流媒体点播的服务器通道调度方案综述*)

覃少华[1] 李子木[2] 蔡青松[1] 胡建平[1]

(北京航空航天大学计算机学院601信箱 北京100083)[1] (清华大学信息网络工程研究中心 北京100084)[2]

摘 要 服务器通道调度技术是解决大规模流媒体点播应用系统资源瓶颈问题的有效途径。它的基本思想是通过让用户尽可能共享同一个数据流来提高系统资源的利用率。本文综述了媒体点播系统中通道调度方案的研究现状，分析了一些典型方案的设计思想和方法，指出目前通道调度方案研究中的不足，并对今后该领域需要进一步研究的问题进行展望。

关键词 流媒体点播，调度方案，流合并，带宽，启动延迟

Scheduling Schemes of Server Channel for Streaming Media-on-Demand: A Survey

QIN Shao-Hua[1] LI Zi-Mu[2] CAI Qing-Song[1] HU Jian-Ping[1]

(School of Computer Science, Beijing University of Aeronautics and Astronautics, Beijing 100083)[1]

(Network Research Center of Tsinghua University, Beijing 100084)[2]

Abstract Scheduling technology of server channel is an effective approach for addressing the bottleneck problem of resources in large scale streaming media application systems. To reach higher level of system resource utilization, it lets users to share a single stream as much as possible. This paper summarizes the state-of-the-art in the research on channel scheduling schemes in media-on-demand system. We also analyze the design idea of several classical schemes. Additionally, some disadvantages of study on channel scheduling algorithms are discussed. At last, the key issues to be further studied in this area are pointed out.

Keywords Streaming media-on-demand, Scheduling schemes, Stream merging, Bandwidth, Start-up latency

1 引言

随着网络技术以及数字多媒体技术的日益发展成熟，基于互联网上的流式多媒体业务，如视频点播业务(VOD)、视频会议系统、远程教育系统、数字化图书馆、网上电子商城、视频网络游戏等有着强大的需求和广阔的应用前景。然而，由于流媒体本身具有数据量大、持续时间长、对传输延迟敏感等特点，在网络传输中需要较高的传输带宽，消耗更多的服务器资源。受硬件成本的限制，服务器性能及传输网络的带宽也不可能无限制地增长，因此，如何充分有效地利用系统的资源，提高流媒体传输系统的服务能力及服务质量就显得非常重要。

基于组播的服务器通道调度技术是提高流媒体应用系统资源利用率的一项关键技术。目前已经提出了许多具体的调度方案，按照服务模式的不同，可以分为两大类：静态通道调度方案和动态通道调度方案。前者一般采用服务器主动广播方式(推模式)，在服务过程中不考虑用户动态的访问行为；后者则采用服务器被动传输方式(拉模式)，媒体数据流的传输是由用户请求驱动的，服务器根据客户请求到达的情况动态选择相应的调度方案做出响应，使不同的用户尽可能共享同一个数据流，从而降低服务器带宽资源消耗。

本文首先对近年来提出的各种服务器通道调度方案进行分类总结，并简要分析一些主要方案的设计思想及其优缺点，在此基础上，展望未来该领域进一步研究的方向和重点。

2 静态服务器通道调度方案

基于对服务器网络带宽划分以及媒体对象分段方法的不同，现有的静态服务器通道调度方案可以分为三类：基于等通道带宽的调度方案(Equal bandwidth—EB)、基于等分段长度的调度方案(Equal segment length—ES)以及基于EB和ES混合方案。

2.1 基于等通道带宽的静态调度方案(EB)

基于等通道带宽的静态调度方案将长度为 T 的媒体对象分割为大小不等的 k 个数据段，每段的长度分别为 $S_1, S_2, \cdots, S_k$。服务器带宽被划分为 k 个带宽相等的逻辑通道，每一个数据段重复在相应的通道中进行广播。属于该类方案的有：金字塔广播方案(Pyramid broadcasting—PB)[1]、摩天大楼方案(Skyscraper Broadcasting—SB)[2]和Greedy Equal Bandwidth Broadcasting(GEBB)[5]等。

金字塔广播方案(PB)[1]将视频对象分成大小按几何级数增长的一系列数据片段，即 $S_{i+1}=\alpha S_i, i=1,2,\cdots k, \alpha \geqslant 2$，$\Sigma S_i = T$。不同媒体对象的编号相同的数据片段组合在一起成为一组，放在同一个逻辑通道中重复广播，如通道 i 重复广播每一个对象的第 i 个片段。为了能够实现连续的播放，客户端需要同时监听两个逻辑通道。在该方案中，媒体对象的第一个片段(起始片段)的长度最小，因而它重复广播的频率最高，有利于降低客户的启动延迟(startup latency)。假设服务器可用的网络通道带宽为 B(Mbits/s)，媒体对象的播放速率为 r(Mbits/s)，可以算出客户端最大的启动延迟及缓冲空间的大小如下：

$$t_{PB}=\frac{rTMk(\alpha-1)}{B(\alpha^k-1)} \tag{1}$$

$$Buff_{PB}=(S_k-\frac{rkS_k}{B}+S_{k-1}) \tag{2}$$

其中 $S_k=\frac{T\alpha^{k-1}(\alpha-1)}{\alpha^k-1}$，$\alpha=\frac{B}{rMk}$

*)基金项目：国家自然科学基金项目(60103005)资助。覃少华 博士后，研究方向为网络多媒体技术、高性能网络体系结构等；李子木 博士后，讲师；蔡青松 博士生；胡建平 教授，博导。

从式(1)可以看到，客户的启动延迟随服务器网络通道带宽的增加而呈指数减少。但是，为了让客户端能够连续播放整个媒体对象，每一个通道需要以很高的速率传输这些数据片段，因而对客户端的 I/O 带宽和缓存空间的大小要求较高。通常该方案需要客户端缓存空间大小超过视频对象长度的75%，并导致很高的磁盘传输率。

针对这个问题，文[2]提出了摩天大楼方案(SB)，它将服务器的网络带宽复用为带宽等于流媒体对象播放速率 r 的一组逻辑通道，假定有 k 个逻辑通道用于某个媒体对象的广播，则将该对象按序列：1,2,2,5,5,12,12,25,25…，分为 k 段，即 $S_2,S_3=2S_1,S_4,S_5=5S_1,\cdots$，段大小的增长比 PB 方案慢，每一个数据段都在一个固定的逻辑通道内按速率 r 重复广播。如图1所示，其中 $k=6$。客户端按照图1阴影部分所示的顺序接收相应的数据段，最多只需要同时从两个不同的逻辑通道中接收数据，就可以确保能够连续播放。客户端缓存空间的占用取决于最后一段的大小 S_k，而启动延迟则完全由第一段的长度来确定，相应的表达式如下：

$$t_{SB}=T/\sum_{i=1}^{k}s_i \tag{3}$$

$$Buff_{SB}=rs_k \tag{4}$$

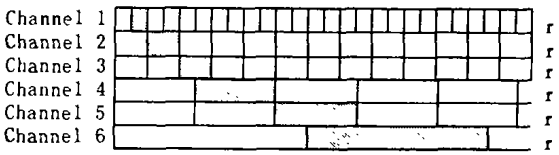

图1 Skyscraper Broadcasting ($k=6$)

从式(3)、式(4)可以看到，无论是客户最大启动延迟还是客户端缓存占用量，SB 方案都优于 PB 方案。对 SB 方案作了进一步优化的有：动态摩天大楼方案(Dynamic skyscraper)[23]、分区的摩天大楼发送策略(Partitioned skyscraper)[4]、扩展幂级广播(Extended exponential broadcasting-EEB)[3]等。

一种性能更好的方案是 GEBB 方案[5]，它允许客户请求到达后可以立即从多个通道中接收不同的数据段，而不必等到每个数据段的起始点才接收。该方案从给定的客户启动延迟出发，在保证客户端能够连续播放的前提下，试图从理论上推导出媒体对象各个分段的大小以及相应的广播通道的带宽，使需要的服务器总带宽最小。由于逻辑通道的带宽不等于媒体播放速率 r，因此，为了保证连续的播放，当前一个数据段播放结束时，后一个数据段应该接收完毕。基于这样的假设，文中得到一个非常有用的结果，即当服务器有无穷多个逻辑通道时，GEBB 方案需要的服务器带宽存在一个理论上的最小值：$r\log(T/t_{GEBB}+1)$，其中 t_{GEBB} 表示客户启动延迟。

2.2 基于等分段长度的静态调度方案(ES)

基于等分段长度的静态调度方案是将媒体对象分割为大小相等的 k 个数据段，为了取得较小的客户启动延迟，通常使 k 较大，因而需要较多的逻辑通道数，但是，由于这类方案是将服务器带宽划分为带宽依次递减的一组逻辑通道，因此，服务器带宽的消耗仅随媒体对象分段数的增加而缓慢增长。它的缺点是客户端需要同时从多个逻辑通道中接收不同的数据段，以满足连续播放的要求。这类方案主要有 Harmonic Broadcasting (HB)[6]、Cautious Harmonic Broadcasting (CHB)[7]、Quasi-Harmonic Broadcasting (QHB)[8]和 Poly-Harmonic Broadcasting(PHB)[9]等。

HB 方案被认为比 PB、SB 方案具有更高带宽效率的静态调度方案。它将每一个流媒体对象划分为大小相等的 k 个数据段：$S_1,S_2,S_3,\cdots,S_k$。即 $S_i=T/k,1\leqslant i\leqslant k$。进一步将第 i 段等分为 i 个子段$\{s_{i,1},\cdots,s_{i,i}\}$。这些子段被放在第 i 个通道(channel i)内进行周期广播，通道 i 的带宽为 $b_{HB,i}=r/i$，如图2所示，很容易求出分配给一个流媒体对象的总的带宽：

$$B_{HB}=\sum_{i=1}^{k}\frac{r}{i}=H(k)\times r,H(k)=\sum_{n=1}^{k}\frac{1}{n}$$

客户端从 Channel 1中下载并播放第一个数据段时，还需要同时从 Channel 2,…,Channel k 中下载其它数据段并缓存在客户端，使得第 i 个数据段开始被播放时，该数据段中至少前面 $i-1$ 个子段已经下载完毕，使客户端能够连续播放。在 HB 方案中，客户端需要等待的最大启动延迟为 $t_{HB}=T(1+(k-1)/k)/k$。可以算出 HB 方案要求客户缓存空间大小约为媒体对象长度的37%。该方案的一个缺点就是并不总是能够满足客户端连续播放的要求。

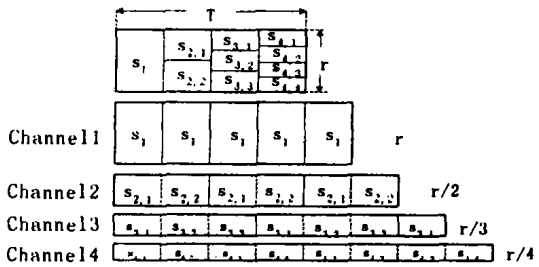

图2 Harmonic Broadcasting($K=4$)

为了改进 HB 方案中的缺点，文[7]提出了两种改进方案：CHB 和 QHB。这些方案虽然克服了 HB 方案中的缺点，但是，它们都不同程度地需要消耗更高的服务器带宽资源作为代价。

文[9]提出一种性能更好的方案(PHB)。与 HB 方案一样，PHB 仍将媒体对象均分为 k 段，需要 k 个通道分别来传输这些数据段，PHB 方案允许客户请求到达后立即开始接收数据，直到从 k 个通道中下载完时间间隔 mT/k(其中 $m\geqslant 1$)内对应的数据片段后，才允许客户开始播放第一个数据段。为了保证数据段 S_i 在被播放时能够及时下载到客户端的缓冲区中，需要分配给 Channel i 的带宽为 $b_i=r/(m+i-1)$，由此可以推出总的带宽：

$$B_{PHB}=\sum_{i=1}^{k}b_i=r\sum_{i=1}^{k}\frac{1}{m+i-1}=[H(k+m-1)-H(m-1)]\times r,m\geqslant 1$$

当 $m=1$时，$B_{HB}=B_{PHB}$。PHB 方案可以克服 HB 方案中的缺点，而且客户端启动延迟为 $t_{PHB}=T/k$，也比 HB 方案小。当 $m=1,k\geqslant 20$时，PHB 方案需要的客户端缓存空间约占到媒体对象长度的40%，与 HB 方案差不多。

2.3 基于 EB 和 ES 混合的静态调度方案

该类方案将媒体对象长度均匀分段，服务器的网络带宽也被划分一组带宽相等的逻辑通道。如 Fast Broadcasting (FB)[4]、Pagoda Broadcasting[10]和 New Pagoda Broadcasting[11]方案等。

FB 方案将可用的服务器带宽 $B=\beta\cdot r,\beta\geqslant 1$均分为 N 个逻辑通道，$N=\lfloor B/r\rfloor=\lfloor\beta\rfloor$，分别用{Channel 0,…,Channel $N-1$}来表示。将媒体对象均匀划分为 k 段$\{S_1,S_2,\cdots,S_k\}$，$k=\sum_{i=0}^{N-1}2^i=2^N-1$，然后将连续的 2^i 个数据段放在 Channel i 中重复进行广播($i=0,\cdots,N-1$)，如图3所示。如果客户端配置了足够的缓冲空间，则它可以在接收第一个数据段 S_1 时，同

时从其它通道中接收数据，直到接收完所有通道中的数据段为止。这样客户端需要等待的最大启动延迟的时间由传输第一个数据段 S_1 需要的时间决定，即 $t_{FB}=\frac{T\cdot r}{k}\cdot\frac{N}{B}=\frac{T}{2^N-1}\cdot\frac{N}{\beta}$。由于客户端在 $2^{N-1}\cdot t_{FB}$ 时间内可以完成所有数据段的接收，而这期间客户端已经播放的数据长度为 $2^{N-1}\cdot t_{FB}\cdot r$，因此客户端需要配置的最大缓存空间为 $Buff_{FB}=T\cdot r-2^{N-1}\cdot t_{FB}\cdot r=(1-\frac{2^{N-1}}{2^N-1}\cdot\frac{N}{\beta})T\cdot r$（约等于媒体对象长度的50%）；如果客户端没有配置缓冲空间，则任意时刻客户端只能从一个通道中接收数据，从图3可以看到，客户端按照阴影部分所示的顺序接收数据就可以满足连续播放的要求。为此客户端需要等待最大启动延迟为 $t_{FB\_no\_buffer}=2^{N-1}\cdot t_{FB}=T\cdot 2^{N-1}/(2^N-1)$，约等于媒体对象长度的一半。

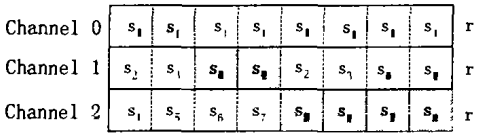

图3 Fast Broadcast($N=\beta=3,k=7$)

Pagoda Broadcasting(宝塔广播)[10]可以避免 HB 等方案中客户端需要同时管理多个数据流的困难，它通过降低靠后的数据段的广播频率而不是降低它们的传输带宽来实现。如图4所示，若媒体对象被等分为9段 $\{S_1,S_2,\cdots,S_9\}$，被映射到三个等带宽的通道上，这样对于一个持续时间为120分钟的媒体对象，客户端的最大启动延迟为14分钟，仅比 QHB 方案多3分钟，而换来的好处是客户端最多只需要同时管理3个数据流，远远小于 ES 类方案。但是该方案需要客户端配置的最大缓存空间比较大，接近于50%。

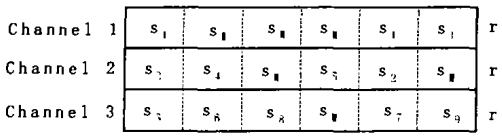

图4 Pagoda Broadcasting

为了进一步提高 Pagoda Broadcasting 方案的效率，文[11]提出了 New Pagoda Broadcasting(NPB)方案，它采用一种称之为矩形矩阵(Rectangular Matrix)的数据段到通道时隙的映射技术，这种映射技术能够使得每一个数据段 S_i 能够更准确地每隔 i 个时隙广播一次（即尽量地保证数据段广播的频率接近其必需的频率），这样在相同带宽资源的情况可以发送更多的数据段，从而降低客户的启动延迟。

从以上分析可以看到，基于 EB 的方案相对比较简单，但是启动延迟较大；而于基于 ES 的方案则要求客户同时从多个通道接收数；基于 EB 和 ES 混合的方案则介于前面两者之间。一般来说，静态通道调度方案的优点是：①结构简单，且不受用户动态行为的影响；②公平性高，对于所有的用户，系统提供的服务性能相似；③随着用户访问请求到达的增加，不会降低系统的服务性能。缺点是：①为用户提供的服务质量(QoS)不高，尤其是用户启动延迟比较长；②不支持用户交互操作(VCR 功能)；③客户端在接收数据时需要在多个通道中进行切换，容易造成同步丢失，而且在大多数情况下客户端需要配置较大的缓存空间。

3 动态服务器通道调度方案

与静态调度方案不同，动态服务器通道调度方案根据客户请求到达情况动态调度服务器通道。根据调度方式的不同，这些方案也可以大致分为三类：批处理方案、补丁方案和流合并方案。

3.1 批处理方案

批处理(Batching)技术[12]是一种最简单的动态通道调度方案，它将较早到达的客户请求进行适当的延迟后，把对同一个媒体对象的请求绑定在一个组播通道中，使更多的用户共享同一个组播流。为了进一步提高系统的吞吐量，相关的研究工作在批处理的基础上提出了一些有效的调度策略如 MFQL[13]、MDP[14]等。虽然批处理方案在一定程度上减少了服务器通道的消耗，且简单实用，但是却人为地增加了启动延迟。

3.2 补丁方案

在补丁方案中，用于组播整个媒体对象的通道称为常规通道，用于传输媒体对象部分数据(补丁)的通道称为补丁通道。这些通道的传输速率均等于媒体播放速率，客户最多需要同时从两个通道中接收数据流，因此客户端需要具备一定的缓存空间及两倍于媒体播放速率的接收带宽，这样客户在加入常规通道后，还可以同时接收补丁通道的数据，实现无延迟服务。典型的补丁方案主要有 Greedy patching[15,20]、Grace patching[15]、Optimized patching[16]、Controlled Multicast[21]和最优批处理补丁方案(OBP)[17]等。

各种补丁方案的不同之处在于如何控制常规通道的启动频度。例如，给定客户端的缓冲空间大小为 δ 时间单位，在 Greedy patching 方案中，只要常规通道中存在客户请求的媒体对象，补丁通道就会被启动。如果客户请求到达时刻超过了 δ 时间单位，则客户端只能缓冲最后 δ 时间单位的数据，导致了一个持续时间很长的补丁流。与此不同的是，在 Grace patching 方案中，如果客户到达的时刻离最近的常规流启动时刻超过了 δ 时间单位，则服务器将启动一个新的常规通道向客户提供服务。一般来说，Grace patching 比 Greedy patching 的性能更好，文[16,21]从理论上分别导出了最优的常规通道启动频率与客户缓冲空间大小、请求到达速率及媒体持续时间长度的关系式。对于最优的补丁策略来说，消耗的服务器带宽与客户请求到达速率的平方根成正比[21,22]。

文[17]将补丁方案和批处理方案结合起来，提出了最优批处理补丁方案(OBP)，该方案的基本服务过程如图5所示。

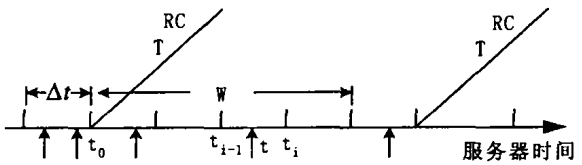

图5 OBP 服务模式示意图

当客户请求到达时刻 $t\in[t_{i-1},t_i)$ 时，如果 $t_i<t_0+W$（W 为补丁窗口的大小），则客户将在 t_i 时刻加入常规组播通道(RC)，同时，向服务器请求一次补丁，补丁大小为 t_i-t_0；如果 $t_i\geq t_0+W$，则服务器启动一个新的常规组播周期。令 $W=N\Delta t(N\geq 1)$，每个批处理区间 Δt 内无客户请求到达的概率为 $p=e^{-\lambda\Delta t}$。如果在第 i 个批处理期间有客户请求到达，则需要的补丁大小为 $i\cdot\Delta t$。由此导出一个常规组播周期内需要的补丁大小的均值为：

$$\mu=\Delta t\sum_{i=1}^{N}i(1-p)=\Delta t(1-p)\frac{N(N+1)}{2}$$

两个相邻的常规组播间隔的均值为：

$$I=W+\Delta t\sum_{i=1}^{\infty}i(1-p)p^{i-1}=W+\frac{\Delta t}{1-p}$$

服务器并发流个数为：

$$\frac{R}{r}=\frac{\mu+T}{I}=\frac{(1-p)W^2+(1-p)\Delta tw+2\Delta tT}{2\Delta tw+\frac{2(\Delta t)^2}{1-p}}$$

由上式解出最优的补丁窗口：

$$W_{opt}=\frac{-\Delta t+\sqrt{p(\Delta t)^2+2(1-p)\Delta tT}}{1-p}$$

文[17]的结果表明，OBP 方案比最优补丁方案能够更有效地降低服务器的带宽消耗，并且随客户请求到达速率的变化具有更好的适应性。

3.3 流合并方案

流合并技术（Steam Merging）是在补丁方案、Dynamic skyscraper[23]等技术的基础上提出来的服务器通道动态调度技术。与补丁技术不同的是，流合并技术允许不同时刻开始的同一个媒体对象的组播流之间不断地进行合并，最终合并到最早开始的那个组播流（该组播流称为当前合并树的根流——Root Stream，其长度等于媒体对象的长度）。流合并方案主要有三种：Early Merging[24,25]（其中又有三个变种：Earliest Reachable Merge Target—ERMT，Simple Reachable Merge Target—SRMT，Closest Target—CT）、Dyadic Tree[26]和 Fibonacci Tree[27]。这些合并方案的不同之处在于如何创建一个有效的合并树以及确定各个数据流的合并顺序，使得服务器消耗的带宽最小。

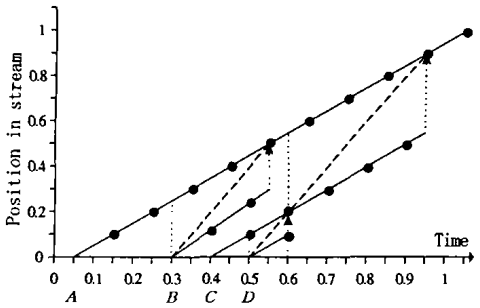

图6 最优合并过程

文[25]提出的 Early Merging 方案的基本思想是让两个相邻的数据流尽早合并，并一直进行下去。基于如何确定一个合并目标流，该文提出了三种不同的策略——ERMT、SRMT 和 CT。对一个新到达的客户请求或者是一组刚刚完成合并的客户，ERMT 总是试图计算出一个最早可达的合并目标流，如图6所示，客户 B 将在时刻0.3处监听并接收 A 的数据流，C 监听 B 的数据流并在时刻0.5处与 B 合并，使 D 无法赶上 C。因此，对于 D 来说，B 的数据流就是 D 的最早可达的合并目标流，这样 D 从时刻0.5处开始监听并接收 B 的数据流，一直到0.95处与 A 的数据流合并；SRMT 则直接根据各数据流起始的时间距离来确定最早可达的合并目标流，这样在图6中，客户 D 的最早可达合并目标就是 A 的数据流；而 CT 在选择合并目标流时，并不关心该目标流是否可达，而是直接选择距离最近的一个数据流作为合并的目标流，如果所选择的目标不可达，则继续使用 CT 进行选择。文中基于泊松请求到达序列的仿真结果表明，Early Merging 方案的性能比摩天大楼广播方案、最优补丁方案更好，并且接近于最优的离线流合并方案（需要事先知道请求到达情况，并通过动态规划方法算出最优的合并过程）。

Dyadic Tree 方案[26]使用递归的间隔划分（recursive interval partitioning）方法来确定不同时刻到达的客户接收数据流的过程。如图7所示，对于一个给定的根节点，其起始时刻为 x，区间 $(x,y]$ 表示在该区间内到达的客户请求最终都可以合并到根节点的数据流中，而超过 y 时刻到达的客户请求将创建一个新的合并树。首先将 $(x,y]$ 分成两个部分，然后将其左边部分再分成两部分，依此进行下去，直到左边部分没有客户请求为止，从而得到一系列间隔 $I_1,I_2,\cdots,I_N$。在这些间隔中第一个到达的客户请求就作为当前合并树的子节点，同时也作为合并子树的根节点。按照上述方法对各间隔再次进行划分，确定各子树的子节点，直到所有请求均被划分到相应的子树中为止。例如，在间隔中 $(0,T/2]$，在时刻0启动一个根流（Root stream），作为当前合并树的根节点，间隔划分方案的执行过程如图8所示，其结果产生如图9所示的合并树及各个流的合并过程。

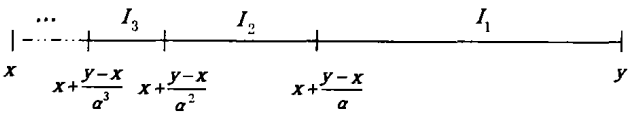

图7 间隔划分方法

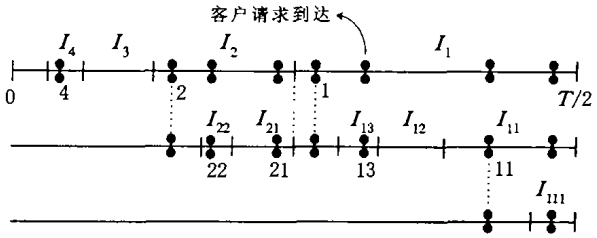

图8 间隔划分算法执行过程

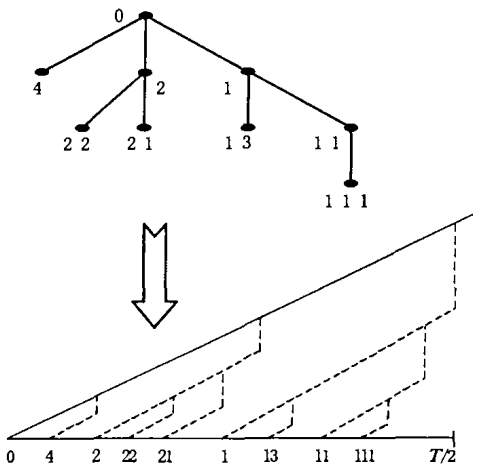

图9 合并树及各个流的合并过程

该文中取 $\alpha=2$，通过分析合并树中所有流的总长度（称合并代价）来度量 Dyadic Tree 方案的性能，并证明了 Dyadic Tree 方案在客户请求到达服从泊松分布的情况下，其合并代价 $EL(T,\lambda)$ 的上、下限满足如下不等式：

$$\frac{1}{4}T\log\lambda-\frac{1}{4}L\leqslant EL(T,\lambda)\leqslant\frac{3}{4}T|\log\lambda|+\frac{23}{8}T,$$

其中 T 为媒体对象长度，λ 为客户请求到达速率。

Fibonacci Tree 方案[27]使用 Fibonacci 数列来创建合并树，用0作为合并树的根节点，其中各子树的节点数刚好是一个 Fibonacci 数，第 i 个子树是由第 $i-1$ 个子树和第 $i-2$ 个子树建立的。客户请求到达后按前序遍历的方式播入相应的子

树中，从而确定客户端接收数据流的过程。与 Dyadic Tree 方案一样，当客户请求到达时刻与合并树中的根节点数据流开始的时刻超过媒体长度的一半($T/2$)时，将重新建立一个合并树。文中的分析和实验结果表明，Fibonacci Tree 方案在两种不同的请求到达序列(连续请求和泊松请求)下，都具有较好的性能。

文[28]对 ERMT、Dyadic Tree 和 Fibonacci Tree 三种方案进行了比较分析，结果表明，三种方案的性能都比较接近于最优的离线合并方案，其中 ERMT 的性能最好，Dyadic Tree 方案次之，而 Fibonacci Tree 稍差；而从系统复杂性的角度看，Dyadic Tree 方案最为简单，Fibonacci Tree 次之，而 ERMT 显得稍为复杂一些，因为它并不是在客户请求到达后就确定好客户接收数据流的过程，而是逐步确定并不断地动态改变，这种灵活性使得它具有更好的性能。由此可见，Dyadic Tree 方案是系统复杂性与性能的一个最好的折衷。

动态调度技术的优点：①可以很容易实现零延迟服务；②更好地支持用户的 VCR 操作；③客户端最多只需要同时从两个通道接收数据。缺点是：①公平性差；②服务器消耗的带宽随客户访问量的增加而增大，当系统服务量达到饱和之后，性能将急剧下降。

4 媒体点播通道调度技术展望

近年来，对流媒体点点播的通道调度技术研究已经取得了上述若干成果，总结目前研究的现状，我们认为还需要在以下方面作进一步的研究。

(1)设计理论模型。在媒体点播系统中，通道调度方案的设计通常需要考虑以下四个方面的基本问题：①服务器带宽消耗；②客户端启动延迟；③客户端的 I/O 带宽消耗；④客户端缓存空间占用。到目前为止，已经提出的这些方案大都侧重于解决部分问题，没有一种方案的性能在上述四种情况下都表现良好，原因是这些方案在设计的时候就缺少有效的理论指导及合理的性能分析与评价模型。因此，需要在设计理论模型上作进一步的研究和探讨。

(2)确定有效的调度方案[29]。一般来说，静态调度方案适用于分送最流行的媒体对象，而动态调度方案则适用于分送具有中等流行度或较低流行度的媒体对象。因此，对于具有不同流行度的媒体对象，系统应能够自动调用不同的调度方案提供服务，才能有效提高系统效率。然而，对流媒体对象流行度的度量及分类方法仍然没有一个合理有效的模型，仅从某一个时间段内的平均“点击量”来判断并不能准确反映媒体对象的流行程度。因此，如何确定有效的调度方案仍需要进一步研究。

(3)支持 VBR 编码的通道调度方案。前面提到的通道调度方案均假定流媒体对象是采用 CBR 编码，这样的假定使得系统分析及设计大大简化。然而对于相同质量的画质，采用 VBR 编码的流媒体的平均位率比 CBR 编码低，可以节省更多的网络传输带宽，所以，在实际系统中大部分的媒体对象均采用 VBR 进行编码。而目前针对这种编码的通道调度方案研究仍然比较少。

(4)VCR 功能的支持。媒体点播系统中的一项重要功能就是提供 VCR 操作(快进、快退、暂停等)，在静态调度方案中，要实现 VCR 功能比较困难，容易造成数据丢失或播放中断。与此不同的是，在动态调度方案中，实现 VCR 功能相对比较容易，但是许多方案都没有分析提供 VCR 操作对方案本身性能的影响，特别是当这种操作比较频繁时。

(5)服务器通道分配及无缝切换问题。在静态通道调度方案中，往往假定在服务过程中对每一个媒体对象都分配一个固定的通道数，而不管媒体对象访问负载的变化情况(流行度随时间变化)。事实上，要想取得更好的带宽效率，服务器的通道分配应能够随着访问负载的变化而动态改变。文[30]以 FB 方案为基础，提出了一种无缝的动态改变通道数的分配方法。文[31]提出了一种无缝的负载自适应广播框架，并引入 Fibonacci 数列作为媒体对象的分段方案，对于流行或不流行的媒体对象都表现出较好的性能。但是这种框架能否应用到其他静态调度方案，以及各种方案之间的性能比较仍然需要进一步研究。

(6)分布式通道调度方案。在网络的边缘部署代理服务器，构建分布式的流媒体分送系统是提高多媒体数据流在网络中传输质量和效率的有效方法。在这样的系统中，需要解决流媒体服务器与代理之间、代理与客户端之间的传输通道调度问题。目前，这方面研究已经有了初步的结果，文[32]提出将媒体对象的前缀部分缓存在代理中，后缀部分采用周期广播的调度方案；文[18,19]则在前缀缓存的基础上，对后缀部分采用批处理补丁调度方案结合代理动态缓存。这些方案不仅大大降低客户访问的启动延迟，节省了主干网络的传输带宽，而且还具有很好的可扩展性。然而，现有的研究工作大都只考虑了在单个代理服务器情况下的调度方案的优化问题。而对于有多个代理以及代理间的协作乃至更复杂的情况则有待进一步研究。

(7)调度方案的具体实现。虽然已经提出许多调度方案，但是大多都还只是停留在理论分析和性能仿真的层面上，在具体应用方面仍然很少。在实践中，仍有许多工作需要进一步完善和加强。

总结 无论服务器还是网络，它们的带宽、磁盘 I/O 等资源终归还是有限的，因而只能支持有限的并发数据流。另一方面，流媒体数据的传输需要消耗较大的资源，而且持续的时间长，如何有效地节约这些资源的消耗，最大限度地提升系统的吞吐量，是大规模流媒体应用系统设计时需要考虑的一个主要问题。通道调度方案的研究是解决这些问题的一个有效途径。本文分析并总结了近年来提出的一些主要通道调度方案的设计思想，从中可以看出，调度方案的设计实际上可以归纳为一个多目标的优化过程，需考虑网络带宽资源、服务启动延迟、客户端 I/O 带宽、缓存空间占用、服务质量等诸多方面因素，才能设计出满足不同应用系统要求的通道调度方案。

参 考 文 献

1 Viswanathan S, Imielinski T. Pyramid broadcasting for video on demand service. In: IEEE Multimedia Computing and Networking Conf. San Jose, California, 1995, 2417: 66～77

2 Hua K A, Sheu S. Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems. In: Proc. of ACM SIGCOMM'97, Cannes, Fance, Sept. 1997. 89～100

3 鄢祥，高远. 一种新的视频点播方案——扩展幂级方案. 计算机研究与发展，2002，39(7)：864～970

4 Eager D L, FerrisM C, Vernon M K. Optimized regional caching for on demand data delivery. In: P roc. of MMCN '99, San Jo se, CA. Jan. 1999. 301～316

5 Hu A, Nikolaidis I, Van Beek P. On the design of efficient video-on-demand broadcast schedules. In: Proc. 7th Int'l Symp. On Modeling, Analysis and Simulation of Computer and Telecommunication Systems, Maryland, MD, Oct. 1999. 262～269

6 Juhn L S, Tseng L M. Harmonic broadcasting for video-on-demand service. IEEE Transactions on Broadcasting, 1997, 43(3): 268～271

7 Paris J F, Carter S W, Long D D E. Efficient broadcasting protocols for video on demand. In: Proc. of MASCOTS'98, July 1998. 127～132

(下转第110页)

关是绝对需要的；

②大多数电子支付系统都是封闭式的，即使用专有技术，仅支持一些特定集合的协议和机制。这些支付系统通常需要一个中央服务器作为所有参与者的可信第三方，有的甚至要求使用特定的服务器或浏览器；

③尽管大多数方案都使用了公钥密码，但多方安全受到的关注远远不够，消费者的匿名性和隐私也还未得到充分的考虑。大多数系统都限制为两方，因此难于集成一个安全连接到第三方，且没有建立一种解决争议的决策程序；

④大多数系统都将销售商服务器和消费者浏览器间的关系假设为主从关系，这种非对称关系限制了在这些系统中执行复杂的协议，而且不允许消费者之间进行直接交易；

⑤所有方案和产品都仅考虑了在线销售，但很少考虑多方交易问题（如拍卖）和公文交换问题（如签合同和可证实电子邮件）等。

结论 本文旨在研究电子支付系统的安全性，我们首先给出了电子支付系统的安全需求：消息传输的机密性、支付交易的安全性、支付业务的不可否认性、交易信息的完整性和支付业务的多边性，从基于信用卡的电子支付系统、基于电子支票的电子支付系统、基于电子现金的电子支付系统、微支付系统和移动支付系统五个方面给出了当前各种安全电子支付系统的基本模型、主要特点及实例代表。最后，对安全电子支付系统的发展现状及未决问题进行了深入的分析和评述，指出了该领域未来的研究方向。

参 考 文 献

1 O'Mahony D, Peirce M, Tewari H. Electronic payment systems for E-commerce. Boston: Artech House, 2003

2 Kalden R, Meirick I, Meyer M. Wireless Internet acess based on GPRS. IEEE Personal Communications, 2000, 7 (2): 8～18

3 Rivest R L, Shamir A. PayWord and MicroMint: Two simple micropayment schemes. In: Lomas M, ed. Security Protocols--International Workshop, Berlin: Springer-Verlag, 1997

4 Lang Weimin, Yang Zongkai, Liu Gan, et al. A New Efficient Micropayment Scheme Against Overspending. In: J. Carlsen ed. Proc. of The Ninth IEEE Symposium on Computers and Communications (ISCC 2004). Los Alamitos, California: IEEE Computer Society, 2004. 137～143

5 Yang Zongkai, Lang Weimin, Yun Mengtan. A New Fair Micropayment System Based on Hash Chain. In: Soe-Tsyr Yuan, Jiming Liu eds. Proc. of IEEE Intl. Conf. on e-Technology, e-Commerce, and e-Service (EEE'04), Los Alamitos, California: IEEE Computer Society, 2004. 139～145

6 郎为民，杨宗凯，谭运猛. 一种可提升用户匿名性的离线电子支付方案. 计算机工程，2004，30 (1)：30～32

（上接第105页）

8 Pâris J F, Cater S W, Long D D E. Effocoent broadcasting protocols for video on demand. In: Proc. of MASCOTS'98, July 1998. 127～132

9 Pâris J F, Cater S W, Long D D E. A low bandwidth broadcasting protocol for video-on-demand. In: Proc. of IC3N'98, Oct. 1998. 690～697

10 Pâris J F, Cater S W, Long D D E. A hybrid broadcasting protocol for video-on-demand. In: Proc. of MMCN'99, Jan. 1999. 317～326

11 Paris J F. A simple low-bandwidth broadcasting protocol for video-on-demand. In: Proc. of IC3N'99, Oct. 1999. 118～123

12 Dan A, Sitaram D, Shahabuddin P. Schduling policies for an on-demand video server with batching. In: Proc. of ACM Multimedia. San Francisco, California, Oct. 1994. 15～23

13 Aggarwal C C, Wolf J L, Yu P S. The maximum factor queue batching scheme for video-on-demand systems. IEEE Trans. Computers, 2001, 50(2): 97～110

14 杨灿，徐重阳，刘政林. VOD系统批处理调度策略优化研究. 计算机学报，2002，25(11)：1263～1268

15 Hua K A, Cai Y, Sheu S. Patching: A multicast technique for true video-on-demand services. In: Proc. of ACM Multimedia, Britstol UK, Sept. 1998. 191～200

16 Cai Y, Hua K A, Vu K. Optimizing patching performance. In: Proc ACM/SPIE Multimedia Computing and Networking. Jan. 1999. 203～215

17 White P P, Crowcroft J. Optimized batch patching with classes of service. ACM Communications Review, 2000, 30(4)

18 Venkatramani C, Verscheure O, Frossard P, Lee K W. Optimal proxy management for multimedia streaming in content distribution networks. In: ACM NOSSDAV 2002, Miami Beach, FL, USA, May 2002. 147～154

19 Frossard P, Verscheure O. Batch patch caching for streaming media. IEEE Communications Letters, 2002, 6(4): 159～161

20 Carter S W, Long D D E. Improving video on demand server efficiency through stream tapping. In: Proc. IEEE ICCCN'97. Las Vegas, NV, Sept. 1997. 200～207

21 Gao L X, Towsley D. Supplying instantaneous video on demand services using controlled multicast. In: Proc. IEEE Int. Conf. Multimedia Computing and Systems, 1999, 2: 117～121

22 Eager D L, Vernon M K, Zahorjan J. Minimizing bandwidth requirements for on-demand data delivery. IEEE Trans. on Knowledge and Data Engineering, 2001, 13(5): 742～757

23 Eager D L, Vernon M K. Dynamic skyscraper broadcasts for video-on-demand. In: Proc. of MIS'98, Istanbul, Turkey, Sept. 1998

24 Eager D L, Vernon M K, Zahorjan J. Minimizing bandwidth requirements for on-demand data delivery. In: Proc. 5th Int'l Workshop on Multimedia Information Systems, Indian Wells, CA, Oct. 1999. 80～87

25 Eager D L, Vernon M K, Zahorjan J. Optimal and efficient merging schedules for video-on-demand servers. In: Proc. of ACM MULTIMEDIA'99, Orlando, FL, Nov. 1999. 199～203

26 Coffman E G, Jelenkovic P, Momcilovic. Provably efficient stream merging. In: Proc. the 6th Intl. Workshop on Web Caching and Content Distributio, 2001

27 Bar-Noy A, Ladner R E. Competitive on-line stream merging algorithms for media-on-demand. In: Proc. the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2001. 364～373

28 Bar-Noy A, Goshi J, Ladner R E, Tam K. Comparison of stream merging algorithms for media-on-demand. In: Proc. Conf. on IS&T/SPIE MMCN'2002, San Jose, CA, Jan. 2002. 115～129

29 Poon W-F, Lo K-T, Feng J. Determination of efficient transmission scheme for video-on-demand (VOD) services. IEEE Trans. on Circuits and Systems for Video Technology, 2003, 13(2): 188～192

30 Tseng Y-C, Yang M-H, Hsieh C-M, et al. Data broadcasting and seamless channel transition for highly demanded videos. IEEE Trans. on Communications, 2001, 49(5): 863～874

31 Guo Y, Gao L, Towsley D, Sen S. Seamless workload adaptive broadcast. In: Proc. of Intl. Packetvideo Workshop, Pittsburgh, PA, April 2002

32 Guo Y, Sen S, Towsley D. Prefix caching assisted periodic broadcast: framework and techniques to support streaming for popular videos. In: Proc. of IEEE ICC'2002, April 2002