Web 缓存的一种新的替换算法*

林永旺, 张大江, 钱华林

(中国科学院 计算机网络信息中心,北京 100080)

E-mail: dajiang.zhang@nokia.com

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,然而由于 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.首先给出了问题的一个最优化模型,分析了替换算法的关键在于能正确地体现 Web 业务的访问模式.在泊松到达模型的基础上,提出一种新的缓存策略——最少正规化代价替换算法(least normalized-cost,简称 LNC).新的替换算法除了考虑 Web 文档的平均引用时间、最近流逝时间、文档大小和单位大小价值以外,还考虑了 Web 业务的访问率动态改变的特征.对轨迹文件所做的性能实验表明,LNC 优于其他主要的算法.

关键词: WWW 业务;代理缓存;Web 轨迹;轨迹驱动;替换算法;泊松

中图法分类号: TP393 **文献标识码**: A

World Wide Web 业务是 Internet 上最主要的应用.基于 HTTP(hypertext transfer protocol)协议的请求产生的通信量在整个网络通信量中所占比重也在不断增加,在 1997 年大约占 60%.为了减少由此带来的网络拥塞和网络延迟问题,提高网络带宽的使用效率,人们在许多方面进行了研究.缓存(caching)技术被认为是提高 WWW 性能的最主要的方法.缓存技术将 WWW 工作方式从客户/服务器方式转变为分布式方式,增强了 WWW 的扩展性.

和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.比如,最近最少使用(LRU)策略,它在内存缓存中被认为是一个有效的算法,但在 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(从几百个字节到几兆)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,而内存中对页的访问来自单个程序.因此,很自然地要求有适合 Web 环境的新的缓存机制.本文提出了一种新的替换算法——最少正规化代价算法(least normalized-cost,简称 LNC).该算法综合考虑了文档大小、文档的访问频率、缓存器里文档的最近流逝时间以及缓存文档的价值.通过轨迹驱动试验对性能指标进行测量,我们发现,这种方法优于其他文献提出的方法.

1 已有结果

在文献[2]中,S. Hosseini 给出了缓存问题的一般化模型.Hosseini 给出了离线(即请求序列是预知的)和在线(请求序列未知)状态下的动态规划算法.在离线时,问题可以简化为 DAG(directed acyclic graph)图的一个最短路径问题.此时目标为寻找最短价值的路径.在线情况下,假定请求按

* 收稿日期: 1999-11-22; 修改日期: 2000-06-13

基金项目: 国家 863 高科技发展计划资助项目(863-306 ZD-08)

作者简介: 林永旺(1972—),男,福建永定人,博士,主要研究领域为计算机网络;张大江(1972),男,山东济南人,博士,主要研究领域为计算机网络;钱华林(1940—),男,上海人,研究员,博士生导师,主要研究领域为计算机网络.

Markov链到达,最小化每次请求产生的未命中文档的价值.然而,图的大小和Markov链的状态随物体数目的增长而成指数增长,离线问题是一个NP完全问题,因此算法不能用在实际的应用中.在实际应用时只能寻求次优解.

这一节我们给出近年来已提出的比较流行的缓存策略.首先我们给出缓存问题的一般化模型.然后讨论这些算法的概要,最后比较这些算法.

在性能的度量上,人们一般采用3种衡量标准:请求命中率、字节命中率和延迟率.定义如下:

请求命中率(request hit rate):文档在缓存的百分率.

字节命中率(byte hit rate):缓存器所传送的字节百分率.

延迟率(latency rate):下载未命中的文档所花的时间和下载所有文档的时间的比率.

为了进行性能比较,我们概括描述几个有代表性的替换算法,更多的算法可以参见文献[3].

• LRU(least-recently-used)算法:最先移出最近最少使用的文档.其优点是实现简单,在内存缓存中很有效.其缺点是没有考虑文档大小或延迟时间.

• LFU(least-frenquently-used)算法:最先移出最少使用的文档.其优点也是简单.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待在缓存器里.

• SIZE算法[1]:首先清除大文档.其优点是移出大文档,可以保留更多的小文档,产生更高的请求命中率.其缺点是可能使小文档永远留在缓存器中,字节命中率偏低,且再次下载大文档时,占用网络资源很多.

• GD-SIZE算法[3]:Greedy-Dual算法.这是个基于代价的贪婪算法.缓存器中的每个文档都有相应的价值H,当网页被带进缓存器时,该网页的H值为1/SIZE.发生替换时,最小H的文档(min$H$)被换出,剩下的文档的H值变为替换前的H值减去min$H$.如果文档被再次访问,则恢复其原来的价值.该算法的优点是不再被访问的文档会被清除,克服了SIZE算法的缺点,还可以使用更复杂的价值函数.然而1/SIZE函数在文献[3]中是所给出的性能标准上表现最好的函数.其缺点是没有考虑文档使用率和网络延迟.

• LRV(lowest relative value)算法[4]:在估计文档时,基于和文档相关的值V.V的计算是对轨迹数据进行经验分析后得到的.其优点是字节命中率优于其他算法.其缺点是,由于是对轨迹进行分析得到的,参数的选取依赖于特殊的轨迹.

• MIX算法[5]:它将几种文档属性综合起来考虑.算法的代价函数为$\frac{l^{r_1} * nref^{r_2}}{tref^{r_3} * S^{r_4}}$,$nref$是文档在缓存里的访问数,$tref$为现在的时间减去最近访问的时间,$r_i$是参数.其优点是在特定的参数下($r_1=0.1, r_2=r_3=r_4=1$),性能表现很好.其缺点是参数选取复杂.

2 LNC算法

这一节描述我们提出的替换算法——LNC.首先我们给出问题的一种最优化模型,然后描述如何实现我们的算法.

2.1 最优化模型

一般化模型只是一般性地对所有请求产生的代价作最小化比较,没有进一步考虑请求对象的引用率.对Web的业务研究[6]表明,对一个网络对象的访问可以用简单的开/关模型表述.即访问处于开(活跃)状态时,以恒定的速率访问该对象;而在关状态,访问速率大大降低,访问处于非活跃状态.开状态的持续时间分布和非活跃状态的时间分布可以用指数分布或Pareto分布来表示.对

象访问率随时间发生变化.因此我们有下面的缓存模型:

给定 n 个物体集合,记为 $U=\{1,2,\ldots,N\}$.每个物体 i 的大小为 $s_i(\geq 0)$,价值为 $C_i(\geq 0)$,时间 t 时的引用率 r_i^t,缓存器的容量为 B,$\delta_i=\begin{cases}1\\0\end{cases}$,其中 1 表示物体 i 在缓存器中,0 表示不在缓存器.问题可转化为下述问题:

目标函数:

$$\max C=\sum_{i=1}^{n}\delta_i * r_i^t * C_i \tag{1}$$

约束条件:

$$\sum_{i=1}^{n}s_i * \delta_i \leqslant B \tag{2}$$

寻找满足条件并使目标函数最大的解.考虑到实际缓存器容量大大超过文档的大小,可以将约束条件变为

$$\sum_{i=1}^{n}s_i * \delta_i \approx B, \tag{3}$$

即在缓存器里的文档大小之和近似等于缓存器容量.

最优化算法. 将各物体按 $r_i^tC_i/s_i$ 的比值排序,比值高的物体排在前面,比值低的物体排在后面,然后按从大到小的顺序将物体依次放入缓存器,直到缓存器满为止.此时得到的解即为最优化模型的满足条件(3)的最优解.

证明:不妨设 U 的子集 O_0 是由最优化算法得到的解,子集 O_1 为最优化模型的满足条件(3)的任一解,我们要证明 $\sum_{O_1}C_ir_i^t\leqslant\sum_{O_0}C_ir_i^t$.

由我们的选择,有

$$\sum_{O_1}C_ir_i^t/s_i\leqslant\sum_{O_0}C_ir_i^t/s_i. \tag{4}$$

不妨设 $O_1\cap O_0=\varnothing$,否则,消除双方相同的元素,式(4)仍成立.令

$$C_{\min}r_{\min}^t/s_{\min}=\min_{i\in O_0}C_ir_i^t/s_i$$

$$C_{\max}r_{\max}^t/s_{\max}=\max_{i\in O_1}C_ir_i^t/s_i$$

由于 O_0 与 O_1 的交为空,并且 O_0 是按最大比值选取的,我们有

$$C_{\min}r_{\min}^t/s_{\min}\geqslant C_{\max}r_{\max}^t/s_{\max},$$

否则与 O_0 的选取原则矛盾.最后我们得到:

$$\sum_{O_0}C_iR_i\geqslant C_{\min}R_{\min}/s_{\min}\cdot S_{cache}\geqslant C_{\max}R_{\max}/s_{\max}\cdot S_{cache}\geqslant\sum_{O_1}C_iR_i. \qquad \square$$

从最优化算法我们可以看到,如何正确地选择引用参数 r^t,以较好地反映实际文档的访问模式,是实际算法是否有效的关键.另一方面,通过灵活地选择价值 C,我们可以得到适合不同要求(ISP、用户等)的替换算法.例如,当 $C=1$(或某一常数)时,我们的目标是取得最好的 RHR;当价值 C 为文档大小时,目标是最大限度地降低网络通信量,即 BHR 最大;当 C 为文档延迟时,目标是 LR 最小.另外,可以根据特定的要求来选择自己的优化策略.

2.2 LNC 算法

以往的算法大都是根据特殊的"兴趣"值来选择合适的价值,有的需要很好地估计参数,在对引用参数的估计上需要复杂的计算和大量的模拟.还有的算法当从一种性能转到另一种性能时缺乏

灵活性.这里,我们提出一种有效的算法——最少正规化代价算法(LNC).LNC 算法是最优化算法的在线近似.其描述为:当缓存器满或高负荷时,如果有新的 Web 文档到达,那么依次替换代价最少的文档,直到新的文档能被存入缓存器为止.代价函数的选取如下所述.

LNC 算法根据引用率评估 Web 文档的代价.引用率是一个反映文档未来被访问的可能性大小的量,和文档单元价值一起形成一个公平、一致性的代价.其描述如下:

设 $C(i)$ 是清除文档 i 的代价,c_i 是文档的价值,s_i 是文档大小.引用率为 F_i,那么由最优化模型,文档 i 代价为 $C(i)=F_i*c_i/s_i$,其中 c_i/s_i 是文档的单元价值.

引用率 F_i 必须能够体现未来文档被访问的可能性大小.但是,由于未来的访问是未知的,只能采用预测的方法,算法的好坏与能否很好地近似体现文档的访问模式密切相关.对引用率 F_i,我们的计算方法如下:

设 t_c 是最近一次访问文档后流逝的时间,t_k 是第 k 次访问文档和第 $k-1$ 次访问文档之间的时间间隔,设第 $k-1$ 次访问文档后的平均访问间隔时间为 λ_{k-1},那么第 k 次访问文档后得到的平均访问间隔时间为

$$\lambda_k=\alpha t_k+(1-\alpha)\lambda_{k-1},$$

其中 α 是参数,α 大于等于 1/2 即可.λ 反映了文档的当前的访问率,当文档访问率发生变化时,能迅速灵活地反映这种变化.令 λ_f 是最后一次访问文档后得到的平均访问间隔,由指数分布的定义,我们可以得到它的 p.d.f.为

$$f(x)=(1/\lambda_f)e^{-(1/\lambda_f)x}.$$

那么,文档经过时间 t_c 后的被访问的 p.d.f 为

$$f_c(x)=(1/\lambda_f)e^{-(1/\lambda_f)(x-t_c)},x\geqslant t_c.$$

文档下一次被访问的平均时间间隔为

$$\int_t^{\infty}xf_c(x)dx=\int_t^{\infty}x(1/\lambda_f)e^{-(1/\lambda_f)(x-t_c)}dx=t_c+\lambda_f,$$

它的平均引用率为 $F=1/(t_c+\lambda_f)$.

引用率体现了在当前时刻,文档下一次被引用的可能性的大小.从引用率的表达式可以看出,它由文档的访问历史和现在流逝的时间组成.在流逝时间相同的情况下,过去引用率$(1/\lambda)$高的文档,未来的引用率也高,而如果流逝的时间增加,引用率则自然下降.这与实际的观测结果是一致的.

LNC 算法是一个高效的算法.与其他算法不同的是,它有 LRU 算法的简洁,对文档代价的计算既不需要保留以前的访问记录,又不需要有复杂的参数估计,同时还能迅速地体现文档访问率的变动情况.由于采用文档单位价值作为权,可以客观地反映清除文档所需花费的代价.对于只访问过一次的文档,只需简单地选取 $1/t_c$ 为引用率.

3 性能实验

我们使用公有的轨迹文件对算法进行比较,它们分别由 DEC 公司[7]和 NLANR 实验室[8]采集.DEC 公司的轨迹是由 Squid 代理产生的,包含的数据是从 1996.9.1 到 1996.9.22,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.NLANR 采用 sj 类的轨迹文档,数据为 1999.8.1到 1999.8.7 的一个星期.我们在 Linux 平台上进行实验,在模拟器[3]中添加了新的算法模块,实验的过程模拟实际的 Web 缓存过程.当有新的请求到达模拟器时,检查缓存器的内容,如

果请求的文档在缓存器中,则更新引用率、延迟和最后一次引用时间;否则,将文档载入缓存器,按照缓存替换策略清除文档.对于CGI-BIN类的动态文档则进行过滤.不同的性能采用不同的价值,缓存器大小按占文档大小总和的0.5%到35%增加运行模拟器.轨迹的基本统计量见表1.

Table 1

表1

Trace file①	Number of requests②	Total size of Web files③
DEC	609 951	6 415.64
NLANR	1 737 526	14 072.53

①轨迹文件,②总请求数,③Web文档大小总和.

我们在RHR和BHR两个性能指标上对LNC算法和以往的算法(LRU,LFU,SIZE,GD-SIZE,MIX)作了一个比较.从图1~4可以看出,在请求命中率方面,LNC,MIX,GD-Size这3种算法要明显优于其他算法,而LRU算法则表现最差.LNC算法由于考虑到各个因数,性能表现始终很好,在缓存器大小为文档总大小的35%时,即近似达到了理想的无限大缓存情形(此时没有替换问题).在考虑字节命中率时,LNC算法由于可以灵活地设置参数,性能始终比其他算法性能要好.最后,可以观测到,当缓存器容量增大到无穷大时,由于基本上不替换缓存器中的文档,各种策略的性能趋于一致.

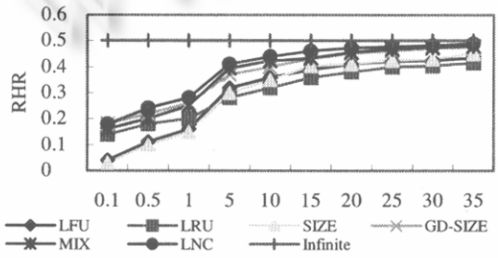

Fig. 1 RHR vs. buffer size (trace:DEC)

图1 请求命中率与缓存器大小的关系(轨迹:DEC)

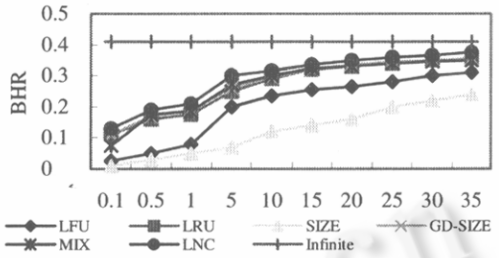

Fig. 2 BHR vs. buffer size (trace:DEC)

图2 字节命中率与缓存器大小的关系(轨迹:DEC)

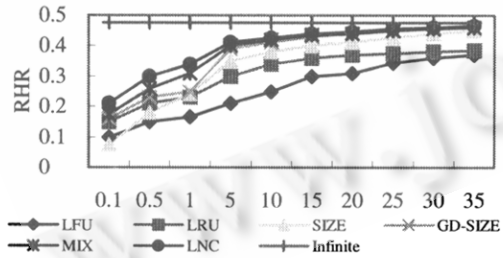

Fig. 3 RHR vs. buffer size (trace:NLANR)

图3 请求命中率与缓存器大小capacity的关系(轨迹:NLANR)

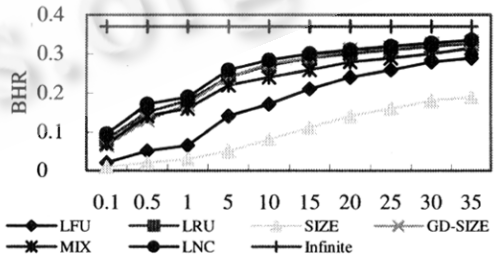

Fig. 4 BHR vs. buffer size (trace:NLANR)

图4 字节命中率与缓存器大小capacity的关系(轨迹:NLANR)

4 结 论

我们在对缓存问题提出一个最优化模型的基础上,给出并检验了最少正规化代价替换算法(LNC).综合考虑影响Web文档的各种因素,在泊松到达模型的基础上,给出了缓存关键性参数——访问率的计算.性能实验表明LNC优于其他几种算法.未来的工作主要包括动态内容的缓存、主动缓存服务器以及传输协议的发展对缓存的影响.

References:

[1] Williams, S., Abrams, M., Standridge, C., *et al*. Fox removal policies in network caches for world-wide web documents. In: Steenstrup, M., ed. Proceedings of the ACM SIGCOMM '96. ACM Press., 1996. 293~306.

[2] Hosseini, S., Cox, J. R. Optimal solution of off-line and on-line generalized caching. Technical Report, WUCS-96-20, Washington University at St. Louis, 1996.

[3] Cao, P., Irani, S. Cost-Aware www proxy caching algorithms. In: Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems. 1997. 193~206.

[4] Lorenzetti, P., Rizzo, L., Vicisanco, L. Replacement policies for a proxy cache. 1997. http://www.iet.unipi.it/luigi/research.html.

[5] Nicolausses, N., Liu, Z., Nain, P. A new efficient caching policy for the World Wide Web. 1998. http://www.cs.wisc.edu/~cao/wisp98/fina-versions/Nicola.ps.

[6] Willinger, W., Taqqu, M., Sherman, R., *et al*. Self-Similarity through high variability: statical ananlysis of ethernet LAN traffic at the source level. IEEE/ACM Transactions on Networking, 1997,5(1):71~86.

[7] Digital's Web proxy traces. 1996. ftp://ftp.digital.com/pub/DEC/traces/proxy/tracelistv1.2.html.

[8] ftp://ircache.nlanr.net/Traces/.

A Novel Replacement Algorithm for Web Caching*

LIN Yong-wang, ZHANG Da-jiang, QIAN Hua-lin

(*Computer Network Information Center, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: dajiang.zhang@nokia.com

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. In this paper, an optimized model to the problem is presented. The analytic result shows that the key issue for the cache updating algorithms is how the algorithm suits the WEB traffic pattern properly. Based on the Poisson arrival model, a new cache policy, Least Normalized Cost (LNC), is proposed. In addition to the consideration of the average reference time duration, the recently passed time, the size of the WEB file and the cost per unit of file, the dynamic characteristic of WEB access rate is also taken into account. The trace driven simulation shows that the performance of the algorithm LNC is better than that of the existing algorithms proposed in the literature.

Key words: WWW service; proxy caching; Web trace; trace-driven; replacement algorithm; Poisson

* Received November 22, 1999; accepted June 13, 2000

Supported by the National High Technology Development 863 Program of China under Grant No. 863-306-ZD-08