Kernel CAM
Rethinking the kernel camera framework

Sergey Senozhatsky, Fei Shao, Yunke Cao, Tomasz Figa, Ricardo Ribalda

Paris, 1st June 2022
Once upon a time
Operating system (for driver developers)

Program -> My framework -> Driver -> Hardware

open(/dev/my_device)  
read(fd)  
ioctl(fd,IOC_,...)  
my_init()  
my_alloc()  
my_read_from()  
inb()  
outb()
Operating system (for driver developers)
Cameras are special
Great diversity of output

- There are more than 200 video formats
  - `grep -c "^#define V4L2_PIX_FMT" include/uapi/linux/videodev2.h` -> 202!
Multiple ways to do the same thing

- Image is too dark?
  - Exposure time
  - Analog Gain
  - Digital Gain

- Do you want a 640x480 image from a 24Mpix (6000*400)?:
  - Cropping
  - Binning
  - Interpolate
Do it fast or don’t do it

- Big data rates
  - 1080p RGB32@60fps → 474 MiB/sec!
- Low latency
- Most of the time handled by other hardware
  - GPU
  - TPU
  - CODEC
Cameras drive the consumer market

- Work from home/anywhere
- E-learning
- Deciding factor for purchase
Cameras in Linux today
Output can be converted via software to 4 standard formats.
- The driver makes most of the decisions for us.
- Hardware produces decent images by default.
Video4Linux 2 (media controller)
Video4Linux 2 (post media controller)

- Software stack needs to know about the internals of the hardware.
- All the decisions are taken by the software stack.
- A lot of Image tuning before we can use the images.
Libcamera
Secret Sauce
Secret Sauce
Secret Sauce
Secret Sauce
Recap

Agnostic Userspace

Kernel

Hardware

V4L2

Aware Userspace

Kernel

Hardware

Media Controller
Kernel
CAM
What is CAM?

- New kernel subsystem.
- No media abstractions.
- Fluid collaboration between userspace and other subsystems.
- Two components:
  - Entities
  - Operations
KCAM components: Entities

- Organized in a tree, based on the hardware architecture.
- Single register-set.
- Can throw events.
KCAM components: Entities

- Organized in a tree, based on the hardware architecture.
- Single register-set.
- Can throw events.
KCAM components: Operations

- Read and/or write to an entity
  - regmap
  - parameter buffers
- Can depend on:
  - Events
  - Other operation
  - Fence
- Can create a fence
KCAM components: Operations

Operation List
- Configure Sensor
- Trigger image
- Init ISP
- Start DMA

ioctl

ready

fence_in

fence_out

Circular buffer

signal
#define CAM_MAX_DEPENDENCIES 8

struct cam_operation_add {
    __u32 id;
    __u32 fence_out;
    __u32 flags;
    /*
     * Pre-execution dependencies list and dependency execution mode
     */
    __u32 mode;
    struct cam_dependency deps[CAM_MAX_DEPENDENCIES];
    /*
     * Execution context specific data (if any)
     */
    __u64 delay_ns;
    __u64 rd_wr_list;
    __u32 entity;
} __attribute__((packed));

enum cam_dependency_type {
    CAM_DEPENDENCY_NONE,
    CAM_DEPENDENCY_OP,
    CAM_DEPENDENCY_EVENT,
    CAM_DEPENDENCY_FENCE_IN,
};

struct cam_dependency {
    __u32 type;
    __u32 id;
} __attribute__((packed));

enum cam_dependency_mode {
    CAM_DEPENDENCY_WEAK_ORDER,
    CAM_DEPENDENCY_STRICT_ORDER,
};
Testing (lots of)

- Software testing:
  - kunit
  - libkc
  - vcam
  - error injection
- Hardware testing
  - ChromeOS test infra
  - human test
Comparison

- Limitless functionality
- Fast upstream
- Small drivers
- Operations vs Streams
Governance

- There must be an open source stack before a driver is merged.
  - Similar to DRM model.
- Compliance-test with minimum requirements.
Future
Benchmark the stack

Clients
- Android Camera Framework
- Chrome Video Capture Device
- VMs

Camera Service
- Secret Sauce

Userspace
- Chrome OS Video Capture Device
- VMs

Kernel
- Generic V4L2 Frameworks
- SoC ISP Drivers
- Sensor/Actuator/Flash/EEPROM Drivers

Hardware
- SoC ISP
- Sensor/Actuator/Flash/EEPROM ROM

Android Camera Framework
- Sors
- Svers
- SoC ISP
- Drivers

Camera Service
- Chrome OS Video Capture Device
- VMs
- Libcamera - cam

Clients
- Android Camera Framework
- Chrome Video Capture Device
- VMs

Generic V4L2 Frameworks
- CAM
- CAM drivers
- SoC ISP
- Sensor/Actuator/Flash/EEPROM ROM

Libcamera - cam
- Secret Sauce
The Chromium Projects

Upstream First

Abstract

This document outlines a process for managing the flow of kernel source patches between Chromium OS and hardware partners.

Goals

Our goal with the "Upstream First" policy is to eliminate kernel version fragmentation.

We know that hardware partners support other devices as well as Chromium OS-based devices. So rather than having everyone target a single kernel, we hope to avoid duplication of effort and get everyone (our hardware partners, too) involved.

Accepting patches

A device driver patch must be accepted upstream before it can be accepted into the Chromium OS kernel. Ideally, it should be accepted into the Linux kernel first.

The following list shows options for getting a patch accepted into the Chromium OS kernel. We encourage all partners to:

1. The patch is accepted into Linus Torvalds’s upstream tree.
2. The patch is accepted into the subsystem maintainer tree (such as Dave Milich’s netdev tree).
3. The patch is accepted into the device maintainer tree (such as Samsung’s or nvidia’s) and a pull request is created.

After the patch is accepted into an upstream tree, Google will cherry-pick it into the Chromium OS kernel. If the patch is acceptable:

Exemptions are granted on a case-by-case basis.
Contributing
How to contribute

- Show me the code!
  - https://chromium.googlesource.com/chromiumos/third_party/kernel/+/refs/heads/kcam
- Mailing List
  - kernel-cam@chromium.org
- Bugs:
  - WIP
- Join the team
  - https://crosjobs.page.link/event
Thank you!
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event

42
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event

47
Thank you!
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

50

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!
Thank you!

ribalda@chromium.org  
https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

ribalda@chromium.org

https://crosjobs.page.link/event
Thank you!

0xffffffff