Fast Transaction Logging for Smartphones

Hao Luo, University of Nebraska Lincoln

Hong Jiang, University of Texas Arlington

Zhichao Yan, University of Texas Arlington

Yaodong Yang, University of Nebraska Lincoln

Outline

- Introduction
- Logging Overhead in Mobile Databases
- Design of xLog
- Evaluation
- Conclusion

Introduction

- The smartphones and tablets have become ubiquitous.
- Storage subsystem impacts the application performance.
  - Database
  - File system

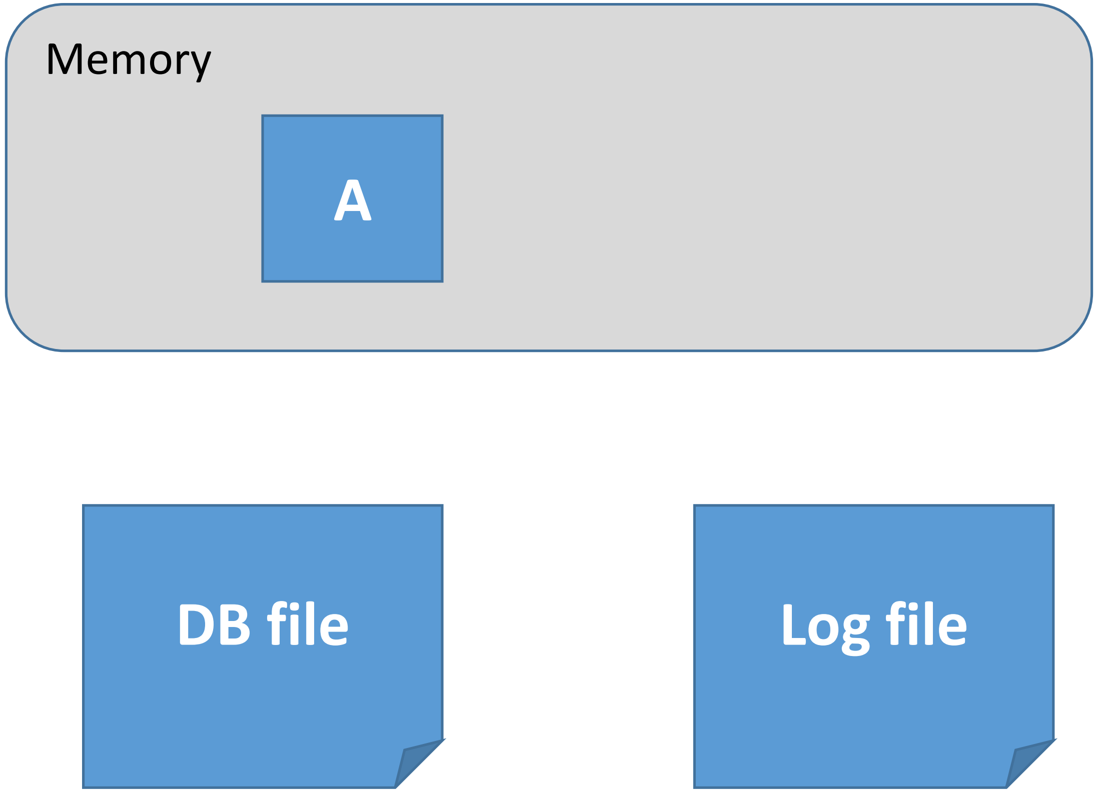
Introduction

- Database engines has become a crucial part of data management in mobile systems.
  - SQLite
  - LevelDB
- Mobile databases employ logging to ensure data persistency
  - Atomicity
  - Consistency
  - Durability

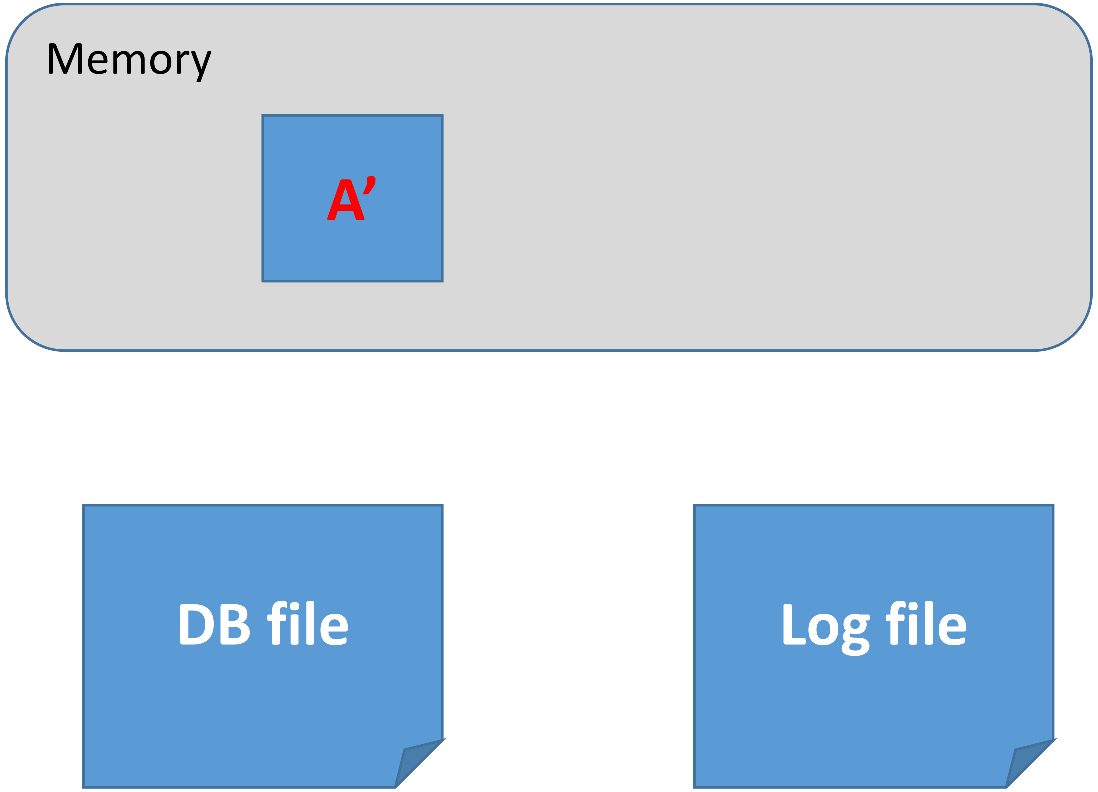
Introduction

- Journal of journaling (JOJ) anomaly drastically slows down the mobile databases
  - Breaks the sequential pattern of log I/O
  - Writes more data.

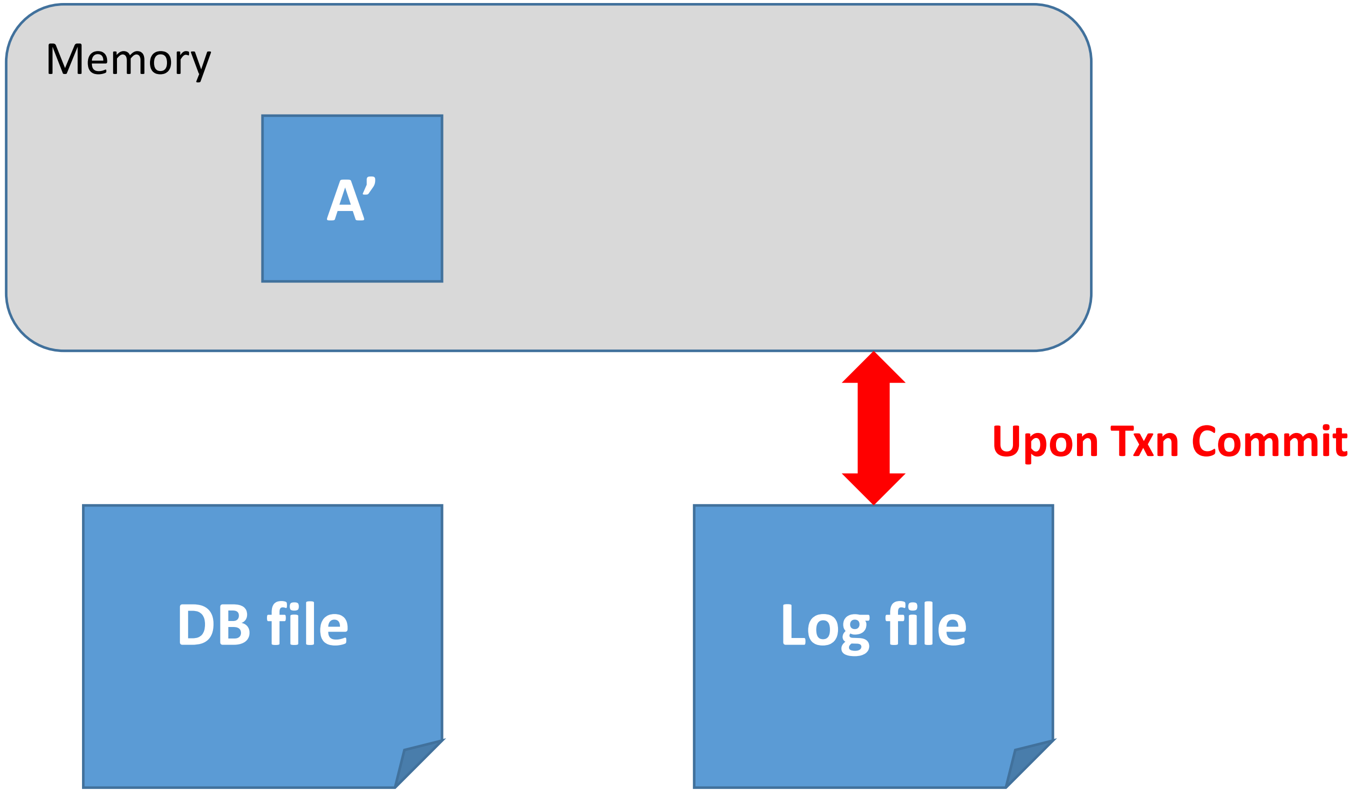
Transaction Commit Path



Transaction Commit Path



Transaction Commit Path



Transaction Logging

- Value Logging
    - Logs after-image of the database pages
    - Used by SQLite
- Command Logging
    - Logs transaction logic (e.g., SQL query)
    - Used by LevelDB

SQLite Write Ahead Log

- Logs modified database pages.
  - Header + modified pages
  - 4KB page size
- fsync() upon transaction commit.

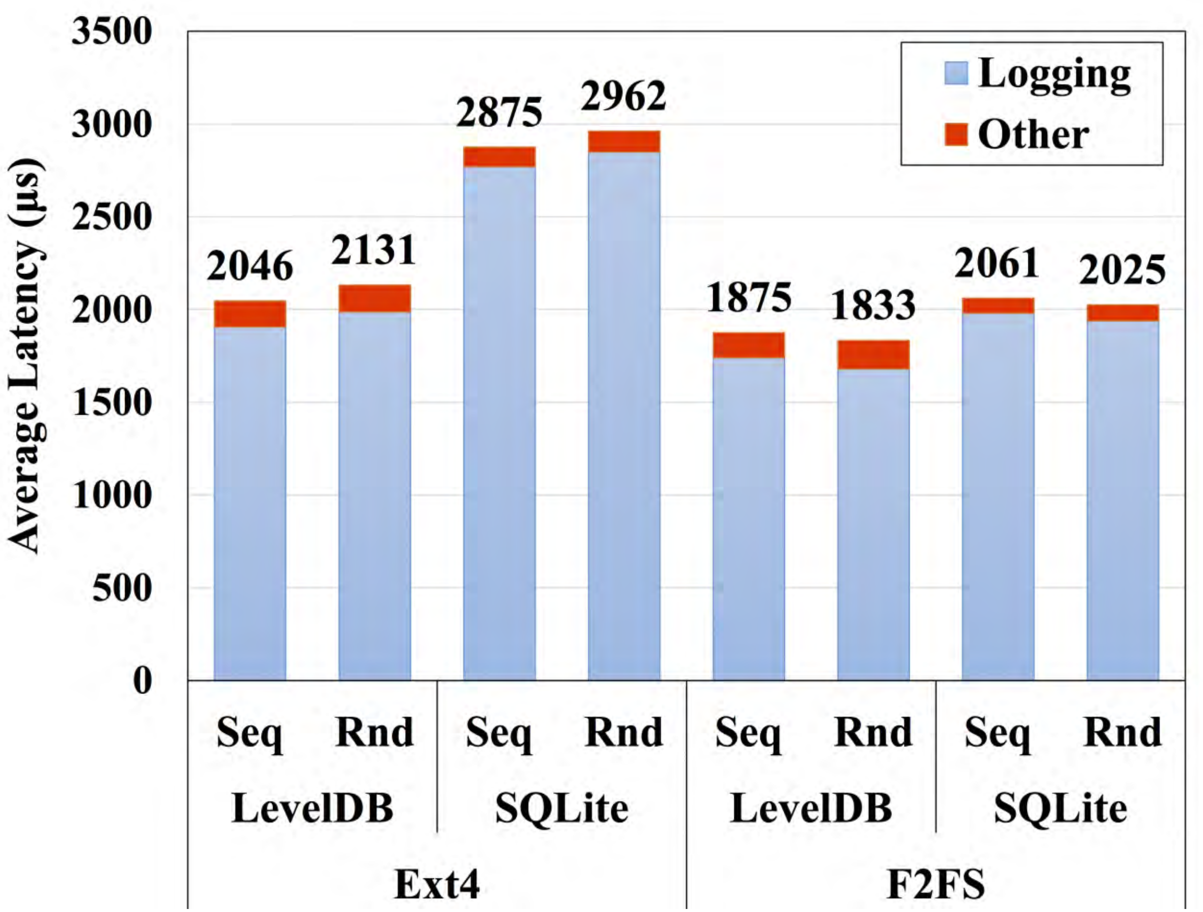
LevelDB Logging

- Logs transaction logic.
  - Put(): kTypeValue + key + value
  - Delete(): kTypeDeletion + key
- fsync() upon commit (in synchronous mode)

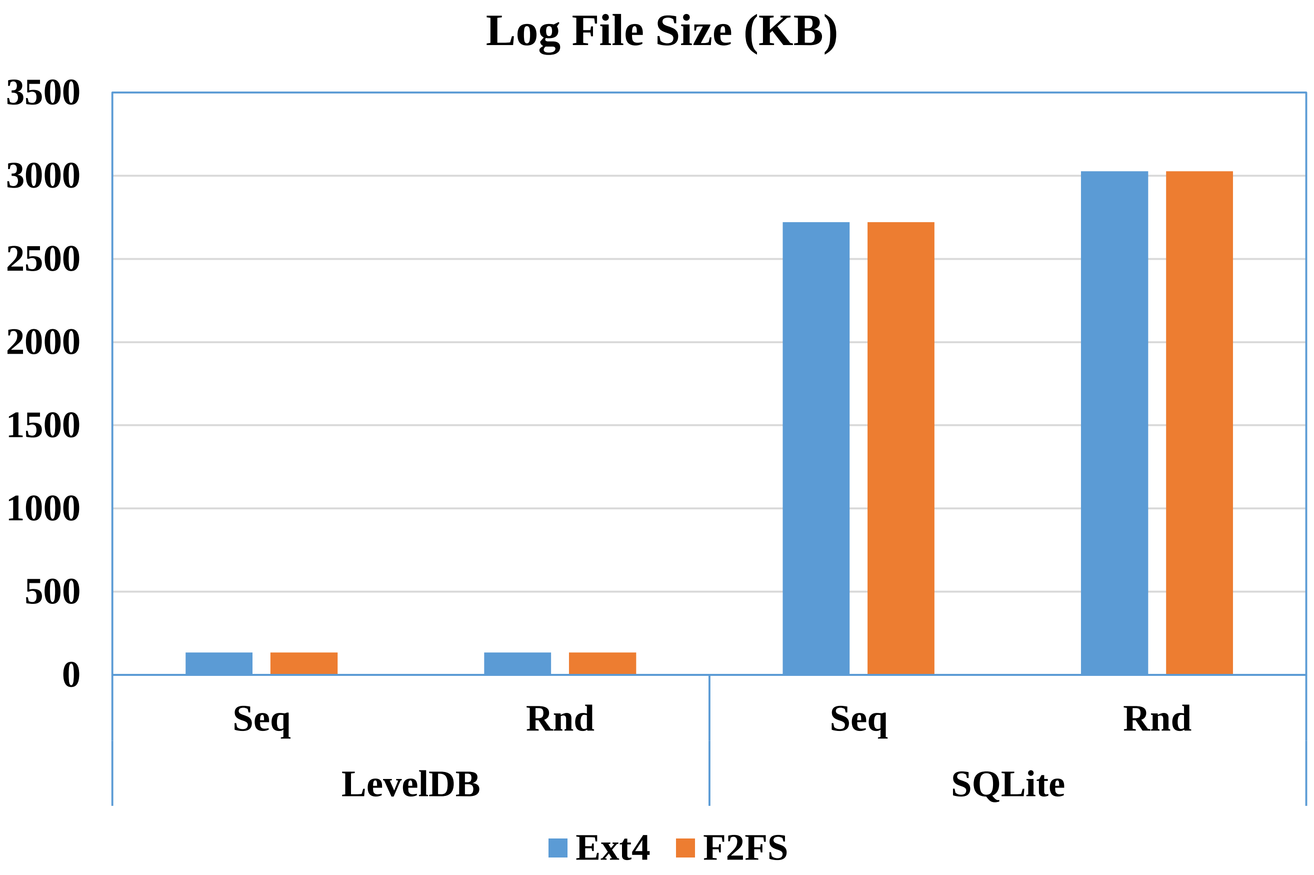
Logging Overhead

- To assess the overhead of logging in mobile devices, we set up a simple benchmark test
  - 4 byte key, 100 byte value
  - 1000 insertion
  - Sequential / Random key order
  - Samsung Galaxsy S4
- Run the benchmark on different configurations
  - DB engine: SQLite, LevelDB
  - File system: Ext4, F2FS

Logging Overhead



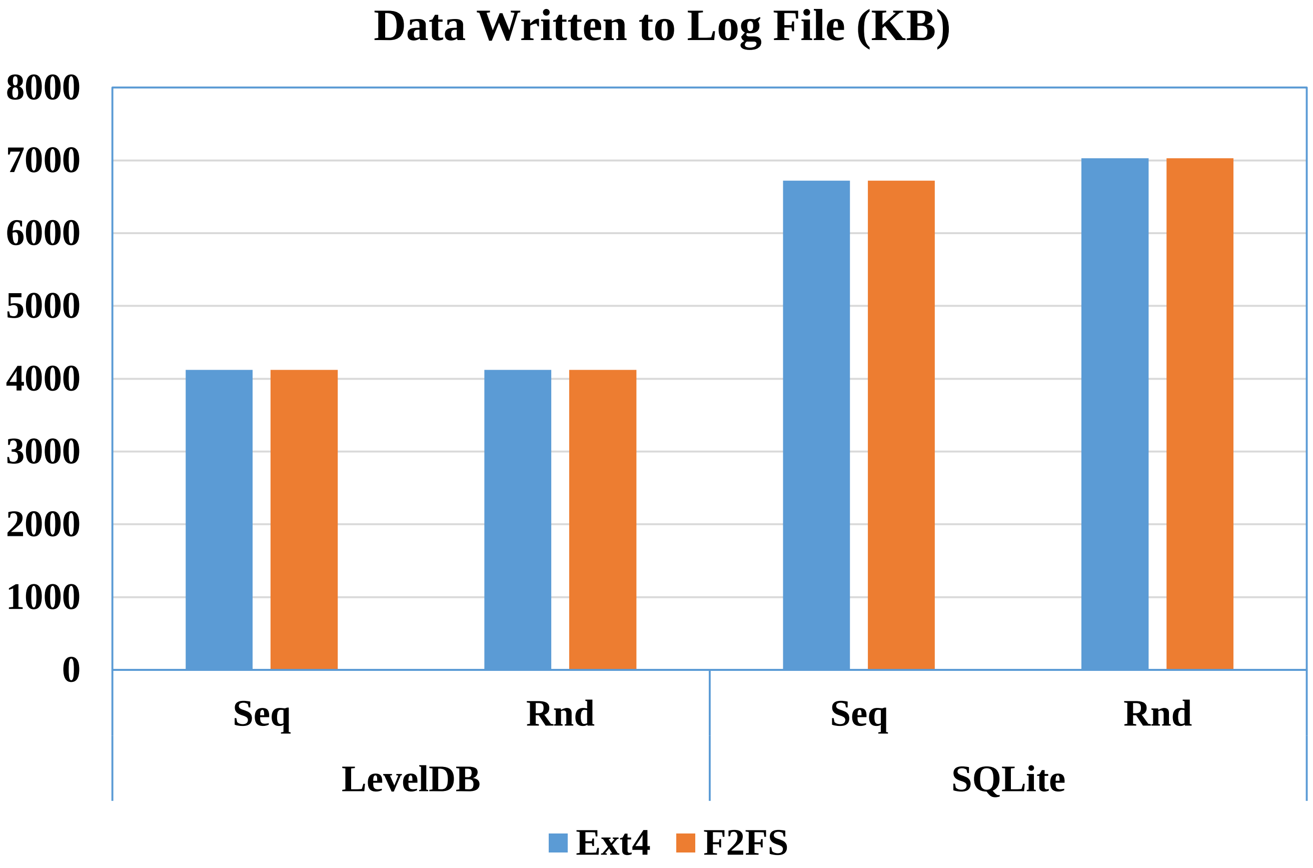
Log File Size



Log File Size

- Log record size
  - SQLite: header + key + value (130 Byte)
  - LevelDB: header + modified pages (several KB)
- SQLite’s value logging writes significantly more data to the log files.

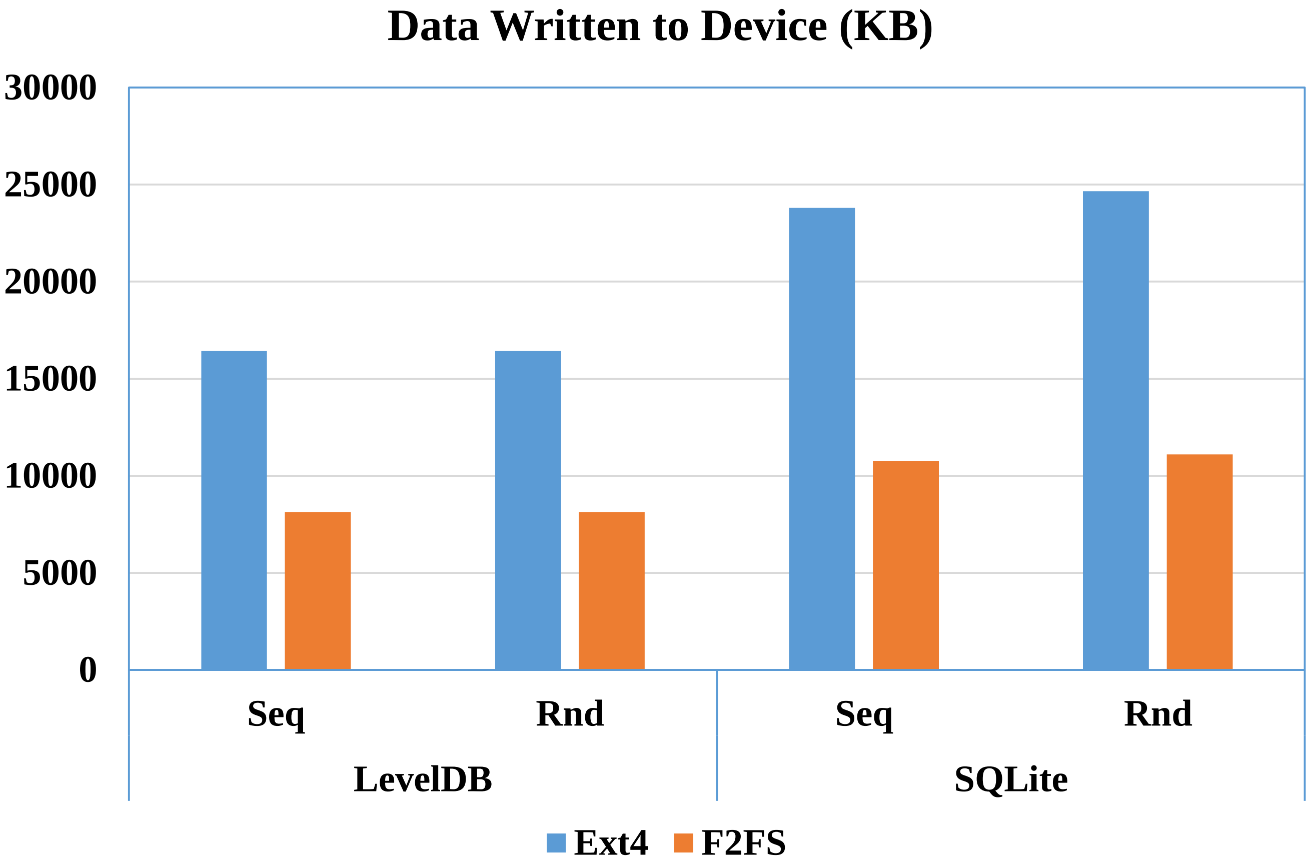
Data Written to Log File



Data Written to Log File

- File system block
  - Data are flushed at file system block boundaries.
  - Usually 4KB in mobile devices.

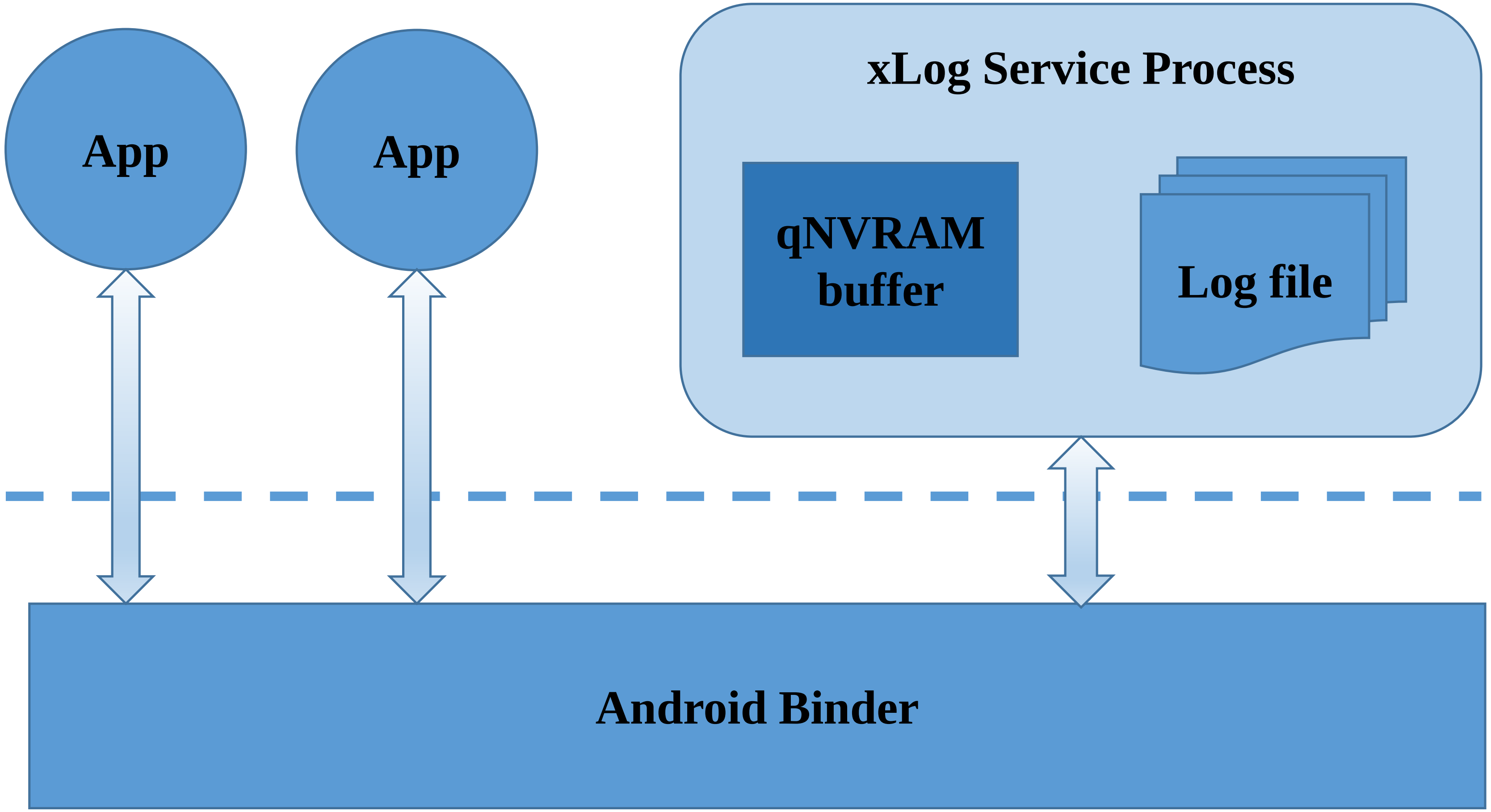
Data Written to Device



qNVRAM

- Nearly non-volatile memory in smartphones
  - Takes advantage of battery-backed nature of mobile devices.
  - Data survive almost all the failure conditions.
    - Application crash
    - Kernel panic
    - Hard reset

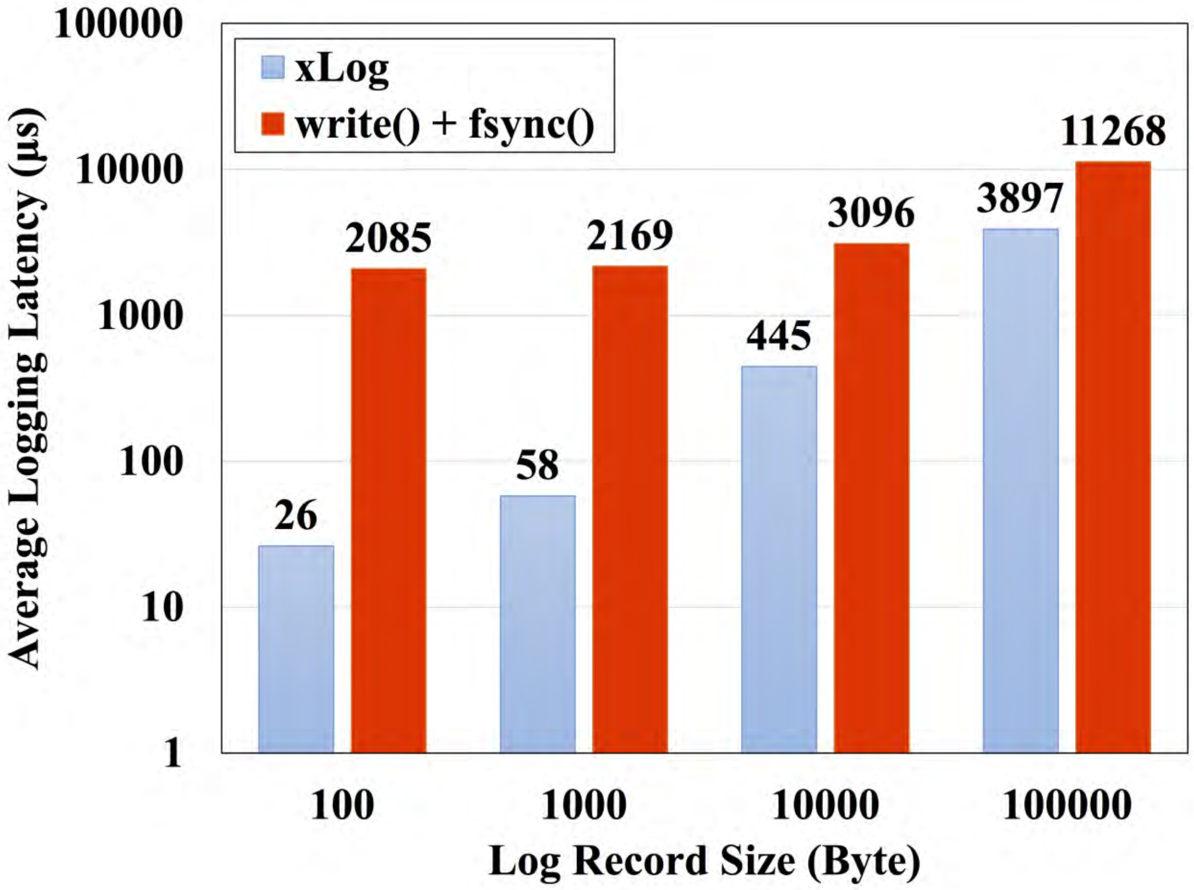
xLog


xLog Service Process
App
App
qNVRAM buffer
Log file
Android Binder

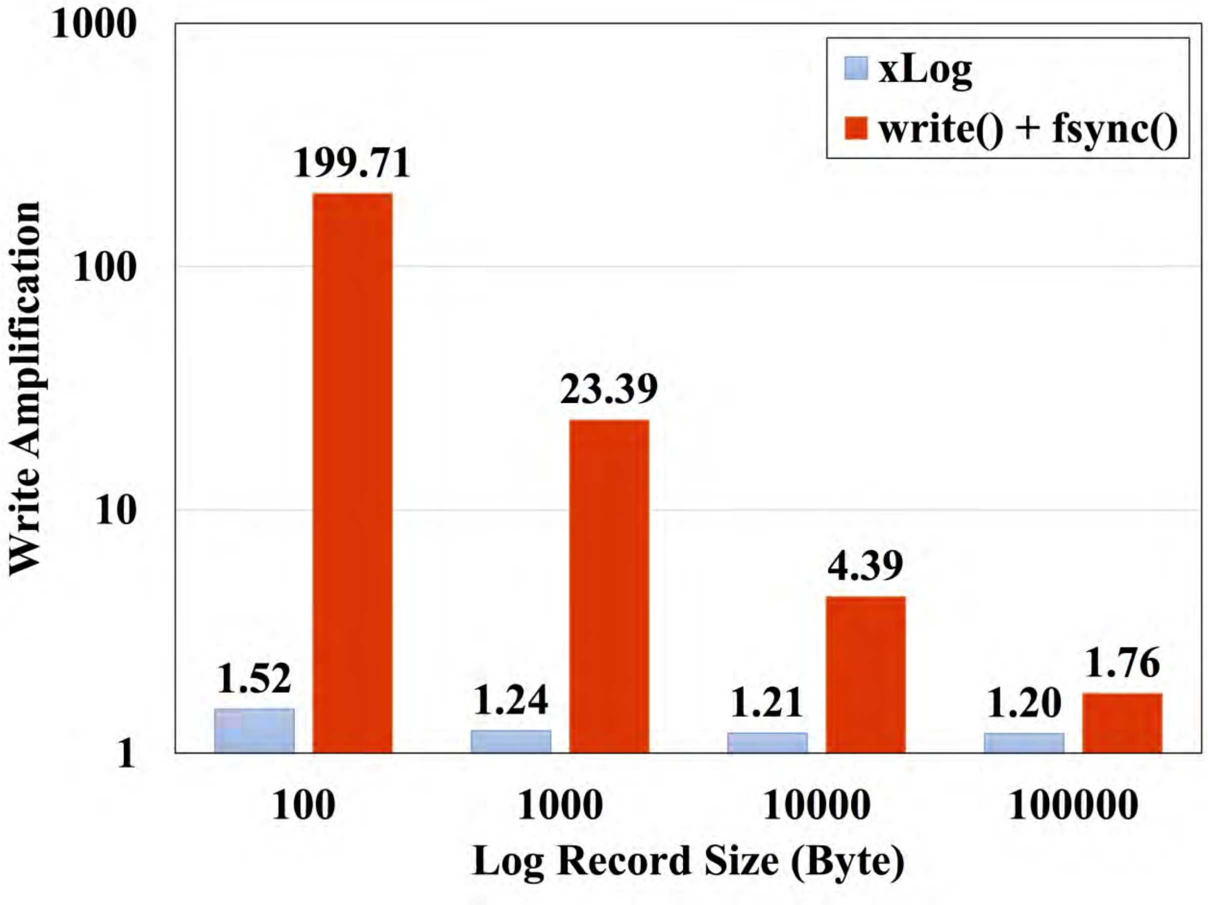
Evaluation

- Microbenchmark
  - Raw performance of the xLog
  - Different log record size

Micro-benchmark Performance



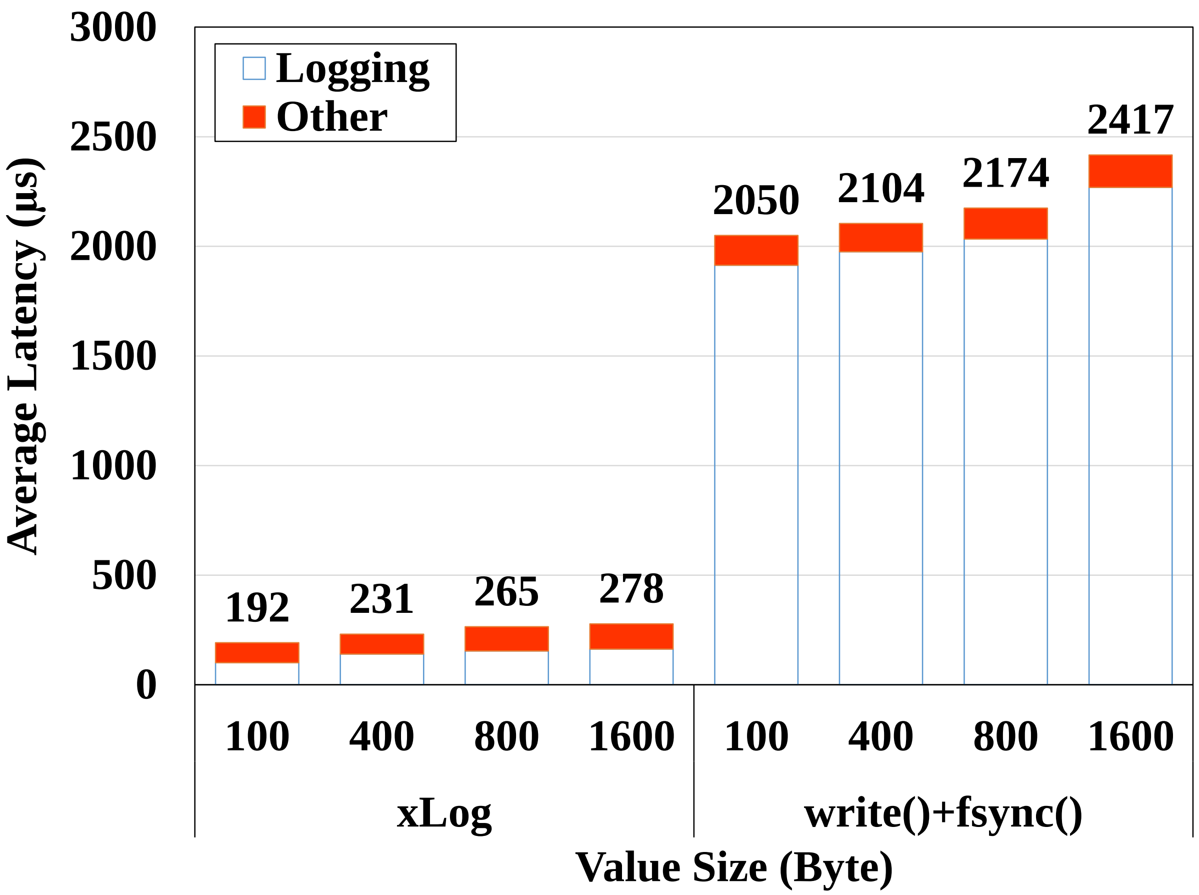
Micro-benchmark Write Amplification



Evaluation

- Micro-benchmark
  - Raw performance of the xLog
  - Different log record size
- Macro-benchmark
  - Transaction latency of LevelDB
  - xLog v.s. write()+fsync()
  - Different size of value

Evaluation



3000
2500
2000
1500
1000
500
0
Average Latency (µs)
□ Logging
🟥 Other
192
231
265
278
2050
2104
2174
2417
100
400
800
1600
100
400
800
1600
xLog
write()+fsync()
Value Size (Byte)

Conclusion

- In this paper we present xLog, a fast transaction logging service that uses qNVRAM as a buffer, for Android smartphones.
- xLog logs up to 77x times faster than the tranditional logging scheme
- xLog drastically reduces the write amplification from 122x to 1.6x.

Thank you!

Q & A