Android Optimization: Myth and Reality

Jim Huang (黃敬群) <**jserv**@0xlab.org>

Developer, 0xlab

http://0xlab.org/

July 8, 2011

Rights to copy

http://0xlab.org/

contact@0xlab.org

creative commons
COMMONS DEED

Attribution – ShareAlike 3.0

Corrections, suggestions, contributions and translations are welcome!

Latest update:July 10, 2011

You are free

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

Under the following conditions

BY: **Attribution**. You must give the original author credit.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.

License text: http://creativecommons.org/licenses/by-sa/3.0/legalcode

Agenda

(1) Android Meets Hardware Revolution

(2) Unproven "Optimization"

(3) Android Evolution

(4) Mythbusters

Revolution

(1) an attempt, by a large number of people, to change the government of a country, especially by violent action (革命)
(2) a great change in conditions, ways of working, beliefs, etc. that affects large numbers of people (巨變)

Evolution

(1) the gradual development of plants, animals, etc. over many years, from simple to more complicated forms (**進化**)
(2) the gradual development of something (演變；發展；漸進)

Quote from Google Translation

My interpret of Android:

Hardware is Revolution;

Sotware is basically Evolution;

Android is Hardware-driven Software Revolution

Android Meets Hardware Revolution

Take ARM for Example

- 32-bit instructions, with extension to support 16-bit Thumb & Thumb-2 instructions.
- Single unified memory address space (i.e. all peripherals and I/O are accessed like normal memory, at certain specific memory locations).
- Relatively low power consumption.
- Run at wide range of clock cycle
  - (June 1, 2011) Qualcomm announced qual-core MSM8964/APQ8064
  - 2.5 GHz / 2MB L2 cache
  - sample in early 2012

Do you rember the specifications of your old laptop computers?

Classic ARM Processors (1)

- ARM7TDMI family
  - Based on ARMv4T architecture with 3-stage pipeline
  - supports the 16-bit Thumb instruction set
  - supports the JTAG Debugger
  - includes a fast Multiplier to support DSP algorithm
  - supports the In-Circuit Emulation interface
- ARM9TDMI family
  - Based on ARMv4T with Harvard cache architecture
  - 5-stage pipeline
  - ARM920T is based on ARM9TDMI with a memory management unit (MMU)

Classic ARM Processors (2)

- ARM9E family / Intel's Xscale
  - Based on ARMv5E architecture
  - Enhanced with DSP instructions
  - Hardware support of Java bytecodes execution
- ARM10 family
  - Based on ARMv5E with MMU
- **ARM11** family
  - Based on ARMv6 architecture
  - Supports SIMD instructions
  - Reduce context switching cost

The first shipped Android phone belongs to ARM11 family.

Android 1.5 (2009/04/27)
ARMv5te optimizations: libc (thumb, memcpy, memset, clz),
dalvik (fast interpreter), skia, surfaceflinger, pixelflinger, audioflinger

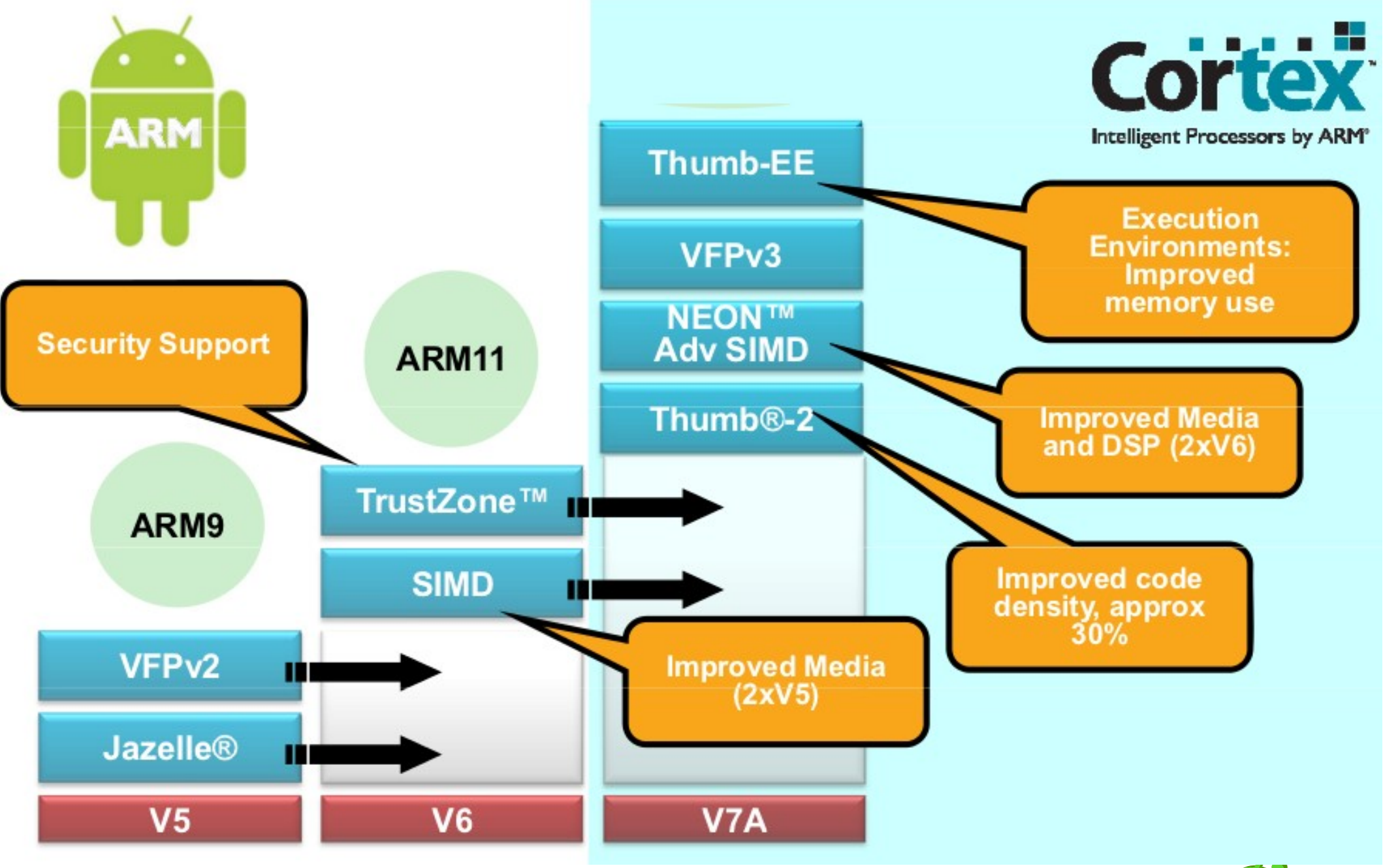
Classic ARM Processors (3)

Android 2.0 (2009q4)
ARMv7-a optimizations: libc (memcpy/neon), dalvik (JIT), skia

- **Cortex** families
  - Based on ARMv7 architecture
  - Supports the new Thumb-2 instruction set
  - NEON SIMD engine
  - **Cortex-A**: For complex OS based applications
    - Production: Cortex-A8 / Cortex-A9
  - Cortex-R: For real-time embedded applications
  - Cortex-M: For deeply embedded, microcontroller type cost sensitive applications
- Cortex-A15
  - Triple dispatch superscalar, out-of-order
  - High performance prefetcher
  - Up to 4M integrated L2 cache
  - 1TB of physical address space
  - Enterprise class performance, mobile power

The era of Android:

High quality mobile computing


ARM
Cortex
Intelligent Processors by ARM
Thumb-EE
VFPv3
NEON™
Adv SIMD
Thumb®-2
Execution Environments: Improved memory use
Improved Media and DSP (2xV6)
Improved code density, approx 30%
Improved Media (2xV5)
Security Support
ARM11
ARM9
TrustZone™
SIMD
VFPv2
Jazelle®
V5
V6
V7A

ARM Optimizations in Android

- Thumb2 optimizations in Dalvik VM
  - On real memory systems with latency the additional instructions in I-cache latency, it makes Thumb2 higher performance than ARM (logic: 1.23x)
- NEON/VFP optimization in skia & dalvik
  - 1.45x in dalvik; 2-5x microbench in skia
- Thumb2 + VFP in V8 JavaScript engine
- ARMv6 atomic/rev in libc & dalvik
- NEON + VFP in StageFright
- SMP in libc, libcutils, dalvik

Again, we were informed of the improvements from hardware and dedicated software.
But, is it enough?

Unproven “Optimization"

“Optimization" → **最佳化** (?)

「最佳化的迷思」

操作績效不理想或想再創佳績
於是就開始進行程式最佳化
在操盤軟體上東調西調
嚐試在變化萬千的參數搭配中找尋聖杯
果然出現創紀錄的績效時
就以為得到聖杯
拿到市場實地操作
卻發現落差很大

最佳化的對象是誰
程式還是策略
拿頭文字D來比喻吧
程式好比藤原拓海的車子
車子當然可以越改越好
但是藤原拓海才是致勝靈魂
對比程式 策略才是致勝關鍵

策略核心邏輯不是藤原拓海等級
參數再怎麼調
也調不出來一個藤原拓海

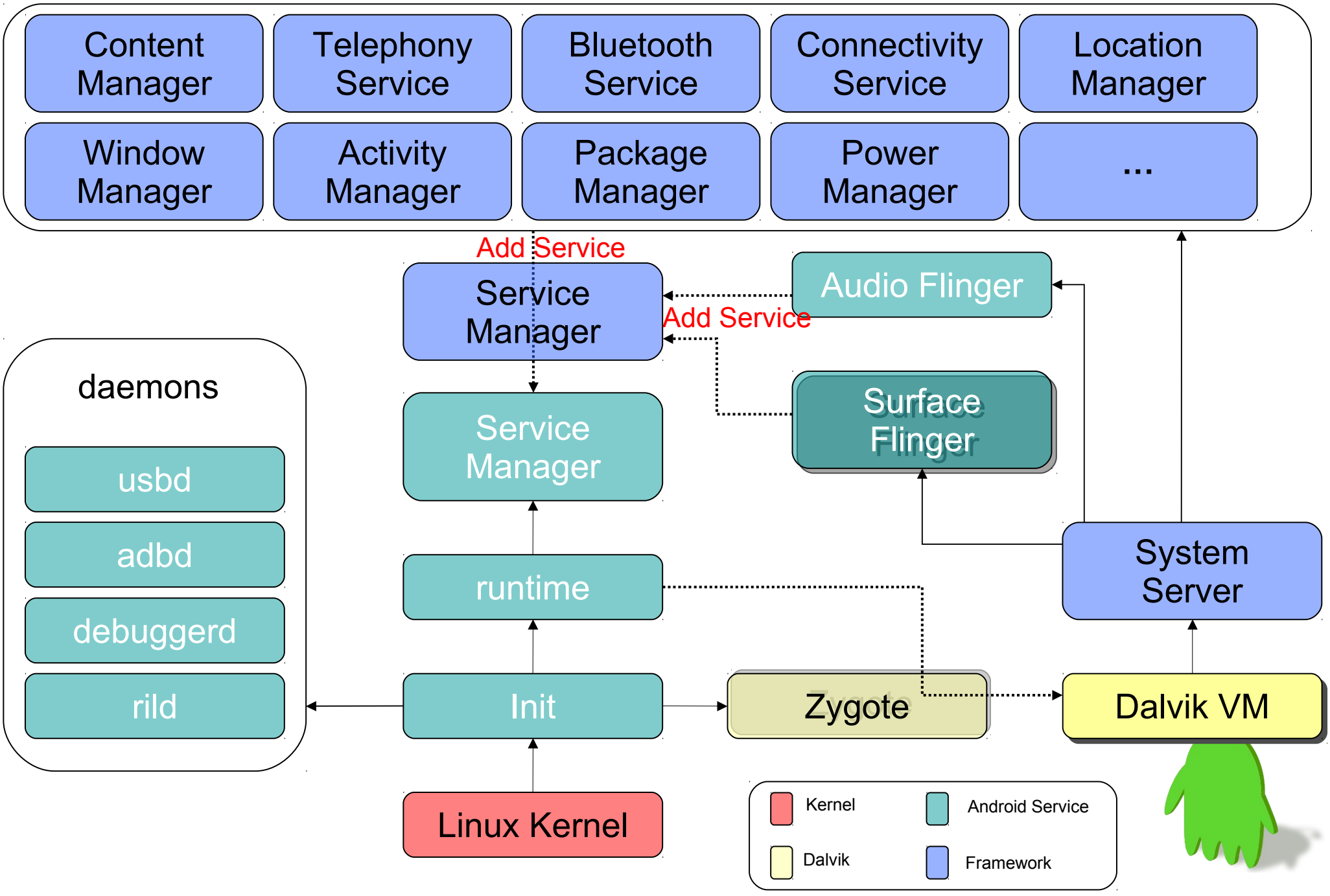
回到以策略為中心點
檢視策略在買賣點、停損停利點與資金控管上
可以改善的空間
才能突破你自己

http://tw.myblog.yahoo.com/trend-trading/article?mid=18&prev=22&next=16

Key Concept: **Strategy**

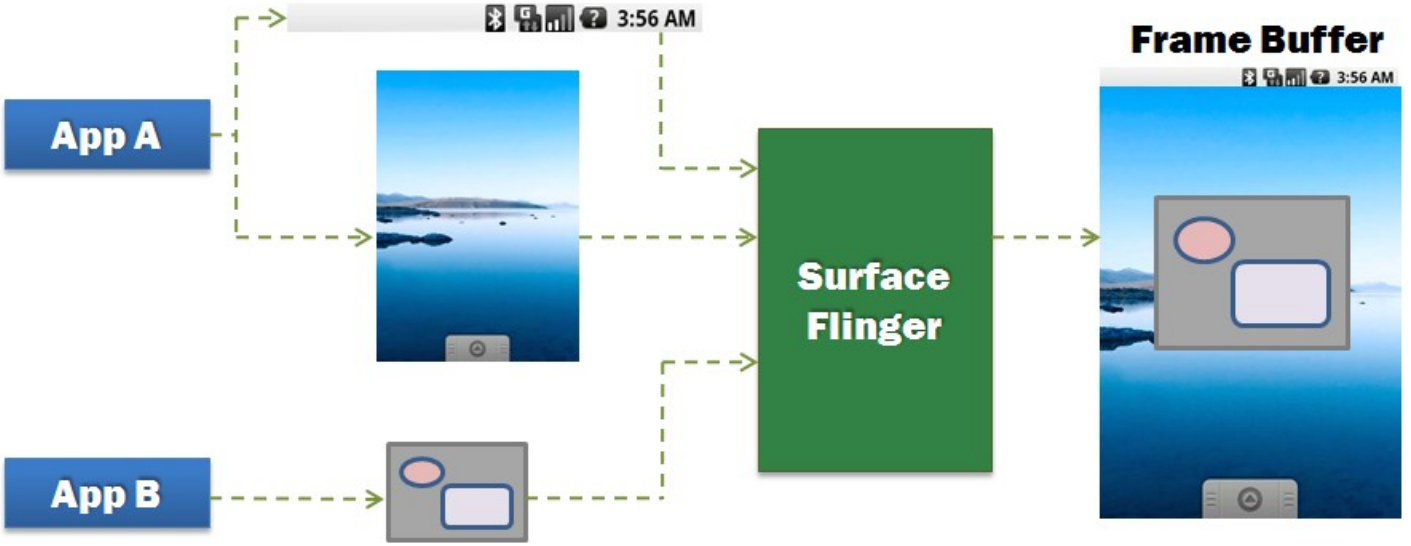
Even runtime is (claimed as) fully optimized, we still have to look into details carefully.

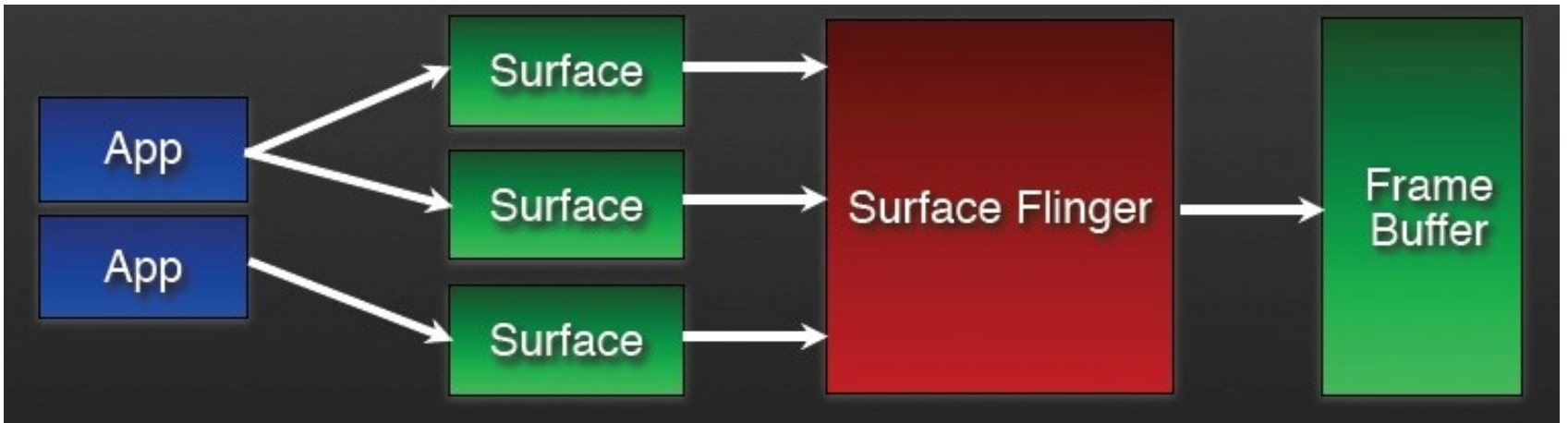
Android Services in Action

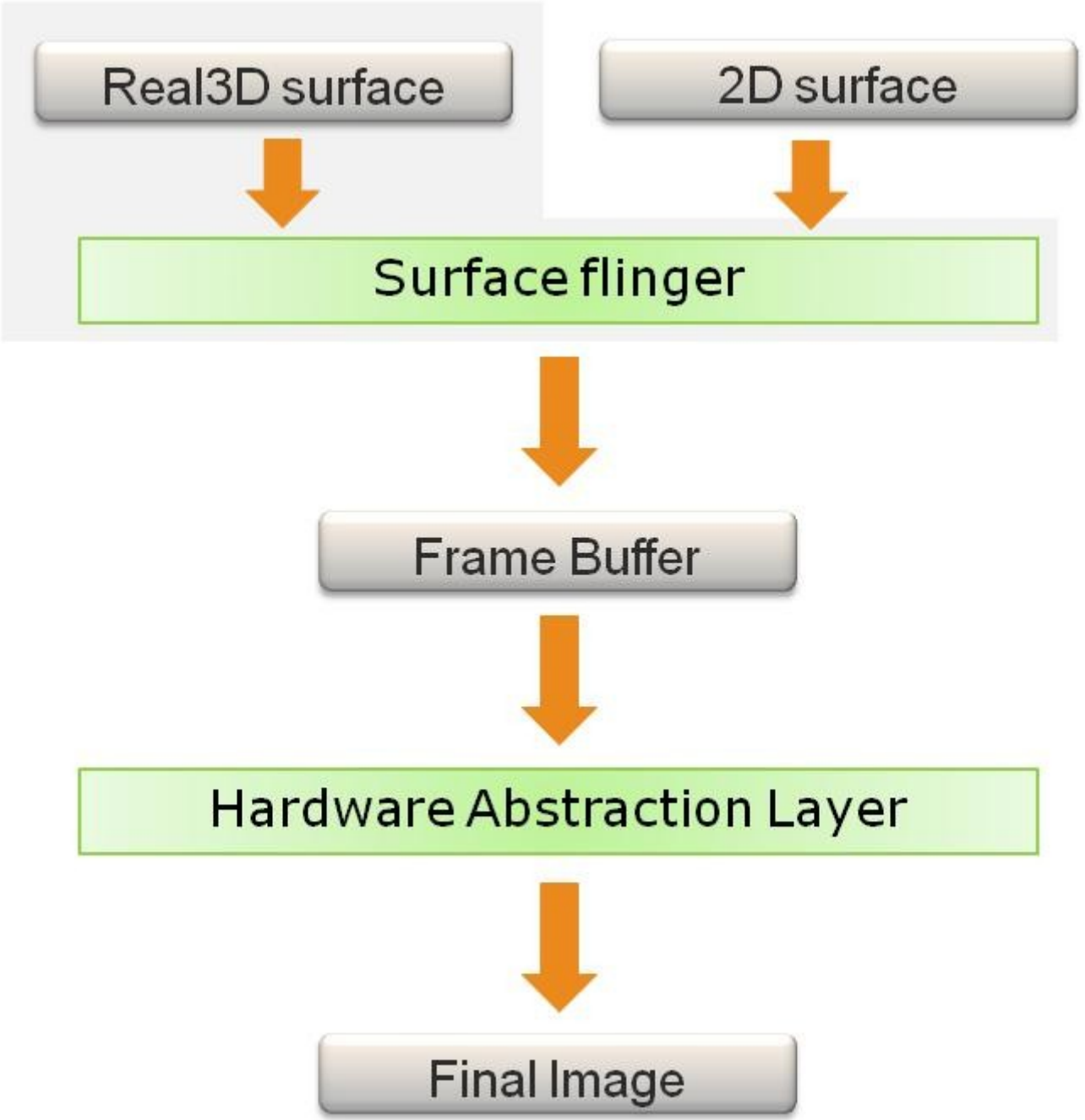


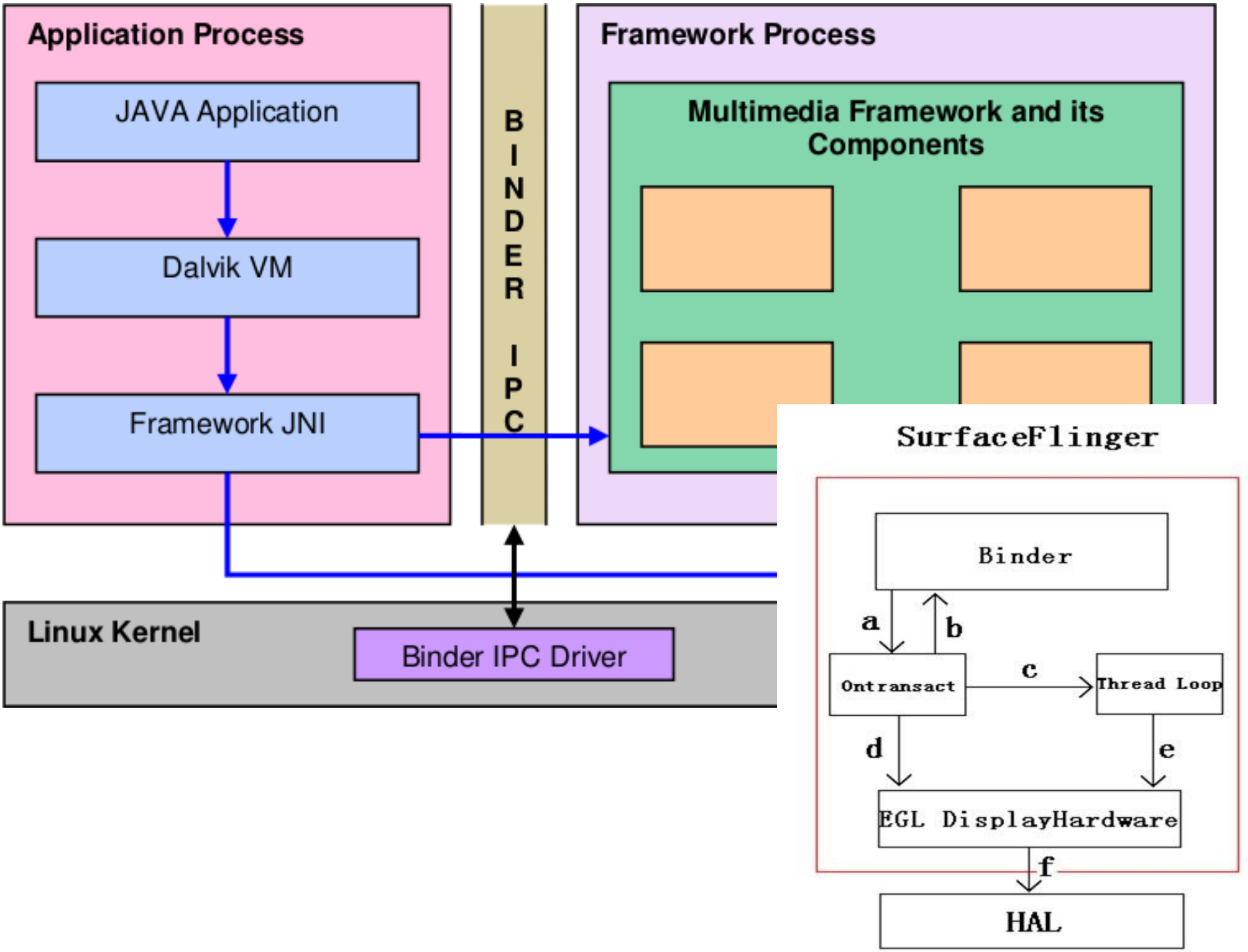
Android SurfaceFlinger

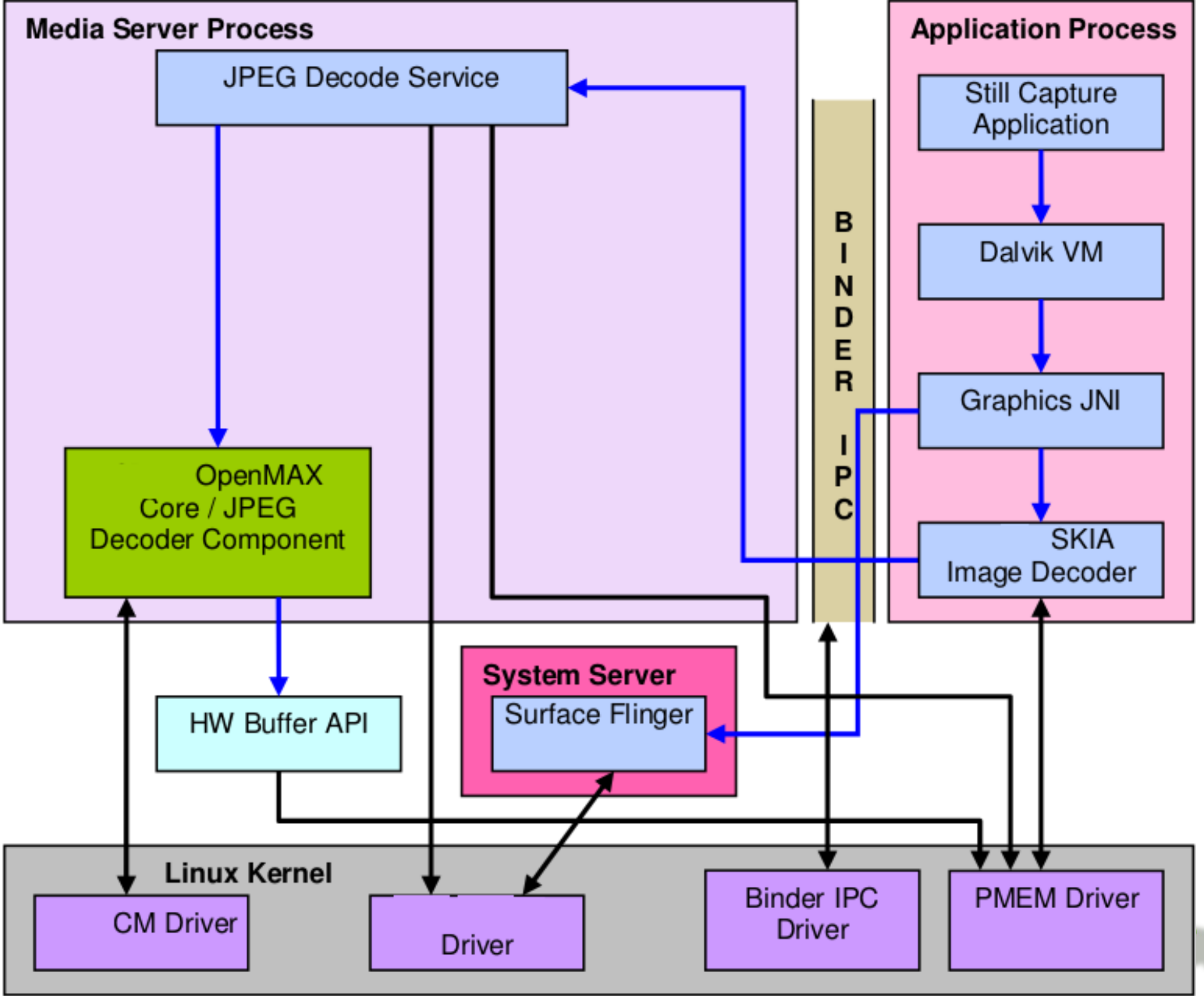
- Properties
  - Can combine 2D/3D surfaces and surfaces from multiple applications
  - Surfaces passed as buffers via Binder IPC calls
  - Can use OpenGL ES and 2D hardware accelerator for its compositions
    - Double-buffering using page-flip

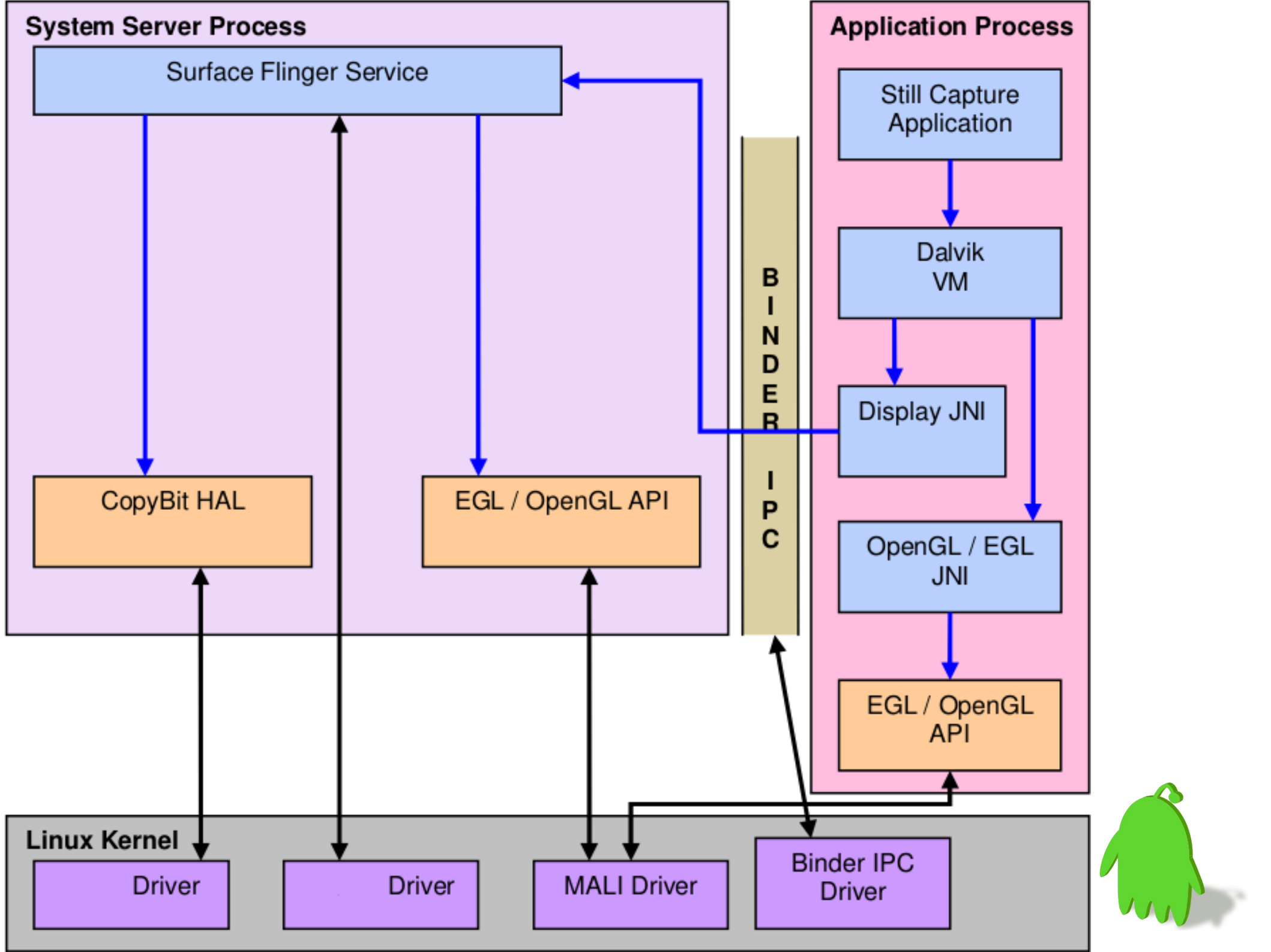





Real3D surface
2D surface
Surface flinger
Frame Buffer
Hardware Abstraction Layer
Final Image

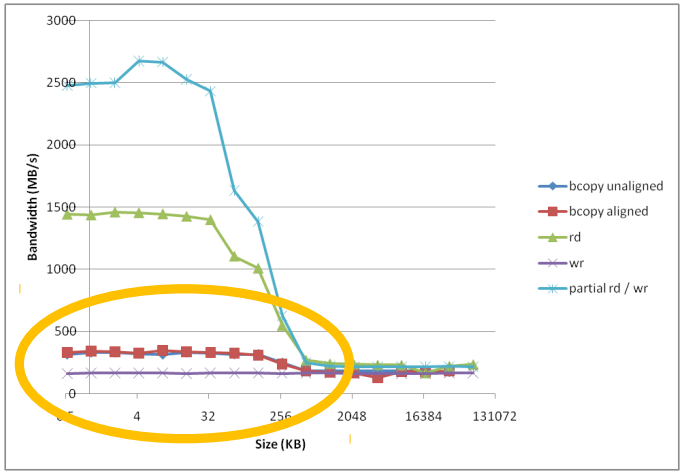

Application Process
JAVA Application
Dalvik VM
Framework JNI
BINDER IPC
Framework Process
Multimedia Framework and its Components
Linux Kernel
Binder IPC Driver
SurfaceFlinger
Binder
a
b
Ontransact
c
Thread Loop
d
e
EGL DisplayHardware
f
HAL


Media Server Process
JPEG Decode Service
OpenMAX
Core / JPEG
Decoder Component
HW Buffer API
Application Process
Still Capture
Application
Dalvik VM
Graphics JNI
SKIA
Image Decoder
BINDER IPC
System Server
Surface Flinger
Linux Kernel
CM Driver
Driver
Binder IPC
Driver
PMEM Driver

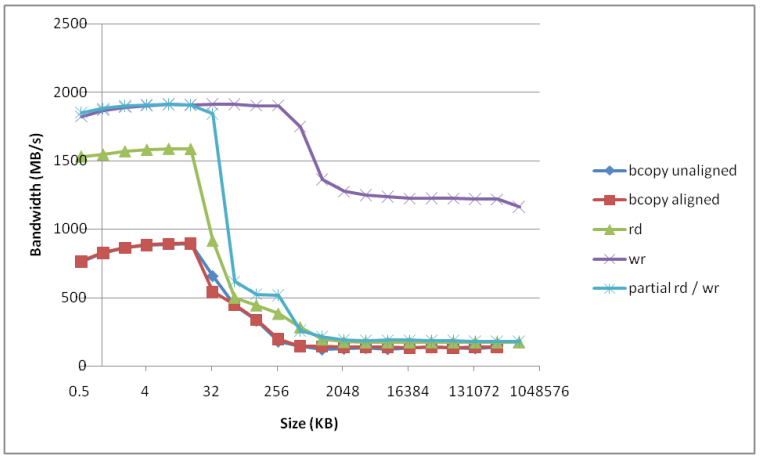

System Server Process
Surface Flinger Service
CopyBit HAL
EGL / OpenGL API
BINDER IPC
Application Process
Still Capture Application
Dalvik VM
Display JNI
OpenGL / EGL JNI
EGL / OpenGL API
Linux Kernel
Driver
Driver
MALI Driver
Binder IPC Driver

Optimizing SurfaceFlinger Techniques

- Measure/Profile the execution flow
- Scenario-driven system analysis
  - 2D/3D animation
  - Video Playback
  - Camera preview/recording
- Check list:
  - Color space converting
  - Eliminate the invalid cases (quality of HAL)
  - Allow zero copy among framework/HAL/Driver
    - Shared memory / IPC cost
  - Any misconfiguration?
  - Cache (write-through vs. write-back)
  - The last: essential routines

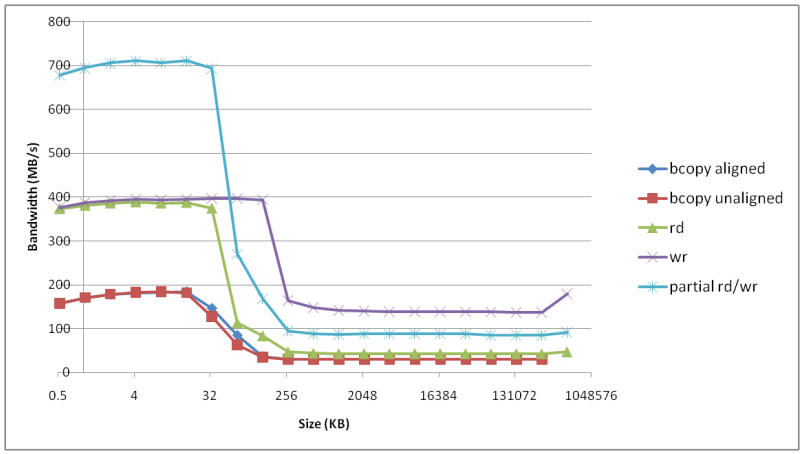


- Write bandwidth greatly affected if caches are configured as write-through
- Configure caches as write-back, with allocate-on -write

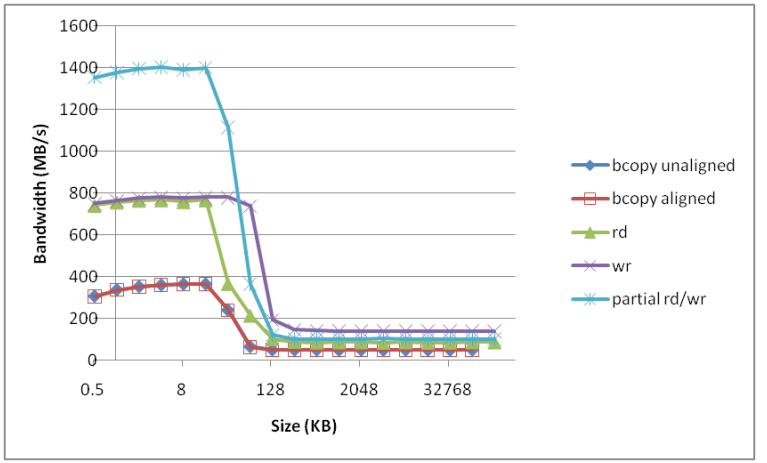


- Memory bandwidth

Single Instance



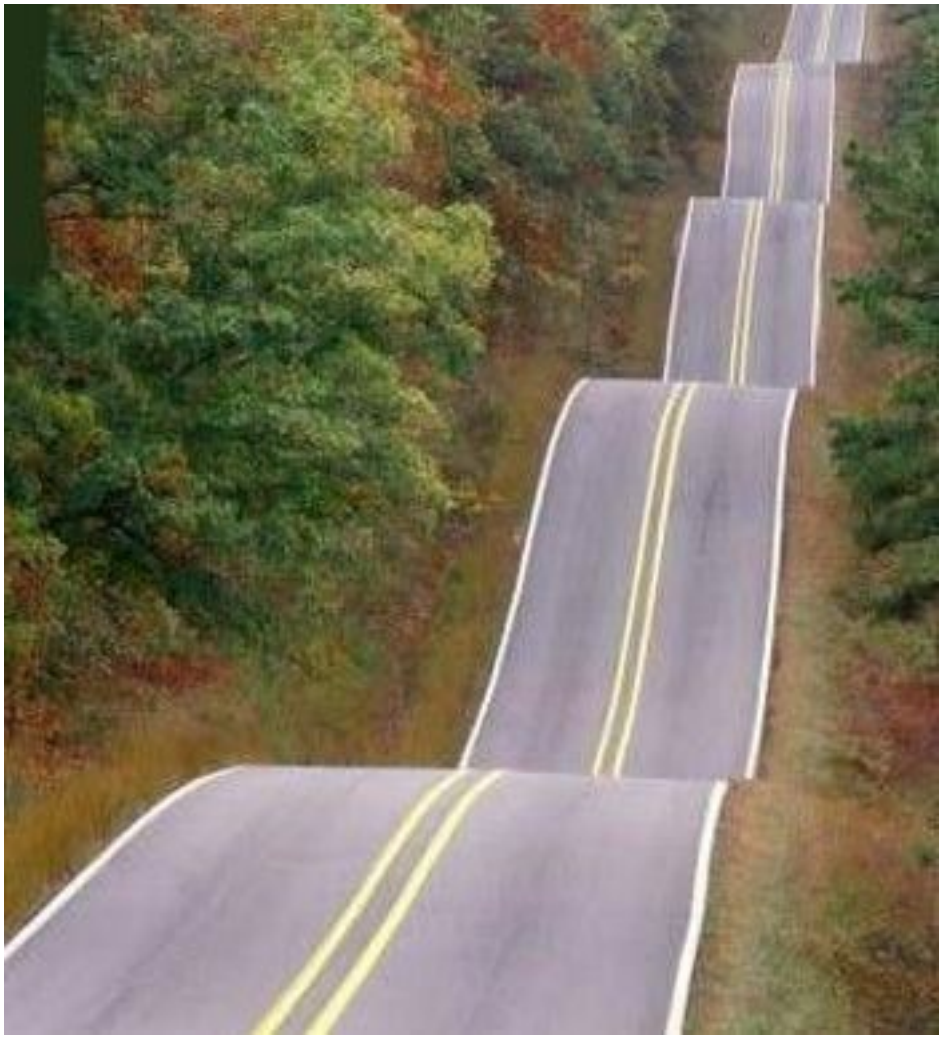
2 Instances



Possibly Premature optimizations

- "**Premature optimization is the root of all evil**"
  - Donald Knuth
- Bionic
  - Glibc-incompatible, No SysV IPC, partial Pthread, incomplete prelink
  - inactive/incorrect kernel header inclusion
  - May not re-use existing system utilities
- Assumed UI behaviors
  - Input event dispatching and handler
  - Strict / non-maintainable state machine (policy)
  - Depending on a certain set of peripherals
- Unclear HAL design and interface
  - Wifi, Bluetooth, GPS, ...

Android Evolution

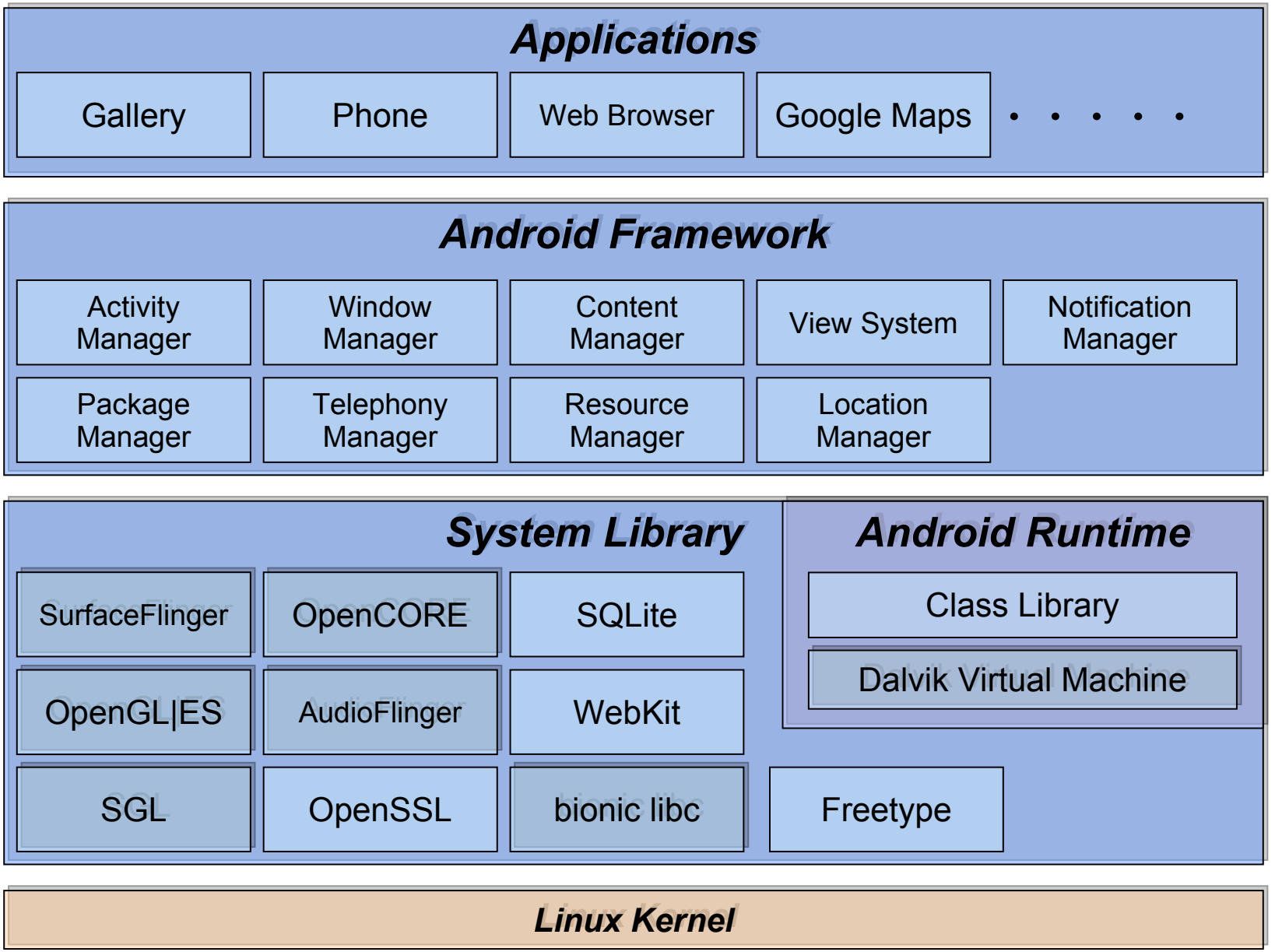


Android is Hardware-driven Software Revolution

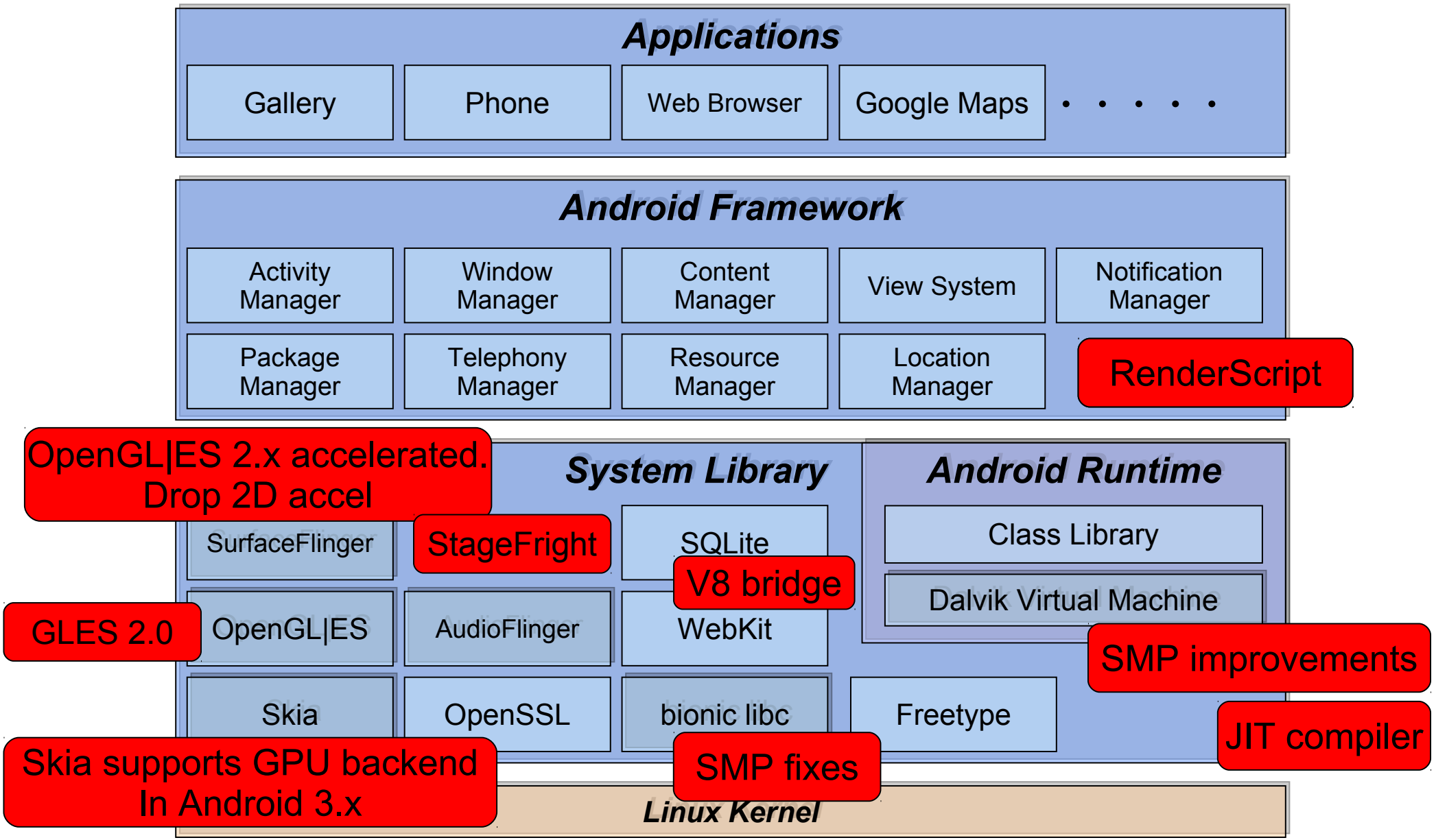
Android Software Evolution

- Excluding the trickery, Android is a huge software project and publishes large parts of source repositories to AOSP (Android Open Source Project).
- “Hardware-driven software revolution” is the trickery. To fairly develop, it has to turn into "evolution".
  - SoC vendors built up the community(-like) hosted Android variants. → Qualcomm, TI, Samsung, ST-Ericsson, ...
- Quality-control is another key factor
  - 0xlab delivered 0xbench and related validation infrastructure.
- Solution to Android ecosystem is to continue to grow the AOSP community
  - need to unify to move forward
  - Reasons why Linaro exists

Functional View (1.5)



Functional View (2.3)



Frequently Asked Questions

- "What is the minimal hardware requirement for running Android?"
  → UX is the key.
- "May I reduce BOM cost in my Android product?"
  → cost down is always possible, but it depends on the overall performance.
- "How can I upgrade Android without pain?"
- "I need XXX feature from newer version of Android. How can I get it into the current and old codebase?"
  → Partially upgrading is feasible for system libraries, but always difficult for Android framework

Mythbusters

Think Difficult

- The Effects of Convergence
  - Mobile computing platform
  - Ecosystem
- Content providers lead the development of containers (devices)
- Innovation is integrated by open source software model. The cost of proprietary software gets much higher.
- To optimize or not to optimize, that is the question.
  - Merge Local optimizations != Optimized globally
  - Hardware revolution sharpens optimization strategies and product directions.
  - The role of your team

http://0xlab.org