STORAGE DEVELOPER CONFERENCE

SDC 22

Fremont, CA
September 12-15, 2022

BY Developers FOR Developers

A SNIA. Event

Improving Flash Storage on Android Phones

Presented by Tejas Chopra

Agenda

- About me
- Android I/O & performance issues
  - Android stack is not flash-friendly
- Proposed solutions
  - Changing journaling mode for SQLite DB
  - Exploring different file systems
  - Using fdatasync
  - Converting small random to large sequential
- Results
- Takeaways

About me

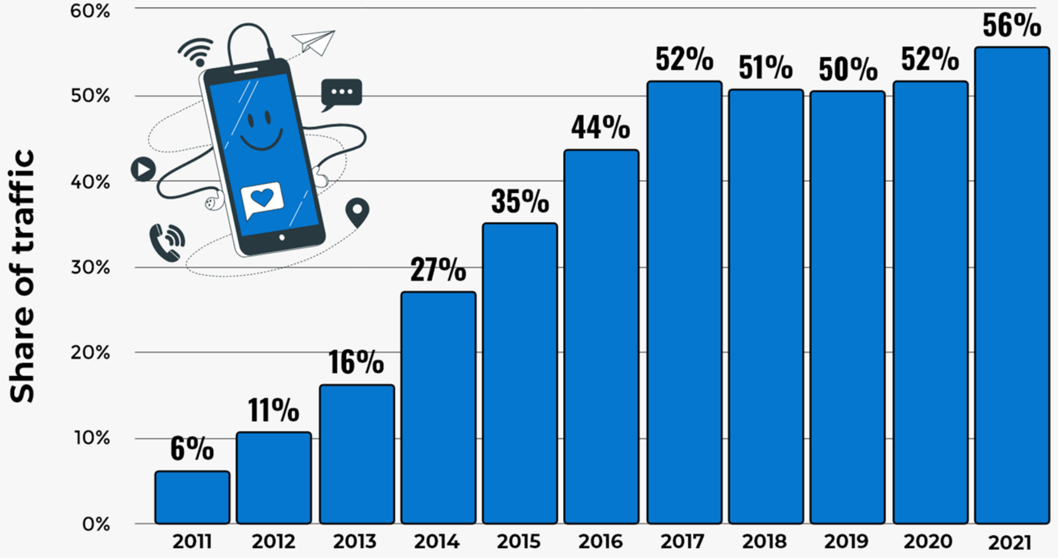
About me

- Sr. Software Engineer, Netflix
- Apple, Samsung, Cadence, Box
- TedX Speaker
  - Cloud computing
  - Storage, Distributed Systems
  - Blockchain, Web3, NFTs
- Advisor
  - Nillion
  - Dorado
- Adjunct Professor, UAT, AZ

Android IO & Performance issues

Smartphones are ubiquitous

Percentage of Global Mobile Traffic, 2011-2021

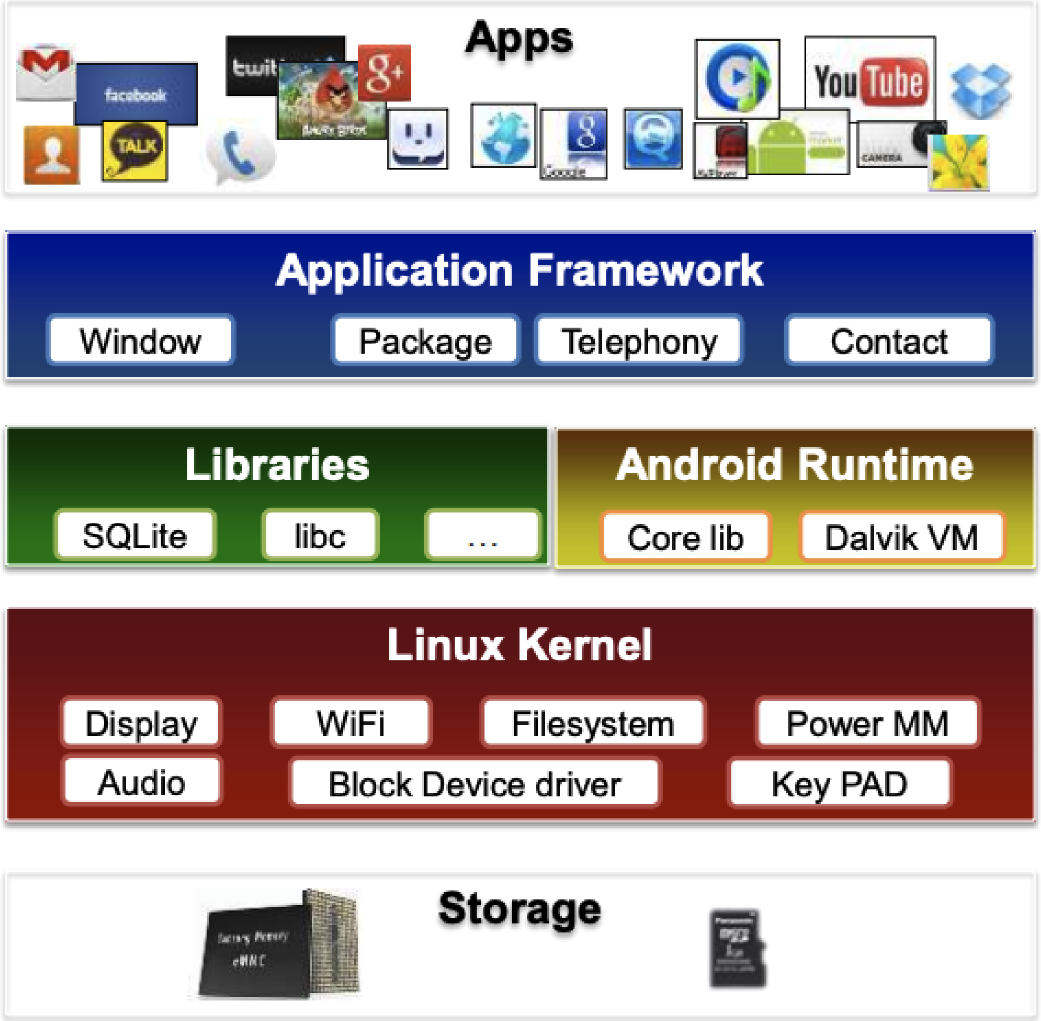


Social Media Sites: Percentage of Usage on Mobile Devices

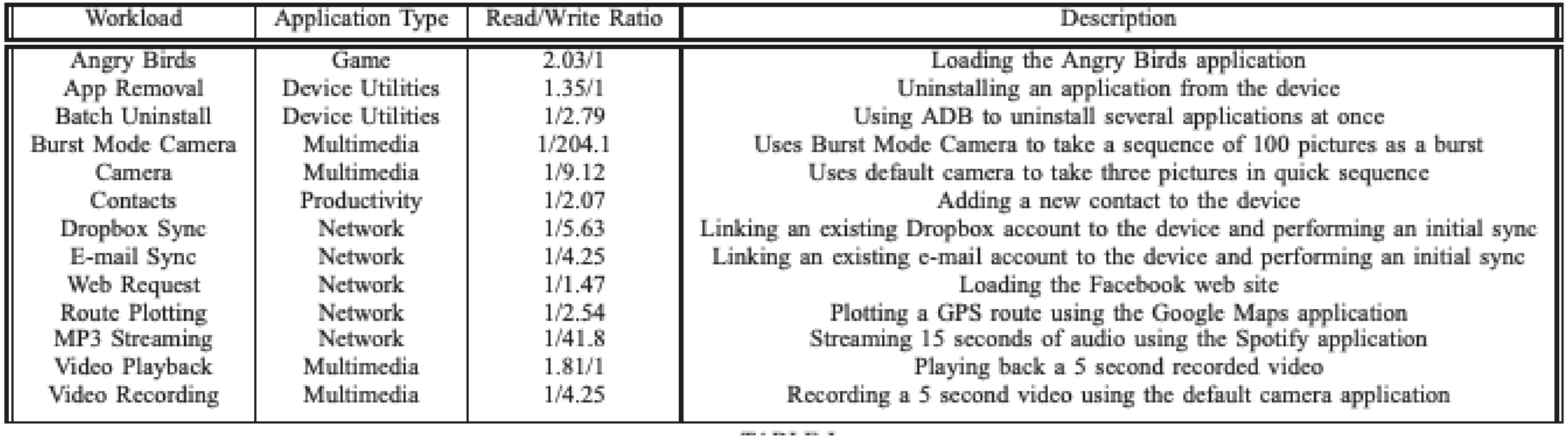
- Smartphone Usage Makes Up 80% of Social Media Browsing
- Of Facebook Users, 95.1% Use a Smartphone for Access
- 86% of Twitter Usage is on a Mobile Device
- 60% of LinkedIn Usage is from Mobile Devices

Storage IO is the bottleneck in performance

Android Platform

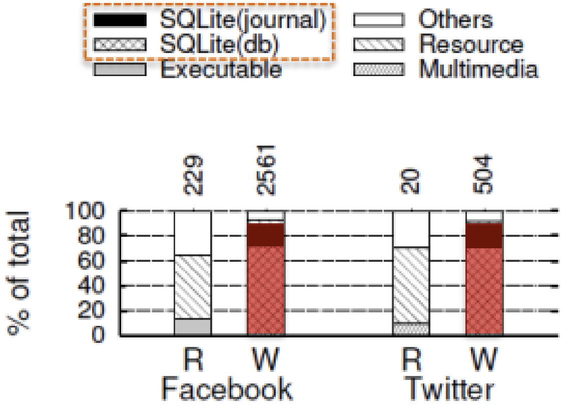


Studying common application patterns

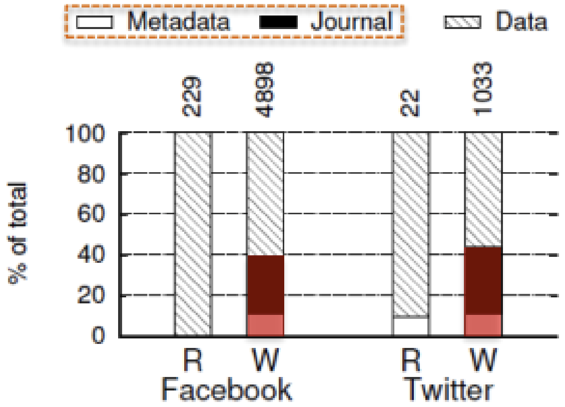


Workload	Application Type	Read/Write Ratio	Description
Angry Birds	Game	2.03/1	Loading the Angry Birds application
App Removal	Device Utilities	1.35/1	Uninstalling an application from the device
Batch Uninstall	Device Utilities	1/2.79	Using ADB to uninstall several applications at once
Burst Mode Camera	Multimedia	1/204.1	Uses Burst Mode Camera to take a sequence of 100 pictures as a burst
Camera	Multimedia	1/9.12	Uses default camera to take three pictures in quick sequence
Contacts	Productivity	1/2.07	Adding a new contact to the device
Dropbox Sync	Network	1/5.63	Linking an existing Dropbox account to the device and performing an initial sync
E-mail Sync	Network	1/4.25	Linking an existing e-mail account to the device and performing an initial sync
Web Request	Network	1/1.47	Loading the Facebook web site
Route Plotting	Network	1/2.54	Plotting a GPS route using the Google Maps application
MP3 Streaming	Network	1/41.8	Streaming 15 seconds of audio using the Spotify application
Video Playback	Multimedia	1.81/1	Playing back a 5 second recorded video
Video Recording	Multimedia	1/4.25	Recording a 5 second video using the default camera application

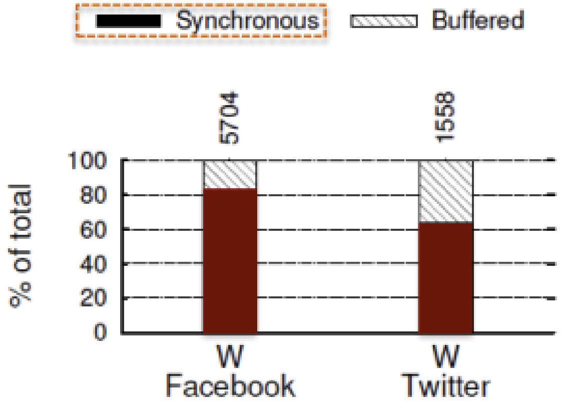
Analyzing R/W profiles



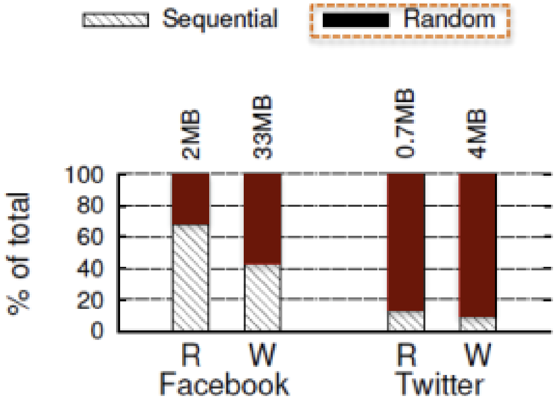
File Types



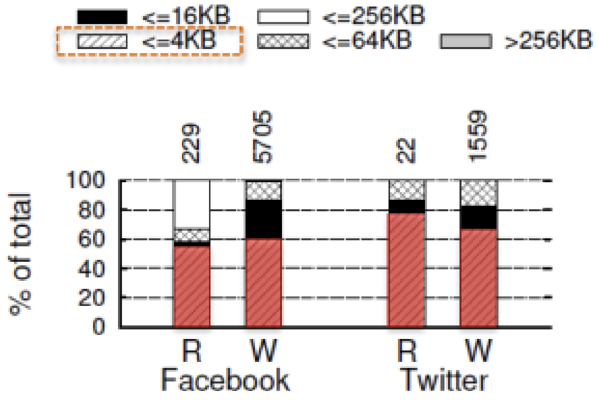
Block Types



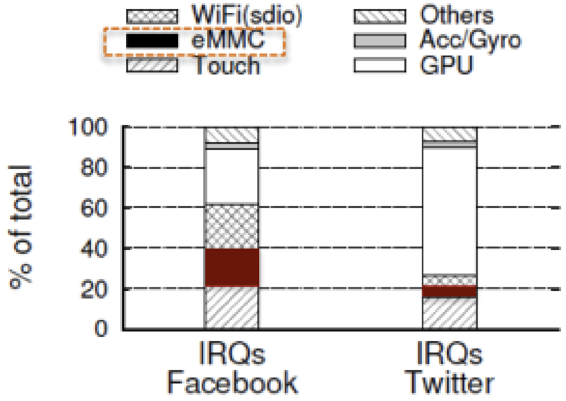
I/O Modes



Locality



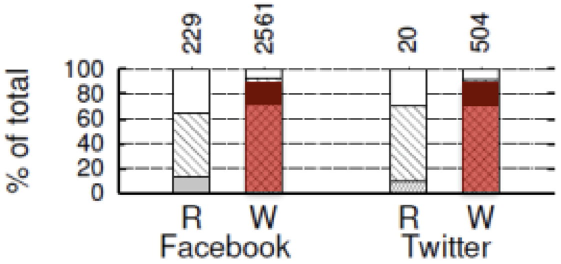
I/O Size



IRQs

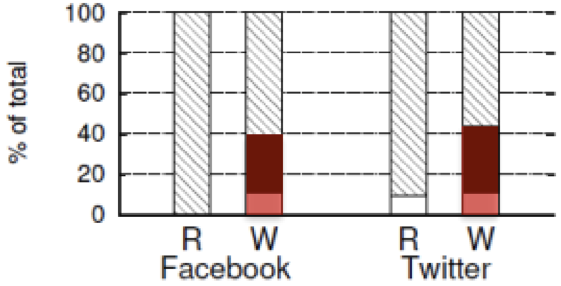
Analyzing R/W profiles

SQLite > 90%



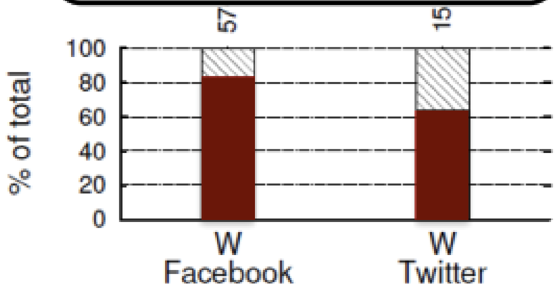
File Types

Metadata & Journal > 40%



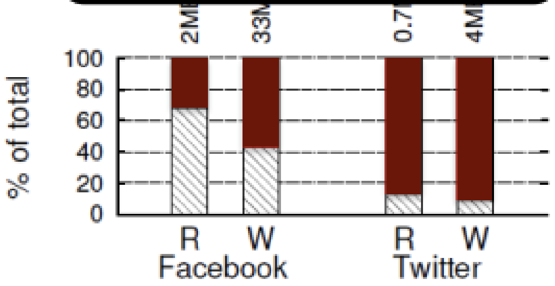
Block Types

Synchronous > 70%



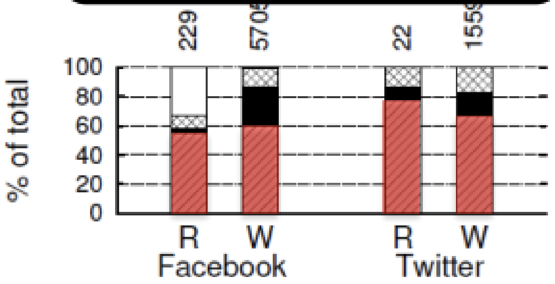
I/O Modes

Random > 80%



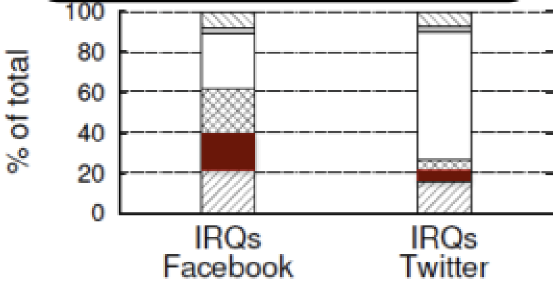
Locality

4KB I/O > 64%



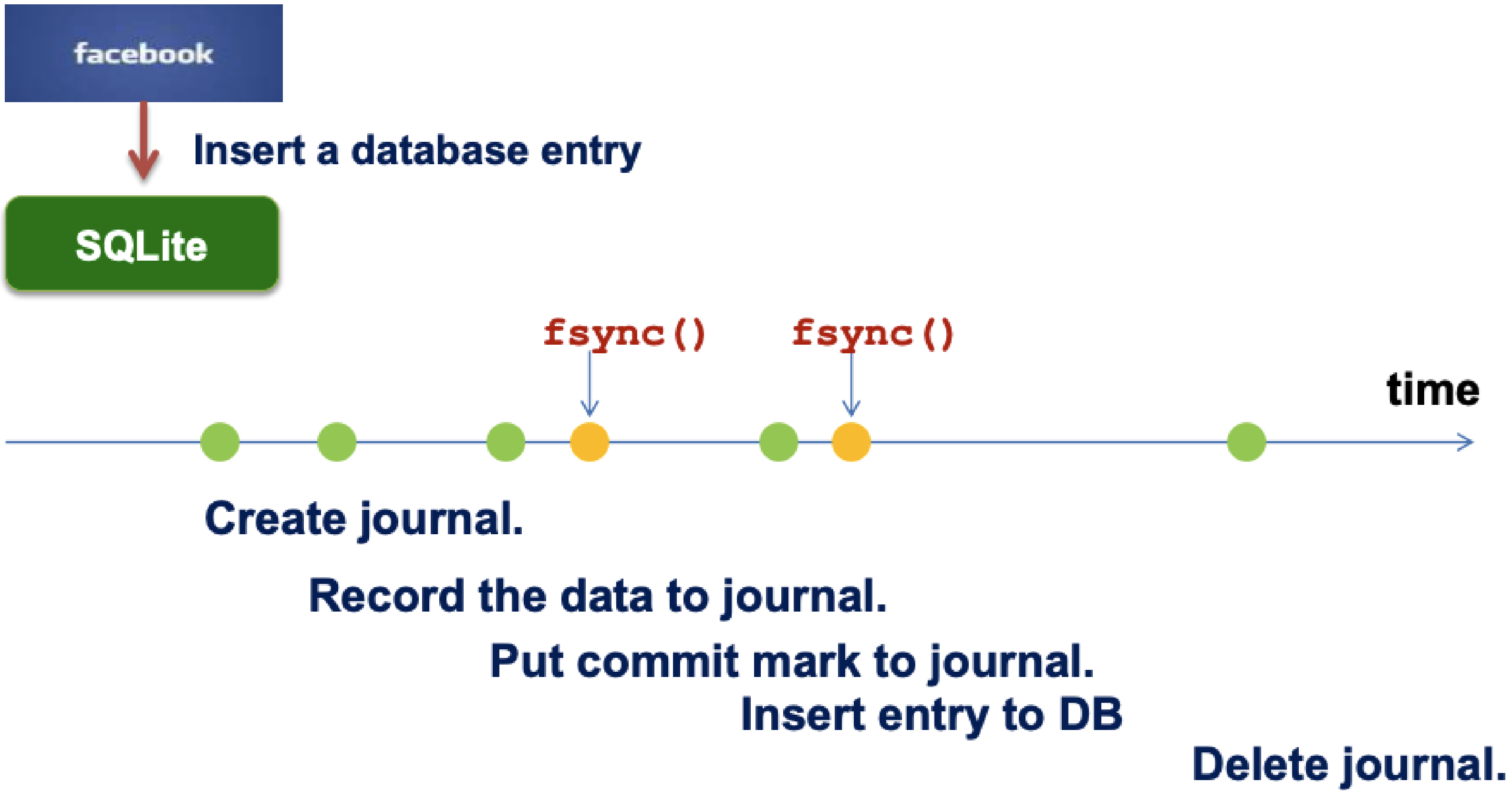
I/O Size

IRQ for eMMC > 18%



IRQs

Insert in SQLite DB



facebook

Insert a database entry

SQLite

fsync() fsync()

time

Create journal.

Record the data to journal.

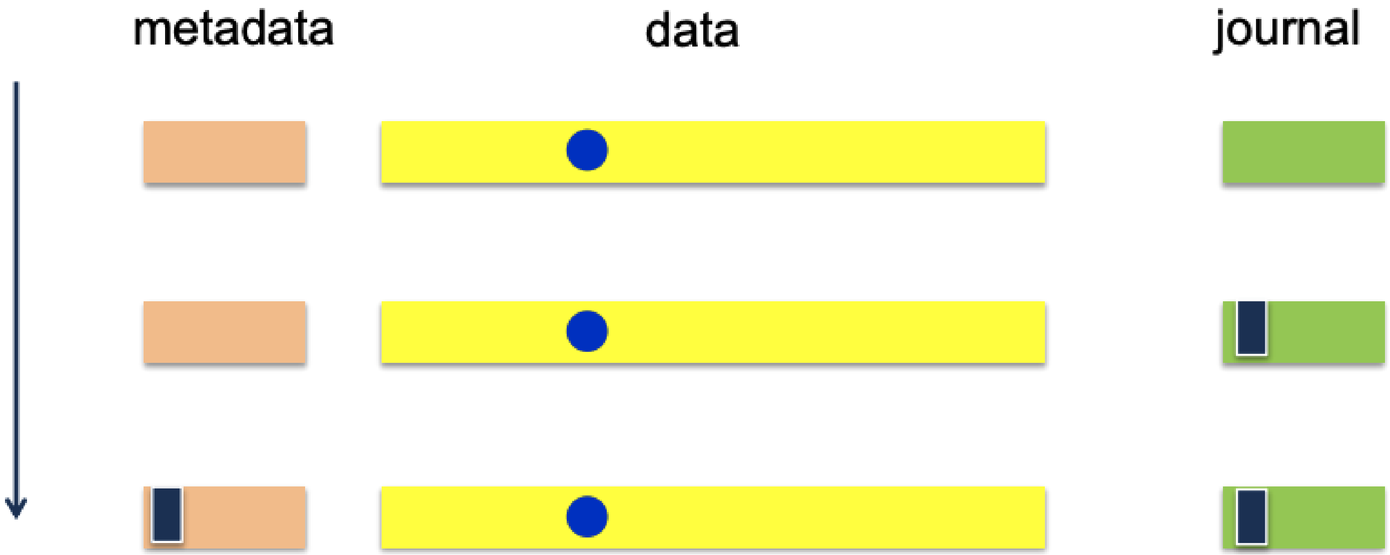
Put commit mark to journal.

Insert entry to DB

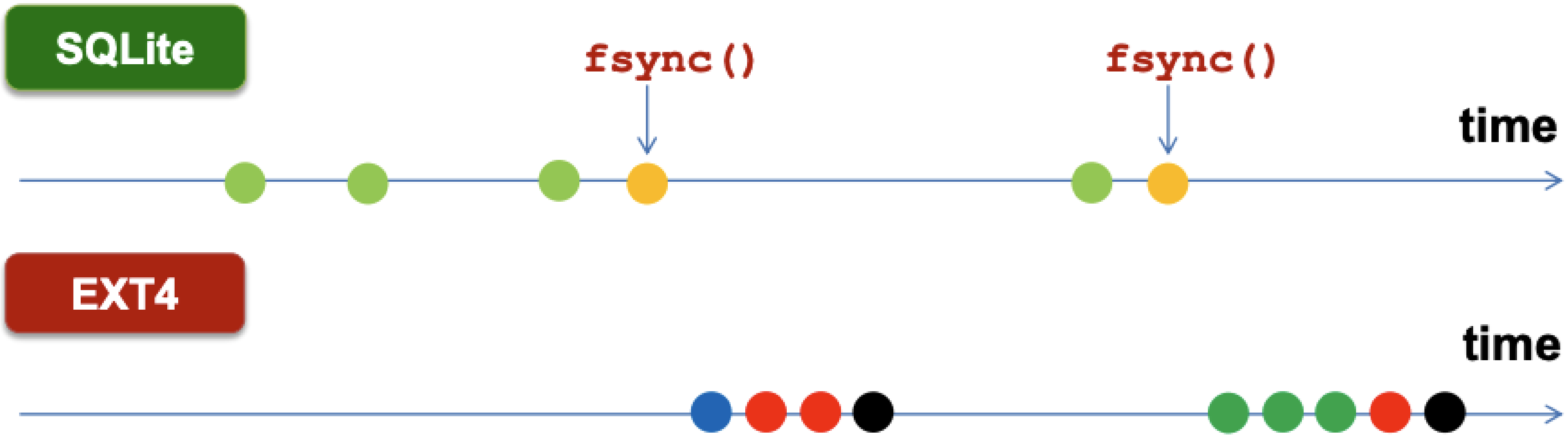
Delete journal.

Insert in ext4

```
write(fd, ● )
```

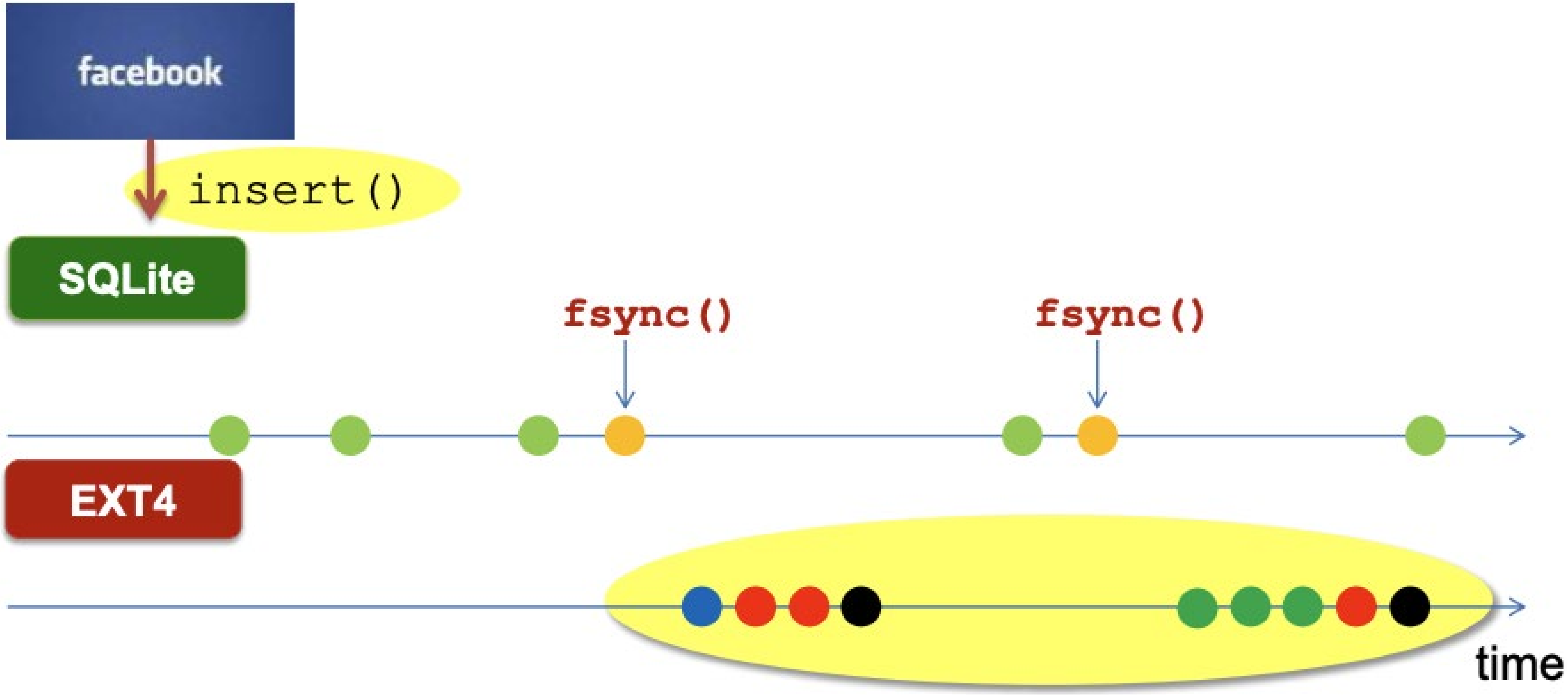


A single insert on Android



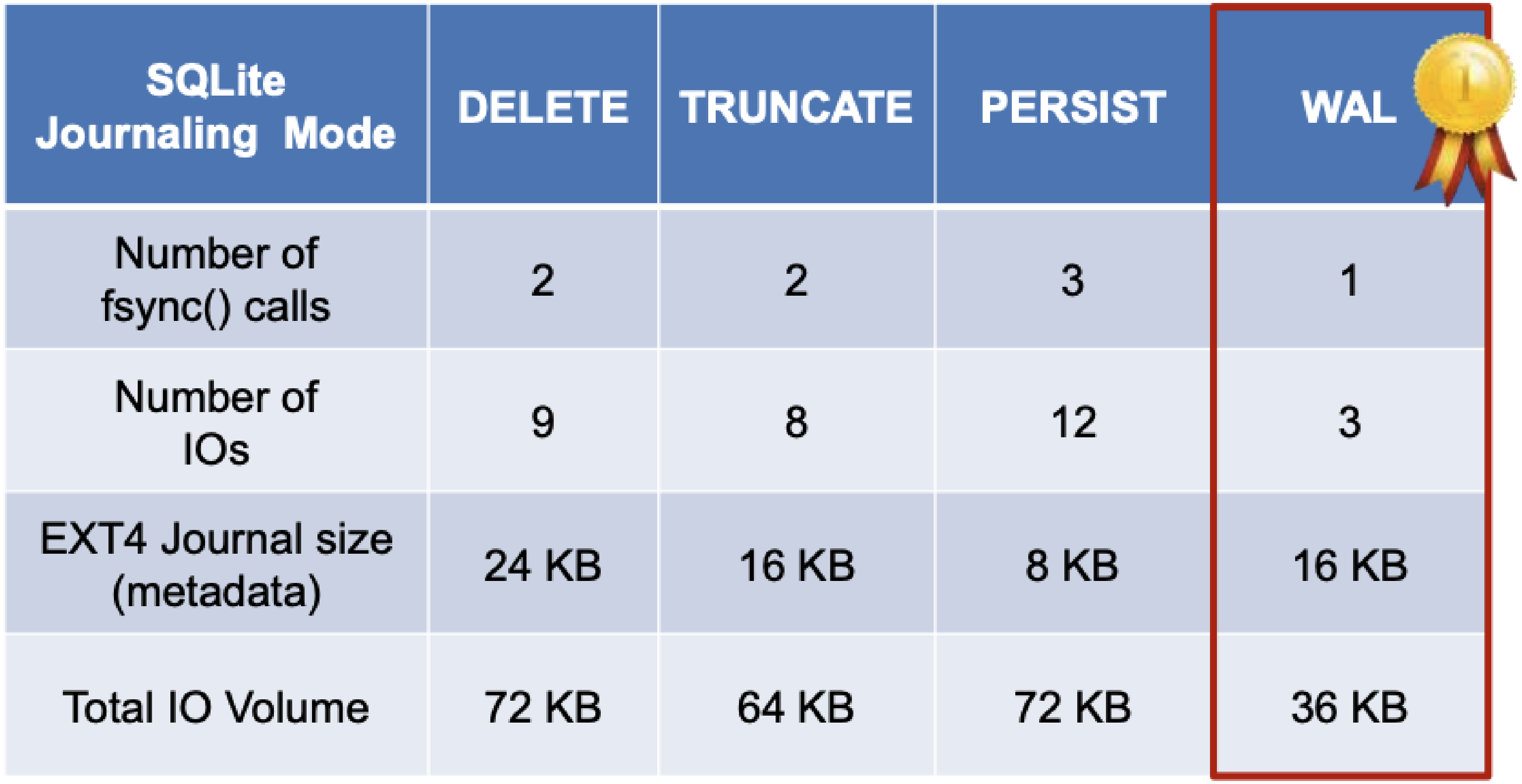
- 🔵 **write SQLite journal to storage.**
- 🟢 **write SQLite DB to storage.**
- 🔴 **write EXT4 journal (descriptor, metadata) to storage.**
- ⚫ **write EXT4 journal (commit) to storage.**

1 write = 9 eMMC writes



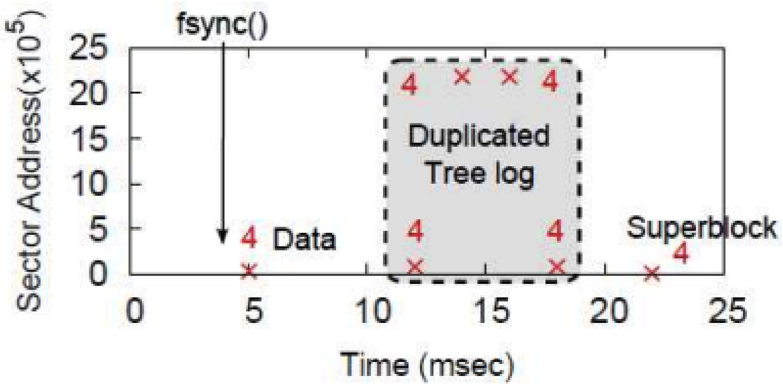
Proposed Solutions

Choosing correct journaling mode

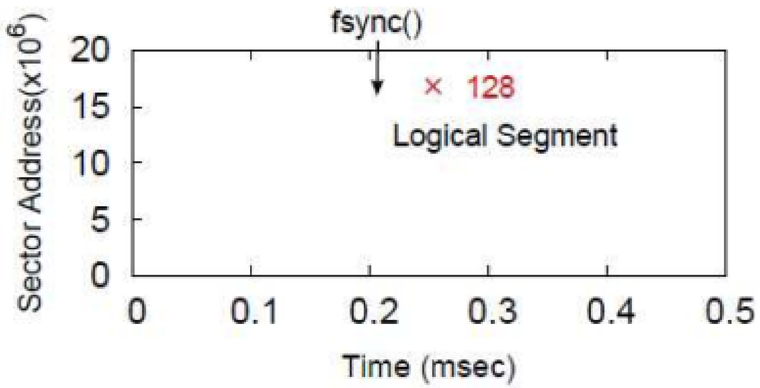


SQLite Journaling Mode	DELETE	TRUNCATE	PERSIST	WAL
Number of fsync() calls	2	2	3	1
Number of IOs	9	8	12	3
EXT4 Journal size (metadata)	24 KB	16 KB	8 KB	16 KB
Total IO Volume	72 KB	64 KB	72 KB	36 KB

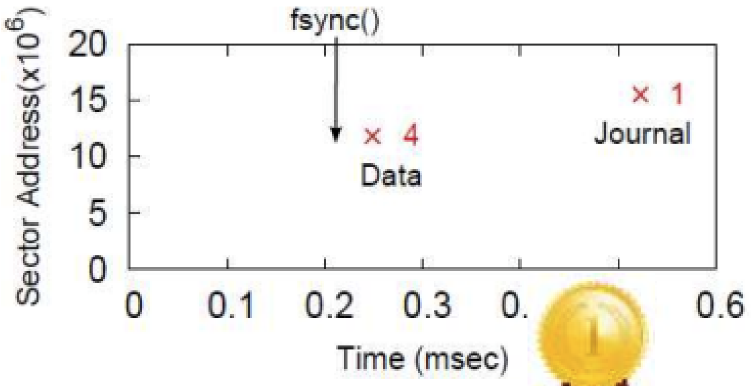
Choosing the right file system!



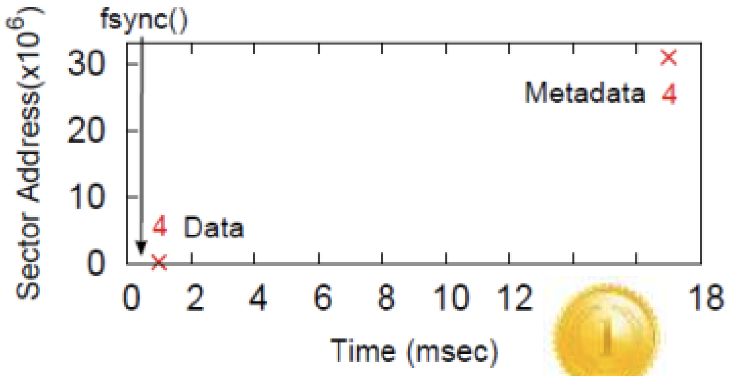
BTRFS



NILFS2

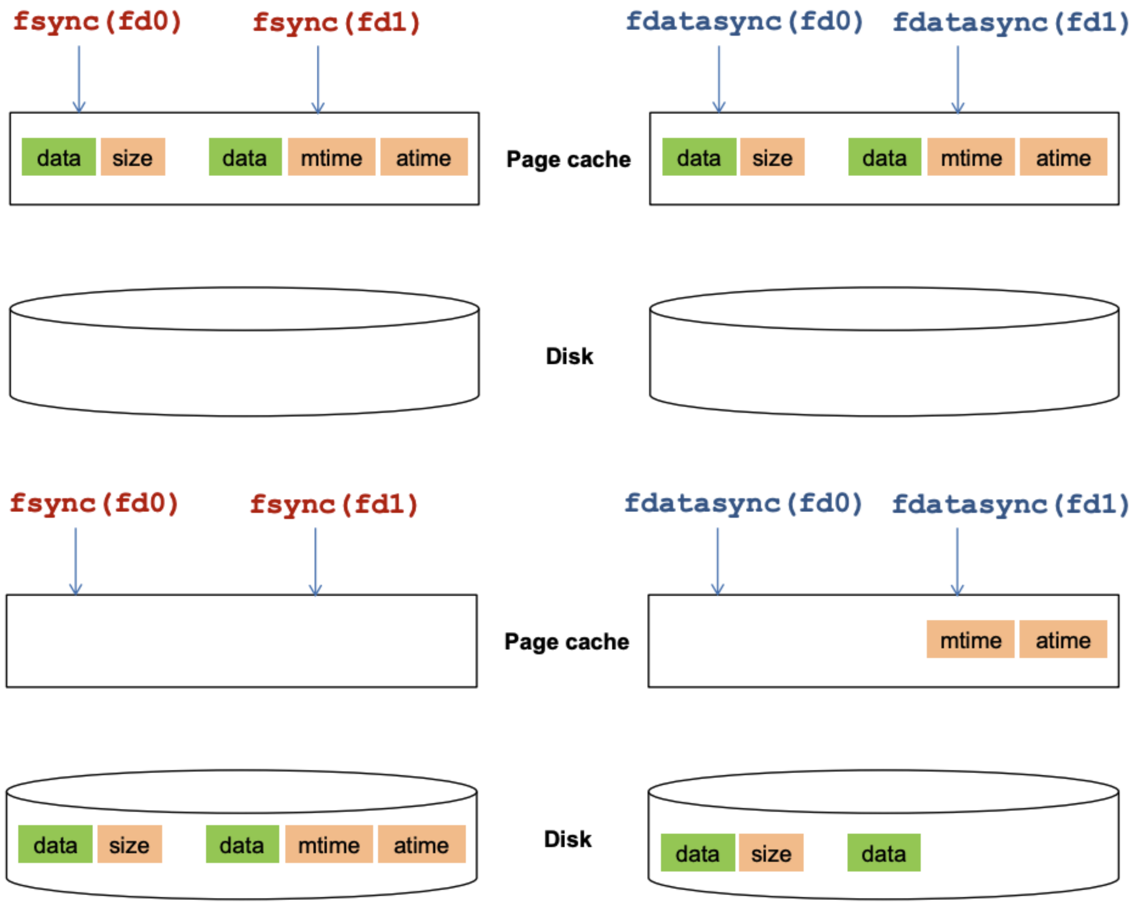


XFS



F2FS

Using fdatasync()



Small random IO → Large Sequential IO

- Small, random IO is not ideal for flash
- Convert small random to large sequential
  - Use a layer of mapping
  - Between two fdatasync() calls, collect the IOs and 'sequentialize' them
  - Write one single write to a sequential location
  - Akin to Log structuring
  - Develop a garbage collector to cleanup rewrites

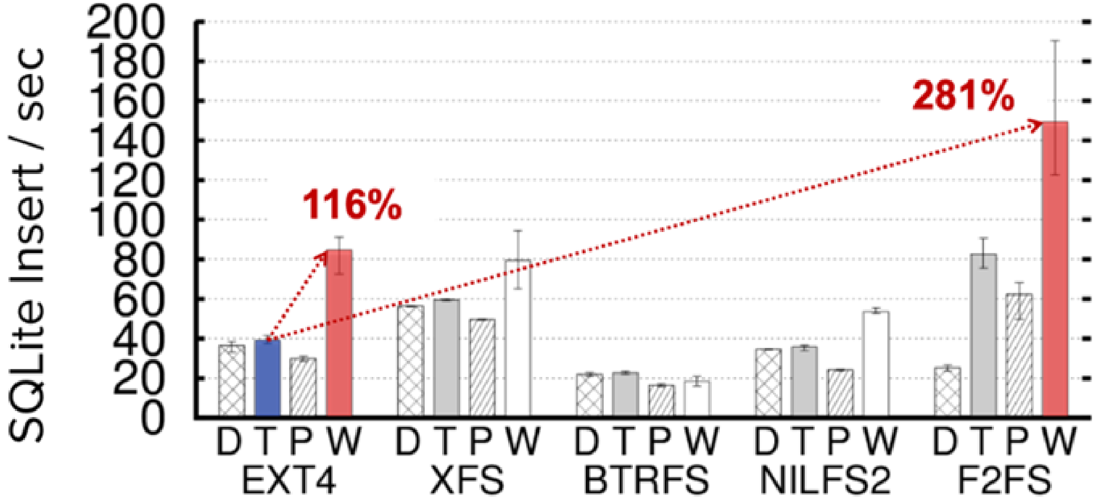
Design details

- Segment size: 1MiB (configurable)
- Log infinite, but disk finite
- Clean old segments to recover space
- Maintain segment liveness and sort it in MinPQ
- Read ‘M’ segments, and compact content in ‘N’ new segments
- Can tune auto cleaning frequency up or down- depending on the application.

Results

Results

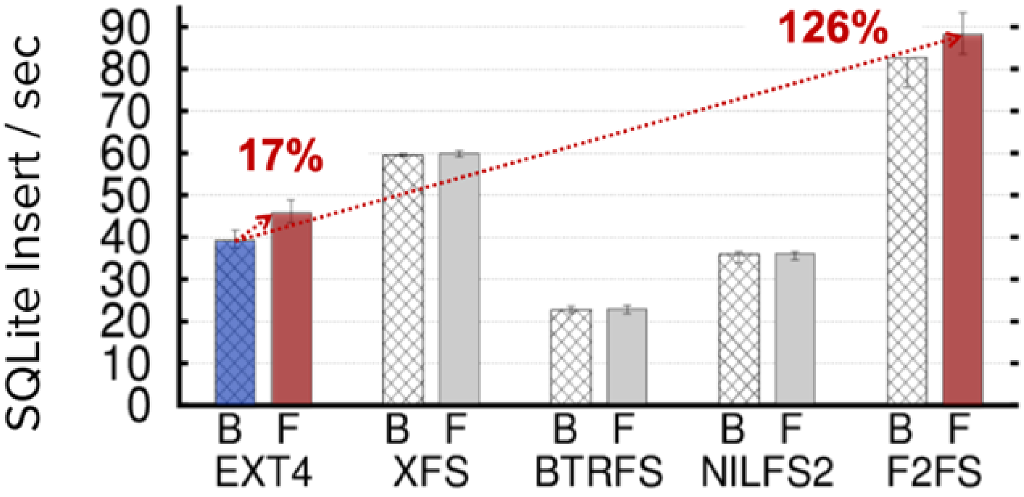
SQLite performance (with fsync()) under varying journal modes for different file systems



Journal modes for different filesystems

D: Delete
T: Truncate
P: Persist
W: WAL

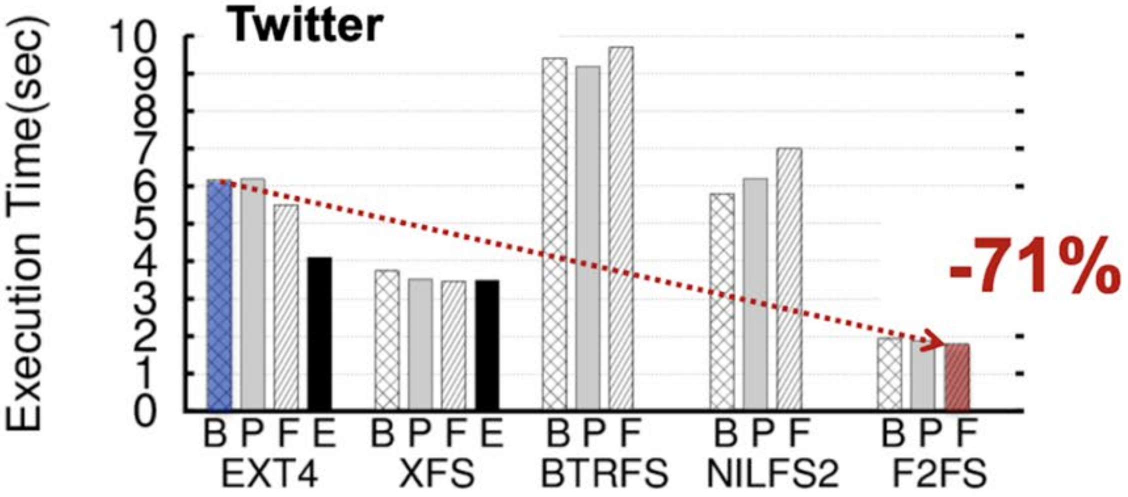
SQLite performance (with fdatasync()) under varying journal modes for different file systems



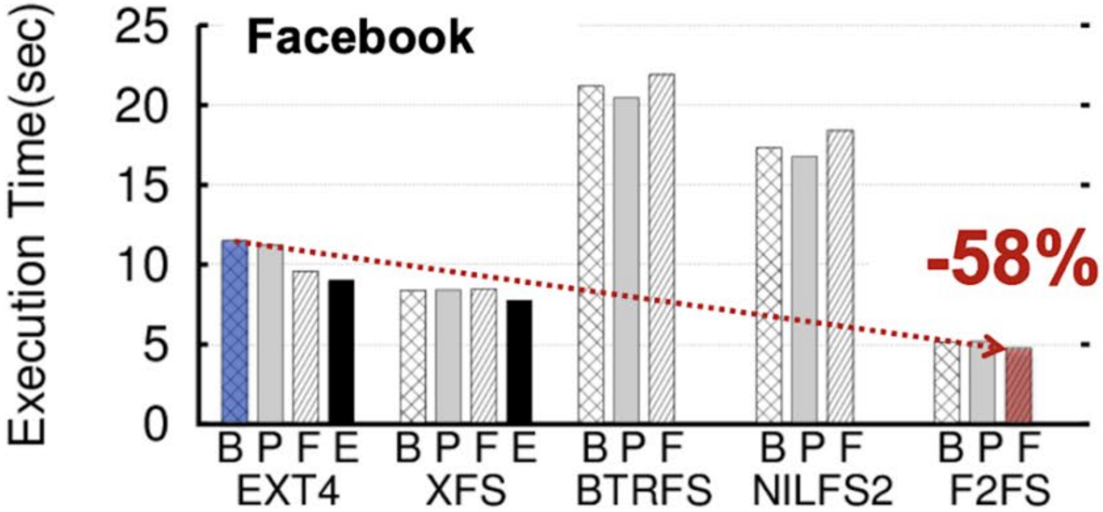
Fsync/Fdatasync for different filesystems

B: baseline
F: fdatasync()

Results



B: Baseline
P: Polling
F: fdatasync()
E: External Journal



Takeaways

Takeaways

- Existing android io stack is not optimized for flash
- Journaling of journal leads to write amplification and impacts flash life
- Understanding SQLite and ext4 behavior helps us make better choice
  - Choosing correct journaling mode (WAL)
  - Choosing correct file system (XFS, F2FS)
  - Replacing fsync with fdatasync
  - Log structuring small random writes to get better performance
- Impact - ~1.5x-3x improvement in ops/sec
- Common applications such as Twitter/Facebook are much faster

Thank You!

https://www.linkedin.com/in/chopratejas

chopratejas@gmail.com

chopra_tejas

Please take a moment to rate this session.

Your feedback is important to us.