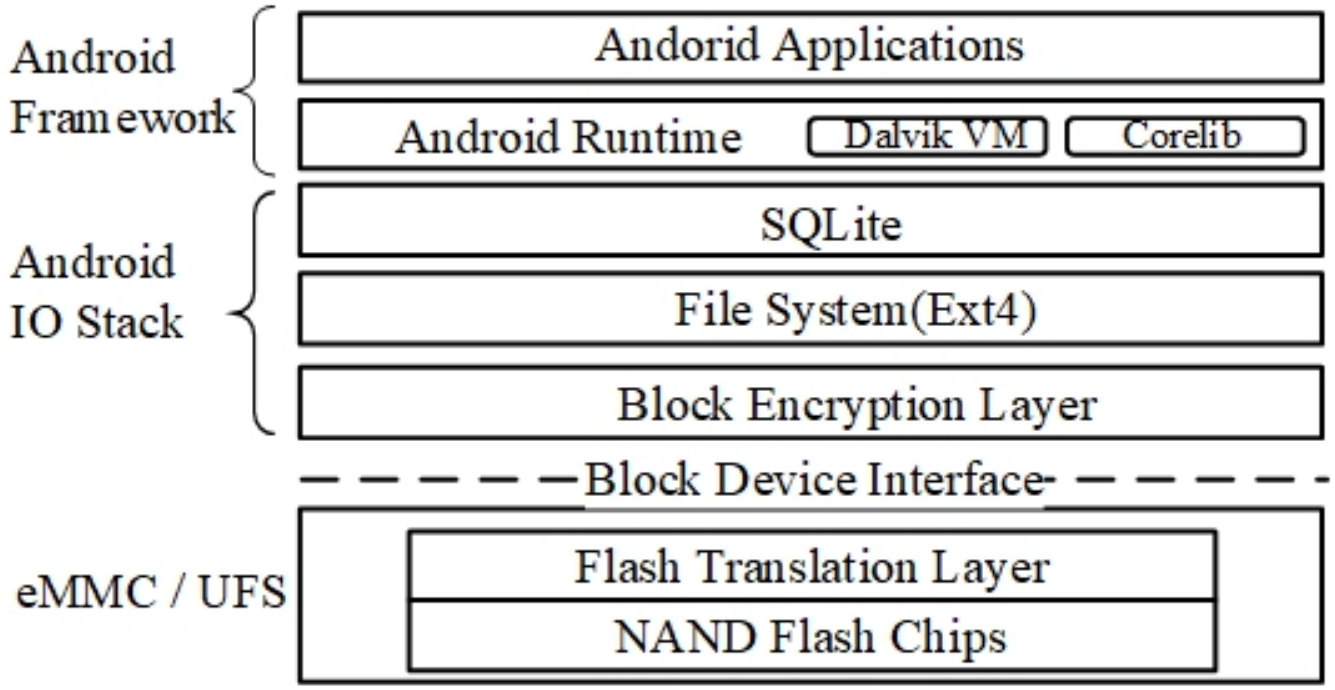
Write Stress Reduction on Mobile Storage: Observations and Methodologies

Prof. Li-Pin Chang/張立平

National Chiao Tung University, Taiwan, ROC

Presented at FISS'19@KAIST, Korea

Android I/O Stack



Multiple Write Amplifications

waf

>=2 for small files

2~5 for moderately utilized flash

Application data

Mobile IO stack (journaling, cleaning)

Block I/O

FTL (garbage collection)

Flash RW

Write performance can be a problem, but storage lifespan is a more important issue!

Mobile Storage Lifespan

- Is mobile storage lifespan a real problem?
  - Smartphone replacement cycle is increasing (3 yr)
  - TLC and QLC flash have low P/E cycle endurance
  - I/O patterns of mobile storage is write intensive, nearly 90% of I/Os are write [Kim,FAST'12][Lee,EMSOFT'12]

Our Current Efforts

- Flash garbage collection and wear leveling are relatively mature topics
- We are focused on how to reduce the amount of write traffic bound for mobile storage

- [Storage firmware] Adding extra components for transparent write stress reduction
- [File System/Middleware] Revising/redesigning file systems and/or middleware

Firmware Approaches

- FTL compression [Ji,EMSOFT'17]
- FTL deduplication [Yen,EMSOFT'18]

- These approaches should
  - be lightweight because they are supposed to be implemented in FTL
  - exploit smartphone I/O behaviors for the best result

The Common (mis?)Beliefs

- Smartphones store multimedia contents (e.g., picture files and video clips)
  - Multimedia contents are not compressible
  - Multimedia files do not share common file fragments
  - So compression and deduplication are useless in mobile storage…?

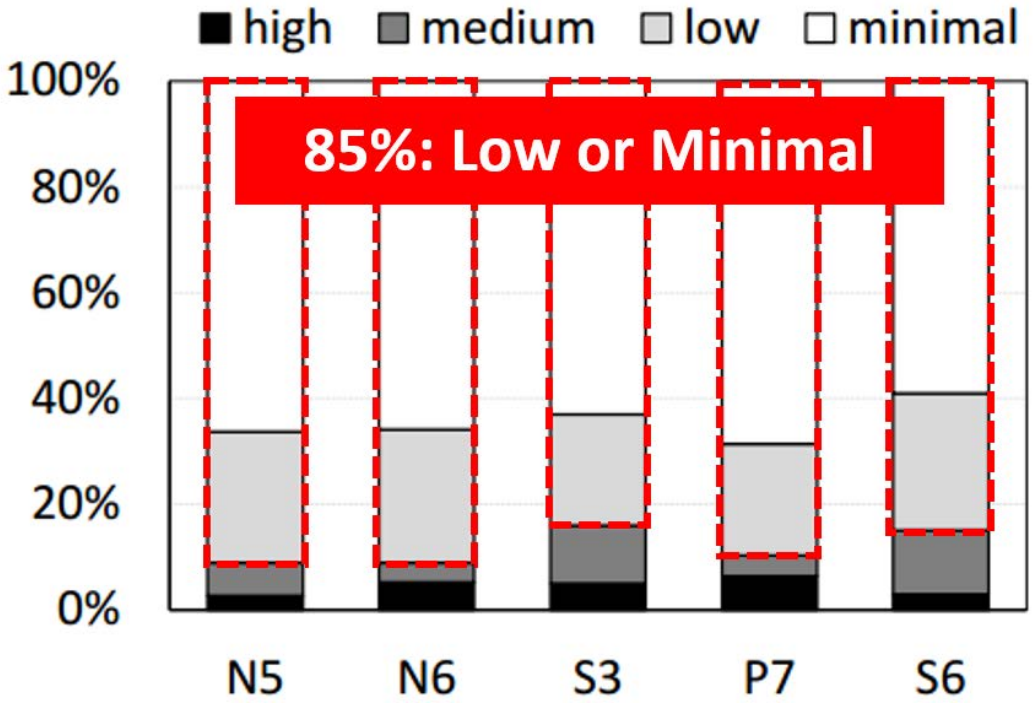
- Nope. We are not talking about shrinking file size. We are talking about reducing write traffic volume!

Compression in FTL

- Main questions

- Are data compressible in mobile storage?
- Is it feasible to implement firmware compression?

Storage Snapshot Analysis

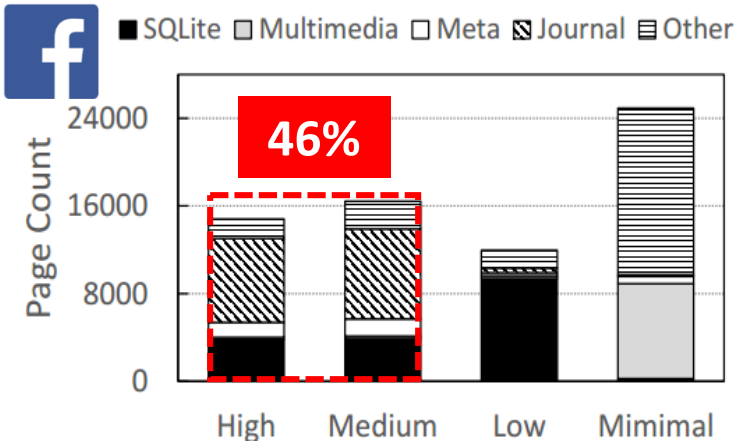


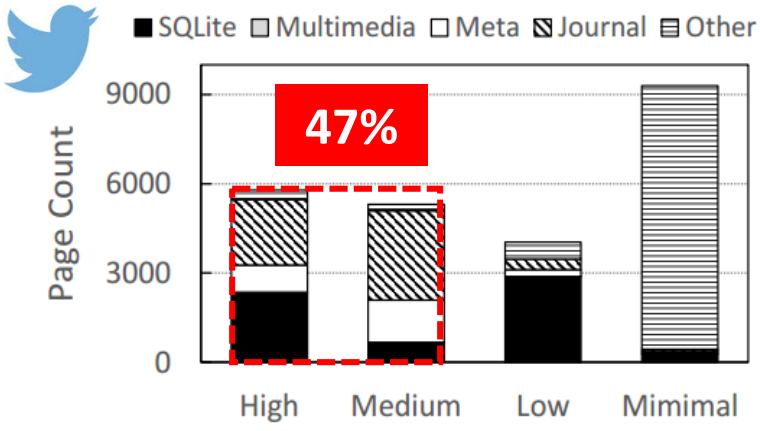
← 5 smartphones

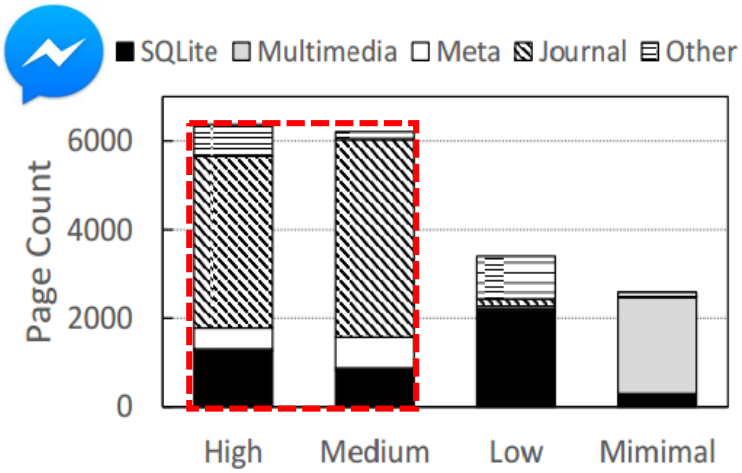
High **(Cr<0.25),** ***Medium*** **(0.25≤Cr<0.65),**
Low **(0.65≤Cr<0.95),** ***Minimal*** **(0.95≤Cr ≤1)**

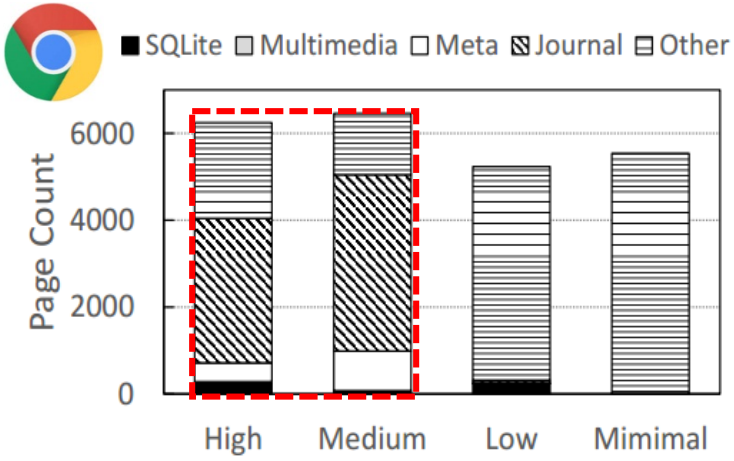
compression ratio ***Cr = (compressed size)/(uncompressed size)***
LZO compression algorithm, Cr: the smaller the better

Write Traffic Analysis









Observations

- Static storage snapshots are almost incompressible, and therefore compression does not help with file system fullness

- Online write traffic is, on the other hand, highly compressible!
- A large amount of writes are focused on a small set of disk blocks
  - Overwriting DB pages to append small records
  - Overwriting FS metadata blocks for fsync() operations

Selective Block Compression

- Firmware-based compression
  - Slow controller SoC and limited RAM
- Compressing incompressible data wastes time and energy
  - Compression must be selective
- How to predict compression ratios before actual data compression?

Entropy-Based **Cr** Prediction

- Predictable correlation between entropy and compression ratio
- Entropy calculation is much faster than compression (1.8% of compression time)

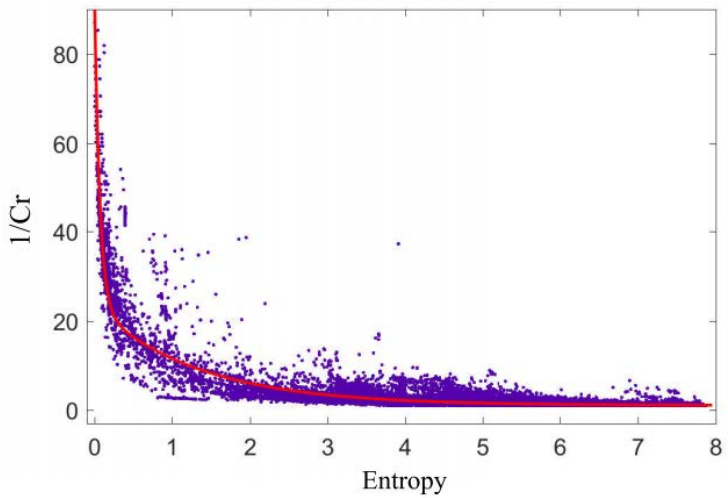
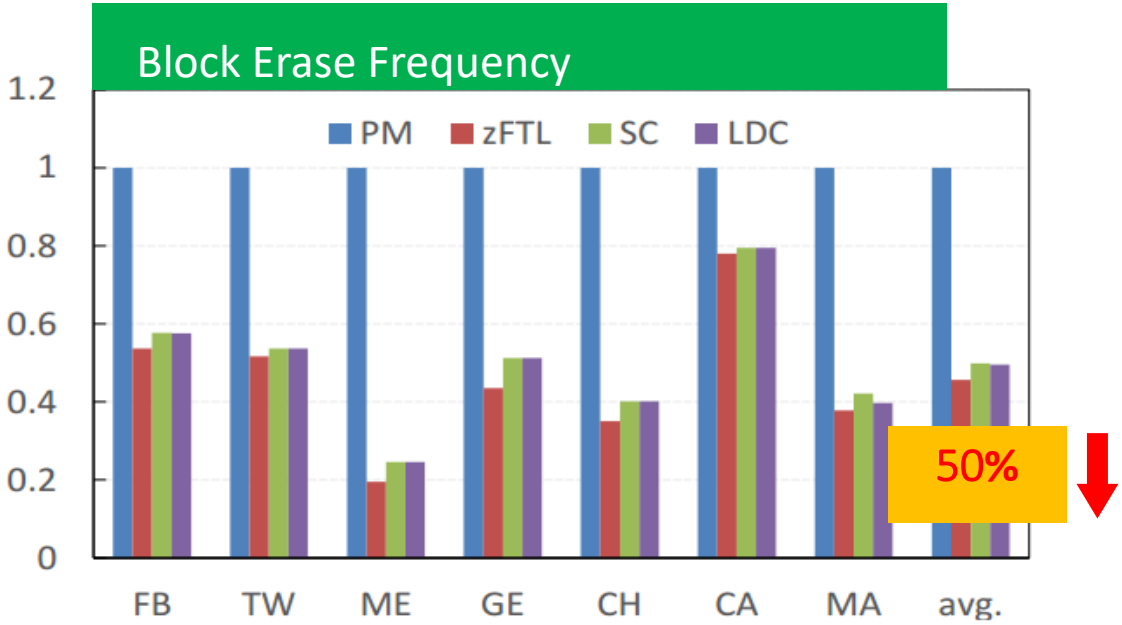


Fig. Entropy and compression ratio of 4KB disk blocks of the volume snapshot of the N6.

Georges Hansel, et. al. 1992. Compression and entropy. In Annual Symposium on Theoretical Aspects of Computer Science. Springer, 513–528

Compression Results

- Total block erase count is reduced by ~ 50%
- Slightly decreased write latency
- But slightly degraded read latency (<+5%)

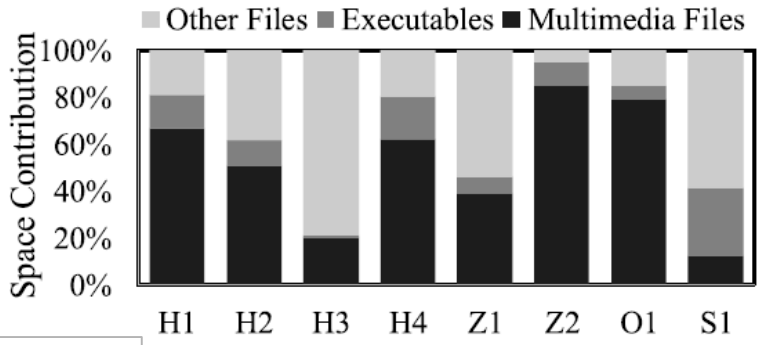


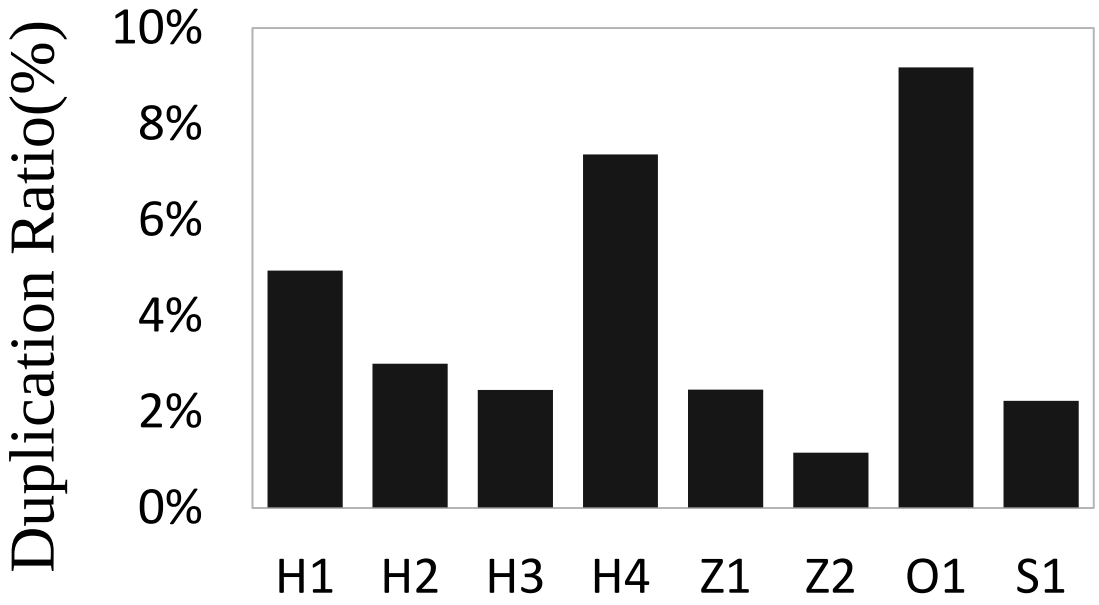
Deduplication in FTL

- Main questions

- Are there sufficiently many duplicate data in mobile storage?
- How to reveal as much deduplication as possible?

Snapshot Analysis



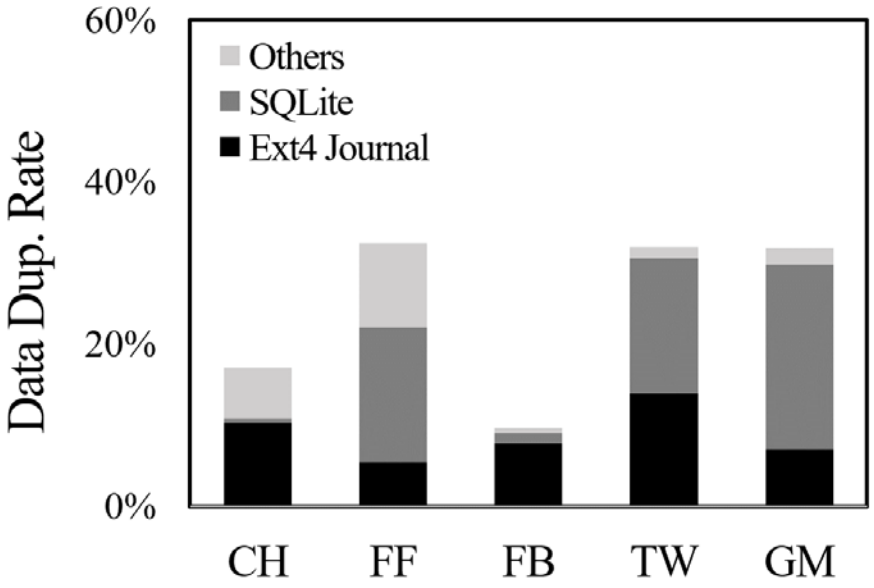


← 8 smartphones

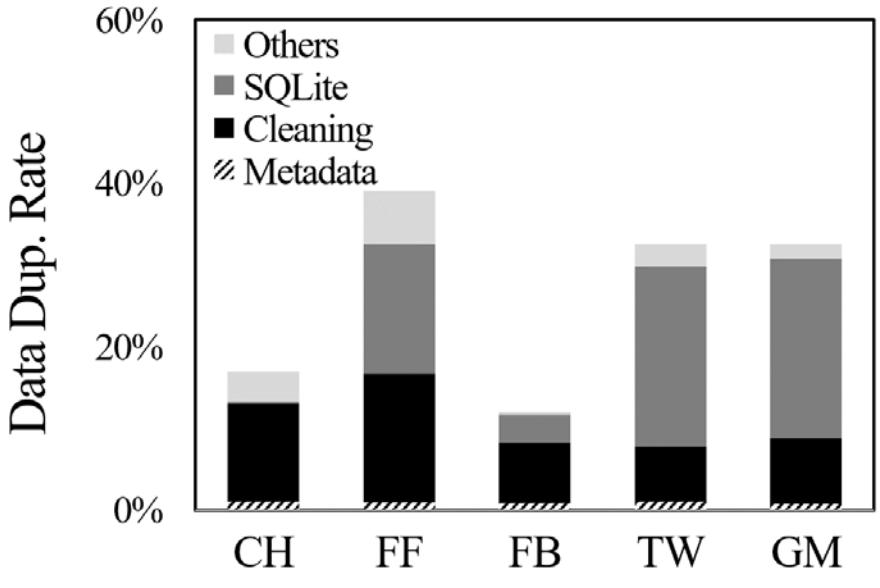
- Data duplication ratio =
  amount of dudup data / total amount of data
- Very pessimistic results

Write Traffic Analysis

- On average, write traffic bound for mobile storage carries ~=30% duplicate data
  - SQLite journaling
  - Ext4 journaling or F2FS cleaning

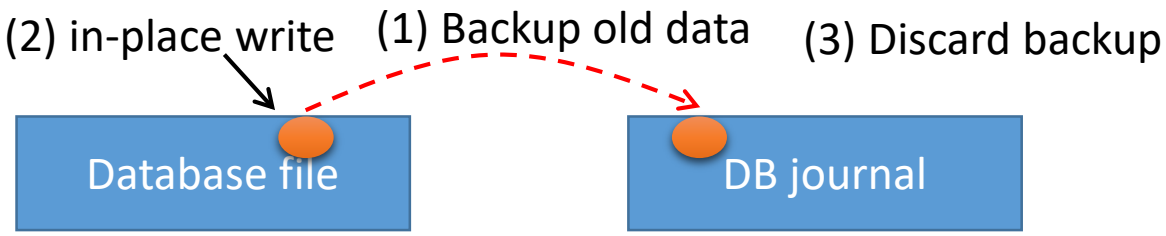


Ext4

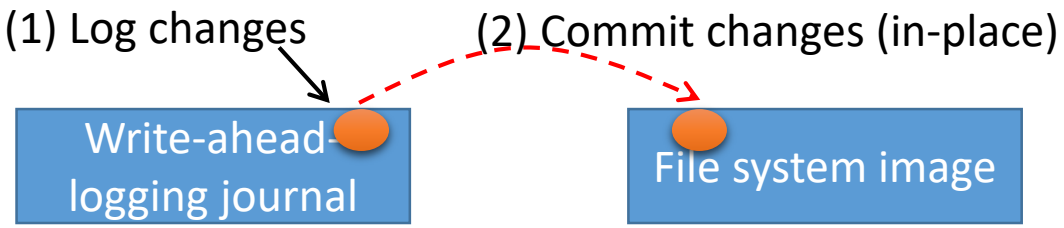


F2FS

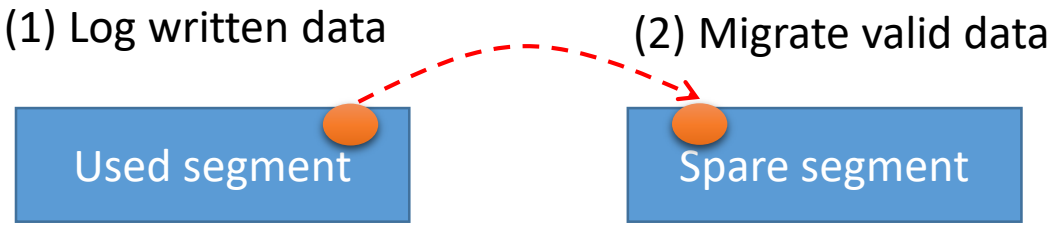
Copy, Copy, and Copy…



SQLite roll-back journaling



Ext4 ordered-mode journaling

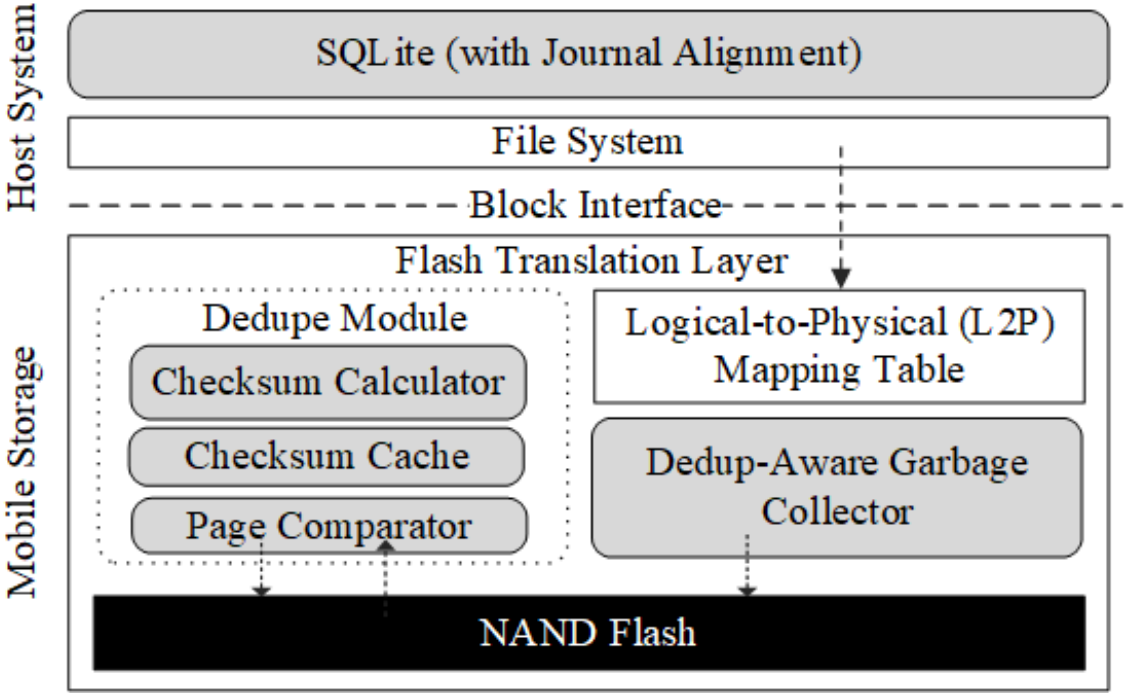


F2FS cleaning

A lot of copy-induced data duplication in write traffic

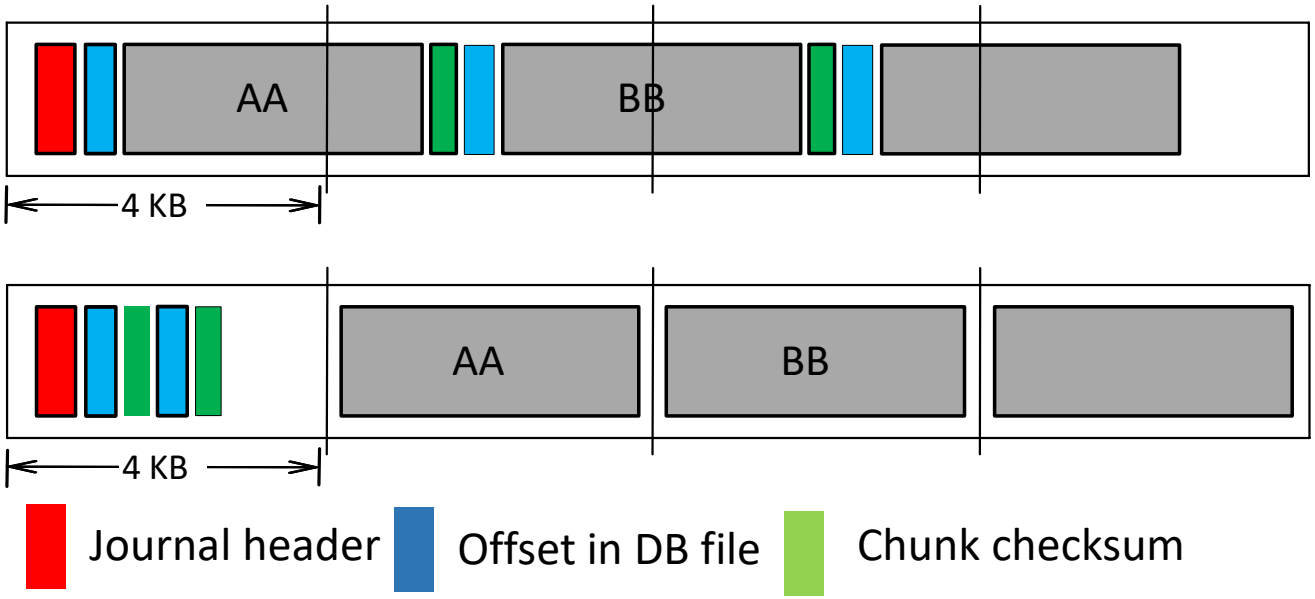
Integrated Deduplication

- Best deduplication results involve assistance from system software (SQLite) and flash management (garbage collection)



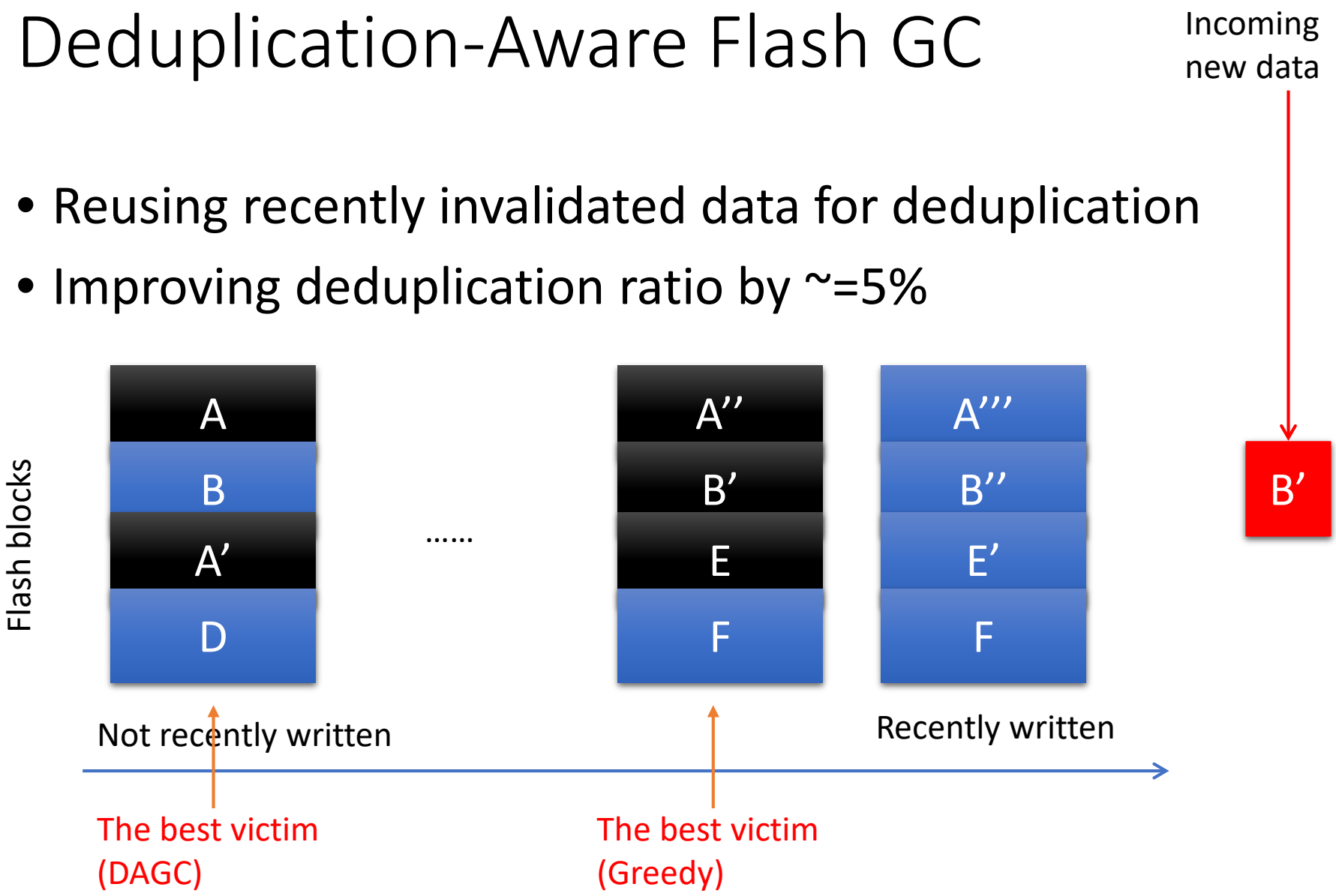
Metadata is *Harmful* to Deduplication

- DB pages are not aligned to 4 KB boundaries
  - Hard to detect duplication
  - Dynamic chunking is too expensive
- Revising SQLite file format



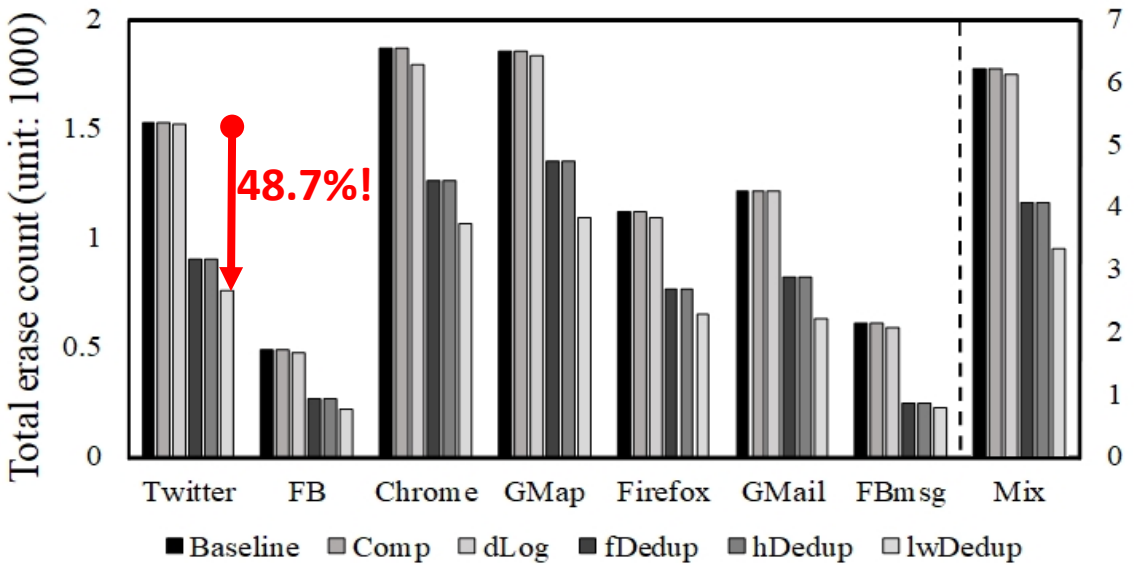
Deduplication-Aware Flash GC

- Reusing recently invalidated data for deduplication
- Improving deduplication ratio by ~=5%



Deduplication Results

- Reducing total erase count by ~=50%
  - Interestingly, dedup removed many random writes
- Improving write response by ~=40%
  - (dedup faster than compression)



Erase count reduction

Compression vs. Deduplication

- Working prototypes on Jasmine OpenSSD
  - Time overhead: dedup < compression
  - Implementation complexity: compression > dedup
  - Total EC reduction: dedup ~= compression
- 2KB dedup is much better than 4KB dedup
  - Reluctant to do it… complex sub-page mapping
- Deduplication works with disk encryption, but compression does not

File-System Approaches

- Exploiting high-level information such as file type
- Not to worry about data encryption
- More RAM space and CPU power
- Current efforts
  - F2FS + compression
  - F2FS + deduplication

One Among Many Questions…

- If F2FS produces duplicate data through cleaning, what if F2FS stops to clean?
- To clean
  - Lower I/O count
  - Efficient medium access
- Not to clean
  - Lower write stress

- Deduplication is a metadata operation, but it introduces (or worsenes) file fragmentation

Conclusion

- Deduplication and compression do reduce write traffic volume of mobile storage
  - They do not squeeze more free space however
- Firmware implementation of deduplication and compression are feasible
- Host-side disk encryption neutralizes FTL compression but FTL deduplication is still applicable

- Working on file-system compression or deduplication…