GPUSqueeze software library is intended for developers implementing high speed video transcoding or encoding on a single or multiple GPUs. This is the only software library available on the market supporting multi-GPU video encoding. The library provides very simple API and can be quickly integrated into a user's framework.

Technical specifications:

- input formats for encoding: Bayer, YUV (planar 420), RGBx, Grayscale.

- input video formats for transcoding: H.264 (AVC), H.265 (HEVC), VP8/9, MPEG2, VC1.

- output video formats: H.264 (AVC), H.265 (HEVC).

- image processing: scaling (bicubic), aspect ratio conversion, demosaicing. User's own filters integration is possible.

- parameters control: 3 quality presets, bitrate, color adjustment (bayer).

The library does not provide any video container functionality. There are several free and open source solutions on the market, for example 'libav'.

The library is not intended for low latency processing, only for high performance and as the result it has very deep pipeline (depending on a number of GPUs in the system) and latency can achieve one or more GOPs.

The current public release supports only NVIDIA GPUs. The encoding performance are in the range of 300-800 fps at 2MP resolution and scales linearly with increasing number of GPUs. The number of simultaneously encoded streams with

different stream parameters are limited on consumer (non Quadro / Tesla) cards to one stream. If all streams are the same there is no limitation.

Functionality:

The following diagram on the following page shows the overall architecture and data flow.

The API is very simple:

- first, configure encoder through Init() function: pass input and output stream parameters and callback function addresses;

- allocate set of input buffers and its descriptors (FRAME_DESC)

- call Process() for each frame;

- receive DeliverOutput() with buffer containing compressed data;

- receive ReleaseInput() when input buffer can be refilled.

- return output buffer back to the library with ReleaseOutput() call.

Notes to developers:

The library is heavily multithreaded and callbacks comes from different threads.

Process() and ReleaseOutput() can be also called from different threads, but functions itself are not thread safe, i.e. only single thread can call each function.

It is important to set correctly ullFrameNumber field in the frame descriptor passed to the Process() function — the value should be different with every frame fed.

ullFrameNumber and ullTimeStamp fields in the frame descriptor are passthrough values for Process() » DeliverOutput() call chain.

DataRef[] and uiRefCountFlags fields in the input frame descriptors are free for use by developers and passed through Process() » ReleaseInput().

uiRefCountFlags in the output frame descriptors are free for use by a developer.

FRAME_DESC_QUEUE class provides atomic linked list functionality for frame descriptors. Note to a developers: FRAME_DESC class supports functionality of being simultaneously in several queues for cases when an input or output frame need to be processed in parallel by various parts of user application. Hint: uiRefCountFlags atomic can be used as bitfield based reference counter.

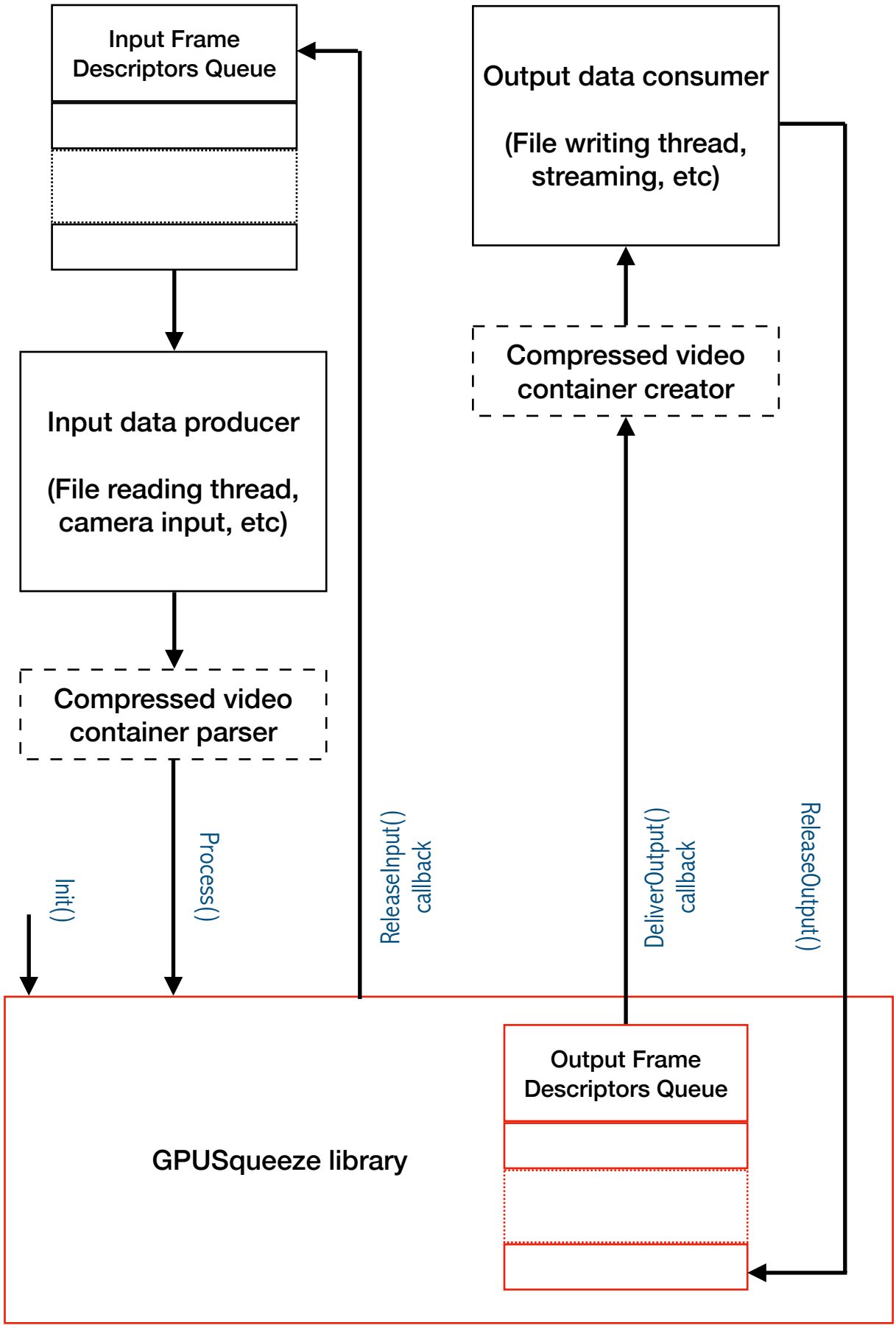
For compressed input streams:

- Feed order should be in decoding order, not display order. Output is always in decoding order.

- If an input stream does not have stream description information (VPS/SPS/PPS) embedded in it, then this information should be passed during initialization in StreamParameters.CodeInfo structure.

To flush the pipeline pass null pointer in the frame descriptor. Call to WaitForIdle() also flushes the pipeline.

Output images will have watermark if the library is initialized without correct key.


Input Frame Descriptors Queue
Output data consumer
(File writing thread, streaming, etc)
Input data producer
(File reading thread, camera input, etc)
Compressed video container creator
Compressed video container parser
Init()
Process()
ReleaseInput() callback
DeliverOutput() callback
ReleaseOutput()
Output Frame Descriptors Queue
GPUSqueeze library