磁带库系统的随机 I/O 调度算法*

石 晶, 周立柱

(清华大学 计算机科学与技术系,北京 100084)
E-mail: {shijing,dcszlz}@mails.tsinghua.edu.cn
http://dbgroup.cs.tsinghua.edu.cn

摘要: 由于磁带库随机存取的性能很差,需要研究有效的随机 I/O 调度策略和算法以改善其在线存取的效率.对已有调度算法进行了分类、提炼和总结,利用仿真实验对静态调度、动态调度和基于复制的调度算法进行了深入研究,讨论了影响各种算法有效性的因素.针对已有算法在较重的负载条件下使系统性能急剧恶化的问题,还提出并研究了一种基于效益-代价均衡的调度算法.该算法引入效益-代价加权的概念,通过调节不同负载下的效益-代价加权比,极大地改善了已有算法在重负载下的有效性.该项研究为设计海量存储系统中的自适应调度算法提供了重要依据.

关 键 词: 磁带库系统;随机 I/O 调度算法;静态调度;动态调度;基于复制的调度;基于效益-代价均衡的调度
中图法分类号: TP311 文献标识码: A

磁带库这种大容量第三级存储设备一直是作为备份和归档设备被人们所熟知的.然而,随着科学技术的发展,各领域的数据呈现爆炸式增长[1],已经出现和将要出现规模在 10^{12} 字节(Terabyte)以上的电信通话记录数据库、大型数字图书馆、地理、空间及环境数据库以及视频音频归档数据库等等.对于这些大型数据库来说,数据的存储和操作完全依赖磁盘存储系统是不合适的.一个重要的发展趋势就是把诸如磁带这样的大容量存储介质直接用于数据的存储和查询操作,使磁带库不再只是“被动”的备份和归档设备,而是成为大型数据库系统的存储结构中的“主动存储层次”[2].

从系统的角度来讲,磁带库要成为“主动存储”设备,首先需要解决数据库系统对磁带库中数据的在线随机存取问题.以顺序存取为特征的磁带存储,其随机存取的性能很差,所以,需要研究有效的随机 I/O 调度策略和算法,以改善磁带存储的在线存取效率.在这方面已有的研究成果包括单条磁带的随机调度算法[3,4]和多条磁带的磁带库调度算法[5~8],这些成果在其特定的应用环境和系统条件下对系统的性能有明显的改善.但由于这些算法的优化方法明显不同,且目前对它们还没有全面的评测和分析,所以,这些算法在各种负载条件下的有效性是需要进一步研究的.

本文侧重对已有的磁带库随机 I/O 调度算法进行深入的探讨,全面分析了影响各种算法有效性的因素.针对已有算法在重负载条件下使系统性能急剧恶化的问题,我们提出“效益-代价”均衡的概念,以改善所有调度算法在重负载条件下的有效性,同时给出了一种基于效益-代价均衡的调度算法.本文的研究以 Exabyte220 磁带库的性能参数为基础,使用磁带库仿真器对各种调度算法进行模拟研究,磁带库系统模型和仿真参数将在第 1 节介绍.第 2 节详细描述了各种调度算法.第 3 节重点介绍各种调度算法的仿真结果和分析.第 4 节给出仿真结果与实验结果的比较分析.第 5 节给出结论.

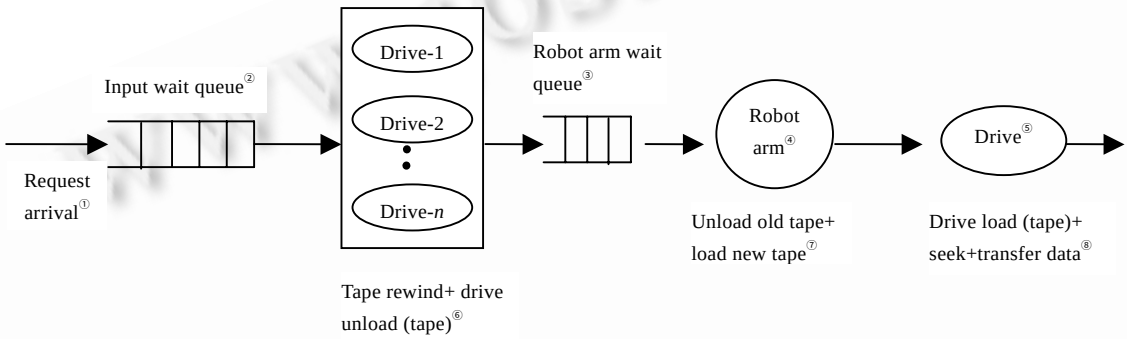
* 收稿日期: 2001-02-26; 修改日期: 2001-06-13
基金项目: 国家重点基础研究发展规划 973 资助项目(G1999032704)
作者简介: 石晶(1969－),女,辽宁辽阳人,博士,主要研究领域为海量信息处理,第三级存储设备操作与管理;周立柱(1947－),男,江苏连云港人,教授,博士生导师,主要研究领域为数据库,海量信息处理,Web 技术.

1 磁带库系统模型

1.1 系统模型

磁带库是自动加载磁带的大容量存储设备,它主要由磁带驱动器、机械手、磁带架和磁带组成.磁带库的一个 I/O 操作的过程如下(假设涉及的磁带不在驱动器中):(1) 磁带反绕,回到磁带开始处(对于 zone 型磁带,只需反绕到最近的一个 zone 点);(2) 驱动器卸载,记录出错日志和块目录信息后把磁带从驱动器中弹出;(3) 机械手卸载,把旧带送回到原来的磁带槽中;(4) 机械手加载,从磁带槽中取出所需的磁带并送到驱动器中;(5) 驱动器加载,完成磁带引导、速度校准等工作;(6) 驱动器搜寻定位,找到需要操作的第 1 个块;(7) 开始传输数据.如果所需磁带刚好存在于驱动器中,则只需要驱动器搜寻定位并传输数据两个操作过程.

对于整个磁带库系统来说,其操作的系统模型如图 1 所示.其中,模型中包含两个等待队列:输入等待队列和机械手等待队列.当请求到达时,首先进入输入等待队列,等待驱动器空闲,这里是多服务器模型,即多个驱动器任一个空闲都可以服务于请求.当请求获得驱动器后,首先进行驱动器卸载,然后进入机械手等待队列等待机械手的服务.机械手空闲后,机械手从驱动器中卸载旧磁带,然后再加载所需磁带至驱动器中.磁带在驱动器进行加载操作,待搜寻定位后开始传输数据.



①请求到达,②输入等待队列,③机械手等待队列,④机械手,⑤驱动器,⑥磁带反绕+驱动器卸载(磁带),⑦送回旧磁带+取新磁带,⑧驱动器加载(磁带)+搜寻定位+传输数据.

Fig.1 The system model of tape library

图 1 磁带库的系统模型

从图 1 可知,磁带库系统的随机 I/O 请求的平均响应时间主要由以下因素组成:在等待队列的等待时间、磁带交换时间、搜寻定位时间和数据传输时间.其中,只有数据传输时间是 I/O 的有效时间.所以,改善磁带库系统的随机 I/O 性能的途径主要有以下几个方面:(1) 减少请求在等待队列中的平均等待时间;(2) 减少磁带的交换次数;(3) 减少请求的搜寻定位的平均距离.

1.2 系统仿真和参数度量

磁带存储设备在扫描方式、容量、传输率和价格上都有很大差别,但其操作的系统模型基本与第 1.1 节所描述的模型类似.本文的研究以第 1.1 节所描述的系统模型为基础,仿真实现磁带库系统,并在其上进行各种调度算法的研究分析.仿真系统以 Exabyte 220 磁带库、Eliant 820 磁带驱动器、EXABTYE 8mm 磁带的性能参数为实验参数,参数见表 1.其中,除磁带容量外的所有参数都为实际测试数据.由于 EXABTYE 8mm 磁带是螺旋扫描方式,所以其搜寻定位时间与搜寻距离是线性关系.本文使用的搜寻定位模型参照了文献[5]中的模型.

Table 1 Measurement of tape library Exabyte 220

表 **1** Exabyte 220 磁带库的性能参数

Tape①(20)	Capacity④	7GB(uncompressed⑭)
Tape drive②(2)	Data transfer rate⑤	0.92MB/sec (uncompressed)
	Mean drive load time⑥	25s
	Mean drive eject time⑦	23s
	Rewind startup time⑧	7s
	Rewind rate⑨	22MB/s
	Seek startup time⑩	8.5s
	Seek rate⑪	30.2MB/s (uncompressed)
Tape robot③(1)	Mean tape unload time⑫	7.2s
	Mean tape load time⑬	7.42s

①磁带,②磁带驱动器,③磁带机械手,④容量,⑤数据传输率,⑥平均驱动器加载时间,⑦平均驱动器卸载时间,⑧反绕启动时间,⑨反绕速率,⑩搜寻启动时间,⑪搜寻速率,⑫平均磁带卸载时间,⑬平均磁带加载时间,⑭未压缩.

2 调度算法

为了便于分析比较,本节对已有调度算法[5~8]的基本思想进行了分类、提炼和总结,同时对我们提出的基于效益-代价均衡的调度算法进行描述和讨论.

2.1 基本调度算法

基本调度算法是除了先来先服务算法外的最直观的算法,它主要包含以下几个步骤:

(1) 把等待队列中的所有请求按照磁带分组,每个磁带有一个候选调度列表.这一步的目的是使针对同一磁带的请求尽可能在一次磁带扫描中完成,从而减少换磁带的次数;

(2) 对每个磁带的候选调度列表中的请求按地址升序排序(如使用蛇型扫描方式的 DTL 驱动器,则请求应按文献[3]中的定位模型排序).这一步的目的是得到最佳的单磁带调度策略;

(3) 按照磁带选择策略选择一个磁带调度列表,并提交给一个空闲的驱动器执行.

在基本调度算法中,磁带选择策略是影响算法有效性的决定因素.常用的策略包括循环(round robin)选择、最多请求选择、最大有效带宽选择、最老请求选择等策略.其中,最大有效带宽选择策略中的"有效带宽"是指一次调度所传输的数据总量(MB)除以该调度执行所花费的时间(秒)所得到的结果.文献[5]中的实验表明,在一般的负载条件下,最大有效带宽选择策略是最有效的策略.

2.2 静态和动态调度算法

静态调度是指系统在一次调度提交后,所有请求(包括未被调度的请求和新到达的请求)都必须等待下一次调度.为了最大限度的提高调度算法的效率,调度计划总是在可以获得一个空闲驱动器时开始计算.静态调度算法的一个问题是没有利用新到达请求的潜在优化可能.它对于新到达的、针对正在驱动器中运行的磁带的请求不进行立即处理,而是使之进入等待队列,等待下一次调度开始.

动态调度是为改进静态调度的这个缺陷而设计的,它把新到达的可以立即调度的请求插入相应的正在服务的调度列表中,使这些请求很快得到服务.为了尽可能有更多的请求,动态调度算法把调度的执行过程分为从头向尾的正向扫描和从尾向头的反向扫描两个过程,相应地,调度列表则由正向调度列表和反向调度列表组成.当新到达的请求满足插入条件(即请求涉及的磁带正在某个驱动器上执行)时,开始处理插入操作.首先获取该调度的当前执行数据块地址(当前位置)和当前扫描方向,然后参照该地址和扫描方向作插入处理:

- 如果当前处在正向扫描过程,则把请求数据块的地址大于当前执行位置的请求按地址顺序插入正向列表,对小于当前位置的请求,按地址顺序插入反向列表;
- 如果当前处在反向扫描过程,则把请求数据块的地址小于当前位置的请求,按地址顺序插入反向列表,而对大于当前位置的请求插入等待队列.

在执行完正向列表和反向列表后,此次调度结束,同时释放驱动器,开始下一个调度计划的处理.

2.3 基于数据复制的调度算法

磁带库系统中的数据复制是指在磁带的尾部复制部分(热)数据,一般是用 20%的带长存放热数据的拷贝.复制热数据的目的是为增加每次调度包含的请求数,减少换磁带次数.由于热数据和热数据拷贝同时存在,所以,该算法把所有请求分为两类,分别称为Ⅰ类请求和Ⅱ类请求.Ⅰ类请求是指请求所涉及的数据无复制的情况,Ⅱ类请求是指请求所涉及的数据有复制的情况.下面是一个基于数据复制的最大有效带宽选择策略的算法描述:(1) 顺序扫描请求等待队列,把Ⅰ类请求直接插入相应磁带调度列表中;把Ⅱ类请求插入新的请求等待队列.扫描结束后,把各磁带调度列表中的最后一个请求的起始地址记为该磁带的分界点.(2) 顺序扫描(1)中生成的请求等待队列,把请求插入相应磁带的调度列表中.对于请求数据地址大于所插入磁带调度列表的分界点的请求,还需要为该请求数据的每个拷贝复制一个存取数据拷贝的请求,并把这些请求插入到拷贝所在的磁带的调度列表中.(3) 计算每一个磁带调度列表的有效带宽,提交具有最大有效带宽的磁带调度列表进行执行.其中,步骤(1)的目的是通过Ⅰ类请求确定必选的所有磁带,并使用分界点标明这些磁带执行必须定位的最远距离.步骤(2)把Ⅱ类请求中的在分界点以内的请求直接插入,这样可以减少定位时间,而对分界点外的请求则考虑其拷贝对选择磁带的作用.

2.4 基于效益-代价均衡的调度算法

前面讨论的动态调度和基于复制的调度都侧重追求每次调度的最大效益,这样的优化策略在较轻的负载条件下是很有效的,而在较重的负载条件下,系统性能急剧恶化(这一结论从下节的实验结果中可以得出),原因是越来越多的请求长时间的等待服务,最终使优化付出的代价超过优化获得的效益.于是我们提出一种基于效益-代价均衡的调度算法,该算法的核心思想是在磁带选择算法中引入效益-代价加权的概念.其中的“效益”是指磁带执行的有效带宽,该值越大,调度的效益越高;“代价”是指磁带上请求的平均等待时间,该值越大,调度的代价越高.下面是磁带选择算法中磁带的效益-代价计算公式(简式):

$$E_t = W_{bw} * \frac{S_t}{S_{avg}} + W_{wt} * \frac{T_t}{T_{avg}}. \tag{1}$$

其中 E_t 是磁带 t 的效益代价估算值,调度算法总是选择 E_t 最大的磁带作为下一个服务的磁带.S_t 是待计算的磁带 t 的有效带宽,T_t 是待计算的磁带 t 上的当前所有请求的平均等待时间,这两个参数是每个磁带的自有参数.S_{avg} 是系统当前平均有效带宽,T_{avg} 是系统当前已完成的所有请求的平均等待时间,两个参数值都是随着系统的执行而不断变化的,反映系统当前的平均性能.W_{bw} 是有效带宽的效益加权,W_{wt} 是平均等待时间的代价加权,两者的比值大小 $R_{w/b}$($R_{w/b} = \frac{W_{wt}}{W_{bw}}$)则决定了磁带选择的不同倾向,或者有效带宽占主导(比值很小时),或者平均等待时间占主导(比值很大时),或者综合考虑两方面因素,该比值的确定是基于效益-代价的调度算法的关键.

基于效益-代价均衡的调度算法是从磁带选择策略上对调度进行优化,它与静态调度、动态调度以及基于复制的调度相结合,就形成了基于效益-代价均衡的(静态、动态、基于复制的)调度算法.

3 仿真结果与分析

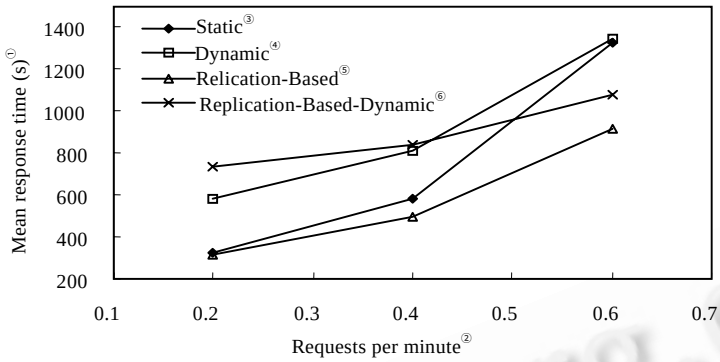
本节给出的调度算法的仿真结果都是在如下仿真条件下得到的:(1) 仿真系统运行在 COMPAQ 的奔III-450 微机,Linux 操作系统;(2) 磁带库系统的系统仿真模型使用第 1.1 节介绍的模型;(3) 系统的请求到达过程符合泊松分布,即请求到达的间隔时间符合指数分布;(4) 系统仿真时间为 2 000 小时.

另外,在仿真实验中还需确定 3 个参数:请求大小、请求倾斜度和热数据的位置.请求大小是指一个请求涉及的连续数据块的大小.请求倾斜度是数据冷热的度量,一般表示为 a/b,即 a%的请求针对 b%的数据,如果 $a>b$,则这 b%的数据就是热数据.热数据的位置是指热数据在磁带库中磁带的分布位置.这 3 个参数对调度算法的效果有很大影响.我们在仿真时使用的请求大小为 64M,请求倾斜度为 70/10,热数据位于每盘磁带的尾部.

3.1 几种调度算法的仿真结果比较

图 2(a)~(c)分别给出了静态调度、动态调度、基于数据复制的(静态)调度和基于数据复制的动态调度 4 种

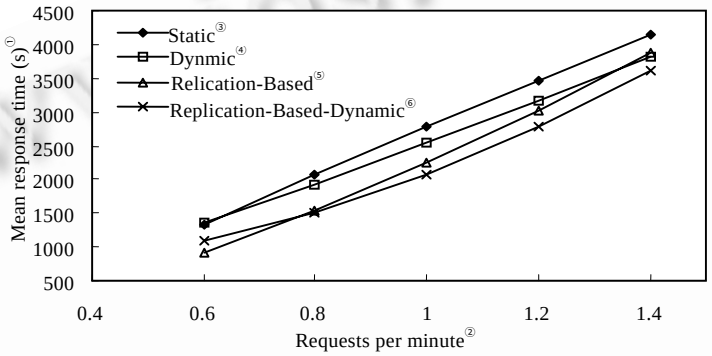
调度算法在不同负载条件下的仿真结果,其中,4 种调度算法所使用的磁带选择算法都是最大有效带宽策略.



①平均响应时间(秒),②请求数/分钟,③静态(调度),④动态(调度),
⑤基于复制(调度),⑥基于复制动态(调度).

(a) Performance comparison of several scheduling algorithms (light workload)

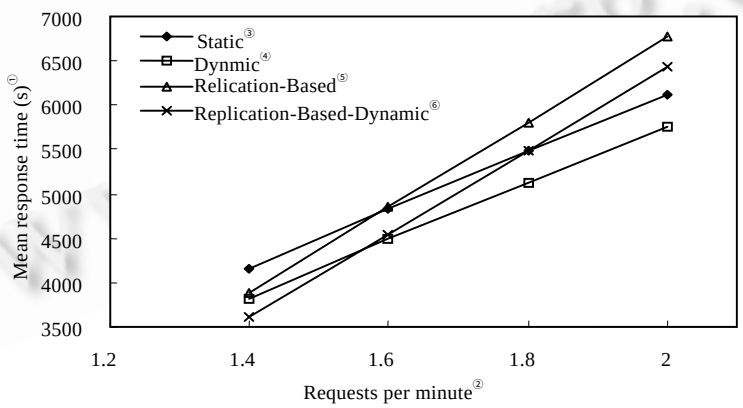
(a) 几种调度算法的性能比较(轻负载情况)



①平均响应时间(秒),②请求数/分钟,③静态(调度),④动态(调度),
⑤基于复制(调度),⑥基于复制动态(调度).

(b) Performance comparison of several scheduling algorithms (moderate workload)

(b) 几种调度算法的性能比较(中度负载情况)



①平均响应时间(秒),②请求数/分钟,③静态(调度),④动态(调度),
⑤基于复制(调度),⑥基于复制动态(调度).

(c) Performance comparison of several scheduling algorithms (heavy workload)

(c) 几种调度算法的性能比较(重负载情况)

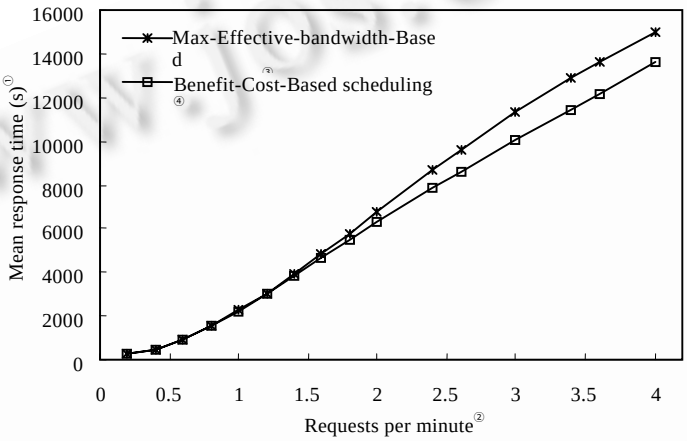
Fig.2

图 2

从图 2(a)~(c)可以看出,在不同的请求负载条件下,4 种调度算法的有效性是不同的.在极轻的负载(到达

率<0.5)下,基于复制的调度算法的性能最好,动态调度算法的性能最差.在较重负载(0.5≤到达率<1.6)下,基于复制的调度算法的性能最好,动态调度算法次之,静态调度算法的性能最差.在重负载(到达率≥1.6)下,动态调度算法最好,基于复制的调度算法的性能最差.图中的基于复制的动态调度算法综合了热数据复制和动态调度两种方法,其对系统性能的影响是两种策略的综合体现.在两种算法都有效的负载条件下,该算法有最佳的性能改善率.

我们提出的基于效益-代价均衡的调度算法,与静态调度、动态调度、热数据复制等算法相结合,可以改善这些算法在重负载条件下的系统性能.该算法通过在磁带选择算法中引入了效益-代价加权,使优化倾向随负载而变化.图 3 给出了基于复制的效益-代价均衡调度与基于复制的最大有效带宽调度(图 2 中的基于复制的调度)的性能比较.在请求负载较轻时,效益-代价均衡调度接近于最大有效带宽调度,这说明此种情况下最大有效带宽策略是最优的策略,效益-代价均衡调度倾向于突出最大有效带宽的效益.随着负载的加重(到达率>1.5),请求长时间等待的代价已经不能被忽视,所以,效益-代价均衡调度开始倾向于考虑请求长时间等待的代价,从而使系统的性能得到了很大的改善.且负载越重,其优势越明显.效益-代价均衡调度与静态调度和动态调度的结合会得到与图 3 类似的结论.



①平均响应时间(秒),②请求数/分钟,③基于最大有效带宽调度,④基于效益-代价均衡调度.

Fig.3 Benefit-Cost-Based scheduling

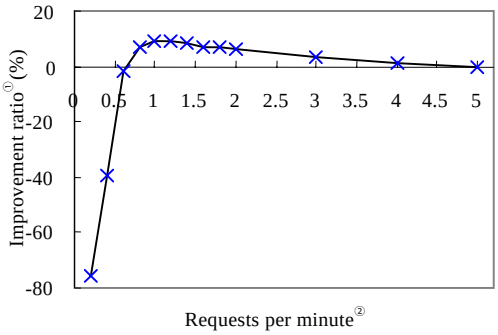
图 3 基于效益-代价均衡的调度

3.2 影响算法有效性的因素

3.2.1 动态调度

从图 2(a)~(c)可以看出,动态调度的效果受请求负载条件的影响.图 4 给出了与静态调度相比,动态调度在各种负载条件下对系统性能的改善率,它从不同的角度更直观地说明了动态调度与请求负载的关系.图 4 表明,在负载极轻(到达率<0.5)时动态调度的效果极差,这是因为负载轻时,请求的动态插入率很低,动态插入的优势不明显;而每个磁带上的请求很少,动态插入的请求的定位时间加长,对性能的影响超过了动态插入请求获得的效益.另一方面,在负载极重(到达率≥2.5)时,动态调度对系统性能的改善率在逐渐减小,这是因为当负载达到一定程度后,请求的动态插入率达到最大并且基本保持不变(本实验中为 9%左右),而负载的加重使等待队列中的请求数迅速增多,等待时间越来越长,从而最终抵消了动态插入请求所带来的效益.

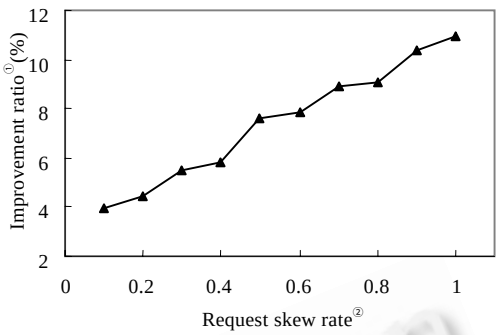
对动态调度有很大影响的另一个因素是请求倾斜度.图 5(请求到达率为 1.0)给出了动态调度在各种请求倾斜度(从 0.1~1.0,对应于 10/10~100/10)下对系统性能的改善率,它表明请求倾斜度越高,动态调度越有效.其原因是随着请求倾斜度的增加,动态插入请求的重复率增加,即多个请求针对同一连续数据块,这种情况下只需要传输一次数据,从而缩短了每次调度的实际服务时间,加快了调度的频率.



①改善率,②请求数/分钟.

Fig. 4 Impact of workload upon dynamic scheduling

图 4 请求负载对动态调度的影响



①改善率,②请求倾斜度.

Fig.5 Impact of request skew upon dynamic scheduling

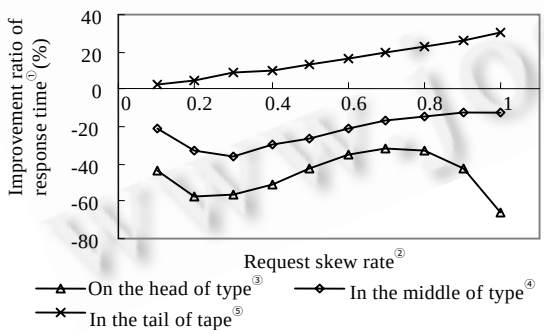
图 5 请求倾斜对动态调度的影响

3.2.2 基于复制的调度

如图 2(a)~(c)所示,基于复制的调度也同样受请求负载因素的影响.由于该调度算法的优化目标是借助热数据的复制,追求最大有效带宽,尽可能减少磁带交换次数,所以在负载较轻时,该策略可以缩短等待队列的长度,减少所有请求的平均等待时间,改善系统的性能.但由于热数据的拷贝数是有限的,所以,最大有效带宽的增长终将滞后于请求负载的增长.于是,随着负载的加重,减少的磁带交换次数使等待队列中的请求数和请求等待时间都迅速增加,并很快抵消了热数据复制所带来的效益,使系统的性能恶化.与动态调度相比,该算法对重负载更敏感.

另一个影响基于复制的调度的因素是热数据的位置,热数据的放置位置直接影响请求的定位时间.图 6(请求到达率为 1.0)给出了热数据分别放在磁带头、磁带中部和磁带尾时,算法在各种请求倾斜度条件下对系统性能的改善率.如图 6 所示,当热数据位于磁带的头部和中部时,对热数据进行复制并不能改善系统的性能,因为选择数据本身比选择位于磁带尾的热数据的拷贝的定位时间要少.而当热数据位于磁带尾时,热数据复制对改善系统的性能是很有效的.另外,图 6 也显示了热数据复制与请求倾斜度的关系,倾斜度越高,复制热数据越有效.

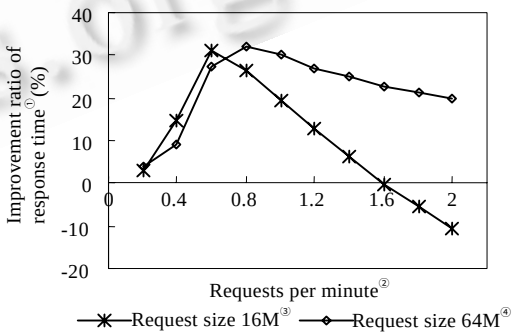
另外,基于复制的调度算法对请求大小较小的情况更有效.其原因在于热数据复制的存在直接优化了热数据存取的定位时间,当请求大小较小时,数据传输的时间很短,相对来说搜寻定位时间占了主导,所以,对定位时间的优化在请求大小较小的情况下显得更有效.图 7 给出了请求大小分别为 16M 和 64M 在各种负载条件下对系统性能的改善率.



①响应时间改善率,②请求倾斜度,③(热数据)在磁带头,④在磁带中部,⑤在磁带尾部.

Fig.6 Impact of replacement polices of hot data upon replication-based scheduling

图 6 热数据的放置策略对基于复制的调度的影响



①响应时间改善率,②请求数/分钟,③请求大小 16M,④请求大小 64M.

Fig.7 Impact of request size upon replication-based scheduling

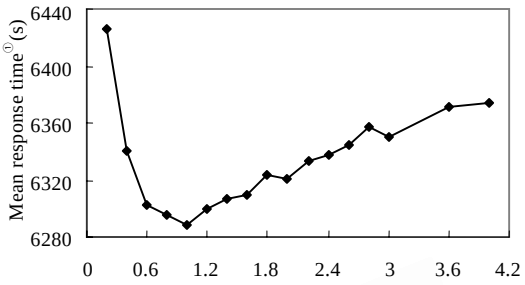
图 7 请求大小对基于复制调度的影响

3.2.3 基于效益-代价均衡的调度

从图 3 可以看出,系统负载的轻重直接影响基于效益-代价的调度算法的有效性,负载越重,对系统性能的改

善越明显.如在图 3 的实验条件下,当到达率>1.5 时,该算法极大地改善了系统性能.

另一个影响算法有效性的因素是 $R_{w/b}$,即平均等待时间的代价加权和有效带宽的效益加权的比值.由于该比值决定了磁带选择的不同倾向,所以,在某一负载条件下取不同的 $R_{w/b}$,算法对系统性能的改善程度也不同.从另一个角度看,只有在该比值的最佳值区间中取值,才能使系统达到某一负载条件下的最佳性能.图 8(请求到达率为 2.0,请求大小为 64M)给出了系统性能与加权比的关系.如图 8 所示,加权比值对系统性能的改善有很大影响.



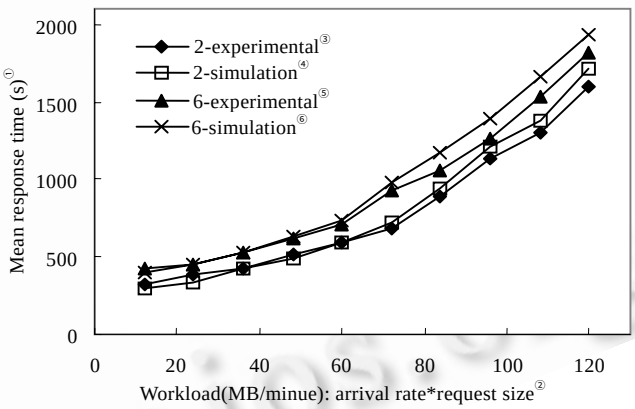
①平均响应时间,②等待时间和有效带宽加权比.

Fig.8 Impact of weight ratio upon benefit-cost-based scheduling

图 8 加权比对基于效益-代价调度的影响

4 仿真结果与实验结果比较

本节以动态调度算法为例,给出仿真结果与实验结果的比较,如图 9 所示.图中显示的是到达率分别为 2.0 和 6.0 的两组比较实验,工作负载用到达率和请求大小的乘积表示.另外,与第 3 节中仿真条件不同的是请求只涉及 10 盘磁带.从两组实验可以看出,本文的仿真结果与实验结果是很近似的.但仿真结果值比实验结果值稍高,原因是,在我们的仿真模型中使用的定位模型、反绕模型和数据传输模型都是线性的,且使用的参数都是近似测量出来的,而在磁带库中相应的模型实际是近似线性的.但仿真结果绝对值与实验结果值的差别几乎不影响仿真实验结果的正确性.



①平均响应时间(秒),②到达负载(MB/分钟):到达率*请求大小,
③2-实验,④2-仿真,⑤6-实验,⑥6-实验.

Fig.9 Result comparison of simulation and experiment

图 9 仿真结果与实验结果比较

5 结 论

本文对已有调度算法进行了分类、提炼和总结,并给出详细描述.同时,利用仿真实验对静态调度、动态调度、基于复制的调度进行了深入研究,讨论了影响算法有效性的因素.提出并研究了一种基于效益-代价均衡的调度算法,该算法侧重改善重负载条件下的各调度算法的系统性能.本文的研究结果在负载确定条件下可以指导调度算法的选择,而对负载复杂多变的环境条件,则可以为设计海量存储系统中的自适应调度算法提供重要依据.

References:

[1] Cariño, F., Kaufmann, A., Kostamaa, P. Are you ready for Yottabytes?. In: Kobler, B., ed. Proceedings of the 17th IEEE Symposium on Mass Storage Systems in Cooperation with the 8th NASA GSFC Conference on Mass Storage Systems and Technologies. Maryland: IEEE Computer Society Press, 2000. 476~485.

[2] Cariño, F., Burgess, J., O'Connell, W., *et al*. Active storage hierarchy, database systems and applications——socratic exegesis. In: Malcolm, P.A., Maria, E.O., *et al*., eds. Proceedings of the 25th International Conference on Very Large Data Bases. Edinburgh: Morgan Kaufmann Publishers Inc., 1999. 611~614.

[3] Hillyer, B.K., Silberschatz, A. Random I/O scheduling in online tertiary storage Systems. In: Jagadish, H.V., Mumick, I.S., eds. Proceeding of the 1996 ACM SIGMOD International Conference on Management of Data. Quebec: ACM Press, 1996. 195~204.

[4] Hillyer, B.K., Silberschatz, A. Scheduling non-contiguous tape retrievals. In: Kobler, B., ed. Proceedings of the 15th IEEE Symposium on Mass Storage Systems in Cooperation with the 6th NASA GSFC Conference on Mass Storage Systems and Technologies. Maryland: IEEE Computer Society Press, 1998. 113~124.

[5] Hillyer, B.K., Rastogi, R., Silberschatz, A. Scheduling and data replication to improve tape jukebox performance. In: Papazoglou, M., Pu, C., Kitsuregawa, M., eds. Proceeding of the 15th International Conference on Data Engineering. Sydney: IEEE Computer Society Press, 1999. 532~541.

[6] Nemoto, T., Kitsuegawa, M. Scalable tape archiver for satellite image database and its performance analysis with access logs——hot declustering and hot replication. In: Miller, E., ed. Proceedings of the 16th IEEE Symposium on Mass Storage Systems in Cooperation with the 7th NASA GSFC Conference on Mass Storage Systems and Technologies. San Diego: IEEE Computer Society Press, 1999. 59~71.

[7] Triantafillou, P., Georgiadis, I. Hierarchical scheduling algorithms for near-line tape libraries. In: Cammelli, A., Wagner, R.R., eds. Proceedings of the 10th International Conference and Workshop on Database and Expert Systems Applications. Florence: IEEE Computer Society Press, 1999. 50~54.

[8] Prabhakar, S., Dvyakant, A., Amr, El A., *et al*. Scheduling tertiary I/O in database applications. In: Roland, W., ed. Proceedings of the 8th International Workshop on Database and Expert Systems Applications. Toulouse: IEEE Computer Society Press, 1997. 722~727.

Random I/O Scheduling Algorithms in Online Tape Library Systems*

SHI Jing, ZHOU Li-zhu

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

E-mail: {shijing,dcszlz}@mails.tsinghua.edu.cn

http://dbgroup.cs.tsinghua.edu.cn

Abstract: Since the tape libraries have very poor random access performance, it is critical to study random I/O scheduling strategies and algorithms in order to improve the performance of tape library. In this paper, the existing scheduling algorithms are summarized first, and then the analytical results of the effectiveness of static scheduling, dynamic scheduling and replication-based scheduling are presented through simulations. In particular, a benefit-cost-based scheduling algorithm is given, which aims to improve the effectiveness of existing scheduling algorithms under heavy workloads by tuning the weight ratio of cost and benefit of scheduling policies according to workloads. This algorithm is significantly effective under heavy workloads. The research of this paper forms the basis of the design of adaptive scheduling algorithms that can be used in massive storage systems.

Key words: tape library system; random I/O scheduling algorithm; static scheduling; dynamic scheduling; replication-based scheduling; benefit-cost-based scheduling

* Received February 26, 2001; accepted June 13, 2001
Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1999032704