郭琳　李欣　陈志军　苏群　程懋泰　赵宇明　(北京中医药大学　总参61所)

网络缓存技术

摘要：本文详细介绍了网络缓存技术的几种实现方法，及其工作原理，性能特点和配置方式，并说明了使用网络缓存的优越性。

关键词：缓存　配置　缩短　提高

1　引言

近年来，互连网用户数量及其对宽带需求的增加远远超过网络基础设施建设的速度，这种供需双方的不平衡发展带来诸多的问题，出现了骨干链路拥塞、服务器过载以及响应时间过长的现象。解决问题的方法很多，但对于希望尽快摆脱带宽瓶颈困扰的用户和网络服务提供商来说，网络缓存是一项行之有效的技术。

2　网络缓存类型

网络缓存类型分为四种：代理缓存 (Proxy Caching)，适用性网络缓存 (Adaptive Web Caching)，推进式缓存 (Push Caching)，活性缓存 (Active Caching)。其中代理缓存又包括了：孤立型 (Standalone) 缓存，透明型 (Routertransparent) 缓存和反向代理缓存 (Reverse Proxy Caching)。

2.1 代理缓存

代理缓存其服务器能够截取客户的HTTP请求，如果在缓存中保存有该请求的对象就直接把它发给客户，反之则把该请求发送出去，从源端服务器获取所需数据并传送给客户，至于是否将该对象添加到缓存中取决于网络缓存的替代策略。代理缓存通常配置在网络的边缘 (如网关和防火墙处) 以便为全体内部用户提供服务。通常使用代理缓存可以达到节省带宽，缩短响应时间和提高静态网络数据和对象可用性的目的。代理缓存分为以下三种类型：

(1) **孤立型代理缓存**：孤立型代理缓存属于第一代网络缓存技术，这种设计的缺陷在于缓存本身有可能成为造成网络失效的环节。当缓存不可用的时候，用户将同时失去与网络的连接。孤立型代理缓存的可扩充性也非常差，几乎没有办法根据需要动态扩展缓存的容量，并且缓存的每次变动都要求所有的用户对WEB浏览器的配置进行手动的改动。尽管IEIF最近在浏览器自动配置方面取得了很大的进展，但孤立型缓存仍然不是一种令人满意的方案。

(2) **反向代理缓存**：反向代理缓存配置在服务器的前面，通过截取和分析所有到来的请求，缓存像个筛子一样，对包含在自己存储区中的对象的请求直接进行响应，其余的部分才发给服务器处理。对于服务器来说，这显然是一种提高访问请求处理能力和服务质量的有效方法。而事实上，反向代理缓存已经被越来越多的ISP所接受，并日益成为一种必需的配置。

(3) **透明缓存**：透明缓存属于第二代网络缓存技术，它重点解决第一代产品的两个缺点，减少了配置的复杂性，并采用分层和集群技术提高可扩展性。在使用透明缓存的情况下，由路由器或交换机检查每一个数据包，将所有HTTP流量重定向到网络缓存，用户无须对WEB浏览器进行配置，但路由器或交换机必须被设置为具备重定向功能，也就是说用户浏览器“看”不到缓存，但路由器或交换机“看”得到。透明代理缓存有两种形式：基于路由器 (Router-Based) 和基于交换机 (Switch-Based)。基于路由器的缓存使用急于策略 (Policy-Based) 的路由法则将用户的请求重定向到特定的缓存。而在基于交换机的缓存中，不仅能够避免因单一的缓存承载所有请求而造成的性能下降，而且提高了缓存的可扩展性。

2.2 适应性网络缓存

适应性网络缓存着重解决的问题是“热点”现象，即由于某些短期的网络内容突然变得十分流行而带来的巨大需求量。适应性缓存由众多分散的缓存组成，这些缓存能够根据需要动态地加入或离开缓存组。缓存组具有自适应和自组织特性，能够对需求的渐变或突变做出反应。自适应缓存使用缓存管理协议 (CGMP) 和内容路由协议 (CRP)，CGMP定义缓存组如何组织以及单个缓存如何参加和离开缓存组。通常缓存组使用投票和反馈技术决定单一缓存的去留，CRP决定在缓存组内部如何放置缓存的内容。这项技术有赖于缓存组成员之间的多播通信并利用URL表智能化地决定如何分配用户的请求。

2.3 推进式缓存

推进式缓存的核心思想是保持缓存的数据靠近有需求的用户。当源数据所在的服务器识别出请求产生的地址后，数据就被动态地镜像。例如，从武汉到北京某站点的流量开始剧烈增加，则该站点可以通过启动配置在武汉的缓存进行响应。因此，推进式缓存的特点之一就是跨越行政边界启动异地缓存的能力。与自适应缓存不同，推进式缓存并不为所有内容提供商的各类内容的分发提供通用的解决方案，而是只服务于采用了该推进式缓存

站点。与传统的缓存结构相比，好的推进算法能使效率提高1.27到2.43倍，但推进式缓存也同样面着两难的境地：传输和存储本地内容的副本必然提高花费，但只有等到这些内容真的被访问之后才会带来总体性能的提高。

2.4 活性缓存

活性缓存的产生源于网上个人内容在不断增加，而传统缓存技术又很难处理这些动态的文件，调查显示，大约30%的HTTP请求包含有Cookies这些请求为个人的元素。而且随着网上一对一方式电子商务的不断增加，这种个人化的趋势还会更加明显。活性缓存通过使用位于其中的JAVA小程序(Applet)来使对象客户化。当对个人内容的请求首次发出的时候，源服务器提供所请求的对象和所有相关的Applet。当对同一内容的后续请求到来的时候，这些Applet在本地缓存中执行，这样既保留了缓存的好处又保证了内容的用户化。

3 网络缓存的配置

网络缓存的配置主要有三种：靠近内容消费者方式(面向用户方式)、靠近内容提供者方式(面向提供者方式)和根据用户接入方式和网络拓扑结构配置在网络中的关键点。

面向用户方式的好处在于能够使一个或多个缓存为众多用户提供服务。如果这些用户对相同种类的内容感兴趣，则这种配置方式所带来的好处将十分明显。将缓存配置在内容提供商一侧能够提高对特定内容的访问效率，对于影音之类对延迟敏感的内容而言，面向提供者配置缓存是一种提高内容的可用性和可扩展性的有效方式。但是很明显，只有缓存的拥有者才能从中受益。在网络拥塞点动态配置缓存是自适应缓存所采用的策略。尽管看起来这是一种最具灵活性的方式，但它还不成熟，尚处在研究当中。此外，动态配置缓存还涉及到一个复杂的问题，即如何对这些缓存进行管理。

4 网络缓存一致性方法

网络缓存一致性方法主要用于保证缓存中不保留过时和无用的数据，即已经在源端服务器上发生了改变的和陈旧的对象。缓存一致性保持有四种类型：Client Polling、Invalidation Callback、Time to Live和If Modified Since。

在Client Polling方式中，每一个缓存对象都被打上时间戳并不断地与源服务器上对象的时间戳进行比较。过期的对象将被删除，而新的对象则视需要重新攫取。Invalidation Callback方式依靠源端服务器来识别过时的缓存对象。服务器必须掌握所有缓存了其内容的代理的位置并在这些对象发生改变的时候与相应的代理进行联系。这种方法的好处是节省了带宽，因为用户无须频繁地对服务器进行检测。但另一方面，由于服务器必须跟踪所有相关的代理，因而不可避免地带来扩展性和安全方面的问题。与数据包在网络中传输时的生命期相类似，也可以赋予缓存的生命期(Time to Live)一旦过期，则旧的对象被抛弃而新的对象被攫取。这种方式能够有效地限制重复攫取的次数，但无法保证重复攫取不发生在静态对象的身上。If Modified Since是Time to Live方式的一个变种，它由需求来驱动，即只有缓存对象已经到期并被用户请求的时候才去检查其有效性。

5 结论

用户端使用网络缓存，能够有效地减少与远端服务器连接的需求以及由此引发的网络通信量，从而降低骨干网上数据的传输量。由服务提供商所管理的缓存则不仅能够提高与之相连接的WEB服务器的可用性及扩展能力，而且可以起到平衡负载的功效。网络缓存能够在一定程度上将用户与网络延迟和拥塞屏蔽起来，对于影音之类的多媒体数据而言，效果更为显著。

参考文献

1 M. Ahamad, M. Raynal, and G. Thia-Kime. "An Adaptive Architecture for Causally Consistent Distributed Services." Technical Report PI-1039, IRISA, Rennes, France, July 1996.

2 M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm. "Enhancing the Web's Infrastructure: From Caching to Replication." IEEE Internet Comput., 1(2):18-27, Mar. 1997.

3 S. Caughey, D. Ingham, and M. Little. "Flexible Open Caching for the Web." In Proc. Sixth Int'l WWW Conf., Santa Clara, CA, Apr. 1997.

4 D. B. Terry, A. J. Demers, K. Petersen, M. J. Spreitzer, M. M. Theimer, and B. B. Welsh. "Session Guarantees for Weakly Consistent Replicated Data." In Proc. Third Int'l Conf. on Parallel and Distributed Information Systems, pp. 140-149, Austin, TX, Sept. 1994. IEEE.