文章编号:1674 - 2869(2018)06 - 0685 - 06

贪婪双尺寸频率算法的优化与改进

黎慧源[1,2],易国洪[*1,2],代 瑜[1],冯智莉[1]
1. 智能机器人湖北省重点实验室(武汉工程大学),湖北 武汉 430205;
2. 武汉工程大学计算机科学与工程学院,湖北 武汉 430205

摘 要:针对贪婪双尺寸频率算法不能反映频率未来走势的问题,在贪婪双尺寸频率算法的基础上,提出了一种新的缓存替换算法。该算法通过对文件和系统的访问次数进行周期性的计数,得到了文件的平均周期访问频率、最近周期访问频率和周期相对频率,并通过周期相对频率来体现文件访问频率的未来走势,弥补了传统的贪婪双尺寸频率算法的不足。新的缓存替换算法具有良好的适应性,提供了周期次数N和频率影响程度k这两个参数。使用者可以通过调整这两个参数让算法适应实际的访问情况。在符合齐普夫定律的数据下进行实验,结果表明该算法比传统的贪婪双尺寸频率算法具有更高的缓存命中率。

关键词:缓存替换算法;贪婪双尺寸频率算法;相对频率;访问周期;命中率
中图分类号:TP393 **文献标识码**:A **doi**:10. 3969/j. issn. 1674-2869. 2018. 06. 020

Optimization and Improvement of Greedy Dual Size Frequency Algorithm

LI Huiyuan[1,2], *YI Guohong*[*1,2], *DAI Yu*[1], *FENG Zhili*[1]
1. Hubei Key Laboratory of Intelligent Robot (Wuhan Institute of Technology), Wuhan 430205, China;
2. School of Computer Science & Engineering, Wuhan Institute of Technology, Wuhan 430205, China

Abstract: To solve the problem that the Greedy Dual Size Frequency (GDSF) algorithm can not reflect the future trend of frequency, we proposed a new cache replacement algorithm. The algorithm obtained the average cycle access frequency, the most recent cycle access frequency and the relative frequency of the file by periodically counting the number of accesses to the files and the systems. This method reflected the future trend of the file access frequency through the periodic relative frequency to overcome the shortcomings of the traditional GDSF algorithm. The proposed new cache replacement algorithm had good adaptability to help the user adjust parameters of the number of cycles N and the degree of frequency influence k for adapting the actual access situation. Experiments based on the data of Zipf's law show that the algorithm achieves a higher cache hit ratio than the traditional GDSF algorithm.

Keywords: cache replacement algorithm; GDSF algorithm; relative frequency; access cycle; hit ratio

随着互联网的发展,Web服务器的压力越来越大,用户对网页的响应速度要求也越来越高。为了减轻Web服务器在高并发访问情况下的压力,缩短用户访问Web网站的时间延迟,可以在Web服务器和用户之间增设Web代理服务器。Web代理服务器将一些经常被访问到的Web对象缓存在代理服务器中,当用户由客户端向Web服务器发出请求时,请求将被发送到代理服务器,由代理服务器来进行处理。如果代理服务器中已经存有用户请求的Web对象,则经过代理服务器对缓存对象的一致性和有效性进行判断后,直接将有效的缓存对象返回给用户,或者去源Web服务

收稿日期:2018-09-03
基金项目:国家自然科学基金(61502355)
作者简介:黎慧源,硕士研究生。E-mail:954478980@qq.com
***通讯作者**:易国洪,硕士,副教授。 E-mail:yiguohong@wit.edu.cn
引文格式:黎慧源,易国洪,代瑜,等. 贪婪双尺寸频率算法的优化与改进[J]. 武汉工程大学学报,2018,40(6):685-690.

器上取回最新的Web对象，存储在代理服务器中后，再将对象返回给用户。当代理服务器缓存空间不足时，缓存机制会根据缓存替换算法会将某些缓存对象移出缓存空间来存储新的Web对象[1]。因此，缓存替换算法的好坏是决定代理服务器性能的重要因素之一。

最著名和最基本的缓存替换算法有最近最少使用替换(Least Recently Used，LRU)算法[2]，该算法优先替换掉那些离上一次被访问时间的时间间隔最长的对象。先进先出(First In First Out，FIFO)算法[3]，优先替换掉最先进入缓存区的对象。基于对象大小(Size)替换算法[4]，将缓存中最大的对象替换出去。还有最不经常使用替换算法，该算法将自进入内存后使用次数最少的对象替换出去。然而，它们有一个主要的不足之处，即它们仅依赖于一种流量特性，例如访问频率、驻留时间或最近访问时间。算法考虑的因素单一，所以造成了以上4种算法的性能有限。

考虑到以上基本算法的不足，研究者们提出了综合考虑多因素的缓存替换算法。最小使用价值算法[5]包括Cao和Irani[6]在1997年提出贪婪双尺寸算法，Cherkasova[7]在1998年提出的贪婪双尺寸频率(Greedy Dual Size Frequency，GDSF)算法，Lee等[8]在2001年提出的最近最少经常使用算法，以及文献[9]提出的基于保存价值的Hybrid算法。此类算法通常从文件对象的大小、访问频率、访问延迟等多个影响缓存性能的方面进行考虑，通过一个价值函数来计算每个对象的保存价值，当需要发生替换缓存时，优先将保存价值最小的对象替换出去。

在最近几年缓存替换算法也取得了一些新的研究成果。Sajeev和Sebastian[10]在2011年提出了一种新的网络缓存对象分类方案，该方案使用多项逻辑回归技术对缓存对象进行分类。2012年，Ali和Shamsuddin[11]提出了基于机器学习技术的智能Web代理缓存方法，以Web代理日志文件作为训练数据，用支持向量机预测稍后可能重新访问的Web对象，与传统Web代理缓存技术LRU和GDSF结合，形成名为SVM-LRU和SVM-GDSF的智能缓存方法。2015年，Negrão和Roque[12]提出了一种用于Web缓存的自适应语义感知替换算法，它为缓存替换过程添加了空间维度，根据从一个对象导航到另一个对象所需的链接数量来测量对象之间的距离，发生替换时，将远离最近访问的页面的对象作为移除的候选者。2017年，Benhamida和Bouallouche-Medjkoune[13]提出了使用相对频率来替换传统的访问频率，相对频率是使用文档访问次数及其在缓存中的生命周期计算的。

文档的访问频率是缓存替换算法的重要考虑因素，本文引入了平均周期访问频率、最近周期访问频率、周期相对频率3个特性，将周期相对频率作为对象流行度的重要影响因子，综合了对象大小、网络带宽因素，提出了贪婪双尺寸周期相对频率(Greedy Dual Size Frequency Periodic Relative Frequency，GDSF-PRF)算法，在用户访问网页的习惯符合Zipf[14]定律的情况下，经过与LRU、FIFO、GDSF算法的实验对比，验证了GDSF-PRF在访问命中率和字节命中率上有更好的表现。

1 相关工作

1.1 缓存替换算法的性能指标

缓存替换算法通常使用的性能评价指标有3个：请求命中率、字节命中率和访问延迟[15]。在发生数据请求时，如果缓存中已经保存了该数据，并且该缓存数据有效，可以直接将该缓存数据返回给用户，则称此次数据请求为一次命中；否则要向原始数据所在的服务器请求数据，此时称为一次缺失，即未命中。

1)请求命中率

请求命中率是代理缓存服务器缓存命中的次数占用户发出的请求总次数的百分比，用 R_h 表示。用 N_r 表示请求总次数，用 N_h 表示缓存命中次数，则 R_h 可表示为：

$$R_h = \frac{N_h}{N_r} \tag{1}$$

2)字节命中率

字节命中率是代理缓存服务器本身向用户发出的字节数与用户总的请求数之比，用 R_{bh} 表示。用 B_h 表示缓存命中字节数，B_r 表示代理接收到的用户总请求字节数，则 R_{bh} 可表示为：

$$R_{bh} = \frac{B_h}{B_r} \tag{2}$$

3)平均访问延迟

平均访问延迟是指从客户端请求数据到收到数据所需的平均时间，用 t_{al} 表示。平均时间越短，缓存性能越好。用 N_r 表示请求总次数，用 t 表示 N_r 次请求总的访问延迟时间，则 t_{al} 可表示为：

$$t_{al} = \frac{t}{N_r} \tag{3}$$

1.2　GDSF算法

GDSF算法，即贪婪双尺寸频率缓存替换算法[16]。该算法的基本思想是使用某个目标保存价值函数来计算出缓存中所有缓存对象的保存价值H，当缓存剩余空间不足以保存新的对象时，根据这个H值将缓存对象由高到低进行排序，优先将缓存中保存价值H最低的对象替换出去。其目标函数为：

$$H(i)=L+F_{\mathrm{r}}(i)\times\frac{V(i)}{S(i)} \tag{4}$$

$H(i)$表示第i个缓存对象的缓存价值，$V(i)$表示将Web对象i引入到缓存中的所需要付出的代价，如带宽、网络延迟等。L为膨胀因子，初值为0，每当有Web对象替换出去时，L都会被重新赋值为那个被替换出去的对象的$H(i)$。$F(i)$为Web对象i的访问次数。该算法综合考虑了对象大小和访问频率因素，但是没有考虑目标的将来流行度因素，会造成较小对象且较短时间内访问频率大的缓存对象常驻在缓存空间中。

1.3　已有的GDSF改进算法

从文件大小角度来看，在GDSF算法中，文件大小越小，文件的H值越大，文件被替换出去的可能性就越小，这就导致小文件被存储的概率增大，大文件被替换出去的概率增大，导致了较低的字节命中率。因此，为了提高请求的命中率，降低文件大小对文件保留价值的影响程度，部分改进方案中，将$S(i)$调整为$\log_2[S(i)]$，即

$$H(i)=L+F_{\mathrm{r}}(i)\times\frac{V(i)}{\log_2[S(i)]} \tag{5}$$

从时间角度来看，文件再次被访问的可能性随时间的延长而降低。因此在计算对象保留价值时，还应考虑对象的时间价值。GDSF算法虽然考虑了文件的访问频率但是没有考虑时间因素。因此，有研究者引入了平均访问频度的概念，提出了GDSF-T(Greedy-Dual-Size-Frequency-Time)算法[17]。

定义 在时间距离t内，Web访问对象i的平均访问频度为：

$$G(i,t)=\frac{F_{\mathrm{r}}(i)}{t}=\begin{cases}\dfrac{F_{\mathrm{r}}(i)}{t_{\mathrm{sys}}-t_{\mathrm{store\_time}}} & t_{\mathrm{sys}}>t_{\mathrm{store\_time}}\\ 1, & t_{\mathrm{sys}}=t_{\mathrm{store\_time}}\end{cases} \tag{6}$$

其中，$F_{\mathrm{r}}(i)$表示Web对象i在时间t内，被访问的次数。t_{sys}表示缓存替换发生时的系统时间，$t_{\mathrm{store_time}}$表示对象i进入缓存时的时间。改进后的算法GDSF-T的如下：

$$H(i)=L+\frac{V(i)}{\log_2[S(i)]}\times G(i,t) \tag{7}$$

2　GDSF-PRF算法设计

2.1　频率因素的改进

GDSF算法中考虑了文件的访问频率因素，但是单纯的考虑文件的访问频率不能很好的反映文件的最近访问热度和将来可能热度，可能造成曾今某一个时段被高频率访问而近期未被访问的文件长时间驻留缓存的情况发生。因此，已经有研究者尝试对频率因素的考量进行改进，如1.3节中提到的GDSF-T算法，该算法引入了平均访问频度的概念来代替GDSF算法原有的单纯的访问频度因素，在文件访问频率的基础上增加了对文件的缓存驻留时间的考量，使得文件的平均访问频度随驻留时间的增长而降低，考虑的因素更全面，但是需要记录下每个文件初次进入缓存的时间，造成了额外的空间消耗，并且没有考虑到文件的最近访问时间。使用平均访问频度可以大致的描述文件在过去的访问热度，不能很好的预测文件将来可能的热度。

因此，本文引入了平均周期访问频率、最近周期访问频率、周期相对频率3个特性。根据文件的平均周期访问频率和最近周期访问频率，得到周期相对频率，分析出文件访问频率的周期走势。对文件的访问频率进行周期性的计数，本文的访问周期使用次数周期N来代替时间周期，以此来避免对文件访问时间的存储，减小空间消耗。

平均周期访问频率描述的是从文件进入缓存至今，在被访问过的周期里，平均每个周期被访问的频率。用$F(i)$表示文件i进入缓存后被访问的次数，$N_{\mathrm{c}}(i)$表示文件i被访问过的周期数，则文件i的平均周期访问频率$F_{\mathrm{a}}(i)$表示为：

$$F_{\mathrm{a}}(i)=\frac{F(i)}{N_{\mathrm{c}}(i)} \tag{8}$$

最近周期访问频率描述的是文件在最近的一个周期里被访问的频率。N_{now}表示当前周期从周期开始时刻起至今的访问次数，$F_n(i)$表示文件i在当前周期里被访问的次数，则文件i的最近周期访问频率$F_{\mathrm{c}}(i)$表示为：

$$F_{\mathrm{c}}(i)=\frac{F_n(i)}{N_{\mathrm{now}}} \tag{9}$$

周期相对频率根据文件的平均周期访问频率和最近周期访问频率得到，周期相对频率描述的是文件的周期走势。如果文件i的最近周期频率$F_{\mathrm{c}}(i)$小于文件i的平均访问周期频率$F_{\mathrm{a}}(i)$，则可以说明文件i被访问的频率呈下降趋势，在未来被

访问的相对频率小，发生缓存替换时文件i将有更大的可能性被替换出缓存空间。如果文件i的最近周期频率$F_c(i)$大于文件i的平均访问周期频率$F_a(i)$，则可以说明文件i被访问的频率呈上升趋势，在未来被访问的相对频率大，发生缓存替换时文件i将有更小的可能性被替换出缓存空间。k为参数，可根据业务调整k的值来调整相对频率对保留价值的的影响力度。周期相对频率$F_{pr}(i)$的表达式如下：

$$F_{pr}(i)=[F_c(i)-F_a(i)]^k \tag{10}$$

2.2 GDSF-PRF算法描述

GDSF-PRF算法将GDSF算法公式(5)中的频率$F_r(i)$替换成周期相对频率$F_{pr}(i)$，即公式(10)，目标函数如下：

$$H(i)=L+\frac{V(i)}{\ln S(i)}\times F_{pr}(i) \tag{11}$$

L为膨胀因子，初值为0，当缓存中有Web对象被替换出去时，L都会被重新赋值为那个被替换出去的对象的$H(i)$。

$V(i)$为缓存空间从Web服务器获取Web对象i需要付出的代价，本文选择数据包的传送个数作为代价，TCP分段大小为536 bytes，因此$V(i)$的计算公式如下：

$$V(i)=2+\frac{S(i)}{536} \tag{12}$$

式(12)中$S(i)$为Web对象的字节大小。

根据公式(11)，可得如下GDSF-PRF算法的伪代码：

```
Begin
输入:要访问的文件d
if 数据d not in cache
while 缓存空闲大小<Size(d)
计算H,得到最小H_min值的文件i
L=H_min
移出i文件
存入文件d
F(d)=1,F_n(d)=1,N_c(i)=1
N_now=N_now+1
F(i)=F(i)+1
F_n(i)=F_n(i)+1
if N_now等于N
if F_n(i)大于0
N_c(i)=N_c(i)+1
N_now=1,F_n(i)=1;
end
```

2.3 算法参数的设定

GDSF-PRF需要预先设定的参数：

1)参数k决定了相对频率对保留价值的影响程度，设定的k值需要使得保留价值与相对频率的影响度相近，一般选取1到5之间的奇数。

2)访问周期N的设置。访问周期N用来确定更新的次数间隔。N的大小会影响更新的频率，进而影响系统的开销和数据存放在缓存中的时间。达到一个次数周期后需要对所有数据进行一次遍历，因此达到N次访问的时间间隔应该大于遍历与更新数据系统所需要的时间开销。N的设置也和业务的数据流行度随时间的变化快慢有关。如果按缓存对象被访问的次数对缓存对象进行由高到低的排序，用r来代表缓存对象在排序表中的序号，用f来表示该缓存对象被访问的频率，则当网站的数据符合齐普夫(Zipf)定律时，r和f的乘积是一个常数，该常数用C来表示。这种规律用公式可表示为：

$$r\times f=C \tag{13}$$

根据M L Hanley[18]有关James Joyce Ulysses的用词数据统计发现，Zipf定律的常数C乘以10与访问总次数很接近。若服务器缓存可以存储m个对象，排名第m的对象周期内被访问的次数为n次，访问周期为N，根据Zipf定律有：

$$m\times n\times 10=N \tag{14}$$

为了让访问次数最低的一个对象在一个访问周期里面都至少能被访问到一次，即$n\geqslant 1$，就需要满足

$$N\geqslant 10m \tag{15}$$

如若缓存服务器能缓存100个平均大小的文件，即m=100，则次数周期N可设置为1 000。

3 实验与分析

3.1 实验环境

实验使用的Web服务器配置为：Intel Xeon L5520的2.4 GHz处理器2个，16 GB的RAM，10 MB带宽，运行Windows Server 2008系统。

代理服务器的配置为Intel Xeon L5520 2.4 GHz处理器，1 GB的RAM，4 MB带宽，运行Red Hat Linux 9.0系统。代理服务器上使用squid反向代理实现，修改Squid中/src/repl目录下的缓存替换算法的源码，即可实现不同缓存算法的对比。

3.2 实验数据与参数设置

互联网用户访问习惯符合Zipf定律，为了测试算法性能，本文选取的实验数据集参数如表1所

示。表2描述的是代理服务器不同的缓存容量对应的访问周期N的设定。公式(10)中的参数k设置为3。

表1　测试数据集参数

Tab. 1　Parameters of test data set

属性	值
缓存容量 / MB	10~80
对象个数 / 个	100
对象总大小 / MB	100
请求次数 / 次	1000
请求总大小 / GB	1
高频对象 / 个	20
低频对象 / 个	80

实验缓存容量为10 MB ~ 80 MB，测试不同缓存容量下，各种算法的命中率，访问对象一个访问周期内的访问次数设定如表2所示。

表2　访问次数与缓存大小的关系

Tab. 2　Relationship between access cycle and cache size

缓存大小 / MB	访问次数
10	100
20	200
30	300
⋮	⋮
70	700
80	800

对象集合包含对象100个，总大小为100 MB。高频对象指访问频率占请求频率80%的对象，低频对象指访问频率占请求频率20%的对象，高频对象和低频对象的比例符合Zipf定律，即“二八原则”，高频对象有20个，低频对象有80个。

在放置请求集合时为了满足Zipf定律，即20%的内容会占有80%的访问量，用JAVA程序在100个文件中随机选取20个文件，对这20个文件进行800次访问，对其他文件进行200次访问，并随机打乱访问顺序，以此生成一组符合Zipf规律的访问序列，请求集合总大小约为1 GB。

3.3　多种替换算法对比实验

对LRU、FIFO、GDSF、GDSF-PRF算法进行请求命中率和字节命中率的比较。Web服务器存放100个文件，设定缓存容量分别为Web服务器文件总大小的10%，20%，依次至80%，对缓存进行1000次访问，分别调用LRU、FIFO、GDSF、GDSF-PRF算法，得到它们的请求命中率和字节命中率。

图1为4种缓存替换算法的请求命中率比较，GDSF-PRF的请求命中率比其他3种替换算法有明显提高，尤其在缓存容量占30%时，GDSF-PRF的请求命中率比位于第二的FIFO提高了30%。

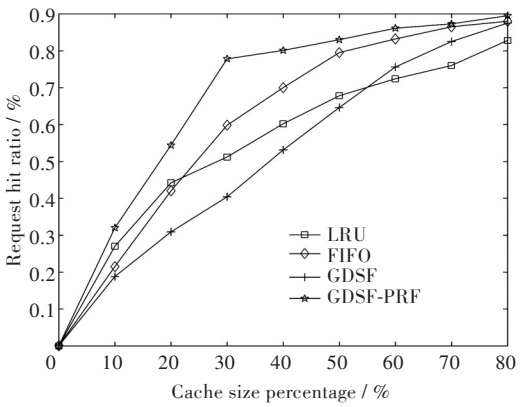

图1　多种替换算法的请求命中率

Fig. 1　Request hit ratio of replacement algorithms

图2为4种缓存替换算法的字节命中率比较。由图2可知，GDSF-PRF的字节命中率仅在缓存容量为10%以下时，比LRU算法略低，在缓存容量为10%以上时，字节命中率比其他3种替换算法有所提高，尤其在缓存容量占30%时，GDSF-PRF的字节命中率比位于第二的FIFO提高了32%。

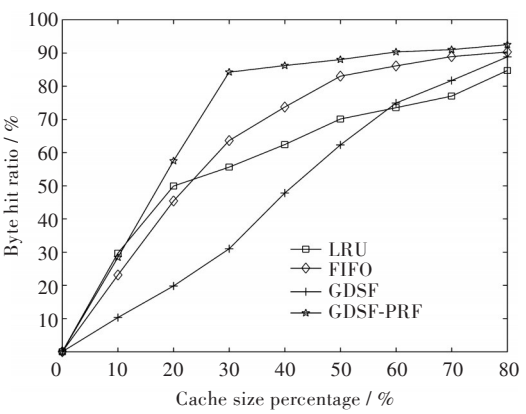

图2　多种替换算法的字节命中率

Fig. 2　Byte hit ratio of replacement algorithms

由实验结果可知，当用户访问习惯符合Zipf规律时，其请求命中率和字节命中率与LRU、FIFO、GDSF比有显著提高。

4　结　语

以上比较和分析了LRU、FIFO、GDSF系列算法，在GDSF算法的基础上，提出了GDSF-PRF算法，然后利用实验对比，证明了此算法的有效性。下一步的研究将是在缓存替换发生之前，增加预测模型，通过预测手段进一步减小Web服务器的压力。

参考文献：

[1] 孙晓星. 基于WEB访问特性的代理缓存机制的研究[D]. 哈尔滨:哈尔滨工程大学, 2011.

[2] 张震波,杨鹤标,马振华,基于LRU算法的Web系统缓存机制[J]. 计算机工程,2006,32(19):68-70.

[3] MICHAEL M M, SCOTT M L. Nonblocking algorithms and preemption-safe locking on multiprogrammed shared memory multiprocessors[J]. Journal of Parallel and Distributed Computing, 1998, 51(1):1-26.

[4] 孟兆会. 基于访问对象大小的动态调节精简缓存摘要算法[D]. 太原:太原理工大学,2008.

[5] 李聪, 温东新. 基于突发集中性访问模式的缓存替换算法[J]. 计算机工程, 2017, 43(1):105-108.

[6] CAO P, IRANI S. Cost-aware WWW proxy caching algorithms [C]// Usenix Symposium on Internet Technologies and Systems on Usenix Symposium on Internet Technologies and Systems, Berkeley: USENIX Association, 1997:18-18.

[7] CHERKASOVA L. Improving WWW proxies performance with Greedy-Dual-Size-Frequency caching policy[J]. Hp Laboratories Technical Report, 1998, 69(1):1-14.

[8] LEE D, CHOI J, KIM J H, et al. LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies [J]. IEEE Transactions on Computers, 2001, 50(12):1352-1361.

[9] 张柏礼, 吕建华, 姚蓓,等. Web代理服务器缓存置换算法研究[J]. 计算机科学与探索, 2010, 4(11):977-983.

[10] SAJEEV G P, SEBASTIAN M P. A novel content classification scheme for web caches[J]. Evolving Systems, 2011, 2(2):101-118.

[11] ALI W, SHAMSUDDIN S M, ISMAIL A S. Intelligent web proxy caching approaches based on machine learning techniques[J]. Decision Support Systems, 2012,53(3): 565-579.

[12] NEGRAO A P, ROQUE C, FERREIRA P, et al. An adaptive semantics-aware replacement algorithm for web caching [J]. Journal of Internet Services & Applications, 2015, 6(1):1-14.

[13] BENHAMIDA N, BOUALLLOUCHE-MEDJKOUNE L, AISSANI D. Simulation evaluation of a relative frequency metric for web cache replacement policies [J]. Evolving Systems, 2018(3):245-254.

[14] BRESLAU L, CAO P, FAN L, et al. Web caching and Zipf-like distributions: evidence and implications [C]// Proceeding of the 1999 18th Annual Joint Conference of the IEEE Computer and Communications Societies, Las Vegas: IEEE, 1999:126-134.

[15] 刘磊, 熊小鹏. 最小驻留价值缓存替换算法[J]. 计算机应用, 2013, 33(4):1018-1022.

[16] 吴俊龙, 杨清. 基于协同过滤的缓存替换算法研究[J]. 计算机工程与科学, 2015, 37(11):2128-2133.

[17] 周扬发. Web代理服务器的缓存技术研究[D]. 北京:北京邮电大学, 2013.

[18] ZIPF G K. Homogeneity and heterogeneity in language [J]. Psychological Record, 1938, 2(14):347-367.

本文编辑:陈小平