qNVRAM: quasi Non-Volatile RAM for Low Overhead Persistency Enforcement in Smartphones

Hao Luo, Lei Tian, Hong Jiang
University of Nebraska, Lincoln
Living in a mobile era

• We are in the mobile era
• The I/O performance dominates the overall application performance.
  • persistency = atomicity + consistency + durability
• To ensure data persistency logging is employed by different components.
Outline

• Introduction
• **Overhead of Persistency**
• Rethinking Memory Volatility in Smartphones
• Design of qNVRAM
• Case Study
Overhead of Persistency

• Logging

Memory

Flash

Real Data Log Area
Overhead of Persistency

• Logging

Memory

Flash

Real Data

Log Area

A

A’
Overhead of Persistency

- Logging
Overhead of Persistency

• Logging
Overhead of Persistency

• Double Write
  • Some applications store data in raw files
  • Applications *overwrite the whole file* when update it.

App

write(“foo.tmp”)
Overhead of Persistency

• Double Write
  • Some applications store data in raw files
  • Applications *overwrite the whole file* when update it.

```python
rename("foo.tmp", "foo")
```
Overhead of Persistency

- Google Maps: 34% Real Data, 5% FS Journal, 32% DB Journal, 8% Double Write
- Twitter: 22% Real Data, 44% FS Journal, 34% DB Journal, 0% Double Write
- Facebook: 38% Real Data, 32% FS Journal, 1% DB Journal, 1% Double Write
- Chrome: 33% Real Data, 42% FS Journal, 7% DB Journal, 18% Double Write
- Youtube: 41% Real Data, 5% FS Journal, 52% DB Journal
- Gmail: 63% Real Data, 27% FS Journal, 7% DB Journal
- Angry Birds: 36% Real Data, 7% FS Journal, 52% DB Journal, 5% Double Write
• Why do we need logging?
• Memory is volatile.
• Why do we need logging?
  • Memory is volatile.
  • Every smartphone has battery
• Why do we need logging?
• Memory is volatile.
• Every smartphone has battery
• DRAM + Battery = ?
Outline

• Introduction
• Overhead of Persistency
• **Rethinking Memory Volatility in Smartphones**
• Design of qNVRAM
• Case Study
Rethinking Memory Volatility in smartphones

• Four types of failures in smartphones
  • Application crash
  • Application hang
  • Self-reboot
  • System freeze
Rethinking Memory Volatility in smartphones

- Application crash / Application hang
Rethinking Memory Volatility in smartphones

• Application crash / Application hang
Rethinking Memory Volatility in smartphones

- Application crash / Application hang
Rethinking Memory Volatility in smartphones

- Application crash / Application hang

What if the application can retrieve data from OS?
Rethinking Memory Volatility in smartphones

• Self-reboot
Rethinking Memory Volatility in smartphones

• Self-reboot
Rethinking Memory Volatility in smartphones

• Self-reboot

Application Memory Space

Memory Management

Physical Memory

Page x
Rethinking Memory Volatility in smartphones

• Self-reboot

DRAM doesn’t lose power over reboot!!! But the OS doesn’t know.
Rethinking Memory Volatility in smartphones

• Self-reboot

What if the data is stored at a fixed physical location?
Rethinking Memory Volatility in smartphones

• System freeze
Rethinking Memory Volatility in smartphones

- System freeze
  - user perform hard reset
Rethinking Memory Volatility in smartphones

• System freeze

What if the OS flushes the important data to flash when user perform hard reset?
Rethinking Memory Volatility in smartphones

- Failure mode and DRAM data

<table>
<thead>
<tr>
<th></th>
<th>Application space</th>
<th>OS</th>
<th>Physical memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Crash</td>
<td>✕</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Application Hang</td>
<td>✕</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Self-reboot</td>
<td>✕</td>
<td>✕</td>
<td>✓</td>
</tr>
<tr>
<td>System Hang</td>
<td>✕</td>
<td>✕</td>
<td>✕</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Overhead of Persistency
• Rethinking Memory Volatility in Smartphones
• Design of qNVRAM
• Case Study
Design of qNVRAM

- qNVRAM: quasi Non-Volatile RAM
  - user library
  - kernel module
Design of qNVRAM

- qNVRAM library provides easy-to-use APIs
  - alloc()
  - free()
  - retrieve()
Design of qNVRAM

• Kernel module manages a small chunk of physical memory.
  • reserved during boot up
  • fixed physical address
  • emergency flush
Outline

• Introduction
• Overhead of Persistency
• Rethinking Memory Volatility in smartphones
• Design of qNVRAM
• Case Study
Case Study

- Persistent Page Cache in SQLite
- In-place update
Case Study

• Persistent Page Cache in SQLite 
  • In place update
Case Study

• Persistent Page Cache in SQLite
  • In place update
Case Study

- Persistent Page Cache in SQLite
- In place update
- Recover from persistent page cache upon failure.
Case Study

- Persistent Page Cache in SQLite
  - In place update
  - Recover from persistent page cache upon failure.
Case Study

- Lazy Flush
  - Dirty pages are flushed asynchronously.
Case Study

- Performance evaluation
  - Samsung Galaxy S4 Google Edition
    - 4 core CPU
    - 2GB RAM
    - 16GB flash
  - SQLite 3.7.12
    - Persistent page cache
    - Lazy Flush
Case Study

- Performance evaluation

![Bar chart showing performance evaluation of different cache configurations.]

- Baseline (WAL)
- pPCache
- +LazyFlush(5)
- +LazyFlush(∞)

Transactions Per Second

~16x
Case Study

- Performance evaluation

![Graph showing WAL and LazyFlush](image)

WAL

LazyFlush (5)
Conclusion

• qNVRAM takes advantage of the battery-backed characteristic of smartphones.
• qNVRAM can speed up SQLite by up to 16x.