DRM/KMS for Android

Kernel display & graphics, testing update

Alistair Delva <adelva@google.com>
Who am I?
- Working at Google with the Android Systems / Kernel team
- Responsible for the Cuttlefish Virtual Device (CVD)
- Also work on dev board support in the Android Open Source Project (AOSP)

Trying to standardize display/graphics/multimedia stacks
- More examples of open source / upstream stacks in AOSP
- Virtual platform should use the same interfaces
- Conformance testing for display via Vendor Test Suite (VTS)

Talk will mostly look at the problem from a kernel PoV
Android on a Legacy Stack

- Kernel
- 3D Driver Core
- Video Codec
- FBDEV
  - Display Driver
    - Scaler
- OpenMAX / Codec2
- gralloc
- hwcomposer
- EGL / Open GL ES
  - Vulkan
- ANDROID FRAMEWORK

- SDL

Vendor driver
Vendor HAL (unmodified)
Vendor-defined kernel interface
Android on an Upstream Stack

- Kernel
  - 3D Driver Core
  - Video Codec
  - DRM
  - Display Driver
  - Scaler

- DRM Rendernode
- New DRM stack
- DRM interop
- Vendor HAL (unmodified)
- Vendor-defined kernel interface
- Implementation challenge for vendor

- OpenMAX / Codec2
- gralloc (e.g. minigbm)
- drm_hwcomposer

- EGL / Open GL ES
  - Vulkan

- ANDROID FRAMEWORK
Upstream stacks in AOSP

- Pixel 3 / 3a / +
- DragonBoard 845c (under review)
  - Same SoC as Pixel 3, but not the same driver
  - Proves Android can run on upstream driver
- Other platforms: Hikey, Hikey960, BeagleBoard X15, Cuttlefish
Cuttlefish Virtual Device (CVD)

- Android for Google Cloud
  - KVM based, built on top of crosvm virtual machine monitor
  - Used by Google for continuous integration testing of changes to Android
  - When you upload to AOSP review, your change is tested on cuttlefish

- Cuttlefish uses an upstream graphics stack
  - Can boot upstream kernels (just a defconfig)
  - SwiftShader, for software GPU use cases
  - Mesa (virgl) for hardware acceleration
    - $ launch_cvd -gpu_mode=drm_virgl
  - minigbm (gralloc), drm_hwcomposer

- Planned features
  - Vulkan support
  - More KMS planes, more pixel formats
VTS enforces shipping one of three kernels for newly launching devices:

- Android P - 4.4, 4.9, 4.14
- Android 10 - 4.9, 4.14, 4.19
- Android 11 - 4.14, 4.19, 5.4 (GKI)

Devices get two years of upgrades too:

- Lots of kernels to test
- Android 11 - 4.4, 4.9, 4.14, 4.19, 5.4

Vendor kernels might make it worse (more on this later):

- Inconsistent uapi / kernel feature set, no LTS fixes, more difficult to test

### Upstream stacks for vendors?

**Vendor A**
- Linux 4.14
- ion from 4.9
- drm from 4.17
- v4l2 from 4.14

**Vendor B**
- Linux 4.14
- ion from 4.14
- drm from 4.14
- v4l2 from 4.19

**Vendor C**
- Linux 4.14
Generic Kernel Image (GKI)

Generic ARM64 kernel for all Android devices

**Branches**
- android-mainline
- android-4.19
- android-5.4 (soon)

**Configuration**
- Single Kernel Configuration (gki_defconfig)
- Suitable for all ARM64 based devices

**Toolchain**
- Single Toolchain (Clang)
- Hermetic Build

**Scope**
- All ARM64 Android devices
- Validation only on x86_64

https://lwn.net/Articles/771974/

tl;dr Aims are to reduce fragmentation, provide security patches for everybody
GKI - ABI Monitoring

- Define a baseline ABI
- Keep it along with your sources
- Establish ABI checking (e.g. build_abi.sh) as mandatory test before merging
- Changes targeting Android Common Kernels have to pass this test in AOSP Gerrit

--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -22,6 +22,7 @@

    extern struct user_namespace init_user_ns;

-struct uts_namespace {
+struct uts_namespace {
  int dummy;
  struct kref kref;
  struct new_utsname name;
-  struct user_namespace *user_ns;
+  struct user_namespace *user_ns;

    "int uts_namespace::dummy": offset diff 0 (in bits) at utsname.h:28:1
    "struct uts_namespace": offset diff 0 (in bits) at utsname.h:24:1
    "struct new_utsname": offset diff 0 (in bits) at utsname.h:26:1

GKI - Compliance Structure

Before GKI

**Kernel**
- boot.img
  - kernel
  - ramdisk.cpio.gz
  - /init

**Userspace**
- system.img
- vendor.img
  - /vendor/lib/modules/*.ko

Most drivers are built-in

With GKI

**Kernel**
- boot.img
  - kernel
  - ramdisk.cpio.gz
  - /init

**Vendor**
- vendor-boot.img
  - ramdisk.cpio.gz
  - /lib/modules/*.ko

- vendor.img
  - /vendor/lib/modules/*.ko

Subsystems are built in, most drivers are not

GKI

Combined by bootloader

Boot drivers

GSI

Non-boot drivers

Vendor modified

AOSP

Vendor

Vendor modified
GKI - Implications for Display/GPU

- Display drivers are modules, can’t be built-in
  - Stable ABI within LTS release (4.19.x through 4.19.y)
    - Maintained by Android kernel team
    - Not the whole kernel, some security changes might break compat
  - Modules can still be patched by vendors as before

- dma-buf, drm, etc. is built in
  - Will get security + bugs fixes via LTS
  - We might backport subsystems to older kernels

- Display/GPU drivers not using DRM/KMS will be vendor’s responsibility
  - Can only use symbols exported by GKI

- Verified as part of Android VTS
Testing upstream stacks

- Not just a kernel effort
  - `drm_hwcomposer` used on many AOSP platforms
    - hikey, hikey960, cuttlefish, db845c
  - Mesa used on cuttlefish (virgl) and db845c (freedreno)
  - Teams at Linaro keeping these projects up to date in AOSP

- `igt-gpu-tools` has been added to AOSP
  - Enables whole DRM subsystem testing from userspace
  - Made some Android build system / porting changes
    - [https://android-review.googlesource.com/q/topic:igt-android](https://android-review.googlesource.com/q/topic:igt-android)
  - Still working on baseline test plan for AOSP platforms, Pixel

- Detection of DRM display driver will be added to Android VTS
  - Detection will trigger `igt-gpu-tools` on those display drivers

- Can be tough to test upstream when device ecosystem runs older kernels
  - Virtual and AOSP platforms can help keep us honest
Porting IGT to Android (again)

- IGT needs to run natively on Android
  - Requirement for VTS integration
  - Have to use Blueprint files (no meson)

- Changes to AOSP to expose dependencies
  - libkmod libelf libunwind

- Mock implementations
  - libcairo libglib2.0 libpciaccess

- WIP
  - ifdef/mock/add libudev libprocps
  - Getting more tests to run on HW

- Future
  - Chamelium testing w/ AOSP devices?
Backporting subsystems?

- Display/graphics/multimedia especially fragmented
  - Vendors forward-port or backport subsystems anyway
  - ‘Upstream first’ isn’t really working for mobile SoCs

- Backporting DRM core from latest LTS to older LTS kernels
  - For Android 11: android-{5.4,4.19,..} with same DRM core?
  - Will it help ‘upstream first’, display/graphics/multimedia fragmentation?

- Other technical debt
  - Deprecate ion, replace with dma-buf heaps (will miss 5.4)
  - Backport dma-buf from 5.5 to android-{5.4,4.19,..}?
  - V4L2 Request API (for Codec2)

- Future
  - Reusable syncs (like DRM syncobj) for all drivers
  - Start looking at codecs, camera
Questions?