Wireless Pers Commun (2017) 92:925–940
DOI 10.1007/s11277-016-3585-1

Improving Web Performance in Home Broadband Access Networks

Yantao Li[1,2] · Gang Zhou[2] · Bin Nie[2]

Published online: 10 August 2016

Abstract Web caching is a significantly important strategy for improving Web performance. In this paper, we design *SmartCache*, a router-based system of Web page load time reduction in home broadband access networks, which is composed of cache, SVM trainer and classifier, and browser extension. More specifically, the browser interacts with users to collect their experience satisfaction and prepare training dataset for SVM trainer. With the desired features extracted from training dataset, the SVM classifier predicts the classes of the Web objects. Then, integrated with LFU, the cache makes a cache replacement based on SVM-LFU policy. Finally, by implementing *SmartCache* on a Netgear router and Chrome browsers, we evaluate our SVM-LFU algorithm in terms of Web page load time, SVM accuracy, and cache performance, and the experimental results illustrate that *SmartCache* can greatly improve Web performance in Web page load time.

Keywords Web caching · Support vector machine · Least-Frequently-Used · Cache replacement · Home broadband networks

1 Introduction

Nowadays, people largely rely on Internet through browsers, which provides latest news around the world, knowledge of diverse areas, and various kinds of entertainments to ease our modern life. To provide better Web performance, researchers measure some popular Web sites and demonstrate that latency is the main bottleneck for Web page load time [1]. Therefore, Internet service providers and application providers are increasingly cognizant of the importance of reducing Web page load time, even small differences in latency can introduce significant effects on usability. Nevertheless, more and more people prefer

✉ Yantao Li
yantaoli@foxmail.com; liyantao@live.com; yantaoli@swu.edu.cn

[1] College of Computer and Information Sciences, Southwest University, Chongqing 400715, China

[2] Department of Computer Science, College of William and Mary, Williamsburg, VA 23185, USA

surfing Internet at home, which requires higher Web performance of broadband access networks. For most normal users at home, several seconds in latency should be acceptable, sometimes even not perceivable, however, their satisfaction significantly drops if they have been kept waiting for longer than a certain time of tens of seconds. Imaging that consumers are browsing an online shopping site and it takes forever to display a product information, most of them will simply give up and leave a bad impression towards this site. Moreover, latency is more rigorous for international corporations, such as Google, Bing and Amazon. Long latency considerably degrades usability, and as a consequence, it directly leads to huge loss in their revenues [2, 3]. Nevertheless, Web proxy caching plays a key role in improving Web performance by keeping Web objects in a proxy cache close to users, which contributes to reduce user perceived latency, improve network bandwidth utilization, and alleviate loads from origin servers. Therefore, how to design Web proxy caching policies of reducing latency on routers that improve Web performance is becoming a hotspot of research area.

There are a wide range of related works concentrated on Web proxy caching policies, which can be categorized into conventional caching strategies and intelligent caching strategies. For conventional caching strategies, Least-Recent-Used (LRU) [4], Least-Frequently-Used (LFU) [5], Most Recently Used (MRU) [6], SIZE [7], Greedy-Dual-Size (GDS) [8], Greedy-Dual-Size-Frequency (GDSF) [5], Hybrid [9], and Lowest Relative Value (LRV) [10] have been deeply studied. However, these conventional strategies only consider one factor in order to make caching decision and ignore the other factors that have impact on the efficiency of the Web proxy caching [4, 13]. For intelligent caching strategies, ICWCS [11, 12], NNPCR [13], NNPCR-2 [14], SVM-GDSF, C. 45-GDS, and SVM-LRU [15], and SACS [16] have been widely investigated. For instance, Ali et al. [15] propose new approaches that depend on the capability of SVM and C4.5 to learn from Web proxy logs files and predict the classes of objects to be re-visited or not. Negrao et al. [16] design semantics aware caching system that measures the distance between objects in terms of the number of links necessary to navigate from one object to another, where when replacement takes place, objects that are distant from the most recently accessed pages are candidates for removal and the closest an object is to a recently accessed page, the less likely it is to be evicted. However, these intelligent caching strategies basically focus on Internet, not on home broadband access networks, and ignore server locations that impact latency.

In this paper, we design *SmartCache*, a router-based system of Web page load time reduction for improving Web performance in home broadband access networks, which can learn users' experience satisfaction and collect desired features of Web objects, and then utilize these information to make cache replacements. *SmartCache* is composed of components of cache, SVM trainer and classifier, and browser extension. More specifically, the browser extension component interacts with users to collect their experience satisfaction and prepare training dataset for SVM trainer component. With the features extracted from training dataset, the SVM classifier component predicts the classes of the Web objects either the objects will be revisited or not. Then, the cache component makes a cache replacement, based on the proposed Web cache SVM-LFU algorithm in cache supported by the browser extension and SVM classification. Finally, we implement *SmartCache* on a Netgear router and Chrome browsers, and evaluate our SVM-LFU caching policy in terms of Web page load time, SVM accuracy, and cache performance. The experimental results illustrate that *SmartCache* can significantly improve Web performance in Web page load time.

The main contributions of this work can be summarized as:

- We design a router-based *SmartCache* for improving Web performance in home broadband access networks, which is composed of cache, SVM trainer and classifier, and browser extension.
- We integrate intelligent technique SVM with conventional policy LFU to provide SVM-LFU caching strategy to extract desired features from training dataset in browser and make cache replacements in cache.
- We implement *SmartCache* on a Netgear router and Chrome browsers, and evaluate our SVM-LFU caching policy in terms of Web page load time, SVM accuracy, and cache performance. The experimental results illustrate that *SmartCache* can significantly improve Web performance in Web page load time.

The remainder of this paper is organized as follows: Sect. 2 presents some background knowledge on Web proxy caching and related work, and Sect. 3 proposes the motivation of our work. In Sect. 4, we elaborate the design details of *SmartCache* composed of cache, SVM trainer and classifier, and browser extension. We evaluate the performance of *SmartCache* in terms of Web page load time, SVM accuracy, and cache in Sect. 5 and conclude this work in Sect. 6.

2 Background and Related Work

We begin this section by providing some background on Web proxy caching. Subsequently, we will look at related work that presents caching strategies in conventional and intelligent aspects, respectively.

2.1 Web Proxy Caching

Web caching is one of the most effective and important approaches for improving the performance of web-based systems [15], which decreases user perceived latency, reduces network bandwidth usage, and reduces page load time from the origin servers. Typically, a Web caching can be located in a browser, a proxy server or an origin server. More specifically, the browser cache is located on a browser in the client machine, which stores previous responses from Web servers and reduces the amount of information that needs to be transmitted across the network, as information previously stored in the cache can often be reused. At the origin server, Web pages can be stored in a server-side cache for reducing the redundant computations and the server load. The proxy cache is located between the client machines and an origin server, such as a router, which works on the same principle as the browser cache, but in a much larger scale. Unlike the browser cache which deals with only a single user, the proxy server serves hundreds or thousands of users in the same way. When a request of Web object is received, the proxy server checks its cache: if the object is available, the proxy server sends the object to the client; if not, or expired, the proxy server will request the object from the origin server and send it to the client, and the requested object will be stored in the proxy's local cache for future requests [15, 17].

Web proxy caching has been widely utilized by computer network administrators, technology providers, and businesses to reduce both user delays and Internet congestion [18–20]. As Web proxy cache size is limited, a cache replacement policy is needed to handle the cache content. If the cache is full when an object needs to be stored, the replacement policy will determine which object is supposed to be evicted to allow space for the new object. The optimal replacement policy aims to make the best use of available

cache space, improve cache hit rates, and reduce loads on the origin server. The cache replacement policy plays an extremely important role in Web proxy caching. Hence, the design of efficient cache replacement algorithms is required to achieve highly sophisticated caching mechanism [21–23].

There are a few factors (features) of Web objects that influence the Web caching [21, 24]:

- Recency: objects last reference time
- Frequency: number of requests made to an object
- Size: size of the requested Web object
- Access latency of Web object

The performance of Web caching methods are normally analyzed using the metrics hit ratio (HR) and byte hit ratio (BHR). HR is referred to as the percentage of the number of requests that are served by the cache over the total number of requests, while BHR is defined as the percentage of the number of bytes that correspond to the requests served by the cache over the total number of bytes requested. A high HR indicates the availability of the requested object in the cache most of the time, while a high BHR indicates reduced user-perceived latency and savings in bandwidth [13, 22].

2.2 Related Work

We bifurcate related work based on the type of caching strategies—those that have dealt with Web performance in conventional caching strategies, and those that have coped with improving Web performance in intelligent caching strategies.

2.2.1 Conventional Caching Strategies

Some conventional caching strategies have been deeply studied, such as Least-Recent-Used (LRU), Least-Frequently-Used (LFU), Most Recently Used (MRU), SIZE, Greedy-Dual-Size (GDS), Greedy-Dual-Size-Frequency (GDSF), Hybrid, and Lowest Relative Value (LRV). The authors in [4] propose the simplest and most common cache management approach, LRU algorithm, which removes the least recently accessed objects until there is sufficient space for the new objects. LRU is easy to implement, efficient for uniform size objects such as the memory cache, and used in other areas of caching, however, it does not perform well in Web caching since it does not consider the size or the download latency of objects. In [5], the authors present another common policy of web caching that replaces the objects with the smallest number of accesses, LFU algorithm, which keeps more popular Web objects and evicts rarely used ones. However, LFU suffers from the cache pollution in objects with the large reference accounts, which are never replaced even if they are not re-accessed again, especially if these objects are large [4, 25–27]. MRU discards, in contrast to LRU, the most recently used web objects first [6]. The authors of [7] provide SIZE policy, which replaces the largest objects from a cache when space is needed for a new object, however, a cache can be polluted with small objects which will not be accessed again. To alleviate the cache pollution, the authors in [8] suggest GDS policy as an extension of the SIZE policy, which integrates several factors and assigns a key value or priority for each Web object stored in the cache. When cache space becomes occupied and new object is required to be stored in cache, the object with the lowest key value will be removed. Since GDS ignores the frequency of the Web object, the

authors of [5] present GDSF by integrating the frequency factor into the key value to overcome the drawback of GDS, however, it does not take into account the predicted accesses in the future. In [28], the authors compare these conventional caching strategies and point out their advantages and disadvantages. The authors of [9] propose Hybrid cache replacement algorithm that makes use of a combination of multiple requirements such as maintaining in the cache documents from servers that take significant time to connect to, those that need to be fetched from the slowest links, those that have been accessed very frequently, and those that are small. In [10], the authors provide LRV that expels the object that has the lowest utility value, where the utility of a document is calculated adaptively on the basis of data readily available to a proxy server. However, these conventional strategies are suitable for traditional caching like CPU caches and virtual memory system, not efficient in Web caching area. The reason is that they only consider one factor in order to make caching decision and ignore the other factors that have impact on the efficiency of the Web proxy caching [4, 13].

2.2.2 Intelligent Caching Strategies

Conventional caching strategies are simple and effective, but they cannot dynamically adapt to environment changes, which is of great importance for web caching. Therefore, some intelligent caching strategies have been proposed recently. In [11], the authors present an intelligent client-side Web caching scheme based on least recently used algorithm and neuro-fuzzy system, which divides client-side cache into short-term and long-term caches, where the Web objects requested in the first time are stored in short-term cache, while the Web objects visited more than once are moved to long-term cache. The authors of [13] provide the neural network proxy cache replacement (NNPCR) which utilizes neural networks for replacement decisions, and in [14], the authors propose an improved strategy of NNPCR referred to as NNPCR-2. However, the performance of back-propagation neural network (BPNN) in NNPCR or NNPCR-2 is influenced by the optimal selection of the network topology and its parameters that are based on trial and error. In [29], the authors propose an integrated solution of BPNN as caching decision policy and LRU technique as replacement policy for script data object. Since the most important factor in Web caching such as recency is ignored in caching decision, the authors of [30] enhance the policy using particle swarm optimization for improving neural network performance. In [15], the authors present SVM-GDSF, C. 45-GDS, and SVM-LRU three intelligent caching strategies with the help of machine learning techniques and decision tree, which predict the probability of the re-visit of a web object and add more weight to the object with higher re-visit probability. The authors of [17] propose intelligent naive Bayes-based approaches for Web proxy caching. In [31], the authors present a pre-fetching technique that uses clustering combined with SVM-LRU algorithm, a machine learning method for Web proxy caching. The authors of [6] present new approaches that depend on the capability of Decision Tree (DT) classifier to learn from Web proxy logs files and predict the classes of objects to be re-visited or not. In [16], the authors design semantics aware caching system (SACS) that measures the distance between objects in terms of the number of links necessary to navigate from one object to another, where when replacement takes place, objects that are distant from the most recently accessed pages are candidates for removal and the closest an object is to a recently accessed page, the less likely it is to be evicted.

3 Motivation

Based on an in-depth analysis of the issue arose from a recent literature [1], we find that non-US users normally suffer from longer page load time than US users, and the relative improvement in delay for their *HomeCaching* for non-US users is less. Moreover, Aptimize, a web content optimization organization, providing softwares that increase Web site performance by speeding up Web sites and intranets, further confirms the fact that the average first view load time for US users is 7.07 s while for non-US users is 9.46 s with a survey among Fortune 500 companies [32]. Motivated by these observations, we come up with an intuitive idea of assigning different weights for domestic and international Web sites, in order to achieve a shorter overall page load time.

As an example, we measure the page load time through a Web performance experiment between US users and non-US users. We explore WebPagetest, an open source project primarily developed and supported by Google [33], to test the first view page load time and repeat view page load time from 4 different locations of US, India, Australia and UK, on nine popular Web sites of CNN, Wikipedia, Facebook, Yahoo, Google, Amazon, Ebay, MSN and YouTube with the browser of Chrome. Note that the nine Web sites tested in this paper are the same as the ones in [1]. In the experiment, the first view page load time is measured on the browser of Chrome with an empty cache and without cookies, while the repeat view is conducted immediately after the first view experiment, which significantly represents the impact of caching. We plot the average page load time of first view and repeat view for the nine Web sites in the 4 locations in Figs. 1 and 2, respectively. As illustrated in Figs. 1 and 2, to sum up, the page load time for non-US (India, Australia and UK) users is much longer than that for US users. More specifically, the average page load time of the first view for US and for non-US users is 4.40 and 6.09 s, respectively, as demonstrated in Fig. 1, while for the repeat view, the average page load time is improved to 2.37 s for US users and to 3.27 s for non-US users, respectively, as shown in Fig. 2.

Furthermore, we introduce a metric of relative improvement to measure the page load time improvement between the first view and repeat view, defined in Eq. 1, where PLT_{first} denotes the page load time of the first view, and PLT_{repeat} indicates that of the repeat view. Based on the above experimental data, we plot the relative improvement of page load times for nine Web sites tested in 4 locations, in Fig. 3. As illustrated in Fig. 3, the relative

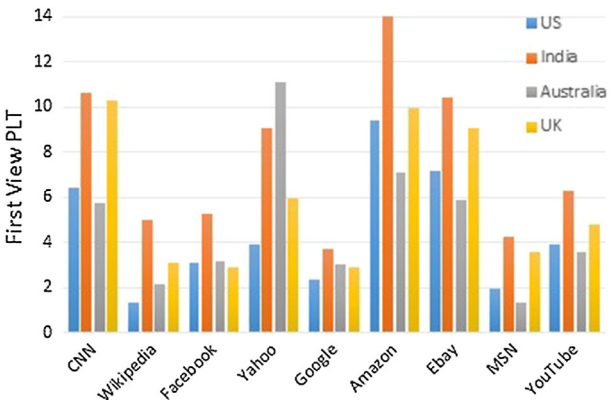

Fig. 1 First view page load time (PLT) for 9 web sites tested in 4 locations

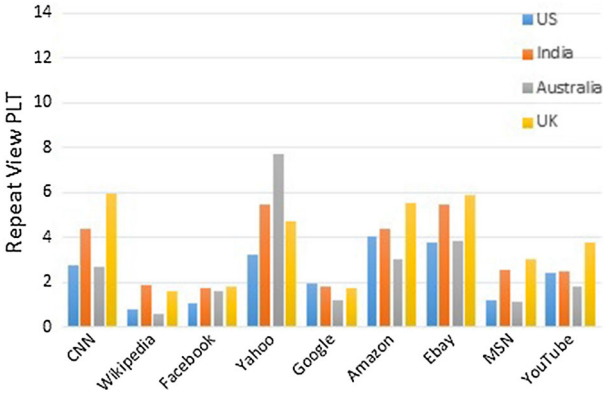

Fig. 2 Repeat view page load time (PLT) for 9 web sites tested in 4 locations

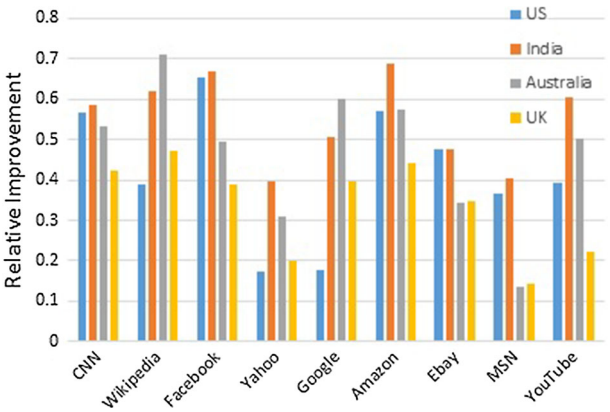

Fig. 3 Relative improvement of page load time (PLT) for 9 web sites tested in 4 locations

improvement of page load times for US users and non-US users are 41.80 and 45.09 %, respectively. As we can see, the difference between relative improvements for US and non-US users is small, which indicates that location information alone might not be sufficient to significantly improve the web caching performance.

$$\text{Relative Improvement} = \frac{\text{PLT}_{\text{first}} - \text{PLT}_{\text{repeat}}}{\text{PLT}_{\text{first}}} \quad (1)$$

In addition, Web users normally have different viewing behaviors or habits [34]. For instance, US residents seldom surf international Web sites while people traveling abroad need to view a lot of non-US Web sites from their home countries. Even for a single user, his or her viewing behavior might constantly vary with time. Therefore, user personal viewing habits probably impact the improvement of web performance.

From the above experiments and analysis, we observe that the reason why existing approaches have difficulty in improving web performance is probably due to lack of detailed knowledge of user location information and user viewing habits. Therefore, it is imperative to design a system architecture that targets to individual users and can dynamically adapt to viewing habit varies for improving Web performance in broadband access networks.

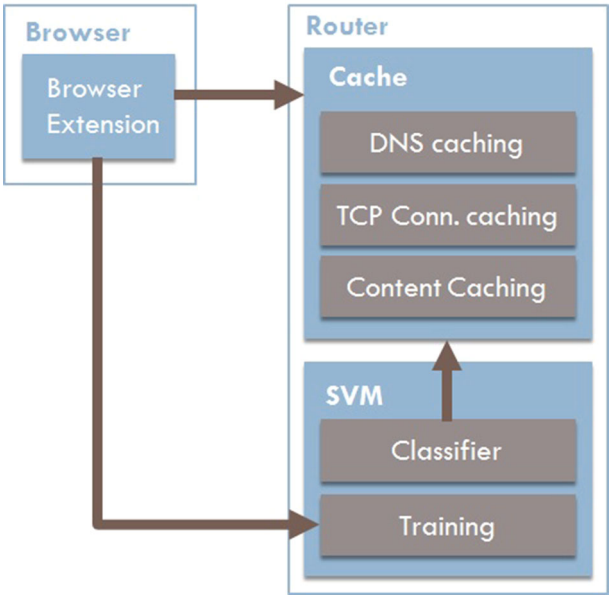

Fig. 4 *SmartCache* system architecture

4 *SmartCache* Design

We design *SmartCache*, a router-based system of Web page load time reduction for improving Web performance in home broadband access networks (Note that we consider a specific issue on Web performance in home broadband access networks in this paper). *SmartCache* can learn users' experience satisfaction and collect desired features of Web objects, and then utilize these information to make cache replacements, which consists of components of cache, SVM trainer and classifier, and browser extension, as illustrated in Fig. 4. Note that our *SmartCache* system resides solely on a router and on a browser with no reliance on a backed server, and the user communicates with it directly for retrieving that page from cache or from server. More specifically, the browser interacts with users to collect their experience satisfaction (such as access latency satisfaction) and prepare training dataset for SVM trainer component. With the common features (such as the first byte time and size of an object) extracted from training data, the SVM classifier component makes a decision that the object will be revisited or not. Then, the cache component makes cache replacements, based on the proposed Web cache SVM-LFU algorithm in cache supported by the browser extension and SVM classification. In this section, we give a detailed description of *SmartCache* on browser extension, SVM trainer and classifier, and cache, respectively.

4.1 Browser Extension

We develop an extension using the Web technology JavaScript for the Chrome browser, which can interact with each individual user to collect the personalized acceptable latency and prepare training dataset with features of URL ID, first view PLT, repeat view PLT, first byte time, size, and location of a Web object for SVM trainer component. When a user

Table 1 Acceptable latency range for common situations [35]

Situation	Acceptable latency range (ms)
Regular web sites	100–800
Heavy web sites	50–400
Web-based remote systems	30–300
Casual online game	200–1000
Action games	10–150
Stock exchange	5–100
Remote administration: Linux	50–500
Remote administration: Windows	50–250

requests a Web page, the user communicates with the browser directly for retrieving that page from a cache or from a server.

The browser extension supports two modes: training mode and normal mode. In training mode, the user is required to give feedback to the extension by indicating whether the current PLT is acceptable, when browsing a Web page, and meanwhile, the corresponding PLT with the other features as training dataset are saved in extension. If the user fails to do that, the extension will treat the current PLT based on acceptable latency range for common situations in Table 1. The user's satisfaction will be sent to cache component immediately and all the collected personalized data will be transmitted to SVM component for training. Since frequent interactions will affect user's experience while browsing, the extension switches to normal node once the training is completed. In this mode, the user can surf the Internet without any interference by the extension. Nevertheless, the users' viewing habits may vary and they probably become unsatisfied with current performance of *SmartCache*. The *SmartCache* provides a re-training mechanism by switching back to training mode through feedback.

4.2 SVM Trainer and Classifier

We choose Support Vector Machine (SVM) combined with RBF kernel to differentiate Web objects. SVM is one of the most robust and accurate methods in machine learning algorithms, which has been successfully applied in many real-world applications, such as Web page applications [15], activity and text classifications [36, 37], and bioinformatics applications.

The training data for SVM are collected by the Browser extension for over one month and are converted to the training pattern in the format of $< s_1, s_2, \ldots, s_6, v >$, where $s_1, \ldots, s_6$ indicate URL ID, first view PLT, repeat view PLT, first byte time, size, and location of a Web object, respectively, and v represents the target output of the object request which can be Class 0 or Class 1. In SVM training, SVM is trained as follows: prepare and normalize the dataset, consider the RBF kernel, use cross-validation to find the best parameters C (margin softness) and γ (RBF width), and use the best parameters to train the whole training dataset [38]. In SVM classification, we simply choose first byte time, size and location as the inputs of the SVM classifier. As a result, all the Web objects are classified into two categories: $W(g) = 1$ and $W(g) = 0$, where $W(g) = 1$ indicates the load time is not acceptable by users, and the Web object is desired to be cached, $W(g) = 0$ represents the load time is acceptable, and g denotes a Web object.

4.3 Cache

We choose Least-Frequently-Used (LFU) caching policy, which is the most common cache management approach, and which is able to remove the least frequently used objects until there is sufficient space for the new objects. In *SmartCache*, cache component resides on a router and contains three sub-caches of DNS lookup, TCP connection and content [1]. For the caching strategy, we integrate intelligent technique SVM with conventional policy LFU to provide SVM-LFU caching strategy to extract desired features from training dataset in browser and make cache replacements in cache, as illustrated in Algorithm 1. SVM-LFU caching strategy gives a key value for each object in the cache buffer. The key value $K(g)$ is calculated by a ratio of frequencies of Web object g over the maximum frequency of all Web objects plus user's satisfaction $W(g)$, as illustrated in Eq. 2. When a user requests a Web page, the object with the lowest $K(g)$ value will be removed first and the corresponding new object will be added in cache.

Algorithm 1: The Web cache SVM-LFU algorithm

```
1  Initialize W(g) = 0; //"W(g)" denotes the satisfaction of users to PLT, where
     "W(g) = 1" indicates that PLT is unacceptable and "W(g) = 0" indicates
     acceptable.
2  for Each Web object g requested by user do
3      if g in cache then
4          Cache hit occurs;
5          Update information of g; //Classify g by SVM classifier
6          W(g) = SVM(TimeofFristByte(g), Size(g)); //"TimeofFristByte(g)"
             denotes the time when the first byte of g is downloaded, and "Size(g)"
             indicates the size of g.
7          K(g) = Frequency(g) / Max(Frequencies(AllofWebSites)) + W(g) //"Frequency(g)"
             indicates the frequency of Web object g visited,
             "Frequencies(AllofWebSites)" denotes frequencies of all Web objects
             visited, "Max()" calculates the maximum frequencies of all Web objects
             visited.
8      end
9      else
10         Cache miss occurs;
11         while no enough space in cache buffer for g do
12             W(g) = min(K(Q)); //"min()" indicates the minimum value of K(Q),
                 and "Q" denotes the Web object sets in cache
13             if K(q) ≤ min(K(Q)) then
14                 Evict q and insert g into Q; //Web object "q" has the minimum
                     value in K(Q)
15             end
16         end
17         Fetch g into cache from origin server;
18     end
19 end
```

$$K(g) = \frac{\text{Frequency (g)}}{\text{Max(Frequencies(All of, WebSites))}} + W(g) \tag{2}$$

where Frequency(g) indicates the frequency of Web object g visited, Frequencies(All of WebSites) denotes frequencies of all Web objects visited, Max() calculates the maximum frequencies of all Web objects visited, and $W(g)$ is a binary value (0 or 1) indicated by the SVM classifier.

5 Performance Evaluation

We evaluate our *SmartCache* with real implementations on a router and browsers. In this section, we first describe the experiment setup, and then evaluate the *SmartCache* in terms of Web page load time, SVM accuracy, and cache performance.

5.1 Experiment Setup

In the experiments, we implement the *SmartCache* on a Netgear router and Chrome browsers, where the end hosts (such as desktops, laptops, tablets) with Chrome browsers are connected to the router. We collect data with features of the first view page load time, repeat view page load time, first byte time, size and location of the Web site server from IP tracker [39], by users browsing the top 100 most visited Web sites in 2014 [40]. Note that some Web sites are not available, we can only collect data for 96 web sites (80 inside US and 16 outside US) [41, 42].

5.2 Web Page Load Time

In this section, we compare the first view page load time and repeat view page load time for both US and non-US Web sites. With the dataset of users browsing the the top 100 most visited Web sites in 2014, we record the page load time and calculate the average time as shown in Fig. 5. As illustrated in Fig. 5, the average page load time of Web sites inside US is still shorter than that of Web sites outside US. We also compare the relative improvement by Eq. 1 between the average first view page load time and average repeat view page load time. As demonstrated in Fig. 6, the average improvements for US users and non-US users are 54.38 and 54.80 %, respectively. The difference is smaller, which further confirms that location information alone will not lead to significantly improvement on caching performance.

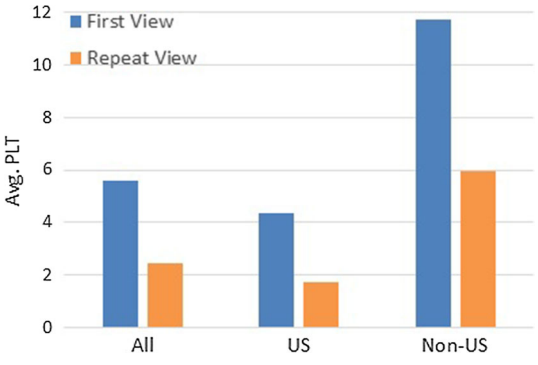

Fig. 5 Average page load time for most popular web sites in 2014

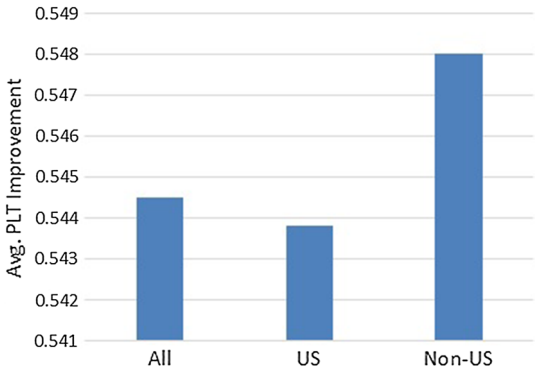

Fig. 6 Average relative improvement of page load time for most popular web sites in 2014

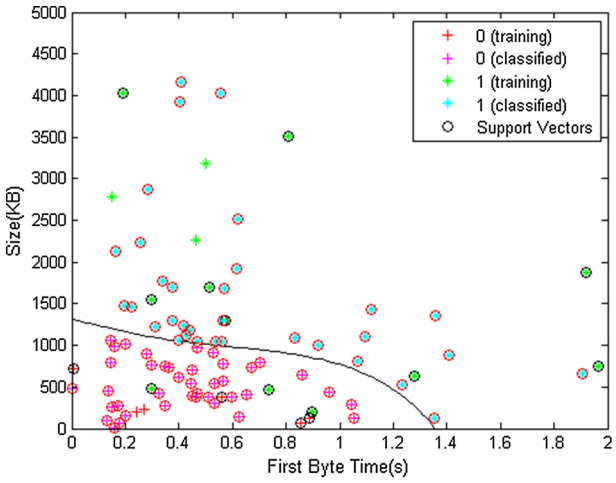

Fig. 7 SVM classification result

5.3 SVM Accuracy

For each Web site, users are required to assign satisfaction values as: 1 if the first view page load time exceeds 3 s; otherwise, 0. In order to train the SVM classifier, we randomly select 10 Web sites from US and non-US Web sites. Therefore, the training data size is 20, and the rest 76 web sites are used to test the performance of our SVM classifier. For the SVM classifier, we input the first byte times and page sizes of Web objects, and obtain values: 1 if the user is not acceptable for the page load time and 0 if acceptable. The classification results are illustrated in Fig. 7, and the accuracy of the SVM classifier is 76.32 %. Note that the SVM classifier is able to categorize the most testing Web sites except 3 extreme Web sites for a clear view of most sites.

5.4 Caching Performance

In this section, we evaluate the cache performance in terms of comparison with other cache strategies and relative improvements.

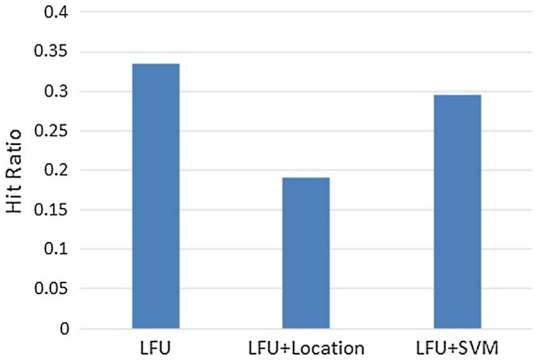

Fig. 8 Hit ratios for different caching strategies

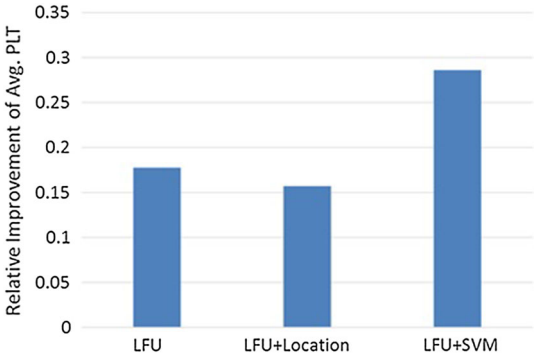

Fig. 9 Relative improvement of average page load time after using different caching strategies

The cache size is set to be 20. As we designed in Algorithm 1, when a cache hit occurs, we utilize the repeat view page load time and when a cache miss occurs, we use first view page load time. We compare our SVM-LFU replacement decision strategy with simple LFU and LFU-Location, respectively. As for LFU-Location replacement decision strategy, we use Eq. 3 to compute a key value $K(g)$ for each Web site g, where $L(g) = 1$ for non-US Web sites and $L(g) = 0$ for US ones. When the cache is full, the object with the smallest $K(g)$ will be replaced.

$$K(g) = \frac{\text{Frequency(g)}}{\text{Max(Frequencies(All of WebSites))}} + L(g) \tag{3}$$

We plot the cache hit ratios for the three strategies in Fig. 8. As illustrated in Fig. 8, LFU has the highest hit ratio, while LFU-Location has the least. Although LFU has the highest hit ratio, it suffers from the cache pollution in objects with the large reference accounts [4, 5].

We further compare the relative improvements (compared to no cache) in terms of the average page load time. As demonstrated in Fig. 9, SVM-LFU enjoys the highest improvement, which shows significantly higher than the second one, LFU; LFU-Location has the least improvement, which gains almost only half of the improvement achieved by SVM-LFU.

The above two comparisons further confirm that *SmartCache* is very effective to consider personalized user acceptable latency, and even simple design will lead to significant performance augment.

6 Conclusion

In this paper, we design *SmartCache*, a router-based system of Web page load time reduction for improving Web performance in home broadband access networks, which can learn users' experience satisfaction and collect features of Web objects, and then utilize these information to make cache replacements. *SmartCache* is composed of cache, SVM trainer and classifier, and browser extension. We implement *SmartCache* on a Netgear router and Chrome browser, and evaluate our SVM-LFU caching policy in terms of Web page load time, SVM accuracy, and cache performance. The experimental evaluation results illustrate that *SmartCache* can significantly improve Web performance in Web page load time.

In the future, we would like to consider more factors to achieve better performance. For example, when training the SVM classifier, we take the type of web sites as an extra input, since users expect different page load time for different types of Web sites. As a result, we can obtain more accurate results from SVM classifier.

Acknowledgments The work is supported in part by the National Natural Science Foundation of China (Grant nos. 61402380 and 61528206), the Natural Science Foundation of CQ CSTC (Grant no. cstc2015jcyjA40044), U.S. National Science Foundation (Grant nos. CNS-1253506 (CAREER) and CNS-1250180), the Fundamental Research Funds for the Central Universities (Grant no. XDJK2015B030), and the Opening Project of State Key Laboratory for Novel Software Technology (Grant no. KFKT2016B13).

References

1. Sundaresan, S., & Feamster, N. (2013). Measuring and mitigating web performance bottlenecks in broadband access networks. In *Proceedings of Internet Measurement Conference*, Barcelona, Spain.
2. Brutlag, J. (2009). Speed matters for google web search http://services.google.com/fh/files/blogs/google_delayexp
3. Lohr, S. (2012). For impatient web users, an eye blink is just too long to wait, 2012 http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html?pagewanted=all&_r=0.
4. Koskela, T., Heikkonen, J., & Kaski, K. (2003). Web cache optimization with nonlinear model using object features. *Computer Networks*, *43*, 805–817.
5. Cherkasova, L. (1998). Improving www proxies performance with greedy-dual-size-frequency caching policy. In *Hp Technical Report*. Palo Alto.
6. Kumar, P. N. V., & Reddy, V. R. (2014). Web proxy cache replacement policies using decision tree (DT) machine learning technique for enhanced performance of web proxy. *International Journal on Recent and Innovation Trends in Computing and Communication*, *2*(2), 302–309.
7. Abrams, M., Standridge, C. R., Abdulla, G., Fox, E. A., & Williams, S. (1996). Removal policies in network caches for World-Wide Web documents. In *Proceedings on Applications, Technologies, Architectures, and Protocols for Computer Communications* (pp. 293–305). Palo Alto: CA.
8. Cao, P., & Irani, S. (1997) Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1997 Usenix Symposium on Internet Technology and Systems*. Monterey, CA.
9. Wooster, R. P., & Abrams, M. (1997). Proxy caching that estimates page load delays. *Computer Networks and Isdn Systems*, *29*(8–13), 977–986.
10. Rizzo, L., & Vicisano, L. (2000). Replacement policies for a proxy cache. *IEEE/ACM Transactions on Networking*, *8*(2), 158–170.
11. Ali, W., & Shamsuddin, S. (2009). Intelligent client-side web caching scheme based on least recently used algorithm and neuro-fuzzy system. In W. Yu, H. He, & N. Zhang (Eds.), *Advances in Neural Networks-ISNN 2009* (pp. 70–79). Berlin: Springer.
12. Ali, W., & Shamsuddin, S. M. (2011). Neuro-fuzzy system in partitioned client-side Web cache. *Expert Systems with Applications*, *38*, 14715–14725.
13. Cobb, J., & ElAarag, H. (2008). Web proxy cache replacement scheme based on back-propagation neural network. *Journal of Systems and Software*, *81*, 1539–1558.

14. Romano, S., & ElAarag, H. (2011). A neural network proxy cache replacement strategy and its implementation in the Squid proxy server. *Neural Computing and Applications*, *20*, 59–78.
15. Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2012). Intelligent web proxy caching approaches based on machine learning techniques. *Decision Support Systems*, *3*, 565–579.
16. Negrao, A. P., Roque, C., Ferreira, P., & Veiga, L. (2015). An adaptive semantics-aware replacement algorithm for web caching. *Journal of Internet Services and Applications*, *6*, 1–14.
17. Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2012). Intelligent naive Bayes-based approaches for Web proxy caching. *Knowledge-Based Systems*, *31*, 162–175.
18. Kaya, C. C., Zhang, G., Tan, Y., & Mookerjee, V. S. (2009). An admission-control technique for delay reduction in proxy caching. *Decision Support Systems*, *46*, 594–603.
19. Kumar, C., & Norris, J. B. (2008). A new approach for a proxy-level web caching mechanism. *Decision Support Systems*, *46*, 52–60.
20. Kumar, C. (2009). Performance evaluation for implementations of a network of proxy caches. *Decision Support Systems*, *46*, 492–500.
21. Chen, H. T. (2008). *Pre-Fetching and Re-Fetching in Web Caching Systems: Algorithms and Simulation*. Peterborough, Ontario, Canada: Trent University.
22. Kin-Yeung, W. (2006). Web cache replacement policies: A pragmatic approach. *IEEE Network*, *20*, 28–34.
23. Vakali, A. (2002). Evolutionary techniques for Web caching. *Distributed and Parallel Databases*, *11*, 93–116.
24. Chen, R.-C., & Hsieh, C.-H. (2006). Web page classification based on a support vector machine using a weighted vote schema. *Expert Systems with Applications*, *31*, 427–435.
25. Wikipedia (2014). Cache pollution http://en.wikipedia.org/wiki/Cachepollution
26. Conti, M., Gasti, P., & Teoli, M. (2013). A lightweight mechanism for detection of cache pollution attacks in Named Data Networking. *Computer Networks*, *57*, 3178–3191.
27. Deng, L., Gao, Y., Chen, Y., & Kuzmanovic, A. (2008). Pollution attacks and defenses for internet caching systems. *Computer Networks, 52*(5), 935–956.
28. Ali, W., Shamsuddin, S. M., & Ismail, A. S. (2011). A survey of web caching and prefetching. *International Journal of Advances in Soft Computing and its Applications*, *3*, 1–26.
29. Farhan, (2007). *Intelligent Web Caching Architecture, Faculty of Computer Science and Information System*. Johor, Malaysia: UTM University.
30. Sulaiman, S., Shamsuddin, S. M., Forkan, F., & Abraham, A. (2008) Intelligent Web caching using neurocomputing and particle swarmoptimization algorithm, modeling and simulation, 2008. In *AICMS 08 Second Asia International Conference on, 2008* (pp. 642–647).
31. Baskaran, K. R., & Kalaiarasan, C. (2014). Pre-eminence of combined Web pre-fetching and Web caching based on machine learning technique. *Arabian Journal for Science and Engineering*, *39*, 7895–7906.
32. Aptimize (2010). Website performance benchmarks, 2010. http://www.aptimize.com/Upload/docs/2010-Website-Performance-Benchmarks.
33. WebPagetest. http://www.webpagetest.org/.
34. Hablinger, G., & Hartleb, F. (2011). Content delivery and caching from a network providers perspective. *Computer Networks*, *55*(18), 3991–4006.
35. Cloud, C. (2012). Some interesting bits about latency. https://www.citycloud.com/city-cloud/some-interesting-bits-about-latency/.
36. Qi, X., Keally, M., Zhou, G., Li, Y., & Ren, Z. (2013). AdaSense: Adapting Sampling Rates for Activity Recognition in Body Sensor Networks. In *Proceedings of the 19th IEEE Real-Time and Embedded Technology and Applications Symposium* (pp. 163–172). Philadelphia, PA.
37. Qi, X., Zhou, G., Li, Y., & Peng, G. (2012). Radio Sense: Exploiting wireless communication patterns for body sensor network activity recognition. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium* (pp. 95–104). San Juan, Puerto Rico.
38. Hsu, C. W., Chang, C. C., & Lin, C. J. (2003). A practical guide to spport vector classification. http://www.csie.ntu.edu.tw/∼cjlin/papers/guide/guide.pdf.
39. IP Tracker. http://www.ip-tracker.org/domain-to-location.php.
40. Afrodigit.com, Top 100 most visited websites in the world 2014. http://afrodigit.com/visited-websites-world/.
41. Marshall, A. D., & Hurley, S. (1996). The design, development and evaluation of hypermedia courseware for the World Wide Web. *Multimedia Tools and Applications*, *3*, 5–31.
42. Huber, J., & Ding, Y. (2013). Adapting web pages using graph partitioning algorithms for user-centric multi-device web browsing. *Multimedia Tools and Applications*, *62*, 209–231.

Yantao Li is an Associate Professor in the College of Computer and Information Sciences at Southwest University, China, and a Postdoctoral Research Associate in the Department of Computer Science at the College of William and Mary, USA. He received the PhD degree from the College of Computer Science at Chongqing University, in December 2012. From September 2010 to September 2012, he was a visiting scholar supported by China Scholarship Council working with professor Gang Zhou at the Department of Computer Science in the College of William and Mary, USA. His research area includes wireless communication and networking, sensor networks and ubiquitous computing, and information security.

Gang Zhou is an Associate Professor in the Department of Computer Science at the College of William and Mary. He received his Ph.D. degree from the University of Virginia in 2007 under Professor John A. Stankovic. He has published more than 60 papers in the areas of sensor networks, ubiquitous computing, mobile computing and wireless networks. The total citations of his papers are more than 4300 according to Google Scholar, among which the MobiSys'04 paper has been cited more than 780 times. He also has 13 papers each of which has been cited more than 100 times since 2004. He is an Editor of IEEE Internet of Things Journal and also an Editor of Elsevier Computer Networks Journal. Dr. Zhou served as NSF, NIH, and GENI proposal review panelists multiple times. He also received an award for his outstanding service to the IEEE Instrumentation and Measurement Society in 2008. He is a recipient of the Best Paper Award of IEEE ICNP 2010. He received NSF CAREER Award in 2013. He is a Senior Member of IEEE and a Senior Member of ACM.

Bin Nie is working toward the PhD degree in the Department of Computer Science at the College of William and Mary. She received her B.S. degree from Xiamen University, China, in 2008 and M.S. degree from Fordham University, USA, in 2014. Her research area includes performance evaluation, mobile computing and wireless networks.