A Visual Method to Find Performance Bottlenecks for Mobile Web and Hybrid Applications

Tao Cai, Li Li, Wu Chou[1], Ching Man Yan, Haifeng Fu, Wong Ka Wah, Kangheng Wu, Zhibin Lei[2]

Abstract. It is very difficult to test web performance on a smart mobile, where performance is often an issue, because of lacking such kinds of tools. In this paper, we provide a new method to let mobile web developers to find the performance bottlenecks of their web and hybrid applications easily. The method is based on the concept of Timed Operation Execution Chart, which allows developers to visualize the internal execution order and duration of hundreds of HTML processing operations for each HTML page, without having to change the web application or the web server. The proposed method has been applied to 3 typical mobile web applications and the preliminary results show the approach can help us to identify bottlenecks in these applications.

Keywords: HTML5 • Mobile Devices • Web • Browser

1. Introduction

After Mark Zuckerberg, CEO of Facebook, declared that “the biggest mistake that we made as a company is betting too much on HTML5 as opposed to native because it just wasn’t there” [13.], Tobie Langel, a Facebook software engineer, detailed the performance issues they encountered while building their mobile web [7.].

1 Tao Cai, Li Li, Wu Chou(✉)
Shannon Lab of Huawei Technologies Inc
e-mail: {t.cai, Li.NJ.Li, Wu.Chou}@huawei.com

2 Ching Man Yan, Haifeng Fu, Wong Ka Wah, Kangheng Wu, Zhibin Lei(✉)
HK Applied Science and Technology Research Institute, Hong Kong Science Park, Shatin, Hong Kong
e-mail: {rachelc, hffu, kawwong, khwu, lei}@astri.org

The first problem Langel mentioned was "the lack of tooling in mobile browsers makes it very difficult to dig down and find out what the real issues are. Hence tooling, or rather, lack-thereof is a key issue." [7.]. It is clear that HTML5 is facing big challenges on mobile platform as there's not any ready-to-use tool that can help the web developers to find the real issues of their web apps.

There are three ways to deploy applications on mobile platforms: native application, web application and hybrid application. Native application is a kind of application built for a specific mobile platform with the platform SDK, tools and programming language.

Web application uses standard web technologies—typically HTML5, JavaScript and CSS to build "application-like" experience within a web browser on smart mobile. Hybrid application is an integration of native app and web app. It embeds web app inside a thin native container, combining the best elements of native and web page.

We developed a method to test the performance of both web and hybrid applications. The method gives detailed and visual performance information in Timed Operation Execution Chart, which shows the execution time and order of main HTML operations involved in processing a web page. The chart facilitates developers to identify performance bottlenecks in individual web page and performance patterns across web pages. It can also serve as a starting point for drill-down analysis.

2. Related Work

J. Huang et al. [1.] developed a cross-platform tool named 3GTest [16.] which has been renamed to MobiPerf [18.] for measuring a group of phones' network performance under controlled network conditions. Their method could not measure the internal operations of the browser, they took it for granted that "HTML rendering and JavaScript execution may become the bottleneck for web browsing" which had been proved to be wrong by Zhen Wang et al. [2.].

Zhen Wang et al. [2.] found that resource loading contributes most to the browser delay. They used a new method to test "how far can client-only solution go for mobile browser speed" [3.]. But for a Hybrid app, their method would not work as the web page can't be loaded in their browser.

3. Method

Our method collects performance data about the operations through an instrumented Webkit. In order to introduce this method, we briefly discuss the architecture and the nature of these operations.

WebKit [5.] is an engine designed to allow web browsers to render web pages. It is used as the basis for the system browser in Apple iOS, Android and Blackberry.

WebView [4.] is an interface between native applications and WebKit. Web pages can be rendered in a powerful WebView that serves as a web browser embedded inside and controlled by native applications.

3.1 Main Processes in WebKit

Figure 1 shows the main internal processes in WebKit [12. , 14. , 15. , 20.]. When a HTML page is loaded, WebKit will parse it and download all its sub-resources. After some resources are loaded, WebKit will compute style and layout and render them on screen. A JavaScript engine will be in charge of executing the JavaScript codes.

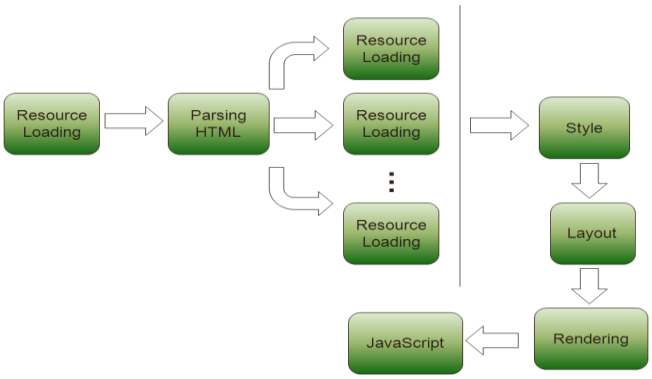


Fig. 1 Main Processes in WebKit

3.2 Libwebcore

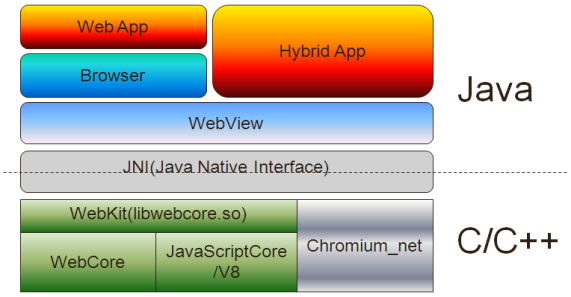


Fig. 2 Architecture of Web App & Hybrid App

Figure 2 shows the main components involved in mobile web and hybrid application on Android platform. These applications will call WebView to load its web page, and WebView will call WebKit APIs through JNI.

Our method is developed on WebKit, which involves the following steps:

- Modify the source codes for recording time information in sub-resource loading, HTML parsing, styling, layout, painting and scripting parts.
- Replace the "libwebcores.so" on the smart phone.
- Open the web app and record the timestamps information

Our method can measure the performance and the bottlenecks of web apps more efficiently. And it is easy to apply to smart phones or smart pads on the market. The method has some distinct advantages:

- Can be deployed onto all kinds of smart mobiles based on Android system.
- Can measure both mobile web app and mobile hybrid app easily.

But it also has a disadvantage: For different version of Android system, it needs to rebuild different version of libwebcore.so to keep it compatible.

3.3 Test Targets and Results

We conducted our experiments on a Huawei Honor+ (U8950D) smart phone with 2 different network setups: WiFi network based on Business Broadband Setup (Broadband Setup) and 3G network (3G Setup).

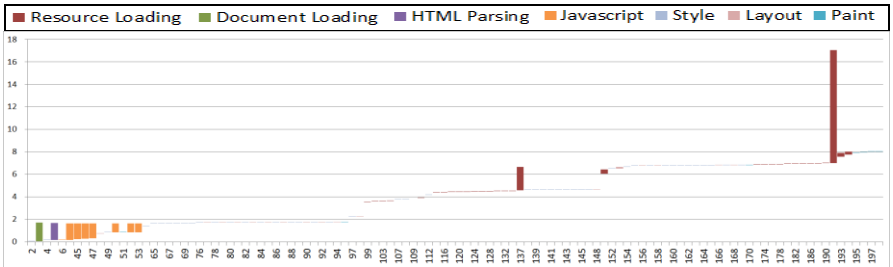
To visualize the operations in the log file, we display operations involved in a HTML page with a Timed Operation Execution Chart. In this chart, the X-axis consists of the resources' sequence number ordered by the operations' start time on them. The Y-axis is the time axis in unit of seconds. Each operation is represented by a vertical bar whose bottom position marks its start time and whose top position marks the end time. The color of the bar gives the operation type. We treat the start time of the test as zero for ease of comparison. From the chart, we can see the start and end time for each operation. From the positions and heights of the operation bars, we can easily see the order and dependence of all the operations executed for a HTML page.

To illustrate how our measuring tool works, we choose three well-known applications: Wikipedia, WordPress and DisneyWorld, respectively from the three popular mobile application frameworks: PhoneGap[9.], WebView and jQuery Mobile [6.].

3.4.1 Wikipedia Mobile App [10.]

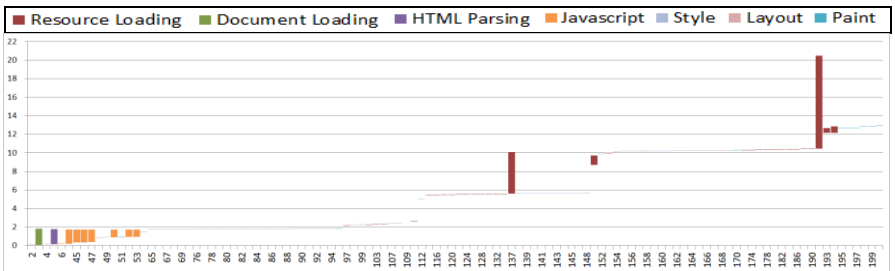
In this test, the application is started and waited until the default page is loaded and displayed properly. The Time Operation Execution Chart and its summary table are displayed in Figure 3 and Figure 4. The first row of the table represents the

number of operations taken in each category. Second row of data represents the total time spent (in seconds) by those operations.



	Resource Loading	Document Loading	HTML Parsing	Paint	Javascript	Style	Layout	Total
Count	32	1	1	7	8	46	17	17.072908
Sub Total Time (s)	13.177913	1.685833	1.543415	0.046057	7.995639	0.072038	0.164295	

Fig. 3 Broadband Setup, Wikipedia Mobile App



	Resource Loading	Document Loading	HTML Parsing	Paint	Javascript	Style	Layout	Total
Count	32	1	1	9	8	46	17	20.482714
Sub Total Time (s)	16.816607	1.761288	1.596424	0.021335	8.114874	0.078624	0.182526	

Fig. 4 3G Setup, Wikipedia Mobile App

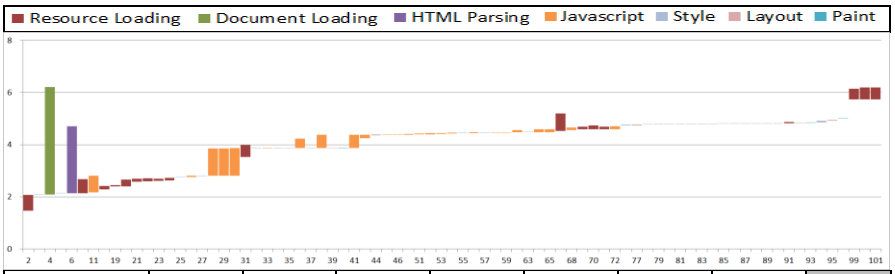
From the result, we know that most of the operations performed are resource loading. Moreover, the time spent on sytle, layout & painting is very small when it compare to resourcse loading.

For this application, out of 32 resources loading, only 3 resources are from the Internet and the rest are from the internal web server. We run a deeper analysis, and finding that, there are 55 resources loading which are not logged as they are from the local file system. The time spent on local file loading is not affected by the network speed.

3.4.2 WordPress for Android [11.]

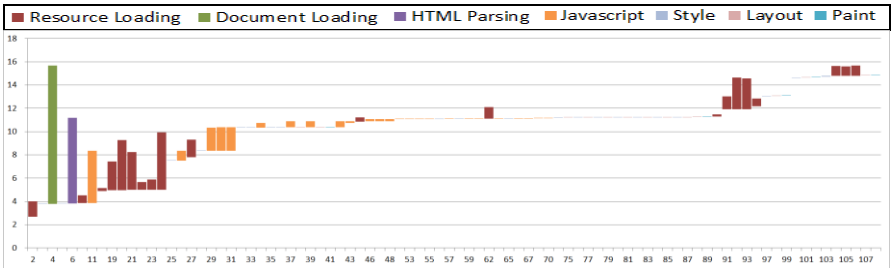
In this test, after the hybrid application is started, we tap “Start a new blog at WordPress.com” until the page is loaded and displayed properly. The chart and summary table are displayed in Figure 5 and Figure 6.

Similar to the result obtained from the Wikipedia application, style, layout and painting only spent a very little time, while most of the time is spent on resources loading. All the resources loaded by WordPress are coming from the Internet. As a result, it takes about 5 times longer to do resource loading.



	Resource Loading	Document Loading	HTML Parsing	Paint	Javascript	Style	Layout	Total
Count	20	1	1	3	25	18	11	6.208835
Sub Total Time (s)	4.885664	4.114629	2.565313	0.000592	6.174871	0.127028	0.065610	

Fig. 5 Broadband Setup, WordPress Mobile App



	Resource Loading	Document Loading	HTML Parsing	Paint	Javascript	Style	Layout	Total
Count	20	1	1	5	25	21	13	15.668970
Sub Total Time (s)	31.285869	11.872775	7.339598	0.031552	14.354112	0.262229	0.046567	

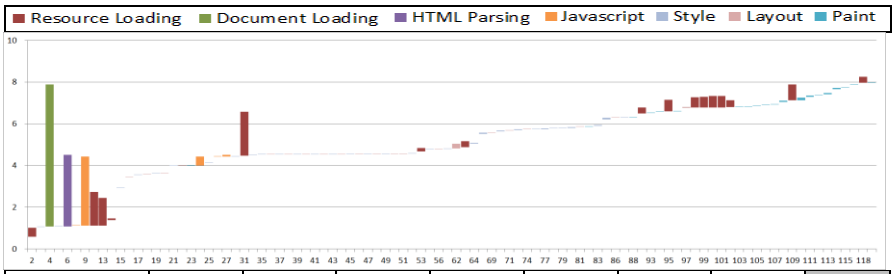
Fig. 6 3G Setup, WordPress Mobile App

3.4.3 jQuery Mobile Based Web App

Disneyworld has a typical mobile website [8.] chosen from the "jQuery Mobile Examples – JQM Gallery" [17.]. The charts and summary tables are displayed in Figure 7 and Figure 8.

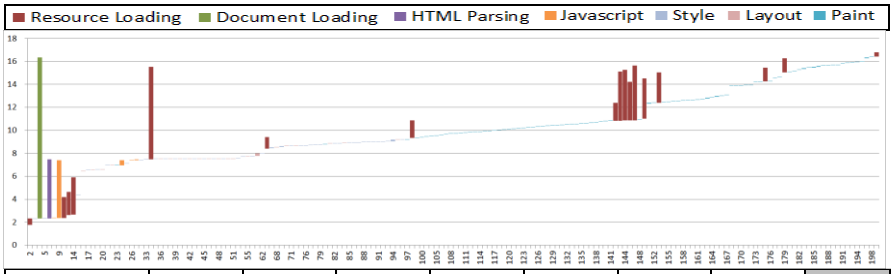
The sequence of beginning steps in 3G Setup is very similar to Broadband Setup. It is because they are loading the same page. However, the resources loading time is much larger, this is due to the network performance difference between business broadband and 3G mobile network.

Secondly, there are more painting operations in 3G Setup than in Broadband Setup, it is likely because the browser will carry out painting operation periodically in 3G Setup in order to refresh the screen with latest information.



	Resource Loading	Document Loading	HTML Parsing	Paint	Javascript	Style	Layout	Total
Count	17	1	1	20	5	32	20	8.255638
Sub Total Time (s)	10.309566	6.814091	3.432892	0.548695	3.848910	0.484607	0.252482	

Fig. 7 Broadband Setup, http://m.disneyworld.disney.go.com/



	Resource Loading	Document Loading	HTML Parsing	Paint	Javascript	Style	Layout	Total
Count	17	1	1	89	5	42	21	16.804664
Sub Total Time (s)	45.530785	14.026234	5.158642	1.901800	5.598895	0.495798	0.272073	

Fig. 8 3G Setup, http://m.disneyworld.disney.go.com/

4. Conclusions

In this paper, we described a new method to measure the performance of mobile web and hybrid applications without changing any application code. The method was implemented on Android platform and applied to 3 target mobile applications over WiFi and 3G networks to visually display the key operations and bottlenecks in those applications.

By comparing the performance of the 3 target systems on WiFi and 3G networks, we found that 3G network does increase the mobile application resource loading time dramatically which constitutes the major performance bottlenecks. But the bottlenecks can be improved by storing most of the resources locally. For example, *Wikipedia Mobile for Android* stores most of its resources locally while *WordPress for Android* and *jQueryMobile-based web application* get most of their

resources from the Internet, the former only spends 3 seconds longer on 3G than on WiFi while the latter doubles the loading time.

Acknowledgments This work is supported by Shannon Lab of Huawei Technologies Inc. We are grateful to all the reviewers for their constructive comments and suggestions.

5. References

1. J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl, "Anatomizing application performance differences on smartphones," in Proc. ACM/USENIX Int. Conf. Mobile Systems, Applications, and Services (MobiSys) San Francisco, California, USA: ACM, 2010.
2. 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.
3. Zhen Wang, Felix Xiaozhu Lin, Lin Zhong, and Mansoor Chishtie, "How far can client-only solutions go for mobile browser speed?" in Proc. the World Wide Web Conference (WWW), April 2012.
4. WebView, http://developer.android.com/reference/android/webkit/WebView.html
5. WebKit, http://en.wikipedia.org/wiki/WebKit
6. jQuery Mobile, http://jquerymobile.com/
7. Perf Feedback - What's slowing down Mobile Facebook , http://lists.w3.org/Archives/Public/public-coremob/2012Sep/0021.html
8. Disney Android App, http://m.disneyworld.disney.go.com/
9. PhoneGap, http://phonegap.com/
10. Wikipedia Android App, https://play.google.com/store/apps/details?id=org.wikipedia
11. WordPress Android App, https://play.google.com/store/apps/details?id=org.wordpress.android
12. How browsers work, Behind the scenes of modern web browsers, http://taligarsiel.com/Projects/howbrowserswork1.htm
13. Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5, http://techcrunch.com/2012/09/11/mark-zuckerberg-our-biggest-mistake-with-mobile-was-betting-too-much-on-html5/
14. JavaScriptCore, http://trac.webkit.org/wiki/JavaScriptCore
15. WebCoreRendering, http://trac.webkit.org/wiki/WebCoreRendering
16. 3GTest, http://www.eecs.umich.edu/3GTest/
17. JQM Gallery: jQuery Mobile Examples , http://www.jqmgallery.com/
18. Mobiperf, http://www.mobiperf.com/
19. Web Application, http://www.w3.org/TR/mwabp/#webapp-defined
20. CSS, Elaborate description of Stacking Contexts, http://www.w3.org/TR/CSS21/zindex.html