Web Performance with Android's Battery-Saver Mode

Utkarsh Goel
Akamai Technologies, Inc.
ugoel@akamai.com

Stephen Ludin
Akamai Technologies, Inc.
sludin@akamai.com

Moritz Steiner
Akamai Technologies, Inc.
mosteine@akamai.com

ABSTRACT

A Web browser utilizes a device's CPU to parse HTML, build a Document Object Model, a Cascading Style Sheets Object Model, and render trees, and parse, compile, and execute computationally-heavy JavaScript. A powerful CPU is required to perform these tasks as quickly as possible and provide the user with a great experience. However, increased CPU performance comes with increased power consumption and reduced battery life on mobile devices. As an option to extend battery life, Android offers a battery-saver mode that when activated, turns off the power-hungry and faster processor cores and turns on the battery-conserving and slower processor cores on the device. The transition from using faster processor cores to using slower processor cores throttles the CPU clock speed on the device, and therefore impacts the webpage load process.

We utilize a large-scale data-set collected by a real user monitoring system of a major content delivery network to investigate the impact of Android's battery-saver mode on various mobile Web performance metrics. Our analysis suggests that users of select smartphones of Huawei and Sony experience a sudden or gradual degradation in Web performance when battery-saver mode is active. Battery-saver mode on newer flagship smartphones, however, does not impact the mobile Web performance. Finally, we encourage for new website design goals that treat slow (and throttled-CPU) devices kindly in favor of improving end-user experience and suggest that Web performance measurements should be aware of user device battery charge levels to correctly associate Web performance.

1 INTRODUCTION

In the last several years, mobile websites have grown drastically in both complexity and size [32, 33]. This growth has led to slower page loads and higher user frustration with the website [3]. As websites continue to grow in complexity, the key to improve website responsiveness is to build faster networks, optimize website content, and produce mobile devices with faster CPUs. While many networks have already begun to deploy new infrastructure and support faster communication protocols [28, 34, 38, 43], and most websites already employ a suite of website optimization techniques [35], the mobile device CPU remains a limiting factor to mobile Web performance [50]. Unlike desktop and laptop CPUs, Mobile CPUs are designed for power efficiency.

More specifically, Android smartphones have a battery-saver mode that lowers the battery consumption when the charge level drops below a certain threshold. Among other things, the battery-saver mode reduces the device's performance by throttling the CPU clock speed, where it deactivates the power-hungry and faster processor cores and activates the battery-saving and slower processor cores [36]. The process of loading Web pages includes HTML parsing, downloading and processing of JavaScript, Cascading Style Sheets (CSS), image resources, executing computationally-heavy JavaScript, and building the Document Object Model (DOM), CSS Object Model (CSSOM), and render tree [37]. Since all these tasks make use of the device's CPU resources, in this paper we seek to investigate whether or not a throttled CPU clock speed under an active battery-saver mode degrades the mobile Web performance for the end-user. Simply put, do mobile websites load slower on Android phones when the battery charge levels drop below a certain threshold?

Since the impact of battery-saver mode on mobile Web performance has not garnered much developer interest in the past, our goal with this paper is to bring awareness to the developer community about potential performance impacts and thus motivate the need for new website design goals and decisions that treat mobile devices differently, especially when battery-saver modes are active and the CPU clock speeds are throttled. We make the following contributions in the paper:

Dataset Richness: To investigate the impact of battery saver modes on mobile Web performance, we utilized a large-scale Web performance dataset collected by Akamai mPulse for websites loaded by real users on various mobile devices [13]. Our dataset contains various Web performance metrics collected for 10 million pages, loaded on 300 different smartphone models, connected to 81 cellular ISPs in 39 countries, from July 2017 to March 2018.

Inferences Drawn: Using a large-scale mobile Web performance dataset, we discover that under battery-saver mode, select phones from Huawei, Sony Xperia, and Samsung Galaxy series degrade mobile Web performance metrics, such as the page load time (PLT), time to first paint (TTFP), total LongTask time, time to interactive (TTI), and frame rate [2, 6, 19, 27]. The data also suggest that the battery-saver mode makes a higher impact on Web performance when Web pages load in faster mobile network conditions. The above findings demonstrate a clear need for new website design goals that would go hand-in-hand with the understanding of how user devices with low battery charge levels could degrade the mobile Web experience. Specifically, developers might want to build websites that adapt to different battery situations to overcome any user-perceived responsiveness issues inflicted by the battery-saver mode. Additionally, to reduce utilization of the throttled CPU, users may use browsers that offload CPU-intensive computations to the cloud [25, 29, 30].

The rest of the paper is organized as follows. Section 2 gives a background on Android's battery-saver modes. In Sections 3 and 4, we discuss our data collection methodology and the impact of battery-saver mode on various Web performance metrics. In Section 5, we discuss the related work. Finally, we conclude in Section 6.

2 BACKGROUND

The battery-saver mode, when activated, reduces the screen brightness, limits the use of WiFi, disables power-consuming location-sharing services, and reduces the application background activity. Android smartphones, such as LG G5, Huawei Y6 Elite, Alcatel Pixi 4, and many others, provide two user-configurable options for the battery charge level threshold at which the battery-saver mode turns on automatically [15, 16, 18, 47]. These threshold values are 5% and 15%. However, other Android smartphones, such LG G3 and LG VS985, provide four user-configurable options for this threshold, which are 10%, 20%, 30%, and 50% [31]. Similarly, Samsung phones, such as Galaxy S6, Galaxy J5, and others, also provide four user-configurable options for this threshold, which are 5%, 10%, 20%, and 50% [10, 20]. Note that even though there is an automated way to turn on the battery-saver mode every time the battery charge level drops below a certain threshold, the activation of this feature on the device depends on a user's interest in saving battery charge.

Other smartphones, such as Samsung Galaxy S8, Note 8, and Galaxy S5, do not provide a way to set a threshold, which means that the activation and deactivation of the power-saving modes must be done manually by users every time they want to save power [12, 17]. The activation of the power-saving modes on these devices may even be lower compared to the devices that offer a threshold for automatically activating the power-saving mode.

Sony's Xperia Z5 Compact has several power-saving modes, such as the Doze mode, that turns on automatically to save power when the device screen is turned off [14]. Other power-saving modes, such as the Stamina mode, allow users to set a battery charge level threshold at which power-saving mode activates. Moreover, the power-saving mode on Sony Xperia Z5 Compact also allows users to decide whether or not the CPU clock speed should be throttled, in addition to, or instead of, disabling mobile data and WiFi [22]. As such, some users may choose to activate power-saving mode without throttling the CPU clock speed, while others may choose to throttle the CPU clock speed.

3 DATA COLLECTION METHODOLOGY

To analyze the different performance metrics pertaining to websites loaded on various real user smartphones, we utilized the data collected by Akamai's mPulse product [13]. mPulse embeds a lightweight JavaScript snippet to some HTML responses and leverages the Web browser-exposed Navigation Timing API to collect performance-related information to estimate the time taken to load a page [2]. mPulse also utilizes the browser exposed Battery Status API to associate the measured website performance data with the device's battery charge level information at the time of measurement [5]. The collected data is reported back to Akamai servers [44], which we analyze to assess the impact of battery saver modes on page load performance. Our data consists of a total of 10 million page load transactions for 480 unique websites loaded on 300 different smartphone models connected to 81 cellular ISPs in 39 countries from July 2017 through March 2018.

To investigate the Web performance experienced on devices with low battery charge levels, we calculate the median page load time (PLT) observed for page loads pertaining to 6-field buckets, comprised of: 1) the country name in which the page was loaded, 2) the ISP over which the page was loaded (note that the first two pieces of information are deduced from the MaxMind's database of mapping IP addresses to geographical locations [40]), 3) the URL of the website loaded, 4) the smartphone model used to load the website, 5) the HTTP protocol version (HTTP/1.1 or HTTP/2) used, and 6) the device battery percentage, ranging from 1 to 100%. Note that we refer to PLT as the time from the start of page navigation until the `loadEventStart` event is triggered by the Web browser [2]. Also note that bucketing the data with this approach helps to precisely understand how fast a given website loads on a given ISP network under specific constraints, such as the user's country, user's ISP, website URL, the smartphone, and the HTTP protocol, and thus mitigate the influence of many factors that might affect the analysis of Web performance.

In addition to calculating the median PLT for beacons in each bucket, we calculate the 10th, 25th, 75th, and 90th percentile PLT values. By calculating these percentile values for each of the 6-field bucket, we could assume that the 10th percentile PLT tends to represent page loads in fast cellular networks, and that the 90th percentile PLT tends to represent page loads in slower cellular networks. This categorization would help us understand how the impact of battery-saver mode changes as network performance improves or degrades.

Finally, similarly to how we calculate PLT distributions, we also calculate the TTFP observed for loading different websites on different smartphones. Additionally, we collect the total LongTask time, the time to interactive, and the frame rate observed when loading websites under various battery charge levels. Note that since the LongTask, time to interactive, and frame rate metrics depend on the device hardware, as opposed to the network performance, for these metrics we do not calculate 10th, 25th, 50th, 75th, and 90th percentile values (because we represent percentile values as network speed) but instead plot the whole distributions in appropriate figures.

Note that the dataset we prepared consists of 25 unique combinations (comprising of country, ISP, URL, smartphone name, and HTTP protocol) for which mPulse library was executed on at least 100 page loads for each of the battery charge levels ranging from 1% to 100% - a total of at least 10,000 beacons for each combination. Additionally, the dataset consists of 63 unique combinations for which mPulse library was executed on at least 100 page-load transactions, for at least 90 battery charge levels - a total of at least 9,000 beacons for each combination. The low number of combinations observed in the dataset is a consequence of the fact that only about 2.5% of the total page loads occurred when battery charge levels were below 15%, which yielded less than 100 beacons for some battery charge levels on some combinations. Additionally, the low number of page loads under battery charge levels less than 15% may suggest that either most users keep their phones charged over 15% at all times or that when battery charge levels drop below 15%, users reduce their Web browsing activities in favor of conserving battery charge.

4 MEASURING WEBSITE PERFORMANCE

To estimate the performance of a website, we analyze several Web performance metrics under different device battery charge levels, such as PLT and TTFP – the time since the start of the page navigation until the browser paints the first pixels. We also analyze the total LongTask time – the time during which the browser main/UI thread is blocked and therefore, the user cannot interact with the page [24]. Finally, we measure the average rate of printing frames on the screen [27].

4.1 Measuring Page Load Time (PLT)

4.1.1 Performance on Huawei Y6 Elite. In Figure 1, we show various percentile PLT values for a page loaded on Huawei Y6 Elite smartphone. From the figure we can observe that across all percentiles, the PLT inflates as soon as the battery charge level drops to 15%, likely due to degraded CPU clock speeds triggered by the battery-saver mode. The figure also suggests that, on most Huawei Y6 Elite smartphones, the default threshold for when the battery-saver mode initiates is set to 15%.

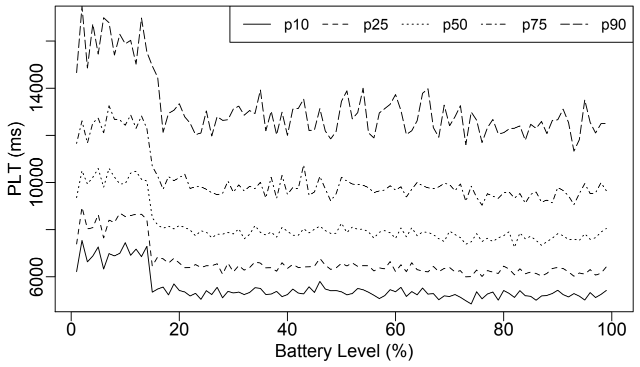

Figure 1: PLT distributions across different device battery charge levels, as measured for a page loaded on Huawei Y6 Elite mobile device.

PLT distribution	PLT when battery charge level = 8%	PLT when battery charge level = 50%	Relative difference in PLT (%)
p10 (fast network)	6.8 seconds	5.3 seconds	28.3%
p25	8.3 seconds	6.5 seconds	27.6%
p50	10.2 seconds	8.2 seconds	24.3%
p75	12.6 seconds	10.2 seconds	23.5%
p90 (slow network)	15.4 seconds	13.4 seconds	14.9%

Table 1: Summary of PLT distributions on Huawei Y6 Elite phone.

In the Table 1, we summarize the analysis from Figure 1 to compare the PLT observed when battery charge levels are, for example, 8% and 50%. Note that the battery charge levels 8% and 50% are just two example representative points for comparing the performance under the battery-saver and normal modes.

From the table we can observe that the relative difference in the page load decreases as the network performance degrades. Specifically, among the two battery levels, 8% and 50%, for page loads represented by the 10th percentile distribution (p10), the PLTs at 8% are higher by 28% compared to the PLTs at 50%. Whereas, for the 90th percentile distribution (p90), the PLTs at 8% are higher by 15% compared to the PLTs at 50%. This downward trend suggests that when websites are loaded on faster networks, the smartphone's CPU becomes a more significant bottleneck for website performance. Indeed this trend is similar to a previous research study that investigates how device performance impacts Web performance [50].

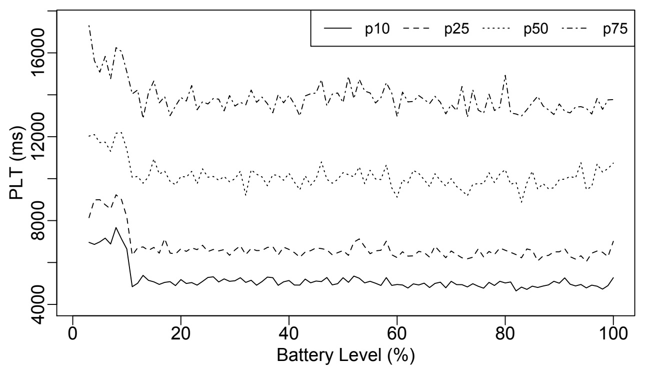

Figure 2: PLT distributions across different device battery charge levels, as measured for a page loaded on Huawei P8 Lite (2015) mobile device.

PLT distribution	PLT when battery charge level = 8%	PLT when battery charge level = 50%	Relative difference in PLT (%)
p10 (fast network)	7.6 seconds	5.2 seconds	45%
p25	9.2 seconds	6.5 seconds	40%
p50	12.1 seconds	10.2 seconds	18%
p75 (slow network)	16.2 seconds	13.6 seconds	19%

Figure 3: Summary of PLT distributions on Huawei P8 Lite phone.

Both PLT and TTFP metrics make direct impact on the user experience. As such, an increase in these metrics, due to degraded CPU clock speeds, represents a degradation in the user experience.

4.1.2 Performance on Huawei P8 Lite (2015 model). Similarly to Figure 1, as shown in Figure 2, we observe a sudden inflation in PLTs when loading a Web page on the 2015 model of Huawei P8 Lite smartphone. Since the inflation occurs when the battery charge level is at 10%, it appears that the battery-saver mode on the Huawei P8 Lite smartphone turns on for most users at 10%.

Additionally, as shown in Table 3, when we compare the relative PLT differences across different distributions, we can observe that the impact of throttled CPU clock speeds on PLT appears to be more significant when pages are loaded in fast network conditions.

We observed similar trends in performance for other websites that loaded on Huawei P8 Lite smartphone, however, we do not show them due to page limits. Additionally, note that for websites loaded on the 2017 model of Huawei P8 Lite smartphone, we did not observe sudden inflation in PLT or TTFP metrics at any battery charge level, potentially because the 2017 model has a faster CPU that does not impact the page load despite the drop in CPU performance [8]. Finally, we noticed that newer smartphones from the same family,

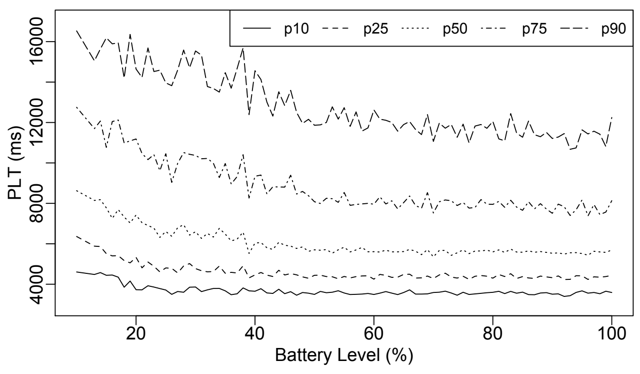

Figure 4: Page load time distributions across different device battery charge levels, as measured for a page loaded on Sony Xperia Z5 Compact mobile device.

such as Huawei P9 Lite and P10 Lite, also do not degrade website performance under low battery charge levels.

4.1.3 Performance on Sony Xperia Z5 Compact. In Figure 4 we show that when pages load on Sony Xperia Z5 Compact smartphone, there exists an upward slope indicating that page load times continue to rise as battery charge levels drop below 40%. We have not been able to identify the cause for such an upward slope, as opposed to a sudden increased in page load time, even though this device allows users to set a threshold at which the battery-saver mode activates. Perhaps, this device has a linear slowdown in the CPU clock speed when the battery charge levels reach 40%. Note that we did not observe such a strong upward slope for other Sony devices, such as Xperia X Compact, Xperia X Performance, and Xperia XZ and in fact, there was no inflation in PLT for these devices under low battery charge levels.

4.1.4 Performance on High-end Phones. We compare the website performance for pages loaded on various new and old Samsung flagship smartphones under different battery charge levels. Note that for the purposes of comparing performance across different Samsung devices, in Figure 5 we only show the PLT values calculated for the 50th percentile distribution at various battery charge levels. Additionally, note that for making the comparison easier to understand, in Figure 5, we compare the performance observed for Huawei Y6 Elite smartphone for the same website as the one used in Figure 1.

From Figure 5, we observe that when the website is loaded on newer devices with high-performance CPU, the PLT gets lower across all battery charge levels. For example, the PLTs on Note 8 are about 4 seconds, whereas the PLTs on Galaxy S6 are about 5.5 seconds. Similarly, PLTs on Galaxy S5 are about 6.5 seconds, and the PLTs on Y6 Elite under non-throttled CPU hardware are about 8 seconds. The performance differences across devices show that the

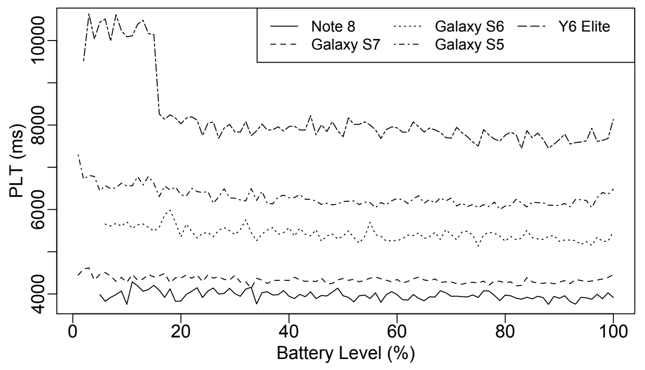

Figure 5: PLT distributions across different device battery charge levels, as measured for a page loaded on various Samsung smartphones and Huawei Y6 Elite.

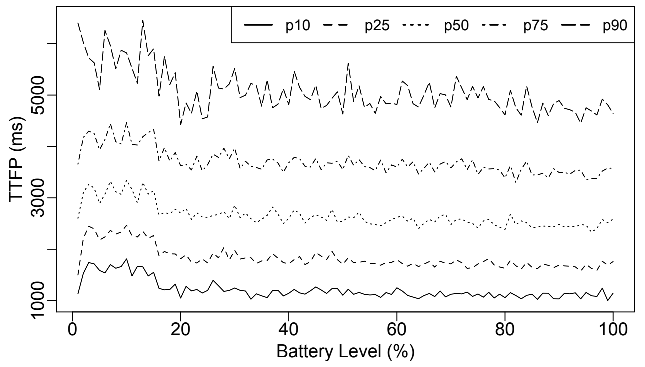

Figure 6: TTFP distributions across different device battery charge levels, as measured for a page loaded on Huawei Y6 Elite mobile device.

website performance can be improved by upgrading the device hardware, regardless of the battery charge level.

Additionally, from Figure 5 we observe that PLTs on Note 8 do not experience sudden or gradual increase at any battery charge level. Since Samsung Note 8 does not provide a default user configurable option to enable the battery-saver mode when the battery charge level reaches a certain threshold and that users must manually activate the battery-saver mode, perhaps most users tend to not activate the battery-saver mode and therefore we do not observe any inflation in PLTs. Alternatively, since Galaxy Note 8 has octa-core processors built-in, perhaps even when the battery-saver mode is active, the throttled CPU clock speed is high enough to not negatively impact the page load process or the PLT [11]. However, we acknowledge that a better understanding of the inner workings of the device would help bear this out. For other devices, such as Galaxy S7 and S6, we do not observe any inflation in PLT regardless of the battery charge level. Similarly to Note 8, perhaps the throttled CPU clock speeds on these devices are high enough to not impact the PLT.

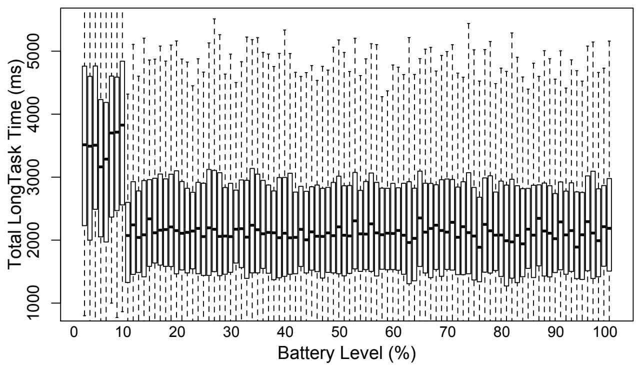

Figure 7: LongTask time distributions across different device battery charge levels, as measured for a page loaded on P8 Lite (2015) smartphone.

4.2 Measuring TTFP

In Figure 6, we show various percentile TTFP values for the same page as Figure 1, when loaded on Huawei Y6 Elite smartphone. From the figure we can observe that there exists a sudden inflation in the TTFP when the battery charge levels drop below 15%. This trend suggests that the time when the browser paints the first pixels also increases when the device enables the battery-saver mode. Similarly to Figure 1, we also noticed sudden inflation in TTFP at battery charge level 15%, when loading websites on the 2015 model of Huawei P8 Lite smartphone.

4.3 Measuring LongTask Time

LongTask is a relatively new Web performance measurement API that allows identification of resources that make websites unresponsive to user interactions [19]. More specifically, Web developers could use the LongTask API to detect the presence of tasks that block the browser UI/main thread for at least 50 milliseconds. When a website loads resources that block the browser UI/main thread, the user is unable to interact with the page. Specifically, a long task prevents the page from responding to user actions, such as scroll, click, tap, key, wheel, etc, until the long task has finished executing. This is because when a long task is executing, all user actions are queued behind the long task.

Poorly designed JavaScript code is one example of what might cause a browser main thread to block for over 50ms [46]. Since the current LongTask API (v1) does not reveal the URL of the long task (though the second version of the API will provide such detail, including the line number that caused the long task [45]), it is unclear as to what particular resources block the browser main thread. Therefore, we only focus on investigating whether or not we observe a rise in the total LongTask time when device CPU clock speed reduces when battery charge levels drop below a certain threshold, as opposed to discussing the root

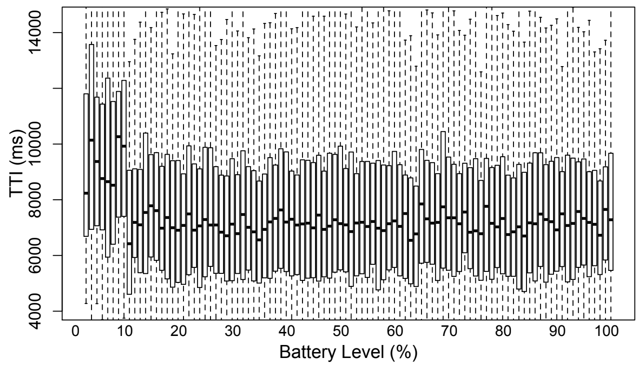

Figure 8: TTI distributions across different device battery charge levels, as measured for a page loaded on P8 Lite (2015) smartphone.

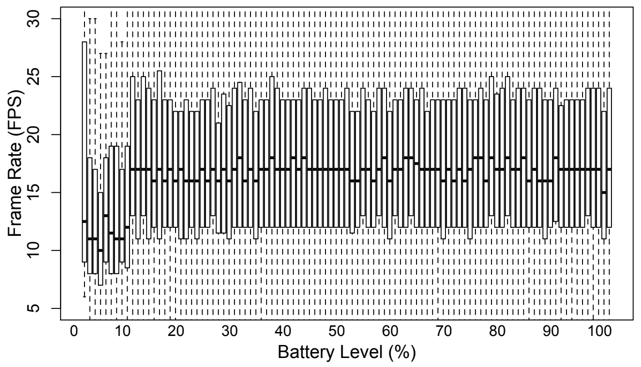

Figure 9: FPS distributions across different device battery charge levels, as measured for a page loaded on P8 Lite (2015) smartphone.

cause of a large or small total LongTask time. Also note that since LongTask time, time to interactive, and the frame rate metrics do not depend on network speed and instead depend on the device CPU performance, we use boxplot distributions in Figures 7-9 instead of plotting different percentile values as we did for previous graphs.

In Figure 7, we show the boxplot distributions of the total LongTask time across different battery charge levels when loading a website on the 2015 model of the Huawei P8 Lite smartphone. From the figure we can observe that the total LongTask time inflates when device battery charge level drops below 10%. The rise in the total LongTask time indicates that when the device CPU clock speed is throttled to minimize battery consumption, LongTasks block the main thread for longer than usual time. Note that we did not observe inflation in the total LongTask time on the 2017 model of Huawei P8 Lite smartphone, likely due to the fact that faster processors on the device do not impact the total LongTask time when their speeds are throttled by the battery-saver mode.

4.4 Measuring Time to Interactive (TTI)

Not only does a LongTask delay user interactions, but events callbacks (such as onLoad) are also delayed. In many pages where many LongTask exists as the page loads, the time at which the user could first interact with the page could also get delayed. Additionally, note that even though LongTask is the prime cause for poor responsiveness to user interactions, other tasks, such as image decoding, heavy rasterization work, or presence of many layers on the page can also cause poor responsiveness [39].

To investigate whether there is any impact of battery-saver mode on the time when the user could first interact with the website, we take a look at the time to interactive (TTI) metric. The TTI metric is calculated based on when the page was visually ready for the user and when the page was ready for interaction [6]. Specifically, the former is calculated by calculating the maximum of time to first paint and time to `domContentLoadedEventEnd` event [21]. Once the time to visually ready is calculated, the first time period of 500 milliseconds during which the browser UI/main thread was idle marks the TTI for the page.

In Figure 8, we show the boxplot distributions for TTI values observed for loading a website on Huawei P8 Lite smartphone under different battery charge levels. From the figure we can observe that the TTI values inflate as soon as the battery charge levels drop below 10%. The sudden rise in TTI values indicates that users on this smartphone, and other similar smartphones, likely experience janks when interacting with the website, thus leading to poor user experience.

4.5 Measuring Frame Rate per Second

The new `requestAnimationFrame` API allows Web developers to strategically schedule paint events on the screen with the goal of achieving a high frame rate during the website load [26]. Typically, 60 frames per second (FPS) is considered ideal for good user experience, which means that the browser has exactly 16.6 ms to produce a frame. However, if the browser needs to perform tasks that delay the frame generation, the FPS declines. Note that a low frame rate can degrade the user experience, because under low FPS the page becomes unresponsive to user interactions. Therefore, we investigate whether or not the battery-saver mode impacts the frame rate observed across different page loads. Using the mPulse Continuity plugin we gathered the frame rate observed under various battery charge levels [6]. Note that we show analysis of frame rates only for one of the devices for which we observe a sudden rise in both the PLT and TTFP.

As shown in Figure 9, the FPS observed on the 2015 model of Huawei P8 Lite drops as soon as the device battery charge levels drop below 10%. This indicates that for websites loading on this device, as the CPU performance degrades, the rate at which the browser paints on the screen also declines - leading to a potentially poor user experience.

5 RELATED WORK

Several tools and studies have relied on active measurements to identify Web performance bottlenecks [1, 7, 23, 51]. Unlike passive measurements via real user monitoring systems [9, 13], active measurement practices inherit several limitations because pages loaded in controlled environments do not represent the characteristics of how pages load in the real world [41]. Steiner *et al.* used a real user monitoring system to compare the impact of CPU processors embedded in old and new smartphones on the web performance – suggesting that page loads on new devices with fast CPUs are significantly faster than page loads on old devices with slow CPU [50]. Like other studies [42], the authors observed that faster CPUs on newer smartphones load webpages faster than those on old phones. Another study reveals that about 35% of the PLT is spent performing CPU-intensive tasks on user devices [51].

6 DISCUSSION AND CONCLUSION

Slow mobile device hardware is a bottleneck to mobile Web performance. We perform a large-scale measurement study to identify the impact of Android's battery-saver (which throttles the CPU clock speed) mode on mobile Web performance. Our data suggests that under low-battery conditions, sudden rises in page load time, total LongTask time, and time to interactive metrics are observed on some devices. The average frame rate on some smartphones also declines, leading to unresponsive and paint-blocked websites.

Through this paper we hope we motivate the need for new website design goals that improve mobile Web experience for slow mobile devices. The Web performance community has developed numerous best practices for developers to deliver high-performance experiences to end users [4, 25, 29, 30, 35, 48, 49].

DISCLOSURE

The positions, strategies, or opinions reflected in this article are those of the authors and do not necessarily represent the positions, strategies, or opinions of Akamai.

REFERENCES

[1] 2009. Gomez Last-Mile Testbed. https://goo.gl/BtwSWY. (Nov. 2009).
[2] 2015. Navigation Timing. http://w3c.github.io/navigation-timing/. (Aug. 2015).
[3] 2016. Mobile Trends in Retail. https://www.kount.com/LiteratureRetrieve.aspx?ID=232433. (Mar. 2016).
[4] 2018. Accelerated Mobile Pages. https://www.ampproject.org/learn/overview/. (Apr. 2018).
[5] 2018. Battery Status API. https://developer.mozilla.org/en-US/docs/Web/API/Battery_Status_API. (Apr. 2018).
[6] 2018. BOOMR.plugins. Continuity. https://akamai.github.io/boomerang/BOOMR.plugins.Continuity.html. (Apr. 2018).
[7] 2018. Cathpoint Synthetic Monitoring. http://www.catchpoint.com/synthetic-monitoring/. (Apr. 2018).
[8] 2018. Compare Specs. https://www.gsmarena.com/compare.php3?idPhone1=7201&idPhone2=8516. (Apr. 2018).
[9] 2018. Dynatrace Real user monitoring (RUM). https://www.dynatrace.com/capabilities/real-user-monitoring/. (Apr. 2018).
[10] 2018. Extend battery life - Samsung Galaxy J5. https://www.helpforsmartphone.com/public/en/samsung/galaxy-j5/android-5-1/guides/14/Extend-battery-life-Samsung-Galaxy-J5. (Apr. 2018).
[11] 2018. Galaxy Note 8 Specs. https://www.gsmarena.com/samsung_galaxy_note8-8505.php. (Apr. 2018).
[12] 2018. How do I use Power saving mode on my Samsung Galaxy S5? https://www.samsung.com/uk/support/mobile-devices/how-do-i-use-power-saving-mode-on-my-samsung-galaxy-s5/. (Apr. 2018).
[13] 2018. How mPulse works. https://learn.akamai.com/en-us/webhelp/mpulse/mpulse-help/GUID-EBEC9222-7876-46F9-81A8-2227CFA89851.html. (Apr. 2018).
[14] 2018. How To Boost Your Battery. https://support.sonymobile.com/us/xperiaz5compact/dm/battery/. (Apr. 2018).
[15] 2018. How to exten battery life on HUAWEI Y6 Elite? https://www.hardreset.info/devices/huawei/huawei-y6-elite/faq/tips-tricks/how-to-save-battery-life-on-android-phone/. (Apr. 2018).
[16] 2018. How to Turn Battery Saver Mode On and Off on Alcatel Pixi 4. https://support.bell.ca/Mobility/Smartphones_and_mobile_internet/Alcatel-Pixi-4.how_to_turn_battery_saver_mode_on_and_off_on_my. (Apr. 2018).
[17] 2018. How to Turn Power Saving Mode On and Off on Samsung Galaxy Note8. https://support.bell.ca/Mobility/Smartphones_and_mobile_internet/Samsung-Galaxy-Note-8.how_to_turn_power_saving_mode_on_and_off_on_my. (Apr. 2018).
[18] 2018. LG G5 Using Power Saving Mode. https://videotron.tmtx.ca/en/topic/lg_g5/using_power_saving_mode.html#step=4. (Apr. 2018).
[19] 2018. Long Tasks API 1. https://w3c.github.io/longtasks/. (Aug. 2018).
[20] 2018. Monitor and Extend Battery Life on Galaxy S6. https://www.samsung.com/us/support/answer/ANS00066391/. (Apr. 2018).
[21] 2018. PerformanceTiming.domContentLoadedEventEnd. https://developer.mozilla.org/en-US/docs/Web/API/PerformanceTiming/domContentLoadedEventEnd. (Apr. 2018).
[22] 2018. Sony Xperia Z5 and Z5 Compact Tips and Tricks. https://www.youtube.com/watch?v=T1BZNsNJBo4&feature=youtu.be&t=71. (Apr. 2018).
[23] 2018. Test a website's performance. https://www.webpagetest.org/. (Apr. 2018).
[24] 2018. Threading and Tasks in Chrome. https://chromium.googlesource.com/chromium/src/+/lkgr/docs/threading_and_tasks.md. (Apr. 2018).
[25] 2018. What Is Amazon Silk? https://docs.aws.amazon.com/silk/latest/developerguide/introduction.html. (Sept. 2018).
[26] 2018. window.requestAnimationFrame(). https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame. (Apr. 2018).
[27] K. Basques. 2018. Analyze Frames Per Second. https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/#analyze_frames_per_second. (Apr. 2018).
[28] M. Belshe, R. Peon, and Ed. M. Thomson. 2015. Hypertext Transfer Protocol Version 2 (HTTP/2). https://tools.ietf.org/html/rfc6146. (May. 2015).

[29] Tiffany Brown. 2012. Opera Mini and JavaScript. https://dev.opera.com/articles/opera-mini-and-javascript/. (Sept. 2012).

[30] Liu Chi. 2018. How the Puffin Browser Works. https://medium.com/coding-neutrino-blog/how-the-puffin-browser-works-440c91cece8f. (Sept. 2018).

[31] R. Devine. 2014. How to use battery saver on the LG G3. https://www.androidcentral.com/how-use-battery-saver-lg-g3. (Jul. 2014).

[32] Tammy Everts. 2013. The average web page has almost doubled in size since 2010. http://www.webperformancetoday.com/2013/06/05/web-page-growth-2010-2013/. (Jun. 2013).

[33] Tammy Everts. 2017. The average web page is 3MB. How much should we care? https://speedcurve.com/blog/web-performance-page-bloat/. (Aug. 2017).

[34] Utkarsh Goel, Moritz Steiner, Mike P. Wittie, Martin Flack, and Stephen Ludin. 2016. A Case for Faster Mobile Web in Cellular IPv6 Networks. In *ACM MobiCom*.

[35] Ilya Grigorik. 2013. *High Performance Browser Networking*. O'Reilly Media, Inc.

[36] I. Grigorik. 2016. Fast and resilient web apps: Tools and techniques - Google I/O 2016. https://www.youtube.com/watch?v=aqvz5Oqs238&feature=youtu.be&t=19m45s. (May. 2016).

[37] I. Grigorik. 2018. Render-tree Construction, Layout, and Paint. https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction. (Apr. 2018).

[38] J. Iyengar and M. Thomson. 2018. QUIC: A UDP-Based Multiplexed and Secure Transport. https://www.ietf.org/id/draft-ietf-quic-transport-14.txt. (Aug. 2018).

[39] Martina K. 2016. Software vs. GPU rasterization in Chromium. https://software.intel.com/en-us/articles/software-vs-gpu-rasterization-in-chromium. (Feb. 2016).

[40] MaxMind. 2018. Detect Online Fraud and Locate Online Visitors. https://www.maxmind.com/en/home. (Apr. 2018).

[41] P. Meenan. 2013. How Fast is Your Web Site?. In *ACM Queue, Volume 11, issue 2*.

[42] Ashkan Nikravesh, Hongyi Yao, Shichang Xu, David Choffnes, and Z. Morley Mao. 2015. Mobilyzer: An Open Platform for Controllable Mobile Network Measurements. In *ACM MobiSys*.

[43] Erik Nygren. 2015. Three years since World IPv6 Launch: strong IPv6 growth continues. https://blogs.akamai.com/2015/06/three-years-since-world-ipv6-launch-strong-ipv6-growth-continues.html. (Jun. 2015).

[44] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. 2010. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.* 44, 3 (Aug. 2010).

[45] S. Panicker. 2016. Long Tasks API v2. https://docs.google.com/document/d/125d69JAC7nyx-Ob0a9Z31d1uHUGu4myYQ3os9EnGfdU/edit. (Sept. 2016).

[46] S. Panicker and N. Jansma. 2017. Reliably Measuring Responsiveness in the Wild. https://www.youtube.com/watch?v=y5qPix1tdOE. (Jul. 2017).

[47] S. Rutherford. 2015. How to Turn on Battery Saver Mode in Android 5.0. https://www.tomsguide.com/us/battery-saver-mode-android-5,news-20491.html. (Mar. 2015).

[48] S. Souders. 2008. *High Performance Web Sites: Essential Knowledge for Front-End Engineers*. O'Reilly Media, Inc.

[49] S. Souders. 2009. *Even Faster Web Sites: Performance Best Practices for Web Developers*. O'Reilly Media, Inc.

[50] Moritz Steiner and Ruomei Gao. 2016. What slows you down? Your network or your device?. In *arXiv:1603.02293*.

[51] Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall. 2013. Demystifying Page Load Performance with WProf. In *USENIX NSDI*.