Android Virtualization

Current and future use cases

Alistair Delva <adelva@google.com>
Android Virtual Devices (AVDs)

- Many implementations:
  - Android Emulator - Part of Android Studio
  - Cuttlefish
  - Vendor IoT or automotive solutions

- Some 'no brainer' virtio drivers:
  - virtio_blk
  - virtio_console
  - virtio_net
  - virtio_rng

Everybody uses these, they 'just work'
Cuttlefish

- AVD based on **crosvm** (previously QEMU)
- Runs locally, or on Google Cloud Platform (GCP)
- Developed upstream: AOSP, mainline Linux
- Originally used a hand-rolled ‘virtual SoC’ architecture, but now aligned to virtio
Cuttlefish in P

- Used vsoc driver, not virtio
  - Shared memory architecture based on QEMU ivshmem

- Upstreamed vsoc as stop gap; long term plan was to move to virtio
  - Wanted to leverage cuttlefish for kernel testing
  - Being able to work with upstream kernels helped virtio transition
Cuttlefish in Q

- Supported crosvm and QEMU
- Some functionality was converted to use emulated hardware instead
  - Not ideal, but no upstream solution
- Still used vsoc driver, but mostly converted to virtio
  - On crosvm we used vsocket for gralloc, hwcomposer, but it was slow
Cuttlefish in AOSP master

- No more *vsoc* driver :)
- New graphics stack isn’t the default, but can be switched on:
  
  $ launch_cvd -gpu_mode=drm_virgl
- Last step is to purge old graphics stack, drop *vsoc* from staging
Learnings from virtio_gpu

- Upstream moves very quickly; things break a lot
  - Fortunately, ChromeOS graphics team adapted

- Three kernels per Android release makes virtio_gpu extra hard
  - ‘Solved’ by backporting virtio_gpu to 4.14, 4.19
  - Backported for EDID support, in/out fence support, other fixes

- Issues we encountered
  - Multiple plane support needed for hardware composer
  - Needs better pixel format support in virtio_gpu_2d
  - Video overlay support (e.g. YV12) would be nice
  - Baked assumptions that virgl will be used, but we want SwiftShader too
  - Stride, format modifier queries needed to expose e.g. FBC
Camera Virtualization?

- Media/V4L2 has no reference in Android Common Kernel / Generic Kernel Image

- Emulated/fake Camera needs a lot of guest CPU, many threads to run well on GCP
  - Offload to host instead?

- Many testing use cases only need ‘EXTERNAL’ e.g. UVC camera
  - Could use USB passthrough, but we don’t have USB at the moment
  - Simplest mode, doesn’t really test Android Camera HAL

- Camera HAL ‘FULL’ or ‘LEVEL_3’ would require complex virtio driver(s)

- Sensor + interface + scaler + IPAs
  - One driver or many?
Virtualization vs Virtual Drivers

- Android WiFi required a real nl80211 driver

  - We used mac80211_hwsim, and tunneled frames using vsoc to mac80211_hwsim on host
    - Host setup required root permissions; not ideal
    - Not Ethernet between mac80211_hwsim implementations
    - Host bridge setup more complicated

  - Upstreamed virt_wifi, an rtnetlink driver that wraps another Ethernet device
    - We get the advantages of virtio_net, but can also pretend to be WiFi!

- But what about inter-VM wifi testing?
  - Not possible with virt_wifi because there are never any 802.11 frames

- Do we need virtio_wifi?
Virtualizing Other HALs

- Other non-upstream virtio drivers:
  - `virtio_audio` (ACRN)
  - `virtio_gpio` (ACRN)
  - `virtio_i2c` (ACRN)
  - `virtio_hdcp` (ACRN)
  - `virtio-vdec` (crosvm)
  - `virtio-wl` (crosvm)
  - ..probably more..

  Fitting needs of a particular use cases

- Other proprietary hypervisors w/ virtio e.g. COQOS, QNX, ...

- More virtio drivers! But not at the expense of hypervisor compatibility

- More Android virtual platform consolidation
Questions ?