SAMSUNG

Automatic I/O stream management based on file characteristics

Yuqi Zhang, Ni Xue, Yangxu Zhou

2021

Samsung R&D Institute China Xi'an(SRCX) established in 2013
and located in Xi'an High-tech Zone, is the only cutting-edge technology
R&D institute of Samsung Electronics in western China.

SRCX

西安三星电子研究所 | SAMSUNG R&D INSTITUTE CHINA XIAN

Contents

Introduction

- Major issues in using SSD

  - Reduced **performance** and bad **QoS** of I/O latency

  - Short **lifespan**
  - Low **capacity utilization**
  - Increasing **power consumption**

  - Sudden SSD **failure**

Performance

Cost

Reliability

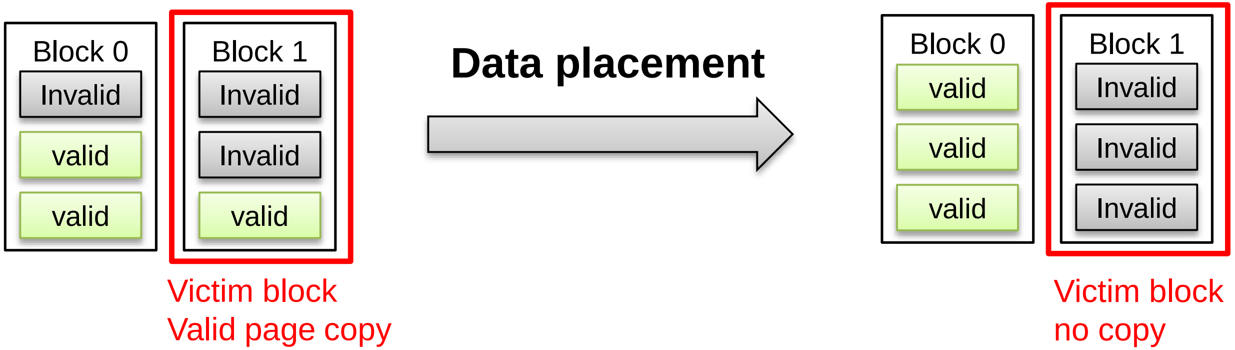
Introduction

■ GC's impact on performance and cost

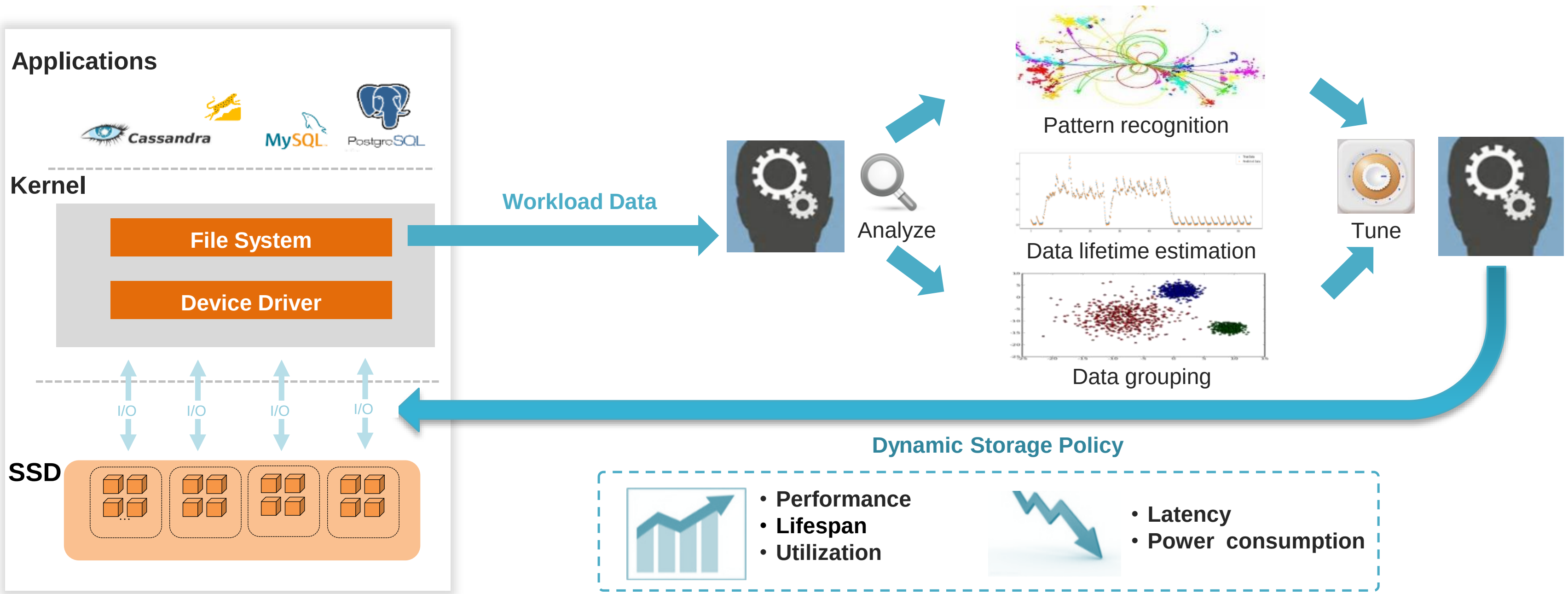
- Valid page copy

- It consumes internal computing and bandwidth resources and decreases the I/O response performance.
- The more data are migrated, the faster NAND flash will be worn out.

■ Placing data based on their lifetime to reduce GC overhead



Introduction



Introduction

- Existing automatic stream assignment schemes

  - Data unit

  - LBA chunks -> AutoStream (SYSTOR ’17) / LKStream (HotStorage’19)
  - Program contexts -> PCStream (FAST’19)

    ✓ Data lifetime varies within the same data unit

    ✓ Additional code modification is required to obtain characteristics

- File — the natural data unit

  - The access patterns of data within the same file are similar
  - Various file characteristics help to group data
  - File characteristics are easy to obtain

File Analysis

Append-only File

- **Sstable files in LSM-Tree**
- **Write ahead logs**
- **...**

✓ Data are updated into the new file

✓ Data are invalid only after the file is deleted

✓ Files are frequently created and deleted

VS

In-place Update File

- **Table files in SQL database**
- **...**

✓ Data are updated inside the file

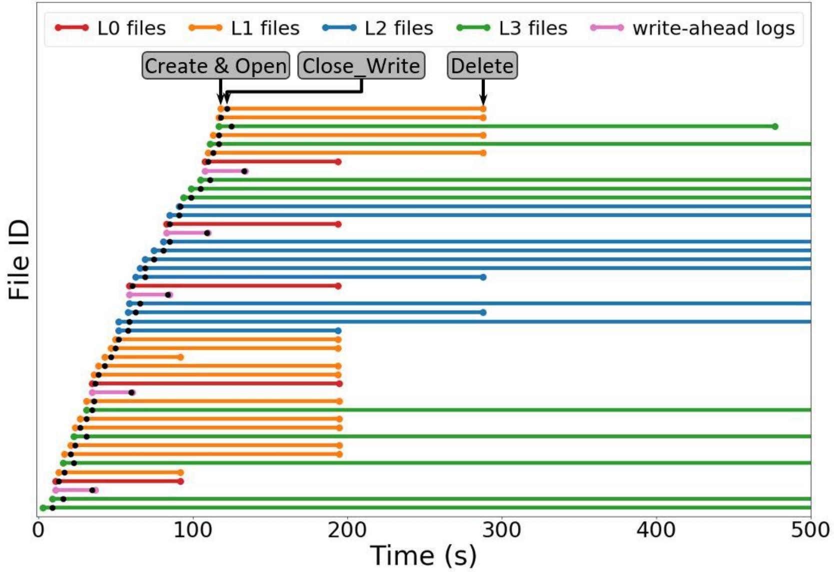
✓ Data are invalid when they are covered

✓ Files usually exist for a long time

File Analysis

■ Append-only File

- ✓ These files are frequently created and deleted. They complete the writing in only one or several seconds and there are a few files written at the same time.
- ✓ The data lifetimes within the file are the same or similar and can be reflected by the file lifetime.
- ✓ The same type of files may have similar lifetime, while different types of append-only files usually have different lifetimes.



The operations of RocksDB files

File Analysis

■ In-place Update File

- ✓ These files exist for a long time and they are created/deleted once the tables are created/deleted.
- ✓ The update frequencies of data in different files are quite different, while those in the same file are relatively similar.
- ✓ Data lifetime is related to update frequency. we propose unit-size-modification (USM), the ratio of the number of file modifications to the file size, to reflect update frequency.

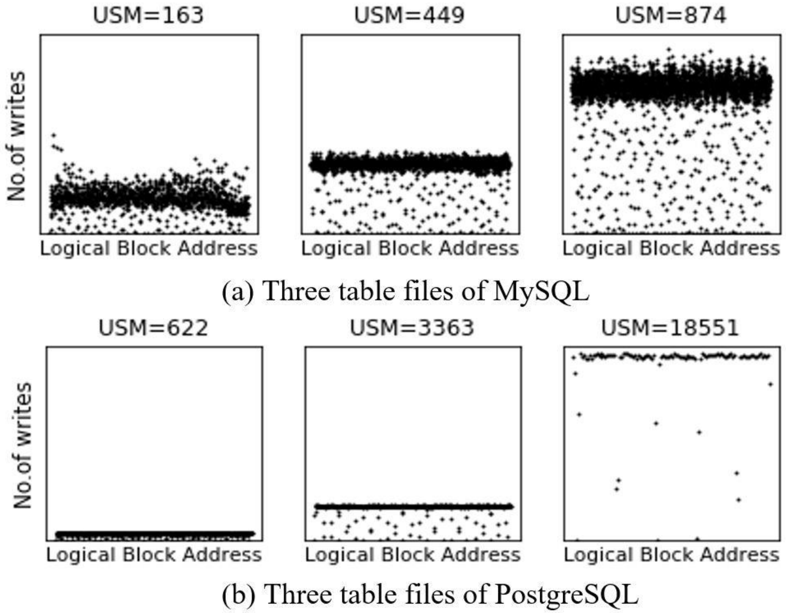


Table files' writes in MySQL&PostgreSQL

FileStream Design

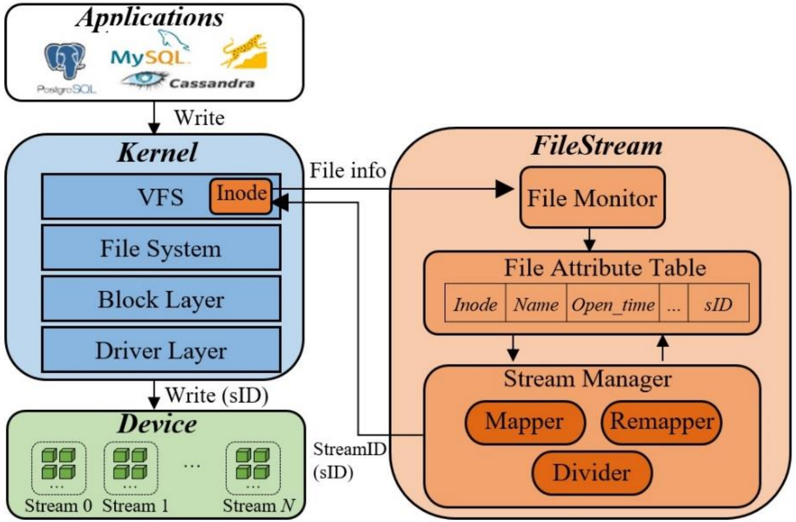
- **File Monitor**
  - Capture file characteristics

- **Stream Manager**
  - Map or remap each file into a stream

  - **Mapper**
    Assign stream IDs to files when they are opened

  - **Remapper**
    Reassign stream IDs to long opened files

  - **Divider**
    Allocate different stream IDs for the mapper and remapper



FileStream Design

Mapper

Some files complete writing in seconds, as in many append-only files. Mapper is responsible for mapping the file to the stream when the file is opened

Minimize the mixing of data between different files, since the data lifetimes within the file are relatively similar

Estimate the amount of data to be written in existing files (less data to be written means less data will be mixed with the new file)

$$amount(f_i) = m(f_i) * (wd(f_i) - cd(f_i))$$

- $m(f_i)$: average number of modifications per second of each existing file f_i
- $wd(f_i)$-$cd(f_i)$: estimated remaining duration of f_i to be written

FileStream Design

Mapper

Minimize the lifetime difference between files written to the same stream

Estimates the lifetime difference between the new file f_o and each existing file f_i based on the lifetimes of the same type of files

$$dif(f_i, f_o) = \frac{1}{N_i N_o} \sum_{f_a \epsilon F_i} \sum_{f_b \epsilon F_o} \frac{|l(f_a) - l(f_b)|}{l(f_a) + l(f_b)}$$

- F_i, F_o : the set of deleted files with the same type as f_i and f_o, respectively
- N_i, N_o : the number of files in F_i and F_o, respectively
- $l(f_a)$,$l(f_b)$: the lifetime of the file f_a in F_i and f_b in F_o, respectively

FileStream Design

Mapper

- Combine the two factors

- Multiply two factors, and integrate the scores of existing files of stream s as the score of the stream s

$$score(s) = \sum_{f_i \epsilon F_s} amount(f_i) * dif(f_i, f_o)$$

  - F_s : The set of existing files of stream s

- Select the stream ID with the lowest score and assign it to the new file f_o

FileStream Design

■ Remapper

Some files tend to remain open for a long time with I/O changes, as in many in-place update files. Remapper is responsible for assigning stream IDs to files that remain open for T seconds

Group files regularly based on their USM, since the data lifetimes of these files are usually related to the update frequency of data

Calculate the USM of files and cluster files by Kmeans++ every T seconds

$$USM = modifications\ /\ size \;\Rightarrow\; \text{Kmeans++}$$

FileStream Design

■ Remapper

Skip clustering operations when workload is relatively stable

Calculate the distances between files and cluster centers. If files are still closest to the previous cluster centers, skip the clustering operation

Advantages:

- Reduce the fluctuation of the remapped stream IDs
- Reduce the CPU and memory consumption

FileStream Design

- Divider

Divider dynamically allocates different stream IDs for mapper and remapper (1 to *d* and (*d*+1) to *N*, respectively)

More files or more data written ⇨ More streams

Calculate the proportion of the streams that mapper should occupy by the number of files managed by mapper and remapper (labeled as FN_m and FN_r) and the total number of their modifications (labeled as MN_m and MN_r)

$$proportion_m = \left(\frac{FN_m}{FN_m + FN_r} + \frac{MN_m}{MN_m + MN_r}\right)/2$$

Adjust d and round d every *T* seconds, i.e., d = $\lfloor N * proportion_m + \frac{1}{2} \rfloor$

Evaluation

Workloads

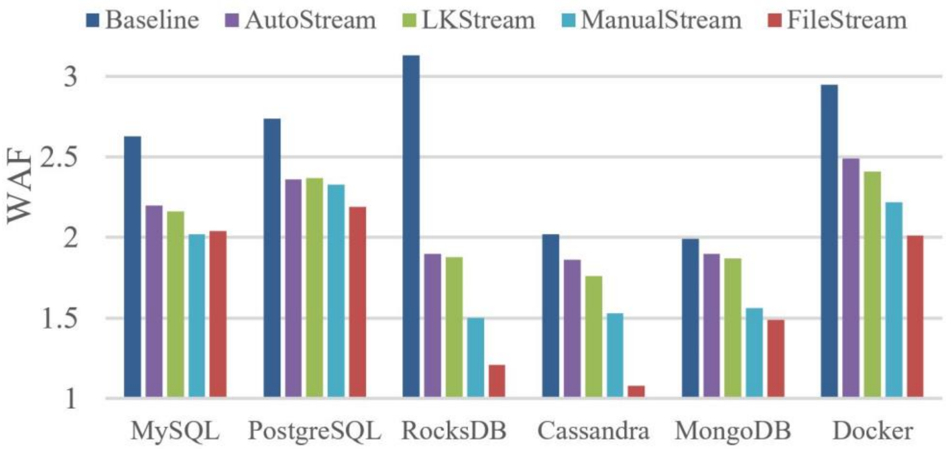
Workload	Benchmark tool	Operation
MySQL (v5.17.2)	TPC-C	TPC-C 64 connections
PostgreSQL (v9.6.15)	pgbench	pgbench 64 connections
RocksDB (v5.17.2)	db_bench	Update 50 million records
Cassandra (v3.6.11)	YCSB	Update 170 million records
MongoDB (v3.6.14)	YCSB	Update 70 million records
Docker(2 MySQL+ 2 RocksDB)	TPC-C db_bench	TPC-C 64 connections Update 20 million records

Settings

- Multi-streamed SSD (SAMSUNG PM963), 8 user-configurable streams
- Comparisons: 1) Baseline, 2) AutoStream, 3) LKStream, 4) ManualStream
- T is set to 60s by default and it is discussed later

Evaluation

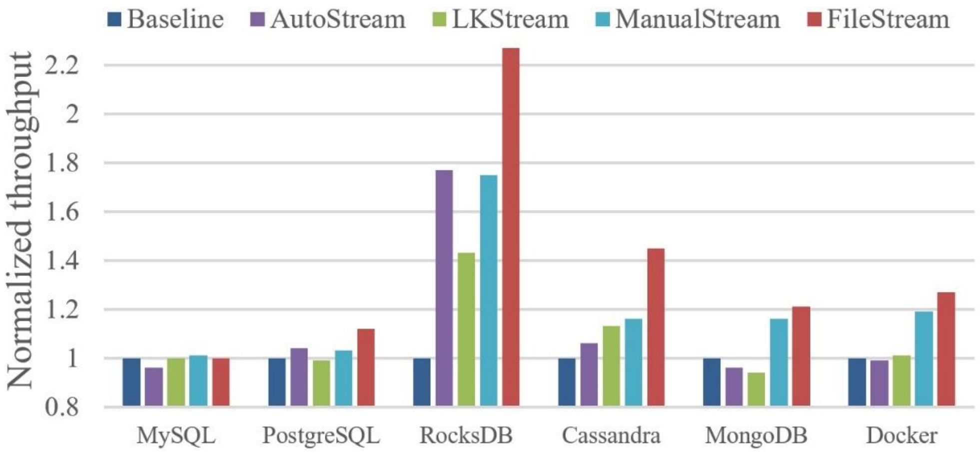
- WAF



- FileStream significantly reduces WAF compared with other schemes
- FileStream works well on all six workloads

Evaluation

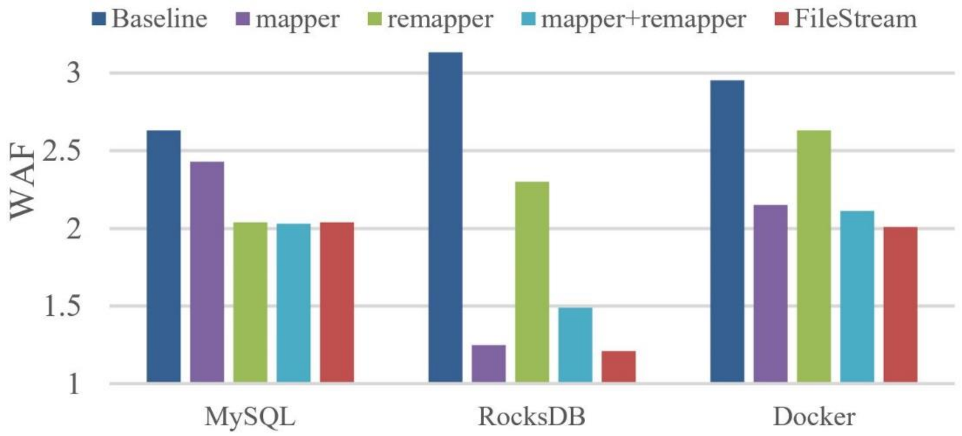
■ Throughput



- FileStream significantly improves throughput compared with other schemes
- FileStream performs very well on RocksDB and Cassandra in particular

Evaluation

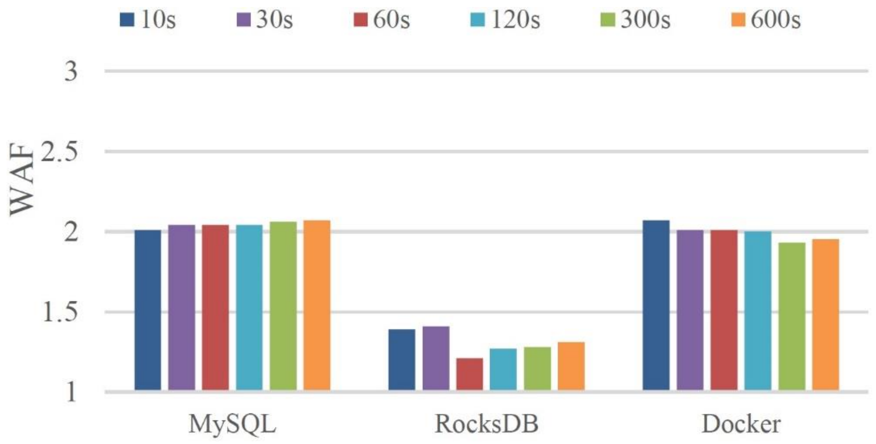
■ Module effectiveness



- Mapper focuses on newly opened files, so it performs better on RocksDB with many frequently created append-only files
- Remapper focuses on long opened files, so it performs better on MySQL with many long term in-place update files
- The cooperation of three modules makes FileStream optimal

Evaluation

■ Key parameter T



- Further research is required about the different WAF changes on three workloads
- FileStream is not very sensitive to this parameter

Evaluation

■ Resource Consumption

Table 2: Comparison of CPU utilization

Workload	Baseline		FileStream	
	average	max	average	max
MySQL	9.2%	23.9%	9.3%	25.5%
RocksDB	3.8%	9.7%	4.3%	10.5%
Docker	5.9%	15.2%	6.8%	16.1%

- Compared with the baseline, the increase in average CPU usage and peak usage is negligible

CONCLUSION

■ Achievements

- ✓ We propose a file-based automatic stream management scheme which greatly reduces WAF and improves the performance of SSDs on various workloads
- ✓ We propose to estimate the mixing degree of data with different lifetimes to separate the data of newly opened files more thoroughly
- ✓ We design a new feature to reflect the access patterns of long opened files. Based on this, we use machine learning to group files

■ Future work

- We will apply our scheme to more devices such as Zoned Namespace SSDs
- We will combine file information and LBA-related information for situations where the data lifetimes within the file are very different

SAMSUNG

2021

THANK YOU

SRCX

Samsung R&D Institute China Xi'an(SRCX) established in 2013 and located in Xi'an High-tech Zone, is the only cutting-edge technology R&D institute of Samsung Electronics in western China.

西安三星电子研究所 | SAMSUNG R&D INSTITUTE CHINA XIAN