维普资讯 http://www.cqvip.com

基于流媒体服务的高性能阵列研究*)

姜国松　谢长生　陆正武　王宇德

(华中科技大学计算机学院外存储系统国家重点实验室　武汉 430074)

摘　要　伴随着流媒体市场的不断扩大,基于传统架构的服务器不能满足流媒体数据的特点。本文从 STRIP SIZE,RAID 处理、文件块等方面对流媒体服务器进行研究,针对流媒体应用的特点进行了流媒体服务器的设计,并建立起具有良好性能的流媒体服务器系统。模拟环境的测试结果表明,流媒体服务器的访问速度得到大大提高。

关键词　冗余磁盘阵列,逻辑单元号,分条

Research about Disk Array in Stream Media Server

JIANG Guo-song　XIE Chang-sheng　LU Zheng-wu　WANG Yu-de

(National Storage System Laboratory, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract　This paper discusses stream media applications server with the analysis of the characteristics about STRIPE SIZE, RAID algorithm and block size. According to the characteristics, we design and implement a storage system of stream media applications server which have the excellent performance. The test consequence shows the great improve of access of the media system through simulative testing.

Keywords　RAID, LUN, Stripe

1　引言

目前宽带网络建设热建冷用的现象非常严重,为了向客户提供增值网络服务,而提出的流媒体服务项目将可以有效地缓解这一现象。伴随着视频运营市场的不断扩大,基于传统架构的服务器不能满足视频数据的特点,这主要表现在三个方面:网络接口带宽的瓶颈因素;存储带宽的瓶颈;主机处理能力的不足。其中,尤以存储瓶颈的限制比较大。适合流媒体服务的高性能阵列控制卡就是为解决视频服务器在存储方面出现的瓶颈现象而设计开发的。

2　流媒体服务器现状

目前许多流媒体产品已经实现了产品化,这为流媒体技术的应用提供了软件的基础,另一方面这些流媒体软件只是通过现有操作系统的文件系统对存储设备上的流媒体数据进行读写访问操作,而并不涉及到数据存储系统的设计和流媒体数据的访问调度问题,更不会涉及到流媒体服务器自身的体系结构。因此它们只是解决了服务器与客户端之间数据传输的优化问题,而没有考虑服务器自身的数据 I/O 的优化和对客户端访问的调度。事实上客户端访问流媒体服务器上的流媒体数据时,从存储设备上读取数据而产生 I/O 延迟是整个访问过程中的最主要的时间消耗;如果忽视流媒体服务器存储系统的设计,将会造成流媒体服务器的瓶颈。

为了有效地发挥流媒体巨大的应用潜力,从流媒体的特性出发设计出专门用于存储与传输流媒体数据的流媒体服务器成为当务之急。目前虽然存在着一些用于 VOD 服务的传统流媒体视频服务器,它们中既有 PC 级的小型视频服务器,也有基于并行处理结构的大型视频服务器,但这些服务器不是性能较差,无法支持大规模的并发访问,就是价格昂贵,无法得到普及应用。

3　存储系统方案

解决磁盘带宽瓶颈问题一个比较好的方案是采用 RAID 技术。但由于视频数据本身的特点依靠普通的 RAID 算法技术无法将存储带宽进一步提高,对目前市场上比较成熟的 RAID 产品的分析和测试性能都不是很强。基于上述原因,针对流媒体服务器视频数据的特点提出了自行定义的一些存储规则,通过一定的手段实现存储输出带宽的极大增长。通过相关资料的分析和专利的查询工作,表明新的 RAID-H 技术具有独创性,可以很好解决存储带宽的提升。媒体服务器的存储方案是采用自行设计定义的 RAID 技术实现硬件平台,选用 INTEL 的 IOP 处理器。直接在 INTEL 的 IOP 开发系统硬件上进行 RAID-H 应用软件的开发,实现新的 RAID-H[1]算法。系统硬件的基本框图如图 1 所示。

本设计分为两个子系统,第一是主机端的 IOP DEMO CARD 驱动子系统,第二是开发卡端的 SCSI CONTROLLER 驱动子系统。

在主机服务器端,我们设计完成 IOP DEMO CARD 的驱动程序,将 IOP DEMO CARD,抽象成为一个标准的 RAID 控制卡,主机通过这个驱动程序可以访问所连接的存储设备,而 IOP DEMO CARD 屏蔽所有的存储设备的连接和实际的 RAID 算法实现。主机通过这个子系统可以实现通用的 RAID 配置管理、RAID 策略选择。

*)国家自然科学基金资助项目(60173043);国家 973 重大基础项目(G1999033006)资助。**姜国松**　博士生,研究方向为计算机存储系统;**谢长生**　硕士,教授,博士生导师,研究方向为 NAS 与 SAN 存储体系、iSCSI 存储设备、图像存储与处理和计算机系统结构;**陆正武**　博士生,研究方向为计算机存储系统;**王宇德**　博士生,研究方向为计算机存储系统。

维普资讯 http://www.cqvip.com

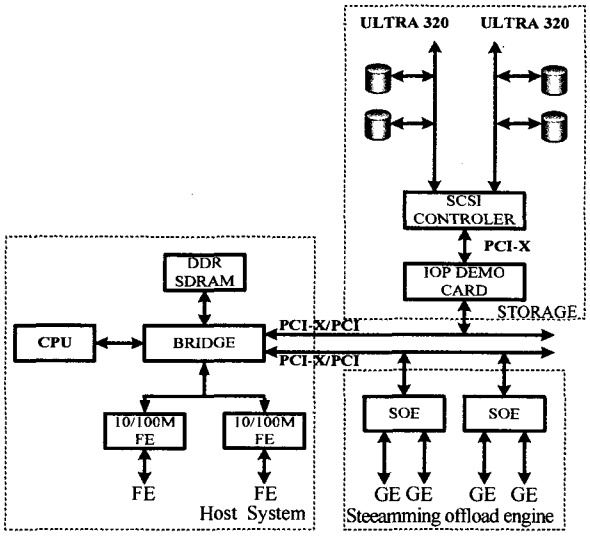

图 1 媒体服务器硬件架构

而在 IOP DEMO CARD 上，我们设计针对 SCSI CONTROLLER 的驱动程序，在这一级按照主机端设置好的 RAID 配置，针对服务器端传来的标准 SCSI 命令进行重新解释，并按照设定好的 RAID 策略，生成新的 SCSI 命令驱动接在 SCSI CONROLLER 端的各个磁盘。在这里，我们实现了各种 RAID 算法和 RAID 管理策略[2]。

综合上述因素，考虑技术的发展在 IOP DEMO CARD 上进行开发能很好满足系统对存储部分的需求，系统总体结构如图 2 所示。

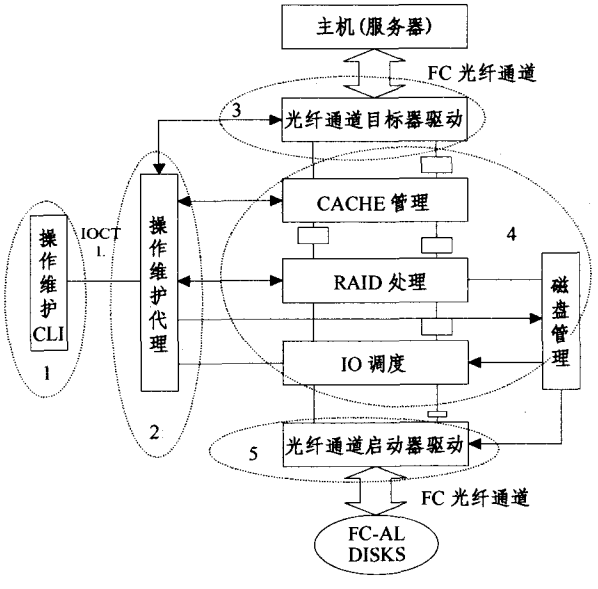

图 2 媒体服务器软件设计架构

存储部分管理的对象主要由两部分组成：系统数据区和媒体数据区。对系统数据区的管理和普通存储管理方法没有什么不同，采用安全和性能都比较出色的 RAID1 实现，与流媒体有关的数据存放在专用的区域，依靠自行设计的 RAID 系统实现上述模型(图 1)中虚线框内的部分。

SCSI 驱动模块主要完成 SCSI 控制器的初始化配置，SCSI 命令的产生、接收、解析等功能，是直接进行操作 SCSI 操作的部分。SCSI 芯片配置部分软件由 LSI 提供，CACHE 管理磁盘 CACHE 是为磁盘扇区设置的缓冲区，当出现一个特定扇区的数据请求时首先进行检测以确定该数据是否在缓冲区中，因此缓冲区对提高性能有很关键的作用。CAHE 管理主要完成缓冲区资源的管理数据淘汰算法的实现、RAID 的数据回写等功能。RAID 磁盘存储资源在物理上被划分为系统资源区和媒体数据区系统资源区，存放除流媒体数据文件以外的所有数据系统区的管理没有特殊性，采用比较可行的 RAID-1 方式。RAID-1 是通过临时复制所有数据来实现冗余，写请求可以由被包含请求数据的任何一个磁盘提供服务，同时写请求对两个响应的条带都进行更新，当一个驱动器失败时可以从第二个驱动器访问到。数据的重构由于 RAID-1 和 RAID-H 算法都使用的冗余和校验措施，因此当发生某个磁盘数据失效时应启动数据重构过程。具体做法是通过校验数据和，为损坏的其它数据恢复被损失的数据并将计算得到的数据写入到备份的磁盘中，磁盘 I/O 调度模块是对性能影响比较大的一个模块，磁盘的读写过程是一个包含寻道、输出等一系列动作的过程。在多任务请求环境中操作系统会为每一个 I/O 设备维护一个请求队列，那么对于一个磁盘队列中可能有来自多个进程的多个 I/O 请求，如何实现系统的最优配置使实现效率最高，是一项比较关键的技术。

4 关键问题及技术

RAID 模块存储系统是流媒体服务器要解决的一个关键技术，从目前市场上所能提供的 RAID 卡来看能够支持四通道 SCSI160 的产品很多，但实际能达到的性能尤其是读性能不是很高，经分析主要有存在以下几个原因：

1. RAID 卡所能支持的 STRIP SIZE 最大只有 128kB，其主要的考虑是均衡读写两方面的性能，按照这种模式系统每次 I/O 操作的效率不是很高，单个硬盘的实际带宽利用率没有充分发挥[3]。

2. 处理器性能不高，PCI 总线带宽成为瓶颈，基于上述原因提出了通过选用高性能 IOP 和专用 RAID 算法来解决存储读带宽的瓶颈问题，定义专用 RAID 算法简称为 RAID-H 算法，实际上它在很大程度上像一个 RAID-5 的组织结构，在 RAID5 算法中，数据块被进行条带化(STRIP)，组织并以条带为基本单位进行校验和的计算，校验和被平均存放到磁盘阵列的所有磁盘上。当一次 I/O 写入的数据块(称为 BLOCK)大于条带的大小时，奇偶校验可以很容易得到，它只需要使用新的数据位进行计算，因此奇偶校验驱动器可以和数据驱动器一起并行地进行更新，从而不需要额外的读和写操作。在 RAID-5 的实现中对于大的数据块分片后的数据和校验块是分散存放在磁盘阵列的每个硬盘中，而 RAID-H 算法的实现是将分块后的数据写入同一个磁盘连续的空间中。所谓的连续是一个物理的概念，是指被指定的数据块在一个不需要跨越磁道的空间中，这样做法目的是为了将来在读取这块文件时能够增加磁盘输出数据的效率，但由此带来的问题是存储系统的写效率会下降很多，并会导致磁盘空间利用率的下降，因为每次写入一块 STRIP 大小的数据时就会计算一次校验和并更新到磁盘中，而不像 RAID-5 中只计算一次或几次的情况。同时为保证在一段连续的空间可能会损失一些局部的空间，关于校验数据的存放，做法是轮循法，即校验数据轮流存放在磁盘阵列的所有磁盘中，这样有利于平均分摊校验数据的读写瓶颈问题，并提高系统的可靠性[4]。

3. 文件块大小的制定：Linux 的文件系统能支持的最大块大小是 4kB，这是由 Linux 所使用的分页机制决定的。然

维普资讯 http://www.cqvip.com

而，以 4kB 为单位的磁盘读取效率是十分低下的。根据目前已有的数据，读取 4kB 数据所需的时间是 6.1ms，而连续读取 128kB 数据所需的时间是 6.9ms。数据量有 32 倍的差别，而所消耗的时间却只有 13.1%的区别。对于流媒体服务器，大小超过 64kB 的数据请求占全部请求数量的 98%以上[5,6]。

5 RAID 性能分析与测试

存储系统性能评价领域内有许多性能指标，本文采用的性能指标有：

(1)吞吐率[7,8]：是指存储系统在单位时间内能处理的 I/O 请求数量，有两种形式：I/O 速率，即每秒钟 I/O 访问次数；数据传输率，即每秒钟传输的数据量。在事务处理应用中追求高的 I/O 速率，在科学计算应用中则要求较高的数据传输率。吞吐率是系统管理维护人员最关心的性能指标，在本文的测试中采用数据传输率。

(2)响应时间[9,10]：指的是存储系统从接收访问请求到完成服务请求的时间。

RAID 系统性能可通过随机 Petri 网模型和排队模型进行分析；而系统性能测试通常采用大家公认的磁盘驱动器测试软件，如 Qbench 和 Iometer 等。

分块结构的网络附属存储系统排队模型如图 3 所示。

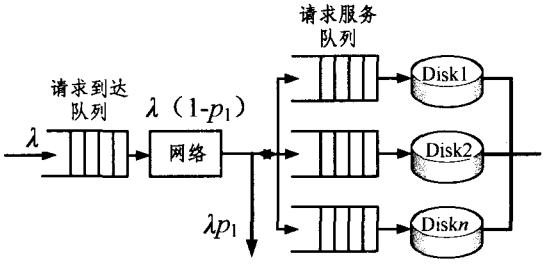

图 3 网络附属存储系统排队模型

其中 λ 为 I/O 请求到达率(若不是附网 RAID 系统，只需将图 3 中网络模块去掉即可)。

整个系统的响应时间由两部分组成：磁盘驱动器服务时间和网络(或队列)服务时间。

$$t_{disk}=\sum_{j=1}^{m}(t_s+t_{seek_{ij}}+t_{rota_{ij}}+b/r_d+b/r_s) \tag{1}$$

(1)磁盘驱动器服务时间

其中 t_{disk} 为磁盘驱动器服务时间，t_s 为磁盘驱动器串控制器启动时间间隔，t_{seek} 为磁盘驱动器寻道时间，t_{rota} 为磁盘驱动器旋转等待时间，b 为传输数据字节，i,j 表示第 i 个磁盘驱动器的第 j 次 I/O 服务，r_d 为磁盘驱动器的数传率，r_s 为 SCSI 总线数传率。

$$t_{seek}=\begin{cases}t_{posi}b & b\leqslant C\\ t_{posi}+\lceil b/C\rceil t_t 2_t & b>C\end{cases} \tag{2}$$

其中 t_{posi} 为磁头平均定位时间，$t_t 2_t$ 为磁头从一个磁道移到另一个磁道的时间，C 为磁道的数据容量，$\lceil b/C\rceil$ 为天棚函数取上整数。

$$t_{rota}=\begin{cases}t_{revo}/2 & b\leqslant C\\ 0 & b>C\end{cases} \tag{3}$$

其中 t_{revo} 为平均旋转时间。

(2)网络(或队列)服务时间

假设请求输入过程为泊松过程，由排队论 M/G/1 模型知，平均排队等待时间为：

$$\overline{T_{nw}}=\frac{\lambda E(T_f^2)}{2[1-\lambda E(T_f)]} \tag{4}$$

其中 T_f 为网络服务时间，而网络平均服务时间为

$$\overline{T_{ns}}=E(T_f) \tag{5}$$

网络服务时间为：

$$T_n=\overline{T_{nw}}+\overline{T_{ns}} \tag{6}$$

为了验证流媒体服务器系统的性能，本文在 100Mb 以太网上对其进行了性能测试。测试是将三台服务器通过 IP 交换机连接起来。采用一台带有高速网络接口的服务器作为对外的流媒体服务器，在该服务器的主板上接一块嵌入式 IOP 卡，IOP 卡上接有一块双通道 SCSI 控制卡，该控制卡上接有三个 SCSI 硬盘。在本嵌入式 Linux 2.4.17 系统上加载我们设计的 RAID 模块，使用 RAID-H 算法进行测试[11]。

为了生成 I/O 请求，更准确地监控和评估系统的 I/O 性能，我们分别在内核态和用户态下生成 I/O 读请求并对 I/O 的频率和次数进行了统计。

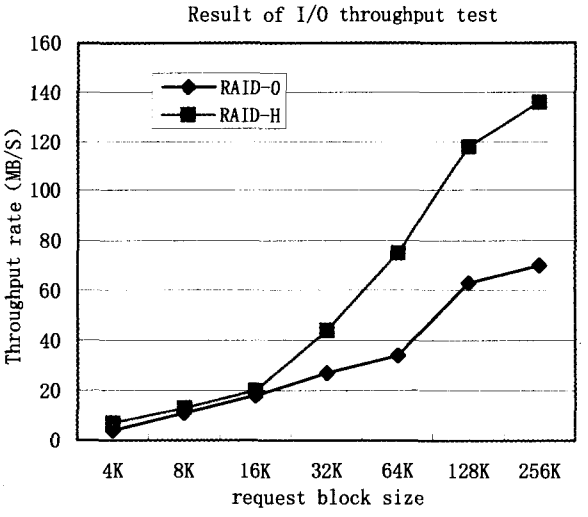

图 4 吞吐率测试结果

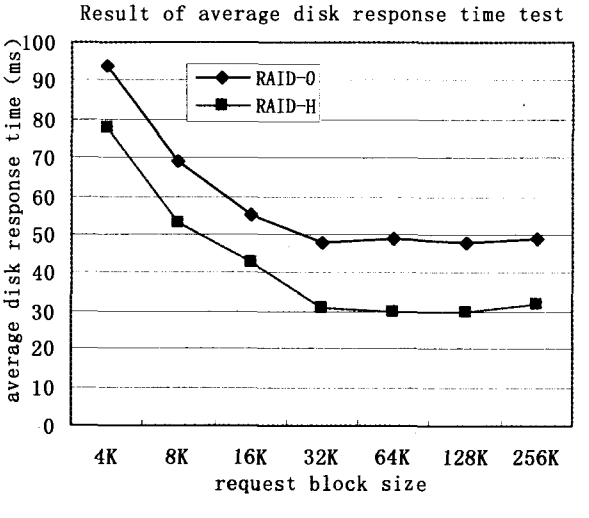

图 5 响应时间测试结果

测试方法：我们在 IOP 处理板上作实际的数据传输测试，测试数据通过 Linux 下自带的命令 iostat 和 vmstat 以及 top 来获得。这些数据是由操作系统内核自行统计出来的，因此可以作为客观的评判标准。

在数据传输率方面，随着块的增大，数据的传输率也随着

维普资讯 http://www.cqvip.com

增长。当块大小增长到256kB时，数据的传输率接近140MB/S，而且还有提高的趋势。

从图4和图5中可以看出，以128kB的块大小来传输时磁盘的平均响应时间最短。而且比LINUX系统下标准的以4kB块大小为传输单位的平均响应时间提高了近一倍，同时本系统使用的RAID-H算法比RAID-0的相应时间要快。

结束语 在流媒体服务器中采用RAID-H算法提高了I/O处理的能力，而传输块单位大小的改变则由于充分地利用了磁盘的内部数据传输带宽，使得存储子系统的性能得到了较大的提高。

参考文献

[1] Keeton-k, Katz-r-h. evaluating video layout strategies for a high-performance storage server. multimedia systems, 1995, 3(2):43-52

[2] Lee K, Kwon J B, Yeom H Y. Exploiting caching for realtime multimedia systems//Proc. of sixth IEEE Intl. Conf. on Multimedia Computing and Systems. Florence, Italy, June 1999:85-93

[3] Yokota H, Idoue A, Hasegawa T, et al. Link Layer Assisted Mobile IP Fast Handoff Method over Wireless LAN Networks [A]//Mobile Computing and Networking Conf [C]. 2002:1-5

[4] Patterson D A, Gibson G A, Katz R H. A case for redundant arrays of inexpensive disks(RAID)[C]//Proceedings of the International Conference on Management of Data(SIGMOD). June 1988

[5] Chen P M, Lee E K, Gibson G A, et al. RAID: high-performance, reliable secondary storage [J]. ACM Computing Surveys, 1994, 26(2):145-188

[6] Vadala D. Manag in RAID on LINUX [M]. O'Reilly & Associates, 2003

[7] Perkins C. IP Mobility Support for IPv4 [S]. RFC3344, 2002

[8] Gibson G A. Network attached storage architecture [J]. Communications of the ACM, 2000, 43(11):37-45

[9] Anderson D C, et al. Interposed Request Routing for Scalable Network Storage. ACM Transactions on Computer Systems, 2002, 20(1):25-48

[10] Chen P M, Lee E K, Gibson G A, et al. RAID: high performance, reliable secondary storage [J]. ACM Computing Surveys, 1994, 26(2):145-188

[11] Hennessy J L, Patterson D A. Computer architecture: a quantitative app roach, Third edition [M]. Elsevier Science P te L td, 2003: 392-448

(上接第76页)

下面对算法执行的基本情况进行一些简单的分析。

4.3 算法的基本分析

在边缘代理节点处，导致客户请求建立连接和发布订阅条件的情况可以分为如下四类：

• 一个全新的客户应用，第一次尝试和边缘代理节点B连接并提交订阅信息。这种情况和B已经与客户连接并且已经接受订阅消息的情况稍有不同，因为在后一种情况下，代理节点B已经知道该客户的存在；

• 客户休眠一段时间后，继续工作。这种情况下，需要缓冲由于客户暂时休眠而错过的通知，但是客户连接的边缘代理节点并没有改变；

• 和边缘代理节点Bnew连接的客户C是一个可移动客户，从旧边缘代理节点Bold处漫游到了Bnew处，并和Bnew连接。这时，Bnew接收到客户C中自动发送的一个订阅消息，其中包括客户C从Bold处接收到的最后一个通知的序号；

• 客户在一个边缘代理节点处暂停工作，然后移动，并和另外一个边缘代理节点连接，发送订阅消息，继续工作。这种情况和第三种情况类似，在此不作详细的讨论。

结束语 本文讨论了支持移动客户应用的分布式订阅/通知框架的体系结构和通知传递路径的重构过程，通过订阅消息的重新发布、Fetch和Replay消息的传递，不但重构了新的消息传递路径，而且把原来发送给应用的通知消息按照正确的时序传递给了客户应用。

该消息发布/订阅框架同时也能适应客户应用从多个数据源处订阅数据，具有良好的可扩展性和适应性。

参考文献

[1] Eugster P T, Felber P A, et al. The many faces of publish/subscribe [J]. ACM Computing Surveys, 2003, 35: 114-131

[2] Costa P, Migliavacca M, et al. Algorithms for reliable content-based publish-subscribe: An evaluation [C]//Proceeding of the ICDCS 2004. Tokyo: IEEE Computer Society, 2004:552-561

[3] Pietzuch P R. Hermes: A scalable event-based middleware. Ph. D. Thesis. University of Cambridge, 2004

[4] 马建刚，黄涛，等. 面向大规模分布式计算发布订阅系统核心技术[J]. 软件学报，2006，17(1)：134-147

[5] 汪洋，魏峻，等. 可扩展和可配置事件通知服务体系结构[J]. 软件学报，2006，17(3)：638-648

[6] Ergen M, Puri A. MEWLANA-mobile IP enriched wireless local area network architecture [C]// Proceeding of Vehicular Technology Conference. IEEE Computer Society, 2002: 2449-2453

[7] Sutton P, Arkins R, et al. Supporting disconnectedness transparent information delivery for mobile and invisible computing [C]// First International Symposium on Cluster, Computing and the Grid. Brisbane, Australia, May 2001:277-287

[8] Cugola G, Dinitto E, et al. The JEDI event-based infrastructure and its application to the development of the OPSSWFM [J]. IEEE Transaction on Software Engineering, 2001, 27(9):827-850

[9] Fiege L, Gartner F, et al. Supporting mobility in content-based publish/subscribe middleware [C] // Proceeding of the 4th ACM/IFIP/USENIX International Conference on Middleware. Springer-Verlag, 2003:103-122