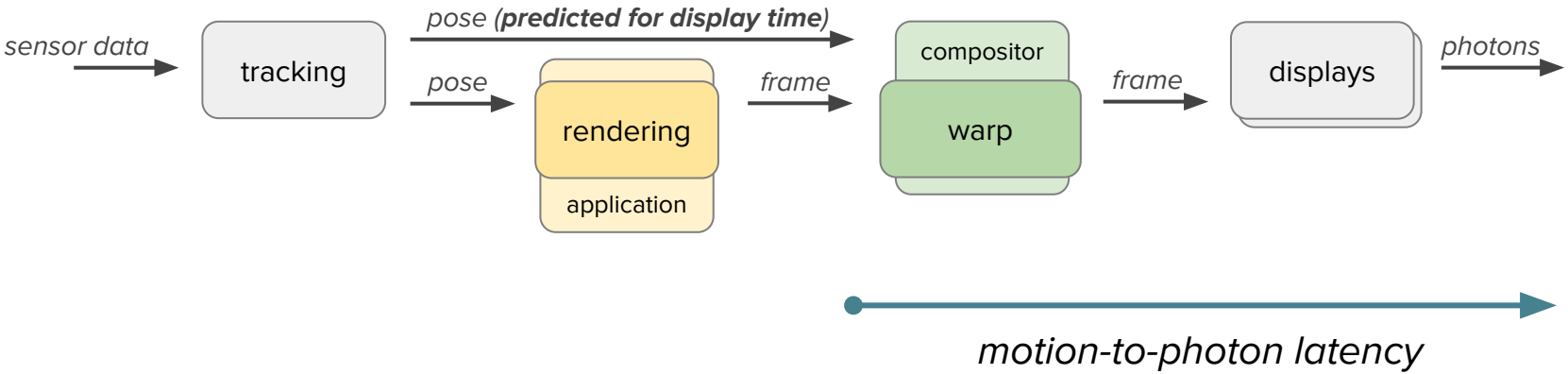
A case study on frame presentation from user space via KMS

Heinrich Fink
DAQRI @ XDC 2019


2 POINT
INSPECTION
WIRE HARNESS 1
WIRE HARNESS 1

AR Pipeline


sensor data
tracking
pose (predicted for display time)
pose
rendering
application
frame
compositor
warp
frame
displays
photons
motion-to-photon latency

Effects of different latencies

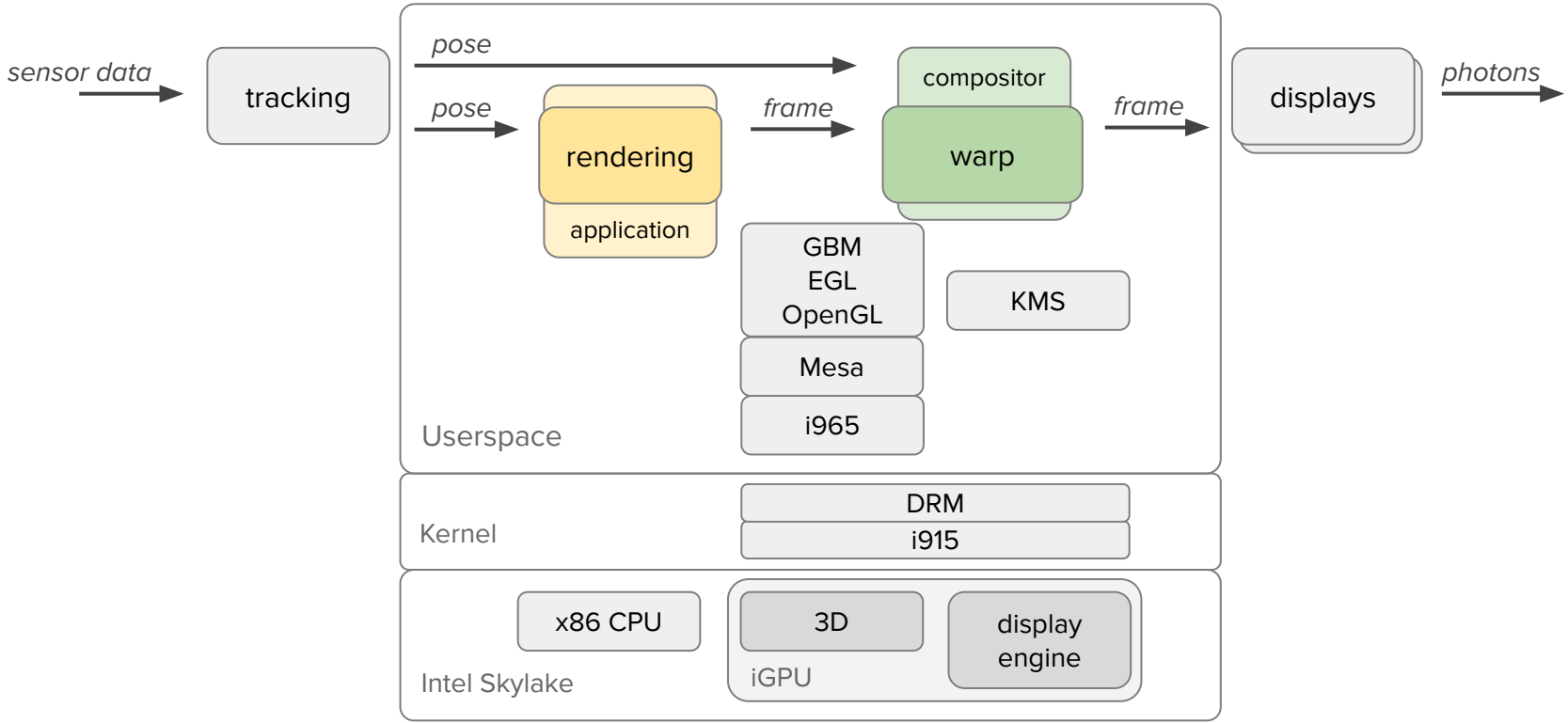
high M2P latency
no prediction

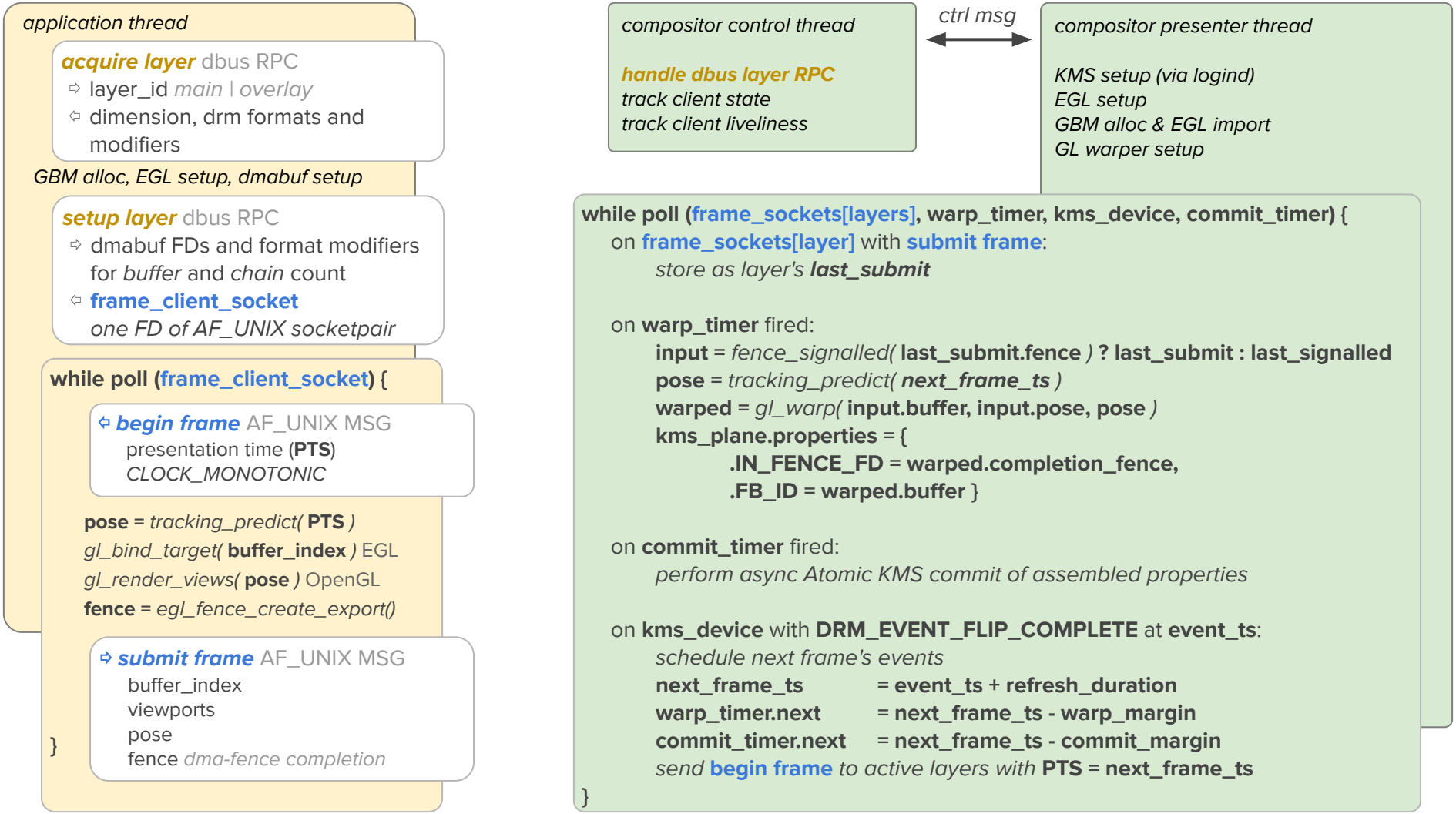
low M2P latency
no prediction

low M2P latency
with prediction

DAQRI KMS Compositor

DAQRI Compositor SW Stack

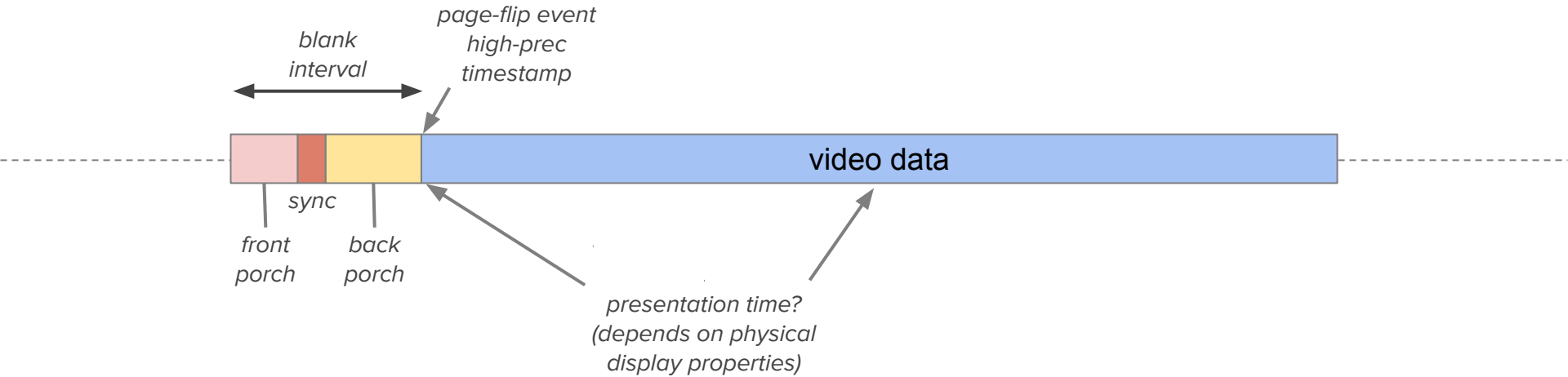



application thread
acquire layer dbus RPC
⇨ layer_id main | overlay
⇦ dimension, drm formats and modifiers
GBM alloc, EGL setup, dmabuf setup
setup layer dbus RPC
⇨ dmabuf FDs and format modifiers for buffer and chain count
⇦ frame_client_socket
one FD of AF_UNIX socketpair
while poll (frame_client_socket) {
⇦ begin frame AF_UNIX MSG
presentation time (PTS)
CLOCK_MONOTONIC
pose = tracking_predict(PTS)
gl_bind_target(buffer_index) EGL
gl_render_views(pose) OpenGL
fence = egl_fence_create_export()
⇨ submit frame AF_UNIX MSG
buffer_index
viewports
pose
fence dma-fence completion
}
compositor control thread
handle dbus layer RPC
track client state
track client liveliness
ctrl msg
compositor presenter thread
KMS setup (via logind)
EGL setup
GBM alloc & EGL import
GL warper setup
while poll (frame_sockets[layers], warp_timer, kms_device, commit_timer) {
on frame_sockets[layer] with submit frame:
store as layer's last_submit
on warp_timer fired:
input = fence_signalled(last_submit.fence) ? last_submit : last_signalled
pose = tracking_predict(next_frame_ts)
warped = gl_warp(input.buffer, input.pose, pose)
kms_plane.properties = {
.IN_FENCE_FD = warped.completion_fence,
.FB_ID = warped.buffer }
on commit_timer fired:
perform async Atomic KMS commit of assembled properties
on kms_device with DRM_EVENT_FLIP_COMPLETE at event_ts:
schedule next frame's events
next_frame_ts = event_ts + refresh_duration
warp_timer.next = next_frame_ts - warp_margin
commit_timer.next = next_frame_ts - commit_margin
send begin frame to active layers with PTS = next_frame_ts
}

DRM_EVENT_FLIP_COMPLETE timestamp

- timestamp passed to drmHandleEvent() ➜ page_flip_handler2
- can be *high-precision* (HW corrected) if supported by driver
  - **`struct drm_driver{ .get_vblank_timestamp }`** *needs to be implemented*
- driver-internal semantics go back to GLX_OML_sync_control:
  - ...***time the first scan line of the display begins passing through the video output port***...
  - i.e. time immediately after vblank
  - this is critical to know for warping, especially when modes have weird front/back porches

How to get *presentation time* in KMS?



Missing in user space

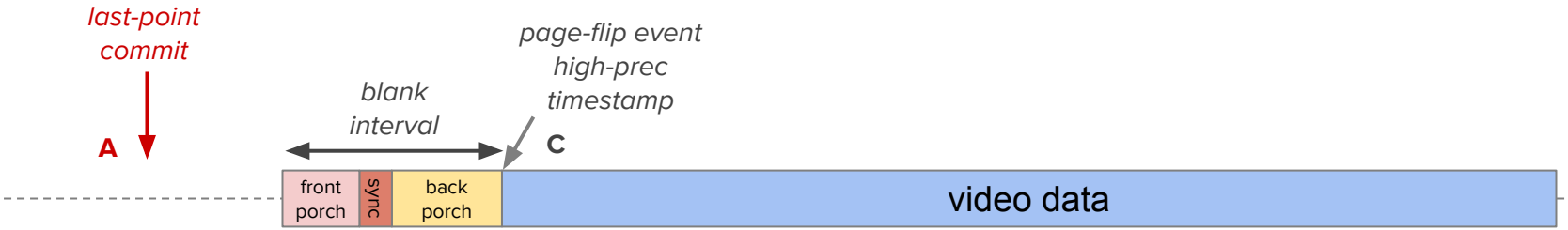
- knowledge whether driver supports high-prec timestamps
  - could be solved by new Atomic KMS property?
- documentation of timestamp semantics
  - i.e. bring over from OML_sync_control
  - but are all drivers with high-prec timestamp support actually implementing it according to spec?

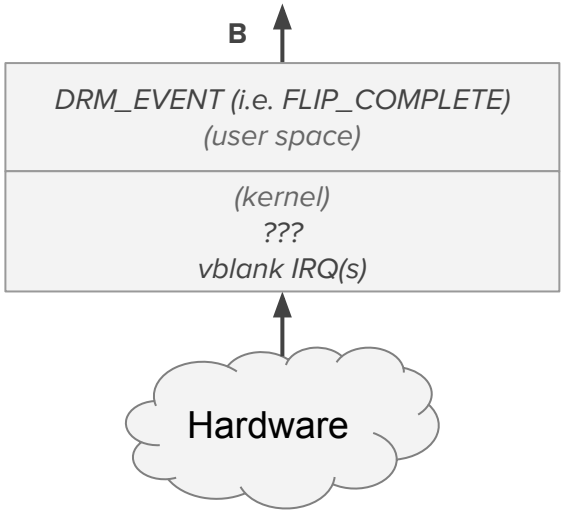
Late KMS Commits

Late KMS commits

- to optimize latency, schedule commit as close to page-flip as possible
  - i.e. execute GPU-warp as late as possible
  - can't commit plane properties while commit is already pending ➜ -EBUSY
- So what's the latest point we can commit for a frame?
  - short answer: *we don't know*

Late KMS Commits





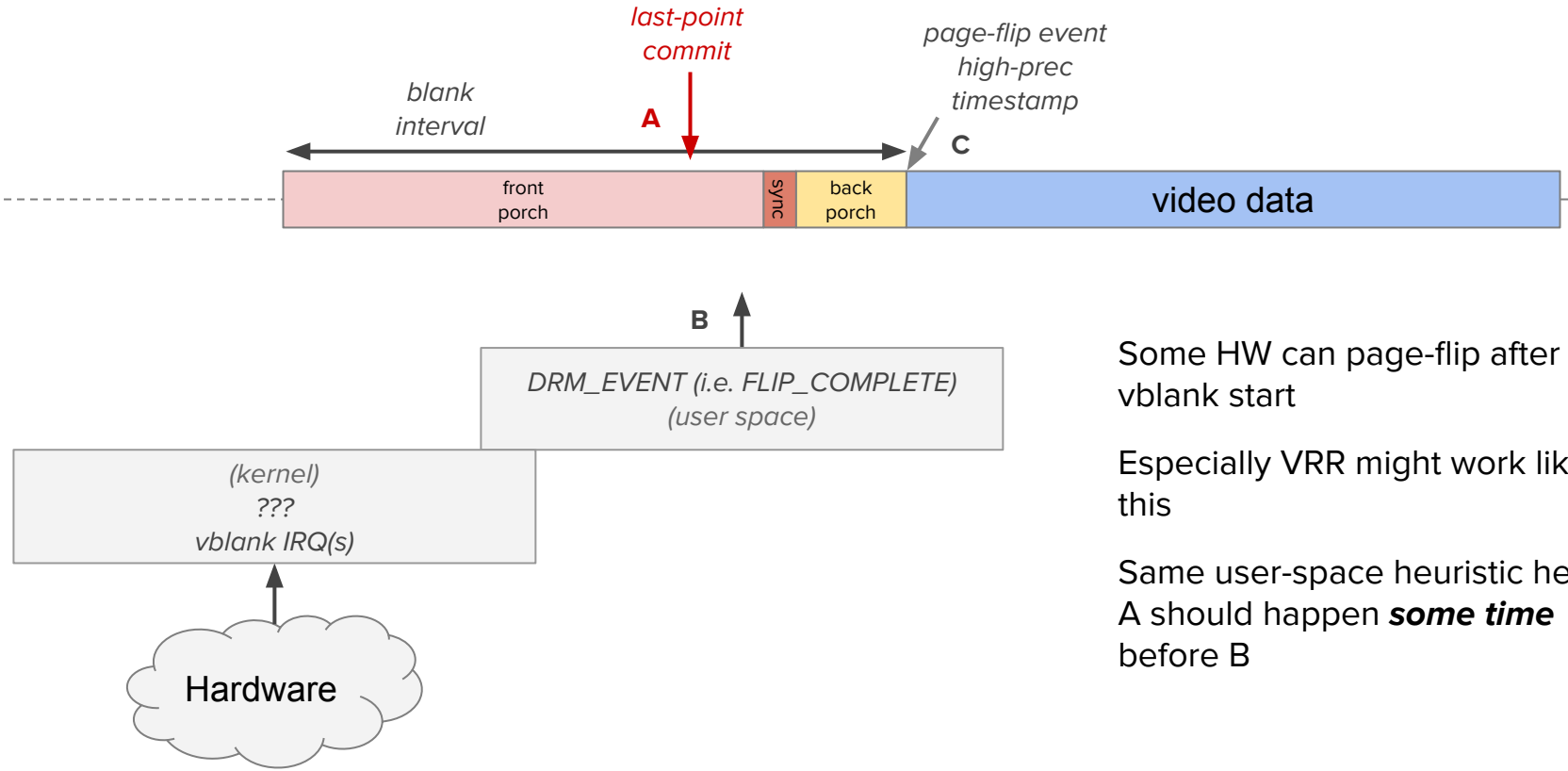
A should happen ***some time*** before B

A_{next} = C + refresh - *some_margin*

using C as the base: less subject to schedule-jitter (as it's HW corrected)

On our platform, a fixed *some_margin* seems reasonable

Late KMS Commits



Some HW can page-flip after vblank start

Especially VRR might work like this

Same user-space heuristic here: A should happen ***some time*** before B

Problems of Late KMS Commits

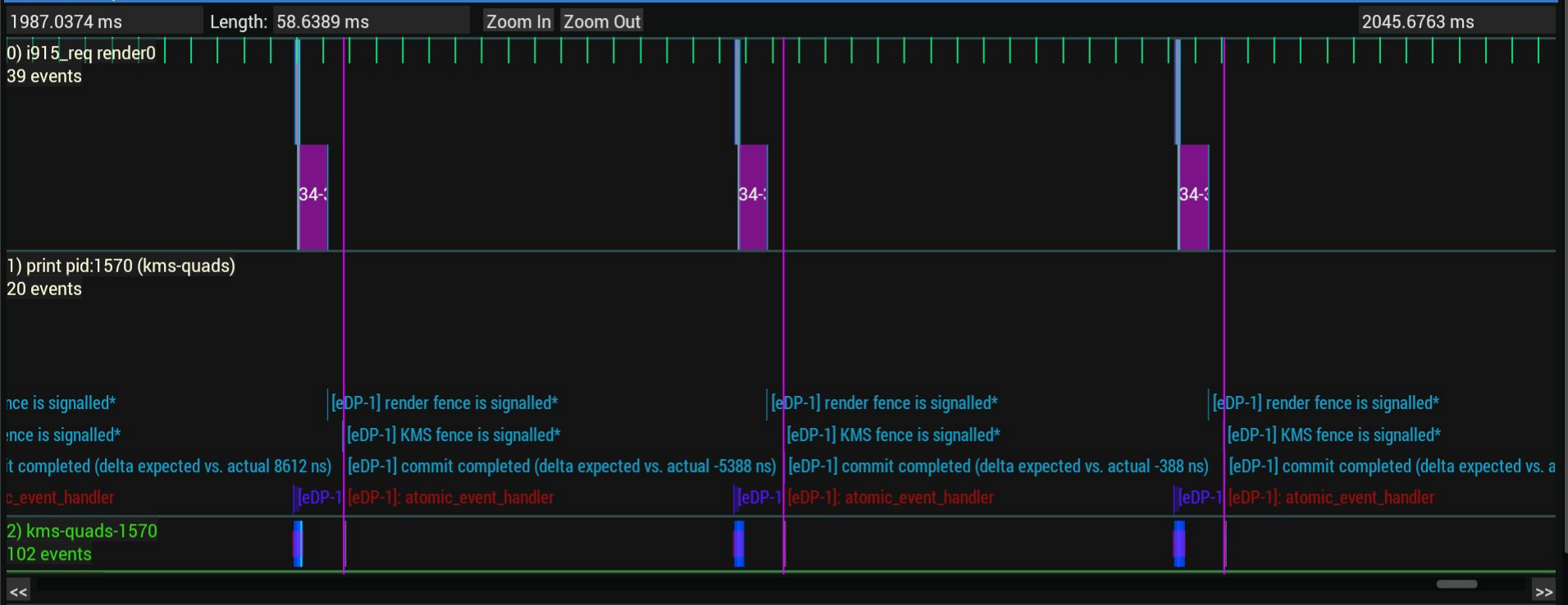
- *"some-margin"* to schedule ahead is not defined
- we only know margin was too small after the fact of having dropped a frame
  - a single dropped frame is *very* visible on AR headsets
- *Can we do any better?*

Visualizing high-prec timestamps (GPUVis)

- extended `drm_vblank_event` tracepoint to carry high-prec timestamp
- extended GPUVis to optionally visualize high-prec timestamp instead of trace-point timestamp
  - and to allow setting an external timestamp of a user-print event (e.g. visualize dma-fence timestamps)
  - both needs `echo mono > /sys/kernel/tracing/trace_clock`

File Options

Event Graph



Event List

Goto Event: 7700 Goto Time: 0.0

Event Filter: Clear Filter

162	16.391824 ms (+0.0026)	<idle>-0	2	drm_vblank_event		crtc=0 seq=19224 time=323821756521 high_prec=1 system=drm
163	16.393776 ms (+0.0020)	<idle>-0	2	sched_switch	0.7586 ms	prev_comm=swapper/2 prev_pid=0 prev_prio=120 prev_state=0 next_comm=kms-quads next
164	16.395306 ms (+0.0015)	<idle>-0	1	sched_switch	0.7601 ms	prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=0 next_comm=systemd next_pi
165	16.395943 ms (+0.0006)	kworker/u9:0-86	0	sched_switch	0.0067 ms	prev_comm=kworker/u9:0 prev_pid=86 prev_prio=100 prev_state=128 next_comm=kworker/0

File Options

Event Graph

2014.0442 ms Length: 3.5116 ms Zoom In Zoom Out 2017.5558 ms

0) i915_req render0
13 events

34-3698

34-3700 kms-quads-1570

1) print pid:1570 (kms-quads)
5 events

[eDP-1] render fence is signalled*

[eDP-1] KMS fence is signalled*

[eDP-1] commit completed (delta expected vs. actual -388 ns)

[eDP-1] repaint output

[eDP-1]: atomic_event_handler

2) kms-quads-1570
34 events

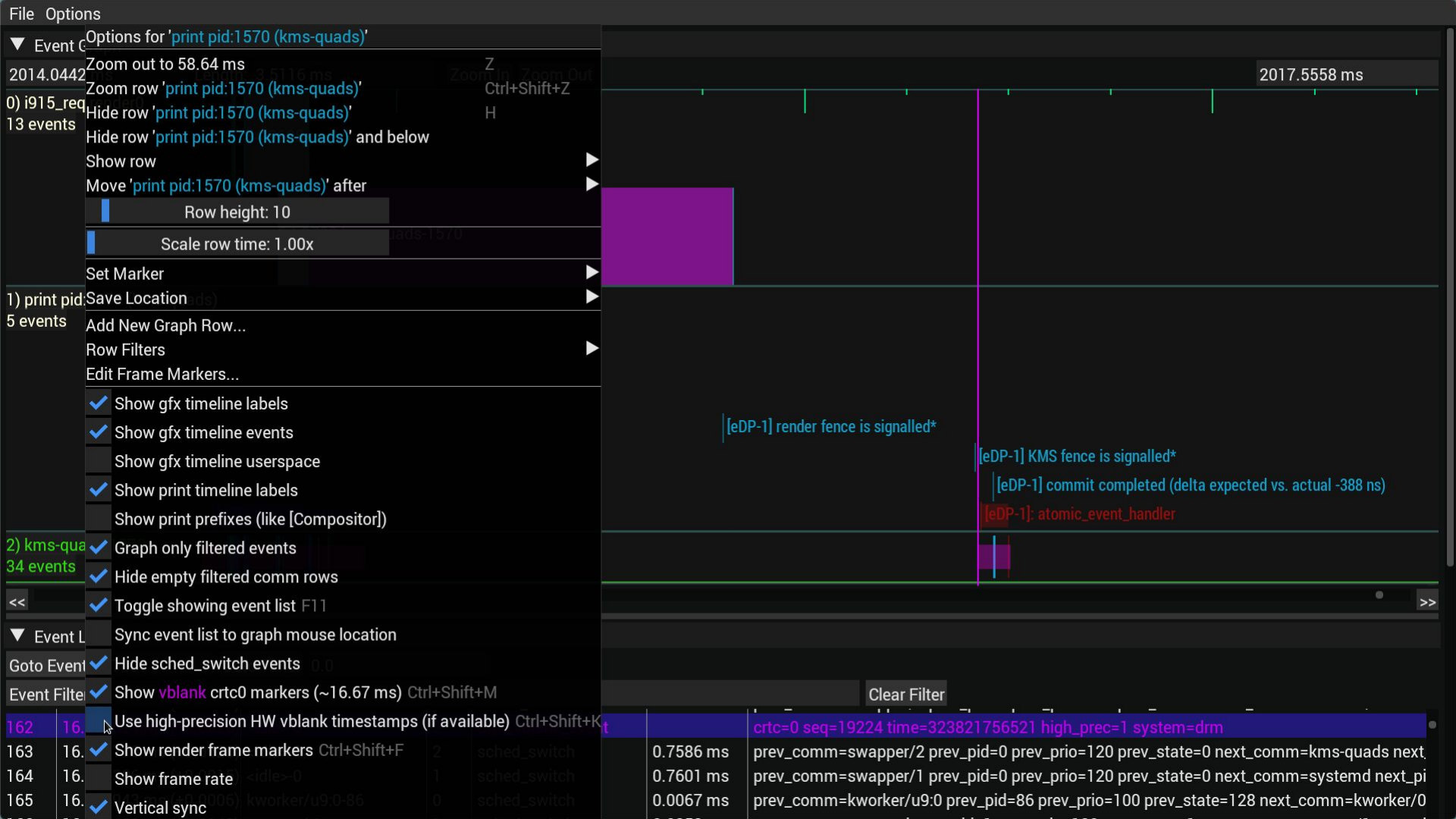
<< >>

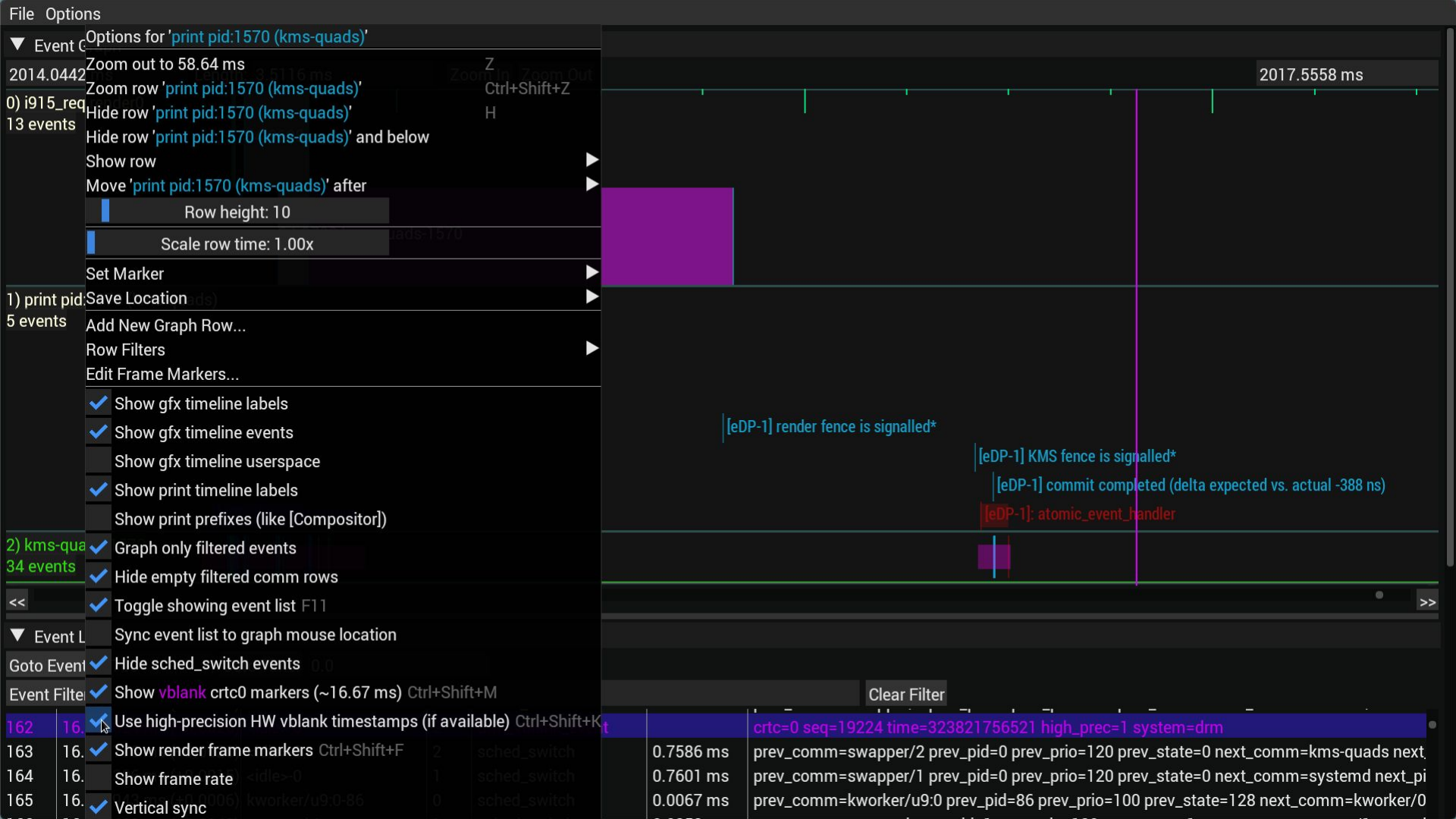
Event List

Goto Event: 7700 Goto Time: 0.0

Event Filter: Clear Filter

162	16.391824 ms (+0.0026)	<idle>-0	2	drm_vblank_event		crtc=0 seq=19224 time=323821756521 high_prec=1 system=drm
163	16.393776 ms (+0.0020)	<idle>-0	2	sched_switch	0.7586 ms	prev_comm=swapper/2 prev_pid=0 prev_prio=120 prev_state=0 next_comm=kms-quads next
164	16.395306 ms (+0.0015)	<idle>-0	1	sched_switch	0.7601 ms	prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=0 next_comm=systemd next_pi
165	16.395943 ms (+0.0006)	kworker/u9:0-86	0	sched_switch	0.0067 ms	prev_comm=kworker/u9:0 prev_pid=86 prev_prio=100 prev_state=128 next_comm=kworker/0


File Options
Options for 'print pid:1570 (kms-quads)'
Zoom out to 58.64 ms Z
Zoom row 'print pid:1570 (kms-quads)' Ctrl+Shift+Z
Hide row 'print pid:1570 (kms-quads)' H
Hide row 'print pid:1570 (kms-quads)' and below
Show row
Move 'print pid:1570 (kms-quads)' after
Row height: 10
Scale row time: 1.00x
Set Marker
Save Location
Add New Graph Row...
Row Filters
Edit Frame Markers...
Show gfx timeline labels
Show gfx timeline events
Show gfx timeline userspace
Show print timeline labels
Show print prefixes (like [Compositor])
Graph only filtered events
Hide empty filtered comm rows
Toggle showing event list F11
Sync event list to graph mouse location
Hide sched_switch events
Show vblank crtc0 markers (~16.67 ms) Ctrl+Shift+M
Use high-precision HW vblank timestamps (if available) Ctrl+Shift+K
Show render frame markers Ctrl+Shift+F
Show frame rate
Vertical sync
2017.5558 ms
[eDP-1] render fence is signalled*
[eDP-1] KMS fence is signalled*
[eDP-1] commit completed (delta expected vs. actual -388 ns)
[eDP-1]: atomic_event_handler
Clear Filter
crtc=0 seq=19224 time=323821756521 high_prec=1 system=drm
0.7586 ms prev_comm=swapper/2 prev_pid=0 prev_prio=120 prev_state=0 next_comm=kms-quads next
0.7601 ms prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=0 next_comm=systemd next_pi
0.0067 ms prev_comm=kworker/u9:0 prev_pid=86 prev_prio=100 prev_state=128 next_comm=kworker/0


File Options
Options for 'print pid:1570 (kms-quads)'
Zoom out to 58.64 ms Z
Zoom row 'print pid:1570 (kms-quads)' Ctrl+Shift+Z
Hide row 'print pid:1570 (kms-quads)' H
Hide row 'print pid:1570 (kms-quads)' and below
Show row
Move 'print pid:1570 (kms-quads)' after
Row height: 10
Scale row time: 1.00x
Set Marker
Save Location
Add New Graph Row...
Row Filters
Edit Frame Markers...
Show gfx timeline labels
Show gfx timeline events
Show gfx timeline userspace
Show print timeline labels
Show print prefixes (like [Compositor])
Graph only filtered events
Hide empty filtered comm rows
Toggle showing event list F11
Sync event list to graph mouse location
Hide sched_switch events
Show vblank crtc0 markers (~16.67 ms) Ctrl+Shift+M
Use high-precision HW vblank timestamps (if available) Ctrl+Shift+K
Show render frame markers Ctrl+Shift+F
Show frame rate
Vertical sync
2017.5558 ms
[eDP-1] render fence is signalled*
[eDP-1] KMS fence is signalled*
[eDP-1] commit completed (delta expected vs. actual -388 ns)
[eDP-1]: atomic_event_handler
Clear Filter
crtc=0 seq=19224 time=323821756521 high_prec=1 system=drm
0.7586 ms prev_comm=swapper/2 prev_pid=0 prev_prio=120 prev_state=0 next_comm=kms-quads next
0.7601 ms prev_comm=swapper/1 prev_pid=0 prev_prio=120 prev_state=0 next_comm=systemd next_pi
0.0067 ms prev_comm=kworker/u9:0 prev_pid=86 prev_prio=100 prev_state=128 next_comm=kworker/0

kms-quads

kms-quads

- *straightforward and well-documented KMS example*
  - Written by Daniel Stone Collabora based on DAQRI's requirements
  - boosted DAQRI compositor development
  - Updated with features and lessons-learned from DAQRI compositor
  - MIT license, get it here)
- shows (and explains!) how to ...
  - use Atomic KMS for page flipping
  - draw into GBM buffers with GL Core/ES3/ES2 (+EGL setup)
    - use *drm format modifiers* to use compressed/tiled surfaces (if available)
  - calculate presentation timing and schedule rendering
    - with a low-latency use-case in mind
  - synchronize with dma-fence via EGLSync
  - use logind (optionally) to safely switch VT and open devices

Upstream collaboration

- bring some GLES extensions to GL Core
  - use EGL_sync in GL core command stream (GL_EXT_EGL_sync)
  - GL_MESA_framebuffer_flip_y
- i915
  - discovered regression of alpha compositing with CCS compressed surfaces (5.0 regression), fixed upstream by Intel
  - Async KMS pageflip task should run on high-prio worker queue, upstream patch submitted by Intel

Conclusions

- KMS was a good choice for DAQRI to implement a lightweight AR compositor
- Timestamp semantics of KMS events are undefined in user space and should be properly defined
- Tightly scheduling late commits from user space is tricky

Special Thanks

X foundation for sponsoring my trip

Daniel Stone, Scott Anderson (Collabora)

Daniel Vetter et al. (Intel)

Q & A


DAQRI