基于 Internet 的视频代理缓存服务器设计的若干关键问题*)

杨书慧　顾铁成　陈道蓄
(南京大学计算机软件新技术国家重点试验室　南京210093)
(南京大学计算机科学与技术系　南京210093)

Key Issues of the Design of Proxy Cache Server for Video Data in the Internet Environment

YANG Shu-Hui　GU Tie-Cheng　CHEN Dao-Xu
(State Key Laboratory for Nevel Software, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract With the boom of the network technology, all kinds of services based on the Internet come into being. Among them, the most promising one is the video service. All of this kind of service has been paid much attention, such as videoconference and video-on-demand. Because of the instability of the Internet environment, the traditional Client/Server mode in the LAN, which is used to support the data transmission, is not suitable any longer. And the newly designed Server/Proxy/Client one is more effective. This paper describes this architecture and its service pattern, does some deep research on the key issues of the design of the proxy cache server, such as the caching and replacement of the video data, the management and coordination of multi-proxy-servers.

Keywords Proxy cache, Video service, Replacement policy, Layered encoded video

1　概述

在网络技术迅猛发展的今天，人们已不满足于从网上获得传统的文本格式的信息了，多媒体，尤其是视频(Video)信息越来越成为热门所选，它在网上信息传送中所占份额也越来越大。另一方面，由于网络应用的增长，以及诸如 Internet 这样的广域网的完全分布特性(即没有任何的集中控制机制)，网络拥塞现象也愈演愈烈。各种解决方法应运而生。例如镜像站点(mirror sites)方法，就是选取若干地点放置服务器的完全复本以减少远程传输延迟。这种方法已被广泛采用，但是因为需要与原站点完全相同的硬件设备，所以花费代价较大。

另一种方法就是代理缓存(proxy cache)，它是 Web cache 的一种，就是利用缓存(cache)的思想，将服务器端的信息尽量拉到靠近客户端的地方，如客户所处局域网的边缘处，但是由于缓存的容量有限，因此只能存储一些相对需求频率较高的"热门"信息，并且还要采用一定的置换策略来保证缓存的高命中率。这种方法可以减少主干广域网上的带宽需求，从而缓解日益严重的网络拥塞现象，同时，由于将高存取率的信息放到客户所在的局域网内，使用户获取信息的速度大大提高；对于源服务器来说也减小了它的负载；当广域网或源服务器出现故障时，代理缓存可以继续提供一定程度的服务，加强了整个系统的健壮性。一些知名的系统如 Harvest[1]，Squid[2]就是这方面的技术研究的成果，专门的研究文献还有文[3～5]等。

无论是 Harvest 还是 Squid 考虑的都是基本的网络数据类型，即文本类型，HTML 格式以及各种格式的静态图像。而对于越来越热门的视频格式的信息并没有给予单独的考虑。视频信息有着数据量大，需要实时传送的特点，因此对网络传输的要求更高。传统的 proxy cache 的设计不能满足它的要求。例如，若不加改动地使用原来的 cache 方法，有限的 cache 空间将很快就被少数几个视频文件充满，使得其命中率很低，置换频繁，从而整个系统的效率很低。所以很需要为这种特别的数据格式设计相应的 proxy cache 策略。本文就这一问题进行了较为深入的研究，对现有的各种视频传送系统中代理缓存服务器的设计框架，视频数据存放策略，置换策略以及代理缓存服务器群的协同运作等进行了分析比较，并就这项技术今后的研究方向进行了探讨。

2　基于 Internet 的视频传送系统抽象模型

我们这里所讨论的代理缓存服务器系统是基于 Internet 环境的，不是以往的局域网内的视频传送系统。由于 Internet 不能像局域网那样可以进行有效的 QoS 管理，因此为了提高系统的服务质量，我们提出了一种新的体系结构。如图1所示，最左端是 Video 服务器群，上面存放所有的视频数据，可以接受点播请求并向请求者传送信息。中间一截是广域网，也是数据传输中的主干网络，这里我们认为是 Internet。右端是高速局域网，其中用户端(client)是直接连到代理缓存服务器系统(Proxy Cache System, PCS)上的，再由 PCS 连接到 Internet 中。PCS 作为代理服务器来接受所有客户的请求，并且向位于 Internet 另一端的远程服务器发送请求，然后再将收到的信息通过内部高速网络传送给客户；PCS 同时作为缓存使用，因为众多客户可能请求同一视频数据，所以作为缓存信息滞留在 PCS 中的数据可以立即传送给客户，减小了客户等待时间，也缓解了主干 Internet 以及远程服务器的负载压力。

在 PCS 中，可以设置一个集中管理者 coordinator，它负责接受客户的请求，查询所需数据是否缓存于本 PCS 中，若是，则确定文件位置，并且命令相应缓存向客户传送数据；若

*)本文的工作得到了国家863项目"应用服务器的运行、管理及调度技术"(项目编号：2001AA113050)及江苏省高技术产业化推进项目"基于多服务器群的大规模多媒体信息服务平台的研究和开发"的资助。

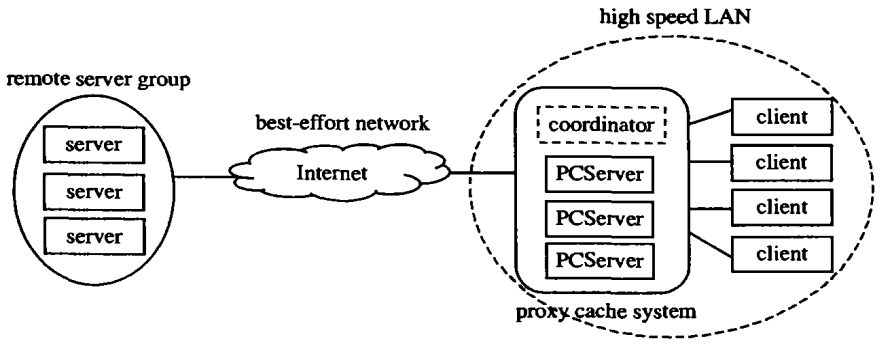


图1 基于 Internet 的视频传送系统抽象模型

否，则向远程服务器请求数据，对于收到的数据，根据一定的策略来决定是否进行缓存以及缓存位置。其它的服务器都是用来缓存数据的代理缓存服务器(Proxy Cache Server，PCServer)。整个 PCS 系统由 coordinator 进行统一管理，可能存在的问题有两个，一是集中可能导致瓶颈；二是集中可能导致瘫痪。对于前者，因为 coordinator 并不进行具体的视频信息传送工作，只是处理客户请求，并且维护缓存数据的位置信息，所以基本不会成为 PCS 系统的瓶颈。对于后者，可以用备份的方法解决，即为了防止 coordinator 发生故障时导致整个系统不可用，可以引入一个后备的 coordinator，只在故障时启动运行，这一情况在本文中被忽略。也可以不要 coordinator 而采用完全分布的方法进行 PCServer 间的交互，在下文中将会有具体的讨论。

3 代理缓存服务器上 video 数据的放置策略

由于 video 信息数据量大，因此如果把一个 video 文件作为一个整体来进行缓存以及置换，其效率将是非常低的，因为 PCServer 的容量有限，所以总共可以存放的缓存文件的个数很少，而且每次置换量也很大，导致命中率低，额外开销大，所以大部分的缓存策略都是对 video 文件进行部分缓存(partial cache)。也就是将文件划分为几个部分，以部分为单位来进行缓存。在文[6]中还特别就部分的大小对系统性能的影响做了研究。客户在使用视频服务时通常的行为是这样的，点播某一视频信息，稍作观看后若发现不想继续观看则取消，否则继续观看直到结束。有统计显示[7]，大约45%的请求都属于前者，其余55%为后种情况。所以在代理缓存服务器上放置多个 Video 的开头部分是一种针对这种需求模式的很好的解决方案，这种方法被广泛认同，称为前缀缓存(Prefix caching)。这里要详细讨论的是另两种特殊的方法。

3.1 video staging 方法

在文[5]中，Y. W. Wang 等人提出了一种横向的分割方法，即不再是对 video 数据流进行通常的以若干帧为一部分的纵向分割，而是横向将所有帧切割成两部分。

广域主干网上数据是按照等比特速率来传送的(CBR)，video 文件在客户端的播放是以等帧速率进行的，而每一帧所含的比特量有所相同，所以为保证客户端的正常放映，只有将主干网中预留的带宽设定成整个 video 文件中最大的放映比特率(peak rate)。很显然这种方法会浪费主干网的带宽资源，而且当主干网出现资源不足等情况时会直接影响到客户端的播放效果。

Video staging 方法就是将那些容量大的帧的一部分放置到代理缓存服务器上，客户请求远程服务器中的 video 文件时，只需传送容量小的帧和容量大的帧的一部分，再由代理缓存补足另一部分，传送给客户，这样一来，在主干网上传的每一帧的容量就不会超过一个固定的切割速率(cut-off rate)了。主干网的带宽资源得到了节省。这种 video staging 方法还可以和平顺处理(smoothing)[8]结合起来，可以是平顺前分割(video staging before smoothing)，也可以是平顺后分割(video staging after smoothing)。

如图2[8]所示，设对于 video 文件 i，F 为每帧的播放时间，共有 N_i 帧，并且第 j 帧的大小为 S_i^j 比特。这样，传送的最大比特率就是 $P_i=(\mathrm{Max}S_i^j)/F$，$(1<j<N_i)$。若设切割速率为 C_i $(0<C_i<P_i*F)$，则基础部分(lower part)将存放在远程服务器中，即每一帧中的 $S_i^{j,l}$ 大小，$S_i^{j,l}=S_i^j-(S_i^j-C_i)^+$，其中 $x^+=\max\{x,0\}$。而超出部分(upper part)将被复制到代理缓存中，为每一帧的 $S_i^{j,u}=(S_i^j-C_i)^+$ 大小。实际上远程服务器中存放的是整个 video 文件，但是传送时只传基础部分。这样，P_i 就减小成为 C_i/F 了。

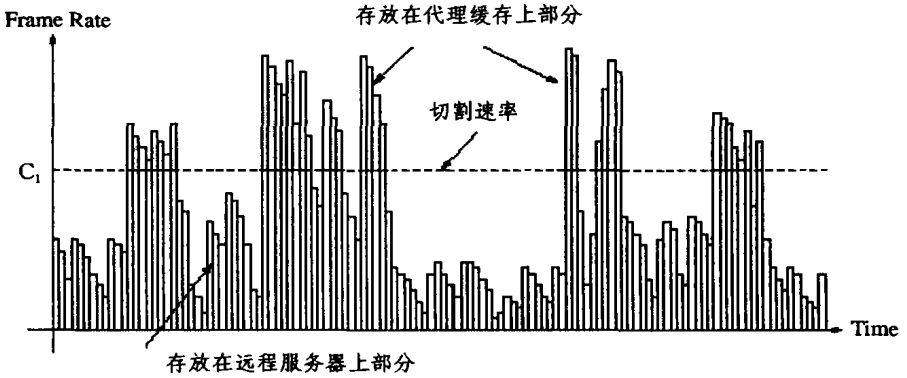


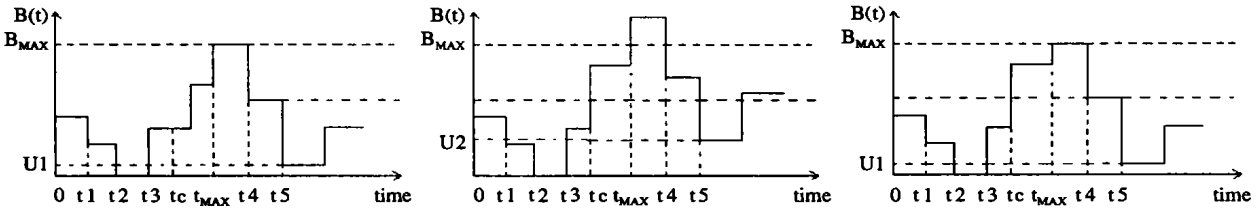
图2 使用切割速率 C_i 对单个 video 文件进行分割

这种 video staging 的方法可以使主干网的资源得以有效的利用。但是，由于代理缓存上没有完整的开始部分的数据，因此并不能减少客户端的开始延时(start delay)，而且代理缓存的可扩展性也不好，如当代理缓存的容量增加时，必须调整 C_i 的大小，也就是对每一帧都必须重新传送一些新增部分到代理缓存服务器。

3.2 选择缓存方法(selective caching)

文[9]提出了一种基于客户端缓冲器的选择缓存算法，并通过模拟试验证明了其有效性。如上所述，前缀缓存方法被认为是有效的，那么在已经缓存了一定的前缀部分之后，若缓存空间还有剩余，该再缓存哪些部分呢?可行的方法只有两种，一是接着顺序缓存前缀之后的那些部分，另一种就是不按顺序而间隔挑选后面的部分进行缓存。

客户端总是将一部分内存设为缓冲器，数据经由缓冲器再行播放，由于数据到达的速率和播放的速率不一致，缓冲器中的数据量是随时变化的，当充满时，将停止接收数据，称为上溢(overflow)，对于缓冲器空的情况则为下溢(underflow)，其中后者对于播放性能的影响更大。另一方面，缓冲器的使用可在一定程度上提高系统的健壮性，即当网络出现较大延迟时，缓冲器中的内容仍然可以保证一段时间的正常放映，从而屏蔽掉网络的暂时失效。选择缓存的方法就是基于客户端缓冲器的，它选择合适的帧来缓存，使客户端缓冲器避免出现下溢的情况，从而提高了整个系统对于广域网络容易出现的阻塞延迟等现象的免疫能力。



(a) cache 之前　　(b) cache 一帧，没有缓冲器大小限制　　(c) cache 一帧，有大小限制

图3　是否有 cache 以及是否有缓冲器大小限制对缓冲器容量的影响

图3引自文[9]。首先，这幅图说明了在代理缓存中事先缓存一帧数据对整个系统的影响。这里，可以用客户端缓冲器中的数据量来代表系统的健壮性，设为 U，则 $U=MIN\{B(t)\}$，$B(t)$ 为 t 时刻缓冲器中的数据量。显然(a)中 $U=0$；另一个"薄弱"点是 t5时刻，此时 $U=U1$。在 t_{MAX}时刻缓冲器满。现设代理缓存中缓存了在 t_c 时刻播放的那一帧，则因为从代理缓存服务器向客户端传送数据的时间可以忽略不计，在客户端缓冲器看来，容量可以瞬时增加，如(b)中 t_c 时刻所示，并且在无缓冲器容量限制或缓冲器容量足够大的情况下，t_c 之后的每一时刻缓冲器中容量都较(a)有所提升，"薄弱"点 t5时刻的 U 值也上升为 U2，也即提高了系统的健壮性。但是如果客户端缓冲器不是足够大，则当到达缓冲器满的时刻，如(c)中的 t_{MAX}，之后将停止接收远程服务器传来的数据，所以 t4之后其中的容量变化将和(a)一样，不再能够提高系统健壮性。这就是选择算法中对于帧的选取的依据。这也说明了缓存开头连续的帧对整个系统的健壮性并没有提高，因为它对于缓冲器充满之后的容量变化没有任何影响，后面仍然可能出现很小的 U 值。

选择缓存算法就是缓存 U 值达到极小值之前的某帧，使得随后的极小 U 值可以增大一些。如上所述，它对缓存器充满之后的 U 值没有影响，所以还必须选择充满之后，达到另一个极小值之前的某帧进行缓存。可以用迭代的算法来实现选择缓存。具体算法见文[9,10]。

选择缓存方法巧妙地利用客户端缓冲器的缓冲数据量的变化来进行选择，而不是只缓存视频文件的起始部分，使特定情况下的系统的健壮性达到了最高。但是这种缓存方法也有一些局限性。首先，它没有考虑到置换策略。视频文件的个数总是很多的，缓存中总是应该保存需求最频繁的所谓"热门"文件，但是"热门"是会更改的，这就需要以一定的策略进行置换，而且缓存的每个文件的大小也应该根据其"热门"程度来决定，而选择缓存方法对这一问题没有考虑。另外，该方法是基于客户端缓冲器中数据量的变化情况来运行的，也就是说知道了数据到达速率，播放速率，影片每帧的比特量之后才能够运行的，它是假定网络状况稳定，为了防止偶尔的不稳定而提高系统的健壮性。而实际应用中的网络状况是很不稳定的，对于数据到达率的估计将可能有很大误差，导致该方法的使用效果受到很大影响。且对于视频会议，新闻实况传输等事先不知道数据量的应用也不适用。

4 代理缓存服务器数据的置换策略

有缓存就有置换。缓存中的数据只是源服务器中的最可能被读取的一部分，随着客户需求的动态变化，缓存中的数据也必须相应变化以使缓存达到最高效率。因为有别于传统的"时间空间局部性"的缓存理论原则，Web cache 中的缓存置换策略也有所变更。主要的缓存置换算法基本可分为三大类[11,12]：

1. 传统的置换算法。如最近最少用方法(LRU)，最近最不常用(LFU)，或者这些方法的适当扩展，如 Pitknow/Pecker[13]方法，即按 LRU 方法来选择被置换对象，但若有多个对象在当天都被读取过，则选择其中最大的对象。

2. 基于关键字的置换算法。即根据某一个或多个键值来选取被置换的对象。如 Size[13]方法，即直接选择最大的对象；LRU-MIN[14]方法，即按大小划分对象，在同一级别大小中实行 LRU，如对于所有大于 S 的对象按 LRU 选择被置换者，若无大于 S 的对象则对所有大于 S/2的再用 LRU，依此类推。

3. 基于花费代价的置换算法。这种算法使用某种代价函数来决定被置换对象，函数的参数可以是上一次读取距今时间、传送代价、对象有效期限等内容。如 GreedyDual-Size 方法，就是使用(代价/大小)这样的函数。

但是，这些 Web cache 中使用的置换算法并不适用于 video 文件。如前文所述，由于 video 文件非常大，一般只能分割后缓存其中一部分，也就是说不能将其整体作为缓存和置换的对象，但是又由于一个 video 文件的若干个部分是有关系的，并且客户必定是顺序读取的，因此这些部分的缓存与置

换必须有其特殊的方法，而不能按照传统的方法当作独立的对象来操作。所以一种通常的方法就是将一个 video 文件划分成大小相等的若干块，开始缓存开头一些块，若是使用频率高则依次逐渐增加后面的块，否则也是依次减少所缓存的块。根据这一思想，一种基于段(segment-based)的缓存与置换方法[15]被提出。

在这种方法中将每个 video 文件划分成大小相等的块(block)，然后将块合并成段(segment)，合并方法为：第0段为块1；第1段为块2；第2段为块3、4；… 第 i 段为块$2^{i-1}+1$、$2^{i-1}+2$、…2^i，共2^i个块，最后一段可以是少于该公式的任意数目。以后的分布与置换的单位为段。这种不均等分割的意义在于让越往前的部分越慢被置换出去，原因即前文所述的每个 video 文件的开头部分的高使用率以及其对客户所见的延迟的影响。缓存的时候预先设置一个限值 K_{min}，单位为段，再对每个 video 文件的每一段设置一个缓存值(caching value)。缓存的时候对于每个 video 文件，小于 K_{min} 的所有段均可以予以缓存，并且是作为一个整体的；对于大于 K_{min} 的所有段，则根据其 caching value 从大到小进行缓存直到代理缓存服务器被充满。这样其实缓存空间被分成了两部分，一部分专门存放每个 video 的前 K_{min} 部分，而另一部分则放某些 video 的后些部分。且置换时此两部分是独立进行的，也即某个 video 的前 K_{min} 部分只能被另一 video 的前 K_{min} 部分所置换。K_{min} 的选取原则是让 K_{min} 之后的段能有足够时间从远程服务器传送过来以保证客户端的连续读取。Caching value 的计算有两个因素，一是该段的位置，一是该段所在 video 的读取频度，所以可以用函数 1/((T－T')＊i)来表示，其中 T 为当前时间，T' 为上一次读取的时间，i 为段号。首次读取 video 的 caching value 是0，所以只缓存前 K_{min} 部分。可以用时间戳的方法来记录读取信息。置换时当前正被读取的 video 的段将不参加 caching value 的比较，即活动段(active segment)不会被置换掉。

这一置换算法在试验中被证明可以有效地提高字节命中率(byte-hit ratio)，减少网络传输量，尤其适用于缓存容量有限，video 文件的读取频率随时间变化，video 文件较长且客户经常只观看 video 文件的开头一部分即中止观看的情况。但是在这种方法中并没有考虑到系统所在网络的拓扑状况，没有考虑到如何节省网络带宽，对于具体的适当的 K_{min} 值也没有给出合理的计算方法。在具体的算法中，前 K_{min} 部分与 K_{min} 之后部分是分开置换的，这也能导致该算法的低效，因为被置换出去的可能不是所有 cache 的对象中 caching value 最小的值。

5 代理缓存服务器群的交互与管理

以上所讨论的 video 数据的存放与置换策略均是针对单个代理缓存服务器的，实际应用中常常使用多台代理缓存服务器形成一个代理缓存系统(PCS)，如图1所示，来提高代理缓存的效率，并且采用集中或分布的管理方法来进行任务分派、负载平衡等工作。本文的第二部分已经就这一结构进行了一定的讨论，这里将进一步研究代理缓存服务器群的交互与管理方法。

5.1 集中式的管理

MiddleMan[16]中的代理缓存只是逻辑上的缓存服务器，物理上是可以和客户机在同一台机器上的。代理缓存服务器分为两种，一种是存储代理服务器(storage proxy server)，一种是本地代理服务器(local proxy server)，前者负责缓存与发送数据，后者则负责处理请求，并不真正缓存数据。系统中还有一个协调者(coordinator)负责记录系统中所有缓存的信息。

当客户请求到达某个本地代理服务器时，它同时与 coordinator 以及远程 Web server 联系。若 coordinator 返回信息为不命中，则等待 Web server 传来的数据，同时由 coordinator 决定对传来的数据是否缓存以及缓存在哪个存储代理服务器上，这样传来的数据同时发送给客户端以及存储代理服务器；若 coordinator 返回信息为部分命中，则由 coordinator 定位文件所在存储代理服务器，并开始传送给客户，同时将没有 cache 的部分由 Web server 传来，按策略执行缓存；若 coordinator 返回信息为命中，则取消与 Web server 的连接，由 coordinator 的数据位置信息读取数据。

MidleMan 中也采用将 video 文件分割的方法，缓存时按顺序由前向后进行，置换时也是置换读取频率低的 video 文件的后面部分。该项目并没有提出某种新的置换算法，它是侧重于整个系统的体系结构的搭建的，在试验中使用了多种置换策略来微调系统的效率，如 LRU，LFU，FIFIO，LRU-k，以及 HistLRUpick。试验结果表明，HistLRUpick 置换算法的效果最好，有较高的命中率以及负载平衡性能。同时，在总共的缓存大小相同的情况下，多个小容量代理缓存服务器的性能比少个大容量代理缓存服务器的性能好(当然是在一定限度之内)。该项目的进一步的研究方向为在现有系统中加入容错、VCR 功能支持、安全验证以及相邻 MiddleMan 系统的协作。

5.2 分布方式的管理与协调

集中的管理方式不可避免地可能会导致瓶颈、单点失效，并且影响系统的可扩展性。让 PCS 中的各个 PCServer 之间进行完全的分布式的交互与协调就可以避免这些缺点了。一般多用的分布式方法是各种各样的散列方法(Hashing)。有缓存阵列路由协议(cache array routing protocol)，一致散列法(consistent hashing)等[17]。散列的主要思想是将客户可能请求的 URL 空间划分成若干部分，每一 PCServer 对应服务其中的一个部分，这样通过简单的散列计算，就可以自动直接将客户请求定位到某台代理缓存服务器来完成所需服务，而不需通过所谓的集中协调者了。散列算法可以在客户的浏览器中进行也可以由域名服务器(DNS)来完成。而各个 PCServer 相对独立地进行缓存与置换。

这种方法简单地实现了分布式的处理，但是仔细分析便可发现它的缺点。那就是它极有可能引起负载不平衡。试想，video 文件的 popular value 是不一样的，这样负责热门影片的 PCServer 很容易就满负载了，而负责冷门影片的则负载很轻。在文[18,19]中，Kun-Lung Wu 等人提出了一种用来解决 hashing 方法所带来的负载不均的自适应受控复制方法(adaptable controlled replication)。

这种自适应受控复制方法的思想就是将热门 video 文件根据需要复制到其它的 PCServer 上，从而减轻其宿主缓存服务器(partition owner)的负载。这种复制完全是由客户需求频率来决定的，所以复制的一定是“热门”的 video 文件，当其不再热门，将会自动被替换掉。这种机制是由如图4[19]所示的 LRU 栈来实现的。这个栈分两部分，上面是本地 LRU 栈(local LRU)，下部分是常规 LRU 栈(regular LRU)。对于存放在此 PCServer 上的 video 文件(或文件片断)可以分为两种，

一种是按 Hashing 算法应该缓存于此的文件称为分派部分(assigned partition),另一种就是它复制其它 PCServer 上缓存的文件,称为非分派部分(non-assigned partition)。前者可以缓存于任意 LRU 栈,后者只能缓存于本地 LRU。如图,当新文件到达时由栈顶压入,本地 LRU 栈底的文件被取出,若是非分派部分则丢弃,若是分派部分则压入常规 LRU 栈的顶部,该栈底部的那个分派部分文件被丢弃。显而易见,对于非分派部分文件,只有当其读取频率很高时才能留在缓存中,否则很快被置换。

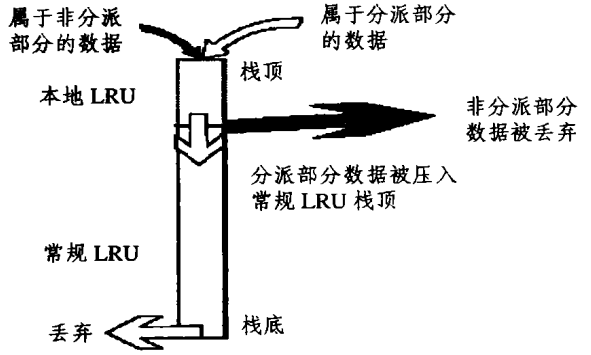


图4 两段 LRU 栈

整个 video 文件 URL 空间仍然按照 hashing 方法被划分,每个 PCServer 负责其中一块,但是客户请求是按照距离远近发给最近的 PCServe 的。当客户请求到达某 PCServer(称为直接服务器)时,它查找本地 LRU 栈看是否有缓存,若有,则发给客户,若无,则看是分派部分还是非分派部分,若是前者,则查询常规 LRU 栈看是否有缓存,若有则发送给客户,若无则向远程服务器请求,将到达文件缓存于本地,并发给客户;若是后者,则计算缓存此文件的 PCServer(称为宿主服务器)地址,向其发送请求,该宿主服务器若有则传送,否则由它向远程服务器请求,将到达文件作为分派部分缓存于此并发给直接服务器,直接服务器将收到的文件作为非分派部分缓存,并向客户传送。

模拟试验表明,这种方法对于分布式控制中的负载平衡有很大帮助。受控的复制完全按照实际的读取频率来进行。但是这种方法中没有详细考虑 video 文件的置换算法以及置换粒度等问题,只是用了 LRU 方法来作简单的处理。

6 对于分层 video 格式的缓存策略

以上的这些 video 数据存放、置换与传送方法都不能调节所传送 video 的质量,也就是说对于各种各样的客户接入网络,无论是百兆网还是千兆网,它提供的是不可变的统一质量的服务,这就缺乏灵活性了,系统的性能也将收到影响。Mocha 系统[20]就把分层 video 结合到了代理缓存的设计中,即根据 video 的存取频度以及代理缓存与客户端之间的网络状况,来调整缓存与传送的 video 数据的质量,使整个系统具有自适应性。

为了适应不同的客户接入网络,传送的 video 数据必须根据网络的带宽限制来调整质量,有两个方法,一是对同一 video 文件用不同的方法生成不同质量的版本;二是用分层编码的方法,在传输时实时根据质量需求抽取其中若干层次传送。显然前者会花费额外的存储空间,后种方法则可以有效地解决自适应问题。

视频数据分层编码的概念就是在编码时划分基本层与加强层,只有基本层时 video 分辨率不高,但数据量小,加强层必须和基本层结合才能传送,会加大 video 的数据量同时提高分辨率,增强播放效果。关于分层编码可以参见文[21]。

由于引入了 video 数据分层编码的概念,Mocha 的缓存策略就与众不同了。它不光将 video 文件纵向分块,还将其横向分层。缓存的时候是以某一层的块(chunk)为单位进行的。当客户请求到达时,若缓存不命中,则由代理缓存向远程服务器发送请求,并将到达的 video 数据传送给客户,同时根据一定的策略予以缓存,这种情况下获得的一般是基本层的 video 数据,所以不能再调整发送给客户的数据质量;若缓存命中,则可以直接由代理缓存向客户发送数据,但若检测到客户接入网络性能高于缓存的数据的质量,则同时向远程服务器请求传送此 video 的较高层次数据以及尚没有缓存的部分,这一过程中还要用到一种细粒度的预取策略(Fine-grained prefetching)[20]。由于分层 video 文件的特殊性,其在代理缓存服务器上的存储管理方法也颇为特殊。在 Mocha 中,是将 Squid[2]的数据结构加以更改来实现存储管理的。如图5[20]所示,若将某一 video 的每一层编码数据看作一个独立的对象,给予一个数组项来记录其信息,就可以重用 Squid 的数据结构了。每一 video 数据层用一个链表来表示,每个链表结点为该层的一个数据块,一个数据块又包含若干数据包,为了方便定位,数据包中有指向下个包的指针。但是缓存与置换的单位仍然是块。

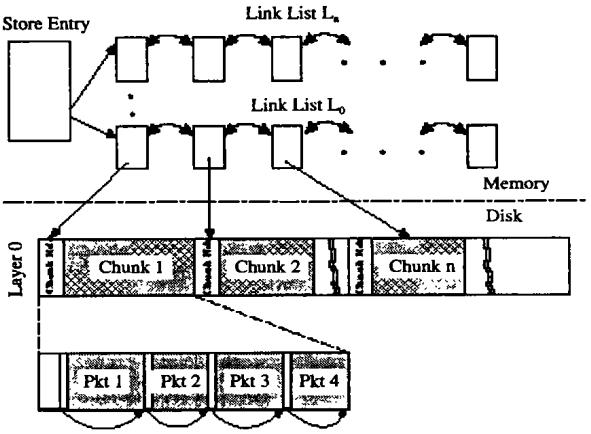


图5 Mocha 系统数据存储结构

由于 video 数据分层分块缓存置换,因此置换算法中所依据的所谓置换参数也是针对每一层次的每一数据块来设置计算的,称为细粒度的置换机制(fine-grained replacement mechanism)[20]。当缓存容量大于某一阈值时启动置换程序。Video s 的 i 层的 popular value 用公式 $P_i=\sum_{x\in[t-\Delta,t]}^{t} whit(x,i)$,其中 $whit=PlayTime(x,i)/StreamLength(s)$。用一段时间内某层的累积的加权命中次数来代表其热门度,就可以使不同 video 的不同层次有不同的 popular value,并且同一 video 的低层的 popular value 一定大于较高的增强层。置换的时候,从热门度列表(popularity list)中选取最小的 video 层,再从其末尾块开始删除,直到低于阈值为止。

这种方案的好处是显而易见的,它不仅有效地实现了代理缓存的功能,而且还考虑到了客户接入网络的异构性,并充分利用这一点来有效节省网络带宽、提高传送质量从而提高系统的性能。不过它的缺点也同样显而易见,对于不是分层编码的 video 文件,它只能按照通常方法进行处理,不再具有自

适应性，额外的存储管理开销可能还会使其性能下降。而且另外一点值得考虑的就是既然该种方案引入了传送给客户的video质量的调节，那么它的性能评估参数就不再应该只是传统的字节命中率(byte-hit ratio)了，应该把客户端的播放质量也纳入到考虑因素中统一进行评估。

结论 本文研究了基于Internet的视频传送系统中的代理缓存服务器的设计中的若干关键问题。既有对部分缓存策略(partial cache)的详细讨论，又有对针对大体积video的特殊置换算法的讨论，还就多个代理服务器群的管理方案进行了一定的研究。最后还针对分层编码的video代理缓存方案做了分析。从中可以看到，这些代理缓存的设计分别侧重于某个方面，并没能够兼顾其它的考虑，如选择缓存方法就没有考虑到缓存数据的置换问题，基于段的缓存置换算法又是只针对单个代理缓存服务器的，而由多个代理缓存服务器构成代理缓存系统的MiddleMan项目又只侧重于系统体系结构的设计，并没有包含针对video特别设计的缓存置换算法，最后的针对分层编码的代理缓存系统Mocha具有其独到性，同时也有不可避免的局限性。我们认为，基于Internet的视频传送系统中的代理缓存应该构成一个具有可扩展性、自适应性的系统，并且针对video数据的编码特点、传输特点来设计缓存置换策略，用带宽需求量、客户端播放性能等来作为最终性能测试指标。

参考文献

1 Chankhunthod A, et al. A Hierarchical Internet Object Cache. In: Proc. of the 1996 USENIX Technical Conf. Jan. 1996

2 http://squid.nlanr.net/squid

3 http://netlab1.bu.edu/~staro/546projects/mocha/

4 http://www.cs.cornell.edu/zeno/Projects/middleman/Default.html

5 Wang Y W, Du D H C. A Network-Conscious Approach to End-to-End Video Delivery over Wide Area Networks Using Proxy Servers. In: Proc. of INFOCOM' 1998

6 Balafoutis E, Panagakis A, Laoutaris N, Stavrakakis I. The Impact of Replacement Granularity on Video Caching, IFIP Networking, Pisa, Italy, 2002

7 Acharya S, Smith B. An Experiment To Characterize Videos On The World Wide Web. In: Proc. of ACM/SPIE Multimedia Computing and Networking 1998(MMCN'98), San Jose, Jan. 1998

8 Salehi J, Zhang Z L, Kurose J, Towsley D. Supporting Stored Video: Reducing Rate Variability and end-to-end Resource Requirements Through Optimal Smoothing. In: ACM Intl. Conf. on Measurement and Modeling of Computer Systems (ACM SIGMETRICS) (Philadelphia, PA, May 1996)

9 Miao Z, Ortega A. Proxy Caching for Efficient Video Services Over the Internet. In: 9th Intl. Packet Video Workshop(PVW'99), New York, April 1999

10 Miao Z, Ortega A. Scalable Proxy Caching of Video under Storage Constraints. IEEE Journal on Selected Areas in Communications, (Internet Proxy Services) 2002

11 Wang J. A Survey of Web Caching Schemes for the Internet. ACM Computer Communication Reviews, 1999, 29(5)

12 Aggarwal C, Wolf J, Yu P. Caching on the World Wide Web. IEEE transactions on knowledge and data engineering, 1999, 11(1)

13 William S, Abrams M, Standridge C R, Abdulla G, Fox E A. Removal Policies in Network Caches for World Wide Web Documents. In: Proc. of Sigcomm'96

14 Abrams M, et al. Caching proxies: Limitations and Potentials. In: Proc. of 4th Intl. WWW Conf. Boston, MA, Dec. 1995

15 Wu K, Yu P, Wolf J. Segment-based Proxy Caching of Multimedia Streams. In: Proc. of 10th Intl. Conf. on World Wide Web, Hong Kong, 2001

16 Acharya S, Smith B. MiddleMan: A Video Caching Proxy Server. In: Proc. of 10th Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Chapel Hill, NC, June 2000

17 Wu K L, Yu P S. Latency-Sensitive Hashing for Collaborative Web Caching, Computer Networks: The International Journal of Computer and Telecommunications Networking, June 2000

18 Wu K L, Yu P S. Replication for Load Balancing and Hot-Spot Relief on Proxy Web Caches with Hash Routing, to be appear in Distributed and Parallel Databases, Jan. 2003

19 Wu K L, Yu P S. Controlled Replication for Hash Routing-based Web Caching, to be appear in International Journal of Computer Science and Engineering, Sept. 2002

20 Rejaie R, Kangasharju J. Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming. In: Proc. of NOSSDAV

21 Cho J, Shin H. Design Issues for Multimedia Information Servers in Mobile Computing Environment. In: Proc. of the 4th Intl. Workshop on Mobile Multimedia Communications, Sep. 1997

(上接第45页)

7 洪子泉，杨静宇. 基于奇异值特征和统计模型的人像识别算法. 计算机研究与发展，1994，31(3)：60～65

8 Lawrence S, Giles C L, Tsoi A C, Back A D. Face recognition: a convolutional neural-network approach. IEEE Trans. Neural Network, 1997, 8(1): 98～113

9 Yoon K S, et al. Hybrid approaches to frontal view face recognition using the hidden Markov model and neural network. Pattern Recognition, 1998, 31(3): 283～293

10 Schofield A J, et al. A system for counting people in video images using neural networks to identify the background scene. Pattern Recognition, 1996, 29(8): 1425～1428

11 Intrator N, et al. Face recognition using a hybrid supervised/unsupervised neural network. Pattern Recognition Letters, 1996, 17(1): 67～76

12 Lin C T. Neural Fuzzy System. Prentice-Hall Press, U.S.A, 1997. 328～330

13 Fukunaga K. Statistical pattern recognition. New York: Academic, 1989. 4

14 於东军，王士同，吴小俊. 层次径向基神经网络的全局逼近理论. 计算机研究与发展，1999，36(11)：1329～1334