基于 SVM 访问预测机制的 Web 缓存数据库级替换策略

杨瑞君[1] 祝 可[1] 程 燕[2]
(上海应用技术大学计算机科学与信息工程学院 上海 201418)[1]
(华东政法大学刑事司法学院 上海 201620)[2]

摘 要 Web 缓存用于解决网络访问延迟和网络拥塞问题,缓存替换策略直接影响缓存的命中率。为此,文中提出一种基于访问预测机制的 Web 缓存替换策略。首先,根据用户之前的访问日志,通过预处理操作提取多项特征以构建特征数据集。然后,通过训练支持向量机(SVM)分类器来预测缓存对象是否可能被再次访问,将分类为不会再次被访问的缓存对象删除以腾出空间。仿真结果表明,与传统的 LRU,LFU 和 GDSF 方案相比,提出的策略具有较高的请求命中率和字节命中率。

关键词 Web 缓存,替换策略,访问预测机制,支持向量机
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.06.030

Database-level Web Cache Replacement Strategy Based on SVM Access Prediction Mechanism

YANG Rui-jun[1] ZHU Ke[1] CHENG Yan[2]
(School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 201418, China)[1]
(School of Criminal Justice, East China University of Political Science and Law, Shanghai 201620, China)[2]

Abstract Web cache is used to solve the problems of network access delay and network congestion, and cache replacement strategy directly affects the hit rate of cache. For this reason, this paper proposed a database-level Web cache replacement strategy based on SVM access prediction mechanism. Firstly, according to previous access logs of users, a feature data set is constructed on the basis of extracting multiple features through a pre-processing operation. Then, a Support Vector Machine (SVM) classifier is trained to predict whether a cached object is likely to be accessed again in the future, and the cached objects that are classified as not being accessed are deleted to free memory. Simulation results show that, compared with the traditional LRU, LFU and GDSF schemes, this strategy has higher request hit rate and byte hit rate.

Keywords Web cache, Replacement strategy, Access prediction mechanism, Support vector machine

1 引言

Web 缓存是提高 Web 服务质量的一种重要技术,用于减少客户端延迟、网络流量和服务器负载[1]。Web 缓存体系结构如图 1 所示,部署在客户端附近的代理缓存为本地 Web 网页提供服务[2]。

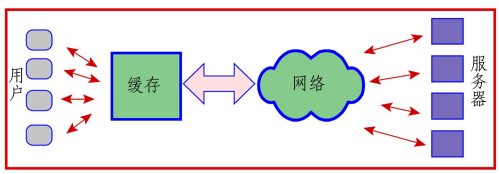


图 1 Web 缓存体系结构

Fig. 1 Architecture of Web cache

高效的代理缓存服务器为大部分请求提供本地缓存资源命中,这样可以大大提高客户端的下载速度并减少流量。Web 缓存的性能取决于其架构、缓存策略、替换策略和一致性处理[3]。其中,缓存策略是当一个对象自原始服务器取回时,决定其是否被允许进入缓存存储。替换策略是当缓存存储满时,决定是否将某个对象从缓存存储中替换或逐出。由于缓存的内存有限,因此将所有对象保存在内存中非常困难[4]。必须使用缓存替换算法,从缓存中驱逐对象并为新的 Web 对象创建一个空间来存储。

学者们对很多流量特征(如自相似性、Web 对象和网站的流行性、Web 的类型和性质等)进行了广泛的研究,并利用这些特性来优化和改进 Web 高速缓存替换方法,以解决高速缓存一致性问题。文献[5]对缓存替换方案进行了广泛的调查,其结论是缓存方案主要基于 4 个日志因素(特征),即频率(过去对象的引用次数)、新近度(从最后一次引用对象起所经过的时间)、大小(对象的大小)和成本(从原始服务器获取对象的成本)。

目前,缓存替换方法主要分为两类:一类是基于特征统计的方法,一类是基于智能预测算法的方法。基于特征统计的

到稿日期:2018-05-28 返修日期:2018-09-02 本文受国家自然科学基金(631233211)资助。
杨瑞君(1977—),男,博士,副教授,主要研究方向为无线传感器网络、网络安全、机器嗅觉,E-mail:yangruijun@sit.edu.cn(通信作者);祝 可(1981—),男,硕士生,主要研究方向为机器嗅觉、无线传感器网;程 燕(1978—),女,博士,副教授,主要研究方向为网络安全。

方法有最近最少使用(Least Recently Used,LRU)、最少使用频次(Least Frequently Used,LFU)、最近最常使用(Most Recently Used,MRU)等算法[6]。LRU 算法会删除最近最少使用的对象,并将新对象填入空出的缓存块。LFU 用于删除最不频繁使用的对象,其必须使用一个计数器来计算对象的使用频率。MRU 算法从缓存中清除最近使用的文档。然而,这些方法都仅考虑用其中一个特征来替换缓存,准确性不高。为此,文献[7]介绍了一种被称为贪婪对偶大小次数(Greedy Dual Size Frequency,GDSF)的算法,其考虑了两个或更多特性的组合用于决策。另外,一些学者通过智能预测模型来预测对象未来的访问概率,以此设定替换策略。例如,文献[8]提出了一个二元逻辑回归(LR)模型来预测未来的访问模式,以此保留访问概率较大的缓存。LR 方法在计算开销方面相对较少,并且可根据其输出进行预测。还有一些学者提出了基于智能算法的预测替换方法。例如,文献[9]使用人工神经网络(ANN)提出了一种自适应方法来预测网页的未来访问。但是,这些智能算法如 ANN、遗传算法(GA)等本质上是复杂的,并且在做出缓存替换决定时的计算量很大。

鉴于此,本文采用支持向量机(Support Vector Machine,SVM)来学习历史访问数据,通过构建预测模型来预测缓存中对象是否可能被再次访问,从而做出替换决策。仿真结果证明了本文方法的有效性。

2 支持向量机分类器

机器学习是一种适应性机制,使计算机能够通过实例学习学到经验。支持向量机(SVM)、朴素贝叶斯(NB)和决策树(C4.5)是 3 种流行的监督学习算法。由于 SVM 被建模为一种二次规划问题,在训练中存在全局最优解。此外,SVM 训练最大化分类边际,因此泛化能力可以最大化,尤其是当训练数据稀缺且线性可分时。另外,SVM 对于异常值是鲁棒的,因为裕度参数控制了错误的影响[10]。为此,本文采用 SVM 来构建预测分类器。

SVM 的基本概念是使用高维空间来寻找一个线性边界或超平面,用两类正负样本进行二分类,SVM 对数据的分类如图 2 所示。SVM 尝试在两个不同的类之间放置超平面(实线),并将其定向为使边界(虚线)最大化。超平面被定向为使超平面与每个类中最近数据点之间的距离最大化。最近的数据点用于定义边界,并称为支持向量(SV)(灰色圆圈和灰色正方形)[11-12]。

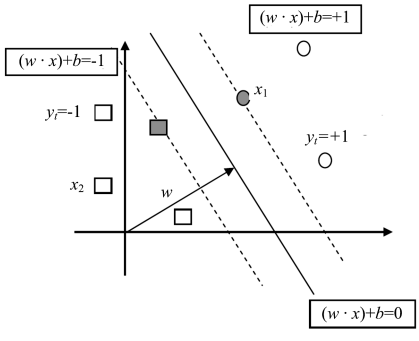


图 2 SVM 分类数据

Fig. 2 Classification data of SVM

超平面可以表示为:

$$(w \cdot x)+b=0, w\in R^N, b\in R \tag{1}$$

其中,向量 w 定义边界,x 是维度为 N 的输入向量,b 是尺度阈值。正类和负类的表达式如下:

$$(w \cdot x)+b=1; (w \cdot x)+b=-1 \tag{2}$$

SV 对应于给定类的数据的末端,因此,使用式(3)来分类正面或负面的任何数据点:

$$f(x)=sign((w \cdot x)+b) \tag{3}$$

最优超平面可以作为以下优化问题的解决方案:

$$\text{最小化}: t(w)=\frac{1}{2}\| w^2 \|$$

$$\text{约束}: y_i((w \cdot x)+b)\geqslant 1, i=1,\cdots,l$$

其中,l 是训练集的数量。根据式(4)可以得到约束优化问题的解:

$$w=\sum v_i x_i \tag{4}$$

其中,x_i 是从训练中获得的 SV,最终的决策函数表示为:

$$f(x)=sign(\sum_{i=1}^{l} v_i (x \cdot x_i)+b) \tag{5}$$

然而,对于许多现实问题,要找到超平面来分类数据(如非线性可分数据)并不容易。如果按照线性情况下的原理对非线性可分数据进行分类,输入数据只能从原始空间转换到被称为特征空间的更高维空间[13-14]。那么,超平面可以分离特征空间中的正面实例和负面实例,如图 3 所示。因此,决策函数变成:

$$f(x)=sign(\sum_{i=1}^{l} v_i (\emptyset(x) \cdot \emptyset(x_i))+b) \tag{6}$$

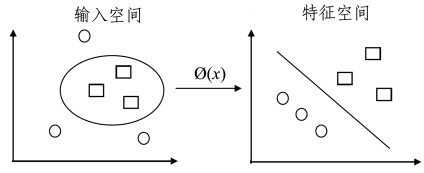


图 3 从输入空间到特征空间的转换

Fig. 3 Conversion from input space to feature space

从输入空间到特征空间的转换是相对计算密集的。为此,可以使用内核函数在单个步骤中执行此转换和点积,这有助于减少计算负担并同时保留较高维度变换的效果。核函数 $K(x_i, x_j)$ 被定义为:

$$K(x_i, x_j)=\emptyset(x_i) \cdot \emptyset(x_j) \tag{7}$$

将式(7)代入决策函数中,SVM 的基本形式变为:

$$f(x)=sign(\sum_{i=1}^{l} v_i K(x_i \cdot x_j)+b) \tag{8}$$

参数 v_i 被用作加权因子以确定哪个输入向量是支持向量。SVM 中可以使用几个核函数来解决不同的问题。本文将 RBF 核作为 SVM 训练的核函数,如式(9)所示:

$$K(x_i \cdot x_j)=\exp(-\gamma \| x_i-x_j \|^2), \gamma>0 \tag{9}$$

参数 γ 表示 RBF 的宽度。如果类与不可分数据之间存在重叠,则可以限制参数范围 v_i 以减少由 SV 定义的边界上的异常值带来的影响。对于不可分离的情况,约束条件变为 $0<v_i<C$。对于可分离的情况,C 是无穷大;对于不可分离的情况,C 可能会有所不同,具体取决于训练方案中允许的错误

数量。较高的 C 值允许少量错误，而较低的 C 值允许解决方案中有较高的错误比例。

3　基于 SVM 的缓存替换策略

3.1　方法框架

图 4 给出了提出的基于 SVM 的 Web 代理缓存替换方法框架。其由两个功能组件组成：离线组件和在线组件。离线组件负责训练机器学习分类器，而训练后的分类器被融入到智能替换方法中，用于有效管理在线组件中的 Web 代理缓存。

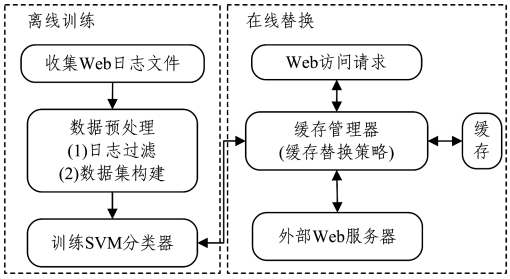


图 4　基于 SVM 的 Web 代理缓存替换方法框架

Fig. 4　Framework of Web cache replacement strategy based on SVM

在线组件中，当用户请求网页时，用户与代理进行通信，代理直接从代理缓存中检索请求的页面。如果请求的对象不在代理缓存中，代理服务器从原始服务器中请求对象并将其发送回客户端。同时，将所请求的对象的副本复制到代理缓存中，以减少未来请求中的响应时间和网络带宽利用率。在某些情况下，需要将新来的对象存储到代理缓存中，但代理缓存中的空间已满，此时代理缓存管理器使用 SVM 智能替换方法来删除不需要的 Web 对象，以便为新来的对象释放足够的空间。

另外，在系统首次启用时，Web 缓存为空。在之后启用时，Web 缓存中的初始对象为上次使用结束时保存的对象，在使用过程中通过缓存管理器实时替换。

在提出的智能缓存替换方法中，当用户请求 Web 对象 g 时，训练的 SVM 分类器预测该对象是否将被再次访问。如果将对象 g 分类为将再次访问的对象，则将对象 g 放置在缓存堆栈的顶部；否则将对象 g 放置在所使用的缓存堆栈的中间。因此，该方法可以在早期阶段有效地删除不需要的对象，从而为新的 Web 对象腾出空间。所提出的智能缓存替换算法的具体流程如算法 1 所示。

算法 1　智能缓存替换算法

```
Begin
For 用户请求的每个 Web 对象 g
  If g 在缓存中
    缓存命中，更新 g 的信息；
    通过训练好的 SVM 分类 g；
    If g 的类别=1 //表示会被再次访问
      将 g 移动到缓存堆栈的顶部；
    Else
      将 g 移动到所用缓存堆栈的中间位置；
  Else
    缓存未命中，从源服务器获取 g；
    While 缓存中没有足够空间用于 g
      删除位于缓存堆栈底部的对象 q；
    EndWhile
    通过训练好的 SVM 分类 g；
    If g 的类别=1
      将 g 移动到缓存堆栈的顶部；
    Else
      将 g 移动到所用缓存堆栈的中间位置；
  End if
End for
```

3.2　数据预处理及 SVM 训练

在代理服务器中，将多个 Web 服务器的用户组行为信息记录在代理日志中，用于训练机器学习分类器，使其能够有效预测下一个 Web 对象。

数据集需要通过预处理步骤将数据形式进行规范化。预处理分为两个步骤：日志过滤和训练数据集构建。在日志过滤中，不相关或无效的请求（诸如不可缓存的对象等）将从日志文件中删除。另一方面，训练数据集的构建是从日志代理文件中提取所需的信息，即使用能够表示用户兴趣的 Web 对象的重要特征来构建训练数据集。这些特征由 URL 对象中的 ID、时间戳、已用时间、大小和类型组成。随后，将这些特征转换为训练阶段所需的数据集模式。数据集模式的格式为 $\langle x_1, x_2, x_3, x_4, x_5, x_6, y\rangle$，$x_i$ 表示输入要素，y 表示请求对象的目标输出。其中，x_1 表示基于后向滑动窗口的 Web 对象访问的新近性；x_2 表示 Web 对象访问的频率；x_3 表示基于后向滑动窗口的 Web 对象访问的频率；x_4 表示 Web 对象的检索时间，以毫秒为单位；x_5 表示 Web 对象的大小，以字节为单位；x_6 表示 Web 对象的类型。

x_1 和 x_3 是基于向后滑动窗口提取的，其公式如式(10)和式(11)所示。其中，一个请求的后视和前视滑动窗口表示发出请求之前和之后的时间。

$$x_1=\begin{cases}\mathrm{Max}(SWL,\Delta T), & \text{if 对象 } g \text{ 在之前被请求}\\ SWL, & \text{otherwise}\end{cases} \tag{10}$$

$$x_{3_i}=\begin{cases}x_{3_{i-1}}+1, & \text{if } \Delta T\leqslant SWL\\ 1, & \text{otherwise}\end{cases} \tag{11}$$

其中，ΔT 是从对象 g 上次被请求以来的时间，SWL 是滑动窗口长度。如果第一次请求对象 g，则 x_3 将被设置为 1。

在 SVM 训练中，可以使用类似 polynomial，sigmoid 和 RBF 的几个核函数。本文考虑使用 RBF 核函数，因为它是最常用的核函数，并且与其他核函数相比，该核函数在许多应用中可以获得更好的性能。使用交叉验证找到最佳参数 C（边缘柔软度）和 γ（RBF 宽度），使用最佳参数来训练和测试整体数据集。训练之后，所获得的 SVM 使用式(8)来预测 Web 对象的类别，即对象是否会被再次访问。

4　性能评估

4.1　实验设置

仿真实验中的网络访问日志由 IRcache 代理缓存服务器收集。处理这些日志以生成模拟运行的数据集。将本文方案与 LRU，LFU 和 GDSF 的性能进行比较。与 LRU 和 LFU 进

行比较，可以评估该方案捕获网络流量的频率和新近程度的能力。GDSF 算法被认为是基线算法中最好的衍生算法之一。

使用 WebTraf 软件进行缓存仿真。仿真设置如表 1 所列。Web 服务器的数量取决于日志文件的唯一 IP 地址或主机名的数量。大型日志文件的 Zipf 参数使用 zipfR 软件包进行测量[15]。由于 Zipf 法则对高速缓存性能的影响并不是本文的研究内容，因此本文不会针对每个工作负载(100～40000)进行衡量。缓存大小设置为从 1kB 到 256MB 不等。

表 1 仿真参数

Table 1 Simulation parameters

参数	取值范围
Web 服务器数量	100～1000
请求数量	5000～40000
输入日志的 Zipf 参数	0.72～0.84
唯一请求的数量	1500～6000

4.2 性能指标

缓存替换策略的性能有几个关键指标，基于这个性能指标，可以比较不同算法的性能。为了评估缓存的性能，最常用的性能指标是对象命中率(Hit Ratio，HR)和字节命中率(Byte Hit Ratio，BHR)。HR 表示所有请求对象中直接在缓存中能找到的百分比。BHR 表示所有请求数据中直接从缓存中能找到的百分比。

$$HR=\frac{\sum_{i\in R}h_i}{\sum_{i\in R}f_i} \tag{12}$$

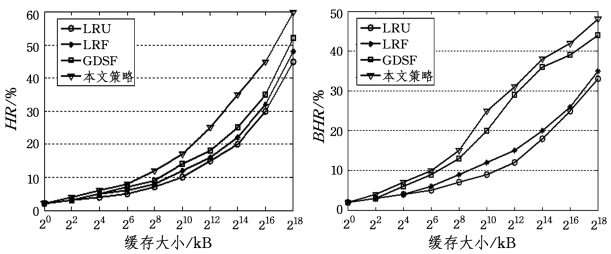
$$BHR=\frac{\sum_{i\in R}s_i\cdot h_i}{\sum_{i\in R}s_i\cdot f_i} \tag{13}$$

其中，s_i 为对象 i 的尺寸，f_i 为对象 i 的请求总数，h_i 为命中的对象 i 的请求数量，R 为被访问的对象集合。

4.3 性能比较

使用 IRcache 生成不同的工作负载，对各种替换策略的 HR 指标和 BHR 指标进行评估。构建两个具有一致性的数据集 sj.ircache.net 和 bo2.ircache.net 用于缓存仿真，它们分别有 23000 和 20000 条日志记录。对于 *HR* 和 *BHR* 的测量，设置替换机制阈值(*HM*)固定为总缓存大小的 90%。当该磁盘使用量达到上限 *HM* 值时，关闭准入控制机制，触发替换机制。

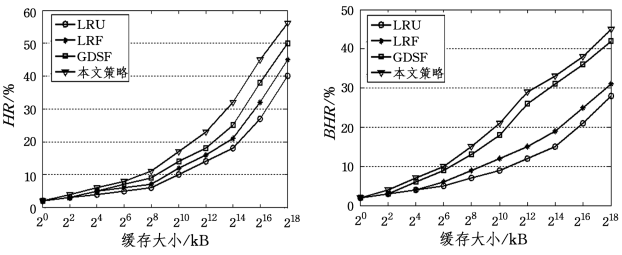
图 5 和图 6 分别显示了 sj 和 bo2 数据集上，缓存大小为 1kB 到 25MB 时，各种替换策略下的 *HR* 值和 *BHR* 值。表 2 统计了两个数据集上，缓存大小为 256MB 时的性能数据。



(a)*HR* 值比较 (b)*BHR* 值比较

图 5 sj 数据集上的 *HR* 值和 *BHR* 值

Fig. 5 *HR* and *BHR* values on sj dataset



(a)*HR* 值比较 (b)*BHR* 值比较

图 6 bo2 数据集上的 *HR* 值和 *BHR* 值

Fig. 6 *HR* and *BHR* values on bo2 dataset

表 2 缓存大小为 256MB 时的性能数据

Table 2 Performance data when cache size is 256MB

(单位：%)

替换策略	sj 数据集		bo 数据集	
	HR	*BHR*	*HR*	*BHR*
LRU	45.5	33.2	40.3	28.4
LFU	48.3	35.	45.1	31.2
GDSF	52.5	44.2	50.5	42.6
本文策略	60.2	48.1	56.3	45.2

从两个数据集上的实验结果可以看出，随着缓存大小的增加，命中率也增加，这是因为缓存较大时，能够保存的对象更多。通过性能比较可以看到，LRU 和 LFU 策略的命中率都很低，这是因为其仅仅采用了一种特征来决定是否替换，不能很好地反映对象的价值。GDSF 策略的性能相对较好，这是因为其使用多个特征参数形成目标函数，改进了 LRU 和 LFU 的不足。然而，本文策略都获得了最高的 *HR* 值和 *BHR* 值。这证明本文采用的 SVM 智能学习分类算法能够准确预测未来可能被访问的缓存对象，有效提高缓存的命中率。

4.4 算法复杂度分析

LRU，LFU 和 GDSF 方法都是基于统计的方法，不需要进行提前训练，且计算复杂度不高。而本文采用的 SVM 智能方法由于需要训练学习，因此在实现上要比 LRU，LFU 和 GDSF 方法多一个步骤。SVM 训练阶段的复杂度为 $O(n,v^2)$，其中 n 是训练样本的数量，v 是数据中的属性数量。

本文方法在训练阶段总共消耗了约 280s，但该过程只需要执行一次。使用训练好的 SVM 进行在线缓存对象分类时效率很高。对于每个缓存对象的分类过程，LRU 方法需要 0.12s，LFU 方法需要 0.13s，GDSF 方法需要 0.18s，而本文 SVM 需要大约 0.21s，可以满足缓存管理的实时要求。可以看到，本文方法在有效提高缓存命中率的同时，并没有明显增加缓存管理时间。

结束语 本文提出了一种基于 SVM 访问预测机制的 Web 缓存替换策略。该策略根据用户之前访问的日志构建了包含多项特征的数据集，用于训练 SVM 分类器，以此来预测缓存对象是否可能被再次访问，并做出替换决策。仿真结果表明，与传统的 LRU，LFU 和 GDSF 方案相比，提出的替换策略具有最高的命中率，能够缓解网络访问延迟和网络拥塞问题。

参 考 文 献

[1] WANG G Q，HUANG T，LIU J，et al. A New Cache Policy

Based on Sojourn Time in Content-Centric Networking[J]. Chinese Journal of Computers，2015，38(3)：472-482. (in Chinese)
王国卿，黄韬，刘江，等. 一种基于逗留时间的新型内容中心网络缓存策略[J]. 计算机学报，2015，38(3)：472-482.

[2] ZHAO Z Q，LIU D. Web Proxy Server Cache Optimization Based on Tree Extended Naive Bayes Classifier[J]. Computer Engineering，2017，43(1)：115-119. (in Chinese)
赵中全，刘丹. 基于树扩展朴素贝叶斯分类器的 Web 代理服务器缓存优化[J]. 计算机工程，2017，43(1)：115-119.

[3] HAO Y，MA T，SHEN W，et al. An Improved Web Cache Replacement Algorithm Based on Weighting and Cost[J]. IEEE Access，2018，25(6)：1352-1360.

[4] WANG Y G，LI Z Y，WU Q H，et al. Performance Analysis and Optimization of Cache Replacement Algorithm in Information Center Network[J]. Journal of Computer Research and Development，2015，52(9)：2046-2055. (in Chinese)
王永功，李振宇，武庆华，等. 信息中心网络内缓存替换算法性能分析与优化[J]. 计算机研究与发展，2015，52(9)：2046-2055.

[5] SHEU J P，CHUO Y C. Wildcard Rules Caching and Cache Replacement Algorithms in Software-Defined Networking [J]. IEEE Transactions on Network & Service Management，2016，13(1)：19-29.

[6] ZHANG J. Replacement Strategy of Web Cache Based on Data Mining[C]// International Conference on P2p，Parallel，Grid，Cloud and Internet Computing. IEEE，2015：821-823.

[7] MA T，QU J，SHEN W，et al. Weighted Greedy Dual Size Frequency based Caching Replacement Algorithm[J]. IEEE Access，2018，25(6)：7214-7223.

[8] SAJEEV G P，SEBASTIAN M P. Comparing the Performance of Multinomial Logistic Regression and Neural Network Models in Web Cache Content Classification[C]// International Conference on Machine Learning and Computing. 2011：53-58.

[9] WANG Z，HE Y L. Cloud storage cache replacement scheme based on hybrid value calculation[J]. Computer Engineering and Design，2017，38(6)：1651-1656. (in Chinese)
王准，何元烈. 基于混合价值计算的云存储缓存替换方案[J]. 计算机工程与设计，2017，38(6)：1651-1656.

[10] CHENG G，CHEN Y X. Identification method of encrypted traffic based on support vector machine[J]. Journal of Southeast University(Natural Science Edition)，2017，47(4)：655-659. (in Chinese)
程光，陈玉祥. 基于支持向量机的加密流量识别方法[J]. 东南大学学报(自然科学版)，2017，47(4)：655-659.

[11] ALI F，KHAN P，RIAZ K，et al. A Fuzzy Ontology and SVM-based Web Content Classification System[J]. IEEE Access，2017，24(5)：25781-25797.

[12] ERGUL E，ARICA N，AHUJA N，et al. Clustering Through Hybrid Network Architecture With Support Vectors[J]. IEEE Transactions on Neural Networks & Learning Systems，2016，28(6)：1373-1385.

[13] AIMTONGKHAM P，SO-IN C，SANGUANPONG S. A novel web caching scheme using hybrid least frequently used and support vector machine[C]// International Joint Conference on Computer Science and Software Engineering. IEEE，2016：1-6.

[14] WU X，XU H，ZHU X，et al. Web Cache Replacement Strategy Based on Reference Degree[C]// IEEE International Conference on Smart City/socialcom/sustaincom. IEEE，2015：209-212.

[15] EVERT S，BARONI M. zipfR：word frequency distributions in R[C]// Meeting of the ACL on Interactive Poster and Demonstration Sessions. Association for Computational Linguistics，2007：29-32.