Revisiting Storage for Smartphones

Hyojun Kim  Nitin Agrawal  Cristian Ungureanu

NEC Laboratories America, Inc.
Relentless passion for innovation
Life in the “Post-PC” Mobile Era

- Smartphone and tablet markets are huge & growing
  - 100 Million smartphones shipped in Q4 2010, 92 M PCs [IDC]
  - Out of 750 Million Facebook users, 250 Million (& growing) access through mobile; mobile users twice as active [FB]

- Innovation in mobile hardware: packing *everything* you need in your pocket
  - Blurring the phone/tablet divide: Samsung Galaxy Note
  - Hardware add-ons: NEC Medias (6.7mm thick, waterproof shell, TV tuner, NFC, HD camera, ..)

- Manufacturers making it easier to replace PCs
  - Motorola Atrix dock converts a phone into laptop
Waiting is undesirable!
Annoying for the user
More so for interactive mobile users
More time, more battery

Easy to lose customers

Aren’t network and CPU the real problem?
Why are we talking about storage?
Understanding Mobile Performance

Well understood!

- Network performance can impact user experience
  - 3G often considered the bottleneck for apps like browsing
  - Service providers heavily investing in 4G and beyond

- CPU and graphics performance crucial as well
  - Plenty of gaming, video, flash-player apps hungry for compute
  - Quad-core CPUs, GPUs to appear on mobile devices

Not well understood!

- Does storage performance impact mobile experience?
  - For storage, vendors & consumers mostly refer to capacity
Flash storage on mobile performs better than wireless networks.

Most apps are interactive; as long as performance exceeds that of the network, difficult for storage to be bottleneck.
Outline

✓ Introduction

Why storage is a problem

Android storage background and setup

Experimental results

Solutions
Why Storage is a Problem
Random versus Sequential Disparity

- Performance for random I/O significantly worse than seq; inherent with flash storage
- Mobile flash storage classified into *speed classes* based on sequential throughput
- Random write performance is orders of magnitude worse

<table>
<thead>
<tr>
<th>Vendor (16GB)</th>
<th>Speed Class</th>
<th>Cost US $</th>
<th>Seq Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transcend</td>
<td>2</td>
<td>26</td>
<td>4.2</td>
</tr>
<tr>
<td>RiData</td>
<td>2</td>
<td>27</td>
<td>7.9</td>
</tr>
<tr>
<td>Sandisk</td>
<td>4</td>
<td>23</td>
<td>5.5</td>
</tr>
<tr>
<td>Kingston</td>
<td>4</td>
<td>25</td>
<td>4.9</td>
</tr>
<tr>
<td>Wintec</td>
<td>6</td>
<td>25</td>
<td>15.0</td>
</tr>
<tr>
<td>A-Data</td>
<td>6</td>
<td>30</td>
<td>10.8</td>
</tr>
<tr>
<td>Patriot</td>
<td>10</td>
<td>29</td>
<td>10.5</td>
</tr>
<tr>
<td>PNY</td>
<td>10</td>
<td>29</td>
<td>15.3</td>
</tr>
</tbody>
</table>

Consumer-grade SD performance

However, we find that for several popular apps, substantial fraction of I/O is random writes (including web browsing!)
Storage coming under increasingly more scrutiny in mobile usage

- Random I/O performance has not kept pace with network improvements
- 802.11n (600 Mbps peak) and 802.11ad (7 Gbps peak) offer potential for significantly faster network connectivity to mobile devices in the future
Deconstructing Mobile App Performance

- **Focus:** understanding contribution of storage
  - How does storage subsystem impact performance of popular and common applications on mobile devices?
  - Performed analysis on Android for several popular apps

- **Several interesting observations through measurements**
  - Storage adversely affects performance of even interactive apps, including ones not thought of as storage I/O intensive
  - SD Speed Class not necessarily indicative of app performance
  - Higher total CPU consumption for same activity when using slower storage; points to potential problems with OS or apps

- **Improving storage stack to improve mobile experience**
Outline

✓ Introduction

✓ Why storage is a problem

Android storage background and setup

Experimental results

Solutions
# Storage Partitions on Android

<table>
<thead>
<tr>
<th>Partition</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Misc</td>
<td>H/W settings, persistent shared space between OS &amp; bootloader</td>
</tr>
<tr>
<td>Recovery</td>
<td>Alternative boot-into-recovery partition for advanced recovery</td>
</tr>
<tr>
<td>Boot</td>
<td>Enables the phone to boot, includes the bootloader and kernel</td>
</tr>
<tr>
<td>System</td>
<td>Contains the remaining OS, pre-installed system apps ; read-only</td>
</tr>
<tr>
<td>Cache</td>
<td>Used to stage and apply “over the air” updates; holds system images</td>
</tr>
<tr>
<td>Data</td>
<td>Stores user data (e.g., contacts, messages, settings) and installed apps; SQLite DB containing app data also stored here. Wiped on factory reset</td>
</tr>
<tr>
<td>Sdcard</td>
<td>External SD card partition to store media, documents, backup files etc</td>
</tr>
<tr>
<td>Sd-ext</td>
<td>Non-standard partition on SD card that can act as data partition</td>
</tr>
</tbody>
</table>
Phone and Generic Experimental Setup

- Rooted and set up a Google Nexus One phone for development
  - GSM phone with a 1 GHz Qualcomm QSD8250 Snapdragon processor
  - 512 MB RAM, and 512 MB internal flash storage
- Setup dedicated wireless access point
  - 802.11 b/g on a laptop for WiFi experiments
- Installed AOSP (Android Open Source Project)
  - Linux kernel 2.6.35.7 modified to provide resource usage information
Custom Experimental Setup
Requirements beyond stock Android

- Ability to compare app performance on different storage devices
  - Several apps heavily use the internal non-removable storage
  - To observe and measure all I/O activity, we modified Android’s *init* process to mount all internal partitions on SD card
  - Measurement study over the internal flash memory and 8 external SD cards, chosen 2 each from the different SD speed classes

- Observe effects of shifting bottlenecks w/ faster wireless networks
  - But, faster wireless networks not available on the phones of today
  - Reverse Tethering to emulate faster networks: lets the smartphone access the host computer’s internet connection through a wired link (miniUSB cable)

- Instrumentation to measure CPU, storage, memory, n/w utilization

- Setup not typical but allows running *what-if* scenarios with storage devices and networks of different performance characteristics
Apps and Experiments Performed

**WebBench Browser**
- Visits 50 websites
- Based on WebKit
- Using HTTP proxy server

**App Install**
- Top 10 apps on Market

**App Launch**
- Games, Weather, YouTube
- GasBuddy, Gmail, Twitter,
- Books, Gallery, IMDB

**RLBench SQLite**
- Synthetic SQL benchmark

**Facebook**

**Android Email**

**Google Maps**

**Pulse News Reader**

**Background**
- **Apps:** Twitter, Books, Gmail
- **Contacts, Picasa, Calendar**
- **Widgets:** Pulse, YouTube,
  News, Weather, Calendar,
  Facebook, Market, Twitter
Outline

✓ Introduction

✓ Why storage is a problem

✓ Android storage background and setup

Experimental results (talk focuses on runtime of apps)

Paper has results on I/O activity, CPU, App Launch behavior, etc

Solutions
Runtime on WiFi varies by 2000% between internal and Kingston
• Even with repeated experiments, with new cards across speed classes
Even without considering Kingston, significant performance variation (~200%)
Storage significantly affects app performance and consequently user experience
With a faster network (USB in RT), variance was 222% (without Kingston)
**With 10X increase in N/W speed, hardly any difference in runtime**
We find a similar trend for several popular apps:

- Storage device performance is important, better card → faster apps

Apart from the benefits provided by selecting a good flash device, are there additional opportunities for improvement in storage?
WebBench: Sequential versus Random I/O

- Few reads, mostly at the start; significantly more writes
- About 2X more sequential writes than random writes
- Since rand is worse than seq by >> 2X, random dominates
- Apps write enough randomly to cause severe performance drop

Paper has a table on I/O activity for other apps
How Apps Use Storage?

- Exactly what makes web browsing slow on Android?
  - Key lies in understanding how apps use SQLite and FS interface

  ![WebBench Storage Schema Diagram]

  - Apps typically store some data in FS (e.g., cache files) and some in a SQLite database (e.g., cache map)
    - All data through SQLite is written synchronously → slow!
    - Apps often use SQLite oblivious to performance effects
What-If Analysis for Solutions

What is the potential for improvements?
– E.g., if all data could be kept in RAM?
– Analysis to answer hypothetical questions

WebBench on RiData

- Placing Cache on Ramdisk does not improve perf. much
- DB on Ramdisk alone improves perf. significantly
- Both Cache and DB in RAM → no extra benefit
- Both Cache and DB on SD without sync recoups most perf

Options:
A. Web Cache in RAM
B. DB (SQLite) in RAM
C. All in RAM
D. All on SD w/ no-sync
Implications of Experimental Analysis

- Storage stack affects mobile application performance
  - Depends on random v/s sequential I/O performance
- Key bottleneck is "wimpy" storage on mobile devices
  - Performance can be much worse than laptops, desktops
  - Storage on mobile being used for desktop-like workloads
- Android exacerbates poor storage performance through synchronous SQLite interface
  - Apps use SQLite for functionality, not always needing reliability
  - SQLite write traffic is quite random → further slowdown!
- Apps use Android interfaces oblivious to performance
  - Browser writes *cache map* to SQLite; slows cache writes a lot
Outline

- Introduction
- Why storage is a problem
- Android storage background and setup
- Experimental results

Solutions
Pilot Solutions

- **RAID-0 over SD card and internal flash**
  - Leverage I/O parallelism already existent
  - Simple software RAID driver with striped I/O
  - As expected speedup, along with super linear speedup due to flash idiosyncrasies (in paper)

- **Back to log-structured file systems**
  - Using NilFS2 to store SQLite databases
  - Moderate benefit; suboptimal implementation

- **Application-specific selective sync**
  - Turn off sync for files that are deemed async per our analysis (e.g., WebCache Map DB)
  - Benefits depend on app semantics & structure

- **PCM write buffer for flash cards**
  - Store performance sensitive I/O (SQLite DB)
  - Small amount of PCM goes a long way
Conclusion

- Contrary to conventional wisdom, storage does affect mobile application performance
  - Effects are pronounced for a variety of interactive apps!
- Pilot solutions hint at performance improvements
  - Small degree of application awareness leads to efficient solutions
  - Pave the way for robust, deployable solutions in the future
- Storage subsystem on mobile devices needs a fresh look
  - We have taken the first steps, plenty of exciting research ahead!
  - E.g., poor storage can consume excessive CPU; potential to improve energy consumption through better storage
Storage Systems Group

NEC Laboratories America, Inc.
Relentless passion for innovation