A Systematic Framework to Optimize Launch Times of Web Apps

Suresh Kumar Gudla, Jitendra Kumar Sahoo, Abhishek Singh, Joy Bose, Nazeer Ahamed
Samsung R&D Institute
Bangalore, India
suresh.gudla@samsung.com

ABSTRACT

Web Applications typically have longer launch times compared to native applications, especially upon device boot up or if the browser is not already running in the background. In this paper, we propose an approach to speed up the launch time for web applications, by considering the user's usage of web applications and pre-launching the predicted web applications. We provide implementation details of our model and perform experiments on various web applications to measure the effectiveness of the framework for fast launch of the applications after the device boots.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services.

Keywords

web apps; launch time; native apps; progressive web apps

1. INTRODUCTION

Native apps and web apps are two categories of apps available for smartphone users. Android internally takes care of native app speedup. Web apps, on the other hand, internally use Web Runtime (WRT) or the Web Browser, which slows them down in comparison to native apps. One way to speed up the web app launch time is to consider the user interest and previous web app usage. In this paper, we discuss such techniques to improve the launch time of the web apps by picking up the right set of web apps to pre-load and pre-render.

2. RELATED WORK

Liu et al [1] compared native and web apps performance using various parameters, and proposed some guidelines to improve the performance. Others [2-6] have proposed methods for predicting and prefetching for speedup of native applications. Our approach is geared for web apps and changing the web engine modules.

3. SYSTEM FRAMEWORK

Our overall architecture is shown in fig. 1. The system on the mobile device consists of a web app launcher, along with other modules to intelligently pre-fetch the web app and its content and thus reduce their launch time. The web app launcher module consists of a number of sub modules. The Listener module listens to system level events like Web App launch indications and Boot-Up broadcasts, and recreates all this information into simple key-value pairs in the form "attribute = value". The Rule-Developer module develops rules based on factors such as the usage of the web apps by the user. A rule consists of several predicates and logical operators.

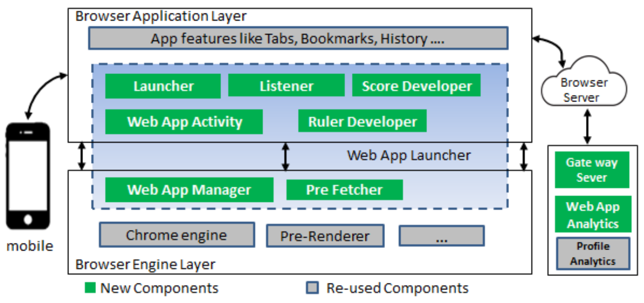

Figure 1. Architecture for the optimized web app framework.

The Score Developer module gives proportional weightage to various rules and generates the overall weighted score index of a web app, as shown in fig. 2. This kind of critical rule looks like

$$R_i = P_1 \cap P_2 \cap P_3 \dots \cap P_n \tag{2}$$

Where R_i is a critical rule identified containing one or more critical predicates. P_i can be a critical predicate or a sub-rule having critical predicates. To get the weighted score index of each web app, we choose the rules which are true and extract the predicates from them. Each of these predicates are given weights. We use the weighted sum model to calculate the weighted score index (WSI) of each web app, as presented in equation 3.

$$APP_i^{WSI} = \sum_{(j=1)}^{n} W_j P_{ij} \text{ for } i = 1,2 \dots m \tag{3}$$

Here, m represents the number of web apps on the device and P_{ij} represents the predicate used in evaluating for that web app, W_j the weight associated with that web app, APP_i is the i^{th} web app we are evaluating and APP_i^{WSI} is the WSI value of the i^{th} web app. After computing the WSI values for all installed web apps, the top few web apps with the highest WSI index values will be chosen for pre-fetching.

The Launch Manager takes the weighted score index of each Web App and decides which web app has to be given to the pre-fetcher module. The Pre-fetcher module is used to pre-render the corresponding Web App resources, scripts and html files and ensure that the actual page is ready for loading in a Tab of the

Browser. The Memory Pruner module plays a crucial role when the system goes down on memory in the case of web apps. The Statistics module stores relevant data corresponding to the web app usage and the user's preferences, and uses this data to generate a weighted score index. This weighed score index is used in successive device boot-up.

4. EXPERIMENTAL RESULTS

To test the system, we selected apps from a list of popular Android apps in India in a few categories. We picked NDTV from the Entertainment category, Twitter from the Social Network category, Times of India (TOI) from News category, VIA from the Travel category and Paytm from the Ecommerce category. We used Samsung Galaxy Note 5 devices for our test.

4.1 Launch Time Test

In our test, the launch time is measured using high speed cameras and measured from the launch of the app till the first screen appears with and without our framework. The graph in Fig. 3 shows above 40% improvement in the launch time of the web apps with our framework. This could be due to the fact that our framework takes care of pre fetching and pre rendering of the web apps, which serves an early composition without loss of data.

4.2 Network Connections Test

We measured the network requests made by web apps with and without our framework for the selected five apps. We took the TCP dump from the mobile device until the app is properly launched and initialized and counted the number of network requests made by the app manually using Wireshark Network Analyzer. Fig. 2 shows there is not much difference in the number of network requests made by the web apps with and without using our framework. This indicates there is no performance degradation because of our optimized framework.

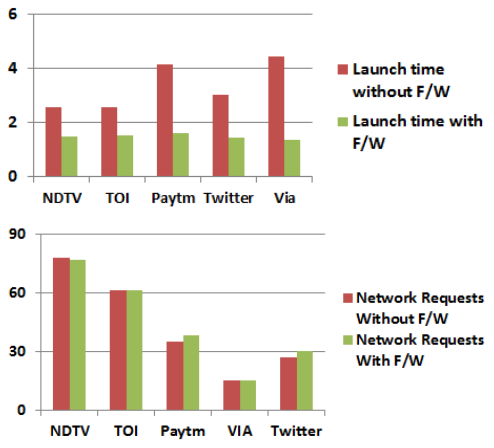

Figure 2. Plot of launch times (in sec) and network requests, without (red) and with (green) our framework.

4.3 CPU Test

The important launch process components executed before showing the web app home screen to the user are networking, loading resources, executing scripts, layout and rendering and user interaction. However the execution order of these modules differs for each of the apps. Some may prefer parallel processing and some may prefer sequential execution. In our experiments, we verified the CPU consumption of the selected apps during the launch time to evaluate the effectiveness of our framework during the launch time. We measured the CPU consumption of each of the web apps by connecting the test device to adb shell and executing the "top -m 5 -d 1 -n 10"shell command. We have taken the CPU evaluation values for the trail whose launch time is nearest to average launch time of web apps during Launch test for both with and without framework cases. The results of the tests with and without our framework are plotted in Fig. 3. The plotted graphs can be fitted with a Gaussian distribution.

Comparing the two graphs, we see that our framework could help in reducing the width of curve (representing the duration of the CPU utilization) to a significant extent and height of the curve (representing the peak of the CPU utilization) marginally.

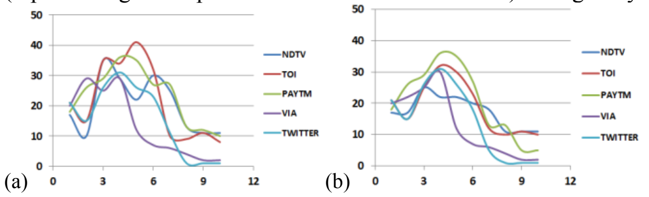

Figure 3. CPU utilization of the web apps in a trial (a) without and (b) with our framework.

5. CONCLUSION AND FUTURE WORK

In this paper we have presented the design of a system to reduce the web app loading time based on user preferences, and presented the results of various tests to show its performance. In future, we plan to generate the rules online based on the user behavior, further improve the model by incorporating online machine learning methods and also run the system for a larger dataset.

6. REFERENCES

[1] Yi Liu et al. Characterizing RESTFul Web Services Usage on Smartphones: A Tale of Native Apps and Web Apps. In ICWS, 2015.

[2] Tingxin Yan et. al. Fast app launching for mobile devices using predictive user context. In MobiSys, 2012.

[3] Ravindranath et al. Give in to Procrastination and Stop Prefetching. In Hotnets, 2013.

[4] Parate et al. Practical Prediction and Prefetch for Faster Access to Applications on Mobile phones. In UbiComp, 2013.

[5] R. Baeza-Yates et al. Predicting The Next App That You Are Going To Use. In WSDM, 2015

[6] C. Zhang et al. Nihao: a predictive smartphone application launcher. 2012. In MobiCASE, 2012.