Storage on Your Smartphone Uses More Energy Than You Think

Jayashree Mohan, Dhathri Purohith, Matthew Halpern, Vijay Chidambaram, Vijay Janapa Reddy
Limited battery capacity is a major concern!

However, battery density doubles only once every 10 years.
What consumes battery?

Usual suspects: screen, network

Is storage a major contributor?
Random writes take 20x more energy than sequential writes.

Storage subsystem takes 36% of total energy for random IO intensive workload.

Random reads take 8x more energy than sequential reads.

Measure energy Differentially to segregate storage sub-system energy on a commercial smartphone.
Overview

How do we measure storage energy?

Energy at different layers of storage stack
  - File IO Operations
  - SQLite Operations
  - Android applications

Implications for File System Design

Conclusions
Tools to measure energy

- **Software Based:**
  - Battery sensor: Periodically check current battery level
  - Apps: Requires power models.
  - Very crude measure.
  - Cannot detect small consumptions.

- **Hardware Based:**
  - More fine-grained measure.
  - Requires specialized hardware to get component-wise energy.
Experimental setup

Samsung Galaxy nexus connected to Monsoon Power Monitor
Differential Energy Analysis

- Hardware tools provide fine-grained energy measurements, but not component-wise.
- Design experiments to measure energy “differentially”.
- IO intensive Workload: 100 MB of random writes of IO size 4KB.
Differential energy measurement

IDLE STATE

Screen On, No background Apps, No IO

CPU AND MEMORY

In-memory writes over network

NETWORK

Writes to internal eMMC

STORAGE SUBSYSTEM

In-memory writes over network

Screen On, No background Apps, No IO
Overall Storage Energy Consumption

- Energy consumed by storage subsystem is almost **equal** to the energy consumed by screen for an IO intensive workload.
Overview

How do we measure storage energy?

Energy at different layers of storage stack
  - File IO Operations
  - SQLite Operations
  - Android applications

Implications for File System Design

Conclusions
File IO operations

Sequential IO Workload:
- **IO Size**: 512KB blocks.
- **Total IO**: 1GB of file reads and writes.

Random IO Workload:
- **IO Size**: 4KB blocks.
- **Total IO**: 100MB of file reads and writes.
- Fsync issued after every IO request.
F2FS vs Ext4: File ops

Storage energy in uJ/KB

Seq Write  
Rand Write

Ext4

F2FS

19X

12X

Seq Write  
Rand Write

0  
50  
100  
150  
200  
250  
300  
350  
400  
450  
500

0  
50  
100  
150  
200  
250  
300  
350  
400  
450  
500
F2FS vs Ext4: File ops

Seq Read
- Ext4: 1X
- F2FS: 8X

Rand Read
- Ext4: 7X
- F2FS: 8X
F2FS vs Ext4: Write Amplification

Ext4:
- In-place updates.
- Fsync forces both data and metadata to be written on to the disk.
- Meta data includes:
  - Inode table
  - Journal transaction begin block
  - Journal transaction end block
  - list of blocks in the transaction.

Random Write (10MB) Actual I/O at the block layer (in MB):

- Ext4: 72
F2FS vs Ext4: Write Amplification

**F2FS:**
- Log structured.
- Maintains NAT table for address translation.
- Only data blocks and their direct node blocks are written after every fsync.
- Meta data includes – File inodes, NAT and SIT updates.
F2FS vs Ext4: Read Amplification

Ext4:
- Android uses aggressive read prefetching.
- blktrace reveals minimum size of read request is 8KB.
F2FS vs Ext4: Read Amplification

F2FS:
- Every read constitutes of a request to read direct node block and the data.
- Every read request to direct node block results in NAT translation.
Overview

How do we measure storage energy?

Energy at different layers of storage stack
- File IO Operations
- SQLite Operations
- Android applications

Implications for File System Design

Conclusions
SQLite operations

Workload:

- Prepopulate 1M entries.
- 15K each of SQLite Inserts, Updates and Deletes.
- SQLite record size: 4KB.
- WAL-NORMAL
F2FS vs Ext4: SQLite Operations

- **Inserts**: Ext4 - 1.5X, F2FS - 1.5X
- **Updates**: Ext4 - 1.5X, F2FS - 1.5X
- **Deletes**: Ext4 - 1.5X, F2FS - 1.5X

Storage energy in mJ/Txn
Overview

How do we measure storage energy?

Energy at different layers of storage stack
  - File IO Operations
  - SQLite Operations
  - Android applications

Implications for File System Design

Conclusions
Android applications

- **Applications Studied:** Mail and Facebook
- **Duration traced:** 180 seconds
- **Energy estimation:**
  - Percentage of random and sequential IO is computed using *blktrace*.
  - Sequential IO between two flushes are merged.
  - IO size < 32KB after merge is tagged as random.
  - Application energy consumption is estimated using File IO energy stats.
F2FS vs Ext4: Android applications

Percentage of Random IO at block level

Mail
Write
Read
Facebook
Write
Read

% of randomness
0
10
20
30
40
50
60

Ext4
F2FS
F2FS vs Ext4: Android applications

Total energy consumed by storage for different Android applications

MAIL: Ext4 - 42.91 J, F2FS - 20.07 J (2.1X)
FACEBOOK: Ext4 - 14.13 J, F2FS - 8.79 J (1.6X)
Implications for File System Design

- Use sequential IO
  - F2FS still performs around 20-28% of random writes and about 12-20% of random reads.
  - Sequentializing the last 20-28% of random writes in F2FS can reduce energy consumption by half.
- Account for trade-off between sequential writes and random reads.
- Use compression to reduce IO.
Conclusions

- Differential analysis gives component-wise energy measurements on commercial phones.
- Contribution of storage to energy consumption in Android is significant - 36%!
- Huge energy benefits by sequentializing I/O.
- F2FS can be made significantly more energy-efficient.
Thank you!