Poster: A Framework for Instant Mobile Web Browsing with Smart Prefetching and Caching

Huan Huang, Hailong Sun, Guoqing Ma, Xu Wang, Xudong Liu
School of Computer Science and Engineering, Beihang University
Beijing, China 100191
{huangh,sunhl,magq,wangxu,liuxd}@act.buaa.edu.cn

ABSTRACT

Mobile users often suffer from a slow page loading time due to intermittently connected wireless networks and increasing size of mobile Web pages. Although existing approaches of prefetching and caching are widely used to reduce the browsing latency, they may fail to work effectively because most Web page visits are singletons and the cache hit ratio is unsatisfying. In this work, we design and implement a framework to reduce Web browsing latency for mobile users with a smart prefetching strategy and caching mechanism. The prefetching strategy leverages the skSLRU model, which predicts and prefetches Web pages based on their contents with consideration of user contexts and the devices' status such as power consuming and cellular data usage. And our caching mechanism mainly consider the resources like CSS and JavaScript files shared among Web pages in a website. Moreover, instead of using RAM, we use ROM, the internal flash memory of devices to store cached resources with proper lifecycle so as to avoid the useful resources to be untimely evicted or expired and thus improve the hit ratio. Our evaluations show that 90% of the homepages and 60% of other pages are fetched before users' visiting, and the page loading time is no more than one second.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems[Distributed Application, Client/Server]

Keywords

Mobile Web browsing; Prefetching; Caching

1. INTRODUCTION

Web browser is one of the most important applications on mobile devices, but it turns out to be slow in the page loading process [5] due to mobile wireless networks with relatively low bandwidth and intermittent connectivity. Furthermore, currently the size of mobile Web pages is getting larger because of the increasing usage of component resources like images, JavaScript and CSS. Therefore end-users suffer a 7-second loading time for mobile Web pages averagely [1], which leads to unsatisfying browsing experience.

In order to reduce page loading time, prefetching [2, 3] and caching [4] have been widely used to avoid real-time resource loading from network. Unfortunately, they may not work effectively for mobile Web browsing. For *prefetching*, most existing methods are based on the historical URL records but 75% of the Web page visits are singletons [6]; As for *caching*, only 30% of the requested resources can be fetched from local storage directly even with unlimited size of cache [6], and current caching mechanisms usually invalidate cached resources very soon for data freshness and evict them quickly as the cache size is limited. However, we observe that (1) most Web browsing falls into specific sections of a few websites and (2) the Web pages in the same website usually share a large portion of page component resources and these resources may still be used in the following Web page visits, which is also pointed out in [3, 6].

Basing on these observations, we design and implement a framework for instant mobile Web browsing with smart prefetching and caching. First, our prefetching strategy is able to predict and prefetch Web pages by computing the relevance between the contents of the new Web pages with the visited pages of each user. Moreover, this strategy takes the limited battery, system resources and network connections of mobile devices into consideration in determining when and how many resources to prefetch. Second, our caching mechanism avoids the useful resources to be untimely evicted or expired by localizing them in the ROM rather than RAM with a suitable lifecycle. We also provide interfaces for users and developers to achieve their personalized caching mechanisms. Finally, the experimental results show that, with our framework, more than 90% of the homepages and 60% of other Web pages can be prefetched before users' visiting, and the time to render out them is below one second.

2. FRAMEWORK OVERVIEW

2.1 Architecture of the Framework

Fig. 1 depicts the architecture of our framework, which consists of a Web browser and a server for accelerating computing of prefetching. Our prefetching strategy relies on the cooperation of both the statistics in the browser and the prefetching model on the server side. And the caching mechanism is implemented in the Web browser.

MobiCom'14, September 7-11, 2014, Maui, Hawaii, USA.
ACM 978-1-4503-2783-1/14/09.
http://dx.doi.org/10.1145/2639108.2642899 .

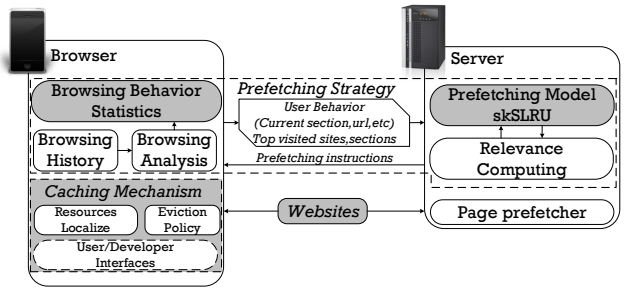

Figure 1: Architecture of the framework

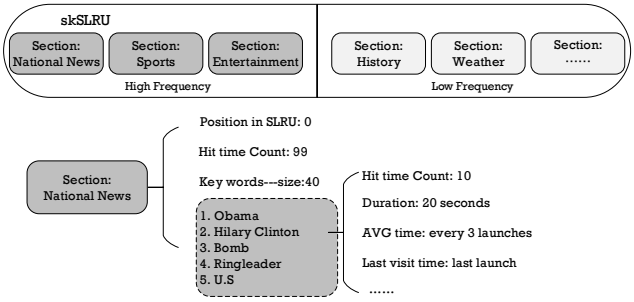

Figure 2: The skSLRU model

2.2 Prefetching Strategy

The prefetching strategy solves the problems of what to prefetch, when to prefetch and how many resources to fetch with corresponding solutions. In terms of what to prefetch, the solution is a combination of the local browsing behavior statistics and a prefetching model on the server, as shown in Fig. 1. As most websites are organized in a tree structure, starting from the homepage to a few sections like *sports* and *news*. Pages are updated all the time while the URL addresses of homepages and the index pages of sections remain unchanged. As a result, we compute the statistics in five time periods each day, i.e. 0:00-3:00, 6:00-10:00, 10:00-14:00, 14:00-18:00 and 18:00-0:00. And the visited websites in each period are ranked based on the visiting frequency. At the same time, via analysing the URL traces, we also get the top visited sections in each website. With the computed browser behavior statistics, the browser will inform the prefetching model on the server of which sites and sections are most likely to be visited at a specific time.

During the web browsing, the browser requests content prefetching instructions from the skSLRU (section and keyword-based SLRU). As depicted in Fig. 2, this model is built on the basis of SLRU and each rectangle represents a frequently visited section. Sections in the "High Frequency" part are visited more recently and frequently, and less in "Low Frequency" part. This model receives a user's browsing behaviors from the browser including the sites and sections that a user is visiting or is likely to visit. In the example shown in Fig. 2, keywords like "Obama" and "Bomb" appear repeatedly in the visited pages, so we assume a user is interested in pages containing these words or other semantically related words. As shown in Fig. 3, these keywords can help us find the most relevant pages in the same section through the executing process of skSLRU. As a result, the browser gets the prefetching instructions from our server about which pages to fetch.

As for when and how many page resources to prefetch, we perform prefetching in parallel with user browsing and fetch resources with the consideration of user contexts and devices' status such as remaining power and network conditions. Existing mobile browsers do not make use of the time when people are reading or searching for interesting pages. According to our traces, people usually stay in a Web page for 20 seconds averagely, which is long enough to prefetch pages possibly to be visited later. In addition, the number of pages to prefetch depends not only on the network condition (e.g. WIFI or 3G/4G), but also on the battery status. In this work, resources are fetched when a WIFI connection is available or when a device is charged for power.

2.3 Caching Mechanism

Caching may not work well for mobile devices. As people tend to use the applications randomly, the memory space may already be recycled by "Garbage Collection" before a future visit. Even if the requested resources are in cache, 60% of them still send requests to the server for data freshness [6], and these requests cause extra latency. More importantly, it is known that mobile users are more sensitive to long Web page loading time than to data freshness. Hence we build a new caching mechanism relying on the following factors:

- 76% of the resources in a Web page are shared by at least another page in the same site [6].
- Homepages and section pages do not change too much during a day.
- JavaScript/CSS files which belong to the website remain unchanged for a long period of time. For example, in our experimental results, most files are not updated even for two months.

In our caching mechanism, resources fetched each time are cached separately. For example, resources like JavaScript/CSS which are unlikely to change and shared by most pages in the same website are stored in the ROM with a long lifecycle. And images belong to the CSS files which we find to be relatively large are fetched in a WIFI network gradually to save the data usage. Localizing these files is important to eliminate latency. Because when the html is being parsed, the requests of JavaScript/CSS files block the following processes and the Web page cannot be rendered out quickly. Resources like jpeg/png/html have a shorter lifecycle. For

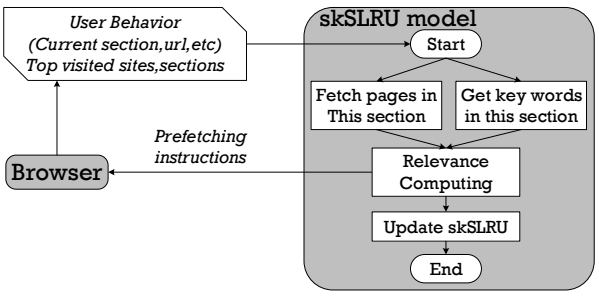

Figure 3: The executing process of skSLRU

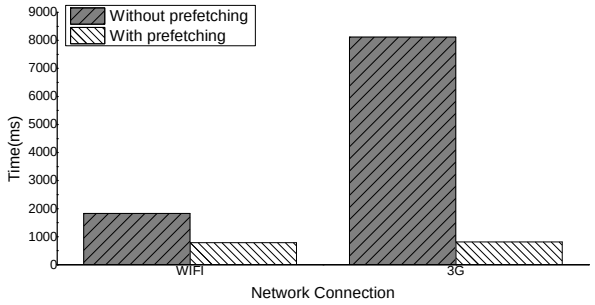

Figure 4: Processing time of rendering Web pages

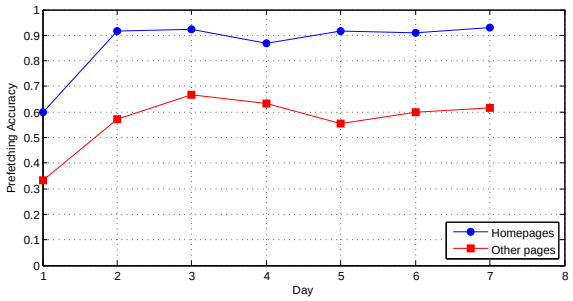

Figure 5: Accuracy of page prefetching during one week

instance, as most pictures related to news in the homepage stay the same through a day, these pictures will be evicted next day if they do not appear again. In fact, those pictures will hardly be visited later, thus evicting these useless images from memory is important. What's more, there are no revalidation requests for localized resources. As this caching mechanism bypasses the HTTP/1.1 protocol.

The mechanism also offers interfaces for website developers and mobile users. As most Web pages do not change too much, we allow users to set their own tolerance levels of Web page freshness for instant browsing. As for the developers, they can customize their own caching strategies to reduce latency on the browser side with a simple configuration file linked to their pages.

2.4 Implementation of the Framework

Our browser is developed based on Android WebView, but the framework we have implemented can be used not only for this browser, but also for a large portion of mobile applications which use WebView to display contents as well. The browser communicates with the skSLRU model on the server through a Web service.

3. EVALUATION RESULTS

We ask 15 students at Beihang University to use the implemented browser to browse Web contents with both WIFI or 3G Internet connections. And we collect the user browsing data for a month, starting from May 20 to June 20 in 2014. In this section, we present the obtained data to show the effectiveness of our framework. Fig. 4 shows the improvements of Web page loading time. It is satisfying to see the main contents of prefetched pages to be rendered out in less than one second. Users can use this framework whether online or offline. And even in the case of slow and unstable network conditions, our browser can load the requested Web pages in a surprisingly short period of time thanks to our smart prefetching strategy.

Fig. 5 depicts the prefetching accuracy of homepages and other pages with skSLRU, which shows 90% of the homepages and 60% of other pages are fetched before users' visiting. We also provide an interesting function in the prototype: the browser can highlight the prefetched pages' titles in the index pages. As a result, users can know which pages are already prefetched and stored locally, and they can read them online or offline. Users can also open the browser under a WIFI condition to get more offline Web contents they want to read.

4. CONCLUSIONS

Mobile users always expect instant Web browsing experience. In this work, we propose a framework for reducing Web page browsing latency, in which two techniques are involved. First, Web pages are prefetched based on the analysis of Web page content and user browsing behavior. Second, common resources of Web pages in a website are cached and reused in page loading. Besides, with the interfaces offered by this framework, the browsing experience can be personalized to better serve various mobile user requirements. A prototype of this framework has been implemented with Android WebView.

5. ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China (No.61370057), China 863 program (No. 2012AA011203), A Foundation for the Author of National Excellent Doctoral Dissertation of PR China (No.201159), Beijing Nova Program(No. 2011022) and the Program for New Century Excellent Talents in University.

6. REFERENCES

[1] Is the web getting faster? http://analytics.blogspot.com/2013/04/is-web-getting-faster.html.

[2] B. D. Higgins, J. Flinn, T. J. Giuli, B. Noble, C. Peplin, and D. Watson. Informed mobile prefetching. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 155–168. ACM, 2012.

[3] D. Lymberopoulos, O. Riva, K. Strauss, A. Mittal, and A. Ntoulas. Pocketweb: instant web browsing for mobile devices. In *ACM SIGARCH Computer Architecture News*, volume 40, pages 1–12. ACM, 2012.

[4] H. Wang, M. Liu, Y. Guo, and X. Chen. Similarity-based web browser optimization. In *Proceedings of the 23rd international conference on World wide web*, pages 575–584. International World Wide Web Conferences Steering Committee, 2014.

[5] Z. Wang, F. X. Lin, L. Zhong, and M. Chishtie. Why are web browsers slow on smartphones? In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*, pages 91–96. ACM, 2011.

[6] Z. Wang, F. X. Lin, L. Zhong, and M. Chishtie. How far can client-only solutions go for mobile browser speed? In *Proceedings of the 21st international conference on World Wide Web*, pages 31–40. ACM, 2012.