Storage Systems for Mobile Devices

746 Guest Lecture

Nitin Agrawal

NEC

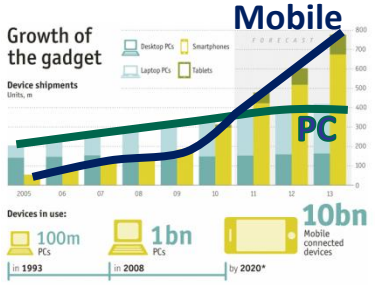
NEC Laboratories America, Inc.

Relentless passion for innovation

NEC Laboratories America
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


amazon
MP3
MADDEN NFL 12
EA SPORTS
Flixster
You Tube
OpenTable
PANDORA
SwiftKey X
htc
10 08
Ti Backup PRO
NETFLIX
GasBuddy.com
IMDb

Simple Pop Quiz

- Who all here have a mobile device?
  - How well do you know your phone/tablet?
- What about its networking capabilities?
  - 3G, 4G, LTE, 802.11n, Bluetooth, …
  - What are their typical uplink/downlink speeds?
- What about CPU?
  - 1 Ghz, dual-core, …
  - Do we really need quad-core CPUs on our phones?
- What about storage?
  - 16 – 64 GB internal flash on iPhone 4S, 32 GB on Nexus S
  - How much of external storage?
  - What kind of file systems do they use?

Understanding Mobile Devices

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

Being understood!

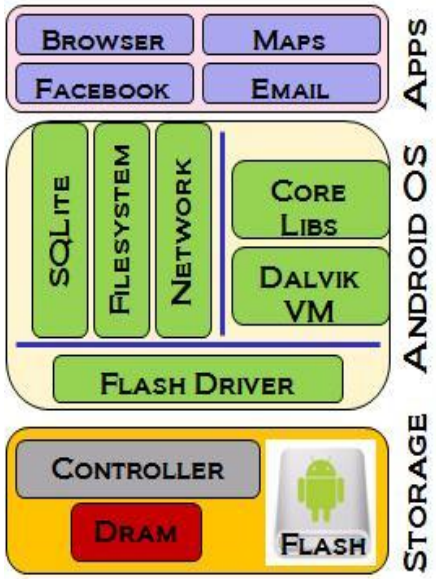
- Energy-efficient orchestration of components
  - Battery life is one of the most important resources in mobile devices
  - How to best manage cellular radio? How to offload computation?

Not well understood!

- Does storage functionality impact mobile experience?
  - For storage, vendors & consumers mostly refer to capacity
  - In this class we will learn about the role of storage in mobile devices

Quick Primer on Android

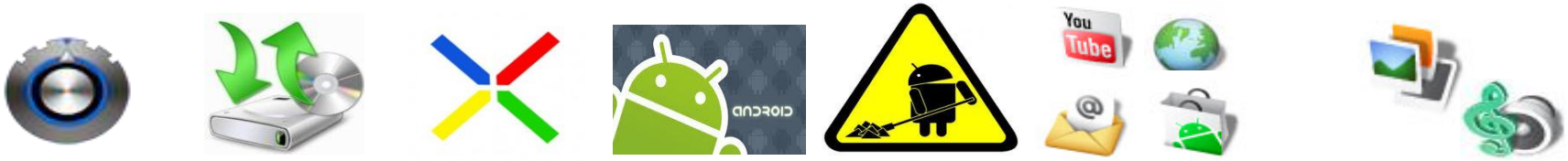
Android OS Architecture



- Android kernel based on Linux
  - Contains low-level drivers for network, storage, power mgmt
- Java middleware: Dalvik virtual machine for application isolation and memory management
  - Each app runs as its own process, with own Dalvik instance
- Libraries to support common needs of applications
  - Libc, Webkit, SSL
- Application framework for development of new apps
  - Abstractions for using system services and hardware
- True multitasking, several apps run as background processes or services

Storage Partitions on Android

NEC Laboratories America
Relentless passion for innovation



/misc	/recovery	/boot	/system	/cache	/data	/sdcard
896KB settings	rootfs 4MB alternate boot	rootfs 3.5MB kernel	yaffs2 145MB read-only	yaffs2 95MB read write	yaffs2 196.3MB read write	FAT32 16GB read write

Internal NAND Flash Memory (512MB) **External SD**

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

Things to Note…

- Storage partitioning similar to Linux
  - Bunch of system and user partitions
- Application's usage of the storage is sandboxed
  - Apps mostly use the */data* partition
- Some mobile-specific partitions
  - "Over the air" updates
  - External, removable media (different usage say from a USB key)

How do Mobile Apps use Storage?

How is Storage Typically Used?

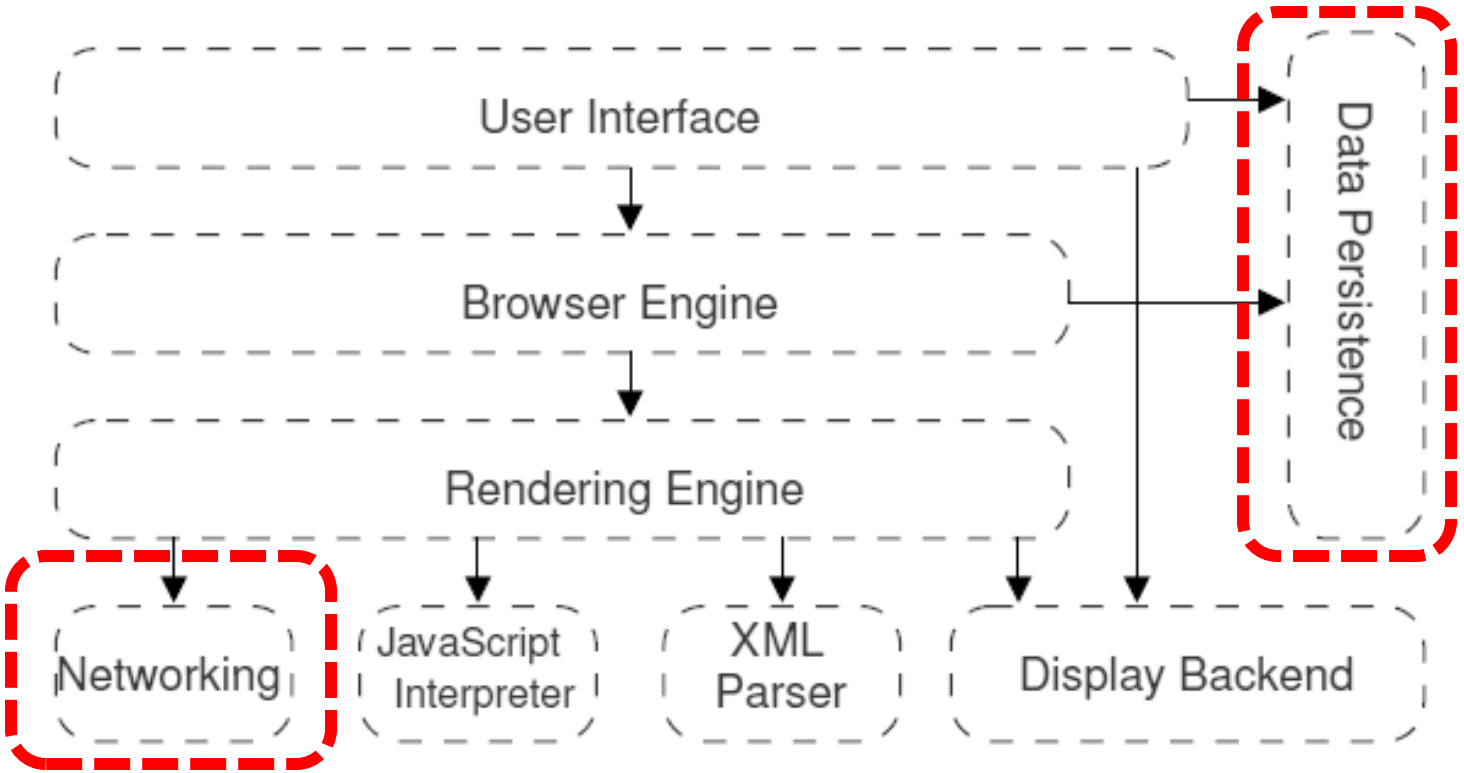
- App binaries/OS software stored on the internal flash
- Apps storing locally-generated user data
  - Photos, movies, voice recordings
- Apps staging user data before network transfer
  - Memos, calendar entries, email drafts,...
- App-private data
  - Periodic game-save data, internal app state for checkpoints
  - App Metadata (not system metadata)
- App caches to reduce network traffic
  - Web browser cache, Google Map tiles, Facebook cache
- Mobile devices as sensors
  - Periodic collection of sensor data through built-in sensors
- Supporting apps to operate in disconnected mode
  - Book reader, subscription services, …
- Provide better user experience
  - Media streaming buffer

Diverse Usage

Different apps use storage in many different ways
Leads to varying performance and reliability requirements

Deconstructing the Browser App

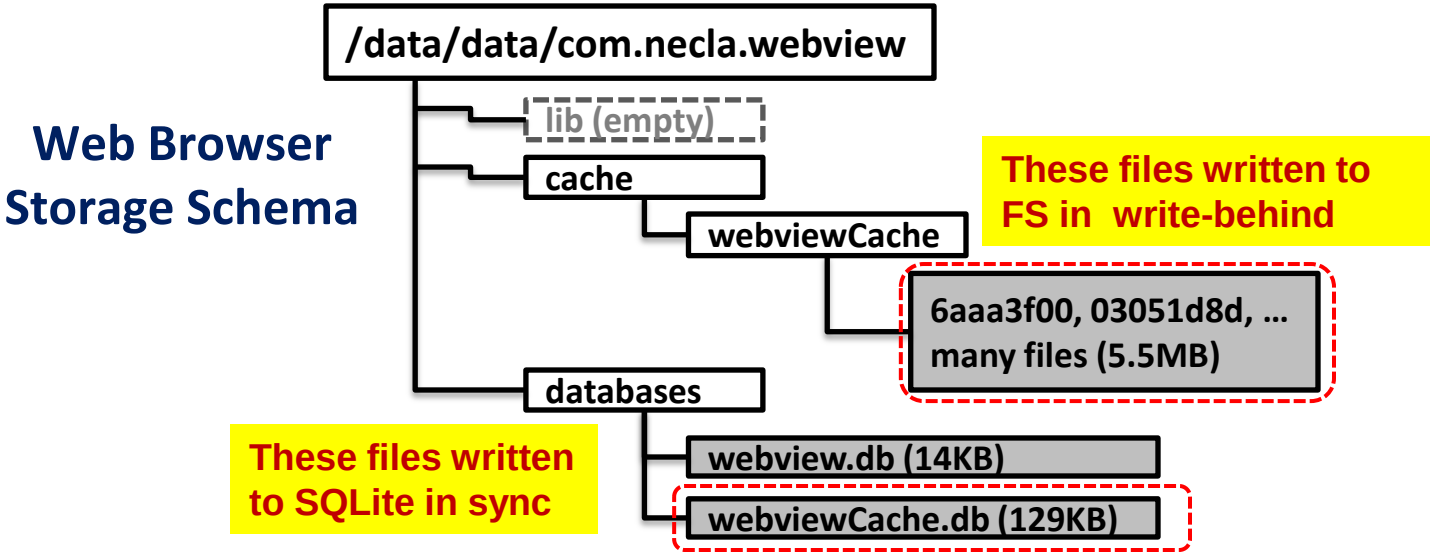
- Reference architecture of a browser



How Does the Browser Use Storage?

- Storage schema for a web browser application

– Uses both SQLite and FS interfaces to store different data



- Apps typically store some data in FS (e.g., cache files) and some in a SQLite database (e.g., cache map)

– Data through SQLite is written synchronously → slow but reliable

– Larger files typically written to FS → fast but less reliable

Programming for Storage on Mobile Devices

APIs, APIs, APIs

- App view of storage and data heavily dictated by APIs
  - Android has one, iOS has one
- Modularity is great for the mobile app developer!
  - Only need to know what the API offers in terms of storage options
  - Few choices to make (examples coming up)
  - Easy for naïve developers to write apps that deliver certain functionality
  - Easy to enforce system-wide policies, data sharing, isolation
- Modularity can have negative consequences
  - Can't pick and choose beyond what the API offers
  - If the API is restrictive, the app developer is stuck with it
  - Unintended side effects on performance and resource utilization

Handling Data on Android

- Key-value API
  - Very popular among apps to store any kind of application and/or user data ; Android uses SQLite databases to provide structured storage
- File system API
  - File I/O interface for internal and external storage
    - openFileOutput(), read(), write(), and close()
- System-managed data
  - Shared preferences, primitive data types stored in key-value pairs
- Sharing data
  - Expose private data to other apps with a **content provider**
    - **Providing applications with access to a user's phone log, or contacts**
  - Provides read/write access to app data subject to imposed restrictions
- Network data and Android data backup
  - Store and retrieve data through Android backup or other web services

Handling Data on Apple iOS

- iOS also uses SQLite to store application data
  - iOS Core Data is a data model framework built on top of SQLite
  - Core data takes care of glue code so apps don't have to worry about SQL syntax in their UI
  - Provides applications access to common functionality such as save, restore, undo and redo
- Sharing mechanisms to share data among apps
  - Similar to ContentProvider in Android (or vice versa)
- iOS 4 does not have a central file storage architecture
  - Every file is stored within the context of an application
  - No external storage on iPhones and iPads

Sample App Code for Data Storage

Internal storage

```
// get a FileOutputStream object by passing the file-name; file is private to the app
String string = “test data”;
FileOutputStream dataFileOutput = openFileOutput("datafile", Context.MODE_PRIVATE);
dataFileOutput.write(string.getBytes());
dataFileOutput.close();
```

SD card/external storage

```
// check if external media is available; files on SD can be accessed by other apps/user directly
if(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
        // can use the external storage …
```

SQLite

```
// create a custom SQLite database for the app
public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE Item (ItemID INTEGER, ItemName TEXT);");
}
```

Shared Preferences

```
// get the preferences, then editor, set a data item
SharedPreferences appPrefs = getSharedPreferences("MyAppPrefs", 0);
SharedPreferences.Editor prefsEd = appPrefs.edit();
prefsEd.putString("dataString", "some string data");
prefsEd.commit();
```

Best Practices and Tradeoffs

- Multiple mechanisms for storage available on mobile platforms
  - Local w/ sync, local w/o sync, network, …
  - Performance and reliability are at odds (as always); judiciously choose type of storage based on needs of the app
    - Don't throw everything in the SQLite databases, partition your schema carefully
- I/O operations can take noticeable time, especially true for interactive apps
  - Avoid storage I/O in the main UI thread; separate threads for I/O operations
- Mobile storage is heavily used as a cache (Browser, Facebook, etc)
  - Consider what data is worth caching, what is best brought fresh over network
  - Tradeoff becomes more relevant on WiFi (penalty of storage I/O becomes non-trivial)
- Choice of storage partition determines data privacy
  - Store on internal media for private app data, store on external for shared
  - Use OS mechanisms to explicitly share data across apps (i.e., ContentManager)

Open Questions for Mobile Storage

- How are the apps using the storage?
  - What is the state of the art?
- What are choices for storage on mobile devices?
  - Flash-based widely popular right now
  - PCM? PCM as a buffer?
  - Plenty of challenges in low-level design and implementation
    - FTL strategies for mobile flash need to operate with limited power + DRAM
- What is the right storage abstraction for mobile apps?
  - How much information should be exposed to the apps?
    - What is the right storage API for mobile app development?
  - How is storage being managed underneath?
- Interplay of storage and wide-area networks raises interesting challenges (and research potential!)

Are there interesting problems to be solved in mobile storage?


amazon
MP3
MADDEN NFL 12
OpenTable
PANDORA
SwiftKey X
LOADING
NETFLIX
GasBuddy.com
Flixster
IMDb

Waiting is undesirable!

Annoying for the user

More so for interactive mobile users

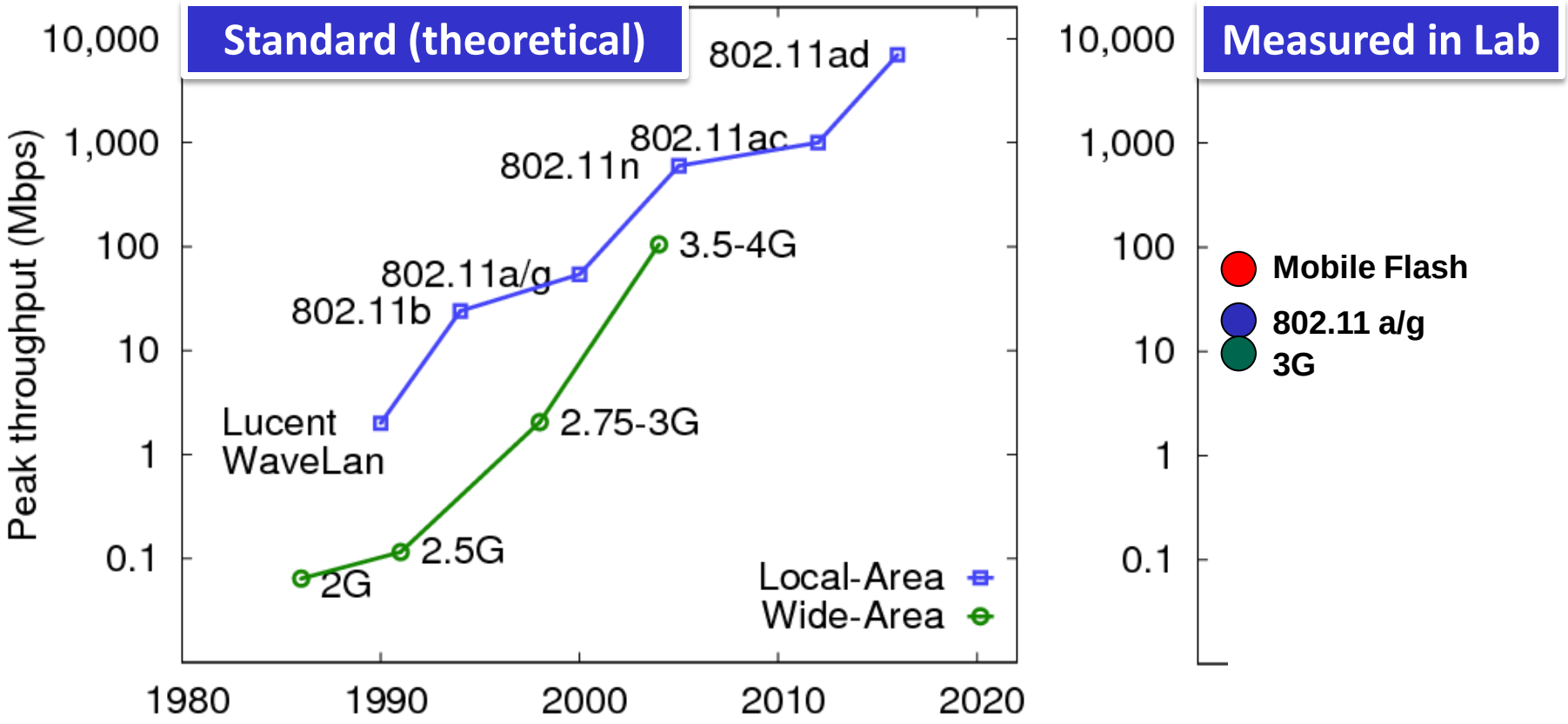
More time, more battery

Easy to lose customers

Aren't network and CPU the real problem?
Why are we talking about storage?

Wireless Network Throughput Progression

NEC Laboratories America
Relentless passion for innovation



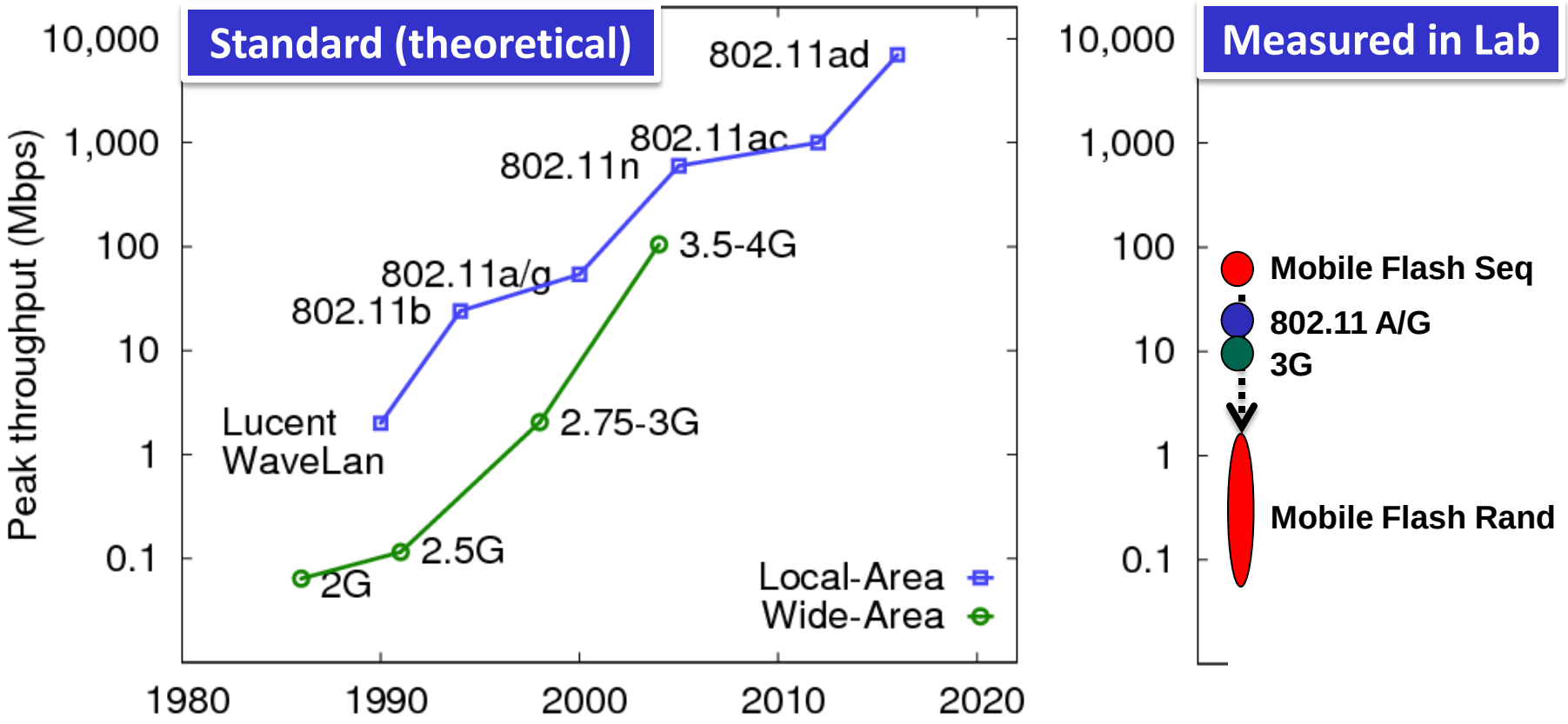
- Flash storage on mobile performs better than wireless networks
- Most apps are interactive; as long as performance exceeds that of the network, difficult for storage to be bottleneck

[FALSE]

Why Storage is a Problem

Shifting Performance Bottlenecks

NEC Laboratories America
Relentless passion for innovation



- 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

Revisiting Storage for Smartphones

Why Storage is a Problem

Random versus Sequential Disparity

NEC Laboratories America
Relentless passion for innovation

- 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

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

Consumer-grade SD performance

Performance MB/s

However, we find that for several popular apps, substantial fraction of I/O is random writes (including web browsing!)

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

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

NEC Laboratories America
Relentless passion for innovation

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,

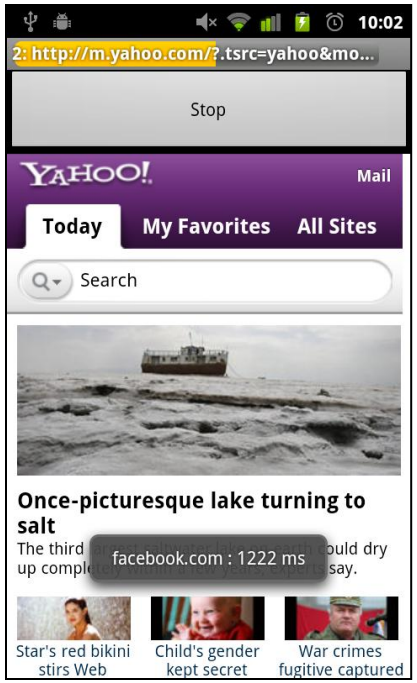
News, Weather, Calendar,

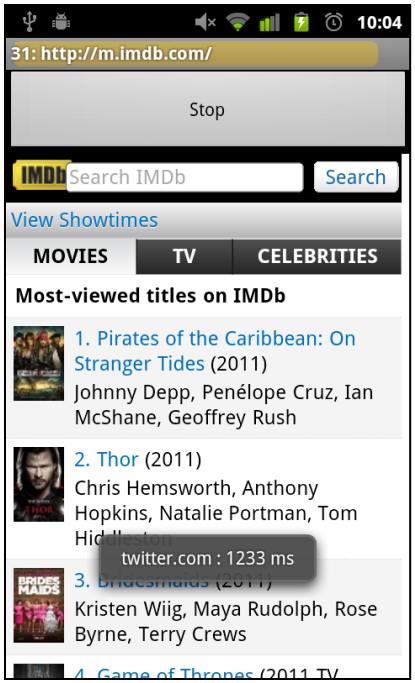
Facebook, Market, Twitter

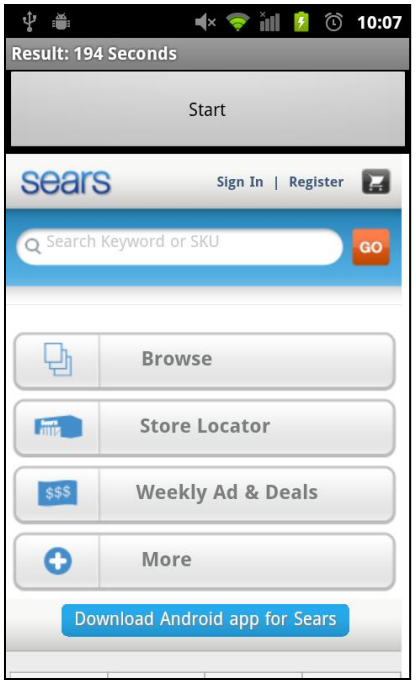
Application Workload: Webbench

NEC Laboratories America
Relentless passion for innovation

- Wrote a custom benchmark based on WebKit to mimic web browsing
- Visits 50 websites* continuously, reports elapsed time in seconds
- HTTP Proxy server is used to minimize external network effect
- Screen shots:

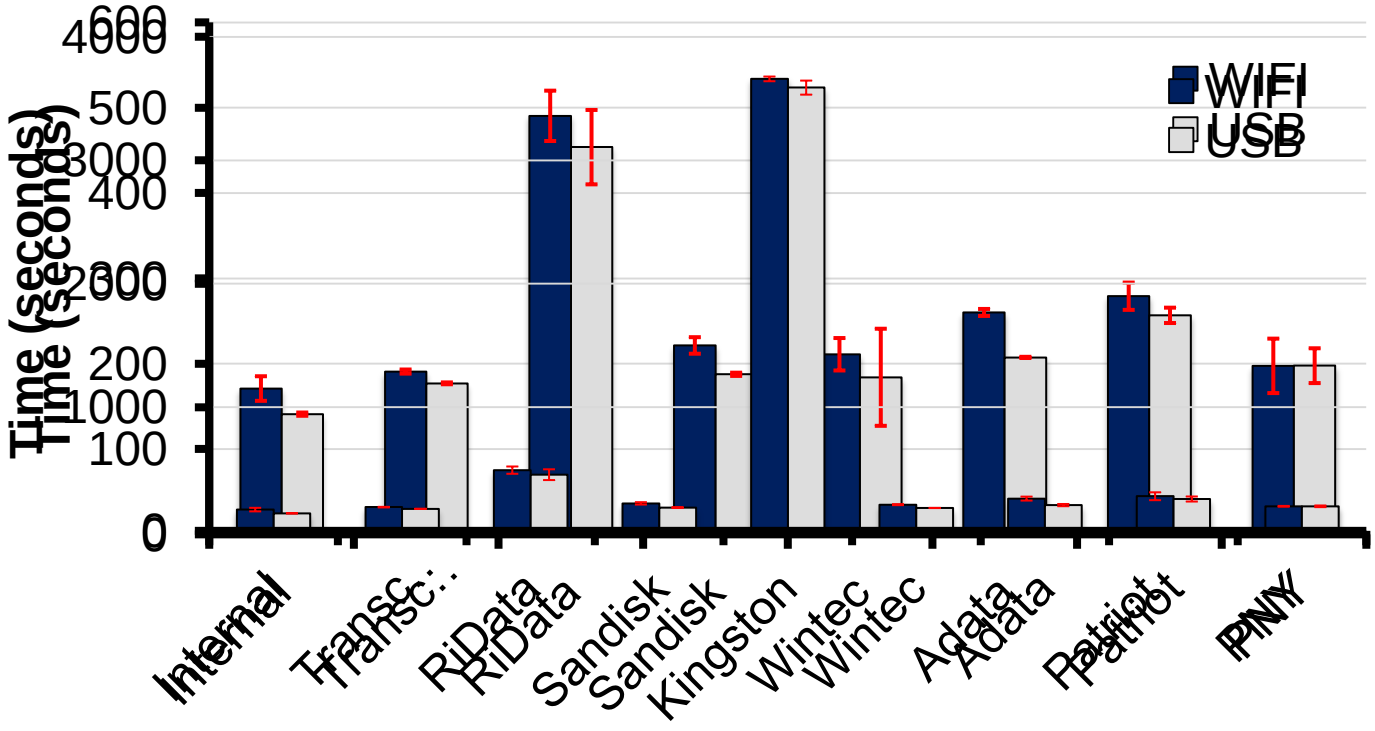


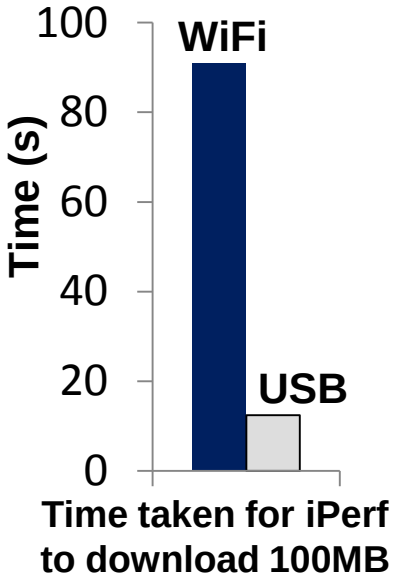




Experimental Evaluation

WebBench Results: Runtime





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%)

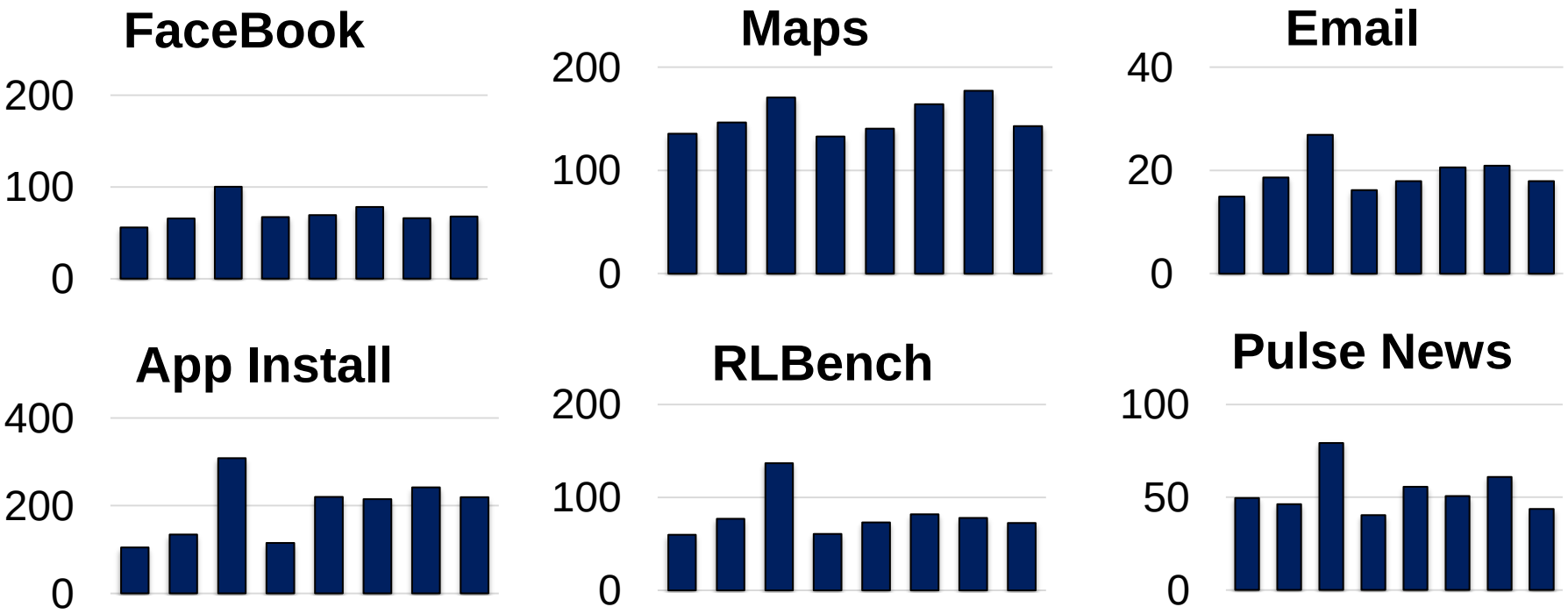
Storage significantly affects app performance and consequently user experience

With a faster network (USB in RT), variance was 222% (without Kingston)

With 10X increase in N/W speed, hardly any difference in runtime

Runtimes for Popular Apps (without Kingston)

NEC Laboratories America
Relentless passion for innovation



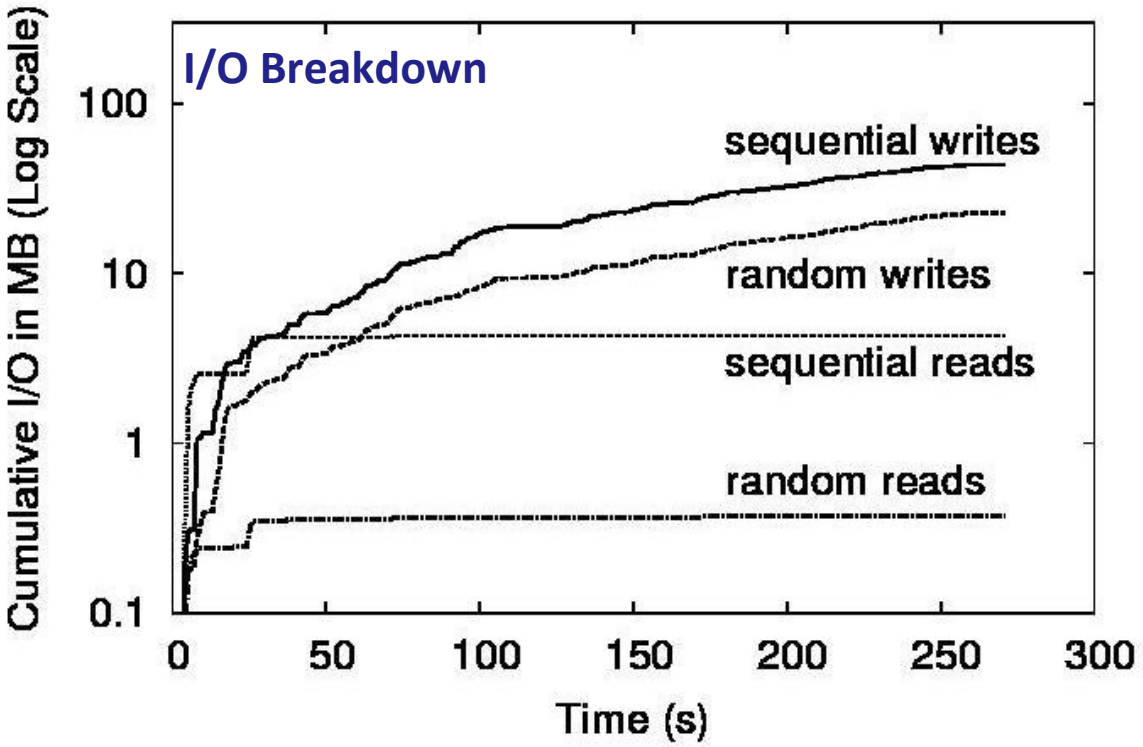
We find a similar trend for several popular apps

Storage device performance 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

NEC Laboratories America
Relentless passion for innovation

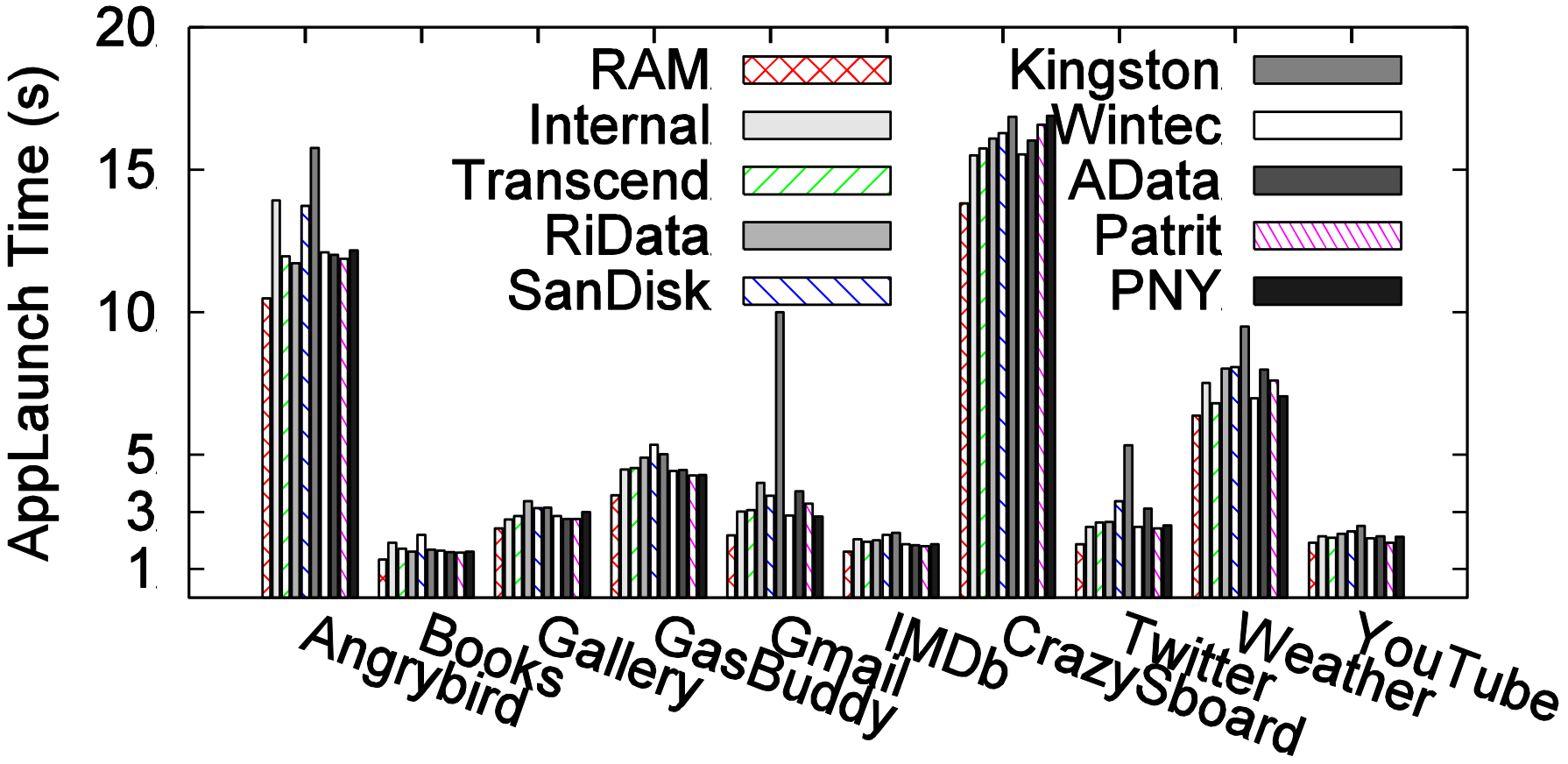


Vendor	Seq:Rand perf ratio	Rand IOPS
Transcend	4	302
Sandisk	8	179
RiData	395	5
Kingston	490	2.6
Wintec	1500	2.6
A-Data	1080	2.6
Patriot	1050	2.6
PNY	1530	2.6

- 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

Application Launch Performance



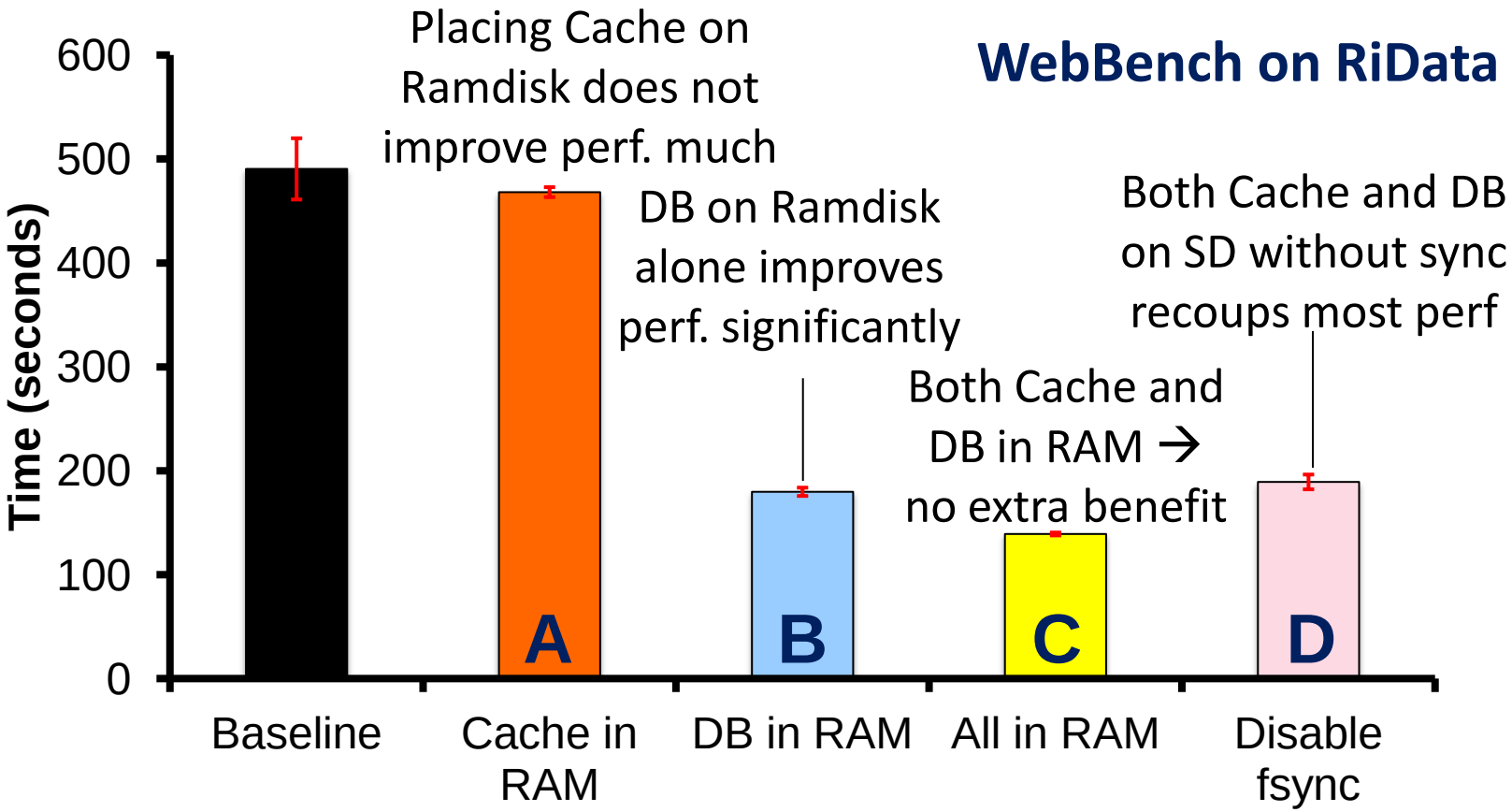
What-If Analysis for Solutions

NEC Laboratories America
Relentless passion for innovation

What is the potential for improvements?

- E.g., if all data *could* be kept in RAM?
- Analysis to answer hypothetical questions

A. Web Cache in RAM
B. DB (SQLite) in RAM
C. All in RAM
D. All on SD w/ no-sync

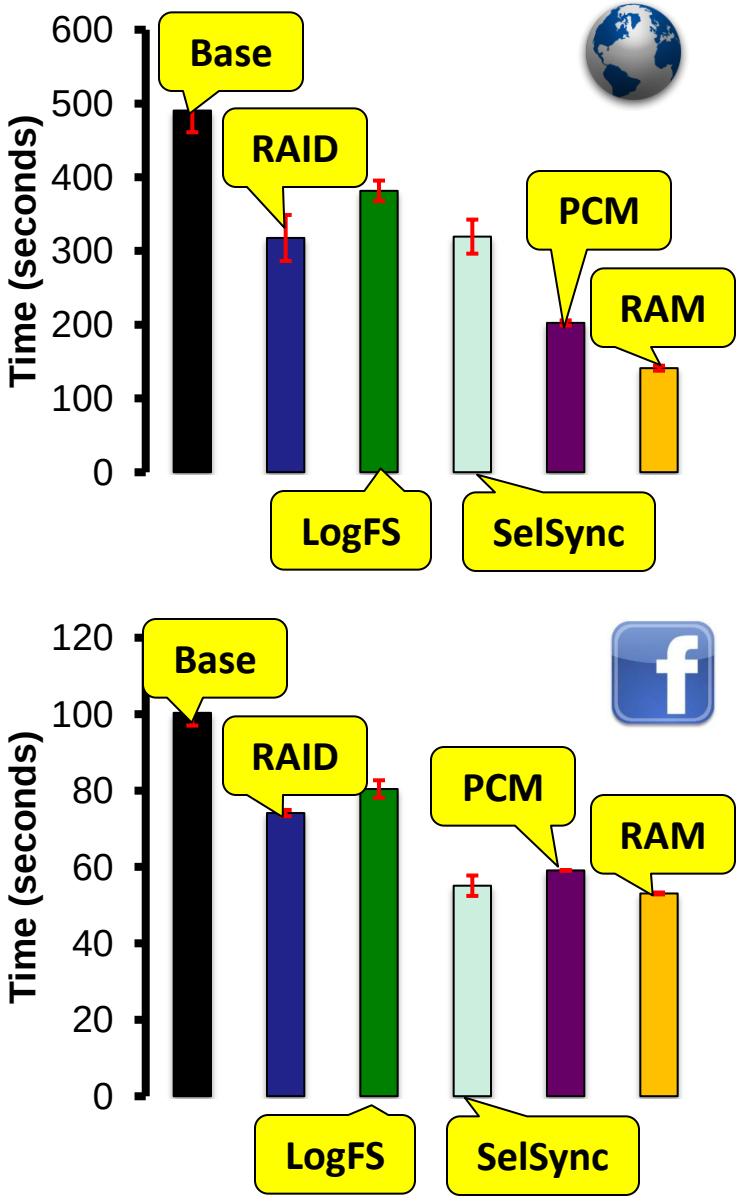


Implications of Experimental Analysis

- 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

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



WebBench on RiData

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