Embedded Linux Conference
North America
**Virtual Experience**
June 29 – July 2, 2020

Baylibre

# War Story: Using Mainline Linux for an Android TV BSP

Neil Armstrong
Baylibre SAS

THE LINUX FOUNDATION

# Timeline

- Android & Mainline
- HAL story
- GPU war story
- Boot Flows
- Other integration issues
- Conclusion

# Scope of the project

- Build an “Upstream” AOSP BSP for new Amlogic SoC
- Targets (for now) the TV profile (for Android TV)
- Will use Android 4.19 as initial kernel base
- New SoCs from Amlogic, not yet supported in mainline
- Team had AOSP port experience on very early Android releases (~1.6)

# Android & Mainline Linux

# Android & Mainline

- Android has a long a complex history with mainline Linux
- Recently, Google outlines multiple efforts
  - Project Treble: kernel ABI as “vendor interface” to have a “Generic System Image”
  - “One kernel to boot them all” project to leverage common kernel build

## Android & Mainline

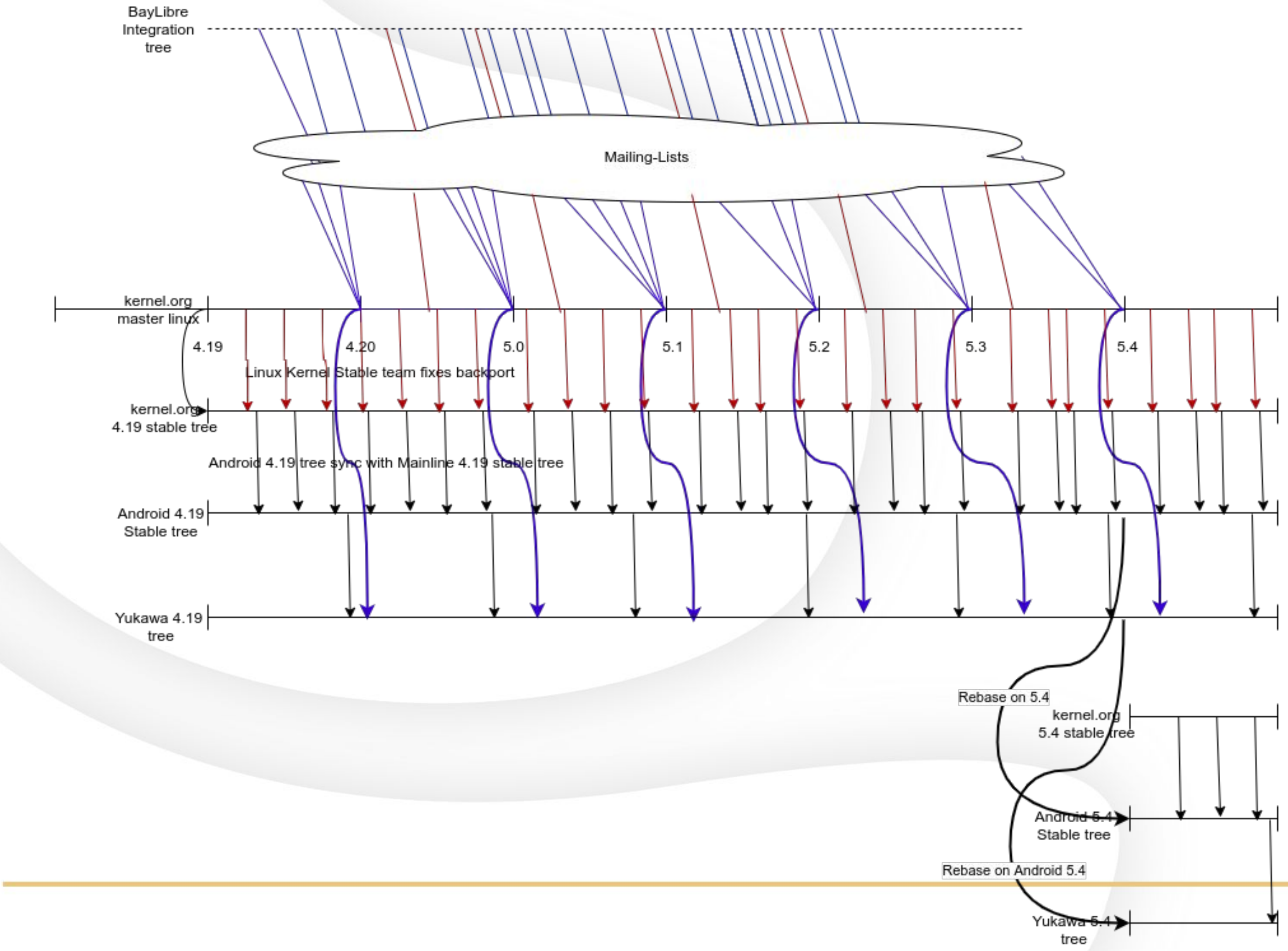
- AOSP 10 can run using pure vanilla kernel
- But we still use an Android derived branch with:
  - Android specific kernel config
  - Android specific kernel patches/fixes
  - Android kernel build YAML

# Android & Mainline

- Our use case ?
  - No vendor, only mainline
  - New SoCs:
    - S905X2
    - S905X3
  - We need to push the support upstream and backport

# Android & Mainline

- The upstream process ?
  - As usual
  - But, we need to backport the upstream patches to the Android tree
  - Using ChromeOS kernel rules for commit message
    - UPSTREAM
    - BACKPORT
    - FROMLIST



BayLibre
Integration
tree

Mailing-Lists

kernel.org
master linux

4.19

4.20

5.0

5.1

5.2

5.3

5.4

Linux Kernel Stable team fixes backport

kernel.org
4.19 stable tree

Android 4.19 tree sync with Mainline 4.19 stable tree

Android 4.19
Stable tree

Yukawa 4.19
tree

Rebase on 5.4

kernel.org
5.4 stable tree

Android 5.4
Stable tree

Rebase on Android 5.4

Yukawa 5.4
tree

# Android & Mainline

- But, why upstream-first ?
  - Easy maintenance
  - Fast rebase (git will drop backports)
  - Ensure code quality
- Cons ?
  - Slow
  - More work to be accepted upstream
  - Upstream won’t accept complex hacky features

# Android & Mainline

Upstream won’t accept complex hacky features ?!

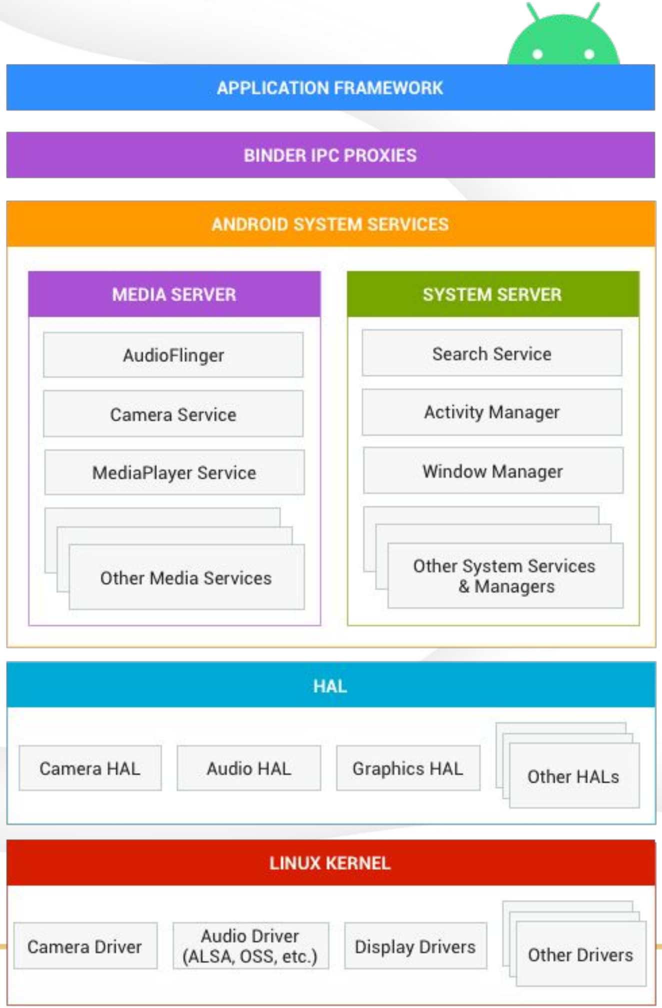
- Not an issue !
- WiP patches can be applied from List
  - So we can take more time to polish them
- Non-upstreamable patches are also possible
  - But we try to limit these
  - We tag them with “ANDROID:”

# Hardware Abstraction Layers

# HAL story

- Android based on Frameworks and HALs
- HALs translates the Frameworks high level system needs into system calls
- Why ?
  - At the time, ARM mainline Linux was very limited
    - No dynamic graphic stack (only fbdev)
    - No sensor framework
    - Very limited Runtime Power Management
    - ...

# HAL story


APPLICATION FRAMEWORK
BINDER IPC PROXIES
ANDROID SYSTEM SERVICES
MEDIA SERVER
AudioFlinger
Camera Service
MediaPlayer Service
Other Media Services
SYSTEM SERVER
Search Service
Activity Manager
Window Manager
Other System Services & Managers
HAL
Camera HAL
Audio HAL
Graphics HAL
Other HALs
LINUX KERNEL
Camera Driver
Audio Driver (ALSA, OSS, etc.)
Display Drivers
Other Drivers

# HAL story

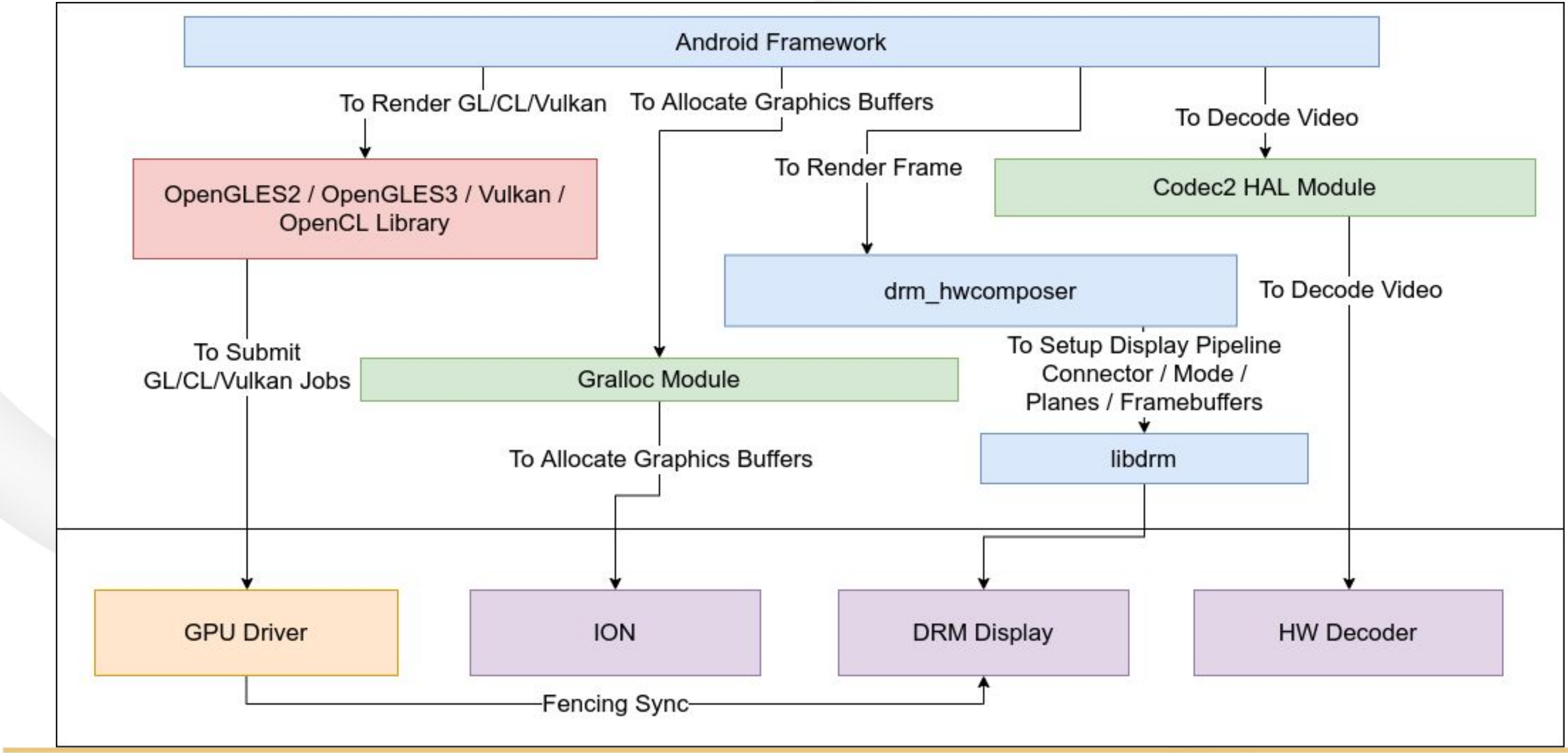
- With the limited mainline Linux kernel
  - Vendor wrote their own HAL for display, GPU, …
  - Google wrote their own PM, syslog… drivers
- It tooks a very long time until AOSP could run on vanilla
  - It took time for Kernel dev to push alternatives
  - It took time for Google to use these alternatives
  - The DRM framework took time to mature
  - There is still a lot of work...

# HAL story

- Our HAL usage ?
- The Yukawa project uses the default HALs for
  - drm-hwcomposer (was a huge blocker)
  - bluetooth
  - Wifi, ....
- Custom HALS :
  - Gralloc for the ARM Mali integration
  - HDMI-CEC, but could be generic
  - Lights

# GPU Integration

# GPU war story


Android Framework
To Render GL/CL/Vulkan
To Allocate Graphics Buffers
To Render Frame
To Decode Video
OpenGLES2 / OpenGLES3 / Vulkan / OpenCL Library
Codec2 HAL Module
drm_hwcomposer
To Decode Video
To Submit GL/CL/Vulkan Jobs
Gralloc Module
To Setup Display Pipeline Connector / Mode / Planes / Framebuffers
libdrm
To Allocate Graphics Buffers
GPU Driver
ION
DRM Display
HW Decoder
Fencing Sync

Generic HAL / Library | Upstream Drivers | Vendor HAL | Out Of Tree Drivers | Closed Source Library

# GPU war story

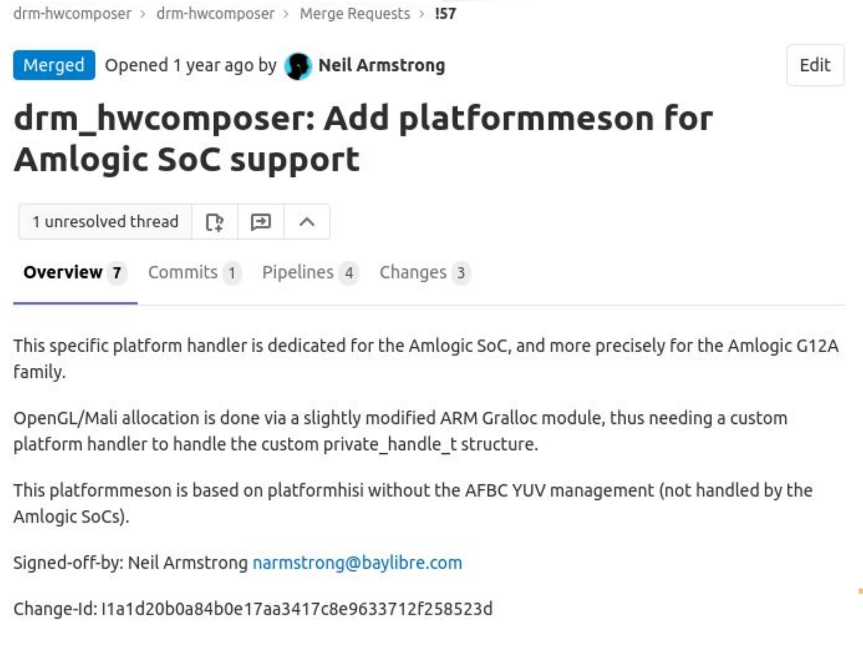
- GPU library <-> gralloc <-> hwcomposer relationship
  - Google made their own OpenGL API
  - A private vendor “private_handle_t” structure is added
  - Is added by gralloc to be used by the HWComposer module
  - Can also be used by the OpenGL library
  - Contains properties of the allocated GPU buffer

# GPU war story

- Mali ?
  - ARM provides a vendor Gralloc module
  - The Gralloc module *version* is **tied** to the OpenGL library *version*
  - E.g: Amlogic modified the *private_handle_t* structure
  - **We are tied to use the Amlogic derived Gralloc module**
- The drm-hwcomposer also needs a vendor implementation
  - Using the vendor gralloc *private_handle_t* define
  - Using the *private_handle_t* structure to import the buffer into DRM

# GPU war story

- But
  - drm-hwcomposer is an external “generic” HAL
  - So -> upstream first !



Opened 1 year ago by Neil Armstrong

**drm_hwcomposer: Add platformmeson for Amlogic SoC support**

This specific platform handler is dedicated for the Amlogic SoC, and more precisely for the Amlogic G12A family.

OpenGL/Mali allocation is done via a slightly modified ARM Gralloc module, thus needing a custom platform handler to handle the custom private_handle_t structure.

This platformmeson is based on platformhisi without the AFBC YUV management (not handled by the Amlogic SoCs).

Signed-off-by: Neil Armstrong narmstrong@baylibre.com

Change-Id: I1a1d20b0a84b0e17aa3417c8e9633712f258523d

# GPU war story

- We still have an issue !
- Low-cost Android TV vendors (Amlogic, Allwinner, Rockchip, …) SoCs usually cannot handle a full 4K UI layer
    - So they limit the Android UI in 1080p max
    - This is done in their Hardware Composer HAL module
- So, can we do the same with drm-hwcomposer ?
    - No
    - It needs a complete HWC API change to separate the
        - Display Mode
        - UI Layer dimensions
    - This are not distinguished as today
    - So we need to “lie” to Android and give a fake “1080p mode” for all 4K modes

# Boot Flows

# Boot Flows

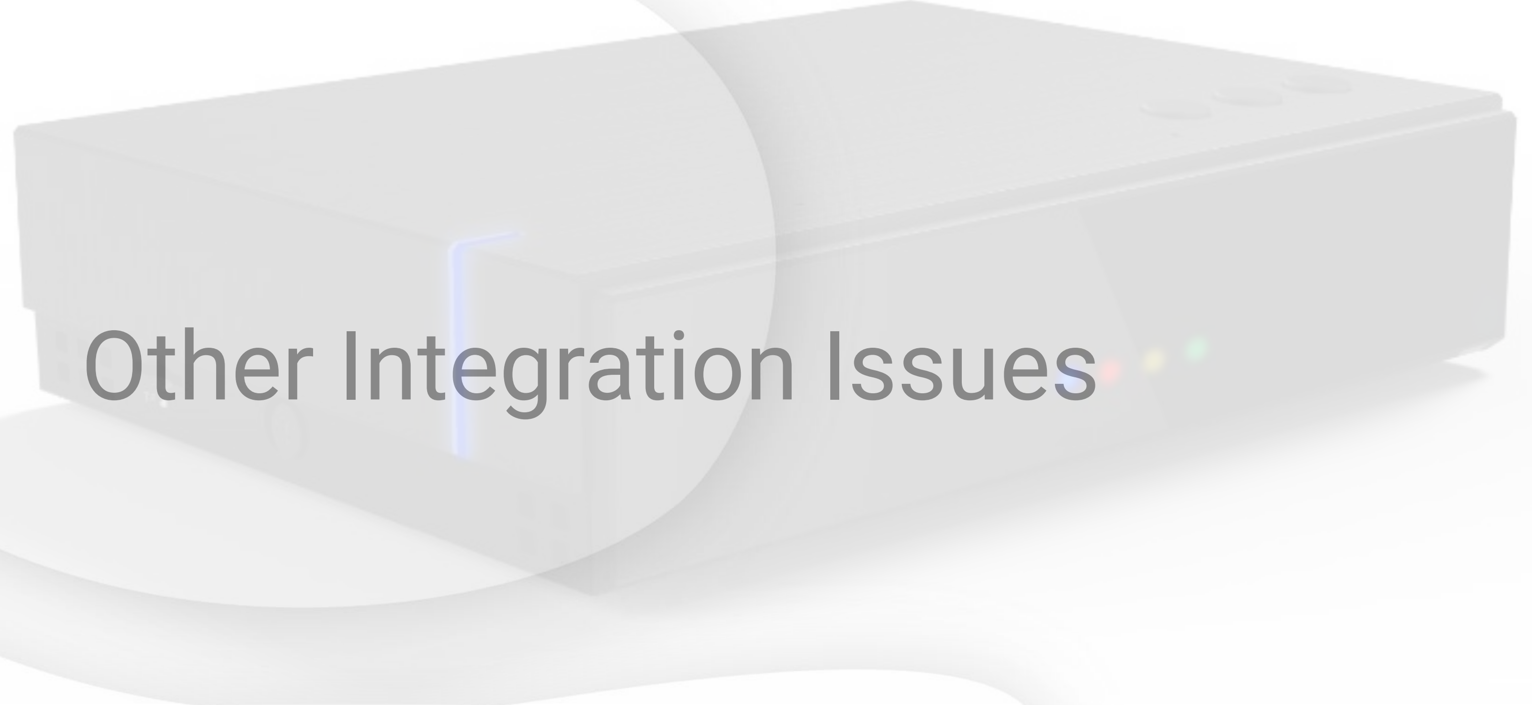
- Old way (pre-Android 9)
  - Kernel as bootimg + initrd (DT added at the end of kernel zImage)
  - Mounts system, mounts vendor and boots
  - Can still be used for Android 9
- New Way v1 (system-as-root)
  - Kernel as bootimg (DT as “second” payload) + eventual DTBO
  - Mounts system using UUID, finds vendor in DT and mounts it
  - Optional for Android 9, Mandatory for Android 10 if not using “New Way v2”
- New Way v2 (dynamic partitions support)
  - Kernel as bootimg (DT as dtb payload) + initrd (required for dm-linear) + eventual DTBO
  - Mounts system & vendor from the “Super” partition and boots
  - Mandatory for Android 10 if not using “New Way v1”

# Boot Flows

- Supporting all boot flows in a single codebase is very hard
- Simplest is to support the last one: Android 10 + System-as-root
- U-boot has regular patchset to support these feature
  - Pushed by Google, TI or other vendors
  - But those are very generic
  - Still needs a complex boot flow script !

# Boot Flows

- The reference board are support in mainline U-Boot \o/
- But we still needs a few hacks on top to meet the complete Android boot flow :-(



# Other Integration Issues

# Other issues

- Audio
  - It’s a mess, Google develops a complete HAL API
  - But no generic ALSA HAL, **at all** !
  - Solution ? re-use the old https://github.com/CirrusLogic/tinyhal
- WiFi
  - It’s a mess, don’t look at it, they still rely(ied ?) on their old wpa-supplicant fork
  - Hopefully it’s moving forward ?
- Similar Display Modes
  - You can’t provide multiple display modes with same width X height X freq
  - No Interlaced support...

# Conclusion

- https://android.googlesource.com/device/amlogic/yukawa/
- Android is much more Mainline Linux friendly
- Common modern Kernel APIs are being adopted
- Still a long road before having:
  - Single kernel for multiple boards
  - Mainline based kernel with very few patches
- Hopefully Panfrost will solve the GPU nightmare
- HWComposer needs some adaptations