Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2024, 11(9):44-46

Research Article ISSN: 2394 - 658X

Zero Copy Camera Capture Pipelines

Karthik Poduval[1], Karthick Kumaran Ayyalluseshagiri Viswanathan[2]

[1]San Jose, CA, USA, karthik.poduval@gmail.com
[2]Tracy, CA, USA, asvkarthick@gmail.com

ABSTRACT
Zero Copy is a concept where the output of a producer module is directly used by its downstream consumer module (without making a copy). Zero Copy pipelines offer lower latencies and reduce the overall memory bandwidth of the system, however they do come with their own set of synchronization and compatibility issues to deal with.

Keywords: Zero Copy, Camera Pipelines, G-Streamer

INTRODUCTION

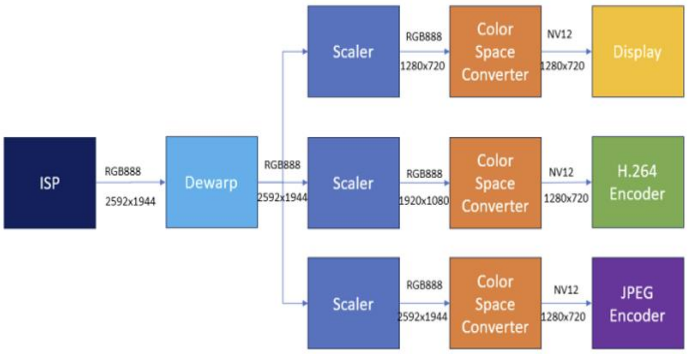

Figure 1: Camera Capture Pipeline

Camera Capture and processing involves using an ISP that captures from an image sensor as shown in Fig 1. The next module is a Dewarp module that performs a lens distortion correction on the image. Depending on the application, we may need various scaled versions of the dewarped output. In this particular case assume a 720p preview, 1080p video encode and a 5MP image burst capture use case. Based on this use case, the pipeline includes color space converter and encoders for video like H.264 (audio capture not shown) and JPEG encoders. In order for the pipeline to be Zero Copy, the output of one module needs to be consumed by the input of the other. With Zero Copy, we can save on the expensive copies between modules that would lead to lower latencies (time spend copying saved) and also save precious memory bandwidth (by avoiding the copying). For a successful Zero Copy pipeline establishment, the modules must understand each other's format directly and also take care of synchronization of buffer usage i.e. only one module is in possession of the buffer at a given time. If the modules are hardware based (typical for embedded systems) then the hardware stride requirements must also match.

MEMEORY ALLOCATION

As we explored in the previous section, there needs to be some sort of allocation scheme for zero copy so that the output of a module is consumable by the input of the next. There are two concepts largely used for this.

• **Global Allocator:** Here the allocation is done globally with the help of a global allocator such as Android's Gralloc [1]. With a global allocator concept the allocator is aware of all the modules of the pipeline and takes care of all such requirements. Let's take an example of scaler and color space converter. Now imagine if scaler needs 16-

byte alignment of every line start, while color space converter needs 64 byte alignment of line start, the Gralloc will know this for the given SoC and make sure that all RGB888 allocations are made with stride length to meet 64 byte alignment (as that alignment would work for all the modules). Gralloc type system will also typically make use of the Linux DMABUF [2] framework that allows for sharing buffers using file descriptor handles that can be imported by various modules (to get access to the underlying memory). Gralloc implementations also typically make use of DMA-BUF Heaps [3] which fits the global allocator model very well where all allocations are made from a centralized allocation heap and used by all the modules. Some implementations also make use of a GEM Allocator where the grahphics stack supports global allocation instead of the DMA-BUF Heaps.

• **Exporter Importer Model:** In this concept, the upstream module allocates or exports the buffer while downstream module uses it or imports it. Ex: ISP allocates RGB888 2491x1944 buffer and Dewarp modules uses or imports it. This is the typical default used by G-Streamer. Frameworks like gstreamer use a step called caps negotiation [4] where there is some back and forth between modules (Gstreamer elements) and upstream will make sure that allocation attributes like format, line stride are all consumable by downstream modules.

BUFFER POOL SIZE

With zero copy pipelines, we will definitely save on latencies introduced by making copies on module boundaries, but we need to understand that there will be a need for an increased shared buffer pool size. For example if ISP (as shown in Figure 2 operated on a buffer pool of 2 buffers and Dewarp needs a buffer pool of 2 as well, when in zero copy mode, the shared buffer pool in between ISP and Dewarp maybe have to be of minimum size of 3. As a general rule of thumb, we can think of this to be.

$$\min shared\ bufer\ pool\ size = MAX(downstream\ module \min buffer\ pool\ size, upstream\ module \min buffer\ pool\ size)$$

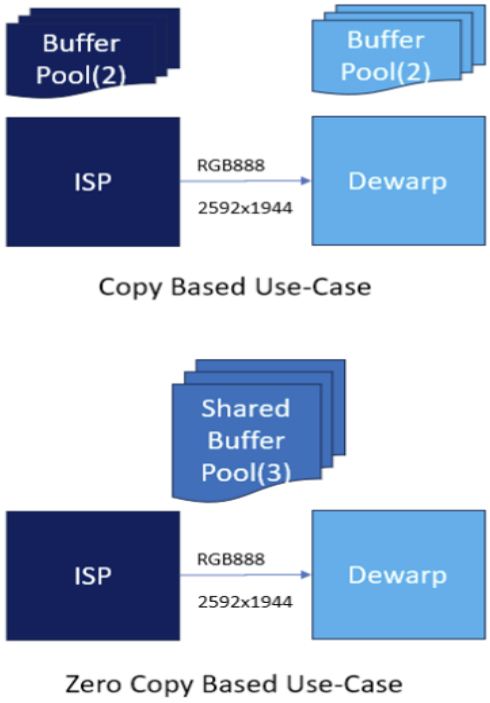

Figure 2: Buffer pool size

GSTREAMER BASED ZERO COPY PIPELINE

GStreamer [5] is a popular open-source framework that allows creation of processing pipelines. Zero Copy is fully supported on GStreamer as a framework and needs to utilize the iomode property of the plugin (iomode=4 is DMABUF). Let us consider a simple example of v4l2src upstream plugin that flows into VPP that performs scaling and color space conversion Figure 3. Here the v4l2src produces a DMABUF capable output buffer that can directly be read by VPP plugin(module) [6], [7]. Gstreamer by design tries to do zero copy and, in this example, two hardware plugins v4l2src and VPP are uisng DMABUF method to perfrom zero copy.

```
# v4l2src produces DMA memory. Doing VPP with scaling and CSC YUY2->NV12.
gst-launch-1.0 \
  v4l2src io-mode=4 ! \
  video/x-raw,format=YUY2,width=640,height=480,framerate=30/1 ! \
  mfx_gst_vpp width=352 height=288 memory=system ! \
  autovideoconvert ! \
  ximagesink
```

Figure 3: Intel GStremer Zero Copy Pipeline

REFERENCES

[1]. "Gralloc." [Online]. Available: https://netaz.blogspot.com/2015/03/ androids-graphics-buffer-management.html
[2]. "Dmabuf." [Online]. Available: https://docs.kernel.org/driver-api/ dma-buf.html
[3]. "Transitioning from ion to dma-buf heaps." [Online]. Available: https://source.android.com/docs/core/architecture/kernel/dma-buf-heaps
[4]. "Gstreamer caps negociation." [Online]. Available: https://gstreamer.freedesktop.org/documentation/plugin-development/ advanced/negotiation.html?gi-language=c
[5]. "Gstreamer." [Online]. Available: https://gstreamer.freedesktop.org/
[6]. "Zero copy with dmabuf." [Online]. Available: https://github.com/ Intel-Media-SDK/gstreamer-plugins/blob/master/README.CAMERA
[7]. "Zero copy pipeline with gstreamer." [Online]. Available: https://gstreamer.freedesktop.org/data/events/gstreamer-conference/ 2017/Nicolas%20Dufresne%20-%20Zero-Copy%20Pipelines%20in% 20GStreamer.pdf