一种保证服务品质的存储网络实时存储调度机制*

冯 泳，张延园
(西北工业大学 计算机科学与工程系，陕西 西安 710072)

摘 要：提出了一种保证服务品质的存储网络实时存储调度机制，它可以在多变的应用环境中保证存储应用获得预先说明的服务品质，该品质不会由于其他存储应用的负载变化而受到影响。这个机制还较好地解决了控制力和系统总体性能之间的矛盾，并且具有自动适应运行环境变化的能力。

关键词：存储网络；实时存储调度；服务品质
中图法分类号：TP393 文献标识码：A 文章编号：1001-3695(2005)01-0204-03

A Real-time Storage Scheduling in Storage Network with QoS Guarantees

FENG Yong, ZHANG Yan-yuan
(Dept. of Computer Science & Engineering, Northwestern Polytechnic University, Xi'an Shanxi 710072, China)

Abstract: In storage network, storage resources are shared by many independent storage applications. In this kind of competitive environment, these storage applications often require predictable quality of service. In this paper, a real-time storage scheduling with quality of service guarantees is presented. The scheduling ensures the request of storage applications is met, irrespective of the varying load imposed by other applications sharing the same storage resources. Furthermore, it gains good tradeoffs between control ability and aggregate performance of storage device, so it not only guarantees the quality of service, but also achieves a good aggregate performance. Moreover it is adaptive and can adjust to different working environments.
Key words: Storage Network; Real-time Storage Scheduling; Quality of Service(QoS)

由于互联网和信息技术的发展，存储的容量、带宽和可用性等需求在过去的十年中发生了爆炸性的增长。存储网络为满足日益提高的存储需求提供了硬件基础，它可以通过高速网络连接众多分散的存储资源和服务器，摒弃了原有的存储系统专属于服务器的工作模式，实现了存储系统和服务器多对多的连接。存储虚拟化技术被认为是存储管理的核心技术，它可以将分散的物理存储资源组成多个 RAID 系统并形成存储池，充分利用存储网络的潜力。存储虚拟化技术向存储应用隐藏了数据块的物理地址和组织结构，不仅可以提高资源利用率，减少管理成本，而且具有更好的适应性和可用性[1]。由于多个存储应用共享同一个存储池，根据存储设备的性能和存储应用的要求，给存储应用分配合适的存储资源成为保障服务品质的基础。但是，即使在总体上分配的存储资源可以满足多个存储应用的要求，由于存储应用之间存在相互影响，故无法保证在任何时刻、任何负载模式下存储应用都能获得预先说明的服务品质。有的访问请求可能获得了高于其要求的服务品质，有的访问请求的服务品质却没有得到满足；或者由于突发情况，某些存储应用占用了其他存储应用的存储资源。此时，就需要一个实时的调度机制控制访问请求的执行顺序和开始时机。

1 相关研究和设计思路

一个好的存储网络实时存储调度机制应当达到如下要求：①服务品质的独立性。各存储应用间不能相互干扰，一个存储应用不能随意占用存储资源而危及其他存储应用的服务品质。②满足服务的要求。应该尽量保证存储应用预先说明的服务品质，每一个存储访问都应该在要求的时间范围内得到响应。③公平。多余的带宽应当正比于预先说明的服务品质分配给各个存储应用。④较高的整体性能。在保证各存储应用服务品质的同时，应当尽量优化以获得较高的存储系统整体性能。⑤自适应。调度机制应该能够根据当前的系统负荷和设备状态进行适当的调整，保证品质目标和系统整体性能。文中的实时调度就以这五点作为设计目标。

在实时操作系统中，存储访问通常被要求在一定的时间范围内完成，所以某些实时操作系统的磁盘调度机制具有类似保证服务品质的能力，如 YFQ 允许进程按权值预约磁盘资源，分配带宽[2]；DS-CSCAN 通过设置每一个访问请求的最迟完成时间，安排服务顺序[3]；VC-CSCAN 根据品质说明和当前的负荷来计算每一个访问请求的最迟开始时间，并以此选择下一个服务对象[4]。在提高设备整体性能方面，它们也有不同的特点：YFQ 首先从各个不同服务类型的等待队列中一次选择多个访问请求，而后根据电梯算法或其他提高吞吐率的算法排列这些请求的服务顺序；DS-CSCAN 和 VC-CSCAN 在保证每一个访问请求的最终期限的基础上，尽量选择距离磁头当前位置最近的访问请求作为当前服务对象。作为实时操作系统的一部分，这些磁盘实时调度机制的目标是减少响应时间超越最终期限所出现的比率，所以它们不能很好地隔离各类存储服务的相互影响。另外，磁盘一次只能处理一个访问请求，它的实时调度有能力控制每一个访问请求的顺序，并且了解磁盘工作的细节，

收稿日期：2004-02-20；修返日期：2004-04-02
基金项目：国家“863”计划资助项目(2001AA142100)；国家教育部博士学科点基金资助项目(20010699018)

可以方便地优化整体性能。但是,作为存储网络的基本存储设备,磁盘阵列比单个磁盘更具有智能性,其处理访问请求的过程也更复杂,可以同时处理多个访问请求。除非实时调度每次只提交一个请求,否则无法控制已提交访问请求的具体执行顺序,也无法通过控制磁盘阵列的工作细节来提高设备的整体性能。

由于磁盘阵列具有智能的控制逻辑并同时管理多个物理磁盘,所以它在接收多个并发的访问请求时,一方面可以在数据传输时准备下一个访问请求,另一方面也可以调整处理顺序以提高整体性能。例如,如果位置靠后的请求所要求的物理设备有空闲,可以先在空闲的磁盘上执行;或者为了减少寻道时间,可以先执行距离磁头近的请求。磁盘阵列同时处理的访问请求越多,其性能潜力越能得到充分的发挥[5]。虚拟存储设备 Facade 已经证明这一观点对大多数磁盘阵列都适用[6]。但是,由于磁盘阵列会重新排列执行队列中访问请求的执行顺序,增加磁盘阵列的执行队列长度意味着削弱实时调度对服务顺序和时机的控制能力,从而增加了违背服务品质的事件的发生率。如果无限度地增加磁盘阵列的执行队列,实时调度也就失去了意义,退化成几乎没有控制。

实时调度的功能在某种意义上类似于滴漏和调节阀,执行队列的长度就相当于滴漏的口径。如果各存储应用产生的工作负荷都低于其要求的品质,并在时间轴上的分布是均匀的,而且各存储应用负荷的相互比例与其所要求的品质的比例相近时,即使实时调度的控制能力不是很强,也能保证存储应用的服务品质。尤其在以下情况出现时,实时调度的控制能力就显得重要了:①相对于要求的性能,某个存储应用产生的工作负荷要明显高于其他存储应用;②相对于要求的性能,某个访问请求太大,会占用其他存储应用的资源,影响服务品质;③出现了违背服务品质的事件。加强实时调度的控制能力除了缩短执行队列的长度外,还包括以下途径:缩短监视和控制周期,提高调度的频率;分割大块的访问请求,进行更细粒度的控制。

实时调度保证服务品质的根本方法是合理分配存储设备为各个存储应用服务的时间,使之在单位时间内的比例与各自的服务品质相匹配,除非某些存储应用的工作负荷低于其要求的服务品质,这时才可以将它们多余的服务时间分配给其他存储应用。文中的调度机制使用访问请求的虚拟开始时间 *VST* 来决定它们的服务顺序。$VST = \max(T_{current}, VET) + l/W_i - l/W$,其中 $T_{current}$是接收到访问请求的时间;*VET* 是所属逻辑磁盘等待队列中最后一个请求的虚拟结束时间;l 是请求的长度;W_i 是分配给所属逻辑磁盘的带宽;W 是所属 RAID 子系统即存储池(以下称之为 RANK)的总带宽。对于磁盘阵列,除了带宽还有许多因素影响一个访问请求的延迟时间,包括服务准备时间、访问模式等,所以 *VST* 并不能用于精确定义一个访问请求的最迟开始时间。但是,由于每一个访问请求的准备时间相差不多,可以与数据传输并行执行,而且访问模式等其他因素的影响较小,所以各访问请求 *VST* 的前后关系从本质上反映了它们要求的服务品质。按照 *VST* 安排访问请求的服务顺序,不仅可以保证服务品质所需的设备服务时间,而且可以公平的分配多余的设备服务时间。

2　系统结构

文中的实时存储调度机制应用于由 Linux 服务器和磁盘阵列组成的存储网络,其系统结构如图 1 所示。实时存储调度机制由用户代理节点和管理节点共同完成。用户代理节点以内核模块的形式运行在应用服务器上,管理节点以用户态进程的形式运行在管理服务器上。用户代理节点中的本地实时调度模块,为它所能访问的每个逻辑磁盘维护一个本地等待队列,并在管理节点的控制下完成实际的数据访问。管理节点由全局实时调度器、控制器、监视器和数据布局控制构成。全局实时调度器为存储网络内每个逻辑磁盘维护一个全局队列,负责为每个 RANK 选择下一个执行的访问请求,并通知对应的用户代理节点;控制器根据工作状态调整控制参数;监视器监视实时调度的效果,并向控制器提供运行状态信息;数据布局控制为调度器,控制器和监视器提供逻辑磁盘的构成信息,例如哪些逻辑磁盘共享同一个 RANK 以及各逻辑磁盘的品质要求和 RANK 的性能参数。

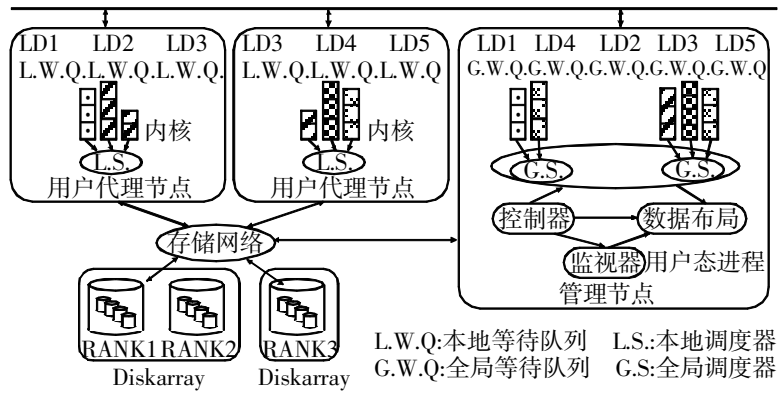

图 1　实时存储调度机制的系统结构

用户代理节点的本地实时调度并不了解逻辑磁盘的 RANK 结构,它对访问请求的调度是在全局实时调度的控制下完成的。管理节点汇总各个用户代理节点产生的服务请求,形成逻辑磁盘的全局等待队列,并根据数据布局提供的信息,将全局实时调度、控制器和监视器按 RANK 划分为彼此相互独立的多个管理单元。用户代理节点需要截获和处理用户的 I/O 请求,所以它以内核模块的形式实现,并向存储应用提供标准的操作系统块设备接口。这种方式不用改变存储应用原有的调用接口,可以方便地适用于各种类型的文件系统以及数据库系统经常使用的 RAW I/O 接口。

3　调度算法

运行在用户代理节点上的存储应用向本地实时调度提交访问请求后,该访问请求的执行过程如图 2 所示。其中:①本地实时调度将该请求的访问对象、数据长度等信息提交给管理节点,并等待回复。②管理节点将访问请求插入全局等待队列,在适当的时候该请求被全局实时调度选择执行,并通知本地实时调度。③本地实时调度开始 I/O 操作。④I/O 操作完毕。⑤本地实时调度向管理节点发送完成报告,管理节点检查实际的服务品质,并调整全局调度系统。

系统内各模块的具体实现算法如下:

W: RANK 的总带宽;W_i 分配给逻辑磁盘 i 的带宽;*l*: 当前访问请求的数据长度;$T_{current}$: 当前时间;*VST*: 虚拟开始时间;

VST: 虚拟结束时间; _Q: 逻辑磁盘的全局等待队列, 等待队列中的访问请求按 *VST* 的大小升序排列; VST_i: 逻辑磁盘 *i* 的 W_Q 队首访问请求的虚拟开始时间; VET_i: 逻辑磁盘 *i* 的 W_Q 队尾访问请求的虚拟结束时间; *N*: RANK 执行队列的规定长度; N_{max}: 执行队列规定的最大长度; $N_{current}$: 执行队列的当前长度; *t*: 系统调度粒度的参数; *ratio*: 衡量运行状态的参数; *E*: 等待队列为空的逻辑磁盘的集合; T_{min}: 控制器执行周期的最小值; T_{max}: 控制器执行周期的最大值。

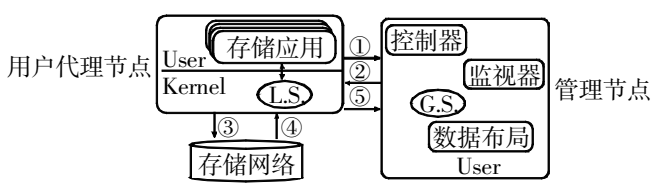

图 2 访问请求处理时序

①本地实时调度接收到访问请求: 将访问请求的长度、逻辑磁盘号和请求序号发送给管理节点, 而后等待管理节点的通知。根据管理节点的通知, 进行指定长度的 I/O 操作, 并向管理节点发送完成报告, 直到该访问请求全部完成。

②全局实时调度接收到访问请求: 如果访问请求的长度 *l* 大于 $t \times W_i$, 则以 $t \times W_i$ 为单位将访问请求分解成多个子请求, 将这些子请求按顺序插入所属逻辑磁盘 W_Q 的队尾。按公式 $VST = \max(T_{current}, VET_i) + l/W_i$ 和 $VST = \max(T_{current}, VET_i) + l/W_i - l/W$ 计算各子请求的虚拟开始时间和虚拟结束时间。如果 *i* 属于集合 *E*, 则将 *i* 从集合 *E* 中删除, 并且如果 $N_{current}$不大于 N, 则调用全局实时调度。

③监视器接收到 I/O 完成报告: 将 $N_{current}$减 1。计算该访问请求的实际执行时间 *realtime*, 并设置 *ratio* 为 $\max(realtime/(l/W_i), ratio)$。如果 $N_{current}$不大于 *N*, 则调用全局实时调度。

④控制器调整控制系统: 控制器按时间 *t* 周期性的执行。首先调整执行队列长度, 如果 *ratio* 等于 1, 设置 *N* 为 $\min(N_{max}, e \times N)$, 其中 *e* 是一个大于 1 的系数(通常的值为 1.2); 对于其他情况, *N* 设置为 $\max(1, N/ratio)$。而后, 设置 *ratio* 为 1。如果 *N* 等于 1, 那么设置 *T* 为 $\max(T_{min}, T/e)$; 否则设置 *T* 为 $\min(T_{max}, T \times e)$。设置集合 *E* 中各逻辑磁盘的 VST_i 为 $\max(T_{current} + T, VST_i)$。如果 $N_{current}$不大于 *N*, 则调用全局实时调度。

⑤全局实时调度选择执行的访问请求: 在 RANK 的所有 W_Q 中, 选择 *VST* 最小的一个 W_Q。如果该队列属于集合 *E*, 则选择结束; 否则取出该队列的第一个访问请求, 并从队列中删除该请求, 如果队列为空了, 那么将它加入集合 *E* 并设置其 VST_i 为 $\max(T_{current} + T, VST_i)$。此后, 向用户代理节点发送 I/O 操作开始通知, 并将 $N_{current}$加 1。如果 $N_{current}$大于 *N*, 那么选择结束; 否则开始选择下一个访问请求。

4 结论

保障存储应用的服务品质需要合理的资源分配和实时的存储调度共同完成。本文对后者提出了一种设计和实现方法, 利用虚拟开始时间控制访问请求的执行顺序和时机, 最终按声明的服务品质分配存储设备的服务时间。为了优化存储系统的整体性能和提高对运行环境的适应力, 还提出了一些方法协调吞吐量和控制能力之间的矛盾, 如分割大块访问请求, 根据当前的负荷特性和实际服务品质控制磁盘阵列执行队列长度等。原型模型表明, 该实时调度机制达到了设计要求, 解决了存储实时调度问题。

参考文献:

[1] Brinkmann, *et al*. Storage Management as Means to Cope with Exponential Information Growth [C]. Proceedings of SSGRR 2003, Italy, 2003. 108-114.

[2] John L Bruno, *et al*. Disk Scheduling with Quality of Service Guarantees[C]. ICMCS, Florence, Italy, vol. 2, 1999. 400-405.

[3] K Gopalan, T Chiueh. Real-time Disk Scheduling Using Deadline Sensitive Scan [D]. New York: SUNY Stony Brook, 2001. 1-6.

[4] Lan Huang. Stonehenge: A High Performance Network Storage Cluster with QoS Guarantees [D]. New York: SUNY Stony Brook, 2003. 1-43.

[5] Richard Barker, Paul Massiglia. Storage Area Network Essentials [M]. New York: Wiley Computer Publishing, 2002. 346-353.

[6] Christopher Lumb, *et al*. Facade: Virtual Storage Devices with Performance Guarantees [C]. Proceedings of the 2nd USENIX Conference on File and Storage Technologies, San Francisco, CA, 2003. 131-144.

作者简介:

冯泳(1977-), 男, 河北清苑人, 博士研究生, 主要研究方向为存储网络、系统软件、网络安全; 张延园(1954-), 男, 河南许昌人, 教授, 主要研究方向为软件工程、存储网络、并行计算。

(上接第 124 页)型信息的集成与管理一直是一个关键问题。本文在分析现有几种集成方案后, 选用了基于 XML 的信息集成方法。通过 CAD 二次开发技术直接提取 CAD 模型的结构信息, 用 XML 语言进行描述, 最后把 XML 文档存储在产品数据管理平台 PDM 中, 便于其他应用系统进行调用和操作, 为基于整个虚拟产品开发平台的信息集成创造了方便的条件。

参考文献:

[1] 熊光楞, 李伯虎, 柴旭东. 虚拟样机技术[J]. 系统仿真学报, 2003, 13(1): 114-117.

[2] 杨流辉. 基于 COM 组件的 CATIA 产品信息集成技术研究与实现[J]. 计算机工程与应用, 2001, 37(24): 132-134.

[3] 杨林, 张田文, 柴旭东. 在复杂产品工程中对虚拟样机产品结构信息的集成[C]. 2003 年全国系统仿真学术年会, 2003.

[4] 白庆华, 何玉林. CIMS 中的系统集成和信息集成[M]. 北京: 电子工业出版社, 1997. 1-14.

[5] 杨流辉. 虚拟样机协同开发平台技术的研究与应用[D]. 北京: 清华大学, 2002. 38-46.

[6] Marcjalb. XML by Example[M]. Que, 1999.

[7] 胡洁. 多学科并行协同设计理论、方法与应用研究[R]. 北京: 清华大学, 2003. 58-70.

[8] 苏铁明, 徐志祥, 欧宗瑛. 基于 XML 的产品数据交换及协同设计技术[J]. 机械科学与技术, 2003, 22(3): 409-501.

[9] 熊光彩, 莫蓉, 毛海鹏, 等. 基于 XML 的 BOM 信息共享和存储[J]. 机械科学与技术, 2002, 21(5): 848-851.

作者简介:

魏坚(1979-), 男, 浙江人, 在读硕士, 主要研究方向为并行工程、产品信息集成等; 张和明(1966-), 男, 浙江人, 副教授, 硕士生导师, 主要研究方向为并行工程、产品信息集成、PDM 等; 张永康(1963-), 男, 江苏人, 教授, 博士生导师, 主要研究方向为先进制造技术、特种加工等。