19th USENIX Conference on File and Storage Technologies
February 23–25, 2021
FAST'21

Pattern-Guided File Compression with User-Experience Enhancement for Log-Structured File System on Mobile Devices

Cheng Ji [§], *Li-Pin Chang* [※], *Riwei Pan* [♯], *Chao Wu* [♯], *Congming Gao* [*], *Liang Shi* [¶], *Tei-Wei Kuo* [♯], *Chun Jason Xue* [♯]

[§] Nanjing University of Science and Technology [※]National Chiao Tung University
[♯]City University of Hong Kong [*]Tsinghua University [¶]East China Normal University

Overview

- Background and Motivation
- Pattern-Guided File Compression
- Implementation
- Evaluation
- Conclusion

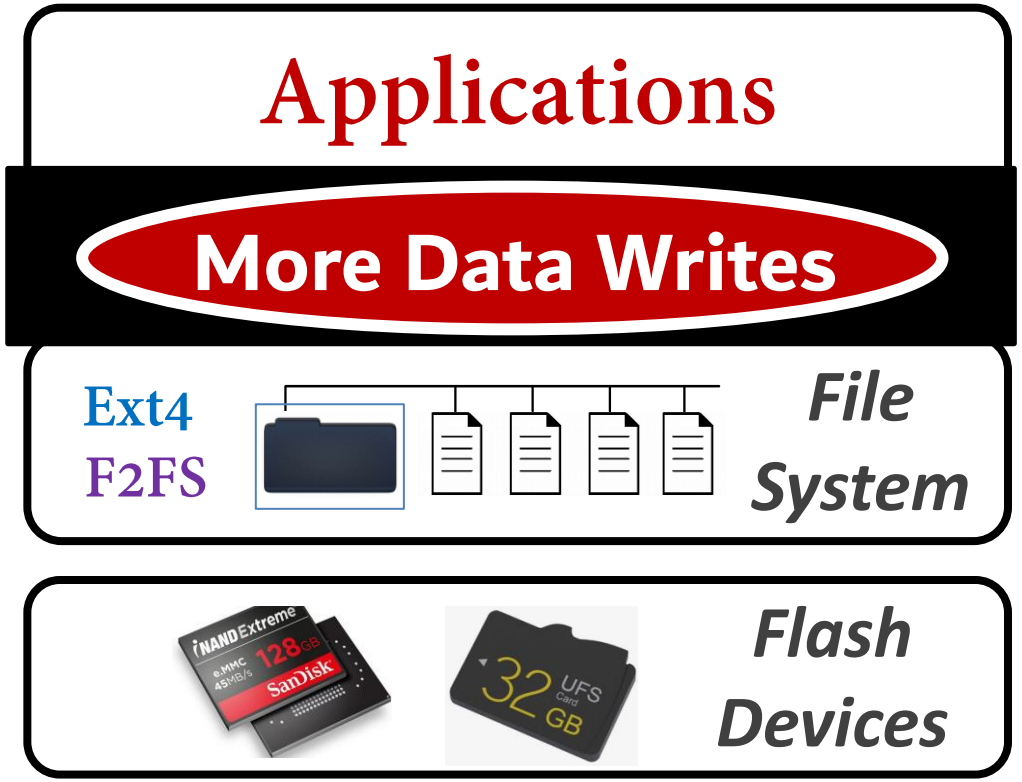
Mobile Device Popularity

Mobile devices are everywhere!

ANDROID

Write Pressure on Mobile Systems

- More data are filling in, but **storage has its limit**



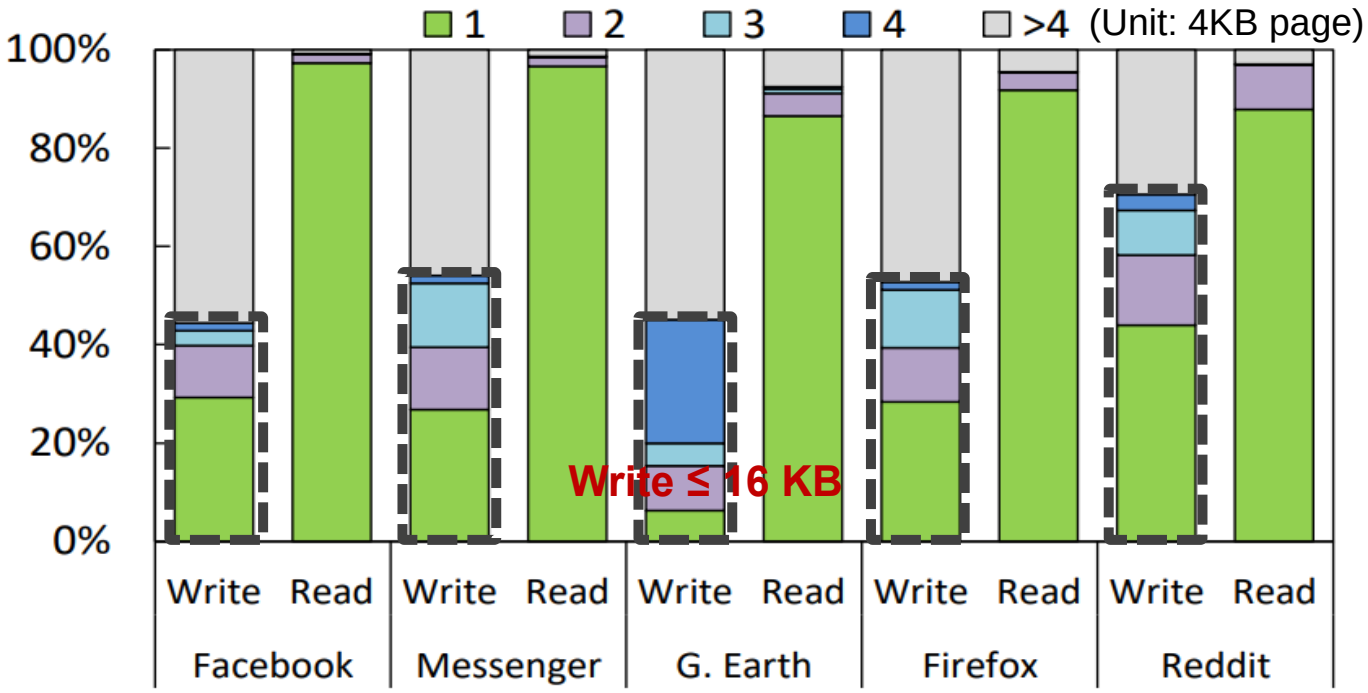
1. *Insufficient storage space*
2. *Excessive write traffic*

Poor Performance and Lifetime

How to perform the compression to solve the above problem

Read and Write Pattern

◆ File writes and reads of mobile application



Two Main sources of reads/writes
SQLite and *.apk files

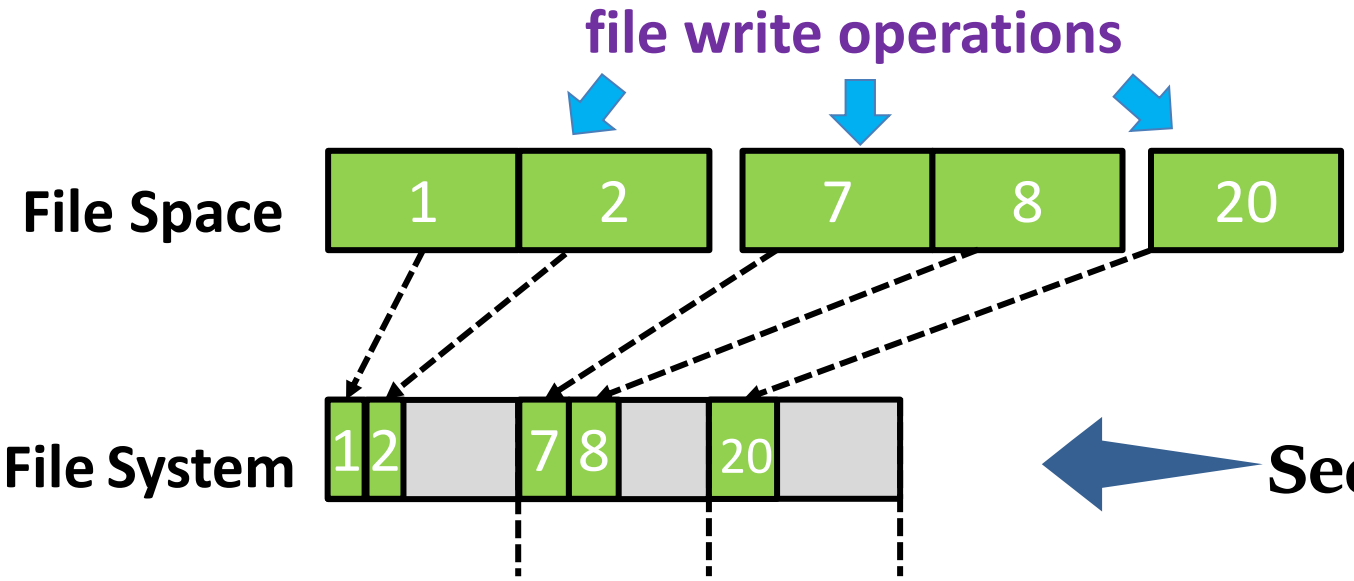
1 50% writes ≤ 16 KB

2 90% reads were one page

File read/writes were small and bound for fragmented file offsets

Limitation of Previous Compression

◆ Compression file system: JFFS2 (read-write)



Compression

Sequential compression method

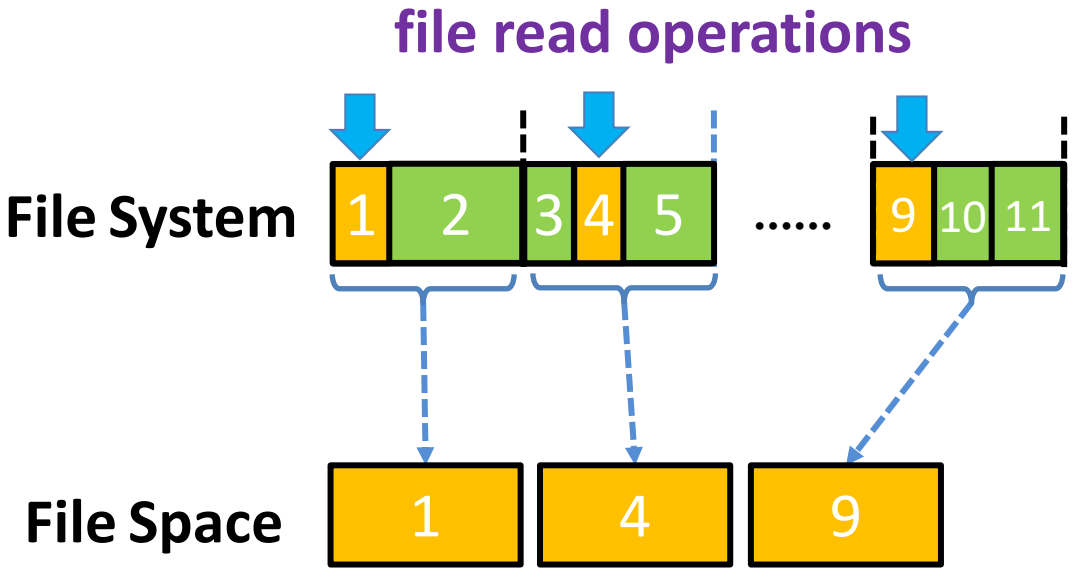
- File blocks of consecutive offsets are compressed in the same storage block

Poor space utilization

Sequential compression on small file writes suffered from ineffective space saving

Limitation of Previous Compression

◆ Compression file system: EROFS (read-only)



Sequential compression method

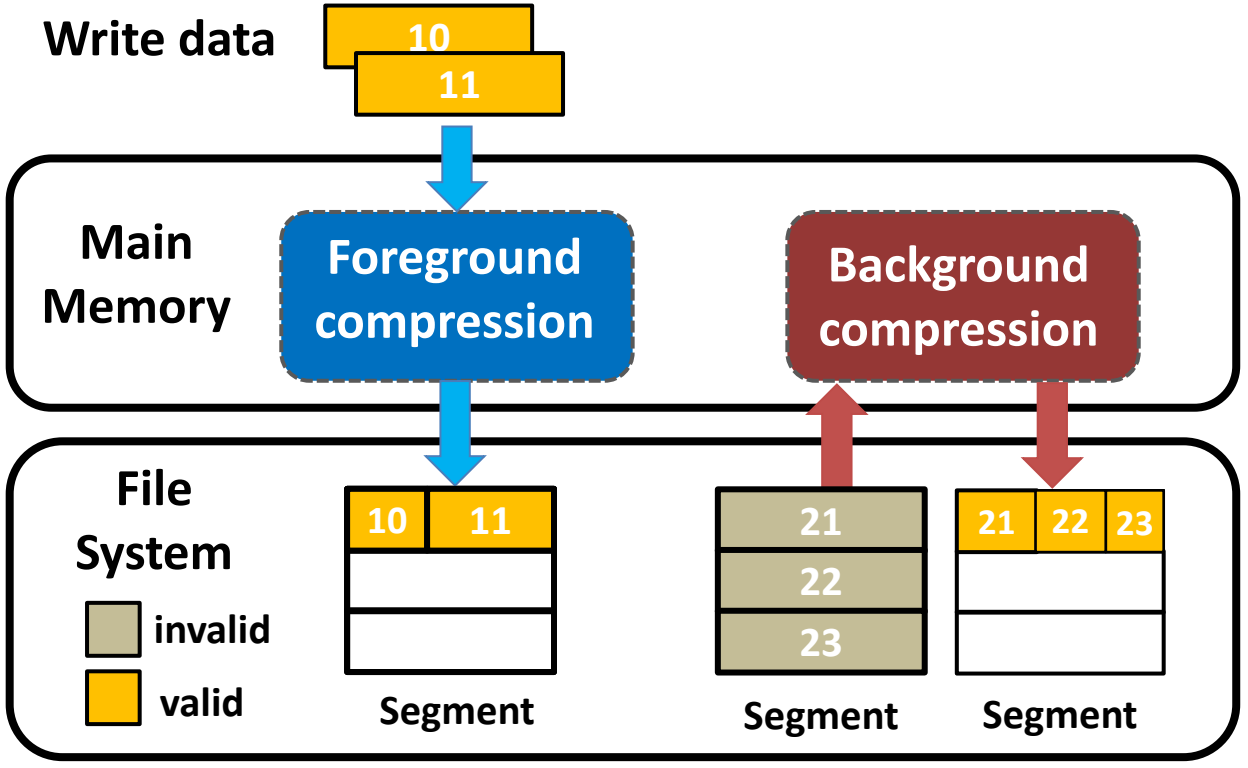
- Need to decompress unnecessary data

Decompression

Increased block read frequency

Decompression on small file reads after sequential compression amplifies the read overhead

File Pattern-Guided Compression (FPC)



Foreground compression

- Address **small file writes**
- Reduced write pressure

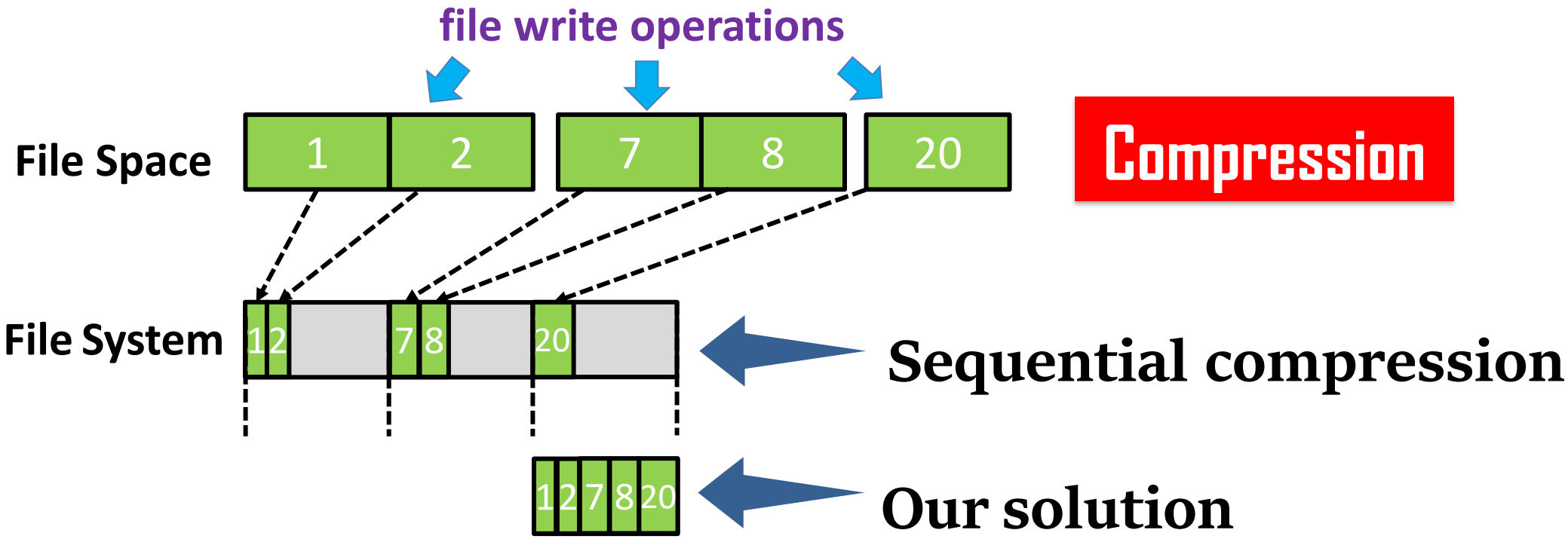
Background compression

- Address **small file reads**
- Reduced space pressure and read penalty

Highly optimized for mobile I/O patterns

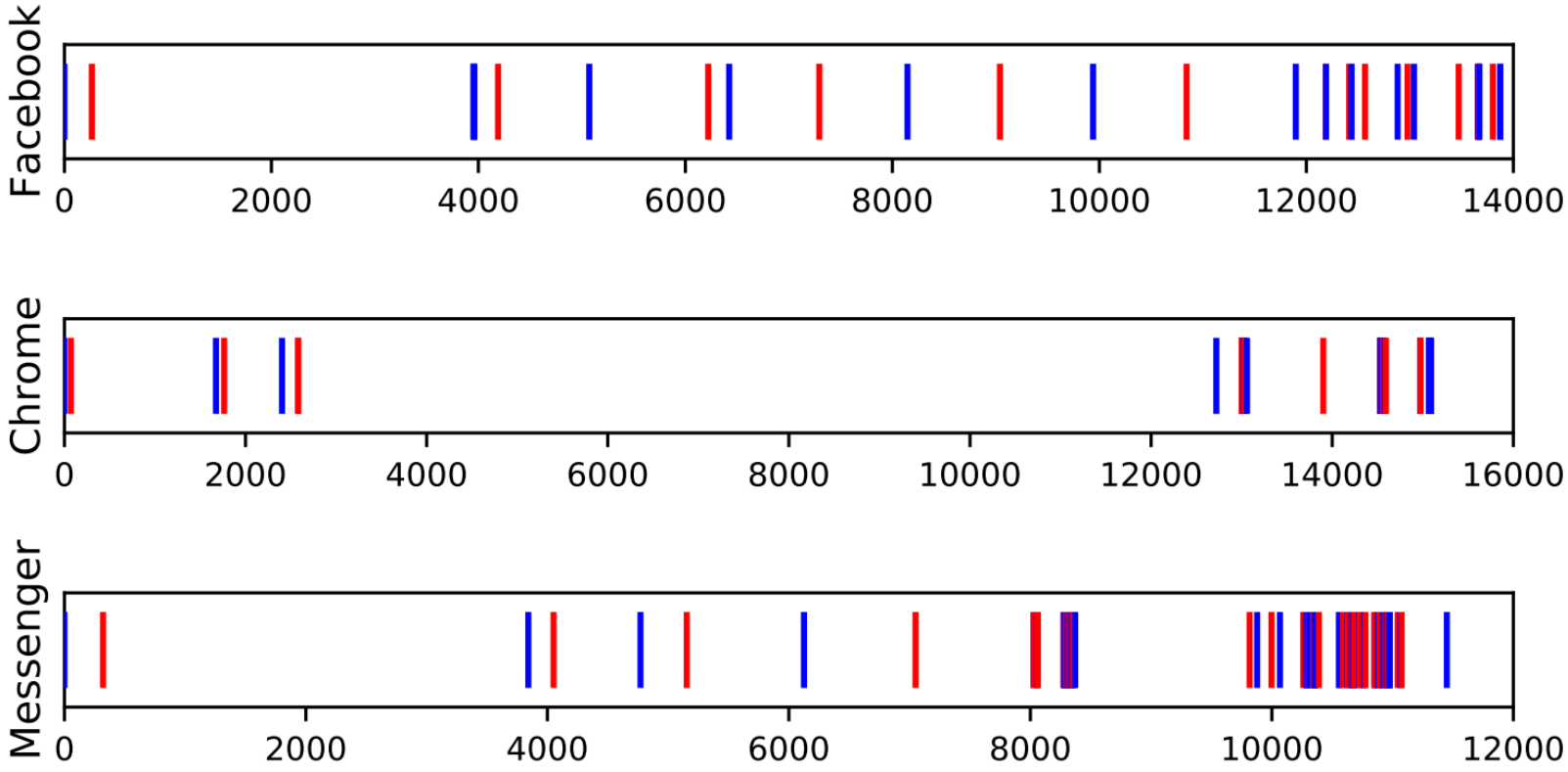
Foreground Compression

- SQLite produces many small writes with random file offsets
- Allow **random writes** to be stored in the same physical block



Background Compression

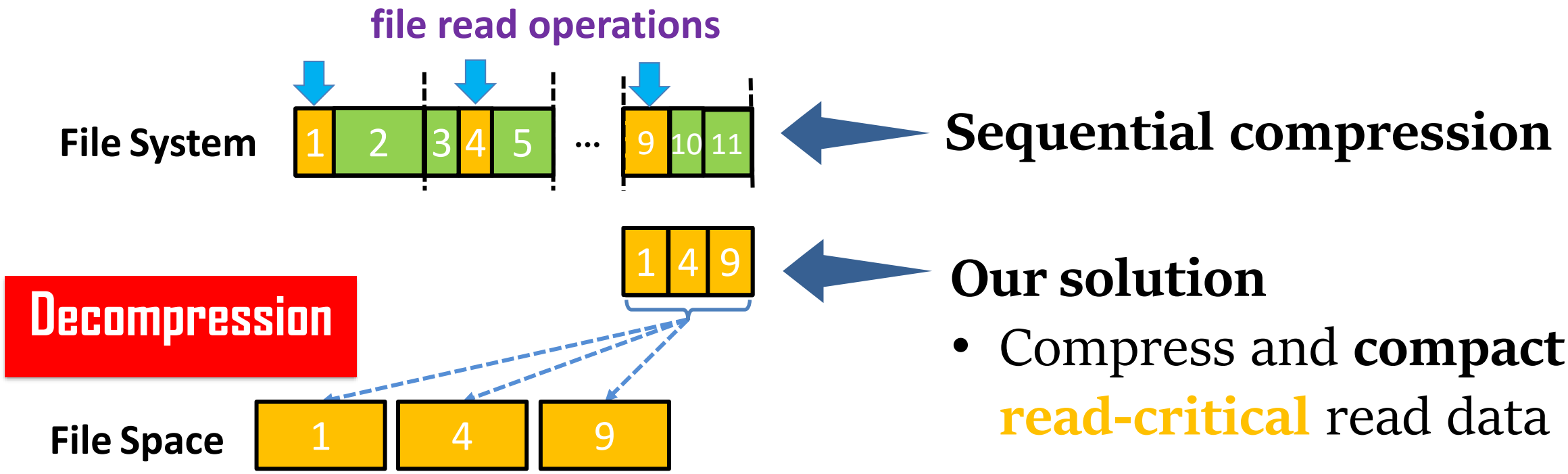
- Compression can save space but **read penalty** is unacceptable
  - **Fragmented reads** on executable files during app launching
  - Highly predictable read pattern for app launching



Read-critical data: exactly required to launch an application in executable files

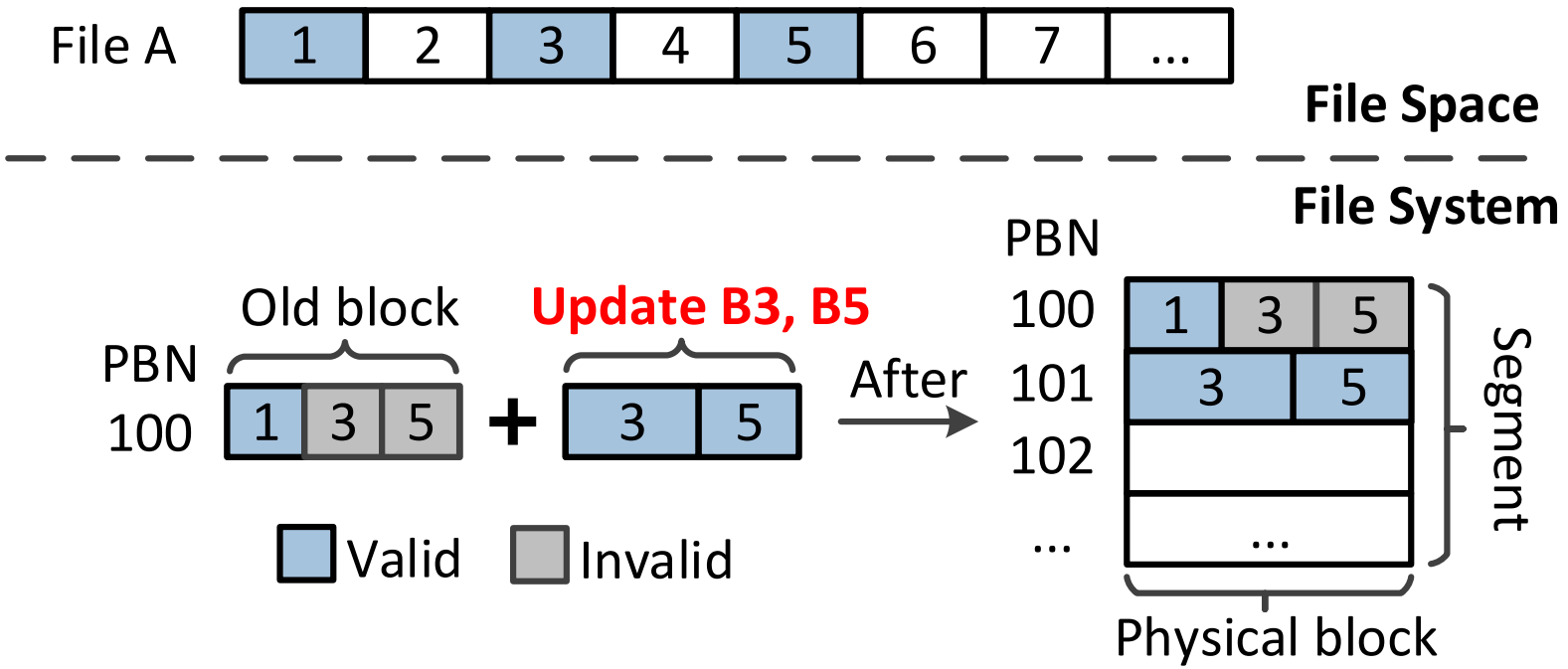
Background Compression

- Compression provides an opportunity reorganizing necessary file blocks
  - Reshape the read patterns for better decompression efficiency



Implementation with Log-structured File System

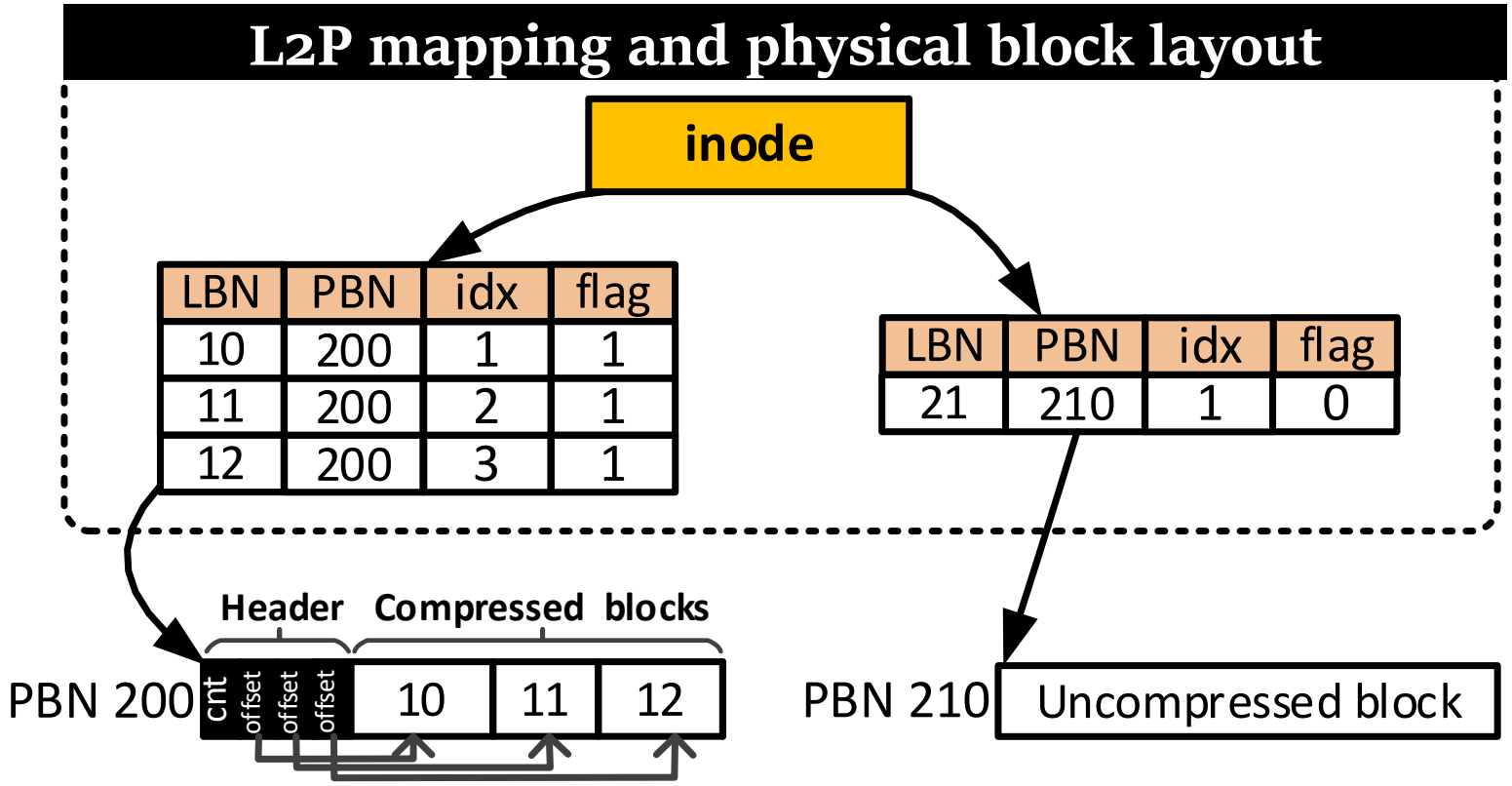
- PFC is implemented based on **F2FS** (LFS for mobile devices)
  - **Compression-friendly** when out-of-place updating compressed data
  - Avoid write amplification due to changed compression ratios



Updating the compressed blocks (B3 and B5) on LFS

Enhanced File Indexing

- **Challenge:** Mapping of compressed logical blocks
- **Solution**: Extend the **logical-to-physical** (**L2P**) mapping table



Augment direct pointers in inode/direct node:

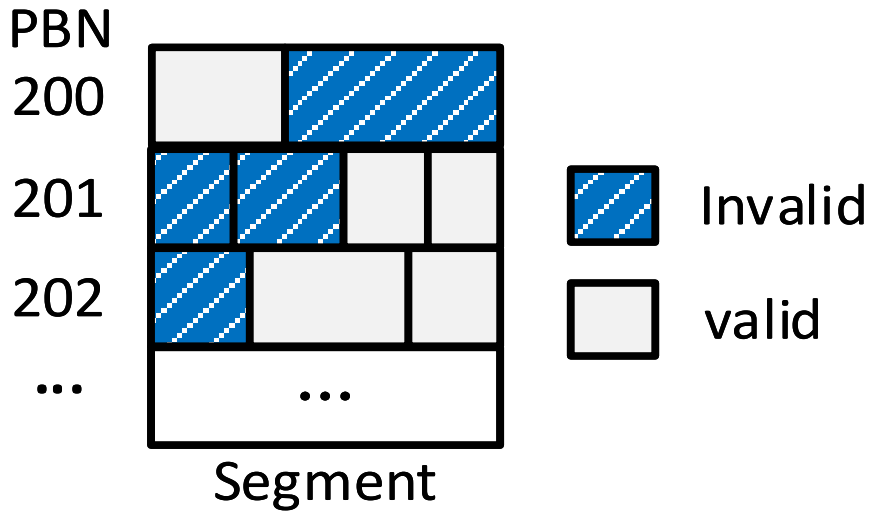
Largest file size decreases from 3.94 TB to 3.50 TB

Enhanced Block State Tracking

- **Challenge**: Partially invalidated after write operations
- **Solution**: Track the **valid/invalid status** of compressed blocks

PBN	Invalid num	Bitmap
200	1	1 0 X X X
201	2	0 0 1 1 X
202	1	0 1 1 X X
...	...	...

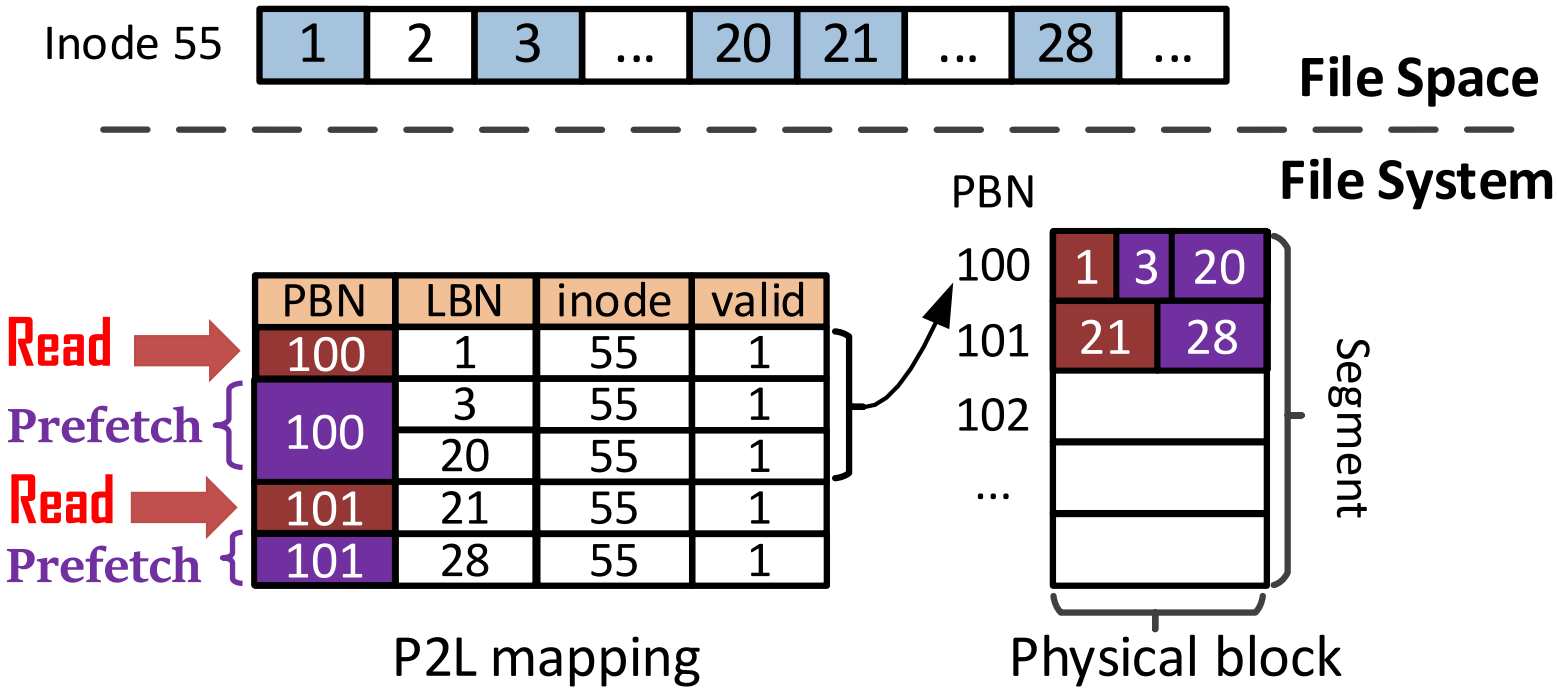
Block State Table



- **Five compressed file blocks in one physical block**
  3 bits for counting invalid compressed blocks
  5 bits for block invalid bitmap
- **For a 16 GB storage space, space overhead is 4 MB**

Decompression with P2L Mapping

- Exploit the P2L mapping of LFS for **decompression speedup**
  - Decompressing all the compressed blocks in the same physical block together



Decompression speedup for read-critical blocks

Experiment Setups

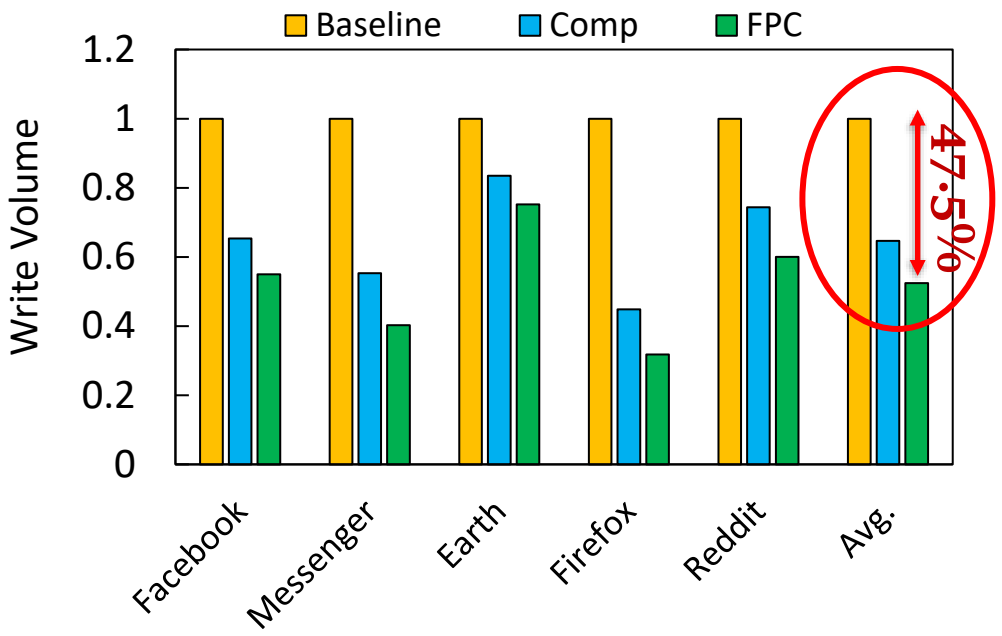
- **Prototype on a mobile platform Hikey 960**
  - 8-core ARM processor, 4GB of RAM, and a 32GB UFS
  - The Android and Linux kernel versions were 9.0 and 4.9
  - File System: ***F2FS;*** Compression algorithm: ***LZO;***

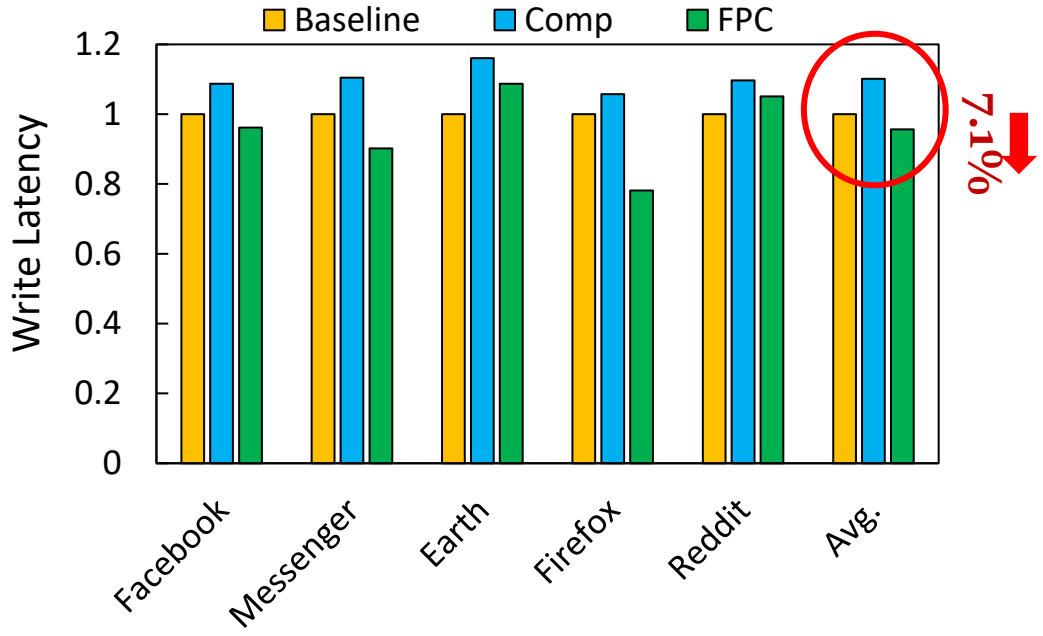
- **Three related methods were evaluated**
  - Original F2FS (*Baseline*), conventional compression (*Comp*), the proposed *FPC*

- **The evaluation was based on a set of popular mobile applications**
  - App scenarios: SQLite writes for FC, application launching for BC
  - Metrics: Write traffic/latency, launching times, file size, etc.

Experimental Results

◆ Results of SQLite write volume and write latency



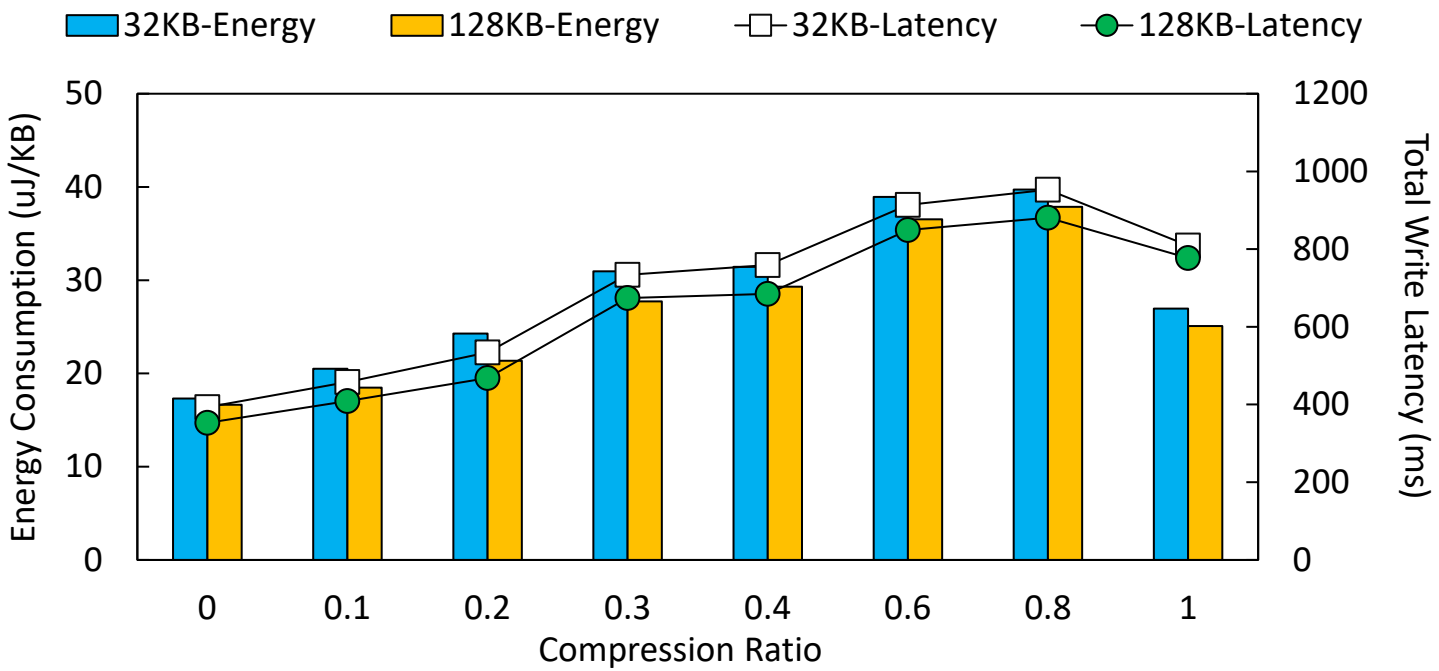


1. Compressing random write data tightly **improved the write reduction**

2. A large write reduction with compression is beneficial to **reducing write latency**

Sensitive Study on Compression Ratio

◆ Sensitive study of write latency and energy consumption



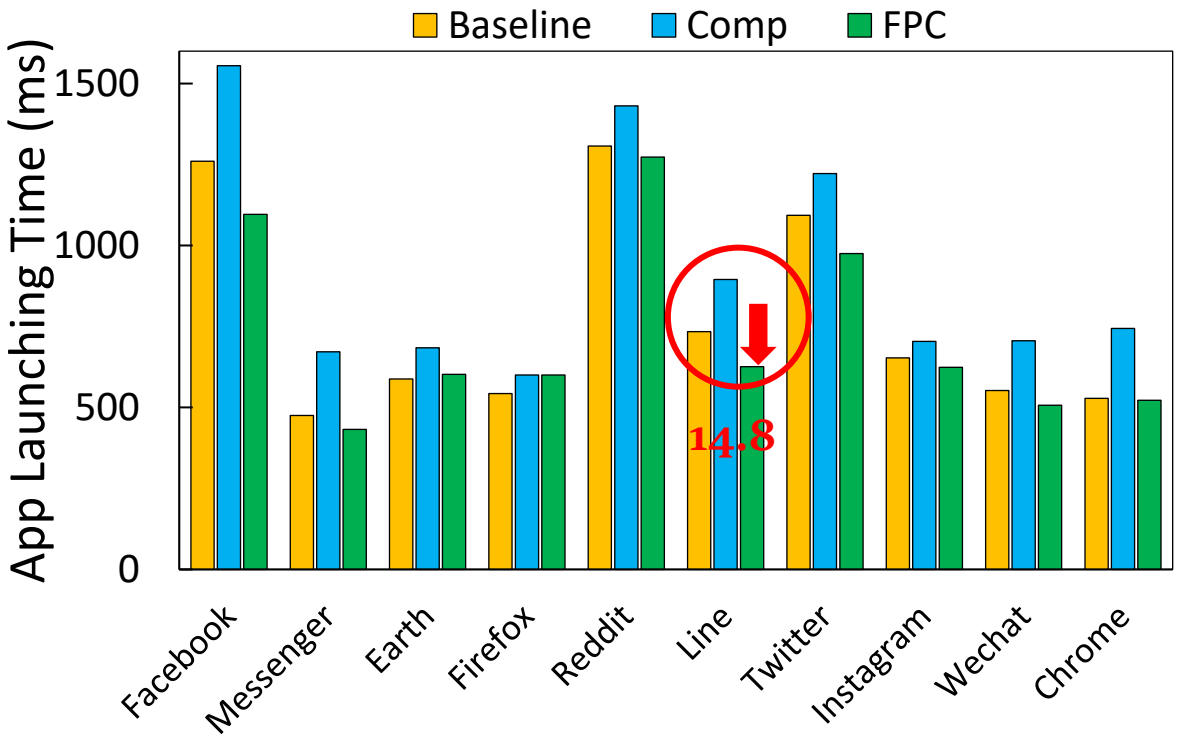
Data are not compressed for compression ratio of 1

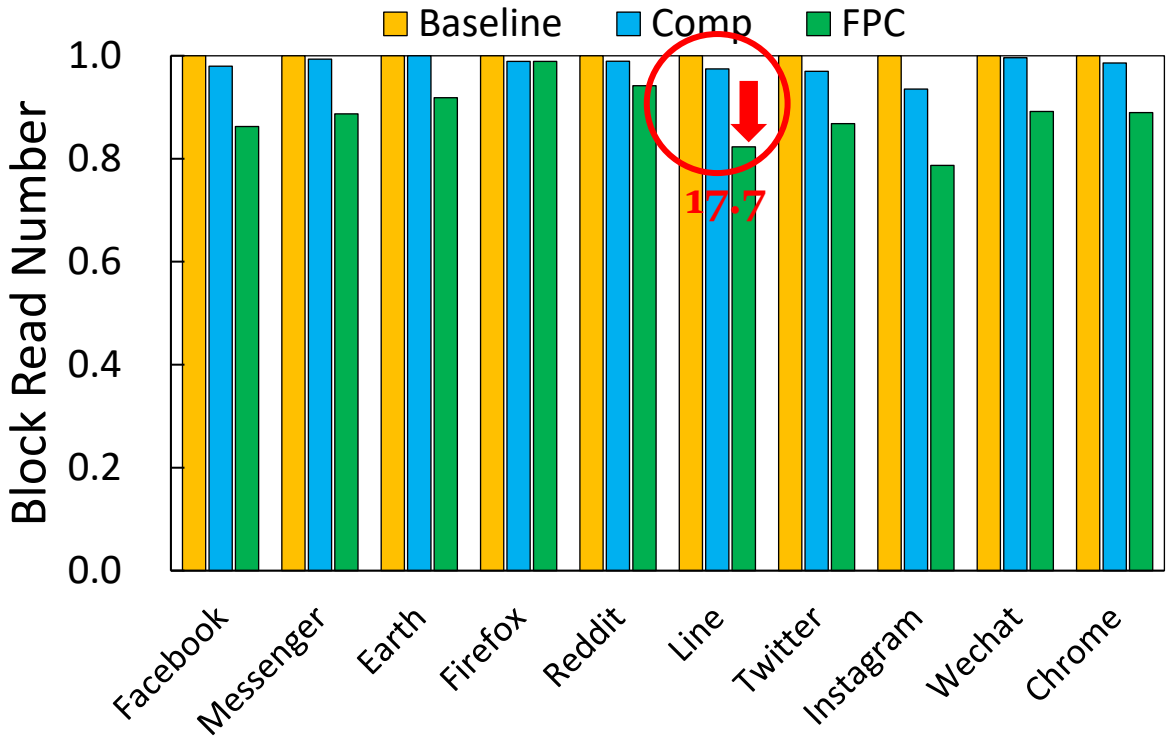
When data were highly compressible, compression benefited both write latency and energy consumption

Experimental Results

◆ Results of perceived latencies

- Block read count and application launching time

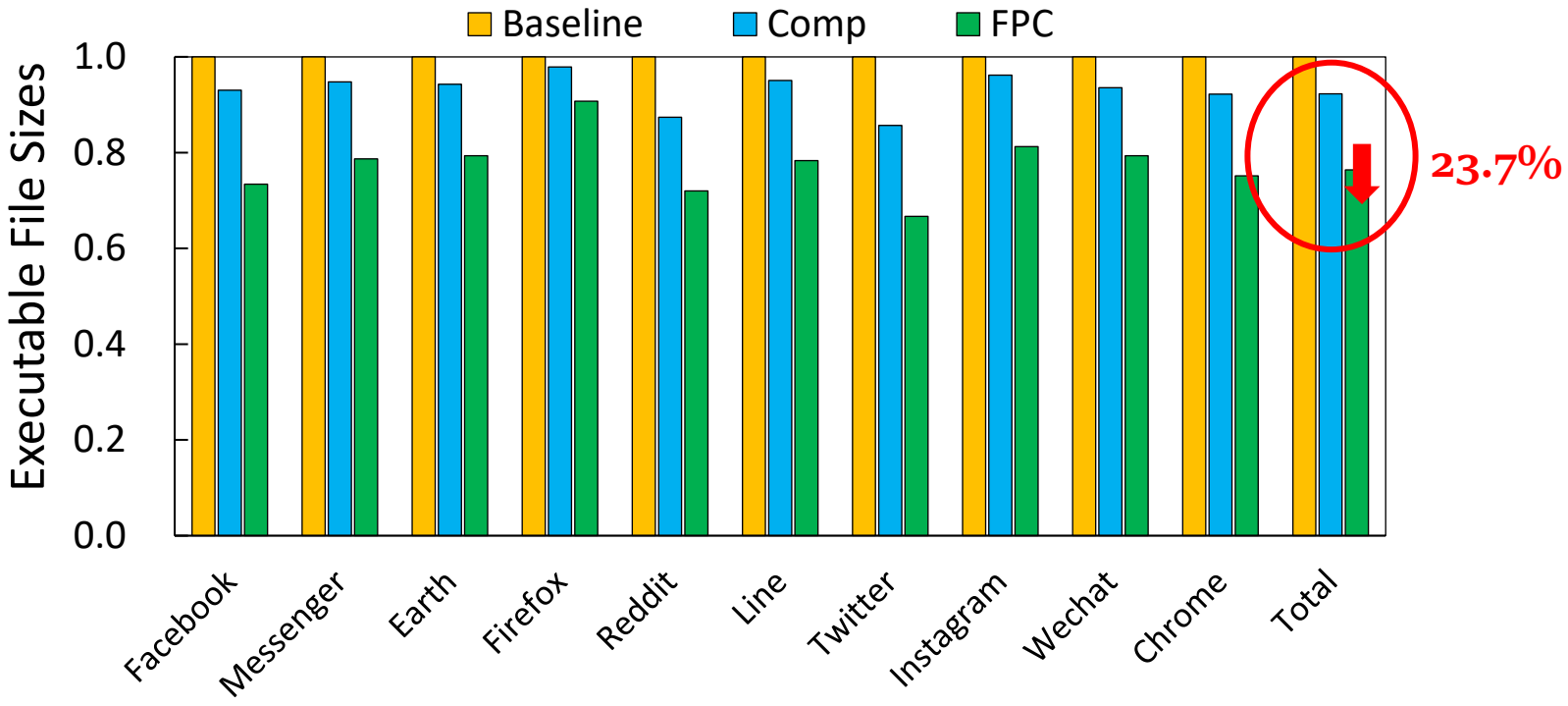




Read-critical data are compacted/compressed leading to a fewer number of block read

Experimental Results

◆ Results of total file sizes



The total size of executable file was reduced from 846 MB to 646 MB

FPC outperformed the Comp by allowing large compression windows

Conclusion

- I/O pattern of mobile devices necessitates the optimization of compression policies for enhanced compression efficacy

- Prototyped the **FPC** on the real mobile platform
  - Proposed a **foreground compression** to compress online incoming small writes
  - Proposed a **background compression** to compress offline data with considering read performance

- Showed the advantage of FPC against previous approaches

Thank you!

cheng.ji@njust.edu.cn

Pattern-Guided File Compression with User-Experience Enhancement for Log-Structured File System on Mobile Devices

Cheng Ji [§], *Li-Pin Chang* [※], *Riwei Pan* [♯], *Chao Wu* [♯], *Congming Gao* [*], *Liang Shi* [¶], *Tei-Wei Kuo* [♯], *Chun Jason Xue* [♯]

[§] Nanjing University of Science and Technology [※]National Chiao Tung University
[♯]City University of Hong Kong [*]Tsinghua University [¶]East China Normal University