Cells: A Virtual Mobile Smartphone Architecture (SOSP 2011)

Lunch-Time Reading Group

11/11/2011

Minsung Jang
Summary

- Novel Architecture for a Virtual Phone
  - How to do away with “overhead”
  - New way to virtualize devices
- Foreground-Background Usage Model
  - How to contribute a low overhead
Motivation

• For “everyday computing,” Mobile > Desktop (+Laptop)
  – Many virtual phones that are isolated on a physical phone

• Existing (system) virtualizations
  – High overhead
  – Hardware devices

• Cells: new, LWT virtualization

Sensors (accessible by APIs)
• Tap sensor
• Orientation Sensor
• Accelerometer
• Ambient Light Sensor
• Proximity Sensor
• Compass
• Rotation Sensor
• 800MPixel camera on back
Overview of Cells

• Virtual Phones
  – Multiple Android instead of multiple OS instances
• Low overhead
  – UnionFS, KSM, Form factor (Small display)
• New usage model
• Device support
• Per-VP Phone number
• Prototype using Real commercial Android phones
  – Nexus 1, Nexus S with Gingerbread
• Scalability
Usage Model

• Display the contents only from the foreground VP
Access Right

- Three different access right for each device on a phone hardware
  - No access, shared access, or exclusive access
- Assigned Statically when a VP is configured
Architecture

User-level virtualization

Android/Java (Dalvik VM)

Single OS Instance

*RIL: Vendor Radio Interface Layer library is loaded by CellID
Architecture

• A virtual phone has
  – ID, Kernel I/F, and virtualized hardware resources
  – Private virtual namespace
  – Filesystem Path
  – Virtualized {PIDs, network names, user names}
    • E.g., The same PIDs can exist if VP ids are different
    • Linux provides the PID virtualization
Virtualization

• Why virtualized resources?
  – Guarantee Isolation among VPs
• How? Kernel-level, User-level

Root Namespace

State of VP 1

Virtual Phone (VP 1)

Proprietary Device Driver

State of VP 1

State of VP 2

Device Driver
## Devices

- **List of Devices that Android must have**

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm *</td>
<td>RTC-based alarms</td>
</tr>
<tr>
<td>Audio</td>
<td>Audio I/O (speakers, microphone)</td>
</tr>
<tr>
<td>Binder *</td>
<td>IPC framework</td>
</tr>
<tr>
<td>Bluetooth</td>
<td>Short range communication</td>
</tr>
<tr>
<td>Camera</td>
<td>Video and still-frame input</td>
</tr>
<tr>
<td>Framebuffer</td>
<td>Display output</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics processing unit</td>
</tr>
<tr>
<td>Input</td>
<td>Touchscreen and input buttons</td>
</tr>
<tr>
<td>LEDs</td>
<td>Backlight and indicator LEDs</td>
</tr>
<tr>
<td>Logger *</td>
<td>Lightweight RAM log driver</td>
</tr>
<tr>
<td>LMK *</td>
<td>Low memory killer</td>
</tr>
<tr>
<td>Network</td>
<td>Wi-Fi and Cellular data</td>
</tr>
<tr>
<td>Pmem *</td>
<td>Contiguous physical memory</td>
</tr>
<tr>
<td>Power *</td>
<td>Power management framework</td>
</tr>
<tr>
<td>Radio</td>
<td>Cellular phone (GSM, CDMA)</td>
</tr>
<tr>
<td>Sensors</td>
<td>Accelerometer, GPS</td>
</tr>
</tbody>
</table>

**Table 1: Android devices**

*custom Google drivers*
Scalability & Security

• Scalability
  – How many VPs can be built on a physical phone
  – Removing overhead per-VM
    • UnionFS (aufs), KSM, LMK

• Security
  – How Cells isolate a VP from others
  – Kernel/user-level namespace
  – Separate FS view for each VP
  – No creation of device nodes in a VP
UnionFS (aufs)

- Stacking file systems
- Join the read-only fs with a writeable files
- Read-only: /data, /system
• Device Virtualization
Graphics

- Multiplexing FB device driver: `mux_fb`
- How FB works: `mmap`, standard IOCTLs, custom IOCTLs
Graphics in Android (Linux)

kernel

FB driver

Device File

device type
Major #
Minor #

control

Application

Buffer

process address space

kernel address space

mmap()

dma_alloc_writecombine()

physical memory

DMAC

LCD device
Graphics

Virtual Phone 1

- process address space's Buffer

Virtual Phone 2

- process address space's Buffer

Device Namespace Management

- Frame buffer for Foreground VP
- Frame buffer for Background VP

- Backing memory

LCD Device
GPU

• GPU is already isolated by underlying Linux
  – Pass-through access in each VP context
• Switching the foreground VP, remap memory buffers
• Need to access a GPU driver at a somewhat level
• Requirements for proprietary GPU driver
  – Remap the GPU driver’s linear addresses
  – Re-initialize On/Off
  – Ignore power management request
• For GPUs without MMU, the backing memory is contiguous
Telephony

• Telephony in Android

Diagram of Telephony in Android:
- Android Java
  - Java Phone / RIL services
  - Libraries
    - RIID
    - Vendor RIL
  - Kernel
    - Drivers / PPP
- Baseband
  - GSM / CDMA
- Closed source codes
- Solicited
  - Call application (Dialer)
  - Radio Interface layer Daemon (RIID) from Google
  - Vendor RIL (proprietary)
- Unsolicited
  - Kernel Driver (proprietary)
Telephony

Virtual Phone

Android Java
- Java Phone / RIL services

VP Libraries
- RiID
- Cells RIL

Root namespace
- CellID
- Vendor RIL

Kernel
- Drivers / PPP

Baseband
- GSM / CDMA

Cells
Radio Interface Layer
Telephony

• Incoming/Outgoing calls
  – Multiple Numbers: Phone number assigned to each VP
  – Challenge: One SIM allows just one phone number, Caller ID

<table>
<thead>
<tr>
<th>Call</th>
<th>Class</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dial Request</td>
<td>Solicited</td>
<td>Foreground</td>
</tr>
<tr>
<td>Set Screen State</td>
<td>Solicited</td>
<td></td>
</tr>
<tr>
<td>Set Radio State</td>
<td>Solicited</td>
<td></td>
</tr>
<tr>
<td>SIM I/O</td>
<td>Solicited</td>
<td>Initialization</td>
</tr>
<tr>
<td>Signal Strength</td>
<td>Unsolicited</td>
<td>Radio Info</td>
</tr>
<tr>
<td>Call State Changed</td>
<td>Unsolicited</td>
<td></td>
</tr>
<tr>
<td>Call Ring</td>
<td>Unsolicited</td>
<td></td>
</tr>
<tr>
<td>Get Current Calls</td>
<td>Solicited</td>
<td>Phone Calls</td>
</tr>
</tbody>
</table>

Table 2: Filtered RIL commands
Telephony/Outgoing Calls

VP1 (VoIP Number:1)  VP2 (VoIP number:2)

Original phone number: 404-433-1112
Telephony/Incoming Calls

VP1 (VoIP Number: 1)  VP2 (VoIP number: 2)

Table:

<table>
<thead>
<tr>
<th>VoIP Number</th>
<th>Original number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>404-433</td>
</tr>
<tr>
<td>2</td>
<td>404-433</td>
</tr>
</tbody>
</table>

Original phone number: 404-433-1112
Network

- Best Practice in desktop/server virtualization
  - Virtual wired network adapter for each virtual machine
  - Configuration in a host
Network

• Requirements
  – Each VP should be able to independently select its wireless connection, and configure it
    • e.g., Security method, WLAN or 3G
  – Some application should be able to access WLAN directly
    • e.g., Location-based services, AppStore, System Updater

![Network Configuration Example]

Each VP should be able to see such selection.
Network

• Similar with telephony
• Android uses `wpa_supplicant` for wireless setup
Experiment

• Measuring overhead/power consumption for 5-VP
• VPs configuration for UI test
  – VP1: Angry Bird
  – VP2: Reckless Racing (3D game)
  – VP3: Office Suite Pro
  – VP4: Android Music Player
Experiment

(a) Normalized Nexus 1 results
(b) Normalized Nexus S results
(c) Normalized Nexus 1 + music results
(d) Normalized Nexus S + music results
(e) Normalized battery capacity
(f) Memory usage in MB
Conclusion

• Problems in existing device virtualization
  – Pass-through: other VMs cannot access
  – Bypass: requires special hardware support
  – In particular, GPU
    • VMware MVP not good for games
    • Xen backend-frontend: vendor support needed
Conclusion

• Pros
  – Lightweight, No hypervisor
  – Leveraging existing device drivers

• Cons
  – Large TCB
  – Different mobile OS on a phone (e.g., Android + iOS)