Two Techniques for Faster Transactional Atomicity on Flash and NVram

• SQLite/PPL [VLDB 15]
• CFS [USENIX ATC 15]

Sang-Won Lee
(swlee@skku.edu)
SQLite Optimization
with Phase Change Memory
for Mobile Applications

VLDB 2015
Gihwan Oh, Sang-Won Lee,
Sangchul Kim, and Bongki Moon
Overview

• PCM: promise, reality, and opportunities

• SQLite
  ▪ Standard data manager in mobile era
    • Android and iOS

• Characteristics of SQLite and mobile apps
  ▪ Write amplification
  ▪ Write locality
  ▪ Small delta

• SQLite/PPL
## PCM: Promise and Reality

- **Latency: DRAM vs. NAND vs. PCM**

<table>
<thead>
<tr>
<th></th>
<th>DRAM</th>
<th>NAND Flash [29]</th>
<th>PCM (theoretical) [ISCA 09]</th>
<th>PCM (measured) [5,25]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Read</strong></td>
<td>~ 30 ns ( (4B) )</td>
<td>156 us ( (4KB) )</td>
<td>48 ns ( (4B) )</td>
<td>408 ns ( (4B) )</td>
</tr>
<tr>
<td><strong>Write</strong></td>
<td>~ 30 ns ( (4B) )</td>
<td>505 us ( (4KB) )</td>
<td>150 ns ( (4B) )</td>
<td>7.5 us ( (4B) )</td>
</tr>
</tbody>
</table>

- Similar observation [FAST ‘14]

**Slower**

**8.5x, 50x Slower**
**NAND vs. PCM**

- **Write Latency: PCM vs. NAND**
  - 4B Write: 7.5us vs. 505us
  - 4KB Write: 7500us vs. 505us

![Graph comparing NAND and PCM write latencies](image)
Unified Memory System

- UMS Architecture
- UMS Board [RSP ‘14]

- 10,000$ ??
Write Amplification in Mobile Application

Huge Write Amplification: ‘Hi’ → 11*4KB page writes

Auto-commit
Force-write
Block interface
Journaling
File system Metadata

Performance; Endurance

Mobile Application

SQL Interface

B-tree module

SQLite (Library)

File System

Buffer Cache

DRAM

Block Interface

SQLite (Library)

Page
Page
Page

Database File (per Application)

Journal File

Nand Flash Storage

Auto-commit
Force-write
Block interface
Journaling
File system Metadata

Huge Write Amplification: ‘Hi’ → 11*4KB page writes [16,20]
Write Locality in Mobile Apps

‘Hi’

‘What’s up?’

Consecutive Single Simple Transactions with Small Bytes

Same pages are updated

Sequence of page-write request in SQLite
Small Delta Between Consecutive Writes

- Mostly less than several 100s bytes
Implications

• The small deltas of SQLite pages
  ▪ Capturing and storing the small deltas will avoid write amplification by SQLite
  ▪ Avoiding write amplification will provide faster response time and longer lifespan of NAND flash

• Byte-addressable, Non-volatile PCM supporting short write latency of small data
  ▪ PCM as a log area of small delta
PPL Architecture

- PPL module is added
PPL Architecture

- PPL module is added
Evaluation Setup

• Compare SQLite/PPL with Rollback journal (RBJ) and WAL journal

• 6 mobile workloads
  ▪ Real workloads: Kakaotalk, Twitter, Facebook, Gmail, Web Browser
  ▪ Synthetic workload: AndroBench

• A Zync-7030 board equipped with the real PCM chip [RSP ‘14]
Baseline Performance Comparison

• Overall Execution Time: SQLite RBJ vs. WAL vs. PPL

• See paper for performance details of Latency, Effect of Log Sector Size/All in PCM, Read Performance
Conclusion

• Present the design and implementation of SQLite/PPL

• Future works
  ▪ Apply PPL to enterprise DB: e.g. Postgres [CACM 91]
  ▪ Xxxxxxxx logging
Q & A
Lightweight Application-Level Crash Consistency on Transactional Flash Storage

Usenix ATC 2015
Changwoo Min, Woon-Hak Kang, Taesoo Kim,
Sang-Won Lee, Young Ik Eom
Two Update Approaches

- **In-place update** vs. **copy-on-write**
  - Durability and atomicity of tx app.

![Diagram showing RAM, IPU, and CoW approaches with old and new data versions](image-url)
In-place Update vs. CoW

• Why IPU > CoW in computer science?
  ▪ Storage cost
  ▪ Clusteredness of pages in a file (for HDD?)

• But, historically, CoW > IPU! (Jim Gray)
  ▪ Multi-version support
  ▪ Clusteredness in flash is less important
X-FTL

- Flash-aware transactional atomicity for application taking IPU (e.g. SQLite)

- Cf. FusionIO’s atomic write (vs. 서울여대)
X-FTL and File Metadata

• X-FTL can support transactional atomicity of updated pages in user files.

• What about the shared metadata pages updated by concurrent transactional applications?
  ▪ Feedback from Prof. Jin-Soo Kim
  ▪ False sharing of metadata
More about CFS

• **System-wise vs. transaction-wise** consistency
  ▪ Redo and undo logging for meta-data update

• cf. Application-level crash consistency  @ Remzi group
  ▪ Vijay@Wisconsin [OSDI12, SOSP13, FAST13, PhD thesis]
  ▪ No multiple file support: cf. CFS

• cf. Failure-Atomic Update of Application Data (Usenix FAST 2015)
More about CFS (2)

• In flash era, “CFS + X-FTL” is an answer to
  ▪ Application-level crash consistency
  ▪ Journaling of journal

• More lightweight solution than “CFS + X-FTL”?  
  ▪ Problem in “CFS + X-FTL”: explicit tx concept (e.g. tid)
  ▪ E.g. SHARE interface
  ▪ SQLite journaling overhead: Xxxxxxxx logging
What if address remapping feature is exposed to applications?
Q & A