MultiDroid: A Novel Solution to Consolidate Interactive Physical Android Clients on One Single Computing Platform

Bin Yang, Intel R&D Center
Shoumeng, Yan, Intel Labs
Agenda

• Background and Scenarios
• Solution
• Evaluation
• Summary and Next
Background

• Leverage computing power of modern CPU to consolidate workloads
• Requirements come from different scenarios such as Retail, Automotive and Game cloud.
Scenario: Rear Seat Entertainment

- HVAC
- Video
- Music
- Game
- Map
- Conference
Scenario: Cloud Gaming
Scenario: Robot
Scenario: Auto Vending Machine
Solution - *MultiDroid*

- Solutions Comparing
- Architecture
- Advantages
- Implementation
Solutions Comparing

Containers are isolated, but share OS and, where appropriate, bins/libraries

Modify framework

Android framework
Advantages

- Lightweight
- Supports different Linux-based OS
- Little overhead.
- Simple I/O sharing solution.
- Recovery mechanism
- Security Guarantee
- Easy Deployment & Maintenance
Implementation: MultiDroid

- File System Layout
- Kernel
- I/O Devices Virtualization
- AOSP
File System Layout

- Different data partitions for different Android instances.
- `system.img` is shared between different Android instances.
- Overlay fs can be used if different `system.img` is required.
Kernel - Binder

Instance 1
Full Android user space stack
/dev/binder
/dev/binder1

Instance 2
Full Android user space stack
/dev/binder
/dev/binder2

Instance N
Full Android user space stack
/dev/binder
/dev/binder(n)

Kernel
I/O Devices Virtualization

- Dedicated I/O devices
  - e.g. USB Host
- Shared I/O devices
  - e.g. GFX, Trusty
I/O - USB Touch

Instance 1
Full Android user space stack
/dev/input
/dev/input0

Instance 2
Full Android user space stack
/dev/input
/dev/input1

Instance N
Full Android user space stack
/dev/input
/dev/input(n-1)

Kernel
I/O - Trusty

Security Storage Daemon

Android Instance 1
- KeyMaster
- GateKeeper

Android Instance N

Trusty OS
- Security Storage APP1
- Security Storage APP2

RPMB Driver

Storage Driver

Trusty IPC Driver

Kernel

RPCB

AuthKey
AOSP

- Low memory Killer
- Recovery
Low Memory Killer

Android Instance 1
- Process A
- Process B
- LMKD

AMS

Update adj with weight

Send kill event

Select Module

Memory monitor

Android Instance N
- Process A
- Process B
- LMKD

AMS

Update adj with weight

Kemel

Select to-be-killed process according to adj

Process A in Ins1
- Process B in Ins1
- Process A in Ins2
- Process B in Ins2
Recovery

Instance Manager watchDog

Monitor init process

Child process will send signal to init before it is exit
Child process will restart process

Init process

Child process will send signal to init before it is exit
Child process will restart process

Init process

Instance 1

Instance N
Evaluation

• Environment setup
• Evaluation result
Evaluation Setup

- Exclusive test: single active instance while multiple instances are created but in idle state
- Concurrent test: system performance where multiple instances run with continuous workload simultaneously
- Memory consumption

Hardware: Intel mini Kaby Lake PC Intel(R) Core(TM) i5-7260U CPU @ 2.20GHz *2 /HDMI display and touch
Software: Android 7.1.2
Result

Exclusive: Antutu

Concurrent: Game FPS

Cold Boot

Memory Consumed(GB)
Summary

• Raised the idea to consolidate workloads using container technology.
• Designed simple I/O sharing solution.
• Implemented POC to consolidate 16 Android Instances in KabyLake Nuc.
Next

- Enhance security.
- Optimize boot time and memory usage.
- Integrate different Linux based OS in one platform.
- Explore the possibility to integrate with VM solution.
LINUXCON
containercon
CLOUDOPEN

CHINA 中国

THINK OPEN
开放性思维