Video Codecs and the Virtual Stateless Decoder Driver (visl)

Daniel Almeida
Consultant Software Engineer, Collabora
daniel.almeida@collabora.com

COLLABORA

But first, who am I?

Who am I?

- Was a LKMP mentee, joined Collabora in 2021
- I mostly straddle the line between GStreamer and the kernel
- Most of my contributions are multimedia related
- Recently, I have been working on video codecs in Rust full-time
- I also have experience with other codec APIs, like VA-API

COLLABORA

Let’s get started :)

COLLABORATA

What are video codecs?

Video codecs explained (really quickly)

- Raw video data is simply unfeasible most of the time

Video codecs explained (really quickly)

- Raw video data is simply unfeasible most of the time
- Video signals are full of exploitable redundancies

Video codecs explained (really quickly)

- Raw video data is simply unfeasible most of the time
- Video signals are full of exploitable redundancies
- Video codecs **compress/decompress** raw video by capitalizing on this

Video codecs explained (really quickly)

- Raw video data is simply unfeasible most of the time
- Video signals are full of exploitable redundancies
- Video codecs **compress/decompress** raw video by capitalizing on this
- Most of the time, this process is lossy, i.e. not perfectly reversible

Video codecs explained (really quickly)

- The objective is to arrive at a **passable approximation**
- For a given **bitrate** and **power envelope**.

Video codecs explained (really quickly)

- The objective is to arrive at a **passable approximation**
- For a given **bitrate** and **power envelope**
- The name codec comes from en**co**der/**dec**oder
- Usually only **decoding** is **standardized** on a specification
- Encoders are free to innovate, so long as it decodes

What are these redundancies?

Compression techniques

- Spatial: pixels close in location tend to be similar
- Temporal: adjacent frames tend to be similar
- Chroma subsampling: eyes more sensitive to luma
- Quantization
- Entropy coding
- AI?

Can we make this faster?

Hardware accelerators

- Tend to be faster
- More power efficient
- Free up the main CPU
- Less flexible (only a subset of the codec, usually)
- **Need driver support and an API to communicate**

We use APIs to communicate with the underlying driver and hardware accelerator

COLLABORA

Video Codec APIs

(Some) Video Codec APIs

- DXVA (DirectX Video Acceleration: Windows, Xbox)
- VA-API (created by Intel, primarily for Unix-like systems)
- NVENC/NVDEC (NVIDIA GPUs)
- Vulkan Video (well, Vulkan, for video codecs)
- **V4L2 (Video4Linux2)**
- ...and so on.

Why so many?

- Some Video Codec APIs are more suitable for some platforms than others.
- Some are vendor-specific (NVENC/DEC, for example)
- Some abstract over video codec hardware found within GPUs (e.g.: VA-API, NVENC/DEC)
- Some focus on video codec hardware embedded within SoCs

COLLABORA

How is the kernel related to this?

Video Codec Drivers

- Some APIs employ a **user space** driver: e.g.: VA-API
  - **Client program uses an API to talk to user space driver**
  - Driver builds a set of command buffers and send these to the kernel
  - Kernel takes care of submitting to the GPU

Video Codec Drivers

- Some APIs employ a **kernel space** driver: e.g.: V4L2
  - **Client program uses API to talk to the kernel**
  - Kernel takes care of programming the hardware
  - This API is known as a uAPI, i.e. **user space API**

This talk is about the V4L2 Codec APIs and the visl driver

V4L2 is a framework/API for various multimedia devices

This includes video codecs (as of somewhat recently)

Inside a codec bitstream

Inside the bitstream

- Metadata
  - Controls the decoding process
  - May persist between frames or relate to a single frame
  - e.g.: VPS/SPS/PPS/Frame headers, etc
- Slice and/or Tile data
  - Actual compressed data

V4L2 Video Codec API types

- Stateful
  - Hardware parses bitstream itself
  - Keeps track of bitstream metadata (**state**ful)
- Stateless
  - Client program parses bitstream (in software)
  - Uses bitstream metadata to program hardware

COLLABORA

Stateless hardware tend to be simpler, but it needs more software to drive it

Codec uAPI

- Stateless APIs also need to provide a way to send bitstream metadata to the kernel together with the bitstream itself
- This metadata is extracted when **parsing** the stream **in software**.
- The API to pass the codec-specific bitstream metadata is known as the **codec uAPI** (e.g.: the VP9 uAPI, the AV1 uAPI)

Codec uAPIs

- Collabora has been steadily merging support for the major codecs in industry

Codec uAPIs

- This includes:
  - H.264/AVC (proprietary)
  - H.265/HEVC (proprietary)
  - VP9 (open, royalty-free)
  - AV1 (open, royalty-free, state-of-the-art)

COLLABORA

Let’s recap what we know so far

Recap

- A video codec compresses and decompresses video, we need this to make video data tractable
- Video codecs benefit from hardware acceleration
- We use APIs to talk to the accelerator
- This presentation is specifically about the V4L2 codec APIs and the visl driver

Recap

- V4L2 APIs come in Stateful **and** Stateless flavors
- For the **stateful** API, we only send the bitstream through V4L2, the hardware does the rest.
- For the **stateless** API, we must **also** send the bitstream metadata through the so-called **codec uAPI**
- Collabora has been merging these uAPIs into the Linux kernel, some of the codecs are proprietary, some are Open Source

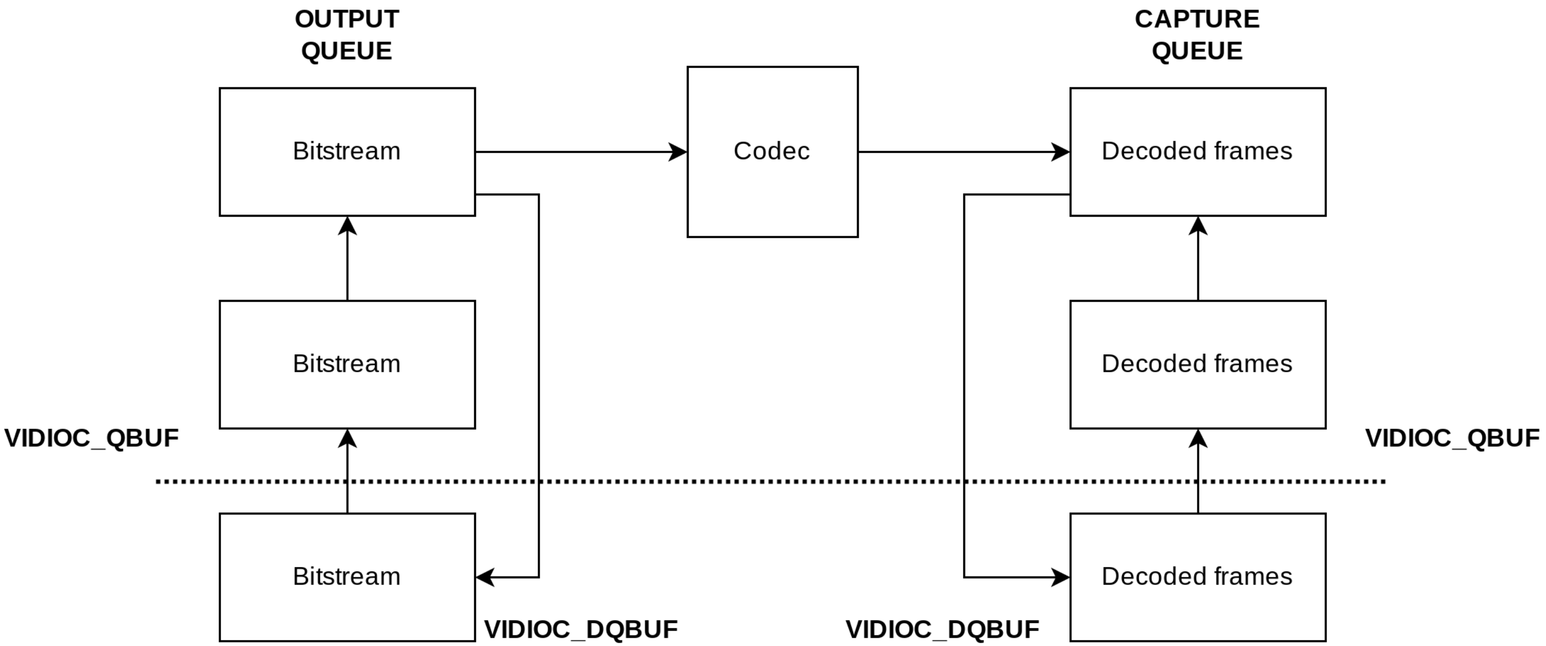
COLLABORA

What is visl?

visl

- A **virtual** stateless decoder driver
- It does **not** drive a real accelerator
- Userspace can talk to it through the **codec uAPIs** we have discussed
- Implements a decode loop like any other codec driver

Decode loop


OUTPUT
QUEUE
CAPTURE
QUEUE
Bitstream
Codec
Decoded frames
Bitstream
Decoded frames
VIDIOC_QBUF
VIDIOC_QBUF
Bitstream
Decoded frames
VIDIOC_DQBUF
VIDIOC_DQBUF

Codec uAPIs supported by visl

- Vp8
- Vp9
- Mpeg2
- FWHT
- H.264/AVC
- H.265/HEVC

What is traced by visl

- State of the queues
- State of the decoded picture buffer (DPB)
- The bitstream metadata (submitted through V4L2 controls)
- The slice/tile data submitted in the OUTPUT buffers
- Note: other APIs have similar tracing mechanisms: e.g.: VA_TRACE

Why should we bother with a virtual driver?

visl as a development aid

- Helps **test** your userspace code even if you do not have the hardware
- Helps you **prototype** new codec uAPIs
- You can run a **working** userspace implementation against visl to trace it
- You can then use the traces to develop the code for another userspace application

How is visl different from a real driver?

- Real drivers will use the metadata transmitted through the codec uAPI to program the underlying device
- visl uses the metadata to program the v4l2 test pattern generator instead
- visl also uses the metadata to dump it through various means
- **Most importantly: visl does not decode video at all**

How is visl different from vicodec?

- vicodec is another driver entirely
- vicodec can actually encode and decode video
- It uses its own video coding standard, FWHT
- FWHT is an “academic” codec, not used in industry
- vicodec also has stateful support

If you understand visl...

COLLABORA

...you understand how codec drivers work!

Example of real codec drivers

- rkvdec (Rockchip video engine)

  drivers/staging/media/rkvdec

- Hantro (video IP from Verisillicon, present in a number of SoCs)

  drivers/media/platform/verisilicon/

- Cedrus (reverse-engineered from Allwinner SoCs)

  drivers/staging/media/sunxi

How do I run visl?

How do I run visl?

- Install a new-ish version of Gstreamer (1.18, 2020)
- Modprobe visl
- Run any pipeline using a v4l2 stateless decoder element, e.g.:
  - gst-launch-1.0 filesrc location=<some video file> ! parsebin ! v4l2slh264dec ! filesink location=<some output file>

What should I expect?

- GStreamer will start “playing” your file
- Its filesink element will write the “decoded” data into a file
- You can inspect this file with, e.g.: YUView to access the frames generated by visl
- The frames will contain a lot of debug and tracing information
- You can use ftrace as well

COLLABORA

Don’t forget to play with the different options when loading the module

COLLABORA

Ok, and finally, why should you care?

The exciting world of codecs

- Cisco: by 2022, 82% of all consumer internet traffic will be video data
- Improving the Linux multimedia stack makes the OS more appealing as a whole
- It is intellectually challenging and rewarding
- V4L2 can use more contributors that can grow into maintainers in the future

Thank you!

We are hiring

col.la/careers