On the I/O Characteristics of the Mobile Web Browsers

Taeho Hwang Myungsik Kim* Seongjin Lee** Youjip Won
Hanyang University, Seoul, Korea
Samsung Electronics, Suwon, Korea*
Gyeongsang National University, Gyeongsang, Korea**

ABSTRACT

Recent advancement of mobile device technologies played a key role in increasing the user base, and a large number of people are using the device to gather information and browse the Internet. Web browsers are in the center of such activities. In this paper, we analyze the I/O characteristics of five popular web browsers: Chrome, Firefox, Opera, Dolphin, and UC. We chose a website that has 52 objects and 509KB as the sum of all the objects. After analyzing the ratio of read and write I/Os, we found that browsing is a write-intensive activity. Three of the browsers issued more than 50% of I/Os using `fsync()` and `fdatasync()`. About 50% of I/Os are for updating file system metadata and journal. The number of I/Os that a browser generates for SQLite and cache files is more than 50% at the lowest and 90% at the highest of the total I/O counts. In this paper, we define write amplification of web browsers to address the amount of extra I/Os browsers generate. We found that web browsers write on average of 11.7 times more data to the storage than the sum of original web contents.

CCS CONCEPTS

•**Information systems** → *Browsers;*

KEYWORDS

I/O Characteristics, Application to Storage Write Amplification Factor, Browsers

ACM Reference format:
Taeho Hwang Myungsik Kim* Seongjin Lee** Youjip Won. 2018. On the I/O Characteristics of the Mobile Web Browsers. In *Proceedings of ACM SAC Conference, Pau,France, April 9-13, 2018 (SAC'18),* 3 pages.
DOI: 10.1145/3167132.3167402

1 INTRODUCTION

As the computing power of mobile devices is increasing and the availability of mobile data network is widening, more people are accessing the web through mobile devices and web browsers are successfully doing its job as an interface between a user and information. Statistics show that the number of mobile web accesses has surpassed the fixed Internet since 2013.1.2 Billion people access websites using mobile web browsers, and about 25% of US users access the Internet through mobile devices.

As network bandwidth increases, the richness of contents also increases both in a number of objects and level of interactiveness. According to the HTTP Archive Project, which aggregates analytic data of web resources, the average size of top hundred websites increased from 1.3 MB on May of 2015 to 1.8 MB in August 2016. Pages with non-optimized images, third party scripts, and interactive contents contribute to the size. As a result, the web browser is becoming an I/O intensive application. In an effort to enhance the performance of web browser, people have come up with various evaluation metrics such as the size of web contents (HTML, Image, and Scripts), network transfer rate, and rendering time which includes parsing the HTML structure. Some also include non-functional features such as intuitive user interface, compatibility with HTML standards in evaluating the web browsers. However, we believe that the metrics cannot successfully capture browser's effect on the I/O stack. As Kim *et al.* have pointed out browsing in the mobile web browser is not only a bounded network but also I/O intensive job [4]. It is also important to understand that naive choice in one I/O layer may cause amplification on other I/O layer and overall I/O performance of the mobile device [3]. Understanding these facts is important, especially in the mobile environment because mobile devices have limited computing power and system resources , and more importantly, it reduces the lifetime of an eMMC storage device.

In this paper, we captured the I/O trace of five well-known mobile web browsers, Chrome, Firefox, Opera, Dolphin, and UC browser to analyze the I/O behaviors using a website that has 52 objects and 509KB as the sum of all the objects. Some of the interesting observations are as follows:

- **Browsers have a high write amplification of at least 6.5:** We defined the browser write amplification factor (B-WAF) as the total amount of data that a browser stores over the sum of original web content. We found that B-WAF of all browsers is very high—the lowest of all is 6.5 and the average is 11.7. The average B-WAF of top 100 sites from Alexa on the five browsers (Chrome, Firefox, Opera, Dolphin, and UC) are 2.6, 5.9, 6.2, 3.9, and 3.9, respectively, and the maximum B-WAF can go up to 24 times larger than the average.
- **Journal mode in Database is not optimized:** PERSIST SQLite journal mode–default in Nexus 5–generates the most I/Os to the storage [3]. We found that all browsers, with a few exceptions, are using PERSIST mode while accessing the databases.
- **Browser features shorten the eMMC lifetime:** Some of the browser features, some of which are not user configurable, generate excessive write I/Os to the storage. For example, Google analytics on Opera is always on and cannot be disabled. Firefox checks for malware or phishing sites every time it visits a website. Dolphin keeps many

SAC'18, Pau,France

DOI: 10.1145/3167132.3167402

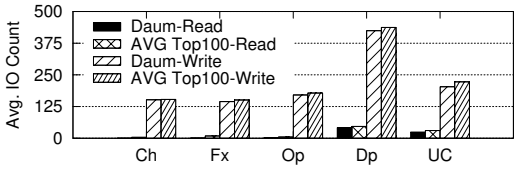


Figure 1: I/O Count in Mobile Browsers

databases to cache the user data. Such features acts adversely to the eMMC lifetime.

2 EXPERIMENT PROCEDURE AND THE WORKLOAD

We used five well-known web browsers (Chrome, Firefox, Opera, Dolphin, and UC), Nexus 5 (Hammerhead) Smartphone and Androtrace [5] to collect the trace data from the smartphone and understand the I/O behavior of five web browsers. We used a high-speed WiFi network to visit *www.daum.net* and top 100 sites from Alexa, the web data traffic information provider, and we acquired I/O traces from each browser. According to Alexa [1], *www.daum.net* ranked fourth popular website in Korea and 92^{nd} among the top 100 sites. We used `input` command in ADB shell [2] to automate the experiment. At the beginning of the experiment, cache directory in the corresponding browser directory is deleted to prevent reusing the objects such as images, cookie, and JavaScript files. After opening the browser, we paused for 3 seconds. There is a 1-second gap between typing in the URL and opening the location. The browser is closed after 30 seconds. We used Mobile Browser Developer mode in Chrome for PC to analyze the website contents. We post-processed acquired I/O trace to categorize the files appeared in the trace. There are 52 objects on *www.daum.net*, and the total sum of all the objects is 509 KB. Three files out of 52 files are HTML, 44 files are images, and 5 files are java and json scripts. The sum of HTML file sizes, images, and script files are 38 KB, 443 KB, and 29 KB, respectively. The top 100 sites uses about 11 different types of objects on the average.

3 I/O CHARACTERISTICS

In this section, we presented primitive I/O statistics for top 100 sites and the *www.daum.net*. Since the space is limited, we focused on the result of *www.daum.net*, and made comment on the result of top 100 sites when macroscopic view of the result is necessary.

Primitive I/O Statistics We examined the ratio of 4 KB writes, synchronous writes, `fsync()` and `fdatasync()` calls made, metadata and journal, and SQLite related for the *www.daum.net* and the top 100 sites. In terms of the I/O count, 4 KB writes are dominant in all five browsers; more than 60% of all issued I/Os are 4 KB writes. In the examination, synchronous writes with `fsync()` or `fdatasync()` are common in browsers. For example, synchronous writes in Chrome, Opera, and UC are 57%, 56%, and 35% of all write I/Os, which results in 68%, 70%, and 74% of all write volume, respectively. Since synchronous write forces file system to write metadata and journal right away, almost half of the issued I/Os are for writing metadata and journal data. We note that the volume of metadata and journal writes in Firefox is only 15% of all writes.

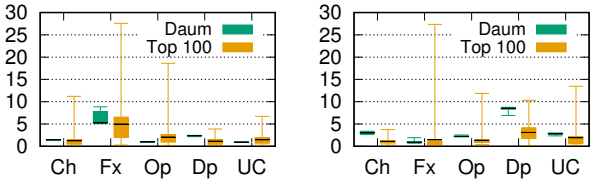
Read vs. Write Fig. 1 shows the average read and write I/O counts of the *www.daum.net* and the top 100 sites. It shows that browsing is write-intensive. There are less than 10 reads observed in all browsers. Opera browser, in particular, does not show any read I/Os. On the contrary, the number of writes varies; about 200 to 300 writes are generated on all browsers, except for Dolphin browser which exhibits 5.6 times more write I/Os than Firefox and 2.8 times more than Chrome.

Buffered vs. Synchronous I/O Fig. 2 shows the count and the volume of buffered and synchronous write operations. Buffered writes are flushed by a kernel process, `pdflush`, which flushes data in page cache to the storage every five seconds. Synchronous writes are flushed immediately, and blocks user process until the flush is completed. Fig. 2(a) shows that the volume of buffered write in Firefox is 4 times more than that of the other browsers and its variability is high. It is because Firefox updates its cache of malware and phish contents which are stored in 15 different files each time the browser launches. The volume it generates for the update varies significantly—the average volume is 9.5 KB and the standard deviation is 3.8 KB. The volume of synchronous I/O on Dolphin browser is about 8 MB, which is 16 times more than the sum of all resources on the website.

fsync vs. fdatasync We examined the ratio of `fdatasync()` and `fsync()` calls. The main difference in the two is that `fdatasync()` flushes metadata of a file only when it is necessary, but `fsync()` call always flushes metadata and data together causing a lot of journal overhead [3]. In the examination, over 90% of all synchronous writes are made with these two calls. Opera, Dolphin, and UC exploits `fdatasync()`–they issue 64%, 94%, and 71% of all synchronous writes with `fdatasync()` calls, respectively. On the contrary, `fsync()` on Chrome and Firefox is about 88% and 86%, respectively. Although the ratio of `fsync()` call is high in Firefox, the browser causes less stress on the storage device because buffered writes are four times more than synchronous writes in Firefox.

4 CONTENT ATTRIBUTES

Block Types We examined the I/O count and I/O volume of file system blocks such as data, metadata, and journal. Although file system is responsible for flushing different block types, applications such as browser and SQLite issues `fdatasync()` and `fsync()` calls to synchronously flush the blocks to the storage. In the examination, journal and metadata updates constitute about 50% of all I/O traffic in browsers. The sum of metadata and journal update count for



(a) Buffered Write (Vol.) (b) Synchronous Write (Vol.)

Figure 2: Write Count and Volume in Mobile Browsers (Unit: KB)

Table 1: Browser Write Amplification Factor (B-WAF, Unit: KB, D: Data, J: Journal, M: Meta)

Case	D	J	M	Total	B-WAF
Chrome	2343	1673	434	4450	8.8
Firefox	6408	790	300	7499	14.8
Opera	1722	1286	262	3270	6.5
Dolphin	5590	4692	434	10716	21.2
UC	2084	1164	449	3697	7.3

Chrome, Firefox, Opera, Dolphin, and UC is 62%, 47%, 51%, 40%, and 58%, respectively.

File Types In terms of volume, all browsers except Firefox stores more than 70% of data on SQLite. Firefox uses 95% of the volume categorized as MISC (e.g. executable files) in checking the malware and phishing contents. About 75% of the size of each requested I/O for malware and phish cache files is 512 KB. Since Firefox rechecks the URL for possible threats, it seems to have much room for optimization. The result also shows that all browsers heavily uses cache to enhance the performance. The volume cached in Dolphin, Opera, and Chrome is 676, 635, and 597 KB, respectively, where the volume of the site is 509 KB.

Database I/O We further break down the I/O count and the size of accessed databases. Dolphin and UC, which are based on Webkit, exploits *webviewCache* to store web cache index information. We note that Dolphin saves about 10 times more data in *webviewCache* than that of UC We also note that the sum of cached data stored in Dolphin (*dolphin_webviewCache.db-wal* and *AppCacheJetpack.db*) is about five times more than the sum of original web contents which is 509 KB.

Even though the database journal mode can be modified, most of the browsers are satisfied with PERSIST mode, regardless of its notorious reputation for a high number of I/Os and `fsync()` calls. Note that WAL mode generates only the half the I/O volume of PERSIST and DELETE mode [3]. Only Opera browser did not make use of WAL journal mode to store a cookie, cache, and history of visited sites. `browser` database used in Firefox (WAL) and Dolphin (PERSIST) stores the history of visited websites and bookmarks. Firefox uses 2 I/Os to store 28 KB and Dolphin issues 98 I/Os to store 476 KB of data after visiting a website.

5 BROWSER WRITE AMPLIFICATION FACTOR

Browser Write Amplification Factor (B-WAF) is a ratio of total bytes received over the network and total bytes a web browser stores to the storage device. Total of 509 KB of data is received over the network (Section 2). Table 1 shows the summary of the result. Dolphin shows the highest WAF of 21.2 which is the result of employing various features such as Jetpack to enhance the overall performance of the browser. The feature stores as much data on local storage to limit network traffic the next time it connects; however, such policy has adversary effect on eMMC based storage system.

Fig. 3 shows boxplot of B-WAF of Top 100 sites by Alexa [1], and it shows that high B-WAF is not an unusual behavior. Max shown in the boxplot spans up to 95% of B-WAF. Although the average

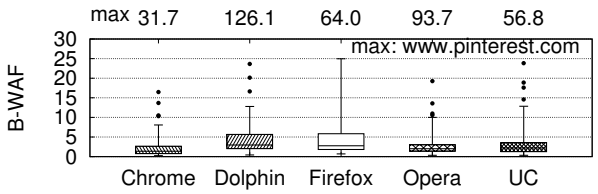


Figure 3: B-WAF of Top 100 Sites by Alexa (values on the top are the max WAF)

B-WAF of top 100 sites on Chrome, Dolphin, Firefox, Opera, and UC is 2.6, 5.9, 6.2, 3.9, and 3.9 respectively, the outliers stretch to as high as 31.7, 126.1, 64.0, 93.7, and 56.8, respectively.

6 CONCLUSION

We analyzed the I/O behavior of five web browsers, namely Chrome, Firefox, Opera, Dolphin, and UC. The original website we used in the experiment has 52 files with the sum of 509 KB. We used Androtrace to analyze the I/O behavior of browsers, and found that browsers issue over 60% of 4 KB writes, and the browsers (Chrome, Opera, Dolphin, and UC) heavily rely on synchronous writes to persist the data (over 68%). Over 44% of the total volume of the browsers are for storing file system metadata and journal, and over 43% are for storing SQLite related files (except for Firefox). We also measured the browser write amplification factor (B-WAF) and found that the average B-WAF is generally very high; we found that web browsers write on average of 11.7 times more data to the storage than the sum of original web contents.

ACKNOWLEDGMENTS

This work was supported by Basic Research Laboratory Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT&Future Planning(MSIP)(No. 2017R1A4A1015498), the ICT R&D program of MSIP/IITP. (R7117-16-0232, Development of extreme I/O storage technology for 32Gbps data services), the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2017- 2012-0-00554) supervised by the IITP(Institute for Information&communications Technology Promotion), BK21 Plus project of the National Research Foundation of Korea Grant, and the research group promotion program, Gyeongsang National University, 2017

REFERENCES

[1] Alexa. 2015. Site Overview - How popular is daum.net. http://www.alexa.com/siteinfo/daum.net. (2015). [Online; accessed 1-Nov-2015].

[2] Android Open Source Project. 2015. ADB Shell Commands. https://developer.android.com/studio/command-line/adb.html. (2015).

[3] Sooman Jeong, Kisung Lee, Seongjin Lee, Seoungbum Son, and Youjip Won. 2013. I/O Stack Optimization for Smartphones. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference (USENIX ATC'13)*. 309–320. http://dl.acm.org/citation.cfm?id=2535461.2535499

[4] Hyojun Kim, Nitin Agrawal, and Cristian Ungureanu. 2012. Revisiting Storage for Smartphones. *Trans. Storage* 8, 4, Article 14 (Dec. 2012), 25 pages. http://doi.acm.org/10.1145/2385603.2385607

[5] Eunryoung Lim, Seongjin Lee, and Youjip Won. 2015. Androtrace: Framework for Tracing and Analyzing IOs on Android. In *Proceedings of the 3rd Workshop on Interactions of NVM/FLASH with Operating Systems and Workloads (INFLOW '15)*. Article 3, 8 pages.