Buffer Sharing Synchronization for Graphics & Media

Gustavo Padovan
Principal Software Engineer @ Collabora
Agenda

• Explicit Synchronization
• DRM
• V4L2
Explicit Synchronization

- Userspace aware synchronization
- Avoid blocking/waiting on userspace
- Synchronization based on fences
- More efficient pipelines
- Initially added by Android
Explicit Synchronization
Explicit Synchronization
Explicit Synchronization
Sync File Framework

- Originally from Android
- Uses fd for fence passing
- Has a timeline for ordering
- `dma_fence` to represent a fence
- `sync_file` for fd passing
Fence Timeline

- Monotonically increasing counter
- Usually one timeline per driver context
- Sometimes one timeline per fence
**dma_fence**

- Represents a value on the timeline
- Three states: active, signaled and error
Sync File

- Wrap `dma_fence` into a file
- Shared via fd-passing to/from userspace
Sync File

- Sync Files can be merged!
- It can contain many dma_fence
Sync File Framework – userspace API

- sync_wait(fd, timeout)
- fd3 = sync_merge(fd1, fd2)
- sync_fence_info(fd, info)
Agenda

- Explicit Synchronization
- DRM
- V4L2
DRM/KMS Explicit Synchronization

- Entirely in DRM Core
- Developed for Atomic Modesetting
- Uses DRM Properties
- In-fences: fences to wait before scanout
- Out-fences: fences signaled at scanout
- Mainline since Linux 4.10
DRM/renderer

- Similar to KMS side
- Extends execbuf ioctl on each driver
- Add sync_file/fences support on each driver
- Supported by a few GPU drivers already
MESA

- EGL_ANDROID_native_fence_sync
  - Receive out-fence fd
- EGL_ANDROID_wait_sync
  - Wait for in-fence to signal
Android

- Explicit Synchronization even before mainline
- HWC2 supports mainline fence semantics
- drm_hwcomposer: DRM-based HWC2
- Vendors can use DRM/KMS now!
Wayland/Weston and X11

- chromium wayland server (WIP)
- Gnome-shell/Mutter wayland server (WIP)
- Vulkan client on MESA (WIP)
- X11/xorg protocol and implementation (WIP)
- Enable smart decisions
Agenda

- Explicit Synchronization
- DRM
- V4L2
V4L2 Explicit Synchronization

- Add fences to CAPTURE and OUTPUT queues
- In-fence: fences to wait before using the buffer
- Out-fence: signals when the buffer is ready
- No guarantee of ordering of buffers
- Only targeting ordered driver queues for now
- WIP
V4L2 – Before Fences

1: QBUF()
2: QBUF()
START_STREAMING(
3: QBUF()
1: DQBUF()
2: DQBUF()
3: DQBUF()
1: QBUF()

1: vb2_core_qbuf()
2: vb2_core_qbuf()
start_streaming()
3: vb2_core_qbuf()
1: wake up
2: wake up
3: wake up

...
V4L2 – in-fences

- in-fence: Passed in QBUF() fence_fd field
- V4L2_BUF_FLAG_IN_FENCE should be set
- VIDEOBUF2 sets fence callback
- Can’t be queued to driver before fence signal
V4L2 – out-fences

- QBUF() should have V4L2_BUF_FLAG_OUT_FENCE
- Remember: No guarantee of ordering
- new V4L2 event: V4L2_EVENT_OUT_FENCE
- Call VIDIOC_SUBSCRIBE_EVENT()
- Receive event with DQEVENT()
- Event provide buffer index and out_fence_fd
V4L2 – After fences

![Diagram of V4L2 after fences process]

1. QBUF()
2. QBUF()
START_STREAMING()
1: DQEVENT()  2: DQEVENT()
1: DQBUF()  2: DQBUF()
1: QBUF()

1: vb2_core_qbuf()
2: vb2_core_qbuf()
start_streaming()
1: buffer_queued()
2: buffer_queued()
1: fence_signal() / wake up
2: fence_signal() / wake up

1: enqueue_in_driver()
2: enqueue_in_driver()
1: buffer_done()
2: buffer_done
V4L2 - usecases

- Android/ChromeOS HAL3
- Camera App
- Color converter and scalers
- Gstreamer
- Encoders
Future Fences

• TODO
• Add a new syscall to get an empty/future fence
• Saves a round trip of waiting for the out-fence fd
Thank you!

Gustavo Padovan

gustavo.padovan@collabora.com

www.padovan.org

www.collabora.com