How video works

Deep Dive

folio3

Marib Sultan

What we will cover

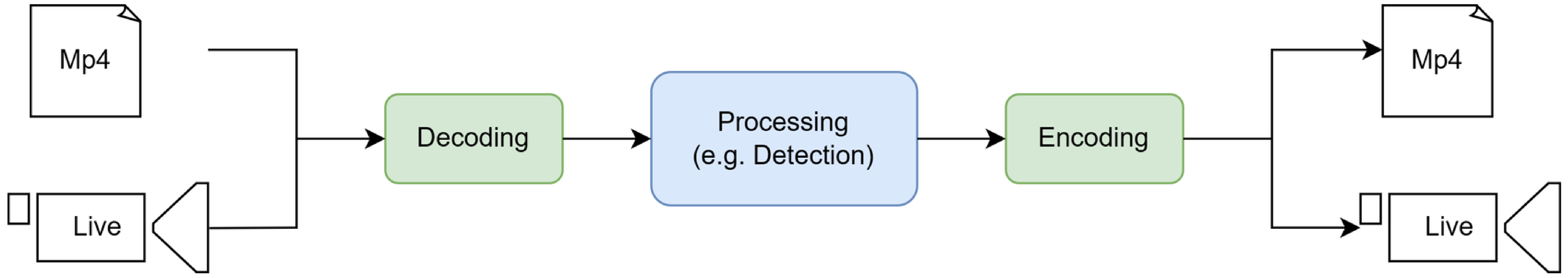
How image/video data is,

- Stored - Compressed/Decompressed (ENC/DEC), using S+H
- Transmitted/Streamed (RTP/HLS)

How/What hardware is,

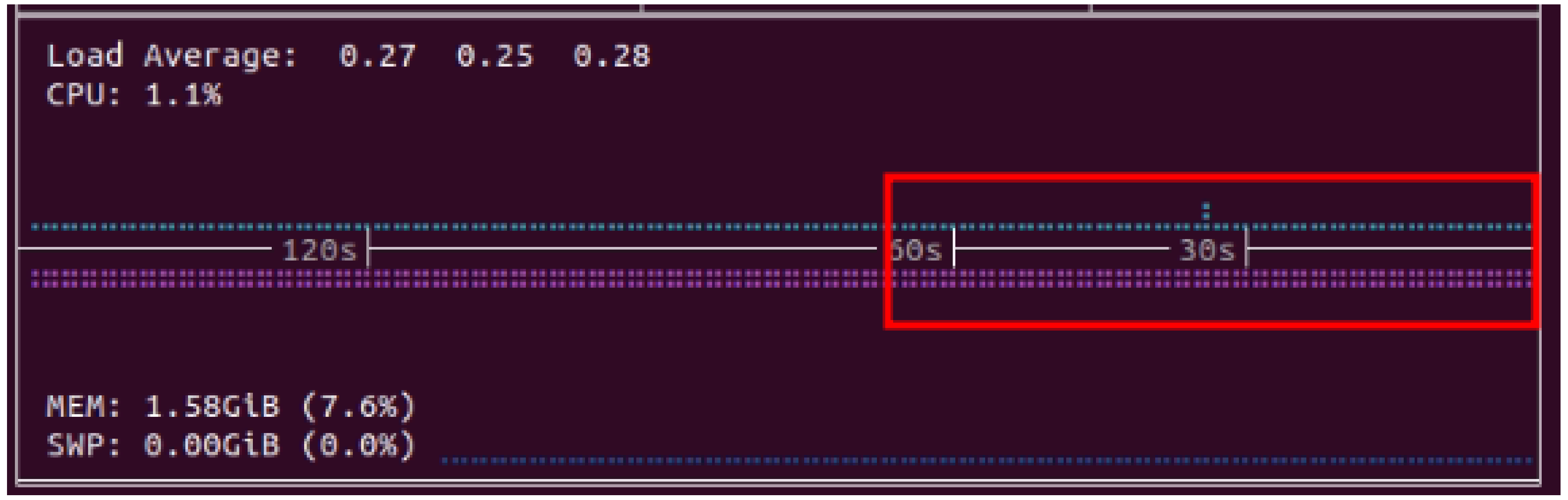
- Involved in above processes
- Used to accelerate the process

Why?


Mp4
Live
Decoding
Processing
(e.g. Detection)
Encoding
Mp4
Live

Why cntd..

```python
import cv2

pipeline_string = """
rtspsrc location=rtsp://admin:abcd1234@192.168.1.105:554/cam/realmonitor?channel=1\&subtype=0 !
tee name=t t. ! queue ! rtph264depay ! h264parse ! mpegtsmux ! filesink location=wow.mp4 -e t. !
queue ! decodebin ! videoconvert ! video/x-raw,format=BGR ! appsink sync=false"""

vcap = cv2.VideoCapture(pipeline_string, cv2.CAP_GSTREAMER)

while(True):

    ret, frame = vcap.read()
    if not ret:
        vcap.release()
        cv2.destroyAllWindows()
        break
```



```
Load Average:  0.27  0.25  0.28
CPU: 1.1%

MEM: 1.58GiB (7.6%)
SWP: 0.00GiB (0.0%)
```

recording x4 1080p60 h264 streams simultaneously

Basics

Efficient storage and transmission at the cost of computation

Transmission - ENC

RRRRRRRRRRRRRRRR
------------16 bytes-----------

16R
------------3 bytes-----------

Reception - DEC

RRRRRRRRRRRRRRRR
------------16 bytes-----------

RRRRRRRRRRRRRRRR
------------16 bytes-----------

The JPG

Spatial, Intra frame compression

- A 1080p image == 1920x1080 pixels
- 1 Pixel == (255,255,255) == 24 bits
- => 1080x1920x24 bits
- => 6.2 MB

1920x1080_sample.jpg

1.5 MB

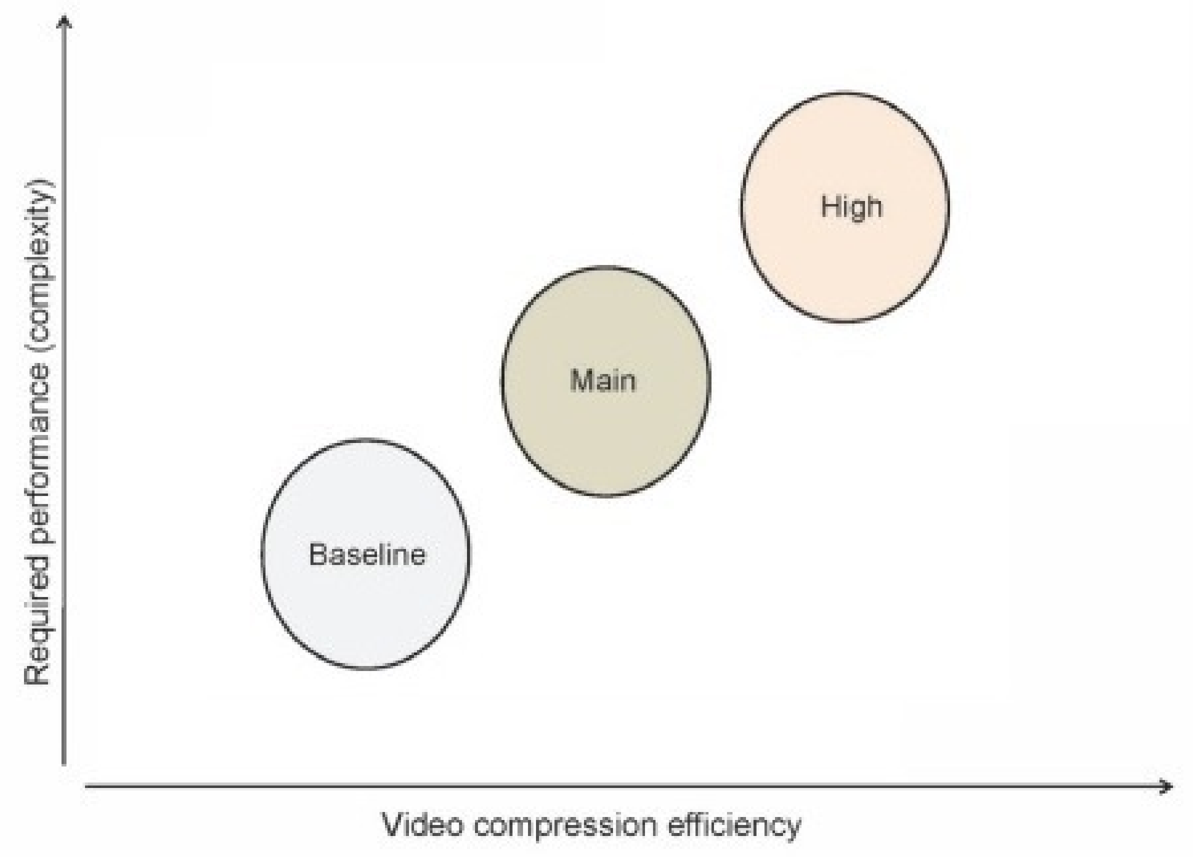
Lossy vs Lossless

Not all compressions are same.

jpg	png
• Smaller file size	• Larger file size
• Lossy	• Lossless
• Lower quality	• Higher quality

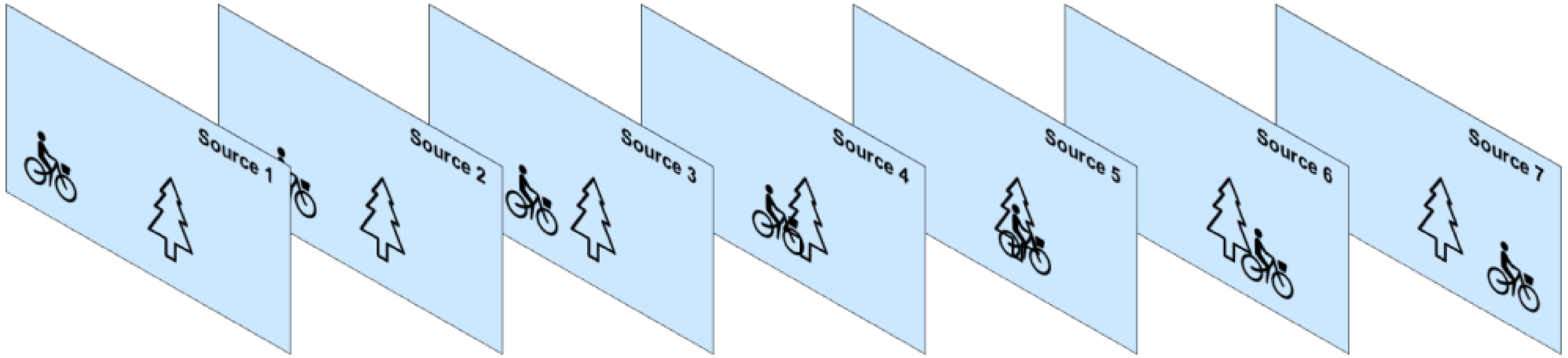
Videos...

- Resolution
- FPS
- Profile
- Bitrate
- Presets


Required performance (complexity)
High
Main
Baseline
Video compression efficiency

Spatial MJPEG

Not to be confused with MPEG

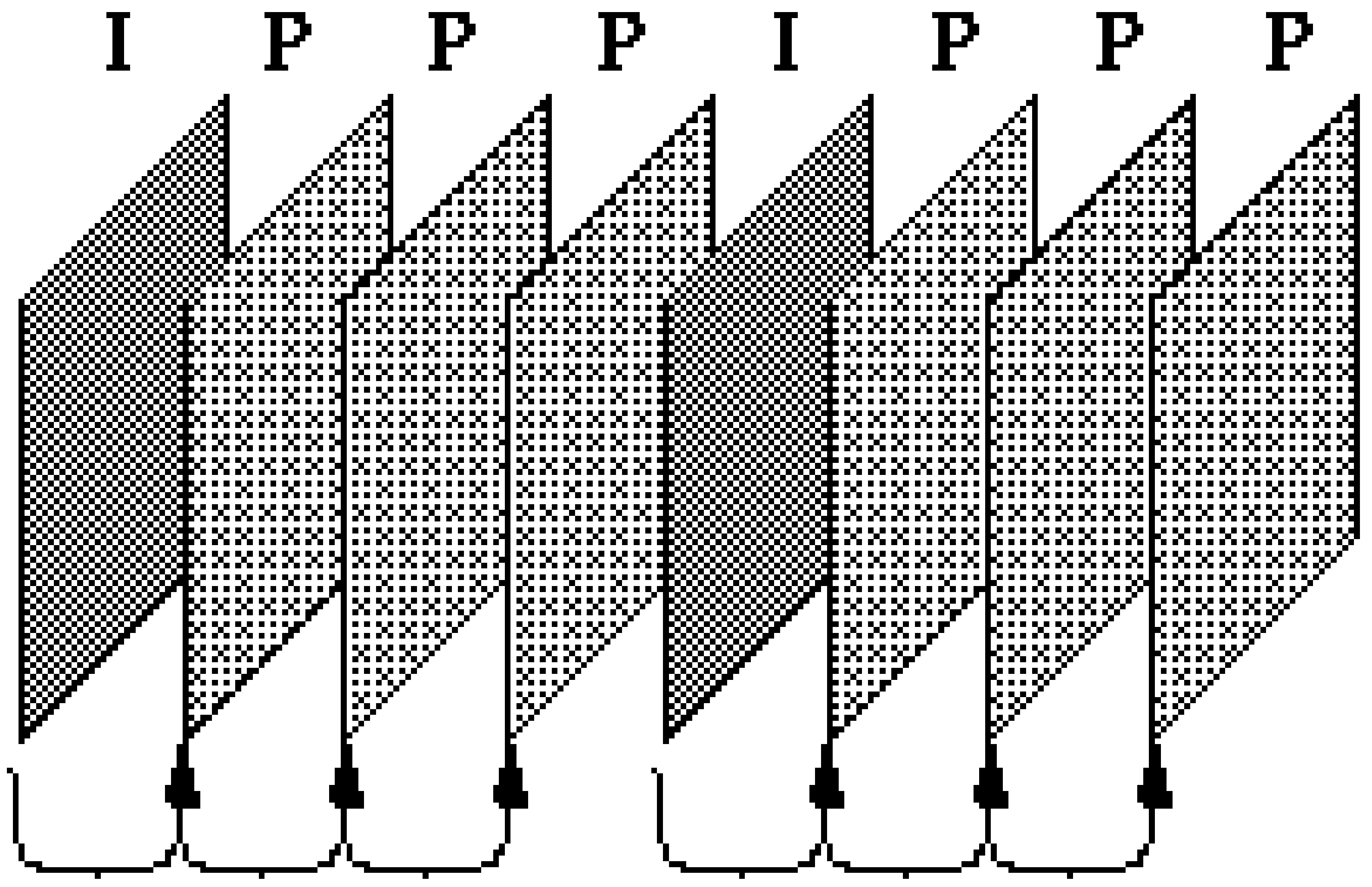


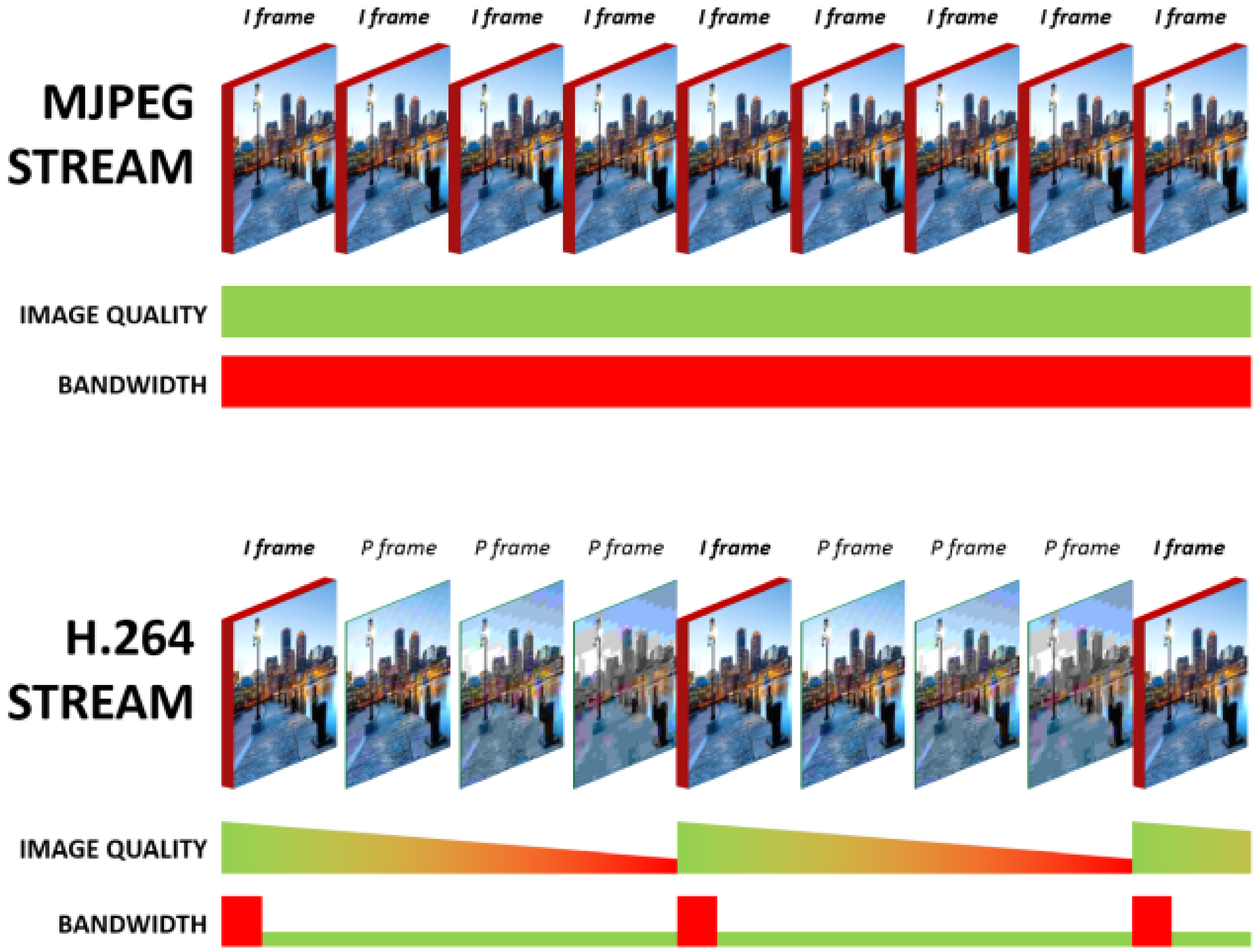
Sequence of JPEG frames

Beyond Spatial H264

Temporal, Inter frame compression

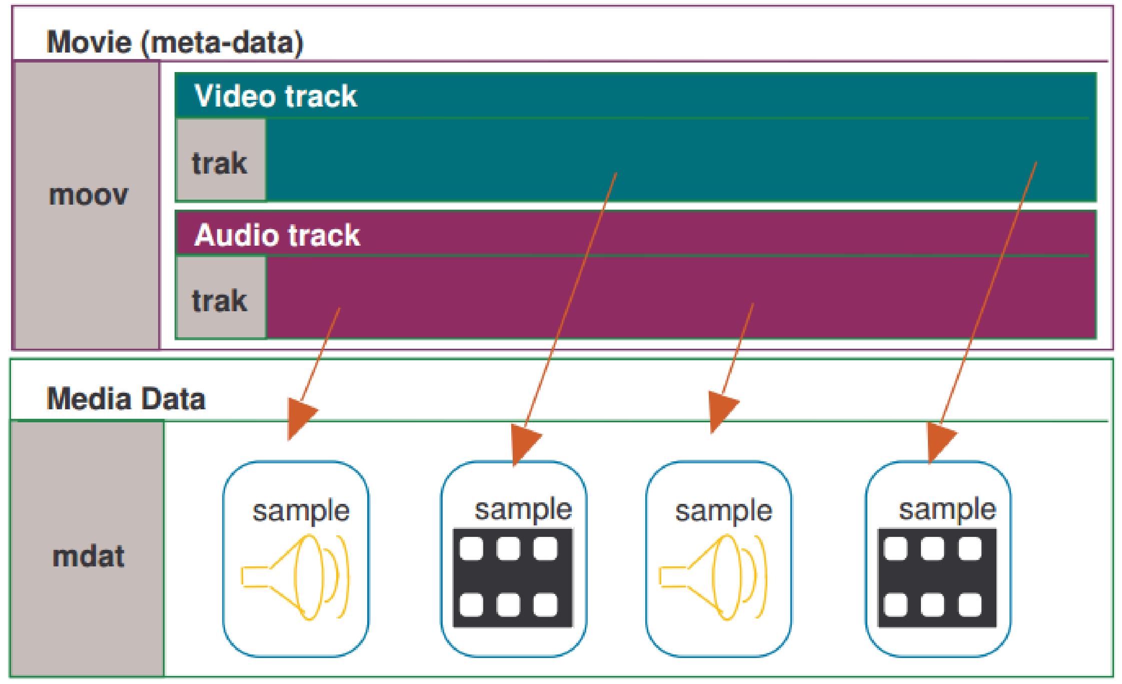
- Typically used to store data as video files
- Can also be streamed

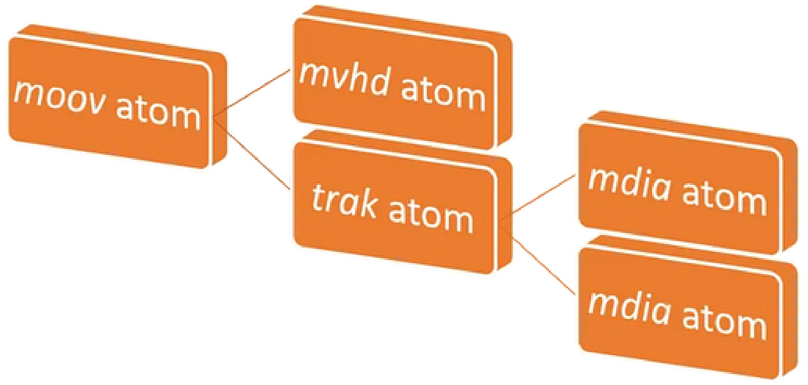


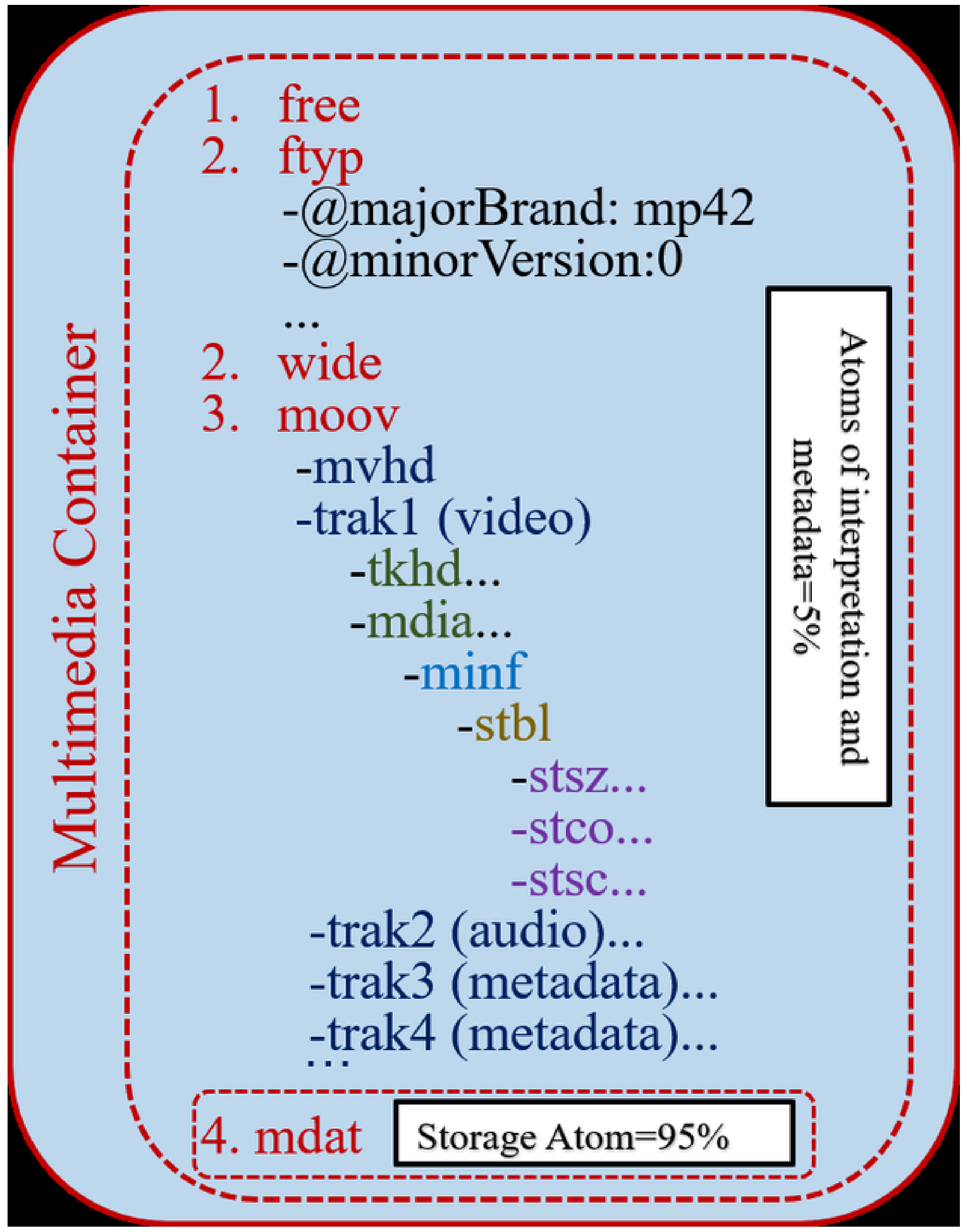

I frame
I frame
I frame
I frame
I frame
I frame
I frame
I frame
I frame
MJPEG STREAM
IMAGE QUALITY
BANDWIDTH
I frame
P frame
P frame
P frame
I frame
P frame
P frame
P frame
I frame
H.264 STREAM
IMAGE QUALITY
BANDWIDTH

Codecs != Containers

H264 != .mp4







Software enc/dec

Not really "software"...... (CPU)

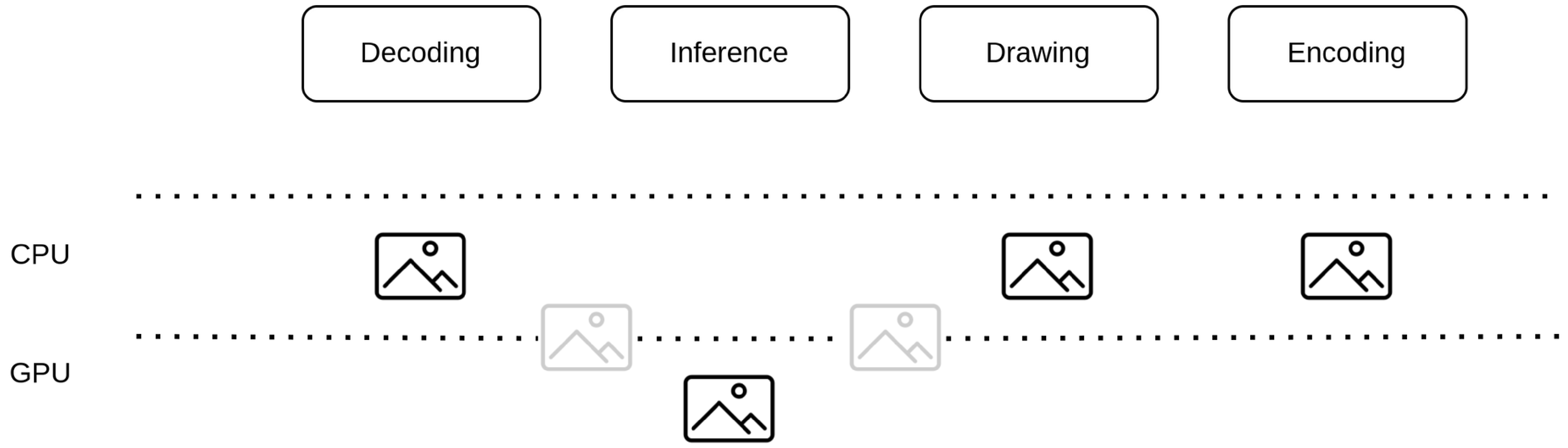
Decoding

```python
import cv2

videoReader = cv2.VideoCapture("1080_1920.mp4")

while(True):

    ret, frame = videoReader.read()
    if not ret:
        break

    cv2.imshow("frame",frame)
```

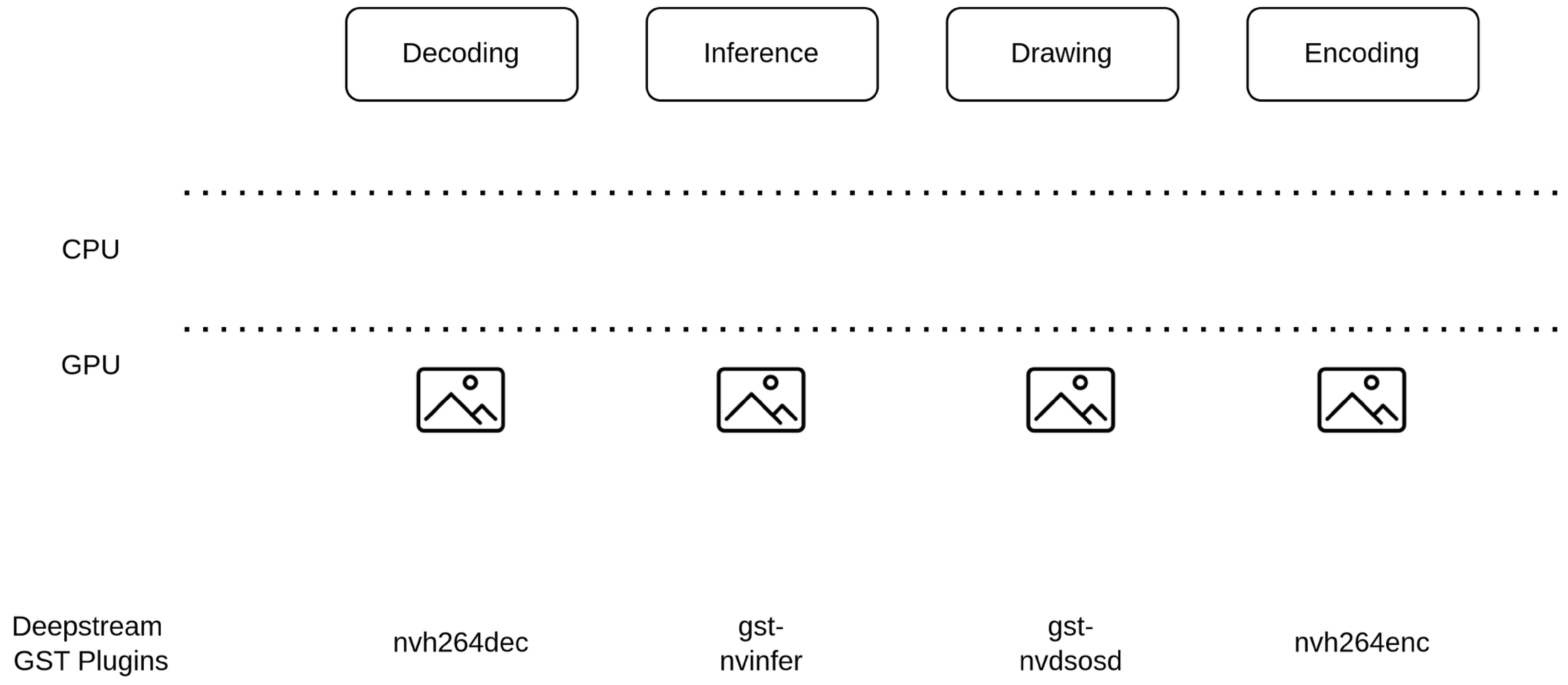
Encoding

```python
import cv2

videoReader = cv2.VideoCapture("1080_1920.mp4")
videoWriter = cv2.VideoWriter('processed.mp4',
    cv2.VideoWriter_fourcc(*'h264'), 25.0, (1920,1080))

while(True):

    ret, frame = videoReader.read()
    if not ret:
        break

    videoWriter.write(frame)
```

Conventionally

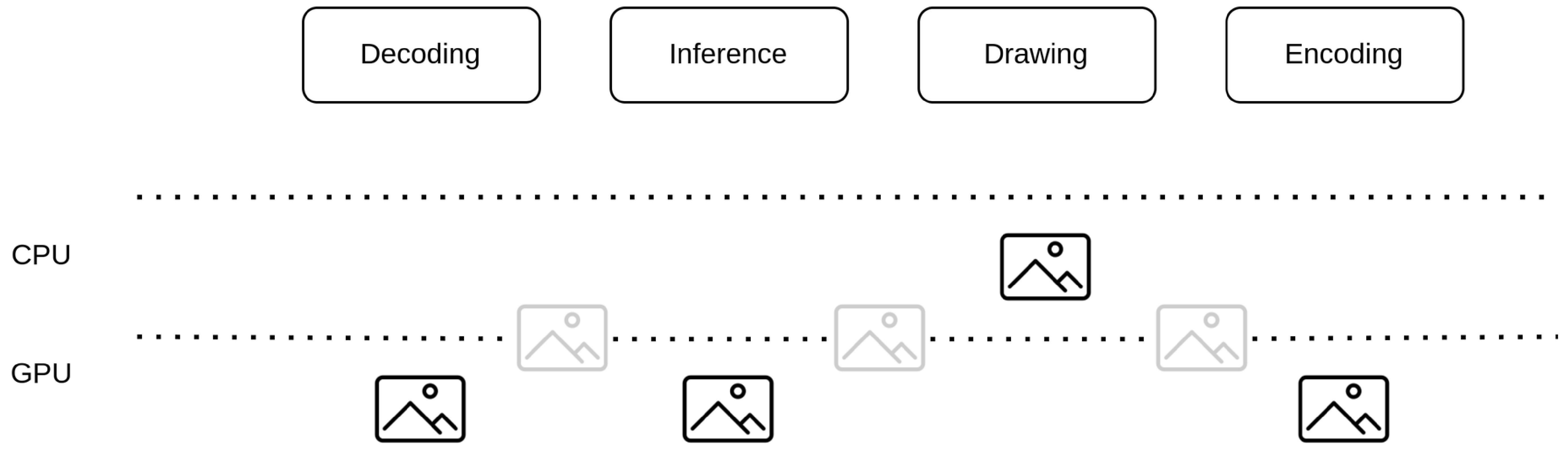


Nvidia Deepstream



Hardware enc/dec

OpenCV+GST FTW, the goodness of Deepstream w/o the ugliness



Opencv with GST Backend

nvh264dec

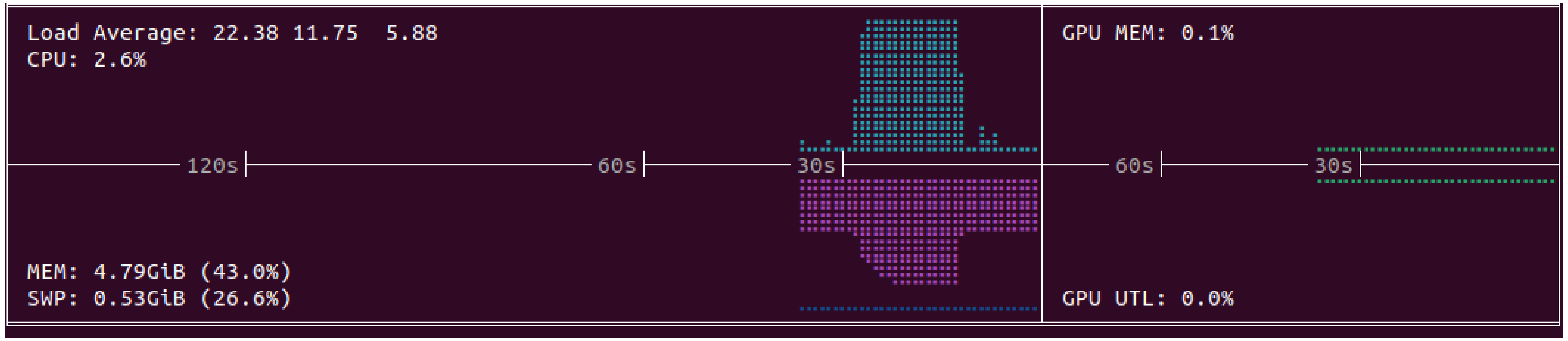
any

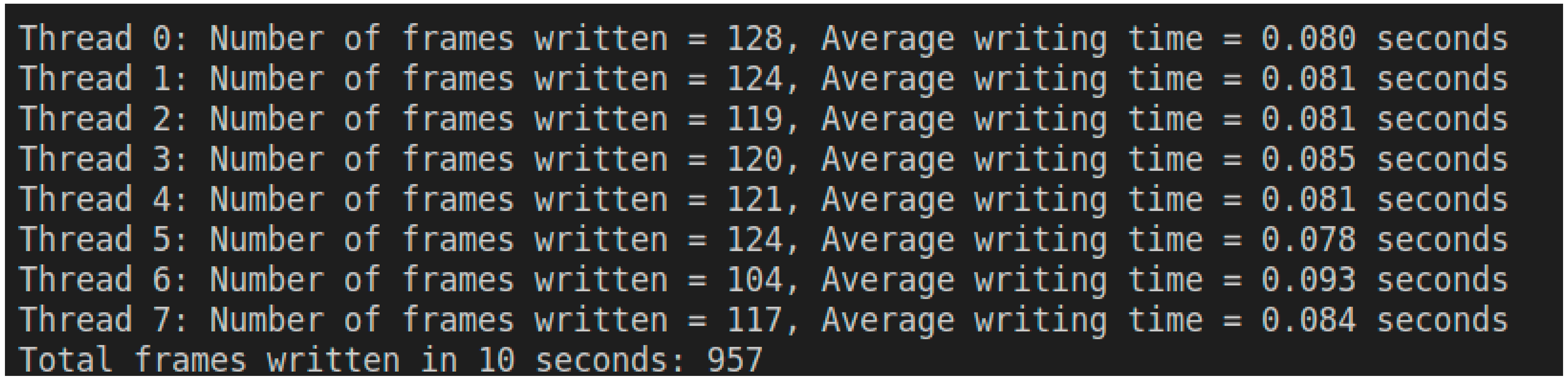
gst-nvdsosd

nvh264enc

Software enc benchmarks

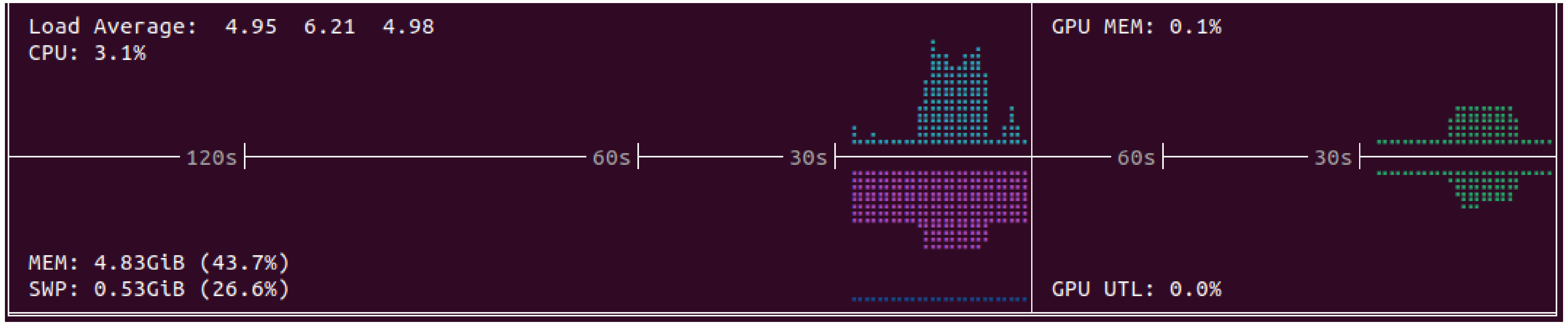
CPU : Encoding x8 1080p60 videos simultaneously, for 10 seconds


```
Load Average: 22.38 11.75  5.88
CPU: 2.6%
MEM: 4.79GiB (43.0%)
SWP: 0.53GiB (26.6%)
GPU MEM: 0.1%
GPU UTL: 0.0%
```

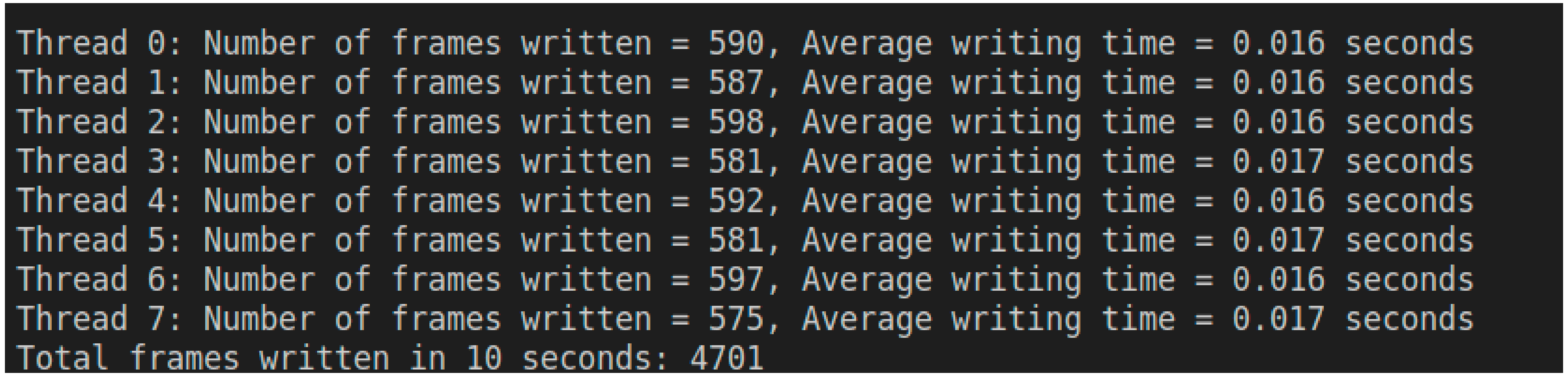

```
Thread 0: Number of frames written = 128, Average writing time = 0.080 seconds
Thread 1: Number of frames written = 124, Average writing time = 0.081 seconds
Thread 2: Number of frames written = 119, Average writing time = 0.081 seconds
Thread 3: Number of frames written = 120, Average writing time = 0.085 seconds
Thread 4: Number of frames written = 121, Average writing time = 0.081 seconds
Thread 5: Number of frames written = 124, Average writing time = 0.078 seconds
Thread 6: Number of frames written = 104, Average writing time = 0.093 seconds
Thread 7: Number of frames written = 117, Average writing time = 0.084 seconds
Total frames written in 10 seconds: 957
```

Hardware enc benchmarks

GPU : Encoding x8 1080p60 videos simultaneously, for 10 seconds


```
Load Average:  4.95  6.21  4.98
CPU: 3.1%
MEM: 4.83GiB (43.7%)
SWP: 0.53GiB (26.6%)
```

```
GPU MEM: 0.1%
GPU UTL: 0.0%
```


```
Thread 0: Number of frames written = 590, Average writing time = 0.016 seconds
Thread 1: Number of frames written = 587, Average writing time = 0.016 seconds
Thread 2: Number of frames written = 598, Average writing time = 0.016 seconds
Thread 3: Number of frames written = 581, Average writing time = 0.017 seconds
Thread 4: Number of frames written = 592, Average writing time = 0.016 seconds
Thread 5: Number of frames written = 581, Average writing time = 0.017 seconds
Thread 6: Number of frames written = 597, Average writing time = 0.016 seconds
Thread 7: Number of frames written = 575, Average writing time = 0.017 seconds
Total frames written in 10 seconds: 4701
```

Numbers

Software

```
videoWriter = cv2.VideoWriter('processed.mp4',
    cv2.VideoWriter_fourcc(*'h264'), 60.0, (1920,1080))
```

- CPU : 100%
- MEM : 80%
- Avg frame write time : 80ms
- Total frames in 10s : 957

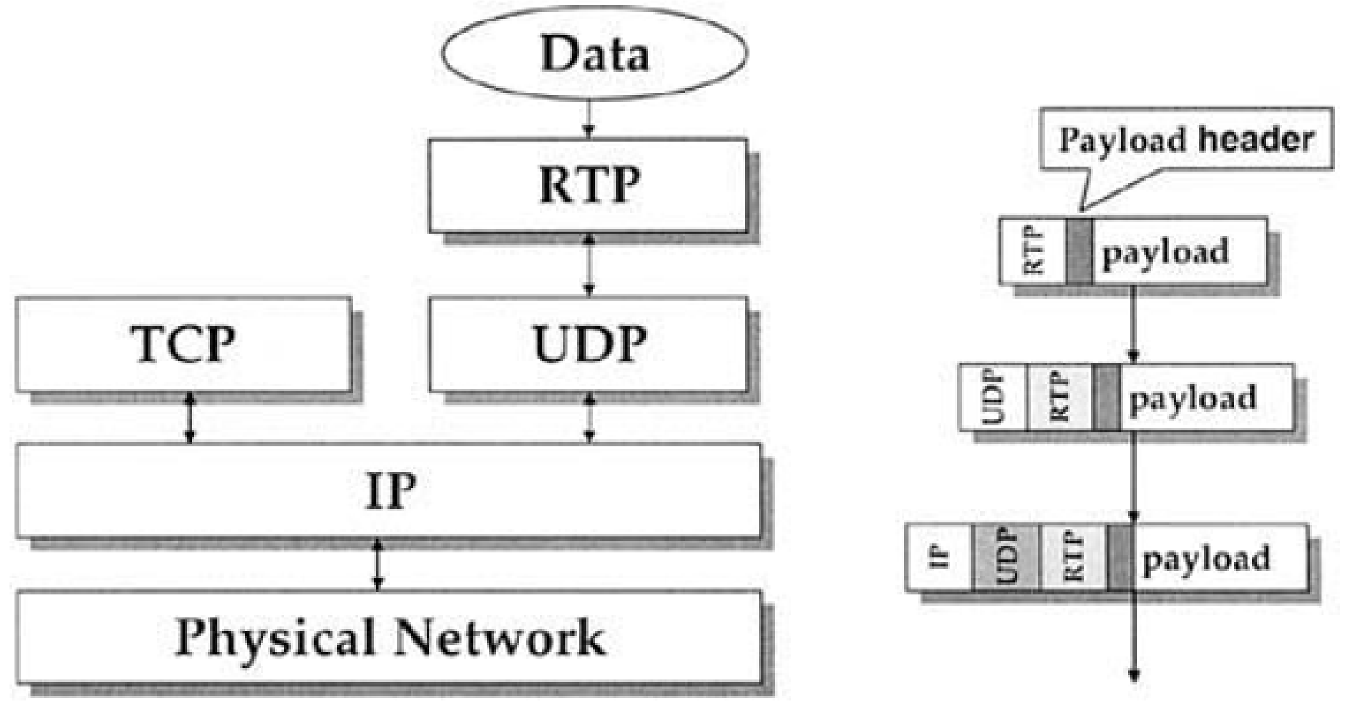
Hardware

```
VideoWriter = cv2.VideoWriter(
    """appsrc ! videoconvert ! nvh264enc bitrate=2000 !
    h264parse ! qtmux ! filesink location=processed.mp4""",
    cv2.CAP_GSTREAMER, 0, 60, (1920,1080))
```

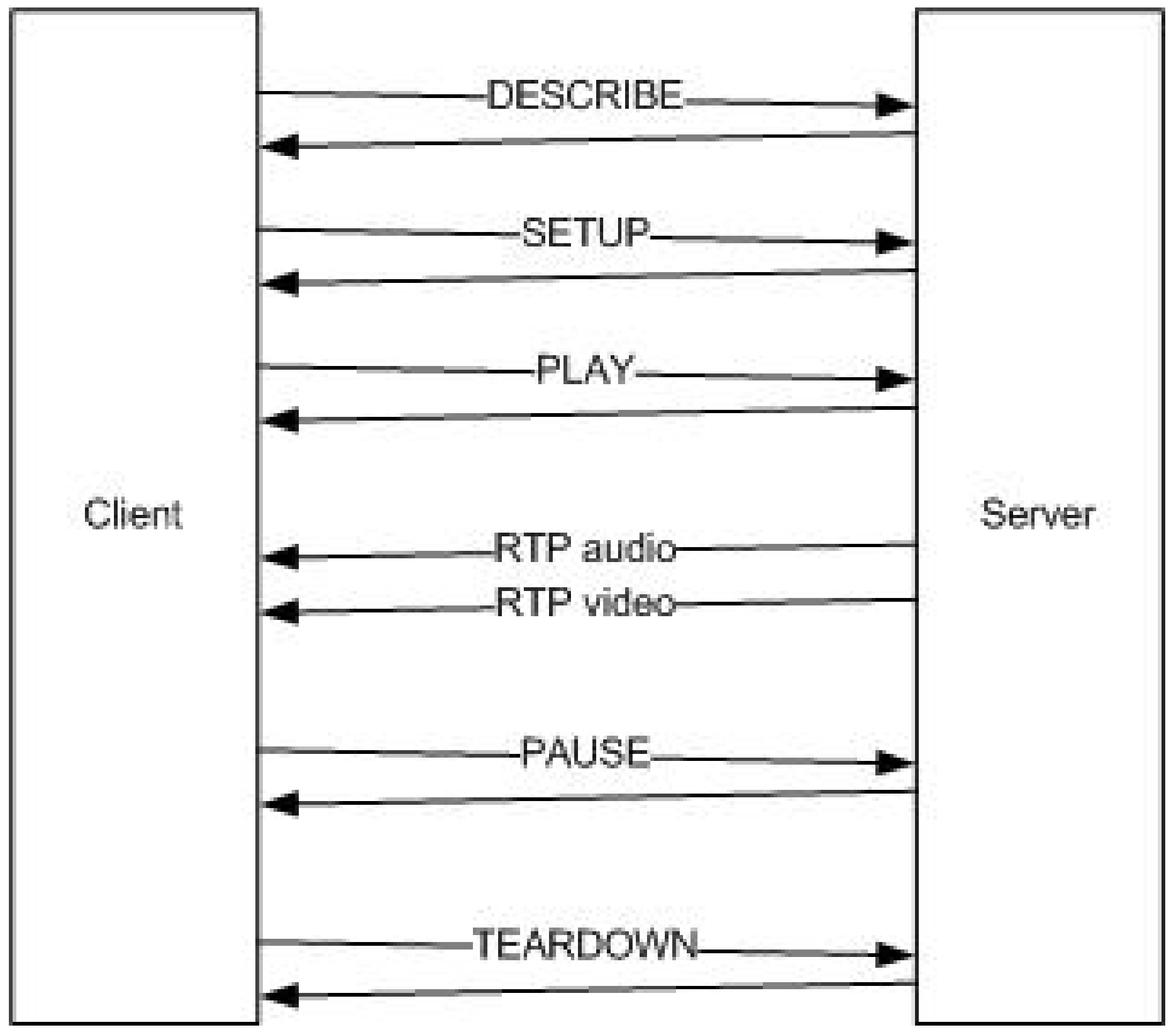
- CPU : 65%
- MEM : 60%
- x5 ↓ Avg frame write time : 16ms
- x5 ↑ Total frames in 10s : 4701

Streaming

RTSP vs RTP



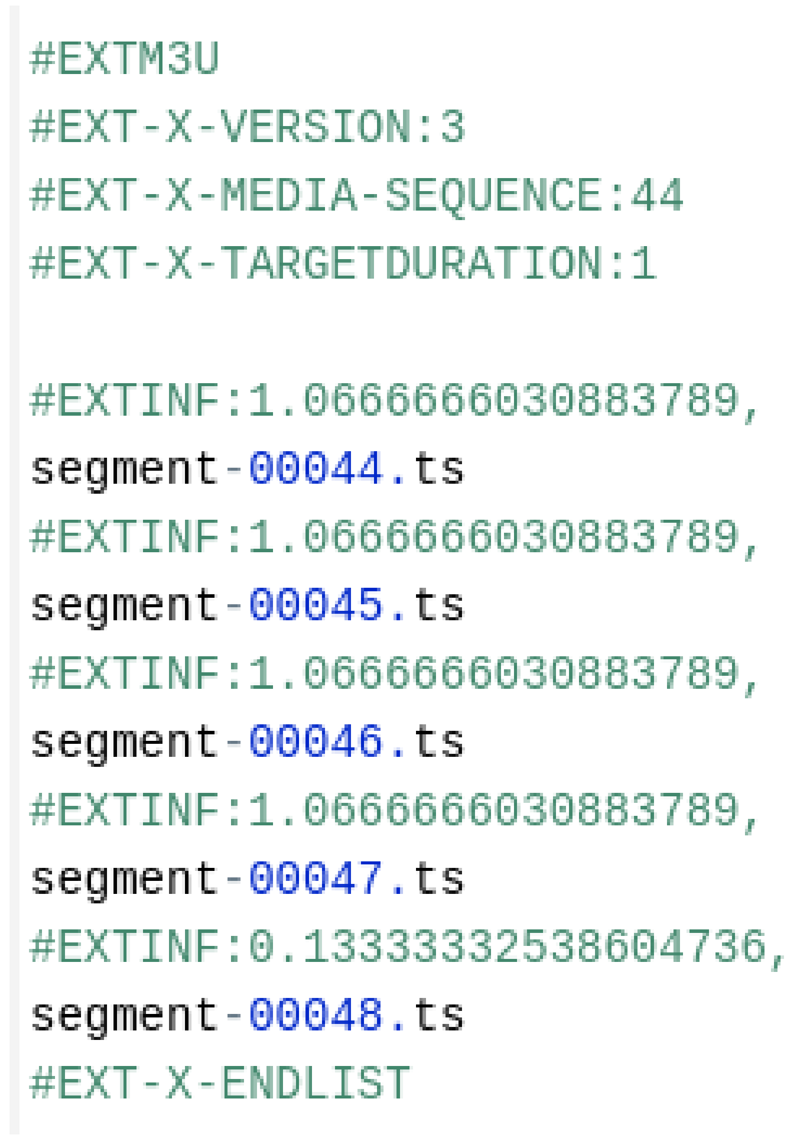
```
VideoWriter = cv2.VideoWriter(
    """appsrc ! videoconvert ! {encoder} bitrate=2000 !
    rtph264pay ! udpsink host=127.0.0.1 port=5000""",
    cv2.CAP_GSTREAMER, 0, 60, (1920,1080))
```



RTSP communication between client and server

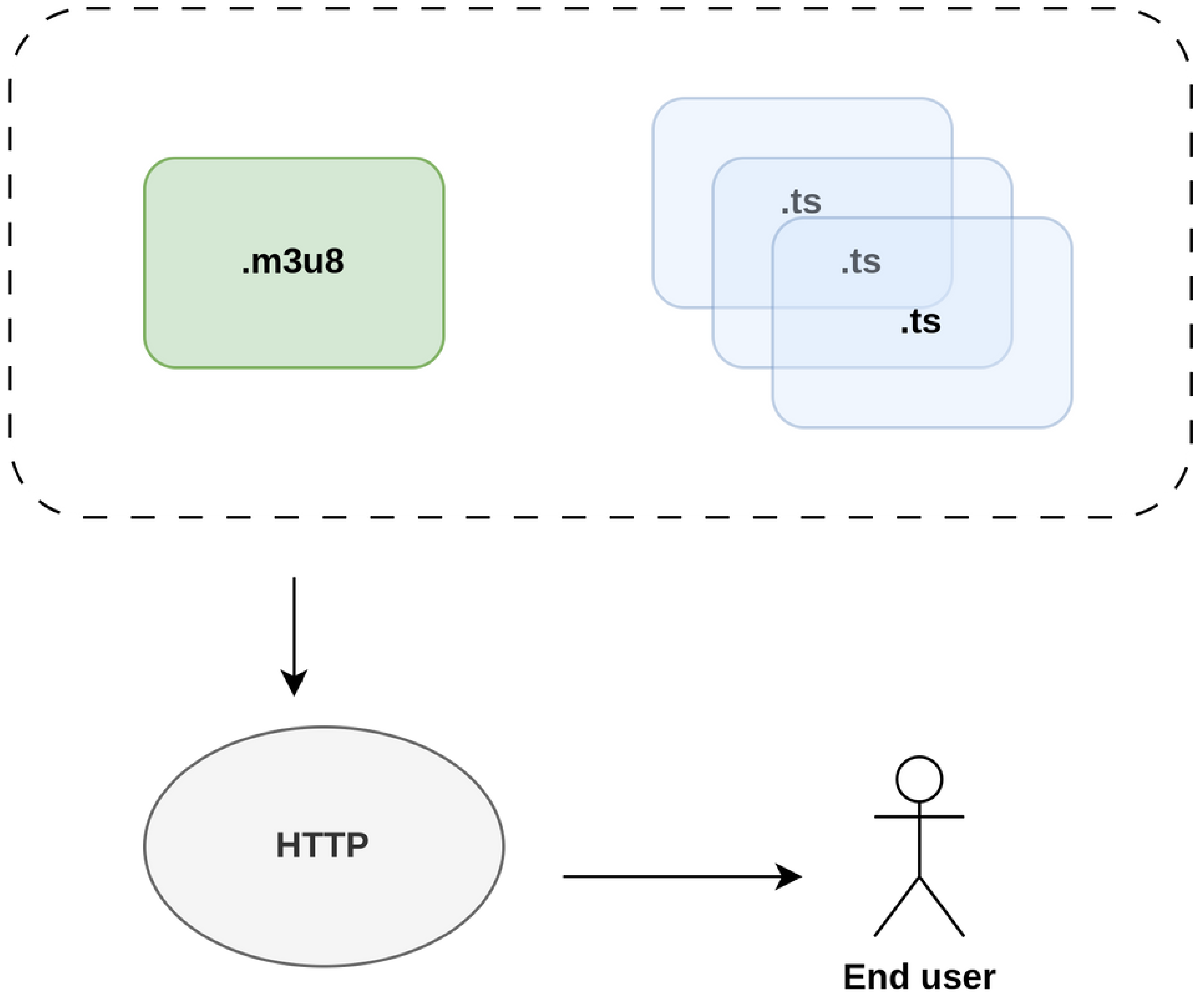
Streaming cntd..

HLS



```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:44
#EXT-X-TARGETDURATION:1

#EXTINF:1.0666666030883789,
segment-00044.ts
#EXTINF:1.0666666030883789,
segment-00045.ts
#EXTINF:1.0666666030883789,
segment-00046.ts
#EXTINF:1.0666666030883789,
segment-00047.ts
#EXTINF:0.13333332538604736,
segment-00048.ts
#EXT-X-ENDLIST
```

.m3u8 file



Recap

- How image and video data is stored - Encoded (jpg/h264)
- What constitutes a video (res, fps, profile etc)
- Video containers and atoms (.mp4)
- Software vs Hardware encoding (gst)
- Streaming videos (rtp,hls)

Thoughts?

https://m-a-r-i-b.github.io/

https://gist.github.com/m-a-r-i-b