Cellular Networks and Mobile Computing
COMS 6998-8, Spring 2012

Instructor: Li Erran Li
(lierranli@cs.columbia.edu)

http://www.cs.columbia.edu/~coms6998-8/

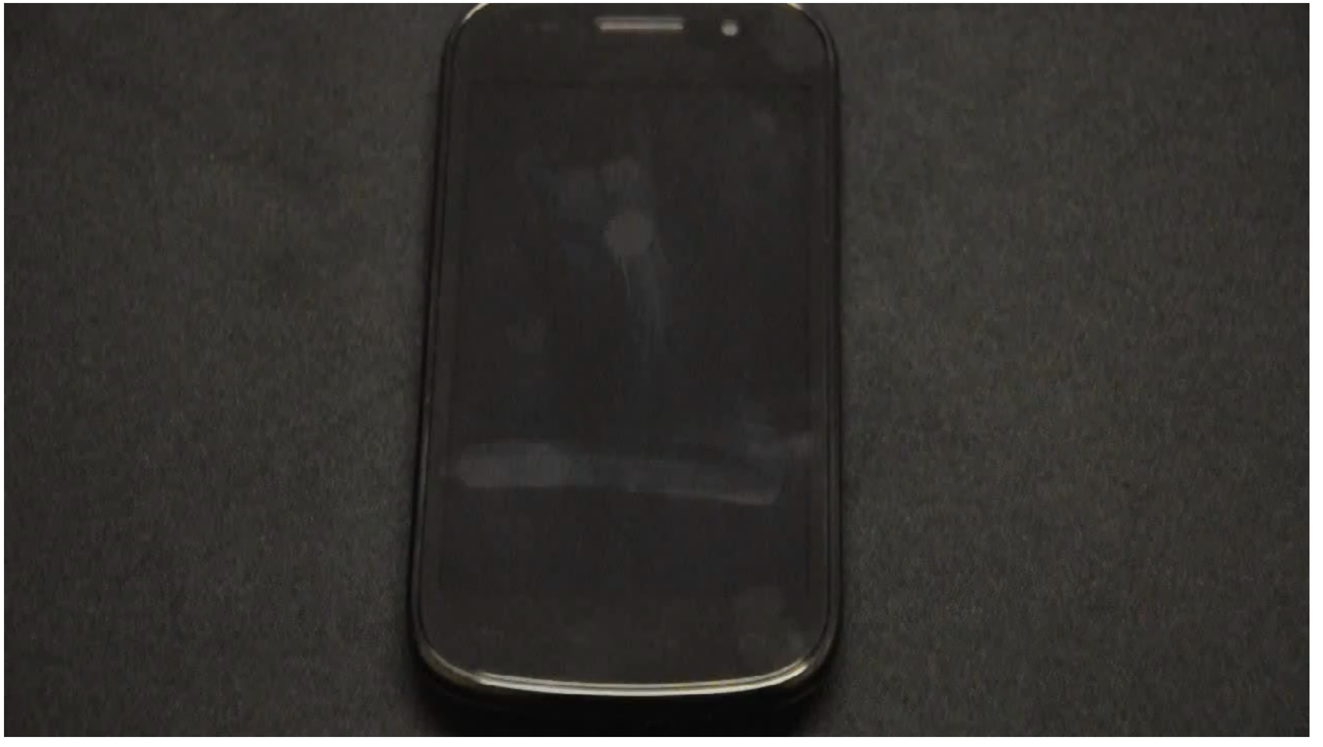
3/19/2012: Smart phone virtualization and storage

Announcements

- Preliminary project report due next week March 26th
- There will be two advanced programming lab sessions: one for iOS and one for Android
  - Email me the topics you would like to cover

Smart Phone Virtualization

Cells video demo



Personal Phone

Business Phone

Developer Phone

Children’s Phone

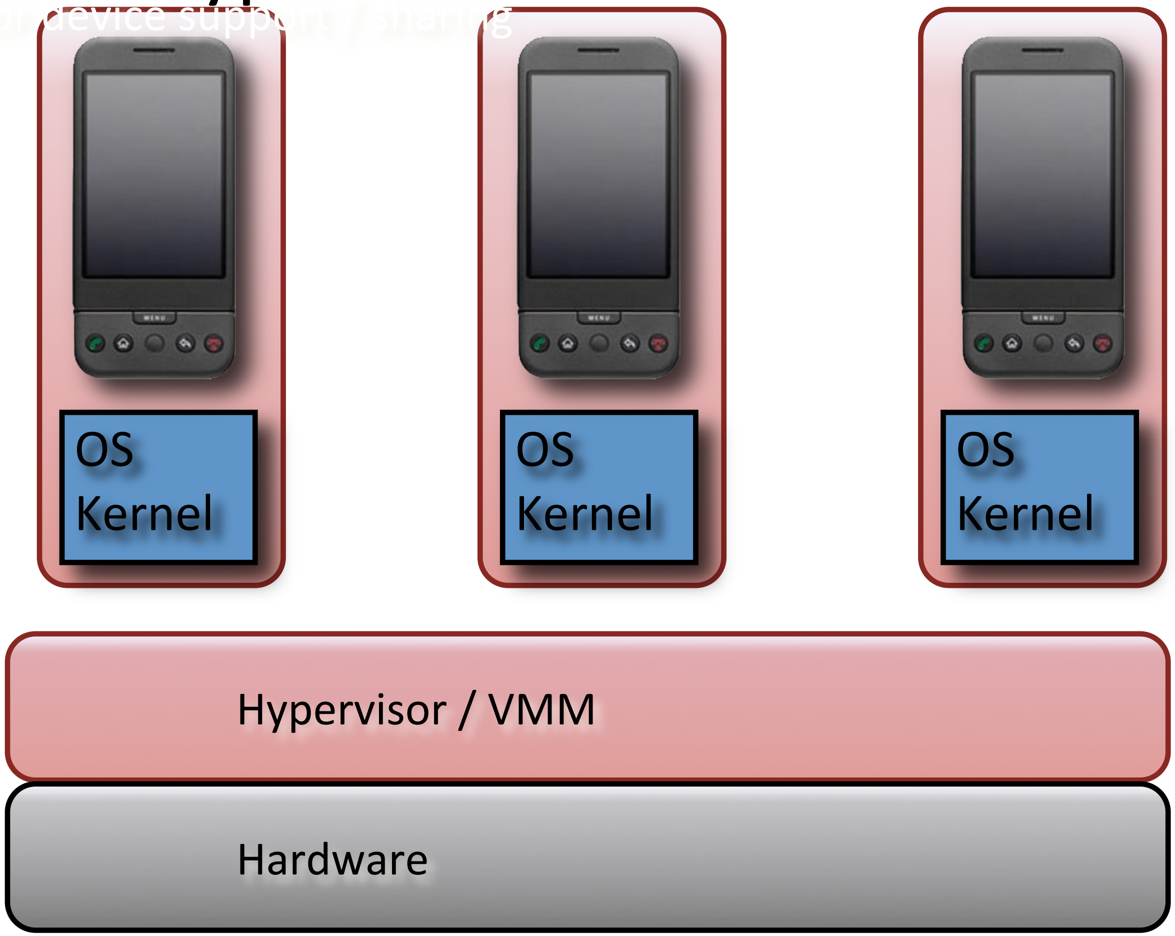
Courtesy: Jason Nieh et al.

Virtualization

Courtesy: Jason Nieh et al.

Server Virtualization

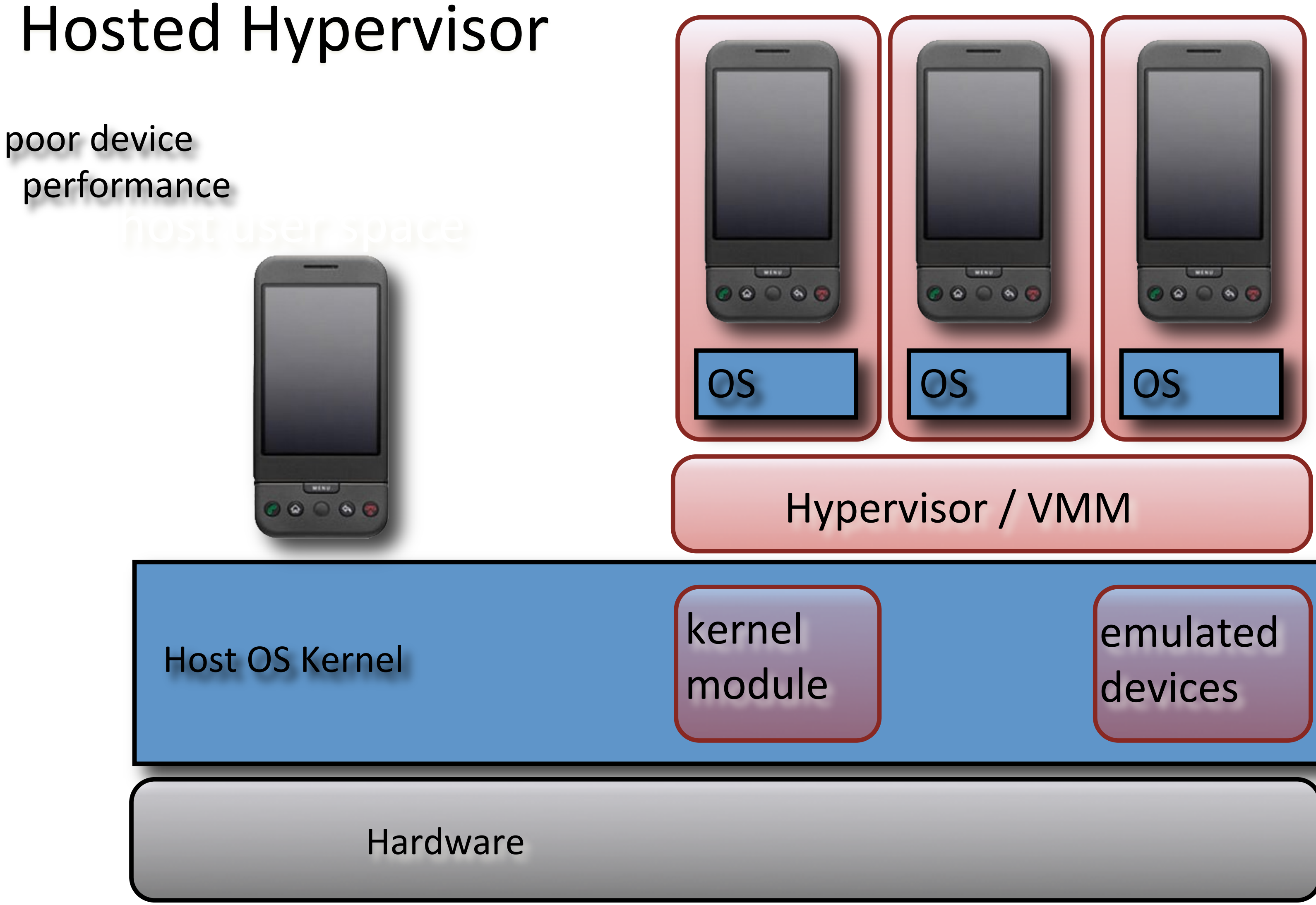
Bare-Metal Hypervisor



Courtesy: Jason Nieh et al.

Desktop Virtualization

Hosted Hypervisor

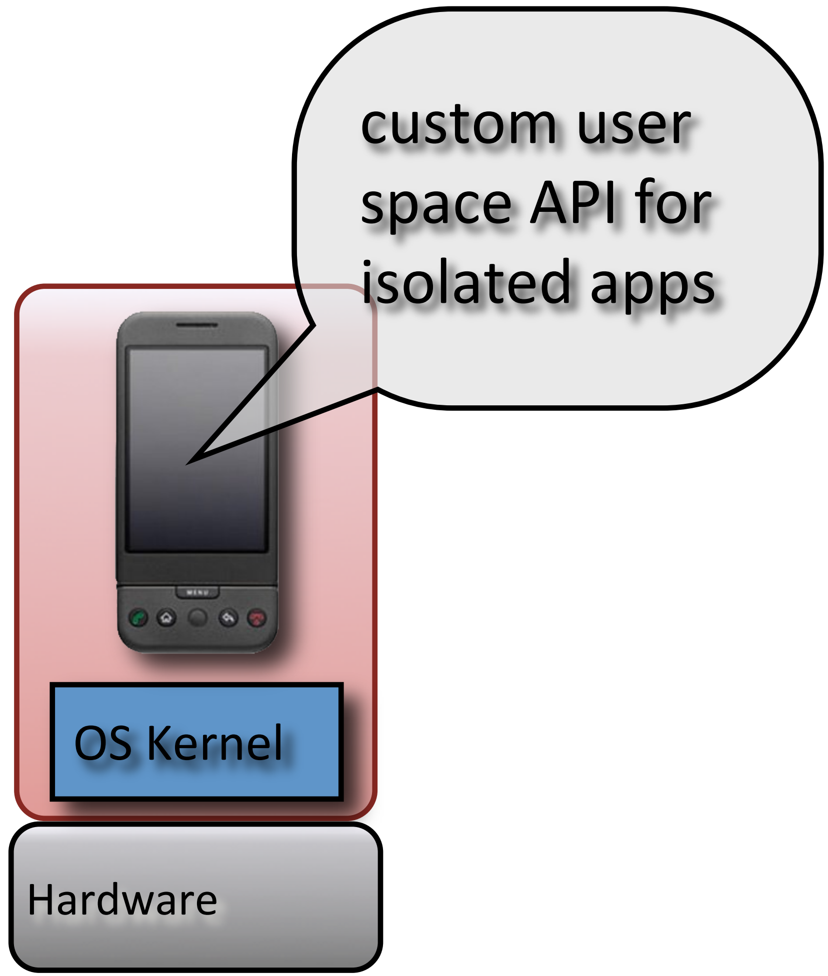


Courtesy: Jason Nieh et al.

Non-Virtualization

User Space SDK

no standard apps
less secure



Courtesy: Jason Nieh et al.

Key Challenges

- device diversity

		microphone	headset
Power	Touchscreen	Buttons	GPS
Cell Radio	WiFi	GPU	Framebuffer
h.264 accel.	pmem	Binder IPC	Compass
camera(s)	speakers	Accelerometer	RTC / Alarms

- mobile usage model

➡ graphics accelerated UI

Courtesy: Jason Nieh et al.

Cells

Key Observation

one app at a time

large: lots of windows/apps

Courtesy: Jason Nieh et al.

Cells

Key Observation

screen real-estate is limited, and mobile phone users are accustomed to interacting with *one thing* at time

Courtesy: Jason Nieh et al.

Cells

Usage Model

foreground / background

Courtesy: Jason Nieh et al.

Cells

Complete Virtualization

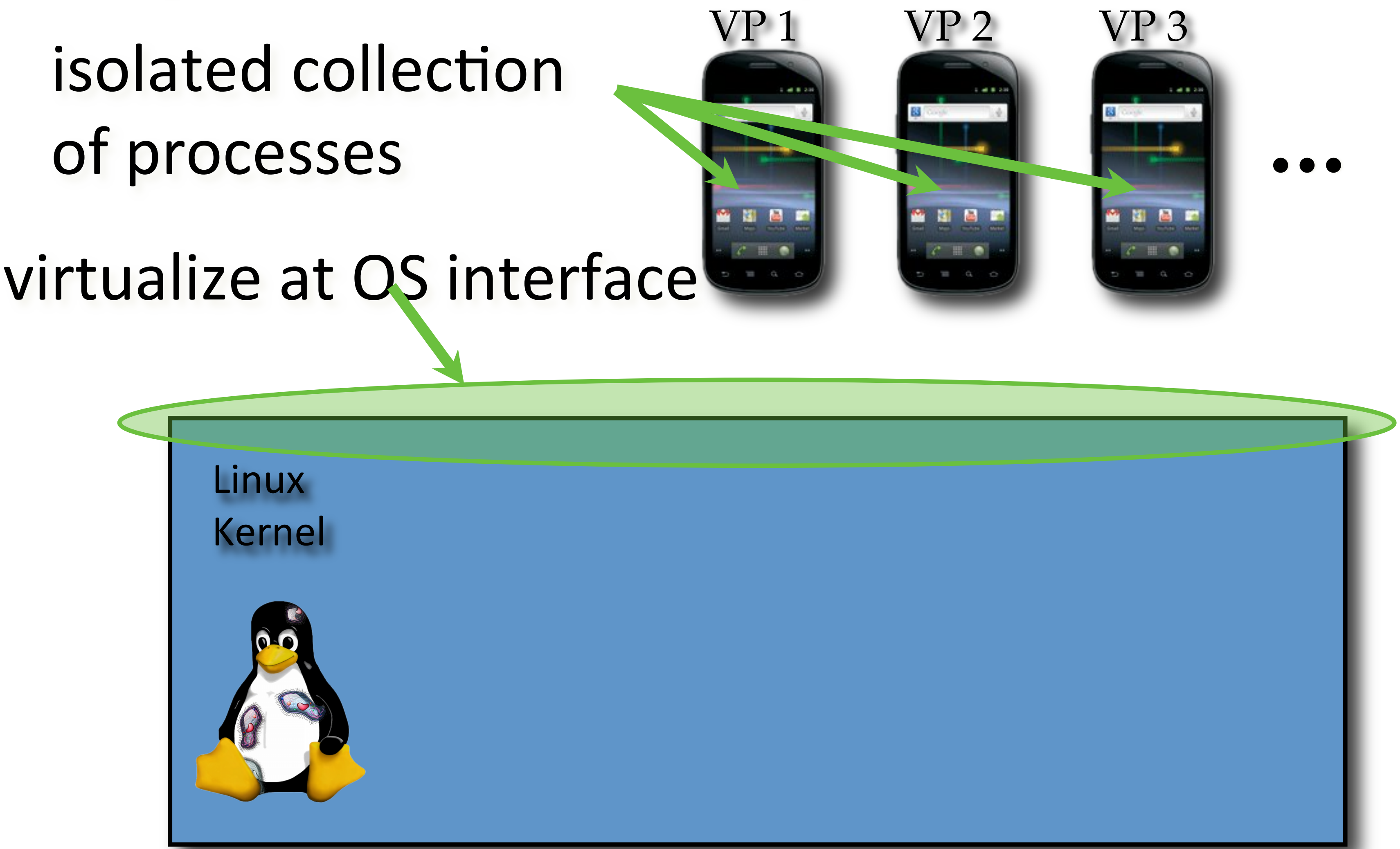
- multiple, isolated virtual phones (VPs) on a single mobile device
- 100% device support in each VP
  - unique phone numbers - single SIM!
  - accelerated 3D graphics!

 Courtesy: Jason Nieh et al.

Cells
Efficient Virtualization

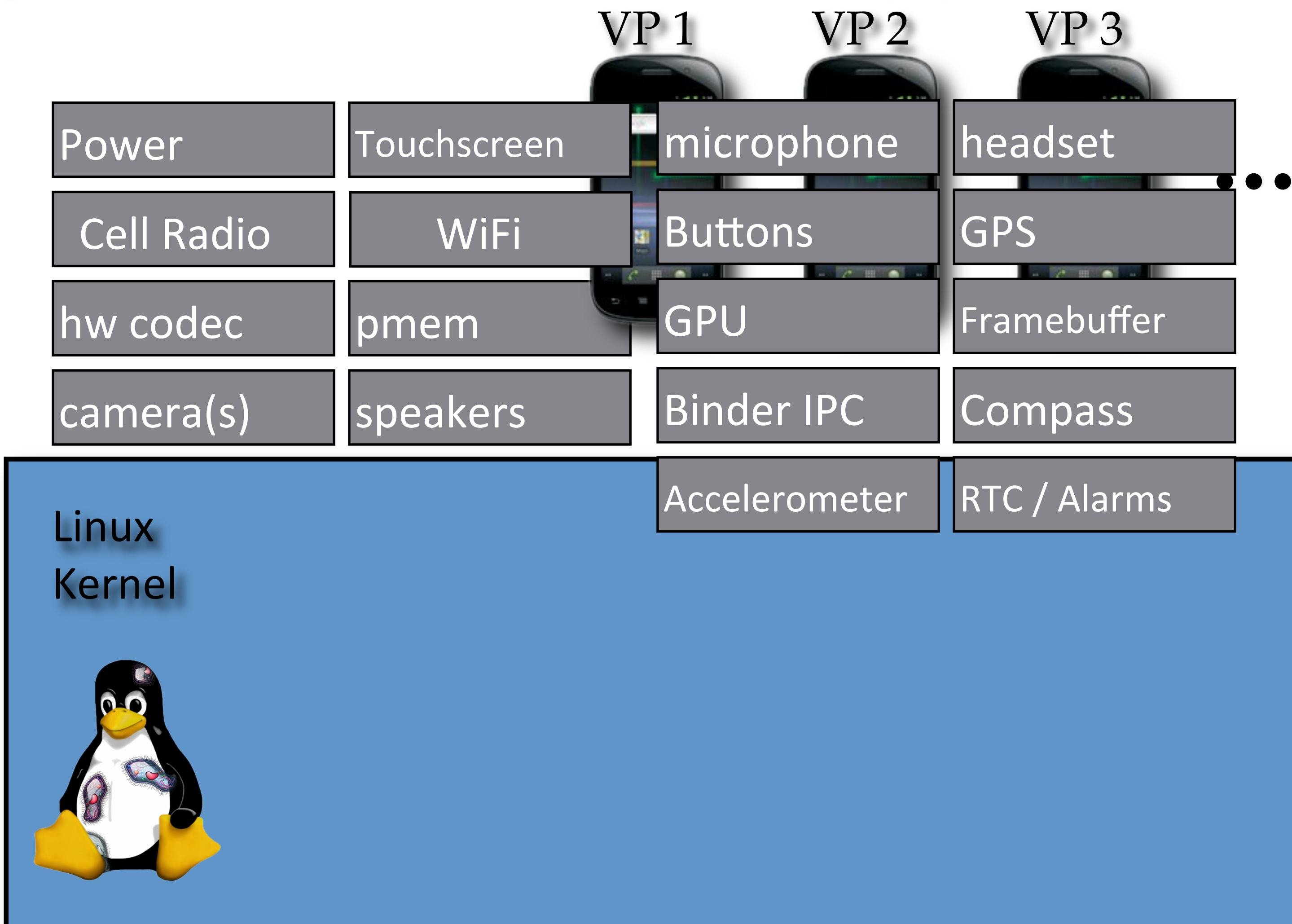
- less than 2% overhead in runtime tests
- imperceptible switch time among VPs

Single Kernel: Multiple VPs



Courtesy: Jason Nieh et al.

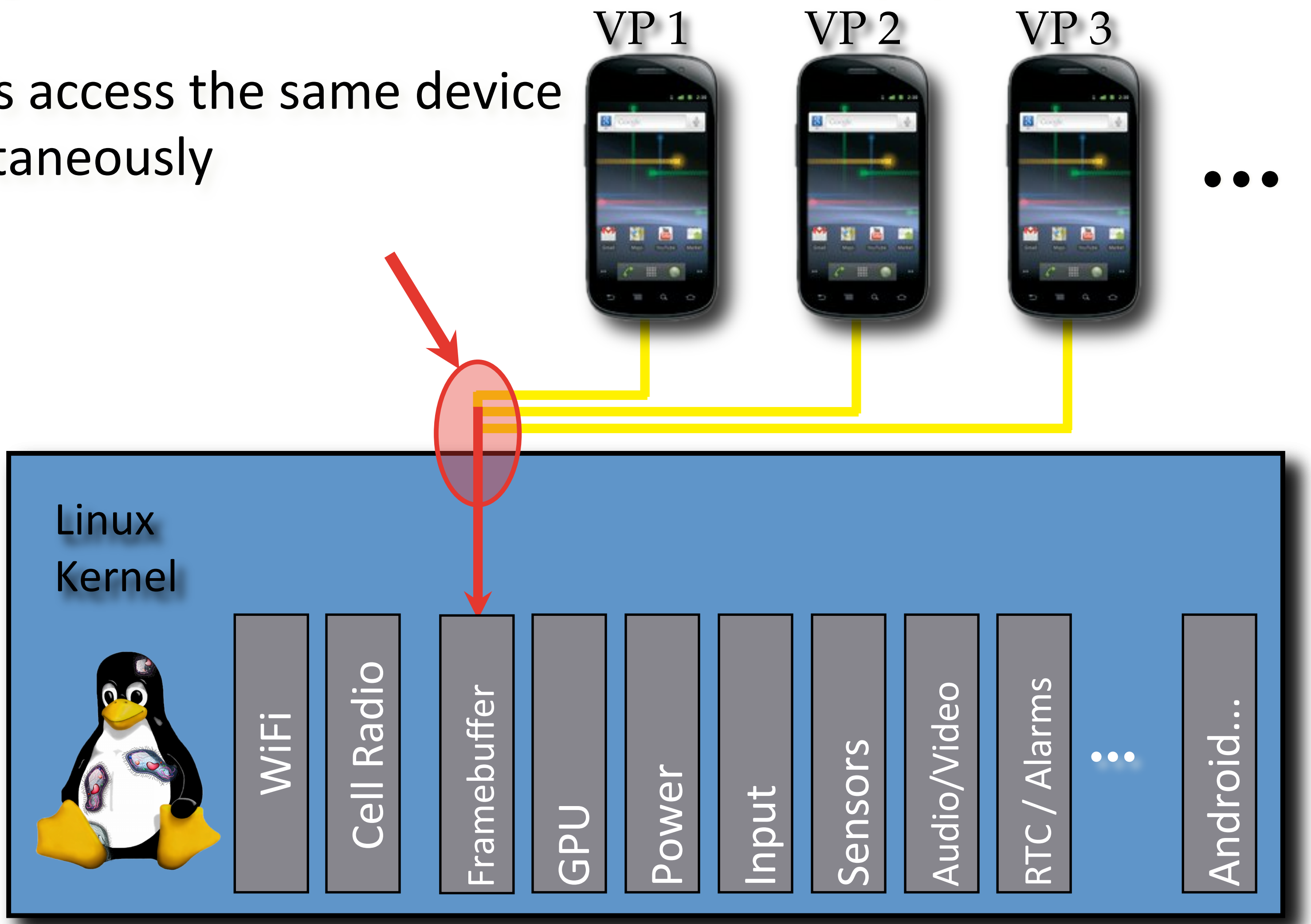
Single Kernel: Device Support



Courtesy: Jason Nieh et al.

Single Kernel: Device Support

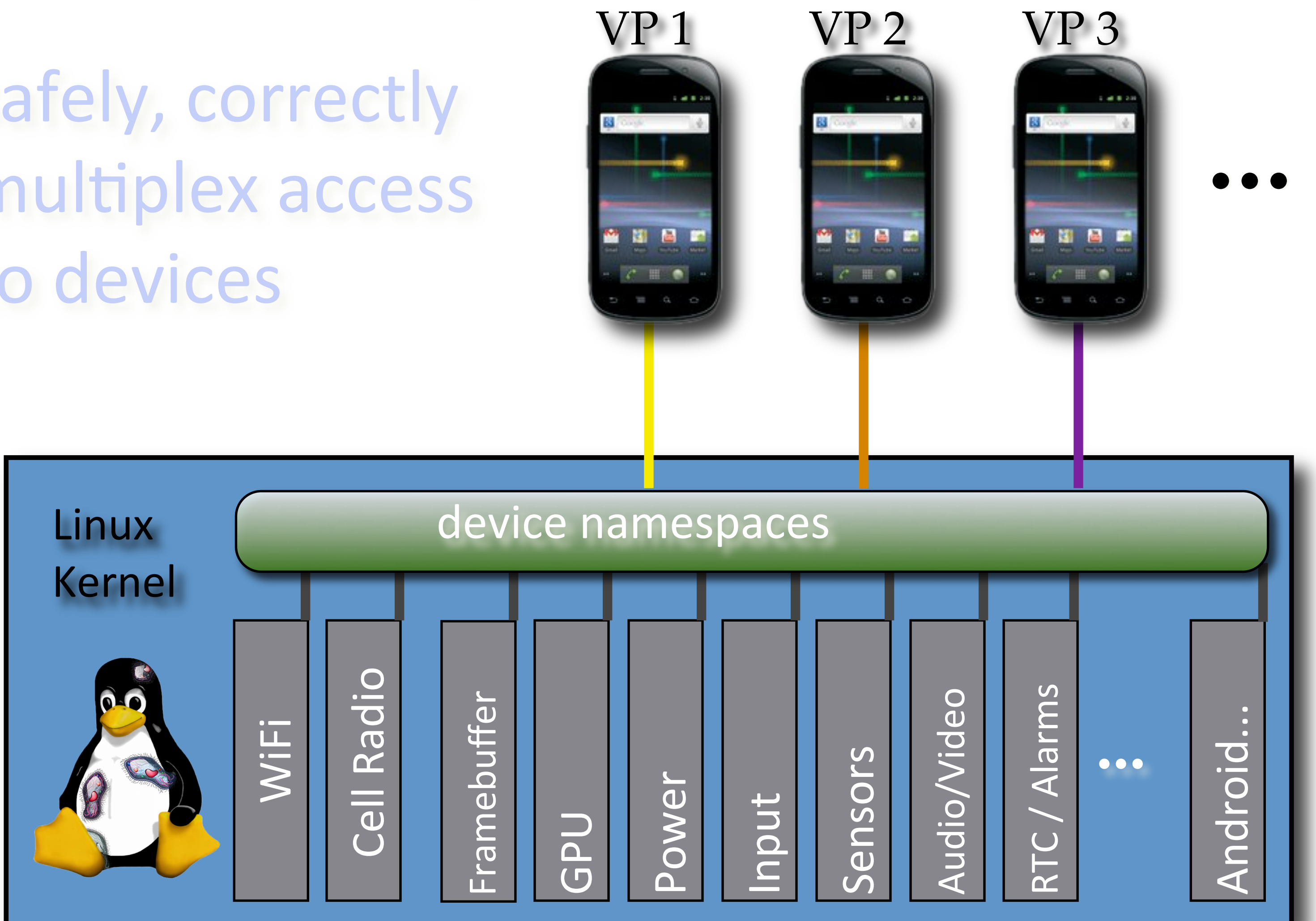
all VPs access the same device simultaneously



Courtesy: Jason Nieh et al.

Device Namespaces

safely, correctly multiplex access to devices



Courtesy: Jason Nieh et al.

Cells

device namespaces

+

foreground / background

=

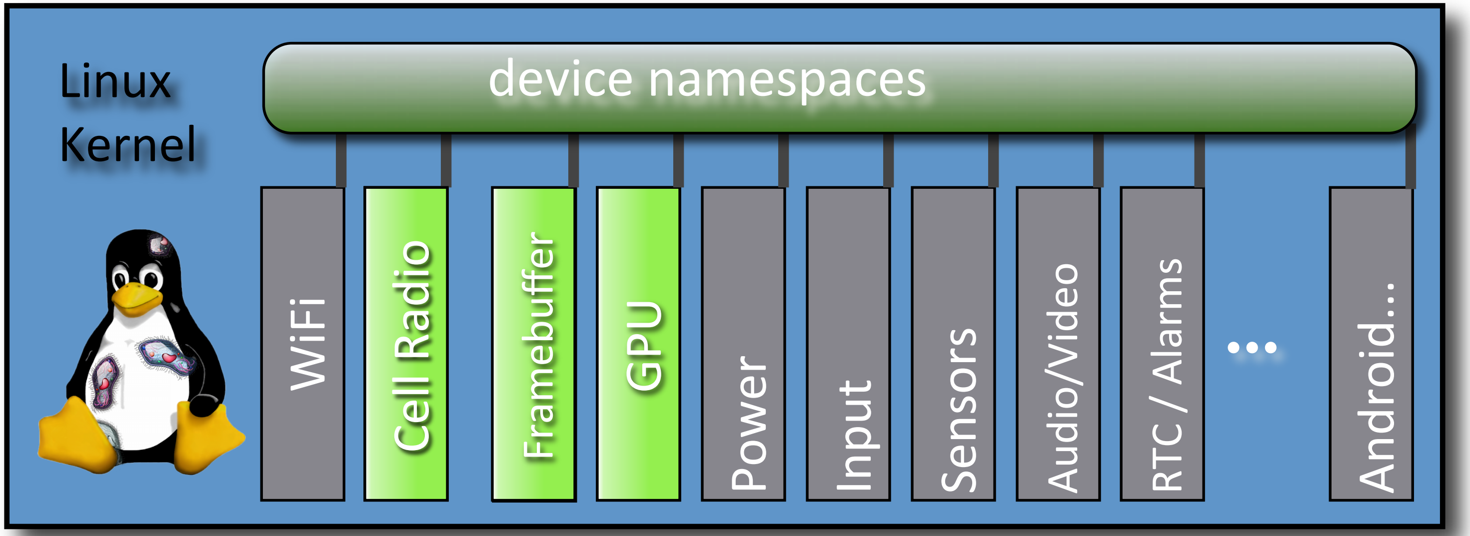
Complete, Efficient, Transparent Mobile Virtualization

Courtesy: Jason Nieh et al.

efficient basic graphics virtualization

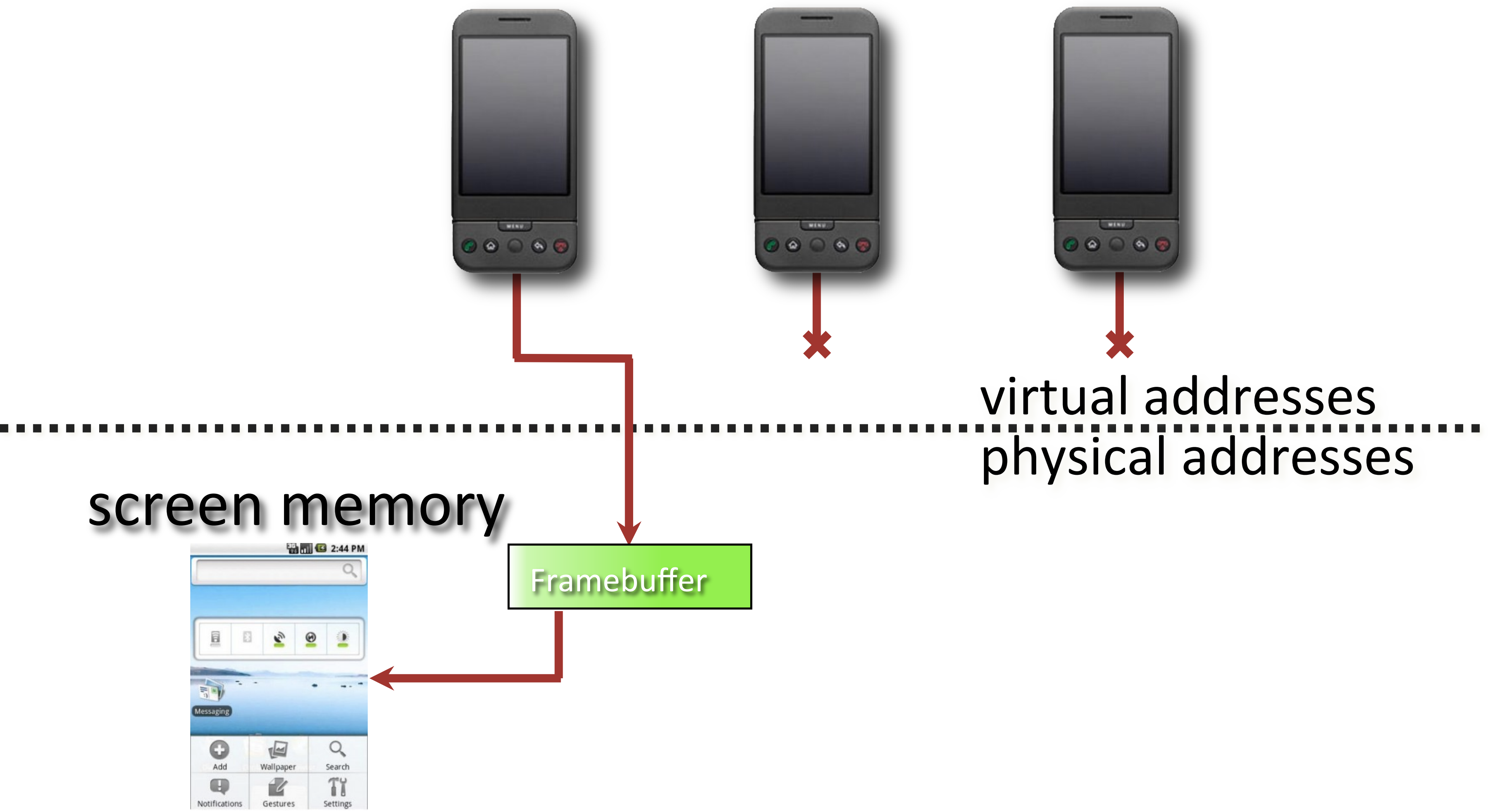
hardware accelerated graphics

proprietary/closed interface

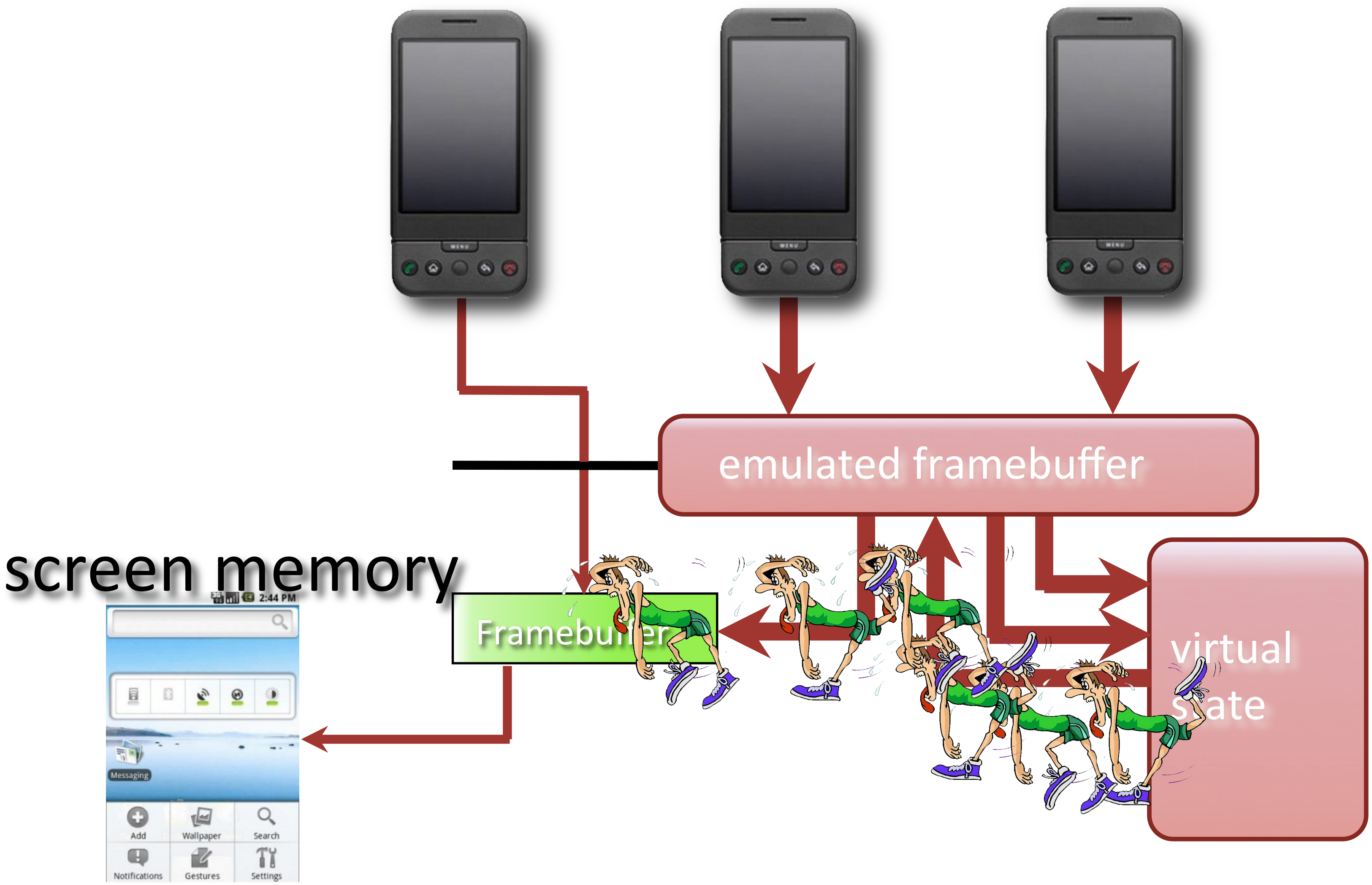


Courtesy: Jason Nieh et al.

Approach 1: Single Assignment

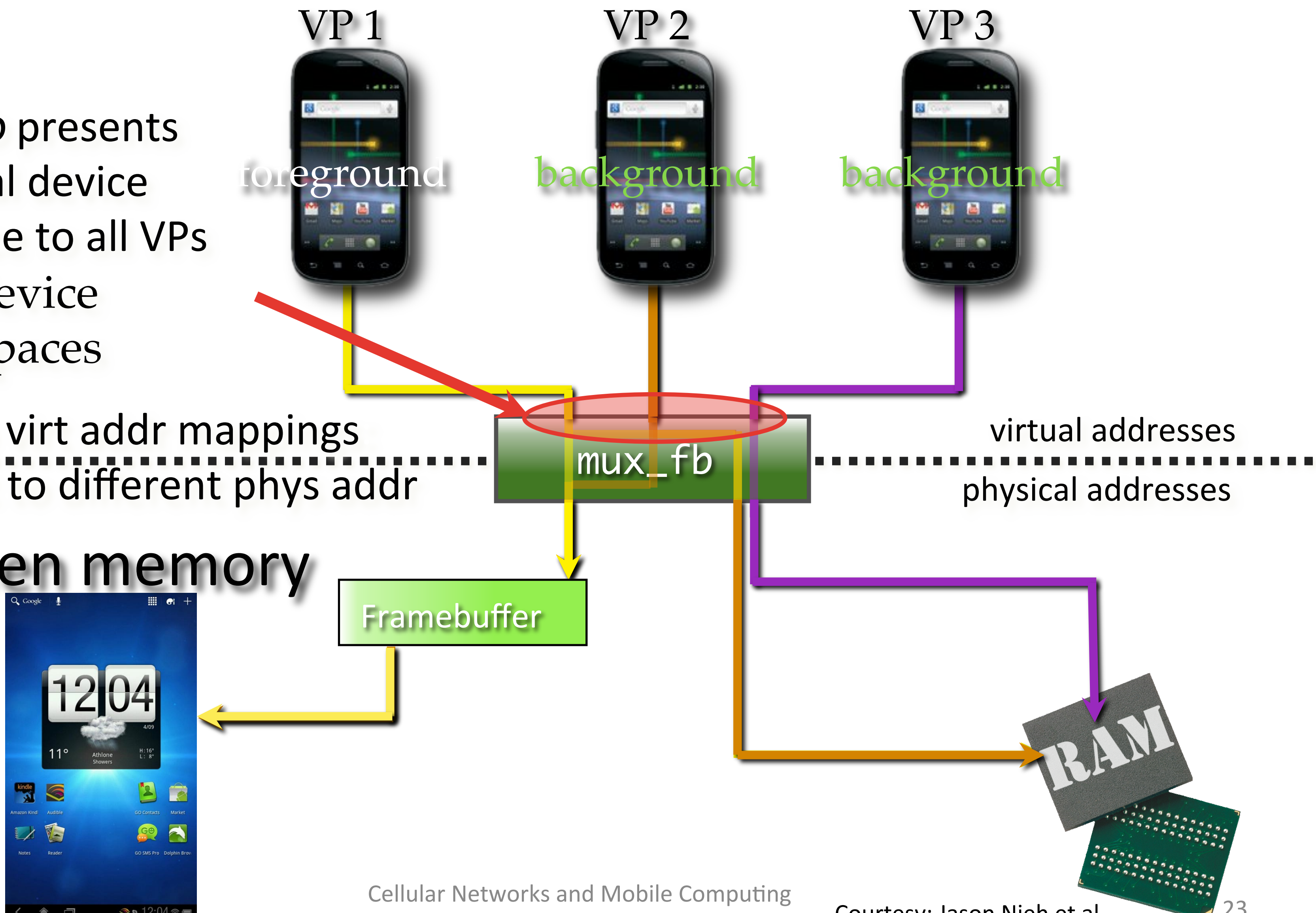


Approach 2: Emulated Hardware



Cells: Device Namespaces

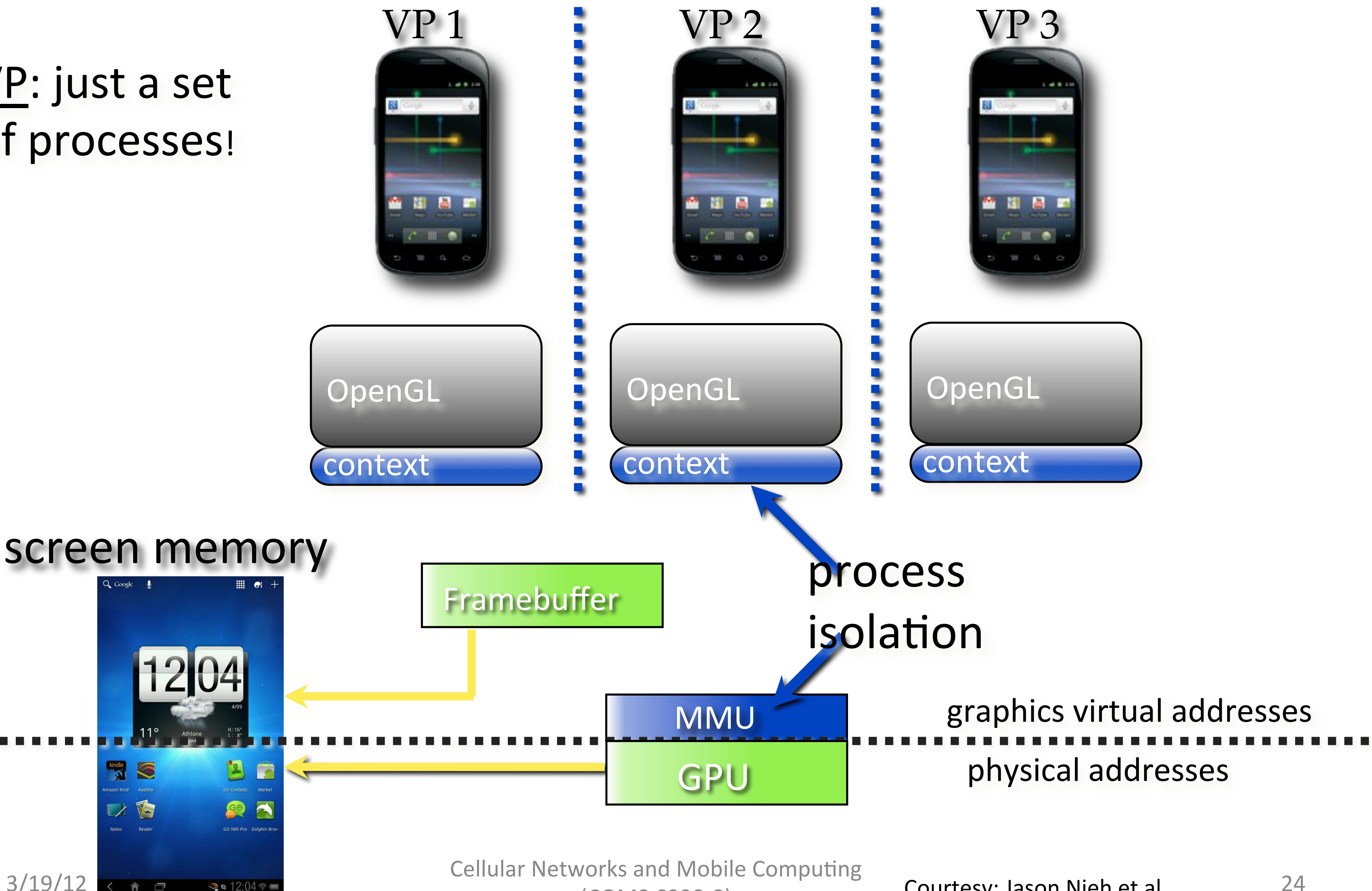
mux_fb presents identical device interface to all VPs using device namespaces



Courtesy: Jason Nieh et al.

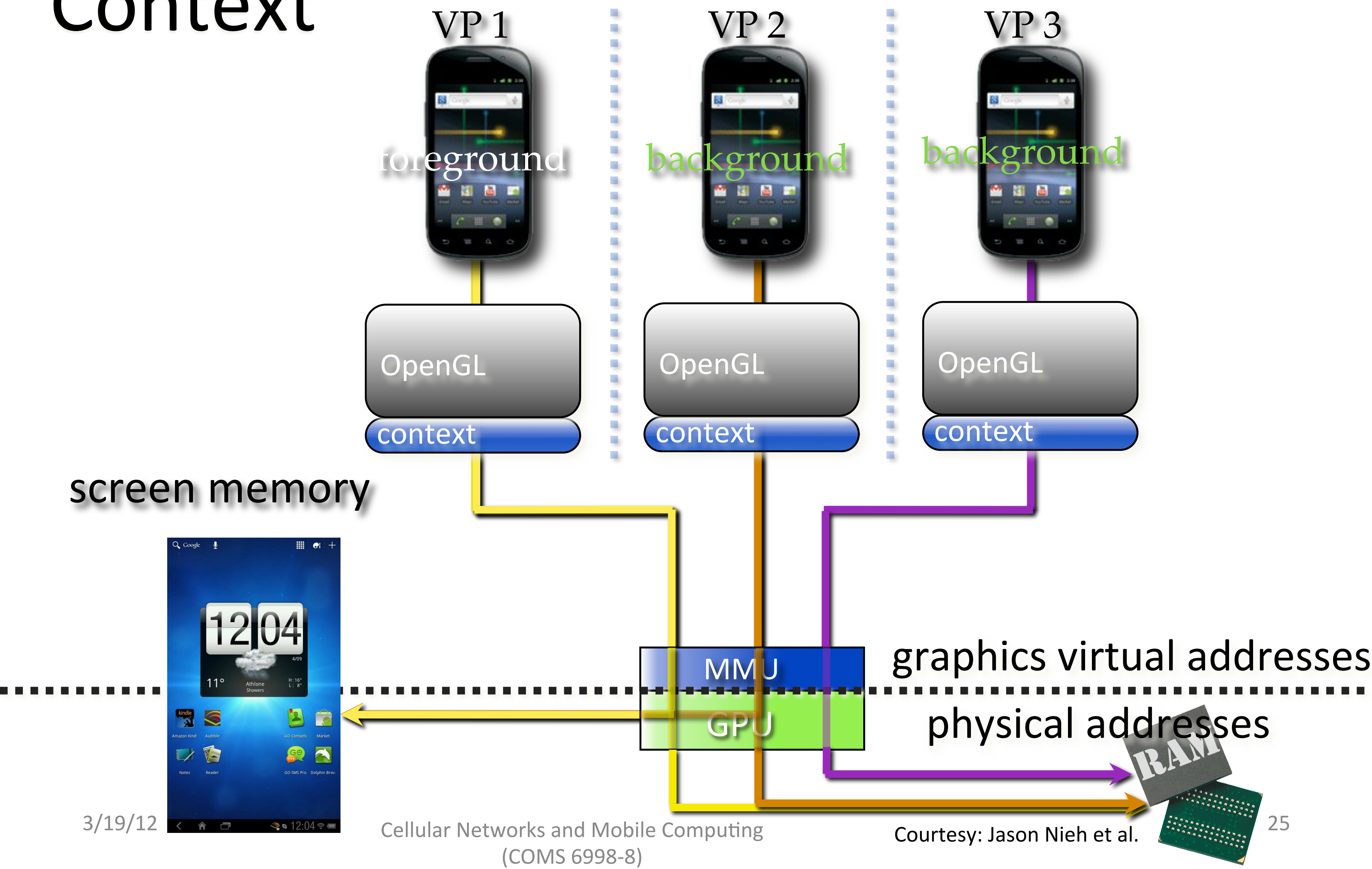
Accelerated Graphics

VP: just a set of processes!



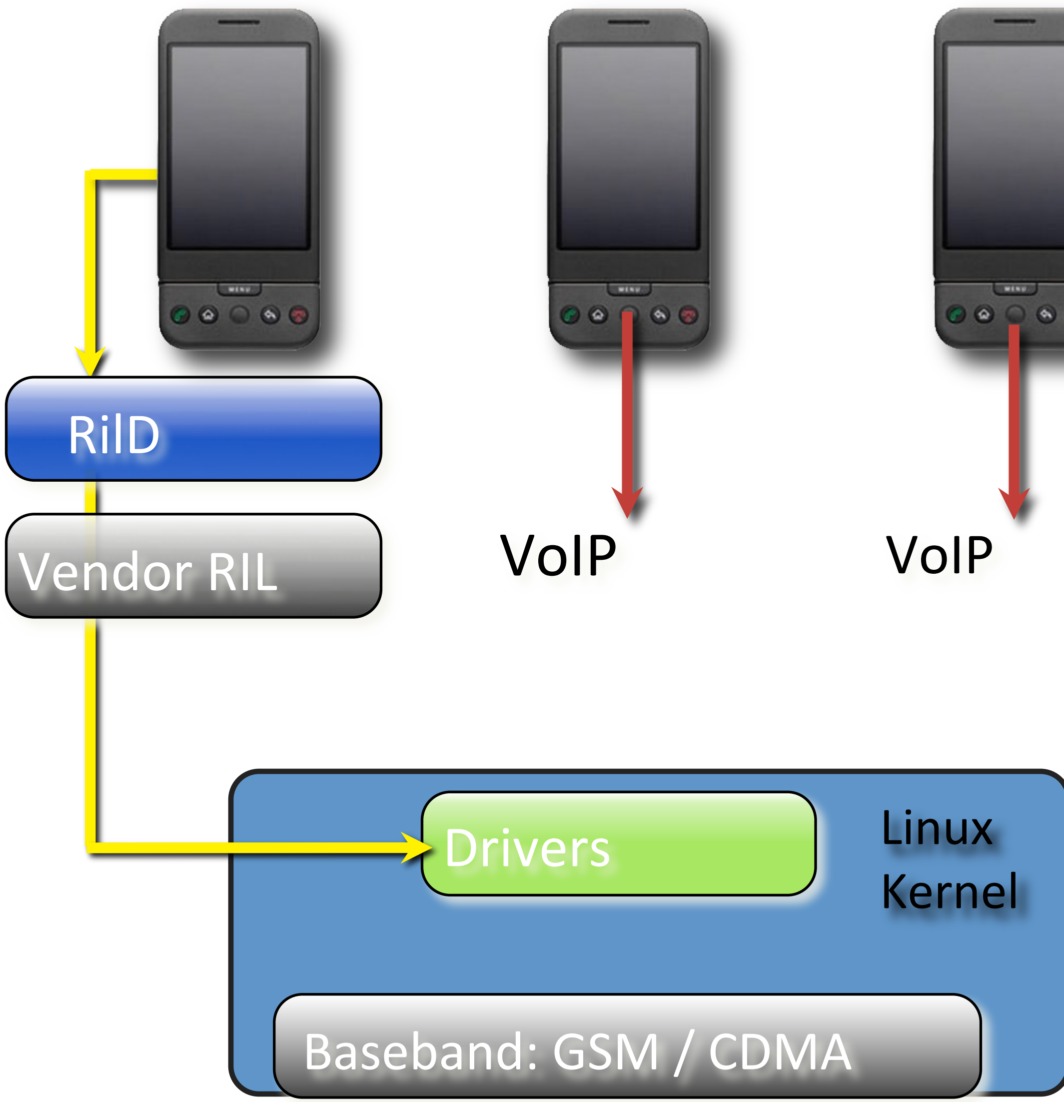
Courtesy: Jason Nieh et al.

Device Namespace + Graphics Context



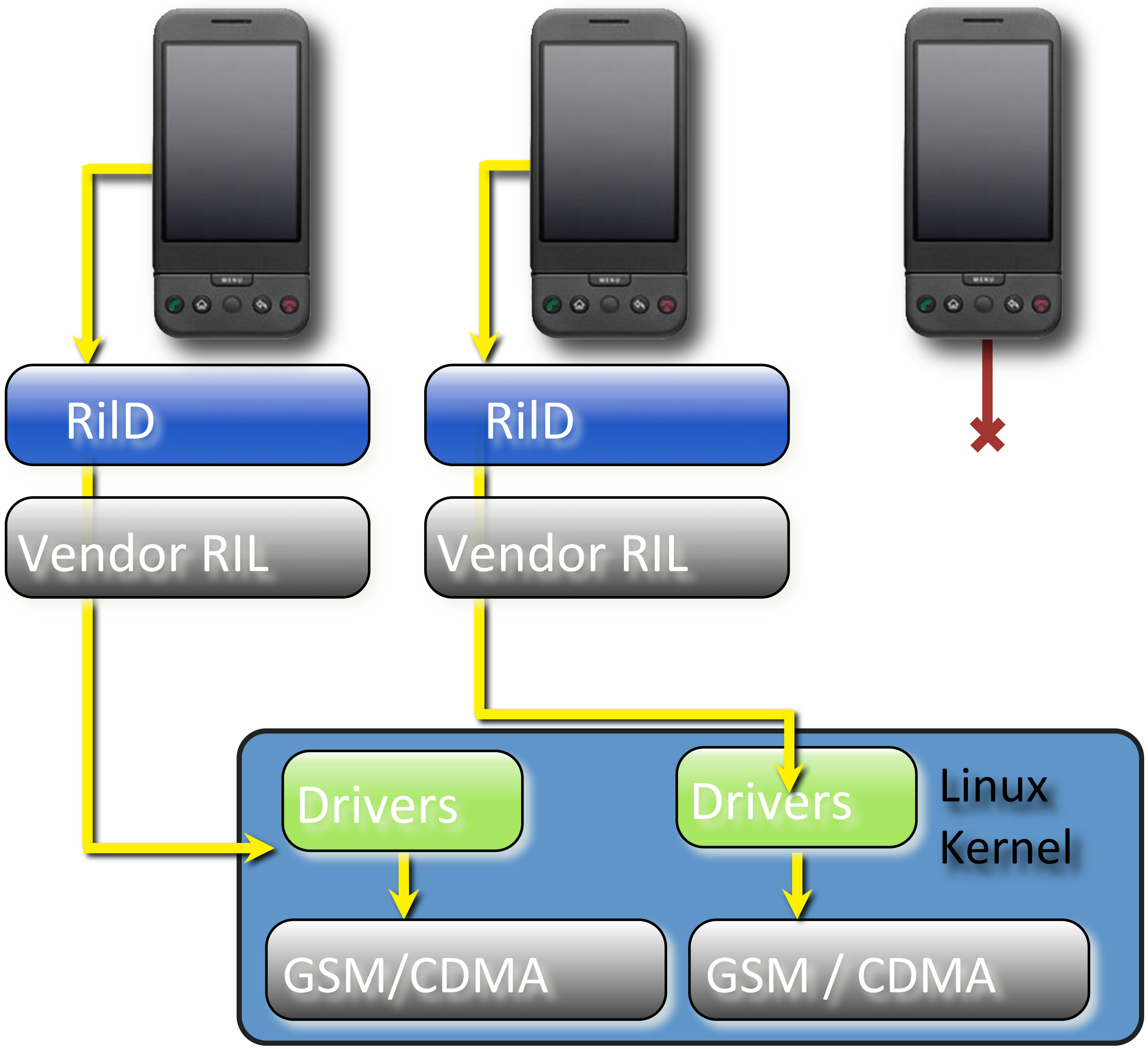
Courtesy: Jason Nieh et al.

VoIP?



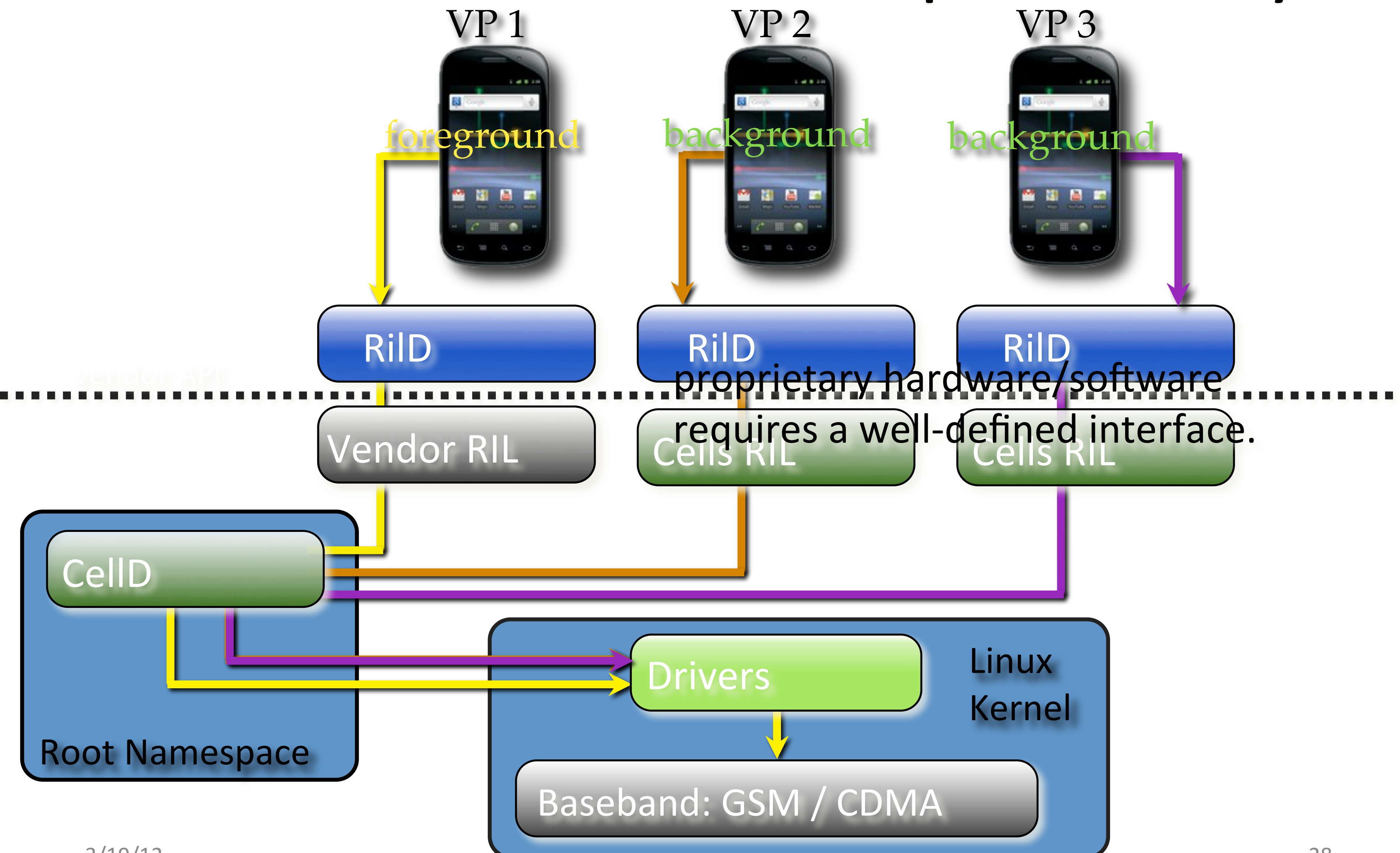
RilD
Vendor RIL
VoIP
VoIP
Drivers
Linux Kernel
Baseband: GSM / CDMA

Dual-SIM?



Courtesy: Jason Nieh et al.

Cells: User-Level Namespace Proxy



Courtesy: Jason Nieh et al.

Experimental Results

Setup

- Nexus S
- five virtual phones
- overhead vs. stock ⟨Android 2.3⟩

Courtesy: Jason Nieh et al.

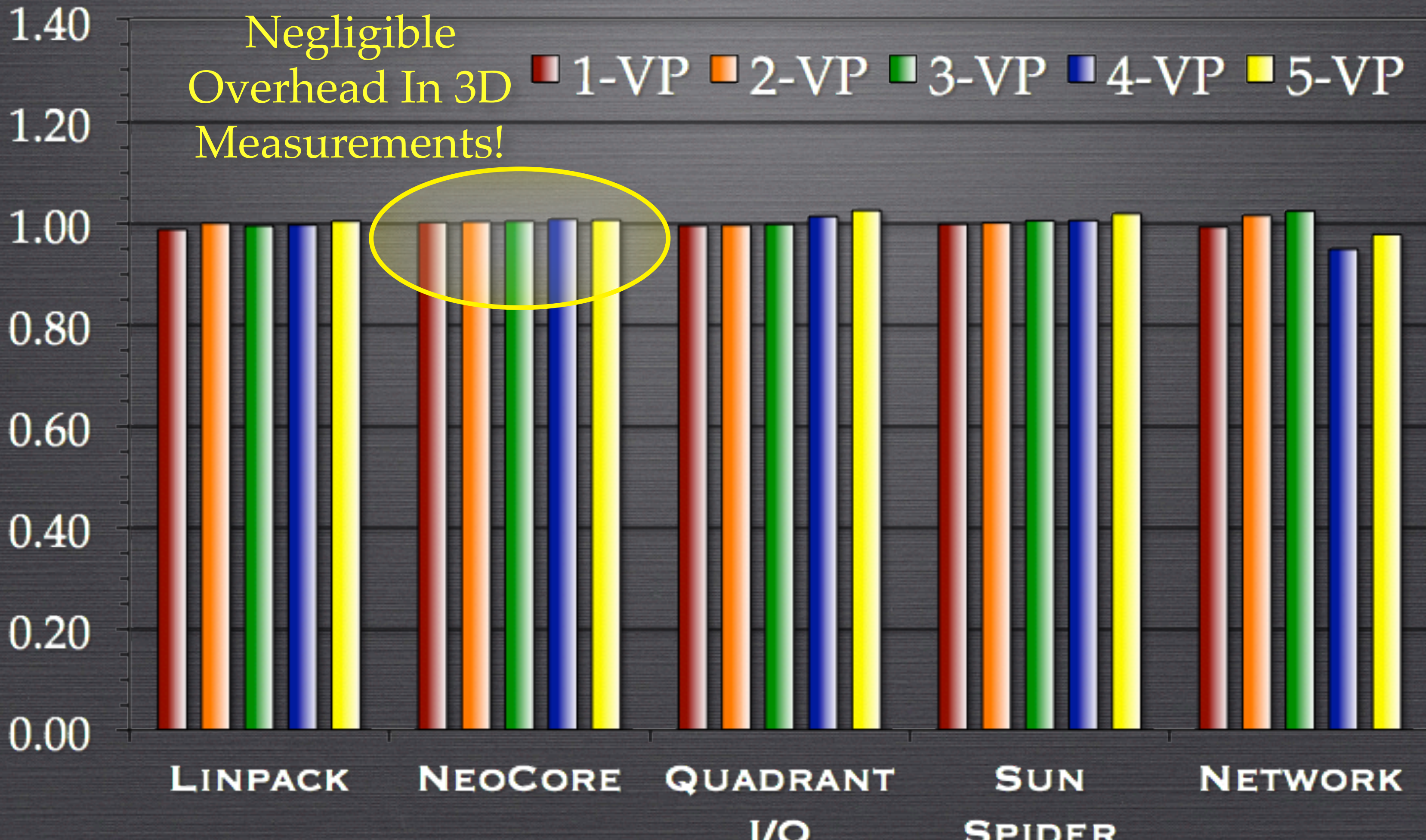
Experimental Results

Setup

- CPU ⟨Linpack⟩
- graphics ⟨Neocore⟩
- storage ⟨Quadrant⟩
- web browsing ⟨Sun Spider⟩
- networking ⟨Custom WiFi Test⟩

Courtesy: Jason Nieh et al.

Experimental Results
Runtime Overhead



Courtesy: Jason Nieh et al.

Cells

Complete, Efficient, Transparent Mobile Virtualization

- device namespaces
  - safely and efficiently share devices
- foreground / background
  - designed specifically for mobile devices
- implemented on Android
- less than 2% overhead on Nexus S

Courtesy: Jason Nieh et al.

More Info

cells.cs.columbia.edu

cellrox.com

Revisiting Storage for Smartphones

Hyojun Kim Nitin Agrawal Cristian Ungureanu

Georgia Tech

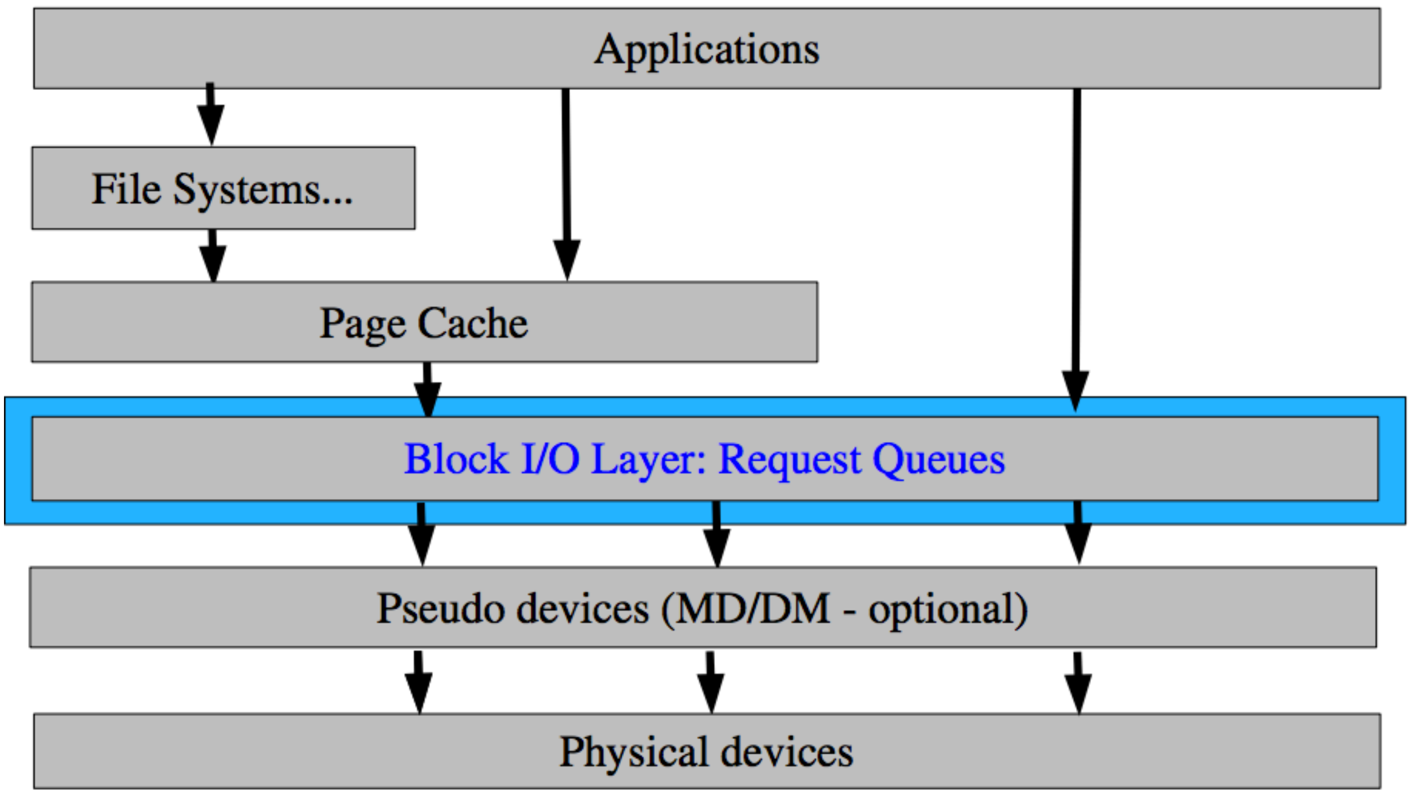
NEC Laboratories America, Inc.

Background

- blktrace: collect block level traces for device I/O
- monkeyrunner
  - installed at *android-sdk-macosx/tools/monkeyrunner*
  - functional testing framework for Interactive Android applications

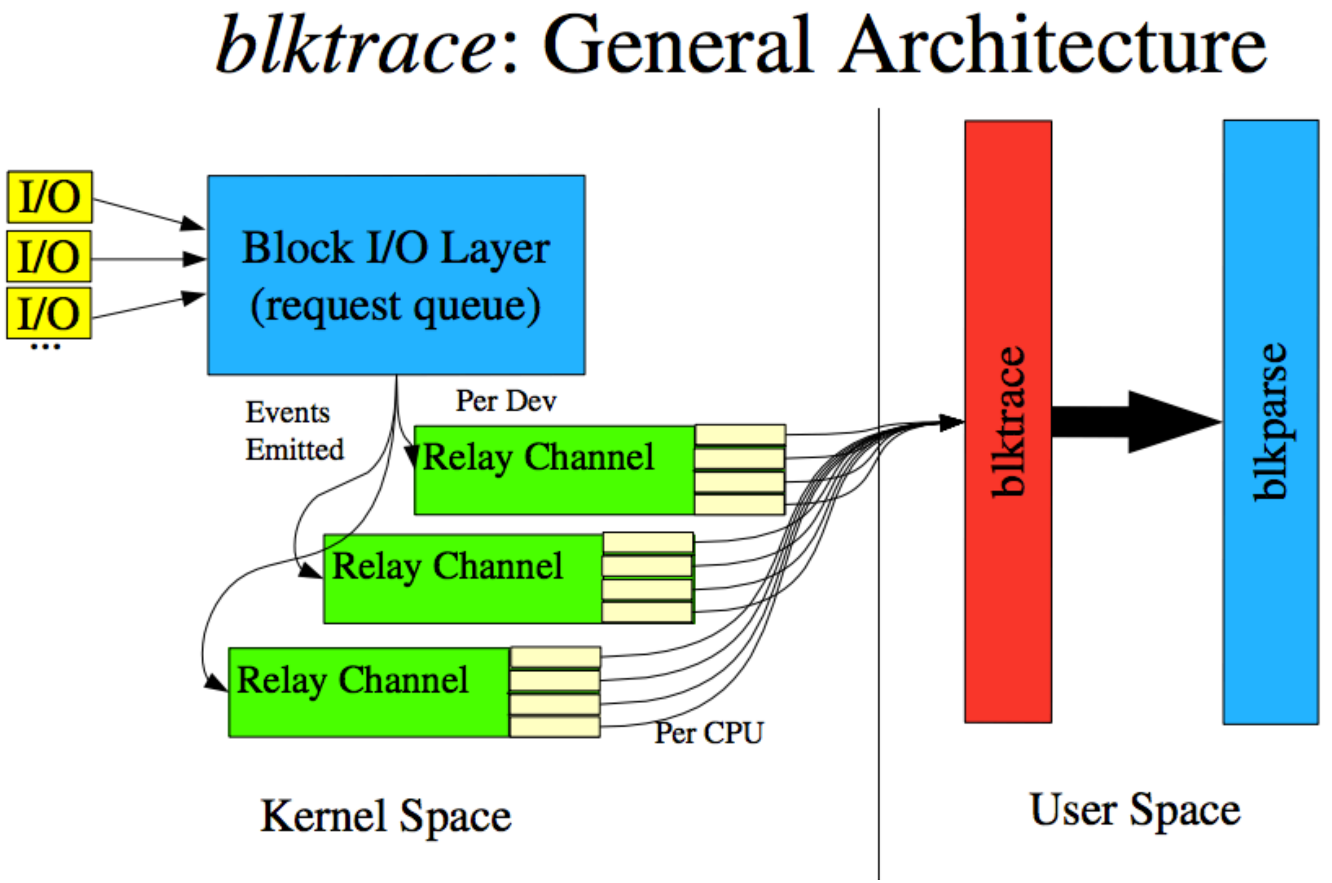
blktrace

- Block IO layer



Source: Alan D. Brunell
http://www.gelato.org/pdf/apr2006/gelato_ICE06apr_blktrace_brunelle_hp.pdf

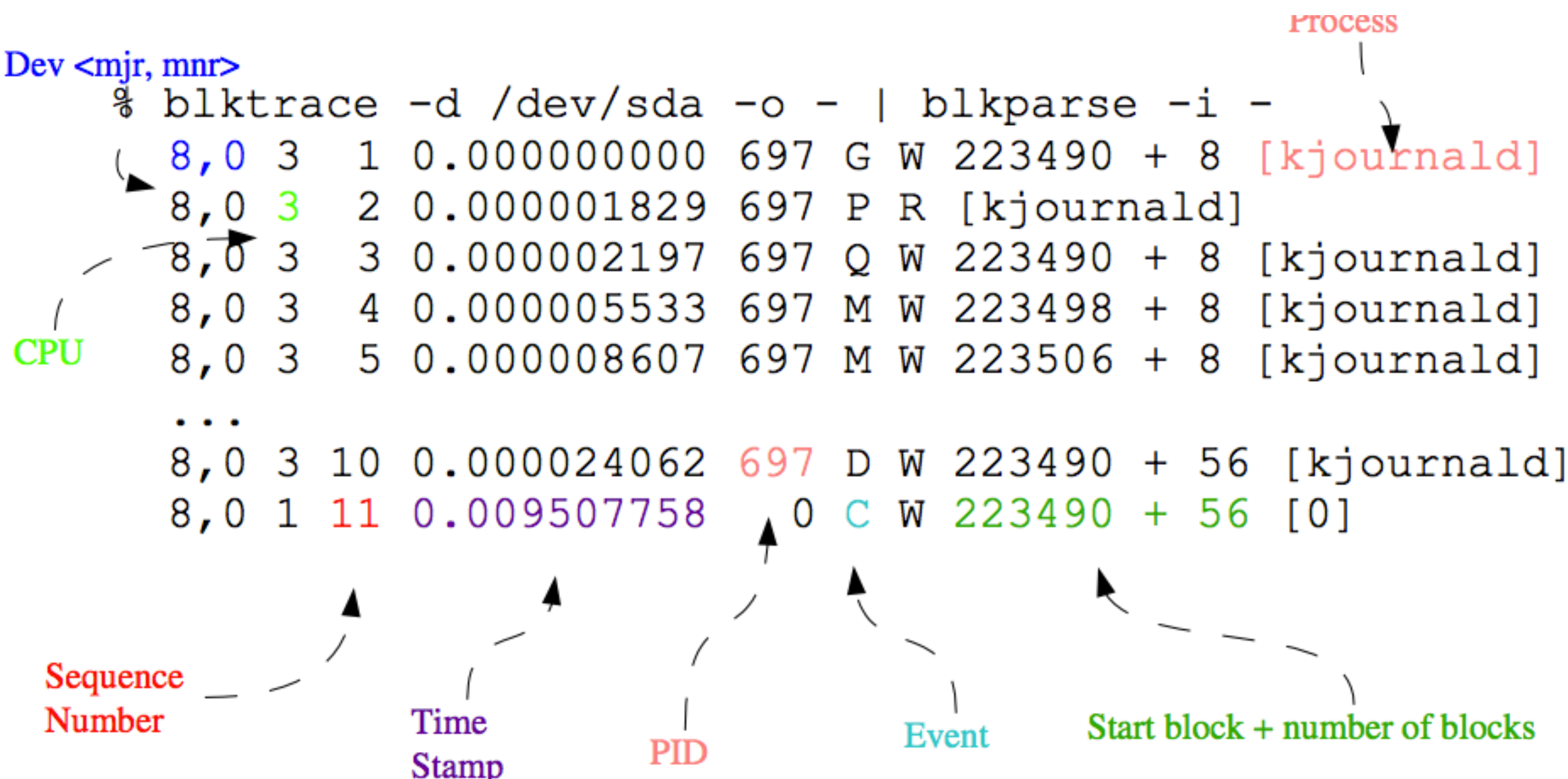
blktrace (Cont'd)



Source: Alan D. Brunell
http://www.gelato.org/pdf/apr2006/gelato_ICE06apr_blktrace_brunelle_hp.pdf

blktrace (Cont'd)

- blktrace sample traces



Source: Alan D. Brunell
http://www.gelato.org/pdf/apr2006/gelato_ICE06apr_blktrace_brunelle_hp.pdf

monkeyrunner

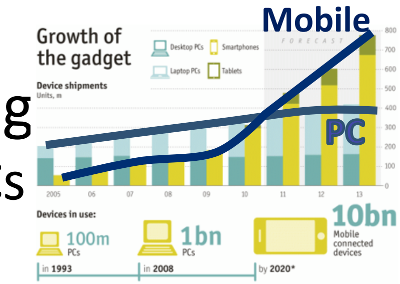
Example code

```
# Imports the monkeyrunner modules used by this program
from com.android.monkeyrunner import MonkeyRunner, MonkeyDevice

def main():
    # Connects to the current device, returning a MonkeyDevice object
    device = MonkeyRunner.waitForConnection()
    print 'waiting for connection...\n’

    package = 'coms6998.cs.columbia.edu’
    activity = 'coms6998.cs.columbia.edu.VoiceRegonitionDemoActivity'

    # sets the name of the component to start
    runComponent = package + '/' + activity

    # Runs the component
    device.startActivity(component=runComponent)

    # Presses the speaker button
    device.press(’DPAD_DOWN', MonkeyDevice.DOWN_AND_UP)
   device.press(’DPAD_CENTER', MonkeyDevice.DOWN_AND_UP)

    # Takes a screenshot
    screenshot = device.takeSnapshot()

    # Writes the screenshot to a file
    screenshot.writeToFile('./device1.png','png')
    reference = MonkeyRunner.loadImageFromFile(./device.png')
    if not screenshot.sameAs('./device.png', 0.9):
        print "comparison failed!\n"

if __name__ == '__main__':
     main()
```

monkeyrunner

- Demo

Life in the "Post-PC" Mobile Era



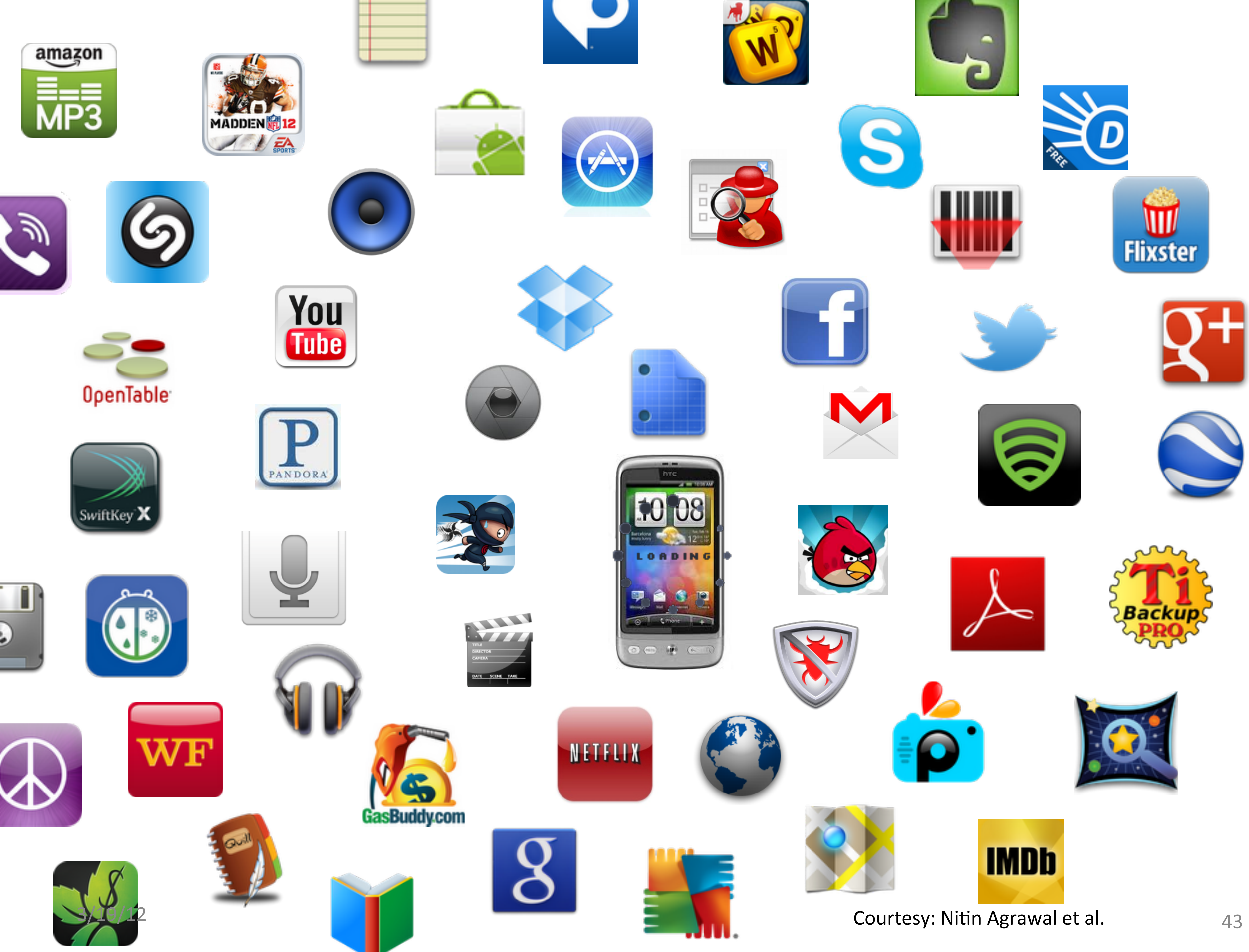
- Smartphone and tablet markets are huge & growing
  - 100 Million smartphones shipped in Q4 2010, 92 M PCs [IDC]
  - Out of 750 Million Facebook users, 250 Million (& growing) access through mobile; mobile users twice as active [FB]

- 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

Courtesy: Nitin Agrawal et al.

Courtesy: Nitin Agrawal et al.



Courtesy: Nitin Agrawal et al.

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?

Courtesy: Nitin Agrawal et al.

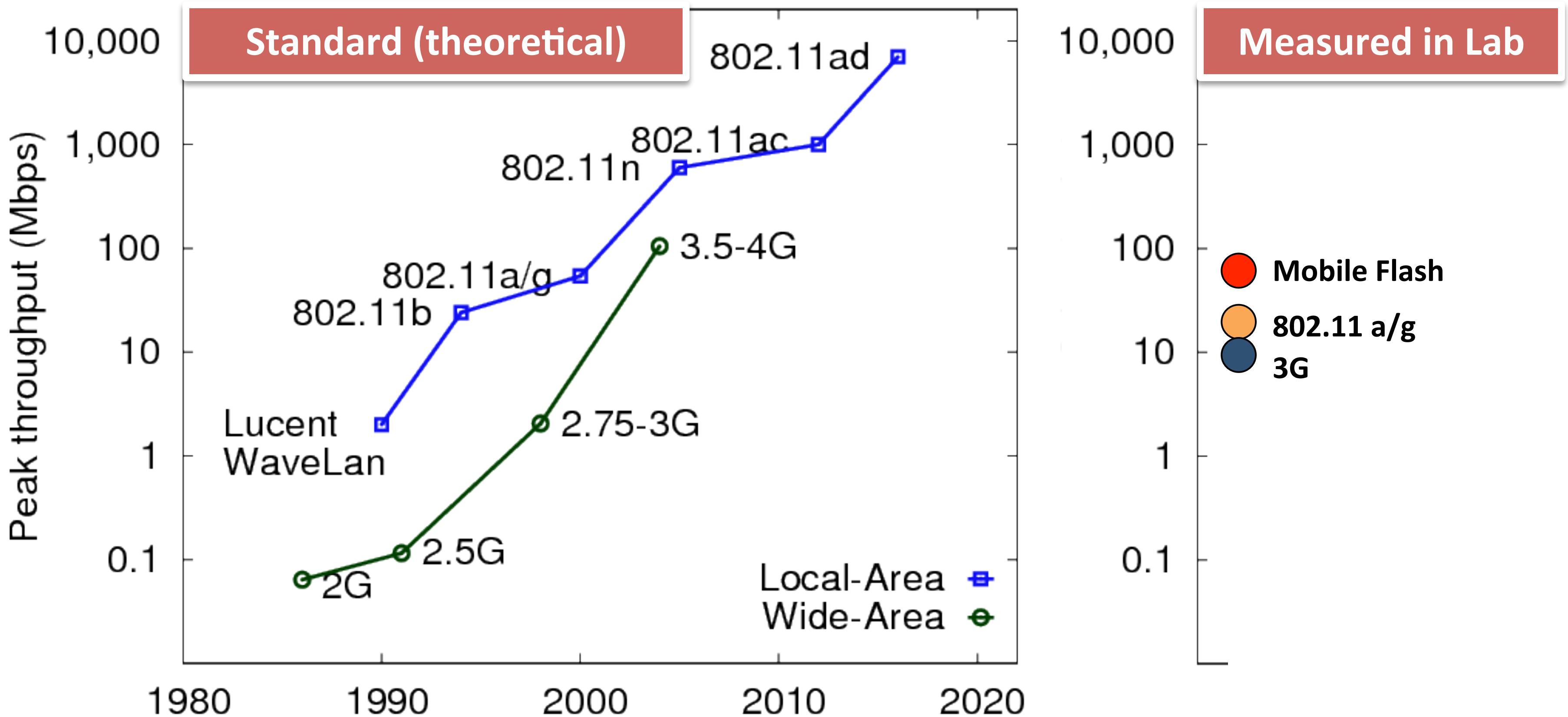
Understanding Mobile Performance

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
  - **Not well understood!** to appear on mobile devices
- Does storage performance impact mobile experience?
  - For storage, vendors & consumers mostly refer to capacity

Courtesy: Nitin Agrawal et al.

Wireless Network Throughput Progression



- 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]

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

Vendor (16GB)	Speed Class	Cost US $	Seq Write	Rand Write
Transcend	2	26	4.2	1.18
RiData	2	27	7.9	0.02
Sandisk	4	23	5.5	0.70
Kingston	4	25	4.9	0.01
Wintec	6	25	15.0	0.01
A-Data	6	30	10.8	0.01
Patriot	10	29	10.5	0.01
PNY	10	29	15.3	0.01

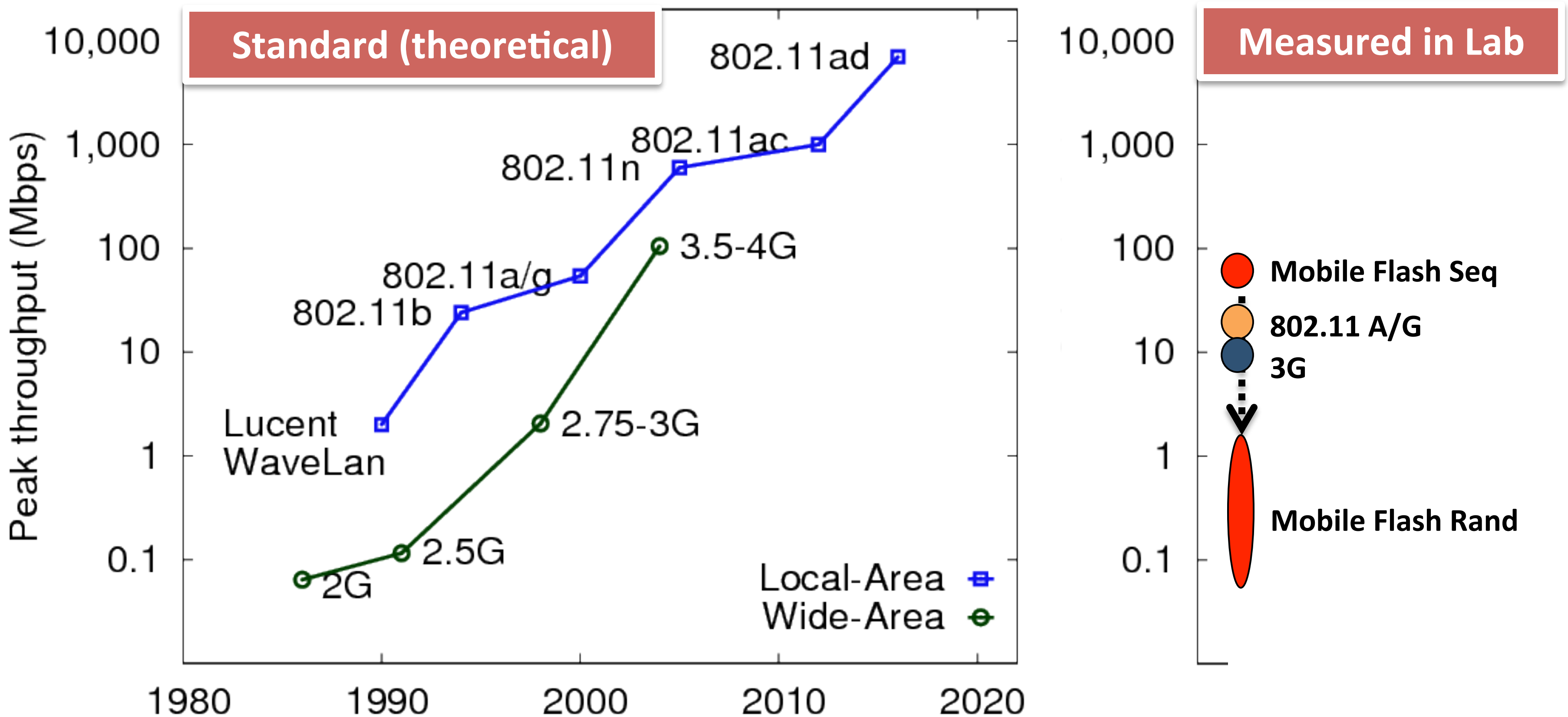
Performance MB/s

Consumer-grade SD performance

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

Courtesy: Nitin Agrawal et al.

Why Storage is a Problem
Shifting Performance Bottlenecks



- 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

Courtesy: Nitin Agrawal et al.

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

Courtesy: Nitin Agrawal et al.

Outline

✓ **Introduction**

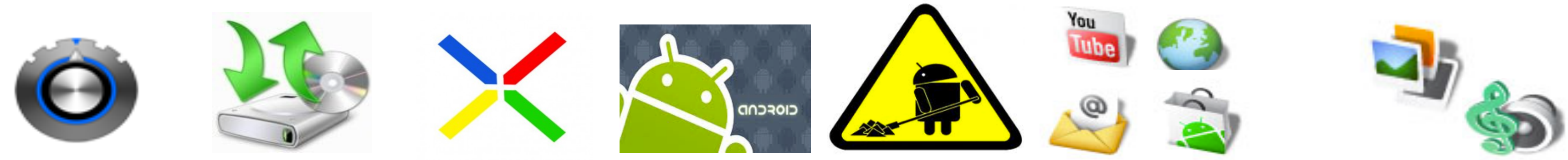
✓ **Why storage is a problem**

Android storage background and setup

Experimental results

Solutions

Storage Partitions on Android



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

Courtesy: Nitin Agrawal et al.

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

Courtesy: Nitin Agrawal et al.

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

Courtesy: Nitin Agrawal et al.

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

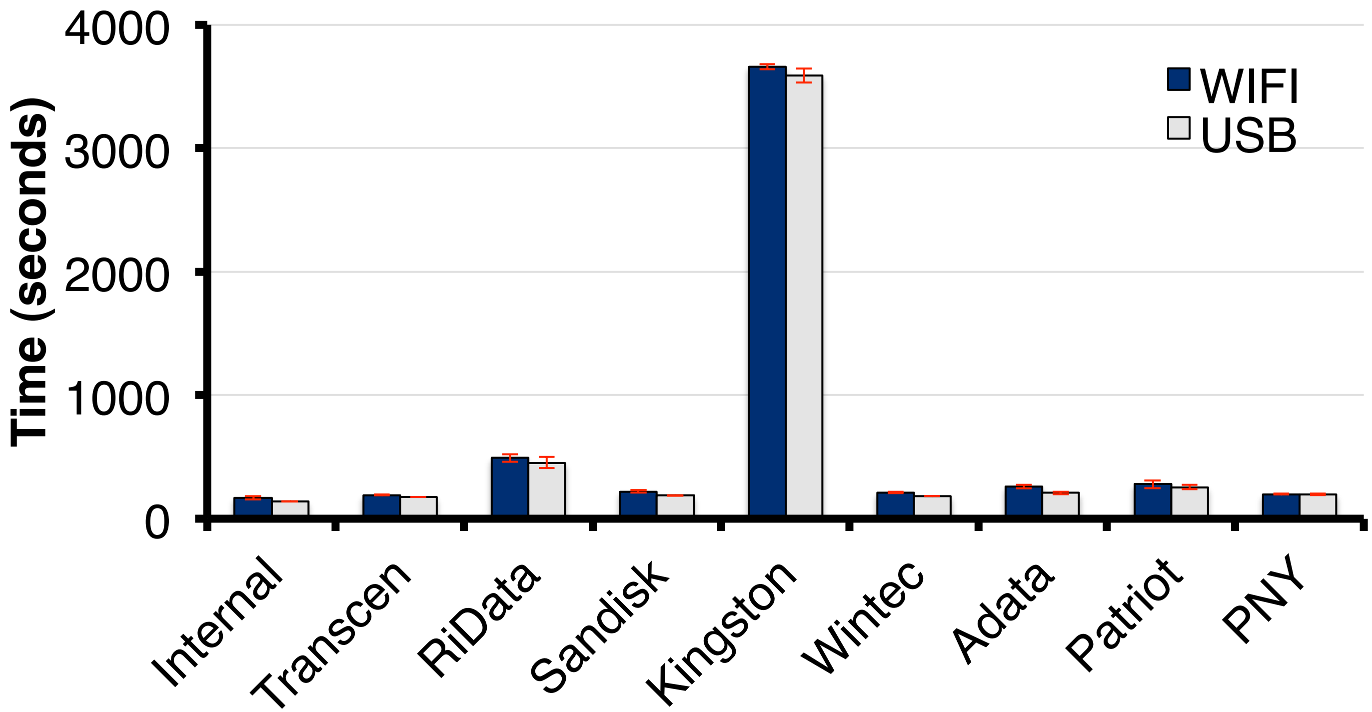
Background
Apps: Twitter, Books, Gmail
Contacts, Picasa, Calendar
Widgets: Pulse, YouTube,
News, Weather, Calendar,
Facebook, Market, Twitter

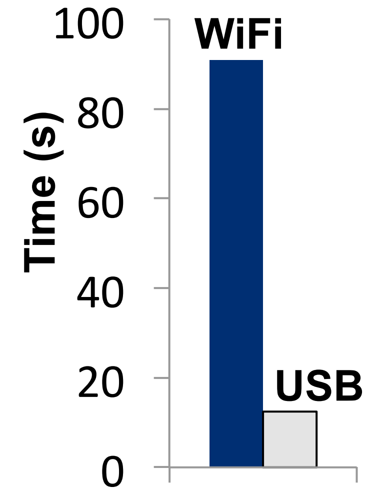
Courtesy: Nitin Agrawal et al.

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**

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%)

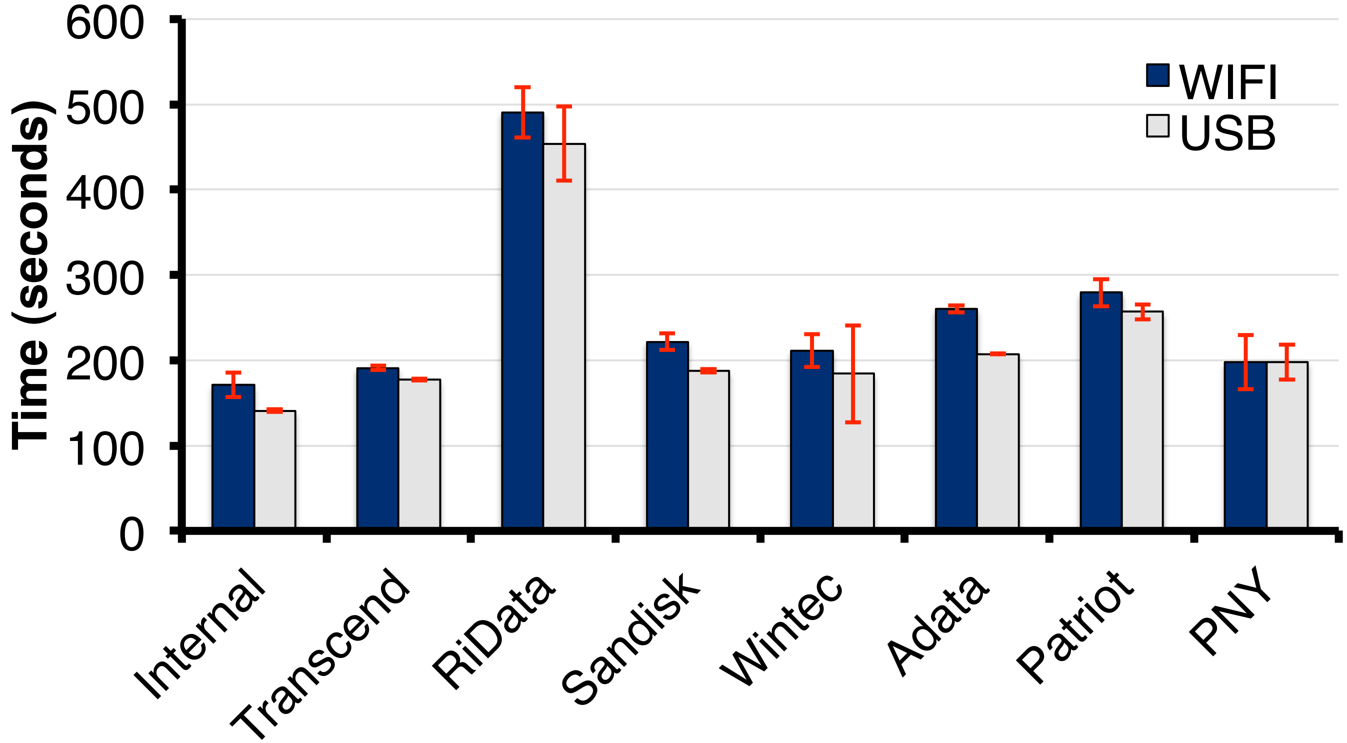
Storage significantly affects app performance and consequently user experience

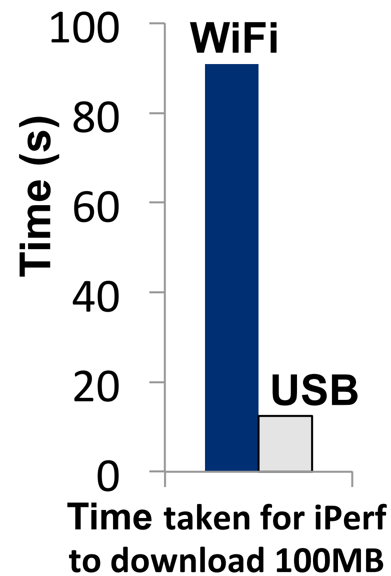
With a faster network (USB in RT), variance was 222% (without Kingston)

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

Courtesy: Nitin Agrawal et al.

WebBench Results: Runtime

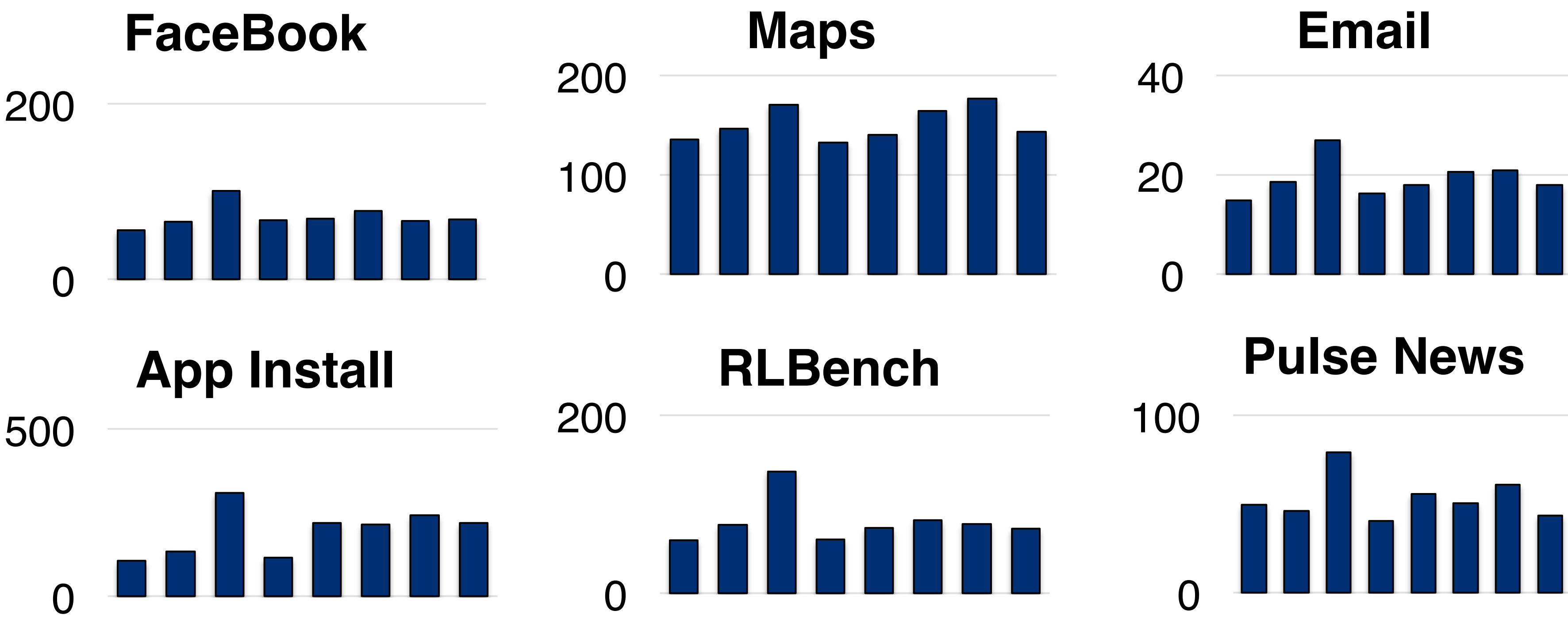




- Runtime on WiFi varies by 2000% between internal and Kingston
  - Even with repeated experiments, with new cards across speed classes
- Even without considering Kingston, significant performance variation (~200%)
- Storage significantly affects app performance and consequently user experience
- With 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)

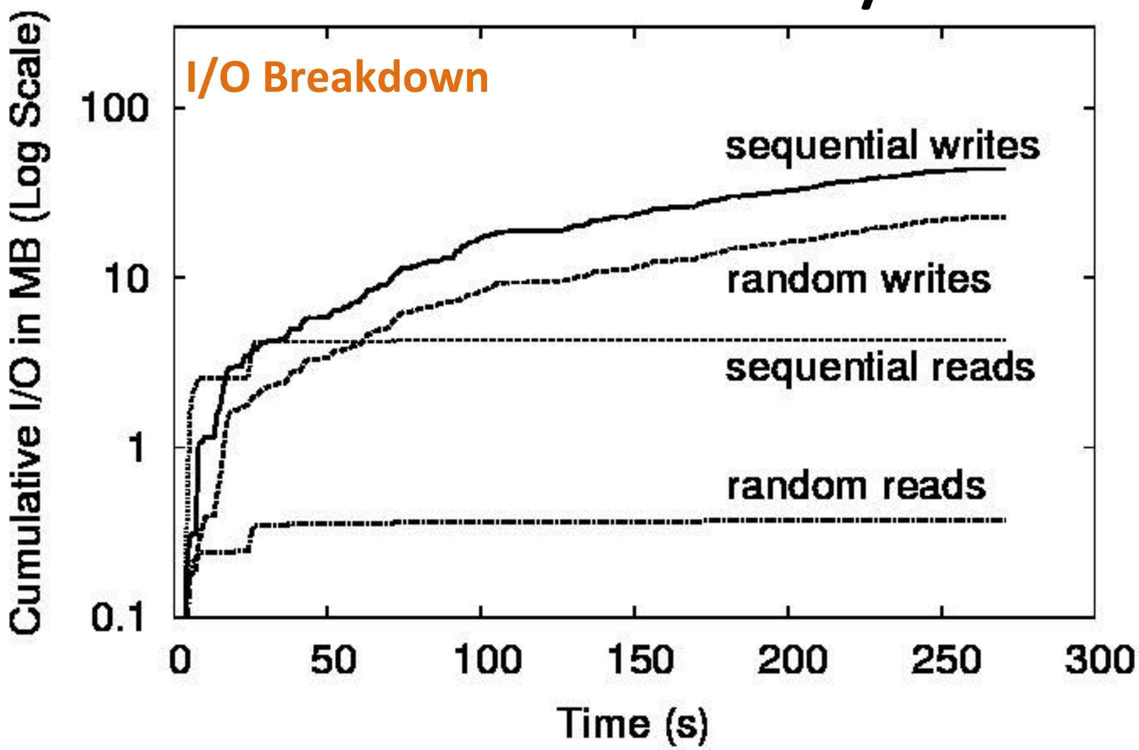


We find a similar trend for several popular apps

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



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

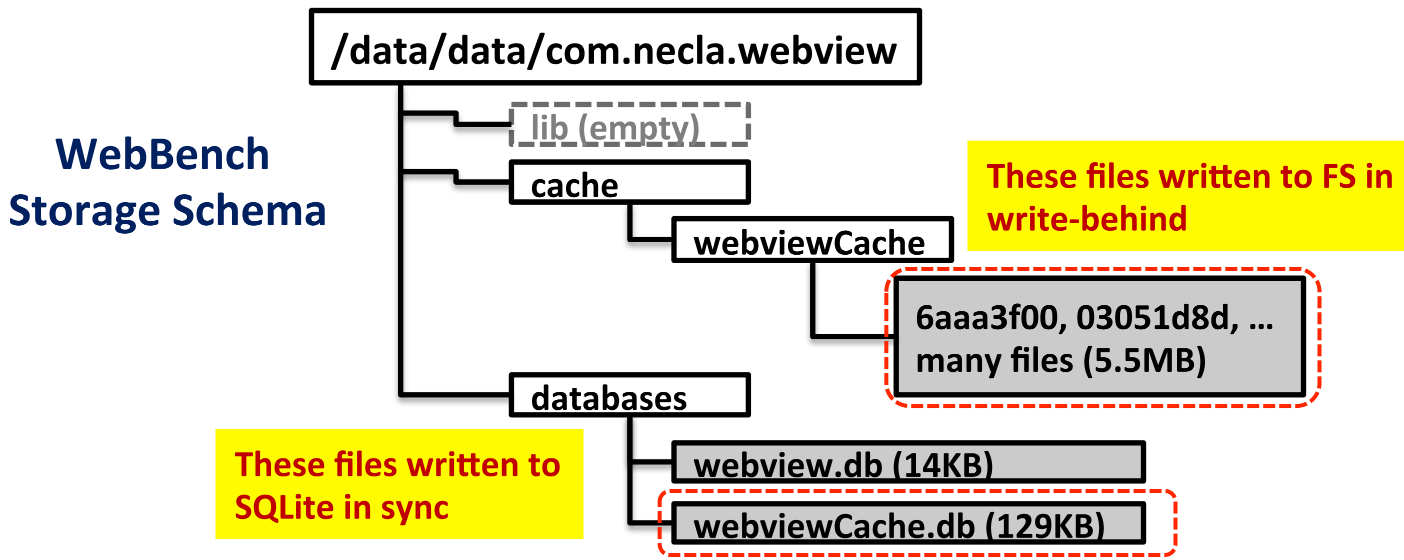
- Few reads, mostly at the start; significantly more writes
- About 2X more sequential writes than random writes
- Since rand is worse than seq by >> 2X, random dominates
- Apps write enough randomly to cause severe performance drop

Paper has a table on I/O activity for other apps

Courtesy: Nitin Agrawal et al.

How Apps Use Storage?

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



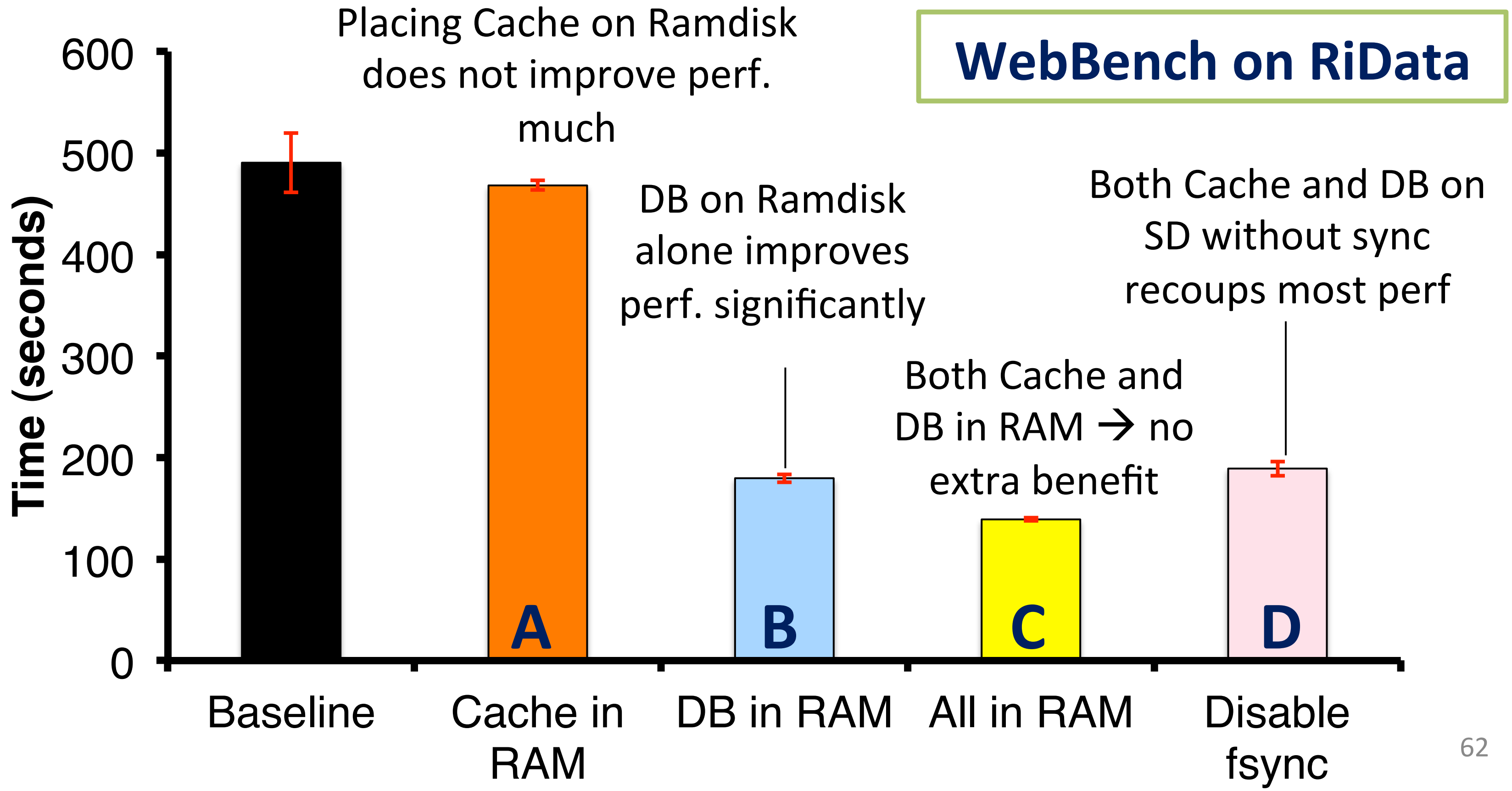
- Apps typically store some data in FS (e.g., cache files) and some in a SQLite database (e.g., cache map)
  - All data through SQLite is written synchronously → slow!
  - Apps often use SQLite oblivious to performance effects

Courtesy: Nitin Agrawal et al.

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

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



Courtesy: Nitin Agrawal et al.

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

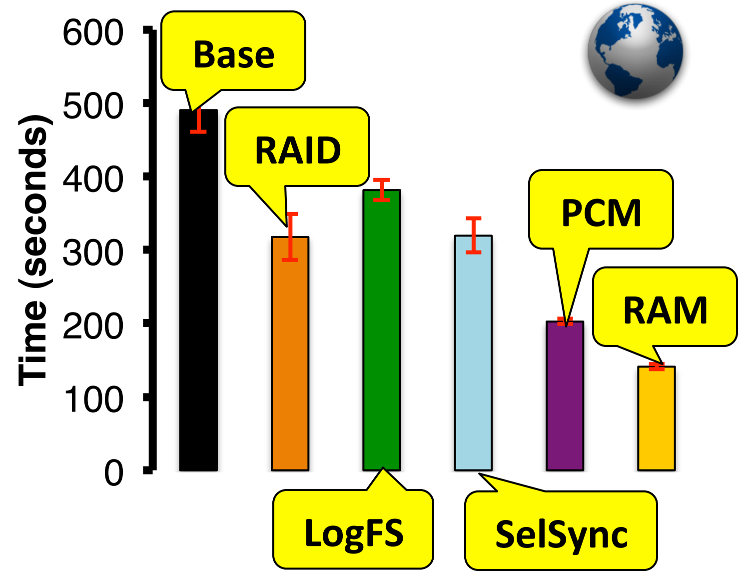
Courtesy: Nitin Agrawal et al.

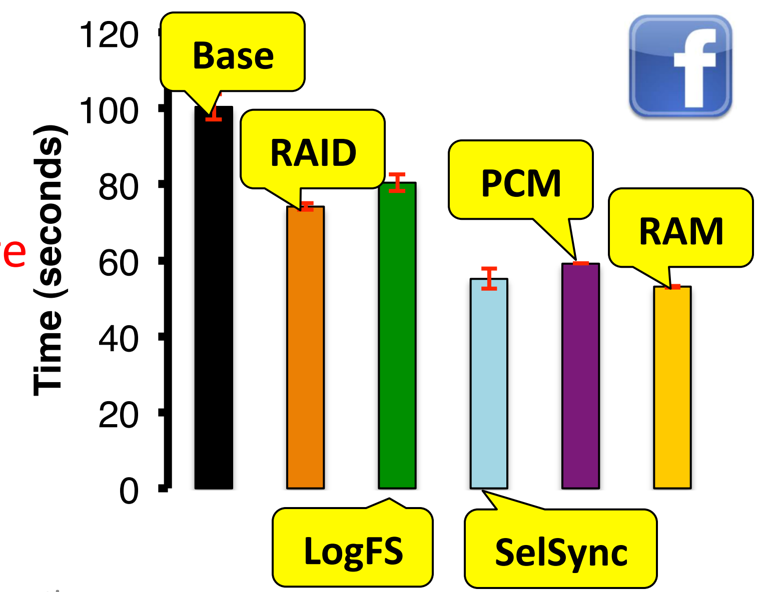
Outline

- ✓ **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





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

Courtesy: Nitin Agrawal et al.

Questions?