2018 7th IEEE Non-Volatile Memory Systems and Applications Symposium

Embedded DBMS Design for In-Vehicle Information Management

Joontaek Oh and Youjip Won
Hanyang University, Seoul, Korea

Abstract

Embedded devices have built-in software, which results in them having to include storage for software. Embedded devices use flash storage since it should be strong against physical impact. However, flash storage has a limited lifespan, which is equivalent to the lifespan of embedded device. Relatively inexpensive embedded devices can tolerate such limitation, but in the case of high cost devices like smartcars, such ephemeral lifespan is unacceptable. In this paper, we improve the lifespan of smartcars by improving the lifespan of their flash storage, by minimizing the write volume of SQLite.

SQLite that is serverless DBMS that is mainly used in embedded devices stores the log in a separated file, journal file, at commit time. This is a logging. Logging doubles the write volume because the SQLite writes data to both the log file and the database file. Moreover, when using a journaling filesystem such as EXT4, the write volume is amplified by the journaling of journal anomaly. It leads to more program/erase operations in flash storage. It short the lifespan of flash storage.

This anomaly leads to the shortened lifespan of its embedded device. If the embedded device is a smartcar and the lifespan is reduced down to five years due to write amplification, we would have to pay extravagant cost every five years. We need to optimize for long-term use of smartcars before using them. There have been many efforts to reduce the write volume of SQLite. This paper examines how smartcars can improve their lifespan when a previous study to reduce the write volume of SQLite is applied to smartcars.

Environment	Write volume	Writing frequency
A	About 500 bytes	100 msec
B	About 150 KB	1 min
C	About 400 KB	1 week

TABLE I
THREE OF ENVIRONMENT MODELS IN SMARTCAR.

We assumed three environments according to the smartcar IO pattern. Table I represents our three assumptions. These environments do not have realistic evidence. Since we are not smartcar vendor, we have assumed environments through famous functions of smartcar.

In environment A, cars exchange their location information with other cars to prevent colliding. Each car sends its location information to the surrounding cars every 100 msec and the data is about 500 bytes. In environment B, the car and traffic information manager server communicate with each other. The car transmits its position and status to the traffic information manager server once per sec and its data is about 150 KB. The C environment is an environment for transmitting and receiving map information. The car sends its current location and destination to the cloud server. The server obtains the route to the destination of the car and transmits the corresponding route data back to the car. The map is cached for one week on the flash storage in the car. In this environment, the software inserts about 400 KB of data into SQLite once a week.

In above environments, the data is stored transactionally by SQLite, amplifying the write volume. We modeled how much the write volume is amplified as a formula. These formulas are primitives based on behavior of SQLite and filesystem. These formulas are available on a Full sync WAL mode.

$$S(x) = 2x + 1 \tag{1}$$

In the Equation 1, x is the number of database pages to change. $S(x)$ means the number of pages to write when SQLite modifies x of pages. First, all the data pages to be changed are written to the WAL file and the database file respectively ($2x$). In addition, the WAL frame header is written as well, adding an extra page.

$$F_{ext4}(x) = x + 16 \tag{2}$$

Equation 2 indicates the number of blocks that the filesystem actually writes to flash storage in EXT4. EXT4 modifies three metadata blocks: GDT, data block bitmap and inode table, and they are logged by journaling. Since the SQLite writes data to WAL files and database files respectively, the write to the metadata is doubled. Thus, 16 blocks including journal descriptor blocks and journal commit blocks are added to the total write volume.

$$F_{f2fs}(x) = x + 2 \tag{3}$$

In F2FS, only data blocks and nodes are written. At this time, `fdataysnc()` is called for both the database file and the WAL file, writing two additional nodes.

$$L_C = \frac{MAX_{P/E} \times C}{W_{day} \times WAF} \tag{4}$$

Jeong et al [1] modeled a formula representing the lifespan of a flash storage through factors: device WAF(WAF), total capacity of storage(C), maximum number of program/erase

2575-257X/18/$31.00 ©2018 IEEE
DOI 10.1109/NVMSA.2018.00028

Env	W_{day} (MB)	Lifespan (year)
A	97875.0	6.9
B	2244.4	300.0
C	0.6	1184083.0
A+B+C	100119.9	6.7

TABLE II
THREE OF ENVIRONMENT MODELS IN SMARTCAR.

cycles($MAX_{P/E}$), and total write volume per day(W_{day} (Equation 4).

Table II indicates the lifespan of smartcar in situations in I extracted from Equation 4 and the environment that is union of all environments in Table I (A+B+C). The storage assumes a 6 GB flash storage with a $MAX_{P/E}$ of 100,000 formatted with the EXT4 filesystem. The WAF assumes Park et al [2]'s worst WAF value of 2.5. In this assumption, it is assumed that five tables are modified.

In environment A+B+C, the lifespan of flash storage is about 6.7 years that is very short lifespan for expensive device like smartcar. We used two studies to increase the lifespan: WALDIO [3], F2FS single file atomic write [4].

WALDIO [3] solves the Journaling of Journal anomaly. WALDIO embeds a WAL frame header on each page, which eliminates further writes about WAL frame header, and the data block of the WAL file is pre-allocated to fix the size, and the writing to the WAL file uses direct IO to eliminate the journaling of the WAL file.

$$S(x) = 2x \tag{5}$$

With WALDIO, Equation 1 is improved to Equation 5 by eliminating writes about the WAL header.

$$F_{ext4}(x) = x + 8 \tag{6}$$

Equation 2 is advanced to Equation 6 because filesystem journaling about WAL file is eliminated.

Env	W_{day} (MB)	Lifespan (year)
A	67500.0	10.0
B	2193.8	306.9
C	0.6	1194634.3
A+B+C	69694.3	9.7

TABLE III
THREE OF ENVIRONMENT MODELS IN SMARTCAR WITH WALDIO.

Table III represents lifespan of flash storage using SQLite with WALDIO. In environment A+B+C, the lifespan has improved about 46%.

F2FS supports multi-block atomic write feature [4]. This feature allows the user to write multi blocks to a file atomically. Recently, SQLite has begun to support logging mode using F2FS atomic writes. When SQLite is compiled with the special option, all transactions will be committed through F2FS single file atomic write.

$$S(x) = x \tag{7}$$

With F2FS atomic write, Equation 1 is revamped to Equation 7. Since the database file is updated atomically, it does not need to log the pages anymore.

$$F_{f2fs}(x) = x + 1 \tag{8}$$

Equation 3 is changed to Equation 8, because there is no other file besides the database file.

Env	W_{day} (MB)	Lifespan (year)
A	23,625	28.5
B	1,080	623.4
C	0.3	2403547
A+B+C	24,705.3	27.3

TABLE IV
THREE OF ENVIRONMENT MODELS IN SMARTCAR WITH F2FS ATOMIC WRITE.

Table IV represents lifespan of smartcar using SQLite with F2FS atomic write. In environment A+B+C, the lifespan has improved about 392%.

We have modeled the write amplification when SQLite performs an insert operation in transaction (Equation 1, Equation 2, Equation 3). We assumed the environment that can occur in a smartcar where flash storage lifespan is lethal and costly (Table I). Finally, we derive Park et al's formula for the lifespan of a smartcar when SQLite write is amplified in our assumption. The derived lifespan was short when we consider the cot of smartcars. We have applied previous studies to minimize the write volume of SQLite to increase the lifespan of flash storage. The role of previous studies is to reduce $S(x)$ and $F(x)$. When we applied these studies, we discovered that the lifespan of flash storage increases from 1.4 $\times$ to 4 $\times$.

The write volume is as important as throughput in embedded devices with flash storage. A technique such as journaling has the advantage of maintaining consistency without reducing throughput. However, it amplify the write volume by writing the metadata twice. Now, a technique that minimizes the write volume should be studied for embedded devices.

Acknowledgement

This work was supported by the BK21+ program (NRF), Basic Research Lab program (NRF No. 2017R1A4A1015498), ICT R&D program (IITP R7117-16-0232) and FutureOS grant (IITP No. 2018-0-00549).

REFERENCES

[1] J. Jeong, S. S. Hahn, S. Lee, and J. Kim, "Lifetime improvement of nand flash-based storage systems using dynamic program and erase scaling." in *Proc. of USENIX FAST*, 2014, pp. 61–74.

[2] C. Park, S. Lee, Y. Won, and S. Ahn, "Practical implication of analytical models for ssd write amplification," in *Proc. of the 8th ACM ICPE*. ACM, 2017, pp. 257–262.

[3] W. Lee, K. Lee, H. Son, W.-H. Kim, B. Nam, and Y. Won, "Waldio: eliminating the filesystem journaling in resolving the journaling of journal anomaly." Usenix, 2015.

[4] J. Kim, "F2FS: support `atomic_write` feature for database," https://lkml.org/lkml/2014/9/26/19.