How Effective is Mobile Browser Cache?

[1]Zhen Wang, [2]Felix Xiaozhu Lin, [1,2]Lin Zhong and [3]Mansoor Chishtie

[1]Dept of Electrical & Computer Engineering and [2]Dept of Computer Science, Rice University, Houston, TX 77005
[3]Web Technologies, Texas Instruments, 12500 TI Boulevard, Dallas, TX 75243

ABSTRACT

We report a study on the effectiveness of the mobile browser's cache. The study is based on the browsing history from 24 smartphone users over one year. We make two interesting findings. Firstly, increasing the cache size of the browser on smartphones will not improve the effectiveness of the browser cache very much. Secondly, revalidations greatly reduce the effectiveness of the browser cache. The findings reveal the limitations of the cache design for mobile browsers and motivate a new level of cache design and speculative revalidation and loading.

Categories and Subject Descriptors

C.4 [Performance of systems]: Measurement techniques

General Terms

Human Factors, Measurement, Performance

Keywords

Smartphone, Browser, Cache

1. INTRODUCTION

Web browsers on smartphones are known to be slow because of slow resource loading [1]. A widely used solution to improve the resource loading part of opening a webpage is caching. Modern browsers cache the content of previously visited webpages and their subresources locally. Therefore, when a cached resource is requested, it is retrieved from the cache, resulting in a much shorter resource loading time and smaller browser delay. Apparently, a larger browser cache may help by saving more resources locally. There have been calls to improve the browser caching [2]. PC browsers' default cache size recently increases to around 300MB. This can be verified easily by checking the PC browser settings. The users can also set the cache size by themselves. However, mobile browser's cache size is still limited. Android Gingerbread browser's default cache size is only 6MB and the users cannot adjust the cache size by themselves. Such small cache size could easily result in many resource evictions in the cache and reduce the effectiveness of cache.

The effectiveness of browser cache, however, is limited by the fact that many resources are either not cacheable or already expired when the browser requests them. The non-cacheable and expired resources require revalidation from the server. If the resource is validated as unmodified, the server does not need to send the resource file to the client. In this process, non-cacheable and expired resources incur at least one round trip. Revalidation increases loading time from several milliseconds (local activity) to several hundred milliseconds (network activity), and sometime even several seconds. Since network round-trip time (RTT) is a key factor to mobile browser delay [1], revalidation will impact the mobile browser performance significantly.

In this work, we report a study on the effectiveness of the mobile browser cache. We find that increasing cache size will not improve the effectiveness of the mobile browsers cache very much. Even though a larger cache does help, increasing the cache size to infinite only reduces 10% of the cache misses. Secondly, revalidations account for 41% percent of the cache requests from the browser with infinite cache when visiting top10 most visited websites. This large amount of revalidations greatly reduces the effectiveness of the mobile browser cache.

The rest of the paper is organized as follows. Section 2 provides the methodology of our cache study. Section 3 presents the results of the cache study. The motivations of new design are discussed in Section 4. Section 5 concludes the paper.

2. METHODOLOGY

The effectiveness of browser cache is closely related to the user's browsing behavior. We utilize the browsing history of LiveLab traces [3] to examine the effectiveness of browser cache under various cache size settings. The traces contain the browsing history for 24 users over a period of one year. The study consists of three steps.

- *(i)* We download all the resources of the webpages with header information based on the LiveLab traces.
- *(ii)* We calculate each resource's file size, cacheability and expiration time based on the file and header information. The expiration time is calculated from the "Cache-Control" or "Expire" header fields. If those fields are not specified, the expiration time can be inferred from the "Last-Modified" fields [4].
- *(iii)* We simulate the cache behavior of the Android browser by going over each user's browsing history using the pre-calculated information of each resource. Four different cache sizes are used in the simulation: 6MB, 32MB, 64MB and infinite cache size. Current Android Gingerbread browser's default cache size is 6MB.

The download is not at the same time as the users visited the webpage. Some pages may be missing or changed. We use average webpage size to simulate those missing pages and their subresources. We also use the same trick for HTTPS webpages. The cache hits and misses for them and their subresources are not counted. HTTPS webpages are not counted because most of them require user login information. We cannot download their actual webpages and subresources. According to the protocol [4],

Table 1: Expiration and cache related header fields of the resources from top visited HTTPS websites (The login related subresources are not included which are usually not cacheable)

HTTPS Website	*Expire in*	*Subresources' Cache-Control field*	*Subresources' Expire field*
owlspace-ccm.rice.edu	55 days	None	None
Google (mail, doc, talk, etc)	1 year	max-age=31536000 (1 year)	Expire in a year
mobilebanking.chase.com	1 day	max-age=86400 (1 day)	None
wtime.teamhealth.com	110 days	None	None
webbranch.uwcu.org	1 day	max-age=86400 (1 day)	None
www.amazon.com	20 days	max-age=(around 20 year)	Expire in 2031
student.testmasters.com	4-180 days	None	None
www.coair.com	2 years	None	None
whentohelp.com	7 days	max-age=604800 (7 days)	None
www.wf.com	1 day	max-age=86400 (1 day)	None

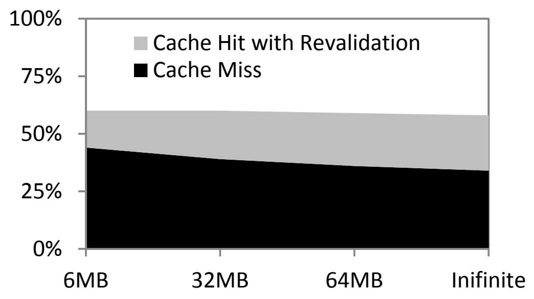


Figure 1: Cache Simulation Results for the webpages from all websites

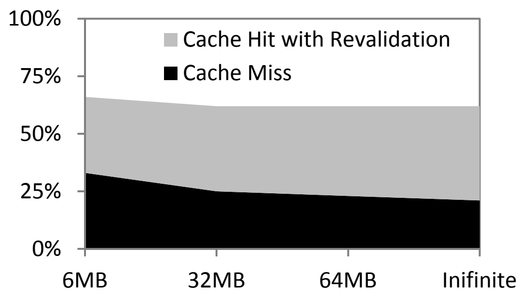


Figure 2: Cache Simulation Results for the webpages from top10 visited websites

HTTPS resources can be cached and will take room in the cache. Therefore, we should not ignore them in the simulation.

In the LiveLab traces, 32% of the webpage visits are not counted, 61% of which are HTTPS webpages. We manually checked the top visited HTTPS websites with our own login information. Table 1 shows the expiration time and cache related header fields for top 10 HTTPS websites' resources (the login scripts are not included, which are usually not cacheable). We can see that the financial related websites (chase, uwcu and wf) have much stricter cache constraints. Other websites either do not specify the "Cache-Control" and "Expire" fields, or the values for those fields are not different from their HTTP counterpart. Therefore, the expiration time of the resources from HTTPS pages are either the same with HTTP pages or shorter, which leads to higher revalidation rate.

3. RESULTS

The cache simulation results of cache misses and revalidations are shown in Figure 1 and Figure 2. Those numbers are calculated by the following equations. The subscript "t" stands for type and can be "all" for all webpages and "top10" for top 10 most visited webpages only

H = num of cache hits

R = num of cache hits which requires revalidation

M = num of cache misses

$\text{Cache Miss}_t = M_t / (H_t + M_t)$

$\text{Cache Miss \& Revalidation}_t = (M_t + R_t) / (H_t + M_t)$

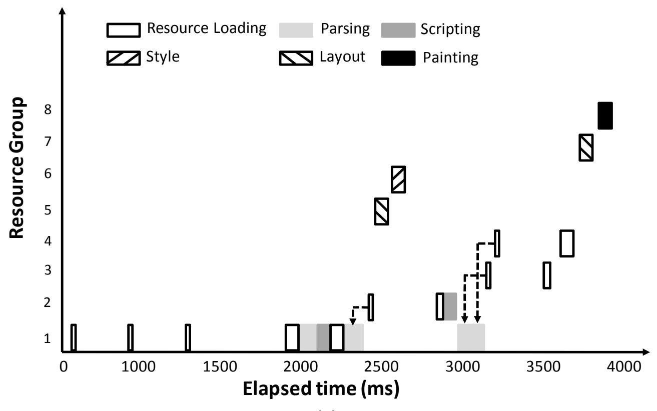
A request is labeled as cache hit if the resource is in the cache when the browser requests it and this number includes expired resources that require revalidation. A request is labeled as cache miss if the resource is not in the cache when the browser requests it. The reason for the cache miss is because that it is the first time to request of the resource (compulsory cache miss) or the cached copy of the resource is evicted (capacity cache miss). In the figure, it shows that the small size (6MB) of current browser cache incurs 10% more cache misses. When the cache size is infinite, all the cache misses are compulsory cache misses, which implies that the small cache size introduces 10% more capacity cache misses.

If considering the cache hits which require revalidation, the resource requires network activity for around 60% of the total requests. When visiting top10 most visited websites, cache hits with revalidation accounts for 41% of the total requests from the browser with infinite cache. Revalidation seems to introduce little overhead for PC browsers, which are often connected with fast Ethernet networks. It actually impacts the mobile browsers' performance significantly. As discussed by [1], network RTT is a key factor to mobile browsers' performance while bandwidth is not. Revalidation can reduce the bandwidth requirement, but do not help much in reducing the network round trips. Therefore, revalidations will introduce large overhead to mobile browsers.

The cache study results imply: (i) the improvement on the cache effectiveness from cache size increase is not very much, even though larger cache size helps to reduce the capacity cache misses; (ii) the revalidations greatly reduce the cache effectiveness, even with infinite cache.

4. MOTIVATIONS OF NEW DESIGN

From the cache study, we can see that cache size is not a big problem to mobile browsers. On one hand, enlarging the cache size of mobile browsers will not improve cache effectiveness much. On the other hand, the memory and storage technologies



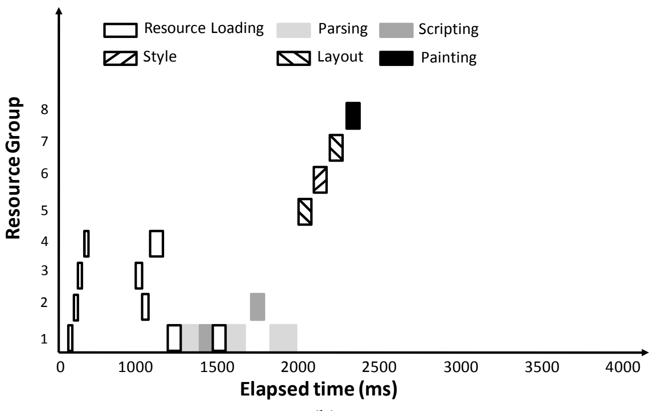


Figure 3: Dependency timeline graph [1] for (a) ordinary resource loading and (b) speculative resource revalidation and loading.

improve quickly and the problems of resource eviction and capacity cache misses can be easily solved by rapid hardware improvement. Why such small cache size is sufficient to satisfy the users' needs? The reason comes from the users' mobile browsing behavior: only a small number of websites are frequently visited by a user on the smartphone. The top visited website of a user accounts for 28% of web usage (median); and the user's top 10 visited websites account for 87% of his/her total webpage visits. Such small set of frequently visited websites indicates a large amount of revisits, which reduces the number of new resources that are needed to be cached. Therefore, the cache size is not the crucial factor for mobile browsers.

The large amount of revisits poses the importance of effective revalidation. From our cache study results, revalidation accounts for 41% of the total requests with infinite cache for top10 most visited websites. Revalidation reduces the bandwidth requirement, but cannot help much to reduce the round trips [4]. Since mobile browser's performance is sensitive to RTT [1], effective revalidation can help to improve mobile browser's performance.

The importance of effective revalidation motivates the design of a new level of caching and speculative revalidation and loading. Traditional caching caches the content of the resources locally and uses their URLs as the identifier. The new level of caching caches each webpage's URL and its subresources' URLs. We call the new level of caching *resource forest*. *Resource forest* contains many *resource trees*. Each webpage's URL is the root node of one tree and its subresources are the child nodes in the tree. The child node can also have its own children. For example, CSS or JS files can further request subresources. *Resource forest* has two advantages. Firstly, the size of a URL is very small. *Resource forest* will take very little space on the smartphone and adds little overhead to existing cache scheme. Secondly and more importantly, *resource forest* relates the resources in the cache. When visits occur, the browser knows which subresources are needed even before downloading and parsing the main HTML file. This removes the dependencies between the resources of a webpage during resource loading and makes the speculative loading and revalidation possible. In contrast, the resources in traditional caching are isolated and have no relation with other cached resources.

Speculative revalidation and loading utilize the multiple concurrent connections supported by modern smartphones [5]. For ordinary browsers, only one connection is active during the process of loading the main resource, as shown in Figure 3 (a). The reason is that the browser doesn't know what subresources are needed until downloading and parsing the main HTML file. It cannot send out requests for subresources with the main HTML file simultaneously. Therefore the benefit provided by concurrent connection is not utilized. Since *resource forest* provides the relation information of the cached resources to the browser, when revisits occur, the browser can download the main HTML file, revalidate the expired subresources and download the evicted or non-cacheable subresources simultaneously. Therefore, the resource loading part of the browser is more parallelized and the subresources can be pre-revalidated or pre-loaded, which saves the revalidation time required later. Figure 3 (b) illustrates the effect of the new design.

5. CONCLUSION

This paper studied the effectiveness of the mobile browser's cache. We found that increasing the cache size of the browser on smartphones will not improve the effectiveness of the browser cache much. And revalidations greatly reduce the effectiveness of the browser cache. Those two findings motivate the new level of cache design and speculative revalidation and loading. They provide the relation information of the cached resources and utilize the concurrent connections supported by modern mobile browsers to further parallelize the resource loading process. The revalidations are also more effective because of the pre-revalidations enabled by the new designs.

REFERENCES

[1] Z. Wang, X. Lin, L. Zhong, and M. Chishtie, "Why are web browsers slow on smartphones?," in *Proc. ACM Int. Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2011.
[2] S. Souders, "Call to improve browser caching," http://www.stevesouders.com/blog/2010/04/26/call-to-improve-browser-caching/.
[3] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum, "LiveLab: Measuring Wireless Networks and Smartphone Users in the Field," in *Proc. Workshop on Hot Topics in Measurement & Modeling of Computer Systems*, June 2010.
[4] "Hypertext Transfer Protocol -- HTTP/1.1," http://www.w3.org/Protocols/rfc2616/rfc2616.html.
[5] Browserscope: http://www.browserscope.org/.