基于对象角色的高精度缓存替换算法 *

牛 伟[1], 成 娟[2], 翟正军[1], 郭阳明[1]
(1. 西北工业大学 计算机学院, 西安 710072; 2. 西安应用光学研究所, 西安 710065)

摘 要: 现有的 Web 缓存器的实现主要是基于传统的内存缓存算法,由于 Web 业务请求的异质性,传统的替换算法不能在 Web 环境中有效工作。研究了 Web 缓存替换操作的依据,分析了以往替换算法的不足,考虑到 Web 文档的大小、访问代价、访问频率、访问兴趣度以及最近一次被访问的时间对缓存替换的影响,提出了 Web 缓存对象角色的概念,建立了一种新的基于对象角色的高精度 Web 缓存替换算法(ORB 算法);并以 NASA 和 DEC 的代理服务器数据为例,将该算法与 LRU、LFU、SIZE、Hybrid 算法进行了仿真实验对比,结果证明,ORB 算法具有较好的性能表现。

关键词: 缓存; 替换算法; 文档命中率; 字节命中率; 角色
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2011)11-4089-03
doi:10.3969/j.issn.1001-3695.2011.11.023

High-precision cache replacement algorithm based on object role

NIU Wei[1], CHENG Juan[2], ZHAI Zheng-jun[1], GUO Yang-ming[1]
(1. *College of Computer, Northwestern Polytechnical University, Xi'an* 710072, *China*; 2. *Xi'an Institute of Applied Optics, Xi'an* 710065, *China*)

Abstract: Currently, the implementation of Web caching is mostly based on traditional cache updating algorithms. However, due to the diversity of the Web traffic pattern, the traditional algorithms for cache updating can not be used in Web environment effectively. This paper studied the basis of Web caching replacement operations, and analyzed the deficiency of previous replacement algorithm. As the size, the access cost, the access frequency and the recently visiting time of Web document affect caching replacement, proposed the concept of Web caching objects roles. Created a new high accuracy Web caching replacement algorithm (ORB algorithm) based on objects roles. In case of NASA and DES proxy server data is used for example, and compared the algorithm with LRU, LFU, SIZE, Hybrid algorithms. The result shows that ORB algorithm has better performance than the others.

Key words: caching; replacement algorithm; document hit-ratio; byte hit rate; role

0 引言

缓冲存储是一种使需要的数据更接近其处理场所的技术。通过将频繁请求的对象保留在本地服务器中,能明显地减少网络延时和满足用户对网络带宽的需求。目前,Web 缓冲存储器技术已经成为 Internet 基本结构中的重要组成部分。

与传统的内存缓存一样,Web 缓存的一个关键性问题是缓存内容的替换策略。现有的大多数代理缓存器是基于传统内存页调度机制来实现的,在 Web 环境中却不是一个好的替换策略[1]。其原因是 Web 文档大小的可变性很大且必须在 Internet 上传输,会经历很大延迟,而在内存缓存中,缓存对象(页)的大小和通信延迟都是不变的;另外,Web 文档的访问来自不同用户,应该考虑不同用户的兴趣度。因此,本文提出了一种基于对象角色的替换算法,该算法综合考虑了文档的大小、访问代价、访问频率、最近一次被访问的时间以及访问兴趣度,通过实验对性能指标进行测量,验证了该方法优于其他文献的方法。

1 现有研究

一个好的代理缓存的替换策略来源于对 WWW 业务访问特性的深刻认识,因此目前所提出的替换策略大部分来源于对 WWW 访问特性的分析。

a) LRU(least recently used)算法[2]。删除缓存中最近、最少使用的文档。算法实现简单,但没有考虑文档的尺寸和访问代价等。

b) LFU(least-frequently-used)算法[2]。最先移出缓存中最少使用的文档,其优点是很简单,其缺点除了 LRU 的缺点以外,如果没有失效机制,可能使过时的文档永远留在缓存器里。

c) SIZE 算法[3]。删除缓存中尺寸最大的文档,删除大的文档可以缓存更多的小文档,从而提高缓存命中率。其缺点是字节命中率偏低,可能会使得小文档永远留在缓存中。

d) Hybrid 算法[4]。主要目标是降低总的访问延迟,通过一个函数来计算每一个文档的替换权值。在进行替换操作时,删除具有最小替换权值的文档,而来自服务器 s 的文档 p 的函

收稿日期: 2011-04-01; **修回日期**: 2011-05-16 **基金项目**: 陕西省自然科学基金资助项目(2010HQ8005)

作者简介: 牛伟(1982-),男,河南洛阳人,博士研究生,主要研究方向为智能信息处理、可靠性和容错技术等(weiniu@126.com);成娟(1982-),女,河南济源人,工程师,硕士,主要研究方向为测控系统集成、半实物仿真等;翟正军(1965-),男,河南人,教授,主要研究方向为电子测试、虚拟现实及预测技术;郭阳明(1978-),男,讲师,博士,主要研究方向为智能信息处理等.

数值为

$$F=\frac{(c_s+\frac{w_b}{b_s})(n_p)^{w_n}}{z_p} \tag{1}$$

其中:c_s 是与服务器 s 连接的代价;b_s 是到服务器的带宽;n_p 是文档 p 的请求次数;z_p 是文档 p 的尺寸;w_b 和 w_n 是常量。该算法综合考虑了文档的尺寸、文档的访问代价以及文档的访问频率等,但最大的缺点是参数计算复杂。

e)Greedy dual-size 算法[5]。由最近最少使用算法 LRU 发展而来,该算法对每一存储在 Web 缓存中的文档 p 设置了一个关联权值 H,当需要将某文档存储到 Web 缓存时,初值 H 被设置为 1/size(总为正值);当进行替换时,具有最低 H 值的文档将被替换,同时所有存储在缓存中的文档 H 值减去某一最小值(被替换文档的 H 值);如果文档被再次访问,则其 H 值恢复为初值。因此,最近使用的网页将比长时间未用的网页具有更大的 H 值,通过时间的推移逐渐减小 H 值并在网页再次被存取时恢复。其缺点是没有考虑文档使用率和网络延迟。

2　ORB 替换算法

现有算法大多基于传统内存缓存算法,虽在实际中取得了一定的效果,但均存在不足之处。

a)对不同站点的文档考虑不足。例如,设缓存器中有两个大小相同的文档 a 和 b,a 来自用户感兴趣的、经常访问的站点 Sa,b 来自用户很少访问的网站 Sb,经过一段时间的访问,两个文档的替换权值 H 相等,当有替换发生时,应尽量将文档 a 留在缓存存储器中,目前的算法并不能做到这一点;其次,同一个 Web 站点,不同的缓存器有着不同的访问量[6]。通过分析得出,每一个缓存器一般存在着固定的用户群体,用户经常使用相同的缓存器去访问他们感兴趣的网站,不同的用户对不同的 Web 站点有着不同的兴趣度,从而不同的缓存器对不同的 Web 站点访问兴趣度也不同。

b)没有考虑文档最近一次访问的时间。例如有一个文档虽然它的访问次数很大,而其权值也比较大,但是由于近期内没有被访问,依据 Web 访问的时间局部性可知在将来的一段时间内可能访问不到,那么它的存在就会阻止其他的文档进入缓存。

针对以上问题,本文综合考虑 Web 文档对象特性,将每个文档赋予不同的角色,把文档的大小、访问频率、访问兴趣度、最近一次被访问的时间以及访问兴趣度作为角色的属性,每个角色根据其所具有的属性计算其价值。本文提出的算法根据每个文档的角色值 $R(p)$ 来评估 Web 文档的价值。

2.1　基于访问频率的改进

当某文档被再次点击时,增加 $R(p)$ 值,使其大于新进入文档 r 的权值,即 $R(r)<R(p)$。定义文档的权值公式为

$$R(p)=L+d(p)\times c(p)/s(p) \tag{2}$$

其中:$c(p)$ 为文档 p 的开销(下载时间、占用频带宽等);$s(p)$ 为文档 p 的大小;$d(p)$ 为文档 p 的访问次数;L 为一个膨胀因子。

为防止被多次点击的文档权值过大,应设置 $d(p)$ 的最大值 $\max_d$,这一设置可以防止早期文档永久地保留在缓存池中,$\max_d$ 值的大小可以根据实际运行情况或经验予以调整。

2.2　基于访问兴趣度的改进

Web 站点的访问规模不平衡,通常不足 10% 的站点承担着 80% 以上的用户访问,图 1 为 Boeing 公司 Web 访问结果[7]。分析表明,不同站点用户的访问兴趣度不同。

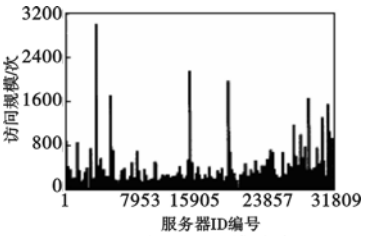

图1　站点访问规模分布

访问兴趣度定义如下:

$$I(p)=D_s/U_s \tag{3}$$

其中:I_p 表示文档 p 的访问兴趣度;D_s 表示缓存服务器 s 的总访问规模;U_s 表示缓存服务器 s 的不重复访问规模。

2.3　基于访问时间间隔的改进

计算文件的最近一次访问时间对角色值 $R(p)$ 造成的影响。令 $t=tk_{\text{now}}-t_{\text{last}}$(单位为 s),$t_{\text{now}}$ 为进行替换的时间,t_{last} 为文档的最近一次访问时间。$f(t)$ 应该随着 t 的增大而减小,不过 $f(t)$ 取值不当会把权值函数中其他项的影响淹没掉,使得算法基本上与 LRU 算法类似。考虑到 Web 请求具有时间局部性[7]:大约三分之一的再度访问发生在上次访问的一个小时内;大约三分之二的再度访问发生在上次访问后的 24 小时内。基于这种规律把 t 分为三种情况来考虑,$f(t)$ 表示为

$$f(t)=\begin{cases}h_1(t) & 0\leqslant t\leqslant 3600\\ h_2(t) & 3600<t<86400\\ h_3(t) & t>86400\end{cases} \tag{4}$$

其中:$h(t)$ 是随时间变化的函数。$t=0$ 表示缓存中没有的文件;$0<t<3600$ 表示上次访问发生在一个小时以内的文件;$3600<t<86400$ 表示上次访问发生在 24 小时以内的文件;$t>86400$ 表示上次访问发生在一天前的文件。从式(4)可以看出,只要适当地选择以上三个函数即可。如果函数取值太小,就不会表现出上次访问时间对函数值的影响;相反,如果取值过大,就会淹没其他因素的影响。当 $0<t<3600$ 时,认为这种文件在近期内还会访问,因此所占的比重应该比较大,选择常数0.5;当 $3600<t<86400$ 和 $t>86400$ 时,由于这两个区间内 t 的取值差距太大,为了减小这种差距,对 t 取对数,得 $f(t)$ 为

$$f(t)=\frac{C}{(\log t)^{\alpha}}=\begin{cases}0.5 & 0\leqslant t\leqslant 3600\\ 1/\log t & 3600\leqslant t\leqslant 86400\\ 1/2\log t & t>86400\end{cases} \tag{5}$$

考虑到文档的原始大小对权值函数的影响比较大,使得较大的文档被替换出去的可能性也相对增大,降低了字节命中率。为了提高算法字节命中率、减小算法对文档大小的依赖性,把文档大小改为 $\log(s(p))$,则文档 p 最终的角色值计算方法为

$$R(p)=L+d(p)\times c(p)\times I(p)\times f(t)/\log(s(p)) \tag{6}$$

3　算法精度检验

本文以缓存命中率(H)和字节命中率(BH)两个指标来衡量缓存算法的性能。

$$H = r/R \tag{7}$$

其中:r 为命中的次数;R 为用户访问文档的总次数。

$$BH = b/B \tag{8}$$

其中:b 为命中的字节数;B 为用户访问文档的总字节数。

缓存命中率与字节命中率的侧重点不同,缓存命中率侧重于减少用户的响应时间,而字节命中率则着眼于减少带宽开销。

4　实验分析

4.1　实验条件

Trace 驱动模拟是基于某些运行的代理服务器或 Web 服务器上对用户访问请求的跟踪记录文件,经过有选择的预处理(如过滤掉动态信息等)生成访问序列,利用这组访问序列作为缓存算法模拟系统的输入数据、仿真程序的流程如图 2 所示。

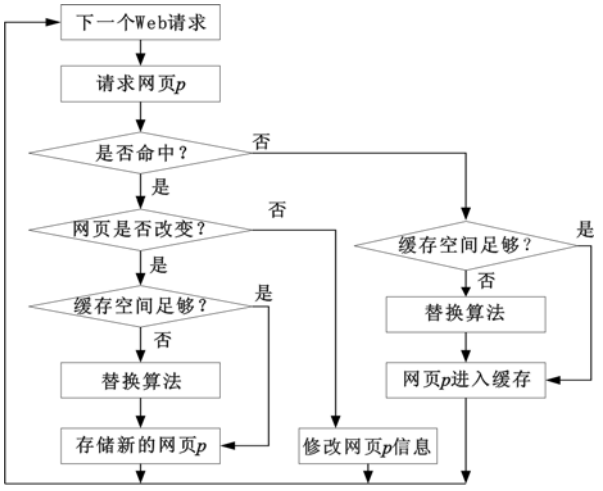

图2　仿真程序流程

4.2　实验结果与分析

实验采用公有的轨迹文件对算法进行比较,它们分别由 DEC[8] 和 NASA[9] 的代理服务器采集。DEC 代理服务器包含的数据是从 1996 年 9 月 1 日1996 年 9 月 22 日,每个域记录了请求时间、来自服务器的服务时间、大小和协议等。NASA 数据为 1995 年 8 月 1 日1995 年 8 月 31 日的所有访问信息。对日志文件先进行过滤处理,去除 CGI 类的不可缓存的文件,然后将处理后的日志作为输入,轨迹的基本统计量如表 1 所示。

表 1　轨迹统计

轨迹文件	总请求数	Web 文档大小总和
DEC	22 432 216	235 950.68
NASA	1 569 898	15 527.31

在缓存命中率和字节命中率两个性能指标上对 ORB 算法和以往的算法(LRU、LFU、SIZE、Hybird、GD-SIZE)作了对比,图 36 中显示了 LRU、LFU、SIZE、Hybird、GD-SIZE 和 ORB 算法在两种轨迹下的命中率情况。

从图 36 中可以看出,在缓存命中率方面,ORB、GD-SIZE、LFU 这三种算法要明显优于其他算法,而 Hybird 算法则表现最差。Hybird 算法由于没有考虑文档访问的时间局部性,缓存命中率较低。式(1)中参数 z_p 采用的是文档的原始大小,对文档的大小依赖性比较大,这样较大的文档被替换出去的可能性就相对较大,因此就导致算法的字节命中率较低。随着缓存尺寸的增加,SIZE 和 Hybrid 算法的字节命中率增加的幅度比较明显,因此这两种算法适用于缓存尺寸较大的系统。ORB 算法由于考虑到各种因素,性能表现始终很好;在考虑字节命中率时,ORB 算法由于调节权值函数的参数,性能也比其他算法要好。实验表明,当缓存尺寸无限增大时,由于基本上不考虑替换文档,所以各算法的性能趋于一种。

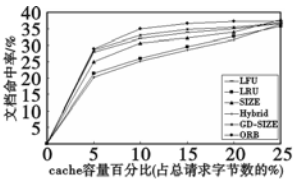

图3　文档命中率测试结果(DEC)

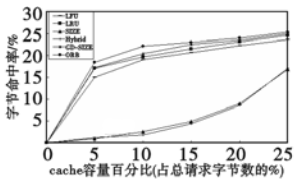

图4　字节命中率测试结果(DEC)

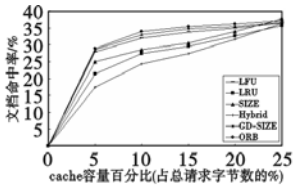

图5　文档命中率测试结果(NASA)

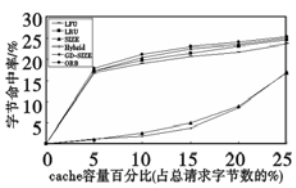

图6　字节命中率测试结果(NASA)

5　结束语

本文针对现有替换算法的局限性,提出了一种新的基于对象角色的高精度 ORB 算法,该算法综合考虑了 Web 文档的尺寸、访问代价、访问频率、最近一次被访问的时间以及访问兴趣度。轨迹驱动模拟实验表明,ORB 算法在缓存命中率与字节命中率方面优于其他算法。未来的工作主要包括动态内容的缓存、预测与预取算法。

参考文献:

[1] 林永旺,张大江,钱华林. Web 缓存的一种新的替换算法[J]. 软件学报, 2001,12(11): 1710-1715.

[2] TANENBAUM A S. Modern operating system[M]. Upper Saddle River: Prentice-Hall, 1992.

[3] WILLIAMS S, ABRAMS M, STANDRIDGE C R, *et al*. Removal policies in network caches for World-Wide Web documents[J]. ACM SIGCOMM Computer Communication Review, 1996, 26(4): 293-306.

[4] 贺琛,陈肇雄,黄河燕. Web 缓存技术综述[J]. 小型微型计算机系统,2004,25(5):836-842.

[5] HYOKYUNG B, KERN K, NOH S H, *et al*. Efficient replacement of nonuniform objects in Web caches[J]. Computer, 2002, 35(6): 65-73.

[6] 原福永,张微微. 一种新的代理缓存替换算法[J]. 燕山大学学报,2005,29(2): 123-127.

[7] RODRIGUEZ P, SPANNER C, BIERSACK E W. Analysis of Web caching architectures: hierarchical and distributed caching[J]. IEEE/ACM Trans on Networking, 2001, 9(4): 404-418.

[8] DAVISON B D. Predicting Web actions from HTML content[C]// Proc of the 13th ACM Conference on Hypertext and Hypermedia. New York: ACM Press, 2002: 159-168.

[9] Lawrence Berkeley National Laboratory[EB/OL]. (1995). http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.