The case for a virtual Rust stateless codec driver

Daniel Almeida

FOSDEM’24
Video codecs explained (quickly)

- Raw video is *huge*
- Video signals contain exploitable redundancies
- Video codecs compress and decompress video by capitalizing on this
Video codecs explained (quickly)

- Usually this process is lossy
- The objective is to arrive at a passable approximation
- At a given bitrate and power envelope
But..we want things to be fast and cool...
Hardware accelerators

- Tend to be faster, more power efficient
- Frees up the main CPU
- Less flexible (usually only a few profiles supported)
- Need driver support and an API to communicate with userland
To understand codec drivers, we must look inside the bitstream.
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, etc.

- **Slice and/or tile data**
  - Actual compressed data
And how can we talk to these devices?
V4L2 Codec API types

• Stateful
  - Hardware parses bitstream itself
  - Hardware keeps track of bitstream metadata

• Stateless
  - Userland parses stream
  - Userland sends metadata to driver
  - Driver uses metadata to program the device
We have a virtual stateless driver :)
visl (Virtual Stateless Decoder Driver)

- A virtual stateless driver
- Pretends it’s decoding data
- Instead gives the programmer a lot of debug information
What is visl good for?

- Developing new userland (based on a working one)
- Fixing bugs on existing userland implementations
- Testing userland when hardware is not available
- Prototyping new codec APIs
Great! What’s the problem then?
What the metadata looks like
There are pages and more pages of this...
Not only this can be *very* complex but...
...we index into arrays and use loop variables from data read directly from the bitstream
The case for Rust

• We are handling a *lot* of metadata per frame
• This metadata is highly structured/complex
• The meaning of fields may change based on the value of other fields
• You may have to juggle multiple versions of a given set of metadata (e.g.: only one is active)
This problem is exacerbated in real drivers :/
The case for Rust

- You have to carefully read the specs to make the right use of said metadata
- Otherwise you may wrongly program the device
- This may hang the device or worse:
- This can change the decoding process in unknown ways
The case for Rust

- Clearly there’s value to having Rust in codec drivers
- A virtual driver is the perfect candidate to experiment
- We can make it even simpler (no debug information)
- If we can make a virtual driver work with Rust, we will have the foundations for real drivers
What do we have so far?
What we have so far

- Abstractions for some V4L2 data types
- A *very* thin videobuf2 abstraction (you can spawn a queue)
- Abstractions for some V4L2 ioctlS
- The necessary code to get the driver to probe
- A sample module that merely prints to the terminal
What do we need?
Rust abstractions we need

- V4L2 controls (to get the metadata)
- Media controller (for V4L2 Request support)
- mem2mem (for device_run() and friends)
- More ioctl support
We still need a green light from maintainers
Feedback and roadblocks

- V4L2 can’t keep up with the workload as is
- Not enough reviewers and maintainers
- Long-standing issues with some C frameworks
- Fear of breaking C code
- Who is going to maintain the Rust layer?
We want to unblock this effort
Which is why we are proposing a virtual driver in Rust
Summary

- Stateless codec drivers take in a lot of untrusted data from userland
- We can minimize the attack surface with Rust
- The visl virtual driver is a prime candidate for experimentation
Thank you!

We are hiring - col.la/careers