2013 IEEE Seventh International Symposium on Service-Oriented System Engineering

Mobile Web Browser Optimizations in the Cloud Era: A Survey

Haoyu Wang, Junjun Kong, Yao Guo and Xiangqun Chen
Key Laboratory of High-Confidence Software Technologies (Ministry of Eduction)
School of Electronics Engineering and Computer Science, Peking University, Beijing, China
{*wanghy11,kongjj07,yaoguo,cherry*}*@sei.pku.edu.cn*

***Abstract*—As most mobile devices are capable of accessing the Internet with a mobile web browser, the performance of mobile web browsers has become an interesting research topic recently. Due to network bandwidth and processing power limitations, mobile web browsers are slower compared to PC web browsers when dealing with large-size web contents and computation-intensive operations. In order to improve the performance of web browsers, researchers have attempted different techniques, including parallel optimization, web page prefetching, speculative loading, etc. With the introduction of mobile cloud computing, the performance of mobile devices can be augmented with the capability of the cloud. Mobile web browsers can also benefit from mobile cloud computing to improve the responsiveness and energy efficiency. This paper surveys current issues of mobile web browsers and approaches to optimize mobile web browsers, including client-side approaches and cloud-based approaches. We also discuss the challenges and future directions in mobile web browser optimization.**

***Keywords*-mobile web browser; cloud computing; mobile cloud computing; optimization**

I. Introduction

Mobile devices are very popular nowadays and people frequently use them for daily tasks. According to a Cisco forecast, the number of mobile devices will exceed the number of human beings on the earth by the end of 2012, and there will be 1.4 mobile devices per capita by 2016 [1]. Mobile phones and other mobile devices such as tablet computers have become the main portal for people to access the Internet.

The mobile web browser[1] is one of the most important applications on mobile devices. It is a critical channel connecting people with the Internet in ways that the PC browser never did [2]. Though mobile applications develop rapidly in recent years and many people would like to choose mobile applications to access the Internet and obtain information [3, 4], the mobile web browser is still dominant because of its security, widely available services and standardization [2].

However, mobile web browsers cannot totally fulfill users' requirements even though the capability of mobile devices is much stronger than ever. The CPU of current mobile devices can achieve dual-core 1.2GHz and the storage capability can be extended to 32GB or more [5]. Nevertheless, current mobile browsers bring much more inferior user experiences compared to PC browsers.

As many people have experienced, it usually takes seconds or even tens of seconds to open a web page on a mobile browser [6]. While loading a site such as Slashdot costs merely three seconds on a laptop, it takes 17 seconds on an iPhone using the same wireless network [7]. Researches have shown that, with the same network condition, the browsers on iPhone are 9x slower than MacBook Pro [8]. This kind of long delays may severely affect user experiences. According to a previous study, a 200ms increase in the page load latency time results in "strong negative impacts", and the delays of under 500ms may "impact business metrics" [9].

Many researchers have done plenty of works to identify the key issues leading to the poor performance of the mobile web browser. Studies have found that the long *round-trip-time* is a lethal factor to slow down the mobile browser [6, 10], which leads to longer resource loading time. Web pages are becoming more and more *complex*, which demands substantial computation resources to parse, format and render them properly [11]. Moreover, web pages are usually processed in a *single thread* manner, which cannot employ the benefit from the multi-core processor of mobile devices [12, 11].

We classify the recent studies on mobile browser optimization into two categories: (1) client-side optimizations and (2) cloud-based optimizations. Client-side optimizations include caching optimization [13, 14, 12], browser parallelization [15, 11, 8], web page prefetching [16, 17, 18, 19], speculative loading [20], etc. Cloud-based optimizations try to empower the mobile browser with the capabilities of external cloud facilities. These optimizing approaches largely enable the mobile web browser to utilize the power of cloud computing in order to offload computation-intensive or energy-consuming tasks to the cloud. The cloud-based optimization approaches include cloud-based parallelization [21], cloud-based preprocessing [22, 23, 24], computation offloading to the cloud [25, 26, 27, 28], and cooperating in the P2P mobile cloud computing [29].

This paper presents a comprehensive survey of the state-of-the-art approaches to optimize the mobile web browser in terms of improving performance and saving energy. The rest of the paper is organized as follows: Section II presents

[1]In this paper, we use mobile web browser and mobile browser interchangeably.

the challenges and issues of the mobile web browser. Section III summarizes the mobile web browsing architectures. Section IV describes the client-side approaches to improve performance of the mobile browser. In Section V, we present the existing cloud-based approaches to augment mobile web browsers. We analyze the future research directions in Section VI. Finally, in Section VII, we conclude the whole paper.

II. Challenges in Mobile Web Browsers

We first analyze the challenges and issues in mobile web browsers. The mobile web browser is similar to the PC web browser. Both of them have a similar framework and usually the same procedure of processing web pages. Thus, some challenges and issues exist in both mobile browsers and PC browsers, such as computation-intensive operations, long resource loading time, security and privacy. However, the mobile browser has its unique characteristics because mobile devices have limited computational resources, limited power supply, and expensive network traffic. These limitations force the mobile browser to carefully consider the network traffic and energy consumption issues as well.

A. Computation-intensive Operations

A web browser is a large and complicated application. It parses the HTML documents, calculates the styles and layouts, executes the JavaScript codes, displays the pages and interacts with users. Each process is complex and important.

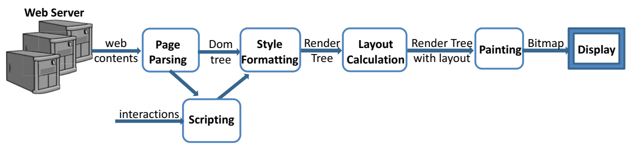

Figure 1. The procedure of opening a web page

Figure 1 shows the traditional work flow of the a web browser. The browser retrieves the web contents from the website server, including HTML documents, Cascading Style Sheets (CSS), pictures, audio files and JavaScript files. The web page is parsed into a DOM tree, which represents the structure of the document. The style for each DOM element is calculated by generating a render tree. With the layout of each element being calculated incrementally, the render tree becomes the render tree with layout. Then the page is gradually drawn to the screen.

We should identify the places where heavy computations take place. JavaScript is no longer the bottleneck thanks to the development of JavaScript optimization techniques, such as just-in-time compilation. Most popular web sites only spend 3% of the CPU time inside the JavaScript runtimes [7]. Most researchers agree that *CSS formatting* and *layout calculation* are the most computation-intensive operations and they account for more than half of the total computation time during web page processing [30, 12, 8]. Besides, some JavaScript methods change the DOM tree structure and CSS style, which may cause the browser to reflow [31]. *Reflow* affect a portion of or the whole page's layout, which is critical to browser performance [32].

B. Long Resource Loading Time

Besides the computation-intensive operations, the long resource loading time is another key issue of the mobile browser. The web contents need to be fetched first before they can be processed. Thus, the resource loading time affects the total loading time of the page. Wang *et al.* find that resource loading contributes most to the browser delay. With a 2x speedup of resource loading, we can improve the browser performance by 70% [6].

Studies [6, 10] have observed that network RTT (round-trip-time) is the key factor that leads to the long resource loading time. RTT is the delay that consists of the transmission time between the browser and the web server. It is related to the size of the web contents and the network bandwidth. In addition, a single access to a web page may cause multiple network RTTs. Because the resource loading is not fully parallelized, new resources can only be discovered by parsing the HTML documents.

C. Network Traffic

Mobile users are always concerned about the amount of network traffic, which is also one of the key differences compared to PC users. The cost of 3G is typically expensive, while Wi-Fi connectivity is not always present. But current web pages are more and more complex to pursue rich functions and delicate visual effects by introducing a lot of CSS files, pictures, and JavaScripts. Recent studies have shown that the average web page size has surpassed 1 MB [33]. Some web sites have the mobile edition, which are optimized for mobile users, such as Google and Sina. The optimized pages have much smaller size and can be more suitable for users to browse on mobile devices. However, most web sites have not provided optimized mobile version, thus we need to provide other techniques to reduce the network traffic in order to release the burdens of mobile users.

D. Energy Consumption

Energy supply is a major bottleneck for most current mobile devices. On one hand, mobile devices have very limited amount of energy available for a long period up to one day, because they usually carry a small battery[2] to supply power for driving the the whole system. On the other hand, the screen of mobile devices becomes larger and

[2]The energy capacity of battery is proportional to the size of the battery, whose size strictly depends on the size of the mobile device.

larger, with the energy consumption becoming larger and larger. Computation-intensive and I/O-intensive operations also consume lots of energy. Thus, a modern mobile browser not only should provide fast speed and quick responsiveness, but also should be energy efficient.

Previous studies [34] have presented a detailed analysis of the energy consumption of the mobile browser while loading a web page. They find that *resources downloading*, *CSS parsing* and *JavaScript execution* consume most of the energy. A recommendation of how to design a web page to minimize the energy needed is given. They also emphasize the importance of building a mobile site optimized for mobile devices, which can save both network traffic and energy.

E. Security and Privacy

Security and privacy are the persistent issues in web browsers. The browsing behavior is very private, and may include plenty of personal information. Especially in the cloud era, the private information is easy to leak. Thus, to keep the security and privacy of browser users is a major priority. Many efforts try to design a new browser to solve the issues. The authors in [35] design a new web browser by combining the operating system design principles with formal methods. The browser is split into subsystems and the communication between these systems are limited. The secure web browser [36] is proposed to modularize the web browser, and it utilizes different processes to isolate modules. In this paper, we mainly focus on approaches to improve the speed and energy efficiency of mobile browsers.

III. Mobile Web Browsing Architectures

The mobile web browsing architecture refers to the organization of the mobile browser, the web server and the cloud, as well as how they interact with each other. In the cloud computing era, the cloud plays an important role in improving the performance of mobile web browsers. The cloud and the mobile browser can work together to provide a better user experience. According to the interactions among the mobile browser, the cloud, and the web server, we categorize the mobile web browsing architecture into four schemes: (1) client-server architecture, (2) cloud-based architecture, (3) cloud-assisted architecture, and (4) cooperating architecture.

A. Client-Server Architecture

Figure 2(a) illustrates the client-server architecture, a traditional architecture for mobile browsers. When a user accesses a web page, the mobile browser in the user-side retrieves the web contents from the remote web server. All the parsing and calculations are completely done in the client side. In this architecture, the mobile browser is a fat client.

This architecture possesses some benefits. On one hand, it is easy and convenient to implement and deploy since it does not need the help of external facilities. On the other hand, it is more secure than the architecture with external supports, in that the client browser communicates with the web server directly, thus reducing the possibility of information leakage in the transmission path.

The disadvantages of this architecture are also obvious: because all the web contents fetching and computations are performed in the client side, and all optimizations can only be realized on the resource-limited mobile device, it is impossible to take advantage of external infrastructures such as the cloud.

B. Cloud-based Architecture

In Figure 2(b), we show the cloud-based architecture, where the cloud acts as an agent/proxy for the mobile browser. All the messages the browser interchanges with the server go through the cloud agent. When a user accesses a web page, the mobile browser sends the request to the cloud instead of the remote web server. The cloud loads the web page and downloads all the web contents. The cloud may perform some optimizations on the data, such as resizing the pictures. The cloud agent then sends the web contents back to the user side mobile browser. In this way, the cloud agent could compress the web contents and pre-calculate some intermediate results. In this architecture, the mobile browser acts as a thin client, in contrast to the above client-server architecture.

The benefit of this architecture is that it fully takes advantage of the ability of the cloud, which can reduce the network traffic and local processing time. However, there are two disadvantages of this architecture. First, it may increase the network latency. In the traditional architecture, the browser connects to the server directly while in this architecture the browser connects to the server via the proxy. Second, this architecture may increase the risk of privacy leakage because all messages go through the cloud.

C. Cloud-assisted Architecture

In the cloud-assisted architecture, the cloud acts as an assistant independent of the web server, which is illustrated in Figure 2(c). The cloud offers auxiliary services to the mobile browser. The mobile browser downloads web contents from the web server, and heavy operations can be offloaded to the cloud. This can substantially improve the user experience. The main difference between this architecture and the cloud-based architecture is that the messages carrying "fresh" web contents do not go through the cloud.

The benefit of this architecture is that it can use the power of the cloud to improve performance without changing its main infrastructure.

D. Cooperating Architecture

In the cooperating architecture, mobile devices in the local vicinity can act as the resource providers, thus forming a mobile peer-to-peer network, which can also be called a

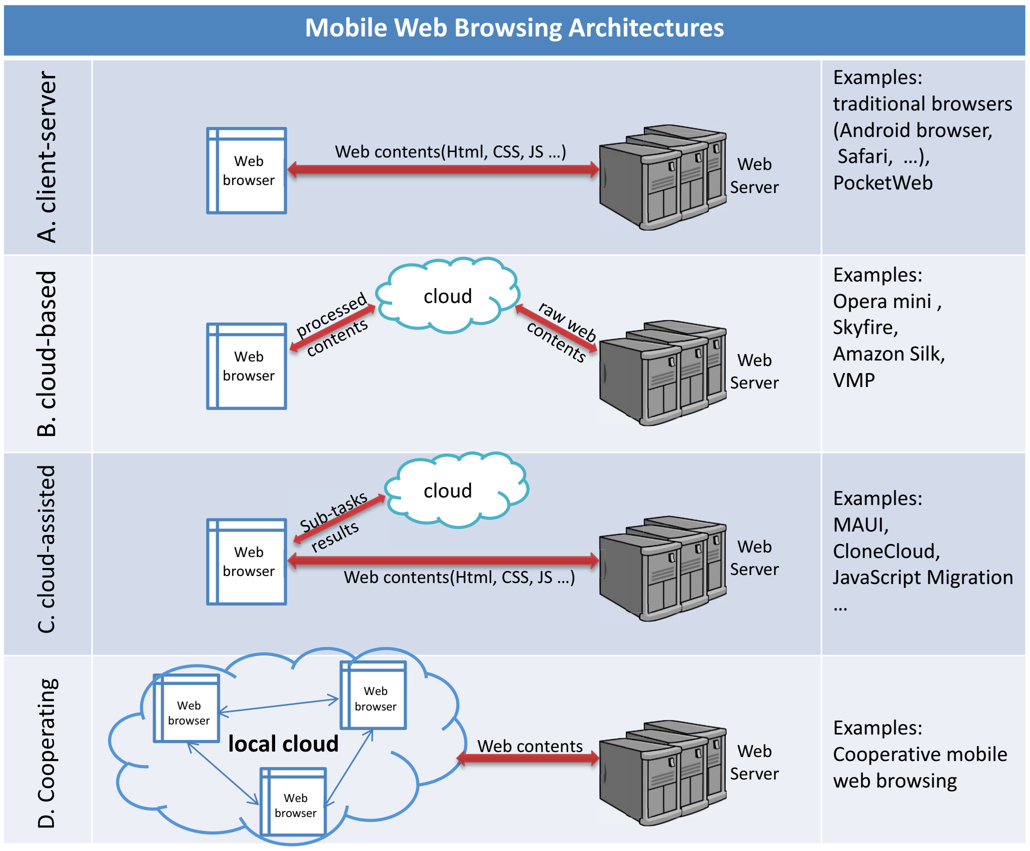
Mobile Web Browsing Architectures
A. client-server
Web browser
Web contents(Html, CSS, JS ...)
Web Server
Examples: traditional browsers (Android browser, Safari, ...), PocketWeb
B. cloud-based
processed contents
cloud
raw web contents
Examples: Opera mini, Skyfire, Amazon Silk, VMP
C. cloud-assisted
Sub-tasks
results
Web contents(Html, CSS, JS ...)
Examples: MAUI, CloneCloud, JavaScript Migration ...
D. Cooperating
local cloud
Web contents
Examples: Cooperative mobile web browsing

Figure 2. An overview of mobile web browsing architectures

local mobile cloud [37]. They cooperate to download web contents and/or process contents, as Figure 2(d) shows. The benefit of this architecture is that it can use the power of other mobile devices and cooperate to offer a faster experience.

IV. Client-side Optimizations

Client-side optimization approaches are the optimizations applied to the mobile browser itself. They are easy to deploy and much secure than cloud-based optimization approaches.

A. *Effective Caching*

Resource loading contributes most to the browser delay [6]. Web browsers can keep frequently used web resources locally within the browser cache to save round-trip-time and bandwidth, thus speeding up resource loading. Intuitively, a good browser caching scheme may improve the performance of the mobile browser.

One important way to improve the browser cache is to reduce the miss rate. To achieve this, two basic approaches are proposed: increasing the cache size [13, 14] and enhancing the replacement algorithm [38, 39]. Only increasing the cache size will not improve the effectiveness of the mobile browser greatly [40]. Even increasing the cache size to infinite, it only can reduce 10% of the cache misses [40].

A study [40] finds that the process of re-validations greatly reduces the effectiveness of the mobile browser cache. The cached resources have two states, either fresh or expired. If the state is fresh, the resource can be reused without confirming with the server. If the state is expired, the browser needs to connect to the server to confirm if the resource can be reused. In this process, non-cached and expired resources bring in at least one round-trip-time, which reduces the effectiveness of the cache. As mentioned earlier, network RTT is the key factor leading to the long resource loading time. Thus the traditional caching scheme may not be effect.

Smart caching is proposed in [12] as a supplementary of the traditional caching scheme. Though the research focuses on the PC web browser, it can also be used in the mobile browser. The traditional caching scheme stores the raw resources such as pictures, CSS files, and so on. It aims to reduce the resource loading time, but the smart caching scheme intends to reduce the time of computation-intensive operations. This research proposes to cache the intermediate results of web page processing. Thus they cache the CSS formatting results and layout calculation results,

Table I
A COMPARISON OF CLIENT-SIDE OPTIMIZATIONS

Approaches	Computation-intensive operations	Resource loading time	Network traffic	Energy efficiency	Security&privacy	Implementation Complexity	Storage Overhead
Increase Cache Size ([13, 14])	/	Lowly Effective	Lowly Effective	/	/	Easy	depend on the cache size
Enhance cache replacement algorithm ([38, 39])	/	Lowly Effective	Lowly Effective	/	/	Easy	/
Smart Caching ([12])	Highly Effective	/	/	/	/	Medium	Low Overhead
Client-Side Parallelization ([15],[11],[8])	Highly Effective	/	/	Highly Effective	/	Difficult	/
Traditional Prefetching ([16],[17],[18])	Lowly Effective	Lowly Effective	High Overhead	High Overhead	/	Medium	/
PocketWeb ([19])	/	Highly Effective	Low Overhead	Low Overhead	/	Medium	Low Overhead
Speculative loading ([20])	/	Highly Effective	/	/	/	Medium	Low Overhead

called smart *style caching* and *layout caching* separately. The smart style caching constructs a SSC tree that is similar to a DOM tree, and records the corresponding style property of each DOM element. A triple `<ID, class, TagName>` is used to identify SSC element. Cached style properties can be applied to the identical element in the subsequent visits to the same page. Layout caching records the layout operation results. During subsequent visits to the same page, if the needed layout operation for the same render object is calculated before and recorded in the cache, the result can be reused. This approach can greatly reduce the computation operations. It is not only effective for web page revisit, but also can eliminate the duplicated or unnecessary computations for the visit to a new web page. Meanwhile, this approach only brings in very small storage overhead.

B. Client-side Parallelization

Web content is usually processed in a single thread manner in order to get the proper result [11]. Google's Chrome browser [41] treats each tab as a separate process to increase parallelism, while the web page in each tab is still processed in a single thread manner. Parallelizing the computation-intensive parts of the browser can improve its performance. Jones et al. [11] analyze the browser and discuss how parallelism can improve the browser to be responsive and energy-efficient. They describe the design of a parallel web browser, and analyze the possibility and algorithms to parallelize each browser component, including parallelizing the fronted, parallelizing page layout and parallelizing the scripting. The parallel approaches suite for both mobile browsers and PC browsers.

Badea et al. [15] claim that they find the CSS selectors heavily biased towards descendant selectors and the matching process for a CSS rule results in non-match in most cases. Firefox [42] uses a sequential CSS rule-matching mechanism. Thus they parallelize the CSS rule-matching component of Firefox and focus on the non-matching descendant selectors. The result shows that it can achieve up to 1.84x page-load speedup with two worker threads. Meyerovich et al. [8] propose a similar approach. They parallelize the layout engine and introduce three new algorithms for CSS selector matching, layout solving, and font rendering respectively. They report that the performance speedups are up to 80x based on their browser model. Fortuna et al. [43] propose a method to parallelize the JavaScript functions, which can greatly increase JavaScript execution speed. Considering that the execution of JavaScript is not a bottleneck in mobile browsers, this method cannot make obvious improvement for the total web page processing time.

C. Web Page Prefetching

Web page prefetching is the approach to predict which web pages user will visit in the near future and download the predicted web pages before actual visits. If the predictions hit, the browser can use the resources preloaded locally. The concept of prefetching is also included in the HTML5 specification.

In some cases, it's possible to predict what a user is likely to click next. For example, when a user is reading a multi-page article, he/she is likely to click on the "Next Page" link [16]. In this case, it would be great if the browser could begin loading the page before the user clicks.

However, in many cases, it is difficult to know what a user is likely to access next. The most simple and naive way is to prefetch all the pages referenced by the current page. A page may have many links. If the browser tries to fetch many links at the same time, it would cause too much traffic and most of the fetched contents may be useless. Many prefetching algorithms are proposed to optimize web prefetching [18, 44, 17]. Nevertheless, these approaches are not applicable to the mobile browser because they don't consider the constraints of mobile devices, such as high network latency and limited energy capacity.

PocketWeb [19] can accurately predict user's web accesses by analyzing user's access history. The authors of PocketWeb analyze the web access traces from 8000 mobile users over 3 months and find that users often visit a small set of web pages from their phones. The small set of web pages (i.e., URLs) account for more than 70% of a user's web accesses. User's web access history is used to train the user's web access model. One feature vector for each history URL is extracted, and these feature vectors can be used to train the web access model using a machine-learning approach. User's action will trigger the prediction, such as opening the web browser. Feature vectors for frequently accessed URLs are generated and then sent to the prediction model. The prediction model calculates the access probability for every feature vector. According to the probability of every feature vector, one or more URLs are prefetched. Model training is executed in the PC or cloud without energy costs in the mobile side. The result shows that for 80% - 90% of the users the model can accurately prefetch 60% of the URLs within 2 minutes before request, which can provide a faster web browsing experience without increasing energy dissipation.

D. Speculative Loading

Wang et al. [20] propose the speculative loading approach in order to reduce the resource loading time. When a user accesses a web page, the browser first predicts which resources may be useful, and then loads the sub-resources along with the main resources, which can save the round-trip-time. The main difference between speculative loading and prefetching is that speculative loading predicts which resource the user may need while prefetching predicts which page the user may visit. The main difficulty is how to accurately predict which sub-resources will be useful. Wang et al.'s approach is based on a web site's *resource graph*, which is constructed using the visiting history. Different pages of the same web site may share lots of sub-resources. If the web page visit is a revisit, the sub-resources needed can be found in the resource graph. If the web page visit is a new visit, the browser can predict which sub-resource may be useful according to the shared sub-resource nodes. A speculative loading prototype called Tempo is implemented based on the WebKit [45]. It can reduce the mobile browser delay by 1 second (20%) under 3G network. This approach causes little wireless data usage and storage overhead.

E. Comparison of the Client-side Approaches

We present a comparison of the existing client-side optimization approaches for the mobile browser in Table I from these aspects: computation-intensive operation time, resource loading time, network traffic, energy efficiency, security and privacy, implementation complexity, and storage overhead. In the table, "highly effective" means that the approach can greatly optimize the corresponding aspect and "lowly effective" means that the approach cannot improve the performance greatly. The implementation complexity is divided into three categories according to the difficulty: easy, medium and difficult.

Based on the comparison, we have the following observations:

- Both smart caching and client-side parallelization are effective to address the issue of computation intensive operations. The client-side parallelization can also improve the energy efficiency. Speculative loading is effective to address the resource loading issue.
- PocketWeb and speculative loading are effective to address the resource loading issue with low or no overhead.
- None of these approaches take into account security and privacy issues.
- The implementation complexity of smart caching, PockerWeb and speculative loading is medium, which is easier than the client-side parallelization.

V. Cloud-based Optimizations

Mobile cloud computing combines cloud computing with mobile devices. Mobile devices are limited by processor capability, battery power and bandwidth. While the cloud has super ability and unlimited resources, which can be used to empower mobile devices. By offloading the computation-heavy tasks to the cloud or moving the huge amount of data to cloud data center, the mobile devices can improve performance and reduce energy consumption. Many applications get benefit from mobile cloud computing and various offloading approaches are proposed. As one of the most important applications in mobile devices, the mobile browser can also improve its performance with the aid of mobile cloud computing. Many research works focus on this. Actually, people began to use the power of external infrastructure many years ago to augment the mobile browser, such as offloading the computation to the vicinal computers.

We survey the related work on augmenting the mobile browser with the cloud, and categorize them into four major themes: (1) cloud-based parallelization, (2) cloud-based preprocessing, (3) computation offloading to the cloud, and (4) cooperating with the P2P mobile cloud.

A. Cloud-based Parallelization

Cloud-based parallelization further exploits task parallelism to speed up mobile web browsing compared with client-side parallelization. A typical work falling into this theme is Adrenaline [21].

Table II
A COMPARISON OF CLOUD-BASED OPTIMIZATIONS

Approaches	Architecture	Computation-intensive operations	Resource loading time	Network traffic	Energy efficiency	Security and privacy	Implement Complexity	Storage Overhead
Cloud-based Paralleliza-tion ([21])	Cloud-Based	Highly Effec-tive	Highly Effec-tive	Highly Effective	/	Weakened	Difficult	/
Cloud-based Preprocess-ing ([46, 47, 48, 49, 22, 24, 23])	Cloud-Based	Highly Effec-tive	Highly Effec-tive	Highly Effective	Highly Effec-tive	Weakened	Difficult	/
Computation Offloading ([25, 26, 27, 28])	Cloud-Assisted	Highly Effec-tive	/	Low Overhead	Highly Effec-tive	Weakened	Medium	Low Overhead
Cooperating ([29])	Cooperating	/	Moderately Effective	/	/	Weakened	Medium	/

Mai et al. [21] propose an approach to parallelize mobile browser at the web page level, and design a prototype system called the *Adrenaline*. Adrenaline consists of a server-side preprocessor called Adrenaline server, and a client-side browser. It adopts a cloud-based architecture (refer to Section III-B). The client-side browser fetches web contents from the Adrenaline server rather than the Internet. The Adrenaline server decomposes the web page into loosely coupled mini pages which is a "complete web page". When user triggers a request in the client side, the request is sent to the Adrenaline server first. The Adrenaline server fetches web contents from the web server, optimizes and decomposes the web page into several mini pages. Then the mini pages are sent to the mobile browser. The browser side downloads and renders these mini pages in parallel using multiple processes. At last, the browser aggregates all these mini pages. Adrenaline can reduce the page load latency by 1.75 seconds and improve the page load latency time by 1.54x on average. However, this approach faces with some difficulties. The first difficulty is how to keep the correct web semantics. The final results should correctly response to the DOM and UI events, synchronize all the global data. Adrenaline chooses to put all JavaScript into one process to keep JavaScript and DOM compatibility. Bloom filter is used to minimize the synchronization overhead. The second difficulty is how to decompose the web page into mini pages. The number of mini pages and the way to decompose a web page can influence the effectiveness of this approach. Unfortunately, this paper does not provide more detailed explanation on this.

B. Cloud-based Preprocessing

In cloud-based preprocessing, the cloud acts as an agent of the mobile browser, and all the traffic between the web server and the mobile browser passes through the cloud. The cloud can preprocess and compress the web contents, then send the results to the mobile browser. Some of current mobile browsers use the similar approach, such as Deepfish [46], Skyfire [47], Opera mini [48] and Amazon Silk [49]. When a user requests a web page, the request will be sent to the cloud first. The cloud fetches, compresses, and processes the web contents before they are returned to the mobile device. The cloud uses its advantage of high bandwidth to load the pages quickly. If the web page is accessed by other users before, such contents exist in the cloud, then resource loading can be faster. The cloud can perform some optimizations, such as resizing the pictures to adapt to the screen resolution of the mobile device. The data sent to the browser is compressed, thus can improve energy efficiency and reduces the traffic for mobile device. For example, Opera mini announces to compress web pages by up to 90%, saving both time and money for mobile users [50]. For some other mobile browsers such as Skyfire, the web page can be partially or fully rendered by the cloud, which can greatly reduce the computation on the mobile side.

Thin Client [22] is proposed to offer a remote execution approach for PDAs. The client sends the user input to the cloud, and the server returns screen updates to PDA for display. The authors in [23] compare the web browsing performance of thin clients against fat client web browsers. The result shows that thin clients can provide better web browsing performance than fat clients in terms of both speed and capabilities.

Similarly, Zhao et al. [24] propose an approach called VMP (*Virtual Machine Based Proxy*) to shift the computation from the mobile browser to the VMP in order to reduce not only browsing delays but also energy consumption. VMP starts a virtual machine for each mobile browser request and the resources are released after the request is completed. VMP handles all the HTTP requests, replies, and executes various JavaScript and Flashes. A compressed screen copy is sent to the mobile browser in the end. They deployed this system in 3G networks and implemented the prototype using the Xen virtual machine and Android phones. The result shows that VMP reduces delay by more than 80% and reduces energy consumption by more than 45% [24].

However, some issues should be concerned for the cloud-based preprocessing approaches. One issue is the additional latency. Traditional web browsing connects to the web server directly while these approaches place an agent in the cloud between the browser and the web server. The additional latency may vary based on the location of the cloud agent, and in some cases, the browser might be unable to access the cloud agent. For example, the server of the international version of Opera mini is blocked in China, and Chinese users can only use the Opera mini China version whose agents are placed in China [51]. Other issues are security and privacy. Though the messages between the browser and the proxy are encrypted, the cloud-based architecture cannot ensure the end-to-end security. All the user's personal information and visiting history pass through the proxy server, which may results in leaking the user's private information.

C. Computation Offloading

Computation offloading to the cloud is an efficient way to improve the mobile browser's responsiveness and energy efficiency [11]. The mobile browser can offload heavy computations of web page processing to a resourceful cloud and receive results from the cloud. Computation offloading is easy to be confused with cloud-based preprocessing. In the computation offloading approaches, the mobile browser offloads sub-tasks to the back-end cloud. It is the mobile browser who decides when and what need to be offloaded to the cloud. While in the cloud-based preprocessing approaches, all the web contents pass through the front-end cloud agent, and the agent processes the contents before they are sent to server. It is the cloud who decides what need to be processed and sent.

A significant amount of research has been performed on computation offloading, such as MAUI [25] and CloneCloud [26]. Most of these methods also can be applied to the mobile browser. Some other research aims at offloading for the mobile browser. Wang et al. [27] propose a JavaScript offloading framework which can offload JavaScript code to the cloud. The authors in [28] present a heterogeneous offloading framework to offload multimedia processing tasks to the cloud and merge remote display with the mobile browser interface.

There might also exists additional latency issue for the computation offloading approaches. However, if the latency of connecting to the cloud is high, then the mobile device could decide not to offload the tasks to the cloud.

D. Cooperating with the P2P Mobile Cloud

Mobile devices are usually considered to be resource consumers in mobile cloud computing. However, mobile devices can also act as resource providers so as to establish a peer-to-peer mobile cloud [52]. A mobile browser in one device can cooperate with various mobile devices within its proximity to improve its performance.

Perrucci et al. [29] propose an approach of cooperative mobile web browsing among multiple mobile phones in the vicinity. In this approach, multiple nearby mobile devices share their cellular links using Bluetooth connections. Cooperatively downloading web contents can increase the virtual capacity of cellular link and reduce the resource loading time. They divide the resource loading phase into two sub-phases: web page processing and components downloading. In the web page processing phase, the browser searches the sub resources from the contents of the URL and ranks these sub resources according to their size. These sub resources are divided to several sub-lists equally according to the number of cooperating mobile devices. Then in the downloading phase, each mobile device downloads its sub-lists resources and then sends the resources to the device who visits the page. There are some disadvantages in this approach. Traditional browsers discover the new sub resources when parsing a loaded resource, in which loading and resource parsing run at the same time. However, in this approach, the resource loading phase is divided into two separate phases. Downloading the sub resources can only happen after parsing of the main resource is finished.

E. Comparison of the Cloud-based Approaches

Similar to the previous comparison of client-side approaches, we also provide a comparison of the cloud-based approaches from these aspects: computation-intensive operations time, resource loading time, network traffic, energy efficiency, security and privacy, implementation complexity, storage overhead. Besides, we also compare the mobile web browsing architectures of these approaches. The comparison result is shown in Table II.

The following observations can be drawn based on the comparison:

- Cloud-based parallelization and cloud-based preprocessing approaches can optimize computation-intensive operations, shorten resource loading time, and reduce network traffic.
- Cloud-based preprocessing and computation offloading are effective to improve energy efficiency.
- All of the cloud-based approaches face the security and privacy issues.
- Cloud-based parallelization and cloud-based preprocessing are difficult to implement, because servers with low network latency are needed and the internal structure of the web browser must be changed.

VI. Future Research Directions

The client-side approaches can combine with the cloud-based approaches to improve the mobile browser performance further. For example, the smart caching scheme can be integrated with all the cloud-based approaches to reduce computation-intensive operations. Pre-rendering techniques can also be combined with PocketWeb [19]. Web content pre-rendering is the extension of prefetching. Instead of merely fetching the resources, the browser can conduct the computation tasks before visiting, such as pre-calculating the CSS style and layout of each element [16]. PocketWeb [19] can accurately prefetch most URLs the user visits, thus the fetched contents can be further processed in advance. The mobile browser can pre-render the prefetched pages, which will offer users an instant browsing experience.

Mobile cloud computing is progressing, and many new technologies are emerging. The mobile browser can further benefit from mobile cloud computing. For example, Cloudlet [53] can be used in mobile browsers to reduce the network RTT. The cooperation of mobile browser with mobile cloud computing can be further explored. They can not only work cooperatively to download web contents, but also calculate and render them collectively. Some research on cooperative mobile cloud computing may provide more benefits [52, 5, 54].

With the development of HTML5, the functions of the web browsers are more and more powerful. There is also a trend to combine the browser and the operating system into an ecosystem, which can be called "boot to browser". An example of this is Mozilla's "Boot To Gecko", which is renamed to Firefox OS [55] recently. In the Firefox OS, everything becomes an HTML5 application. Web applications can directly access different kinds of resources (hardwares and softwares) through WebAPI in the Firefox OS, thus it runs much faster. However, there are many challenging issues in "boot to browser", such as standardization and security, which brings a lot of potential research topics and opportunities.

VII. Conclusion

This paper analyzes the key issues in mobile web browsing and surveys a body of research associated with mobile web browser optimization. We classify these optimization approaches into two categories: the client-side approaches and the cloud-based approaches. For each category, we analyze the effects of corresponding approaches and provide a comprehensive comparison. We believe that the optimization of mobile browsers is a promising research direction, and more researchers should put their efforts into making mobile browsers more powerful and more energy-efficient at the same time.

Acknowledgment

This work is supported in part by the National High-Tech R&D Program under Grant No.2011AA01A202, the National Natural Science Foundation of China under Grant No.61103026, and the Science Fund for Creative Research Groups of China under Grant No.60821003.

References

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 20112016." [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[2] Y. yu, "The mobile browser dominates in emerging markets." [Online]. Available: http://allthingsd.com/20120914/the-mobile-browser-dominates-in-emerging-markets/

[3] "In the app economy, does the mobile browser matter?" [Online]. Available: http://gigaom.com/2010/03/26/in-the-app-economy-does-the-mobile-browser-matter/

[4] "The mobile content war continues: Apps edging out mobile browser." [Online]. Available: http://socialfresh.com/mobile-apps-vs-browser/

[5] E. Miluzzo, R. Cáceres, and Y.-F. Chen, "Vision: mclouds - computing on clouds of mobile devices," in *Proceedings of the third ACM workshop on Mobile cloud computing and services (MCS '12)*, 2012, pp. 9–14.

[6] 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 (HotMobile '11)*, 2011, pp. 91–96.

[7] Leo A. Meyerovich, "Rethinking browser performance," in *USENIX;login*, 2009.

[8] L. A. Meyerovich and R. Bodik, "Fast and parallel webpage layout," in *Proceedings of the 19th international conference on World wide web (WWW '10)*, 2010, pp. 711–720.

[9] "Performance related changes and their user impact." [Online]. Available: http://www.techpresentations.org/Performance_Related_Changes_and_their_User_Impact

[10] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl, "Anatomizing application performance differences on smartphones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys '10)*, 2010, pp. 165–178.

[11] C. G. Jones, R. Liu, L. Meyerovich, K. Asanović, and R. Bodík, "Parallelizing the web browser," in *First USENIX Workshop on Hot Topics in Parallelism (HotPar '09)*, 2009.

[12] K. Zhang, L. Wang, A. Pan, and B. B. Zhu, "Smart caching for web browsers," in *Proceedings of the 19th international conference on World wide web (WWW '10)*, 2010, pp. 491–500.

[13] J. Bixby, "Early findings: Mobile browser cache persistence and behaviour." [Online]. Available: http://www.webperformancetoday.com/2012/07/12/early-findings-mobile-browser-cache-persistence-and-behaviour/

[14] S. Souders, "Call to improve browser caching." [Online]. Available: http://www.stevesouders.com/blog/2010/04/26/call-to-improve-browser-caching/

[15] C. Badea, M. R. Haghighat, A. Nicolau, and A. V. Veidenbaum, "Towards parallelizing the layout engine of firefox," in *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, 2010.

[16] "Web developer's guide to prerendering in chrome." [Online]. Available: https://developers.google.com/chrome/whitepapers/prerender

[17] D. T. Hoang, P. M. Long, and J. S. Vitter, "Dictionary selection using partial matching," *Inf. Sci.*, vol. 119, no. 1-2, pp. 57–72, Oct. 1999.

[18] C. Bouras and A. Konidaris, "Predictive prefetching on the web and its potential impact in the wide area," *World Wide Web*, vol. 7, no. 2, pp. 143–179, 2004.

[19] D. Lymberopoulos, O. Riva, K. Strauss, A. Mittal, and A. Ntoulas, "Pocketweb: instant web browsing for mobile devices," in *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, 2012, pp. 1–12.

[20] 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 (WWW '12)*, 2012, pp. 31–40.

[21] H. Mai, S. Tang, S. T. King, C. Cascaval, and P. Montesinos, "A Case for Parallelizing Web Pages," in *Proceedings of the 4th USENIX Workshop on Hot Topics in Parallelism (HotPar '12)*, 2012.

[22] J. Kim, R. A. Baratto, and J. Nieh, "pthinc: a thin-client architecture for mobile wireless web," in *Proceedings of the 15th international conference on World Wide Web*, 2006, pp. 143–152.

[23] A. M. Lai, J. Nieh, B. Bohra, V. Nandikonda, A. P. Surana, and S. Varshneya, "Improving web browsing performance on wireless pdas using thin-client computing," in *Proceedings of the 13th international conference on World Wide Web (WWW '04)*, 2004, pp. 143–154.

[24] B. Zhao, B. C. Tak, and G. Cao, "Reducing the delay and power consumption of web browsing on smartphones in 3g networks," in *Proc. of 31st International Conference on Distributed Computing Systems (ICDCS)*, 2011, pp. 413 –422.

[25] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys '10)*, 2010, pp. 49–62.

[26] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems (EuroSys '11)*, 2011, pp. 301–314.

[27] X. Wang, X. Liu, Y. Zhang, and G. Huang, "Migration and execution of javascript applications between mobile devices and cloud," in *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity (SPLASH '12)*, 2012, pp. 83–84.

[28] Y. Zhang, X. tao Guan, T. Huang, and X. Cheng, "A heterogeneous auto-offloading framework based on web browser for resource-constrained devices," in *Proceedings of International Conference on Internet and Web Applications and Services*. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 193–199.

[29] G. P. Perrucci, F. H. P. Fitzek, Q. Zhang, and M. D. Katz, "Cooperative mobile web browsing," *EURASIP J. Wirel. Commun. Netw.*, vol. 2009, pp. 7:1–7:9, Jan. 2009.

[30] "Ie8 performance." [Online]. Available: http://blogs.msdn.com/b/ie/archive/2008/08/26/ie8-performance.aspx

[31] L. Simon, "Minimizing browser reflow." [Online]. Available: https://developers.google.com/speed/articles/reflow

[32] "Browser reflows & repaints: How do they affect performance?" [Online]. Available: http://ajaxian.com/archives/browser-reflows-how-do-they-affect-performance

[33] C. Gurney, "Average web page size grows to over 1 mb." [Online]. Available: http://torsionmobile.com/2012/06/26/average-web-page-size-grows-to-over-1-mb/

[34] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery?: analyzing mobile browser energy consumption," in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 41–50.

[35] C. Grier, S. Tang, and S. King, "Secure web browsing with the op web browser," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, 2008, pp. 402 –416.

[36] S. Ioannidis and S. M.Bellovin, "Building a Secure Web Browser," in *Proceedings of the FREENIX Track of 2001 USENIX Annual Technical Conference*, 2001.

[37] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84 – 106, 2013.

[38] K. Muralidhar and N. Geethanjali, "Fuzzy Replacement Algorithm for Browser Web Caching," *International Journal of Engineering Research and Applications (IJERA)*, vol. 2, no. 3, pp. 3017–3023, 2012.

[39] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374–398, Dec. 2003.

[40] Z. Wang, F. Lin, L. Zhong, and M. Chishtie, "How effective is mobile browser cache?" in *Proceedings of the 3rd ACM workshop on Wireless of the students, by the students, for the students (S3 '11)*, 2011, pp. 17–20.

[41] "Google chrome." [Online]. Available: www.google.com/chrome

[42] "Mozilla firefox." [Online]. Available: http://www.mozilla.org/en-US/

[43] E. Fortuna, O. Anderson, L. Ceze, and S. Eggers, "A limit study of javascript parallelism," in *Proceedings of the 2010 IEEE International Symposium on Workload Characterization (IISWC)*, 2010.

[44] E. Cohen and H. Kaplan, "Prefetching the means for document transfer: a new approach for reducing web latency," in *Proceedings of INFOCOM 2000*, 2000, pp. 854–863.

[45] "The webkit open source project." [Online]. Available: http://www.webkit.org/

[46] Wikipedia, "Microsoft live labs deepfish." [Online]. Available: http://en.wikipedia.org/wiki/Microsoft_Live_Labs_Deepfish

[47] ——, "Skyfire (web browser)." [Online]. Available: http://en.wikipedia.org/wiki/Skyfire_(web_browser)

[48] ——, "Opera mini." [Online]. Available: http://en.wikipedia.org/wiki/Opera_Mini

[49] ——, "Amazon silk." [Online]. Available: http://en.wikipedia.org/wiki/Amazon_Silk

[50] "Mtn's opera mini users grow by 1665%." [Online]. Available: http://www.opera.com/press/releases/2012/02/27_4/

[51] D. Bournique, "Opera mini blocked in china." [Online]. Available: http://blog.wapreview.com/5719/

[52] E. E. Marinelli, "Hyrax: Cloud computing on mobile devices using mapreduce," Masters Thesis, 2009.

[53] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[54] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing and Services*, 2010, pp. 6:1–6:5.

[55] "Firefox os." [Online]. Available: http://www.mozilla.org/en-US/firefoxos/