boldsymbol{b}_1=\frac{\sum_{i=0}^{m-1}(x_i-\bar{x})(x_{i+1}-\bar{y})}{\sum_{i=0}^{m-1}(x_i-\bar{x})^2},\ \boldsymbol{b}_0=\bar{y}-\boldsymbol{b}_1\bar{x},\text{其中}\ \bar{x}=\frac{1}{m}\sum_{i=0}^{m-1}x_i,\ \bar{y}=\frac{1}{m}\sum_{i=1}^{m}x_i.$$

在参数确定过程中,这 m 个采样时间段类似于滑动窗口,一个个时间段往下推进,因此,$\boldsymbol{b}_0$ 和 $\boldsymbol{b}_1$ 是动态确定的.

当然,如果具体环境是异构盘阵,则必须针对每个设备进行测试,计算出各自的 $\boldsymbol{l}$, $\boldsymbol{q}$, $\boldsymbol{b}_0$ 和 $\boldsymbol{b}_1$,用向量表示这

些值.

4 模拟实验

由于不能找到 D/H 分布的动态盘阵环境,我们采用模拟的方式,仿真一个盘阵由 4 个磁盘扩展成 8 个磁盘的情况.为了简单起见,我们仿真同构的 RAID0 盘阵动态扩展与迁移的情况.

4.1 实验模型

我们编写了专门用于本研究的模拟器.模拟器主要包括:负载对象(ClientWorkload)、盘阵对象(DiskArray)、磁盘设备部件(Disk).请求产生器根据一定负载合成方式,产生 I/O 请求,向盘阵发出;盘阵根据 D/H 映射算法,将上层的请求转发到相应设备.而当盘阵动态扩展时,盘阵对象根据控制理论原理,对重构速度进行调节.为了能够进行反馈控制,盘阵记录各设备的客户端请求率和 QoS 情况;磁盘构件完成盘阵调度的请求,每个磁盘对象有自己的请求处理队列,以 FCFS(first come first service)服务队列中请求,并返回结果,不考虑请求合并等 I/O 优化策略.

模拟中,只考虑磁盘操作的时间,不考虑 cpu,cache 和总线传输的影响.

测试中盘阵的 SU 大小为 256K 字节,也就是说,数据分布是以一个 track 大小、以 round-robin 方式来进行 D/H 轮循分布的.在盘阵扩展的重构中,是一个个 track 进行迁移.模拟实验采用磁盘 Quantum Atlas 10K II 73.4GB,其参数见表 1,但是为了简单起见,在模拟器中,磁盘的每个磁道的大小认为是 256KB,并且取数据的传输率是 42MB/S,不考虑磁盘的内磁道与外磁道差别的影响.

模拟器中,磁盘的请求服务时间 SW 包括寻道时间、旋转时间和数据传输时间.其中:第 1,数据传输时间是 SU 大小除以设备传输率;第 2,关于旋转时间,仿真测试负载有两种,一种是基于 trace 的,在此类负载中,每个请求的大小不是一个 track,服务此类请求时,需要将磁头旋转到请求的起始扇区,然后才能读取数据,因此,在仿真器中,我们采用 0 至最大旋转时间(6ms)之间的一个随机数作为此类负载的磁头旋转时间.但是为了提高重构效率,请求处理时,如果请求的数据所处的磁道是需要迁移的,那么将整个数据读出,再写入目的设备;另一种是基于合成负载,请求是 track-aligned 访问,在此类负载中,因为请求大小是整个磁道,所以所有旋转时间都是有用的,我们不考虑旋转时间[18];第 3,寻道时间采用文献[19]中的连续函数:

$$Seek(x)=\begin{cases}0, & \text{if } x=0\\ a\sqrt{x-1}+b(x-1)+c, & \text{if } x>0\end{cases},$$

a,b 和 c 由以下公式进行估计:

$$a=(-10\min Seek+15avgSeek-5\max Seek)/(3\sqrt{Cyl}),$$

$$b=(7\min Seek-15avgSeek+8\max Seek)/(3Cyl),$$

$$c=\min Seek,$$

其中 min*Seek*,*avgSeek*,max*Seek* 和 *Cyl* 分别是 head switch,average seek,full stroke seek 和设备的最大柱面数.

Table 1 Disk parameter in simulation

表 1 模拟实验采用的磁盘参数

Disk model	Quantum Atlas 10K II 73.4GB
Cylinders	17 337
Total tracks	346,740
Sector per track	301~528
Bytes per sector (bytes)	512
Head switch (ms)	0.6
Average seeking time (ms)	4.7
Full stroke seek (ms)	12
Revolution (rpm)	10 000
Media transfer (MB/s)	31.0~42.0
Year	2000

4.2 负载模型

我们的仿真负载分成两种:基于 trace 的和基于合成负载的.Trace 是一些系统上的 I/O 序列,反映实际的一些情况.一般情况下,trace 负载有一定局限性,不能反映所有的负载情况,特别是不能充分反映负载的随机变化,系统分析时往往需要采用一定方式合成负载进行测试[16].为此,我们还根据一定的特性,随机合成客户端请求负载,进行模拟.

实验中基于 trace 的负载是根据 sprite trace1 进行的.将 sprite trace[20]中的请求记录,作为该仿真器的客户端输入负载,这些请求是访问一些块,不是以磁道为边界的[18].在该 trace 中,记录的请求有 open,close,delete,make link,set attributes 等所有的文件操作,我们只处理 Read,Write 两种请求.

而基于合成负载是指,针对 MPEGII 格式(4Mb/s)视频点播的情况进行,客户端请求到达时间服从 Poisson 分布,客户端的访问以磁道为边界[18](256KB,Track-alinged).

4.3 参数确定

在仿真中,对于这两种负载,我们规定 QoS-guarantee 的请求响应时间 RT 是 0.5s,且规定 QoS-guarantee 允许的请求失效率 $\boldsymbol{s}$ 为 0.001%.

经过大量仿真测试可知,对于基于 trace 的仿真(随机寻道),单个磁盘的请求服务率 $\boldsymbol{l}$ 为 93.6,QoS-guarantee 时的 $\boldsymbol{q}$ 为 0.91;而对合成负载(Track-aliged 访问),单个磁盘的请求服务率 $\boldsymbol{l}$ 为 93.5,QoS-guarantee 时的 $\boldsymbol{q}$ 为 0.92.

对于 $\boldsymbol{b}_0$ 和 $\boldsymbol{b}_1$ 的动态确定,我们采用在相连的 10 个采样时间段,每个采样时间段为 1s 的输入情况来估计.

4.4 实验结果

在实验中,我们比较基于控制理论的在线重构与固定资源重构 eager 方法.而 eager 重构中采用 0.2-eager 和 0.5-eager.这里,*x*-eager 是指,在重构过程中,每个单位时间内,盘阵控制器的重构控制用 x 个单位时间由盘阵控制器的重构模块进行动态重构的数据迁移.

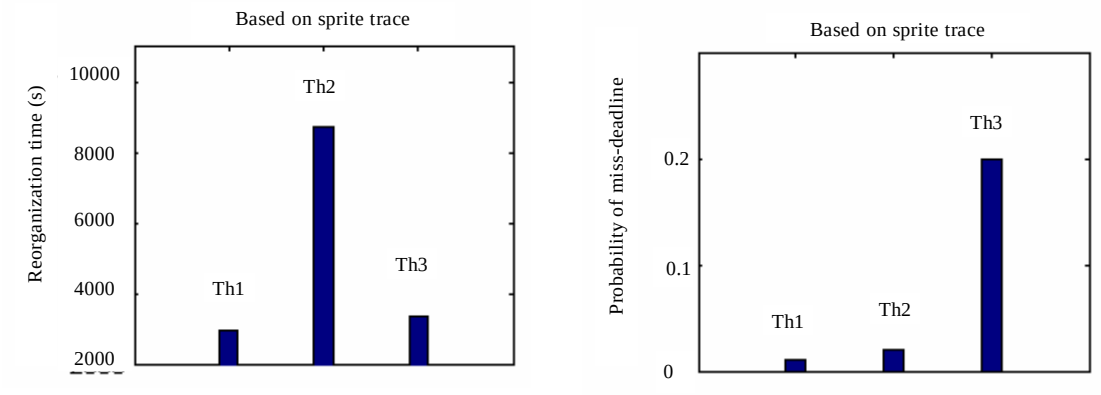
在模拟中,重构是这样进行的:第 1,如果当前客户端应用的 I/O 访问的磁道正好是需要进行迁移的(二者重叠),则将该磁道全部读出,然后一边给客户端返回数据,同时将该道的数据写入扩展后的目的设备;第 2,除了第 1 中与客户端 I/O 重叠的重构以外,基于控制理论反馈的方法,盘阵的重构控制构件产生其余的重构请求;而 *x*-eager 方法则是以固定的速率产生其余的重构请求.

仿真盘阵动态扩展过程,测试重构时间和 QoS 情况.重构时间是指从重构开始到重构结束的时间段;而 QoS

用重构过程中的请求失效率来衡量.请求失效率是在重构过程中所有客户端请求中失效(miss deadline)数目与总客户端请求数目的比值.

实验结果如图 3 和图 4 所示.为了描述方便,在图中 Th1 是指基于控制理论的在线重构方法,而 Th2 是指 0.2-eager 方法,Th3 是指 0.5-eager 方法.图 3 和图 4 的结果通过 4 次测量的平均值.这 4 次是指时间从模拟开始的第 2000s,第 6000s,第 10000s 和第 14000s 时动态扩展一次.

图 3 是基于 sprite trace 的重构情况.由图 3(a)可知,控制理论反馈法的重构时间最少,而 0.5-eager 方法次之,0.2-eager 方法的重构时间最长.我们认为,这是因为通过控制理论反馈,能够尽可能地利用空闲资源进行重构.由图 3(b)可知,基于控制理论方法的请求失效率最小,0.2-eager 次之,而 0.5-eager 最差.这说明,基于控制理论反馈原理的方法,不仅提高重构效率,而且能够保障请求 QoS.

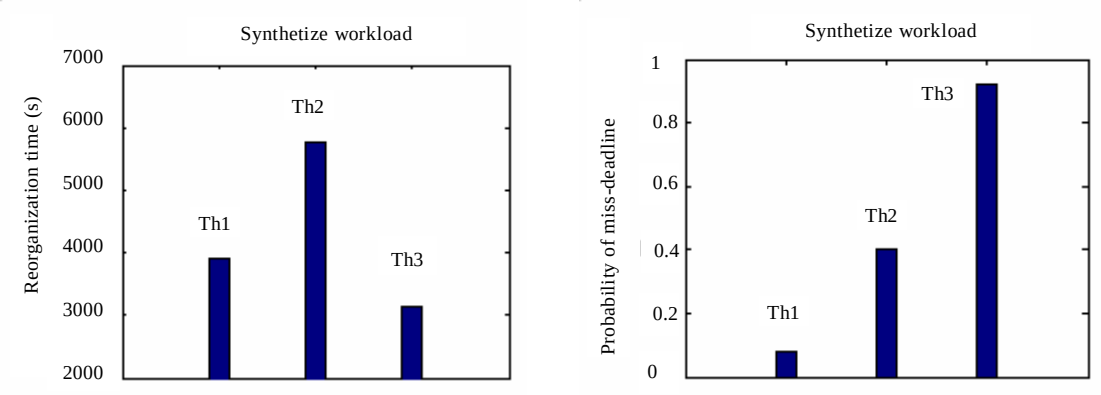


(a) On-Line reorganization time (b) Probability of request miss-deadline

(a) 在线重构时间 (b) 请求失效率

Fig.3 Simulation result based on Sprite-trace

图 3 基于 Sprite-trace 的模拟结果图



(a) On-Line reorganization time (b) Probability of request miss-deadline

(a) 在线重构时间 (b) 请求失效率

Fig.4 Simulation result based on synthesize workload

图 4 基于合成负载的模拟结果图

基于合成负载的重构情况如图 4 所示(在实验中,合成负载的客户端请求输入率为 $4q\ l$).由图 4(a)可知,0.5-eager 方法的重构时间最少,而基于控制理论方法的时间次之,0.2-eager 方法的重构时间最长.该情况下,在 4 个设备构成的盘阵中,客户端请求负载已经使得系统忙不过来,能用于重构的资源就不是很多,因而最初基于控制理论方法的重构速率很慢,只有随着重构进行,新增加的设备承担部分负载,原来磁盘上的客户端请求减少,从而加快重构速度.由图 4(b)可知,基于控制理论的方法的请求失效率最小,0.2-eager 方法的 QoS 也不理想,而 0.5-eager 方法最差,请求失效率特别高.

图 3 与图 4 的实验结果是有差别的.我们认为这种差别主要来自于:① 客户端的负载率不同,在 Sprite Trace 1 中的统计表明,其读写请求的输入率仅为每秒 3.25 个请求,请求的平均长度为 4KB.Sprite Trace 1 的请求输入率远远低于盘组的处理能力,此时系统可用来重组的资源很多.而合成负载的请求输入率则几乎接近了系统的处理能力,用于重组的资源较少,因而基于预测的重组时间很长;② 客户端负载与重构负载重叠的情况不同;③ 客户端负载的随机波动性不同,使得反馈控制时,采用线性回归模型预测的准确程度不同.

该实验说明,基于控制理论反馈原理的方法能使重构效率和 QoS 达到较优.但是在研究中,采用简单线形回归模型预测后面一个时间段的客户端随机的请求输入情况,有一定的误差.在实验中,我们也发现,简单线性回归的反馈预测方式接近于用前面采样时间段的客户端输入率的均值,与实际的请求输入情况有一定误差,有时误差较大,此时不能使重构速度和 QoS 达到优化,特别是,在大量的模拟实验中,我们发现很难保障原来 QoS Guarantee 的请求失效率 s 小于 0.1%的要求.

5 结束语

由于能适应系统变化的 I/O 需求和系统规模的可扩展性需求,动态盘阵有着广泛的应用.但是动态盘阵为了实现其性能的扩展,必须选择很好的数据分布算法和在线重构策略,使得盘阵扩展后设备上空间保持一种平衡性,并使得动态扩展时引起的数据迁移对当前系统的 QoS 影响较小.为此,本研究提出了 D/H 数据分布算法和基于控制理论的在线重构策略.研究中,D/H 扩展与数据分布算法能使扩展开销、空间平衡性和负载平衡性达到优化.基于 D/H 数据分布,对这种在线重构策略进行了模拟,实验表明,基于控制理论的在线重构策略,能使系统 QoS 和动态重构速度达到较优.

但是基于控制理论的算法关键是基于当前输入进行反馈,预测后面的情况.在本研究中采用线性回归模型.实验表明,该模型太简单,与实际的客户端负载情况有误差,因而不能得到最优的结果.下一步工作需要寻找更准确的反馈函数,使重构达到最优.

References:

[1] Shen XH, Choudhary AN, Matarazzo C, Sinha P. A distributed multi-storage resource architecture and I/O performance prediction for scientific computing. Cluster Computing, 2003,6(3):189 - 200.

[2] Harting H. An I/O architecture for microkernel-based operationg systems. Technical Report TUD-FI03-08-Juli-2003, Dresden, TU Dresden, 2003.

[3] Ho TK, Lee JYB. A row-permutated data reorganization algorithm for growing server-less video-on-demand systems. In: Lee S, Sekiguchi S, eds. Proc. of the 3rd IEEE/ACM Int'l Symp. on Cluster Computing and the Grid (CCGRID 2003). Tokyo: IEEE/ACM Press, 2003. 44 - 51.

[4] Liu J, Tang YH, Wang L, Jiang YH, Yang XJ. The dynamic mass storage system. Journal of Computer Research and Development, 2002,39(Suppl.):162 - 167 (in Chinese with English abstract).

[5] Ghandeharizadeh S, Kim D. On-Line reorganization of data in scalable continuous media servers. In: Feitelson DG, Rudolph L, eds. Proc. of the 7th Int'l Conf. on Database and Expert Systems Applications. Lecture Notes in Computer Science, Zurich, 1996. 751 - 768.

[6] Goel A, Shahabi C, Yao S-YD, Zimmermann R. SCADDAR: An efficient randomized technique to reorganize continuous media blocks. In: Chaudhuri S, Carey M, Garcia-Molina H, eds. Proc. of the 18th Int'l Conf. on Data Engineering (ICDE 2002). San Jose: IEEE CS Press, 2002. 473 - 482.

[7] Lu CY, Alvarez GA, Wilkes J. Aqueduct: Online data migration with performance guarantees. In: Chase J, Cole J, eds. Proc. of the USENIX Conf. on File and Storage Technologies (FAST). Monterey, 2002. 219 - 230.

[8] Abdelzaher TF. An automated profiling subsystem for QoS-aware services. In: Sztipanovits J, Abdelzaher TF, Atkins EM, eds. Proc. of the 6th IEEE Real Time Technology and Applications Symp. Washington DC: IEEE Press, 2000. 208 - 217.

[9] Lumb CR, Merchant A, Alvarez GA. Facade: Virtual storage devices with performance guarantees. In: Chase J, Cole J, eds. Proc. of the 2nd USENIX Conf. on File and Storage Technologies (FAST). San Francisco, 2003. 131-144.

[10] Madell T, Madell T. Disk and File Management Tasks in HP-UX. Printice-Hall: Pearson Education POD, 1997.

[11] Franklin GF, Powell DJ, Workman ML. Digital Control of Dynamic Systems. 3rd ed., New York: Addison-Wesley, 1998.

[12] Li BC, Nahrstedt K. A control-based middleware framework for quality of service adaptations. IEEE Journal of Selected Areas in Communication, Special Issue on Service Enabling Platforms, 1999,17(9):1632-1650.

[13] Lu CY, Stankovic JA, Tao G, Son SH. Feedback control real-time scheduling: Framework, modeling, and algorithms. Journal of Real-Time Systems, 2002,22(1/2):85-126.

[14] Holland M, Gibson GA. Parity declustering for continuous operation in redundant disk arrays. In: Mukherjee S, McKinley KS, eds. Proc. of the 5th Int'l Conf. on Architectural Support for Programming Languages and Operationg Systems (ASPLOS-V). Boston MA: ACM Press, 1992. 23-35.

[15] Gribble S, Manku G, Roselli E, Brewer E. Self-Similarity in file systems. In: Gibson TJ, Gribble SD, eds. Proc. of the 1998 ACM SIGMETRICS Joint Int'l Conf. on Measurement and Modeling of Computer Systems. Madison: ACM Press, 1998. 141-150.

[16] Jain R. The Art of Computer Systems Performance Analysis. New York: John Wiley & Sons, Inc. 1991.

[17] Smith W, Foster I, Taylor V. Predicting application run time using historical information. In: Feitelson DG, Rudolph L, eds. Proc. of the Workshop on Job Scheduling Strategies for Parallel Processing. Lecture Notes in Computer Science, San Juan, 1999. 202-219.

[18] Schindler J, Griffin JL, Lumb CR, Ganger GR. Track-Aligned extents: Matching access patterns to disk drive characteristics. In: Chase J, Cole J, eds. Proc. of the Conf. on File and Storage Technologies (FAST). Monterey, 2002. 259-274.

[19] Lee EK, Katz RH. An analytic performance model of disk arrays. In: Lazowska ED, Felten EW, eds. Proc. of the 1993 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems. Santa Clara: ACM Press, 1993. 98-109.

[20] Baker MG, Hartman JH, Kupfer MD, Shirriff KW, Ousterhout JK. Measurements of a distributed file system. In: Proc. of the 13th ACM Symp. on Operating System Principles. Pacific Grove: ACM Press, 1991. 198-213.

附中文参考文献:

[4] 刘军,唐玉华,王磊,蒋艳凰,杨学军.动态海量存储系统.计算机研究与发展,2002,39(Suppl.):162-167.