Comparative Study on I/O Characteristics of Mobile Web Browsers

Myungsik Kim, Seongjin Lee, and Youjip Won
Department of Computer Science Engineering, Hanyang University, Seoul, Korea.
{mskim77, insight, yjwon}@hanyang.ac.kr

***Abstract*—Web browsers are one of the most widely used application in smartphone. Since there are a number of different layout engines for the web browsers, the IO efficiency of each browsers are very different. In this paper, we identify areas of improvement by analyzing and comparing the IO characteristics of four widely used web browsers, that is Chrome, Firefox, Opera, and Dolphin. Via analyzing I/O characteristics of the four mobile web browsers, Firefox browser is the most efficient I/O behavior and Dolphin Browser is the worst one. Firefox has highest buffered writes in 57%. Journal and metadata are 21% which is the smallest fraction of all I/O size. In contrast, Dolphin has highest synchronous write ratio in 79%. Dolphin generates 502 of `fsync()` calls which is 6× more than Firefox. Through the comparative study, we identify areas for improvement by comparing I/O characteristics of the mobile web browsers.**

***Index Terms*—Android, Mobile Platform, Smartphone, Mobile Web Browser, I/O Workload Characterization**

I. Introduction

The smartphones are gradually replacing the role of desktop computers in everyday usage and becoming the primary computing device [1]. Some of the most important use cases of mobile devices are entertainments such as game, music, and video, social networking, and browsing information. It is interesting to see how web browsers and web engines are competing each other to take the lead in the browser war [2]. Since mobile devices are widely spreading and competing each other intensively, it is worthwhile to evaluate and compare the I/O performance of different browsers.

Operations on web browser can be roughly categorized into three groups: (i) receiving web data over the network, (ii) parsing HTML file using rendering engine, and (iii) displaying the data. When network bandwidths were not fast enough the dominant performance bottleneck of mobile web browsers were observed on receiving the data over the network; thus, optimizing network performance of web browsers were one of most important issues [3]. As network bandwidth is becoming higher, the bottleneck seems to lie somewhere between network I/O and storage I/O. More recent web browsers provides many more features than mere display of web data. It now has to support multi browsing process and also follow HTML5 extended web standards which all leads to more I/O operations for local storage devices.

In this work, we explore I/O behavior of four different web browsers and identify some of design factors that could be improved. We find that Dolphin browser shows naive policy in disk caching for web data. Caching policy of Dolphin browser is tightly coupled with SQLite database and it creates excessive amount of I/Os. In order to reduce network I/Os for web data, Dolphin browser rigorously caches the web data which in result generates significant amount of disk I/Os. Note that SQLite is a transaction based database and manages roll-back or roll-forward journal depending on SQLite journal modes, which generates more I/Os than what user has updated, and moreover file system writes journal for the database journal written by the SQLite [4].

II. Experiment

In this study, we use Nexus5 (Hammerhead) smartphone, which uses Android Kitkat Ver. 4.4 with Linux kernel Ver. 3.4. The device uses EXT4 file system on top of built-in 16GB eMMC. The default browser for Android platform is Chrome browser and SQLite is default database. The Chrome browser adopts Webkit which is an opensource based web rending engine. Along with Chrome browser (Ver 41.0), we installed Firefox (Ver. 36.0.4), Opera (Ver. 2.80,1764), and Dolphin (Ver. 11.4.3). We evaluated the I/O performance of web browsers by visiting the same web page (*www.daum.net*) on each browsers.

In order to acquire I/O traces from Android platform while visiting a web page, we used `Androtrace` [5] which is a real-time I/O acquisition tool for Android platform. This tool allows collecting LBA, I/O size, I/O request time, and process name of running applications. It is also possible to distinguish different synchronization system calls such as `fsync()` and `fdatasync()`, and identify file name as well as file type. Table. I describes the captured fields in `Androtrace`.

Fig. 1 illustrates trace acquisition process of `Androtrace`. The trace logging mechanism of `Androtrace` are based on

TABLE I: Trace Log in Androtrace

1	I/O issue date and time
2	I/O operation type (read, write, sync write)
3	Device number (major : minor)
4	I/O size and Sector Address
5	Filesystem block type: Meta, Journal, Data
6	File name
7	Original process name
8	Flags for I/O called by `fsync()`, `fdatasync()`
9	Flags for I/O caused by SQLite related file

blktrace [6], and uses blk_peek_request() in block layer to trace I/Os when I/O schedule dispatches requests to the storage.

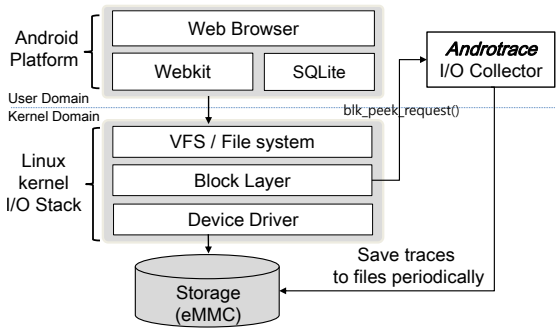


Fig. 1: I/O Traces Acquisition Process

III. Results

Using the Androtrace, we analyzed six I/O attributes, that is I/O sizes, I/O type (read or write), filesystem block type (data block, metadata block, journal block), synchronization mode (buffered vs. synchronous I/O), database related files, and fsync()/fdatasync() count.

A. I/O Attributes

After analyzing the I/O access pattern and investigating the access behavior of database related files, we found that there are significant difference in how each web browser manage their data. Fig. 2 shows volume of read and write I/Os, and write volume is further distinguished. we find interesting that read I/Os are rare in loading a web page and it is write intensive workload. While the volume of the visited web page (*www.daum.net*) is about 500KB, the volume each browsers generated is about 15 times larger on average. And, synchronous 4KB I/Os were dominant in all browsers. Other than Firefox which shows the least fraction of 4KB I/Os (29% of all I/O sizes), 4KB I/Os on all browsers constitutes 67% of all I/O sizes on the average.

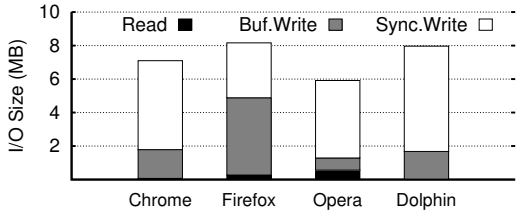


Fig. 2: I/O Attributes

B. Filetype

Fig. 3 shows amount of data, metadata, and file system journal observed on each browsers. In terms of file type, the amount of I/Os for SQLite related files vary among the browsers. On Chrome, Firefox, Opera, and Dolphin browser, 23%, 8%, 26%, and 61% of all writes are SQLite related files, respectively. SQLite related files are database file (*.db), rollback journal file (*.db-journal), WAL (Write Ahead Log) file (*.db-wal), etc.

Filesystem journal (JBD2 in EXT4) and metadata (e.g., inodes, bitmaps, directories, group descriptors) account for 50%, 20%, 35%, and 41% of the total writes on Chrome, Firefox, Opera, and Dolphin browser, respectively. Chrome browser has a higher fraction of journal and metadata than the other browsers.

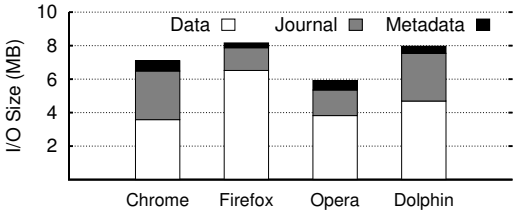


Fig. 3: File Types

C. Synchronization

Fig. 4 shows the synchronizations calls made on each web browsers. Note that in general, synchronous I/Os are known as blocking I/O because the process issued the I/O needs to wait until the requested data block is written the to storage device; thus, it is used only when it is necessary in order to increase the I/O performance. However, we find it interesting that much of I/Os on all web browsers are flushed to the storage using synchronization calls. About 68%, 44%, 54%, and 83% of all I/Os are synchronously written to the storage. fsync()/fdatasync() are linux system calls which are issued by application to synchronize data on storage. The number of fsync()'s generated by the web browsers in loading a same web page is surprisingly high; there were 339, 84, 149, and 502 fsync() calls on Chrome, Firefox, Opera, and Dolphin browser, respectively. Note that Dolphin browser generates almost 6× more fsync() calls than that of Firefox.

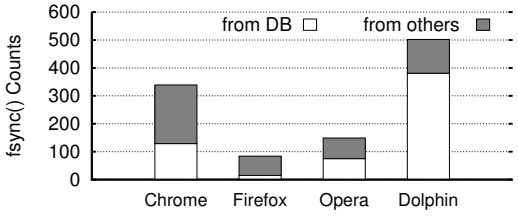


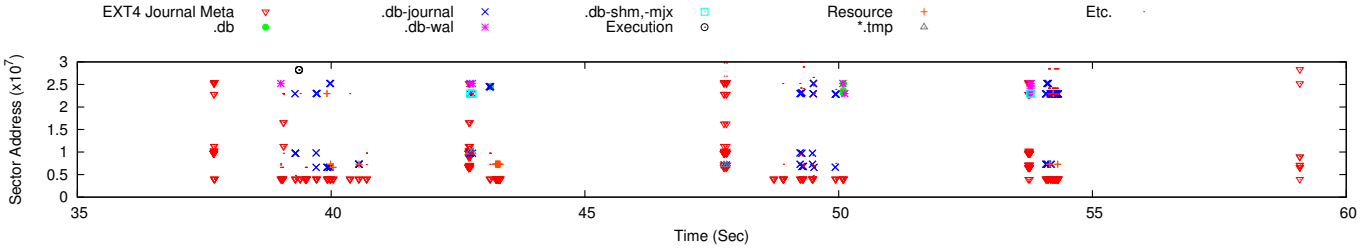
Fig. 4: fsync() Call Counts

D. I/O Workload Behavior

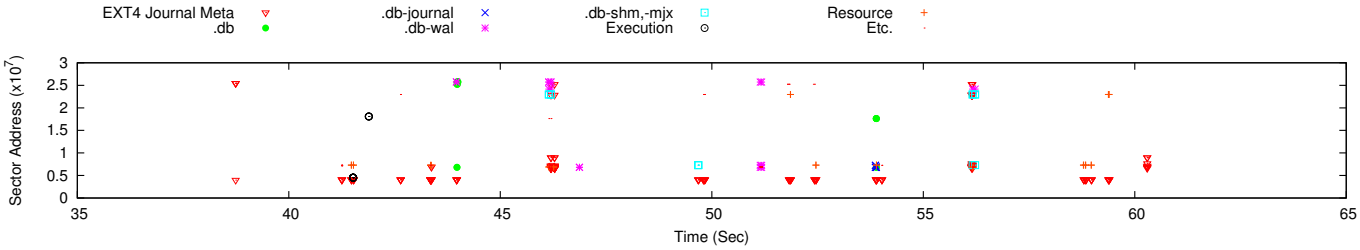
Fig. 5 shows sector address against time graph of all web browsers, and each point in the graph is categorized as one of the followings: EXT4-journal/metadata, SQLite related files (.db-journal, .db-shm/.-mjx, .db, .db-wal), executive files, resources, .tmp, and etc. Note that we have segmented SQLite related I/Os in detail to illustrate

TABLE II: I/O Characterization Summary in Mobile Web Browsers, Cnt: I/O Counts, Size Unit: KByte (% of Total)

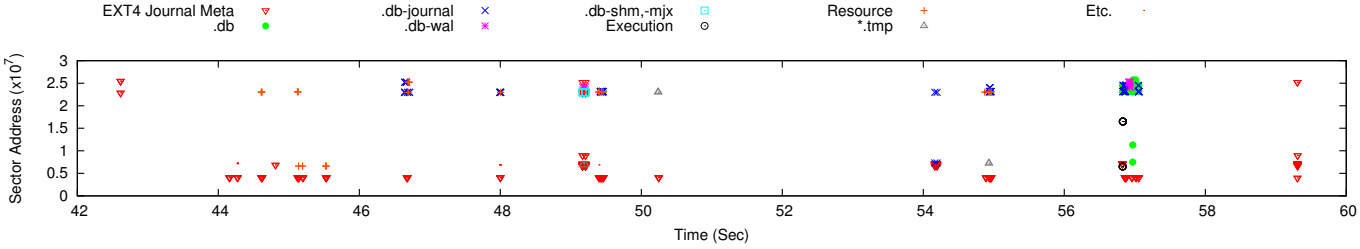
		Chrome	Firefox	Opera	Dolphin			Chrome	Firefox	Opera	Dolphin
Tot	Cnt	575	411	292	631	Cnt	Data	330(57)	105(51)	166(57)	376(60)
	Size	7096	8164	5916	7964		Journal	118(21)	49(24)	63(22)	172(17)
4KB	Count	391(68)	119(58)	194(66)	421(67)		Meta	126(22)	50(25)	62(21)	83(13)
	Size	1564(22)	478(6)	778(13)	1686(21)	Size	Data	3580(50)	6516(80)	3824(65)	4692(59)
Cnt	R	4(1)	9(4)	22(8)	0(0)		Journal	2904(41)	1360(17)	1528(26)	2848(36)
	W	211(37)	106(52)	111(38)	109(17)		Meta	612(9)	288(4)	564(10)	424(5)
	WS	360(63)	90(44)	159(54)	522(83)	DB	Count	131(23)	16(8)	75(26)	382(61)
Size	R	68(1)	268(3)	564(10)	0(0)		Size	1828(26)	264(3)	2516(43)	3716(47)
	W	1720(24)	4616(57)	724(12)	1680(21)	fsync	fsync	339	84	149	502
	WS	5308(75)	3280(40)	4628(78)	6284(79)		fdsync	264(78)	20(24)	95(64)	460(92)
Cnt: I/O Count, fdsync: fdatasync							DB	129(38)	15(18)	75(50)	381(76)



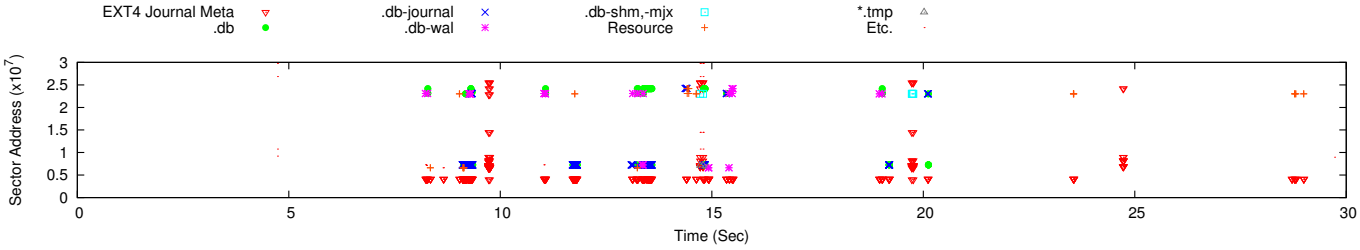
(a) Chrome



(b) FireFox



(c) Opera



(d) Dolphin

Fig. 5: I/O Workload Behavior in Mobile Web Browsers

its heavy I/O activity. Although web page is almost instantly displayed on the web browser; however, we observe that there are a lot of background I/Os even though there is no user input. Table II shows summary of the Fig. 5. The graph shows that I/O activities on Dolphin are bursty but does not last as long as other browsers, in fact all I/O activities are completed within half the time of the other browsers. As there are significant amount of `fsync()` calls on browsers, it may interfere with user inputs as a result the responsiveness of the browser can be slowed down.

IV. Discussion

The results on Table II shows that there are significant amount of file system journal/metadata and SQLite journal I/Os, and they consume more than half of all the I/Os observed on web browsers. There are a number of database files used in loading a web page. Another finding we made is that these database files are synchronously written to the storage using `fsync()`. After reviewing the source of the I/Os, we find one or two database files are responsible for more than half of all database related file I/Os. On Dolphin browser, for example, about 61% of all I/O count are for SQLite related files. There were 40 updates on `ztrack.db` and as a result `ztrack.db-journal` received 100 updates. We suggest that web browsers should be more concerned about I/O overhead on storage, and should use `fdatasync()` over `fsync()`.

`browser.db` is a platform-wide database in Android which is shared among different browsers. One interesting observation we made is that different browsers use different journal modes in updating the database. Firefox uses WAL mode and other browsers uses a rollback mode (persistent, truncate, and delete). As a result, Firefox made 8 I/Os using WAL journal on `browser.db-wal`, and Dolphin used rollback journal on (`browser.db-journal`) and issued 144 I/Os to access the database. SQLite recommends to use WAL journal mode since it is generally faster than other journal modes [7]. We believe app developers first have to decide if it is really necessary to persistently store the data, and if it is indeed necessary then it should be stored in appropriate journal mode that creates least number of I/Os.

According to our analysis, Firefox web browser yields the most efficient I/O behavior in number of ways. First, it generates the least number of synchronous I/Os (40% of all I/O size); second, it also generates the least number of database accesses; and third, it uses WAL journal mode to access a database. On the contrary, Dolphin browser was the worst in terms of I/O efficiency, and leaves much room for improvement. The I/O behavior of Dolphin browser is just the opposite of the behavior of Firefox, it creates the most number of synchronous I/O, frequently updates the database, and uses rollback journal.

It is interesting to see that each browsers are aggressively caching web data. The purpose of using cache is to avoid long latency of accessing data over the network and making the reaccess of the data faster. However, if an user never accesses same web page twice, it is needless to cache all the web data which has heavy I/O overhead.

V. Conclusion

In this work, we analyzed I/O behaviors of four widely used web browsers, which are Chrome, Firefox, Opera, and Dolphin, in loading a web page. Although the volume of the web content is only about 500KB, the I/O volume created by each web browsers are as high as 16 times the original volume. We show that loading a page is write intensive workload with much of writes being issued with synchronous I/Os to flush SQLite related files. Only one or two databases are responsible for most of SQLite related file I/Os. All browsers heavily uses caching to store web data received over the network; since all the cached data is stored in the SQLite database, they are subject to SQLite journal and file system journal, which generates bursty I/Os. After analyzing and comparing the I/O behavior four web browsers, we find Firefox is the most I/O efficient web browser and Dolphin is the most inefficient web browser. Firefox has highest buffered writes in 57%. Journal and metadata are 21% which is the smallest fraction of all I/O size. In contrast, Dolphin has highest synchronous write ratio in 79%, 6× of `fsync()` calls compared to Firefox are generated. We expect that the findings can provide direction of I/O handling improvement scheme of mobile web browsers.

VI. Acknowledgment

This work was sponsored by IT R&D program MKE/KEIT (No.10041608, Embedded system Software for New-memory based Smart Device). This research was supported by the MSIP(Ministry of Science, ICT&Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2015- H8501-15-1006) supervised by the IITP (Institute for Information&communications Technology Promotion).

References

[1] G. Janessa Rivera. (2014) Gartner says worldwide traditional PC, tablet, ultramobile and mobile phone shipments are on pace to grow 6.9 percent in 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2692318

[2] R. Metz. (February 7, 2013) The browser wars go mobile. [Online]. Available: http://www.technologyreview.com/news/510486/the-browser-wars-go-mobile/

[3] I. Grigorik, *High Performance Browser Networking: What every web developer should know about networking and web performance*. " O'Reilly Media, Inc.", 2013.

[4] S. Jeong, K. Lee, S. Lee, S. Son, and Y. Won, "I/O stack optimization for smartphones," in *Proceedings of the 2013 USENIX Conference on Annual Technical Conference*, ser. USENIX ATC'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 309–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=2535461.2535499

[5] K. Kim, E. Lim, S. Lee, and Y. Won, "Real-time io trace analysis of smartphone users," in *In Proceedings of 2nd International Conference on Advances on Computer, Electrical and Electronic Engineering (ICACEEE 2014).* International Journal of Information Technology Computer Science (IJITCS), 2014, pp. 7–8.

[6] A. D. Brunelle, "Block I/O layer tracing: blktrace," *HP, Gelato-Cupertino, CA, USA*, 2006.

[7] S. Consortium. (2012) Write-ahead logging. [Online]. Available: http://www.sqlite.org/wal.html