
SACC
2021 中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2021
数字转型 架构重塑
IT168.com
ChinaUnix.net
ITPUB
云上会议 网络直播 | 2021.5.20-2021.5.22

Challenges and Best Practices of FFmpeg/Gstreamer

Xu Guangxin

Zhao Juan

Intel

Out Line

- FFmpeg/Gstreamer Introduction
- Build Your Optimized Pipeline
- DevOps with FFmpeg/Gstreamer
- Embracing new technologies

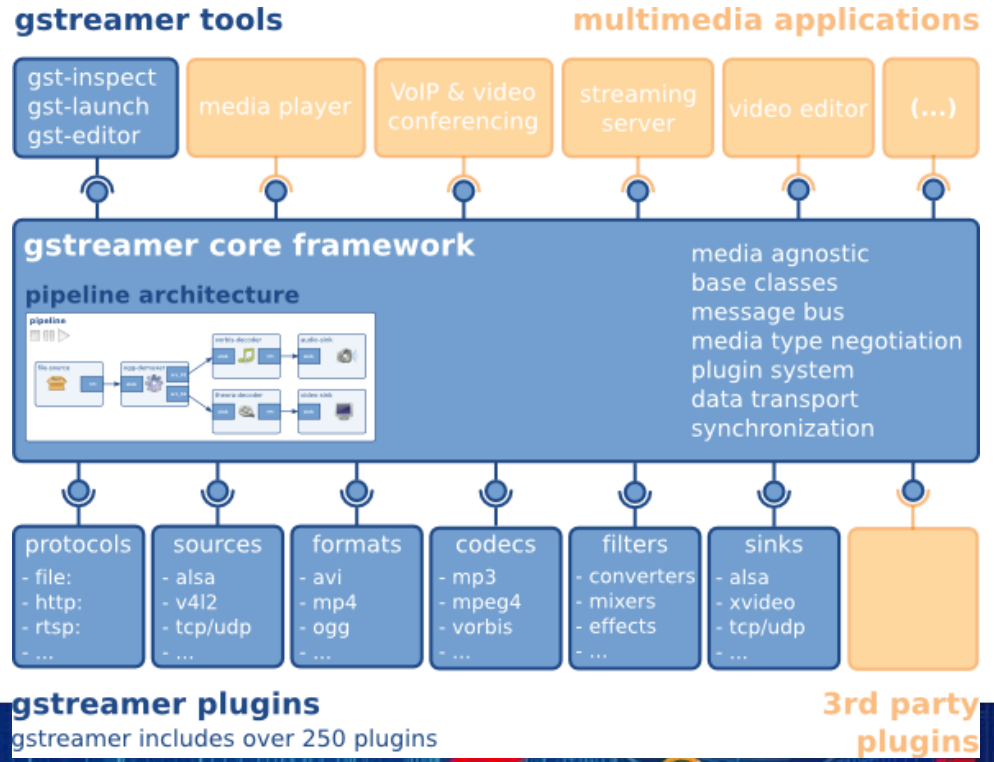
FFmpeg/Gstreamer Introduction

What is FFmpeg

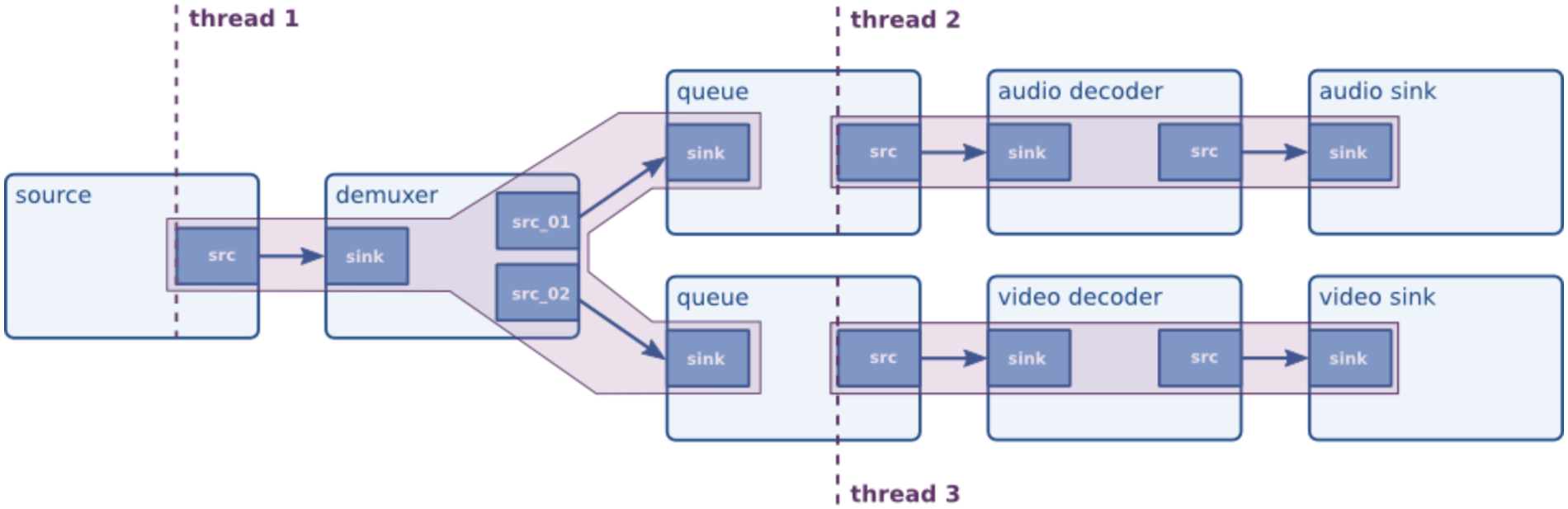
FFmpeg is the leading multimedia framework, able to **decode**, **encode**, **transcode**, **mux**, **demux**, **stream**, **filter** and **play** pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations.

What is gstreamer

GStreamer provides
- an API for multimedia applications
- a plugin architecture
- a pipeline architecture
- a mechanism for media type handling/negotiation
- a mechanism for synchronization
- over 250 plug-ins providing more than 1000 elements
- a set of tools



Typical GStreamer pipeline



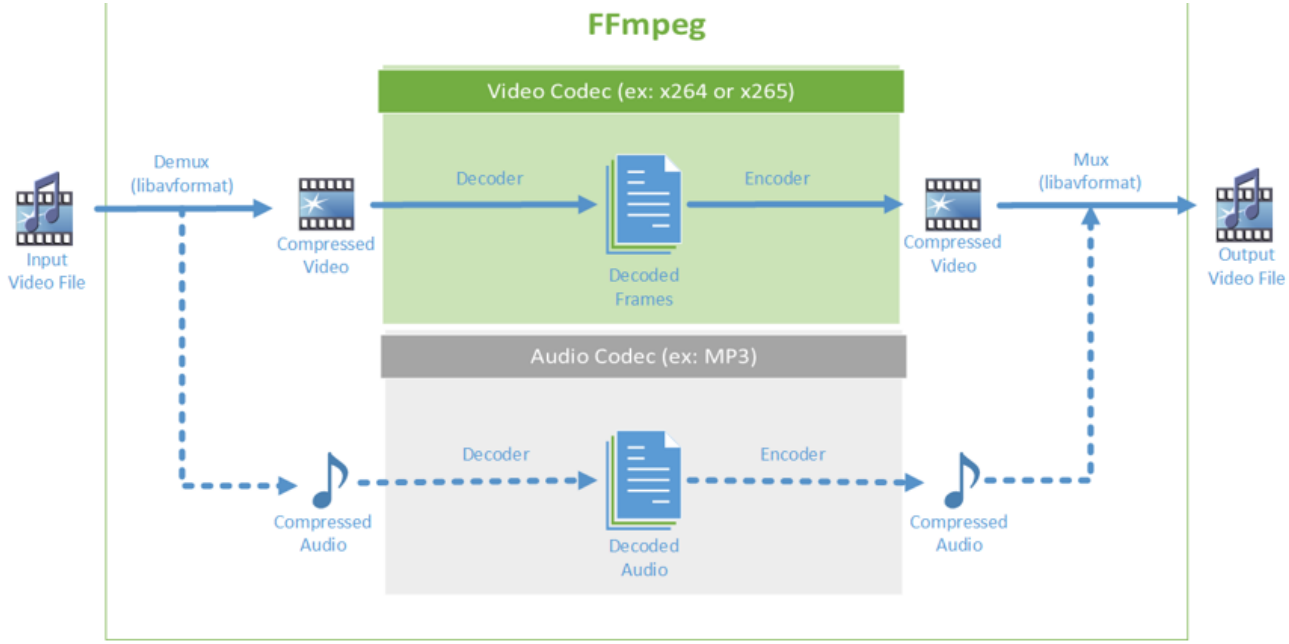
FFmpeg/GStreamer Ecosystem

- Adopted opensource software
  - x264,x265, av1d,av1e, SVTAV1…
- Adopted by opensource software
  - Chrome, Firefox(FFmpeg)
  - Handbrake, OBS (FFmpeg)
  - VLC,Mplayer (FFmpeg)
  - OpenCV (FFmpeg, GStreamer)
  - Open WebRTC Toolkit (FFmpeg,GStreamer)
  - …

Build Your Optimized Pipeline

Software codec pipeline optimization

- Compile stage optimization
  - Enable O3, and AVX
  - Using better compiler, eg ICC
  - Using dynamic library vs static library
- Profile stage
  - Test video clips
  - CPU utilization
  - Hotspots list
  - Memory Access analyze
  - Memory Usage analyze
- Hotspot optimization
  - Handwriting assembly code.



X264 transcoding process

Hardware Acceleration

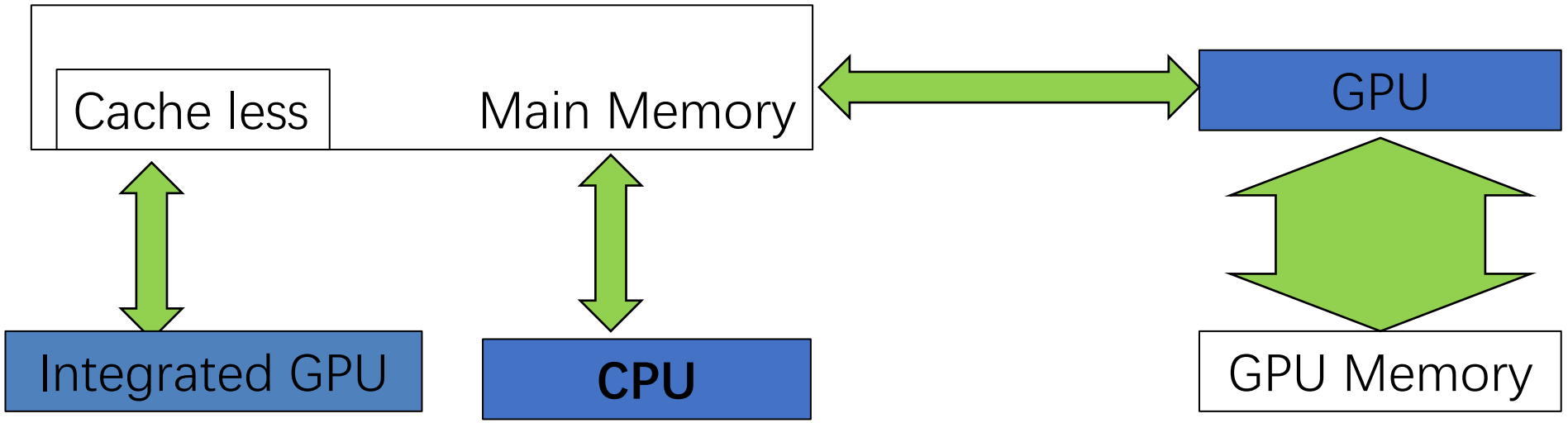
- Step 1. Know your hardware capabilities
- Step 2. Select best software package
- Step 3. Choose FFmpeg and GStreamer plugins

Pipeline Level Optimization

- Know you bottleneck
  - CPU
  - Different GPU engines
  - Memory bandwidth
  - Synchronization overhead
- Latency profile
  - First frame to show out
- The balance between hardware resource and Latency
  - Distributed transcoding

Take an example

- Why accessing decode output is too slow?
  - Integrated graphics card uses tiled memory to store decode output, there's no cache for read.
  - Discrete graphics card has a dedicated memory, CPU need use PCI bus to read



Why accessing decode output is too slow

- Solutions:
  - Do not read the GPU memory.
  - Use OpenCL/OpenGL/Vulkan to access hardware decoder output.
    - eg: background aware subtitles
  - Copy memory to a linear memory using vpp.
    - hw decoder + sw encoder
  - Use AVX2/AVX512 to read the cache less memory.

Best known method to build a high performance HW pipeline

- Pay attention to resource bottle neck.
  - Shift workload to other threads, if one core is full.
  - Shift vpp workload to EU, if fix function is fully loaded.
- Do not wait for GPU job done.

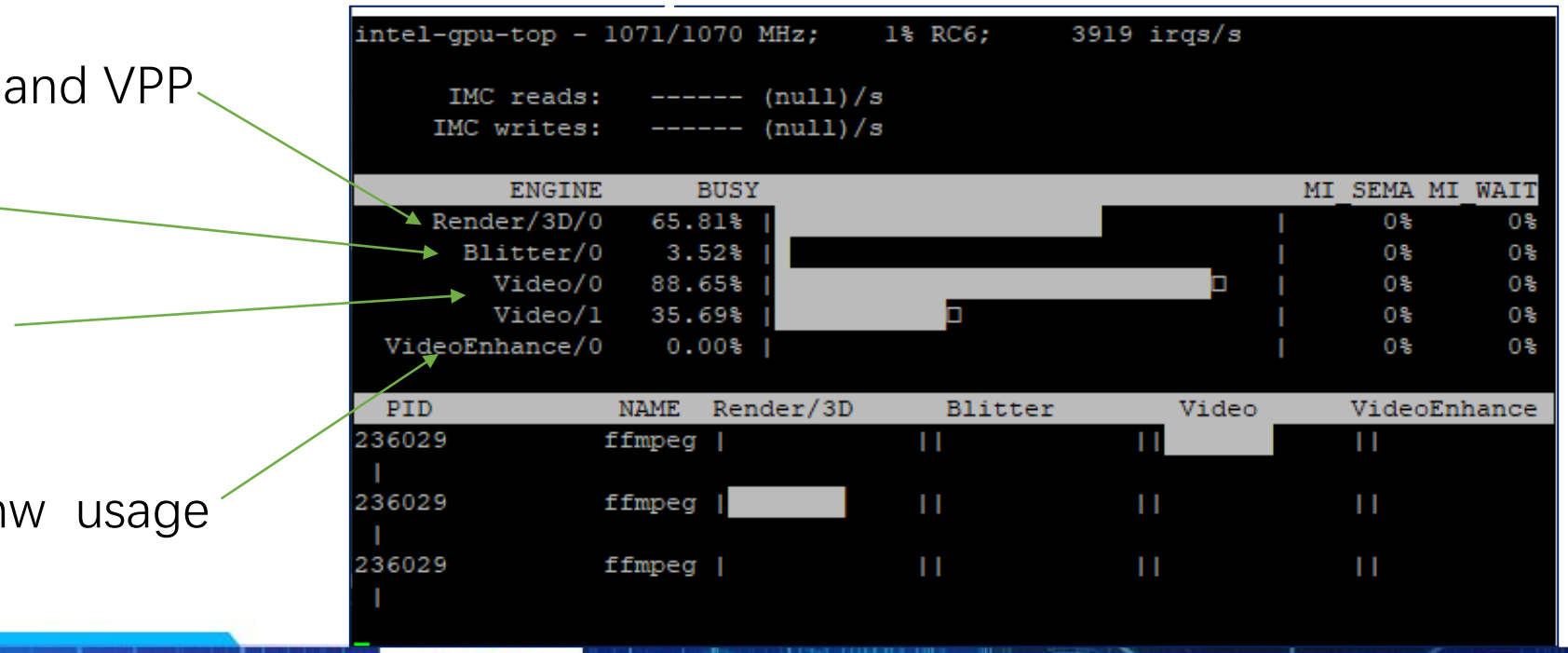
GPU frequency

Render/EU used by 3D, OpenCL and VPP

Blitter to copy data from/to GPU

Decoder and encoder hw usage

Fix function Video Post Process hw usage



DevOps with FFmpeg/Gstreamer

Upstream & Downstream

- Upstream
  - https://git.ffmpeg.org/gitweb/ffmpeg.git
  - https://gitlab.freedesktop.org/gstreamer
- Downstream
  - Any release or commit you forked to build your service

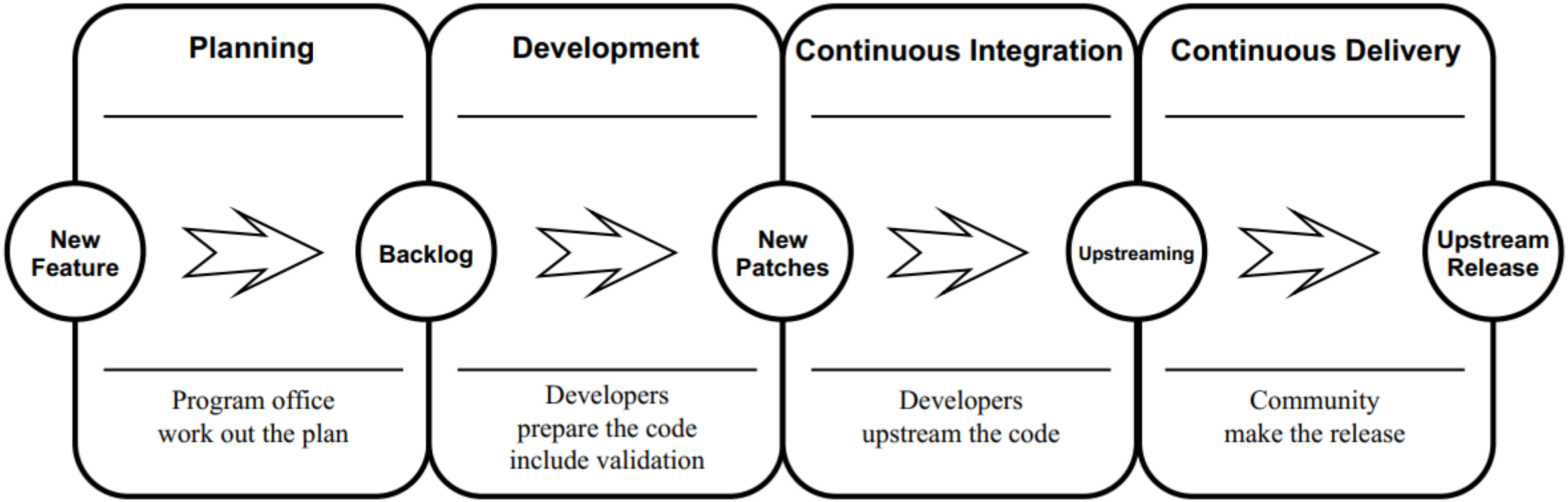
Why upstream

- If you use the upstream, you will get
  - More bug fix
  - Optimized performance
  - More features
- If you maintain a local fork, you will
  - not get upstream fix, back port fixes may not possible.
  - queue too many local patches, they are hard to upstream.

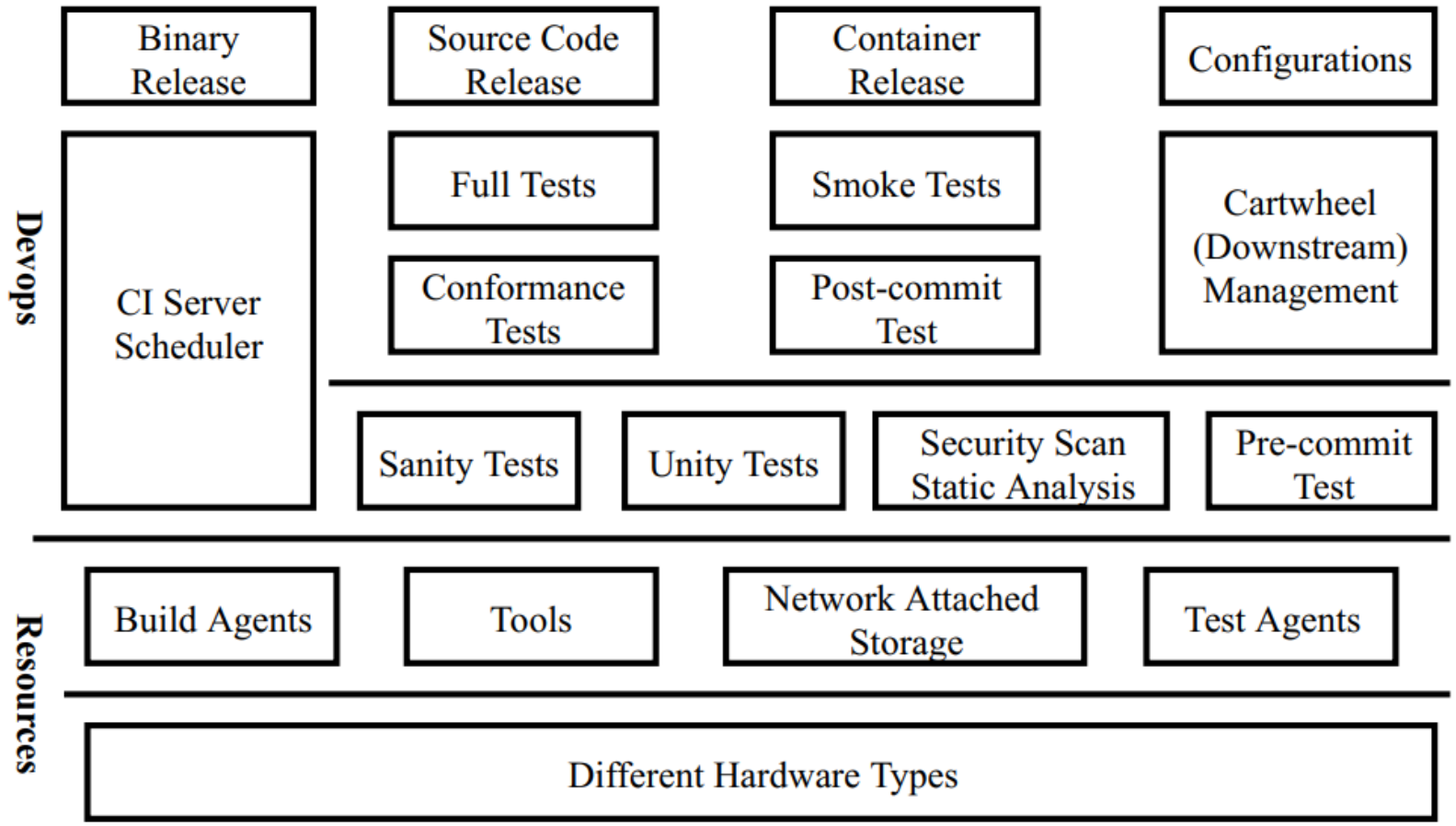
Pain points for downstream FFmpeg/GStreamer development

- There are 50+ commits every week for Ffmpeg/GStreamer, how to make sure it's no regression for our components?
- We maintained 5 generations of hardware. How to make sure they are well tested.
- Community review is slow, how to make sure our customer get updated hw features.
- How to make sure the pending patches are well tested and reduce the rebase efforts.

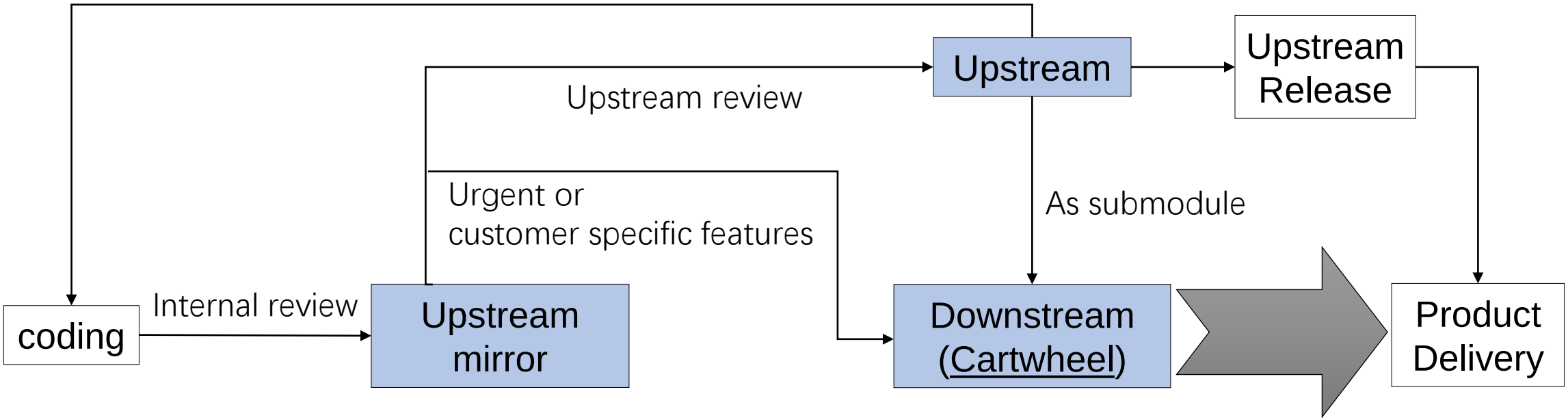
Open Source Development Process



CI system & Cartwheel



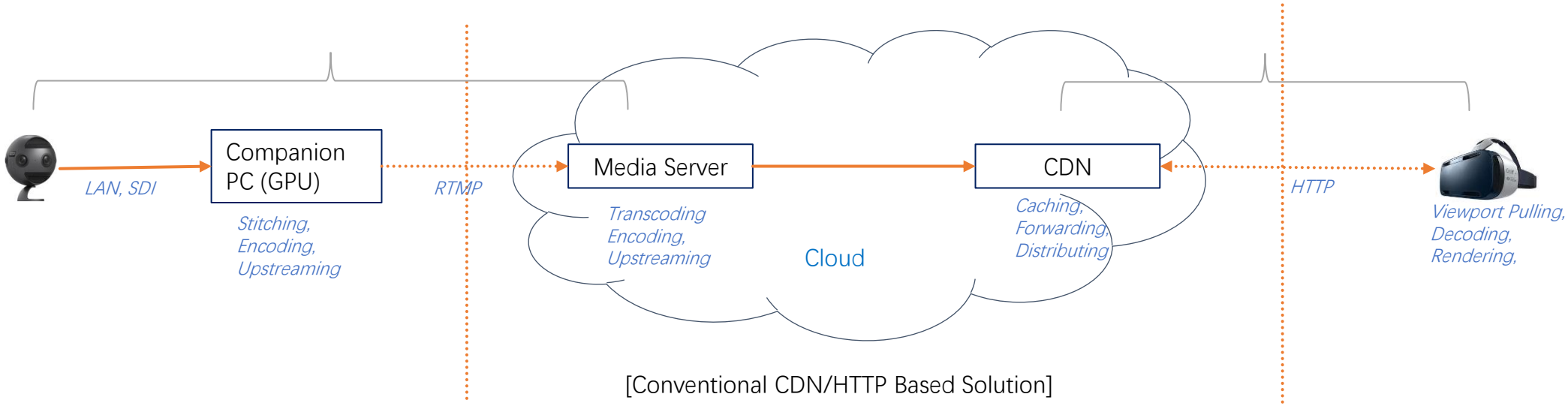
A line of code's real life



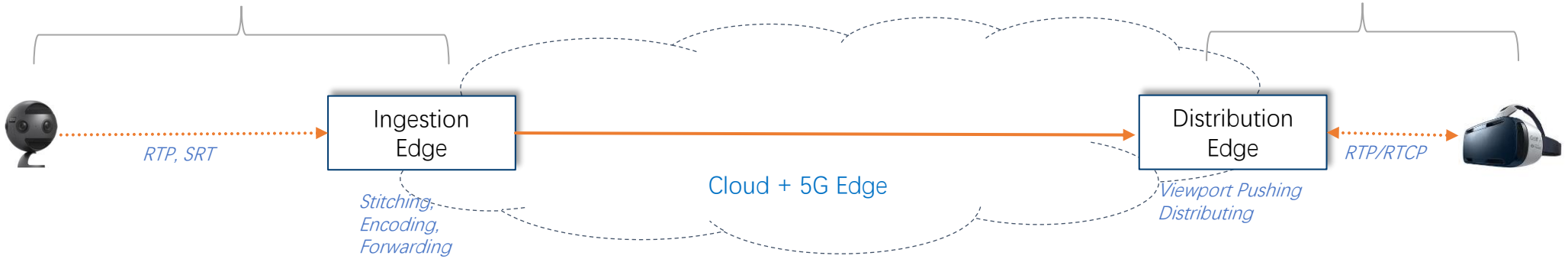
CI covered

Embracing New Technologies

The value of 5G to media framework



[Conventional CDN/HTTP Based Solution]



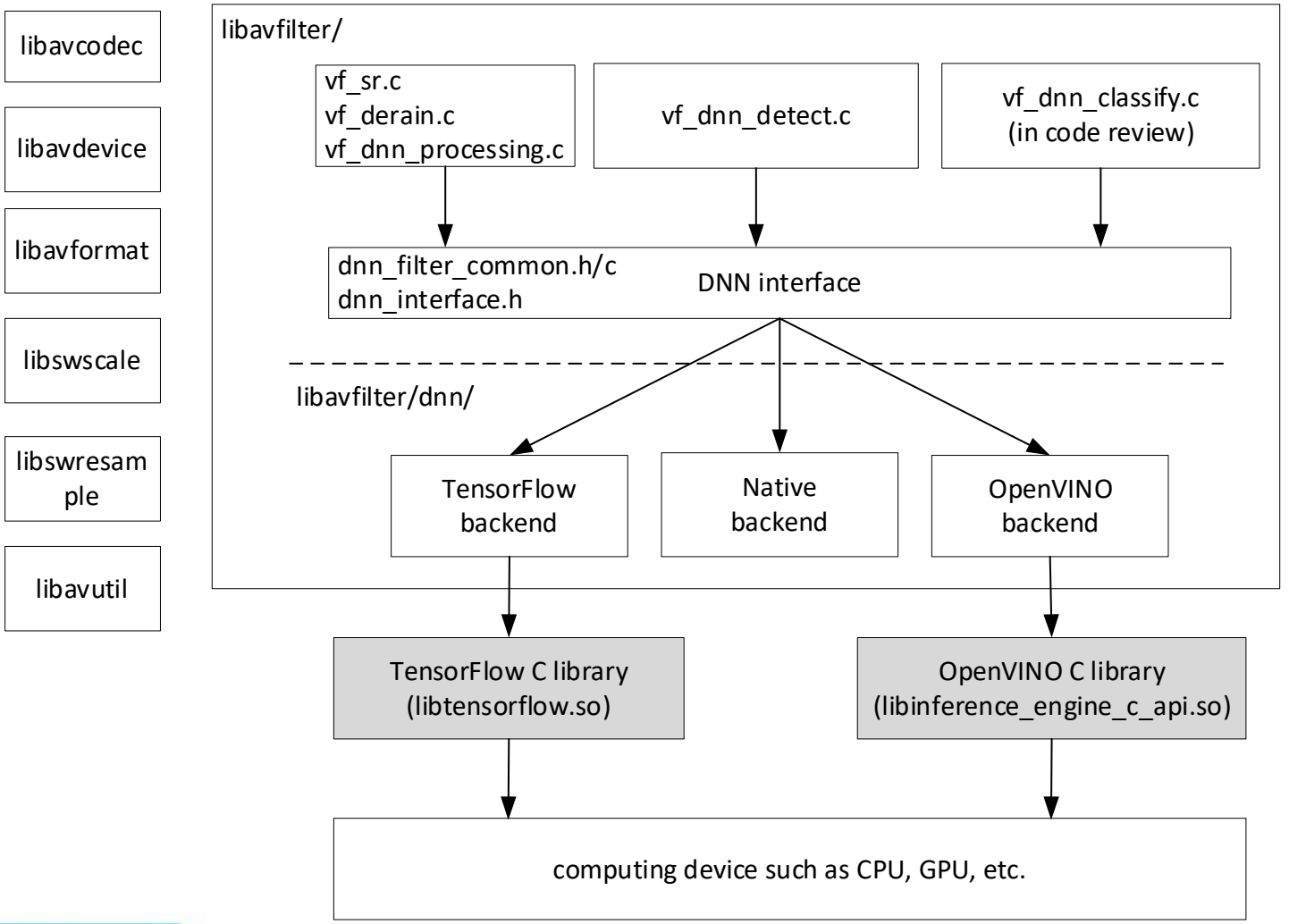
[5G Edge Based Solution]

The value of AI to Media

- Video analysis
  - Pixel analysis
- Visual quality enhancement.
  - AI denoise
  - Super resolution
  - Frame interpolation
  - ROI
  - Beautify, smooth skin
  - Per tile encode optimization

FFmpeg Deep Neuro Network

- Support AI through OpenVino Tensorflow and Native mode
- Video Enhancement: Derain, Super Resolution
- Video Analysis: Object Detection




SACC
THANKS