VIRTUALIZATION
Server Virtualization
Bare-Metal Hypervisor

Poor device support / sharing

Hypervisor / VMM

Hardware
Desktop Virtualization
Hosted Hypervisor

- poor device performance
- host user space

Hypervisor / VMM

Host OS Kernel

kernel module
emulated devices

Hardware
Non-Virtualization
User Space SDK

- no standard apps
- less secure

- custom user space API for isolated apps

OS Kernel

Hardware
**Key Challenges**

- **device diversity**
  - Power
  - Cell Radio
  - h.264 accel.
  - camera(s)
  - microphone
  - Buttons
  - pmem
  - Accelerometer
  - speakers
  - Binder IPC
  - RTC / Alarms

- **mobile usage model**
  - graphics-accelerated UI
Cells
Key Observation

large: lots of windows/apps

small: one app at a time
**Cells**

**Key Observation**

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

**Usage Model**

foreground / background
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!
Cells: Efficient Virtualization

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

• each VP sees / uses all devices
• user can run any unmodified apps
• foreground VP switches like an app
Single Kernel: Multiple VPs

isolated collection of processes

virtualize at OS interface
Single Kernel: Device Support

all VPs access the same device simultaneously

- Power
- Touchscreen
- WiFi
- pmem
- camera(s)
- speakers
- microphone
- Buttons
- GPU
- Binder IPC
- Accelerometer
- RTC / Alarms
- headset
- GPS
- Framebuffer
- Compass
- hw codec
- pmem
- camera(s)
- speakers
- microphone
- Buttons
- GPU
- Binder IPC
- Accelerometer
- RTC / Alarms
- headset
- GPS
- Framebuffer
- Compass
- hw codec
Device Namespaces

safely, correctly multiplex access to devices

Linux Kernel

device namespaces

VP 1
VP 2
VP 3

WiFi
Cell Radio
Framebuffer
GPU
Power
Input
Sensors
Audio/Video
RTC / Alarms
Android...

Motivation
Overview
Architecture
Graphics
Cell Radio
Evaluation
Demo
**Cells**

**device namespaces**

**foreground / background**

**Complete, Efficient, Transparent Mobile Virtualization**

- WiFi
- Cell Radio
- Framebuffer
- GPU
- Power
- Input
- Sensors
- Audio/Video
- RTC / Alarms
- Android...

Linux Kernel
efficiency graphic virtualization

hardware accelerated graphics

proprietary / closed interface
**Approach 1: Single Assignment**

- **Screen Memory**
- **Framebuffer**
- **GPU**

Virtual addresses

Physical addresses
**Approach 2: Emulated Hardware**

- **Screen Memory**
- **Emulated Framebuffer**
- **Virtual State**
mux_fb presents identical device interface to all VPs using device namespaces.

Swap virt addr mappings: point to different phys addr.

Cells: Device Namespaces

Mux_fb presents identical device interface to all VPs using device namespaces.

Swap virt addr mappings: point to different phys addr.

Screen memory

Framebuffer

Mux_fb

Virtual addresses
Physical addresses
Accelerated Graphics

VP: just a set of processes!

OpenGL context

Framebuffer

GPU

MMU

Process isolation

graphics virtual addresses
physical addresses
Device Namespace + Graphics Context

Overview

Motivation

Architecture

Graphics

Cell Radio

Evaluation

Demo

Device Namespace + Graphics Context

OpenGL context

OpenGL context

OpenGL context

foreground

background

background

screen memory

MMU

GPU

graphics virtual addresses

physical addresses

24 Slides © 2013 Jeremy A. Andrus <jeremya@cs.columbia.edu>
**VoIP?**

- **Vendor RIL** → **RilD**
- **VoIP**
- **VoIP**
- **Drivers** → **Baseband: GSM / CDMA** → **Linux Kernel**
**Cells: User-Level Namespace Proxy**

- **Vendor API**
  - **Foreground**
  - **Background**
  - **Proprietary hardware/software requires a well-defined interface.**

- **CellD**
  - **Root Namespace**
  - **Drivers**
  - **Linux Kernel**
  - **Baseband: GSM / CDMA**
**Experimental Results**

**Setup**

- **Nexus S**
- **five virtual phones**
- **overhead vs. stock** (*Android 2.3*)
**Experimental Results**

**Setup**

- CPU (*Linpack*)
- graphics (*Neocore*)
- storage (*Quadrant*)
- web browsing (*Sun Spider*)
- networking (*Custom WiFi Test*)
**EXPERIMENTAL RESULTS**

**Runtime Overhead**

Negligible Overhead In 3D Measurements!
Demo
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
More Info

Software Systems Laboratory

cells.cs.columbia.edu

cellrox.com