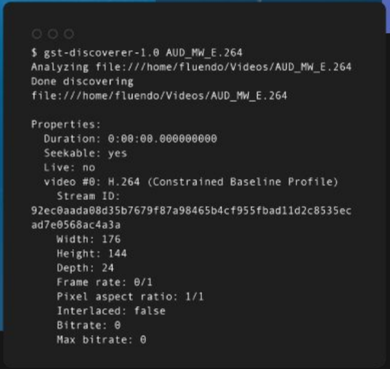
fluendo

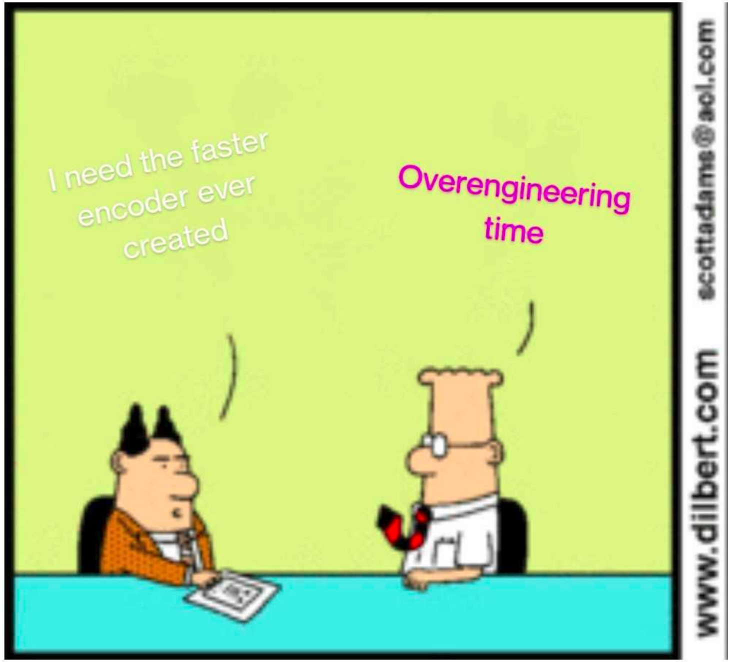
Hype
HYbrid Parallel Encoder

Ruben Gonzalez
A Coruña, September 2023

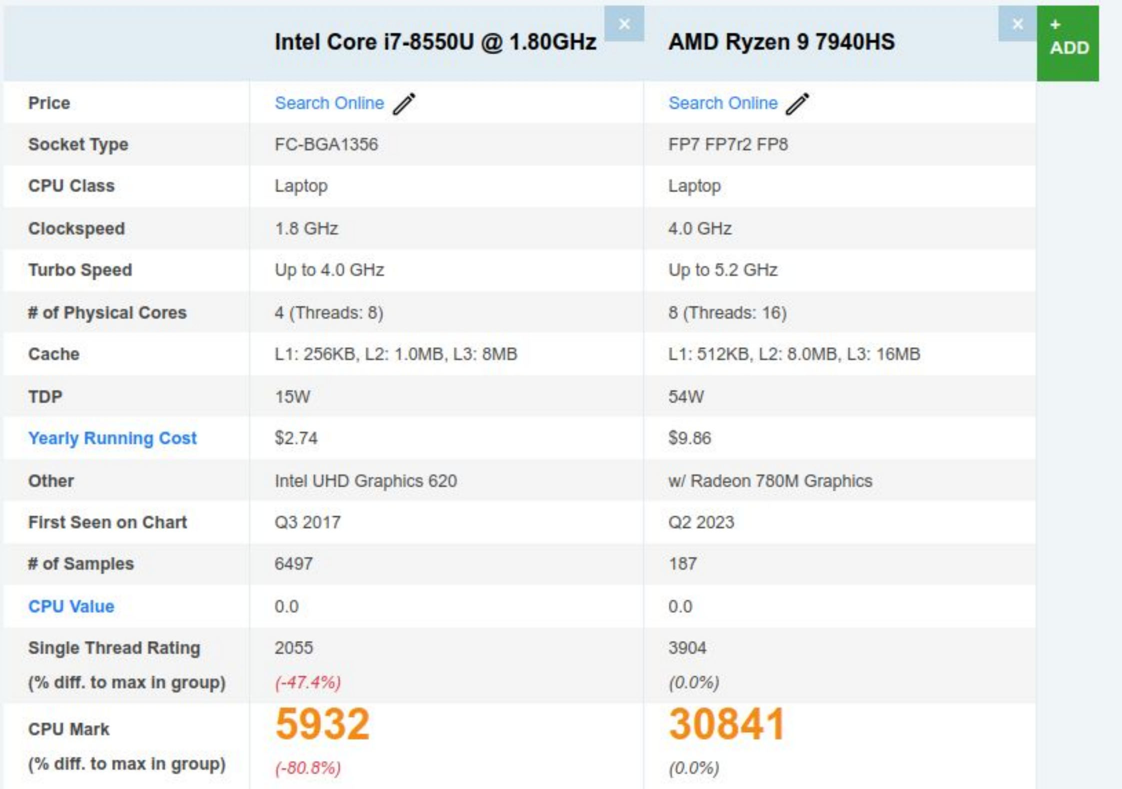


```
$ gst-discoverer-1.0 AUD_MW_E.264
Analyzing file:///home/fluendo/Videos/AUD_MW_E.264
Done discovering
file:///home/fluendo/Videos/AUD_MW_E.264

Properties:
  Duration: 0:00:00.000000000
  Seekable: yes
  Live: no
  video #0: H.264 (Constrained Baseline Profile)
    Stream ID:
92ec0aada08d35b7679f87a98465b4cf955fbad11d2c8535ec
ad7e0568ac4a3a
    Width: 176
    Height: 144
    Depth: 24
    Frame rate: 0/1
    Pixel aspect ratio: 1/1
    Interlaced: false
    Bitrate: 0
    Max bitrate: 0
```

One day in the office



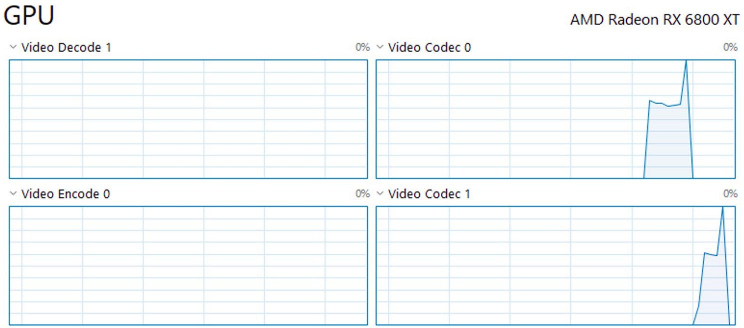
CPUs: fast CPU with a lot of cores



	Intel Core i7-8550U @ 1.80GHz	AMD Ryzen 9 7940HS
Price	Search Online	Search Online
Socket Type	FC-BGA1356	FP7 FP7r2 FP8
CPU Class	Laptop	Laptop
Clockspeed	1.8 GHz	4.0 GHz
Turbo Speed	Up to 4.0 GHz	Up to 5.2 GHz
# of Physical Cores	4 (Threads: 8)	8 (Threads: 16)
Cache	L1: 256KB, L2: 1.0MB, L3: 8MB	L1: 512KB, L2: 8.0MB, L3: 16MB
TDP	15W	54W
Yearly Running Cost	$2.74	$9.86
Other	Intel UHD Graphics 620	w/ Radeon 780M Graphics
First Seen on Chart	Q3 2017	Q2 2023
# of Samples	6497	187
CPU Value	0.0	0.0
Single Thread Rating (% diff. to max in group)	2055 *(-47.4%)*	3904 *(0.0%)*
CPU Mark (% diff. to max in group)	**5932** *(-80.8%)*	**30841** *(0.0%)*

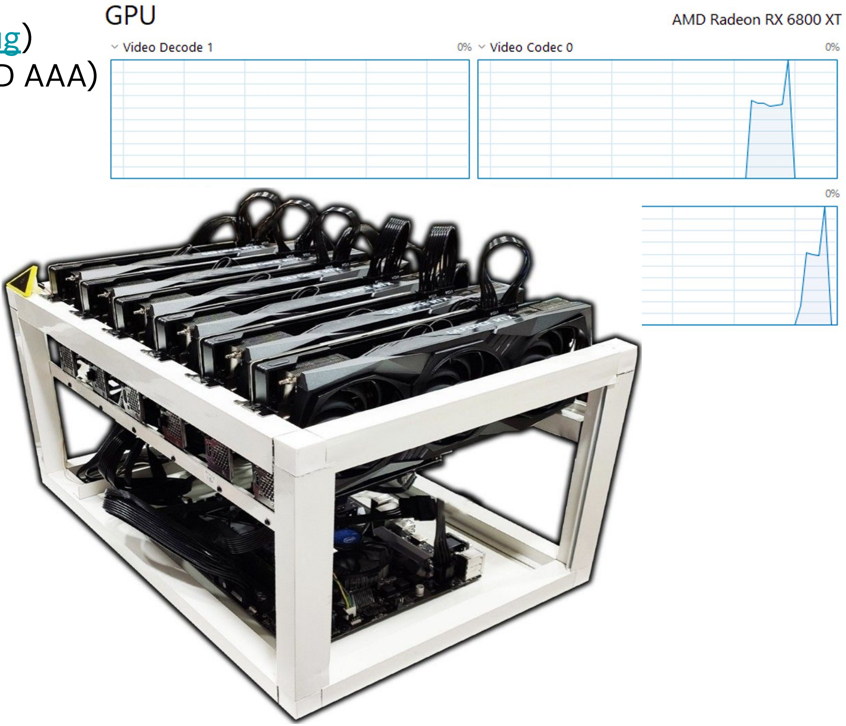
GPUs: Video Accelerator Hardware encoder

- GPUs with multiple hardware encoding like AMD (AMF bug)



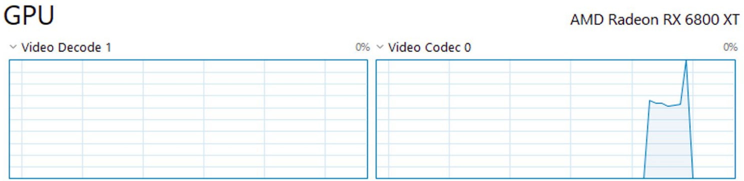
GPUs: Video Accelerator Hardware encoder

- GPUs with multiple hardware encoding like AMD (AMF bug)
- Dual GPU is becoming mainstream (Intel iGPU/dGPU, AMD AAA)
- Multi GPU easier than ever (ML or Mining use cases)



GPUs: Video Accelerator Hardware encoder

- GPUs with multiple hardware encoding like AMD (AMF bug)
- Dual GPU is becoming mainstream (Intel iGPU/dGPU, AMD AAA)
- Multi GPU easier than ever (ML or Mining use cases)



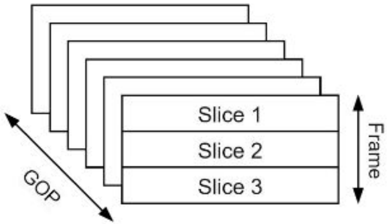
CPU & GPUs:

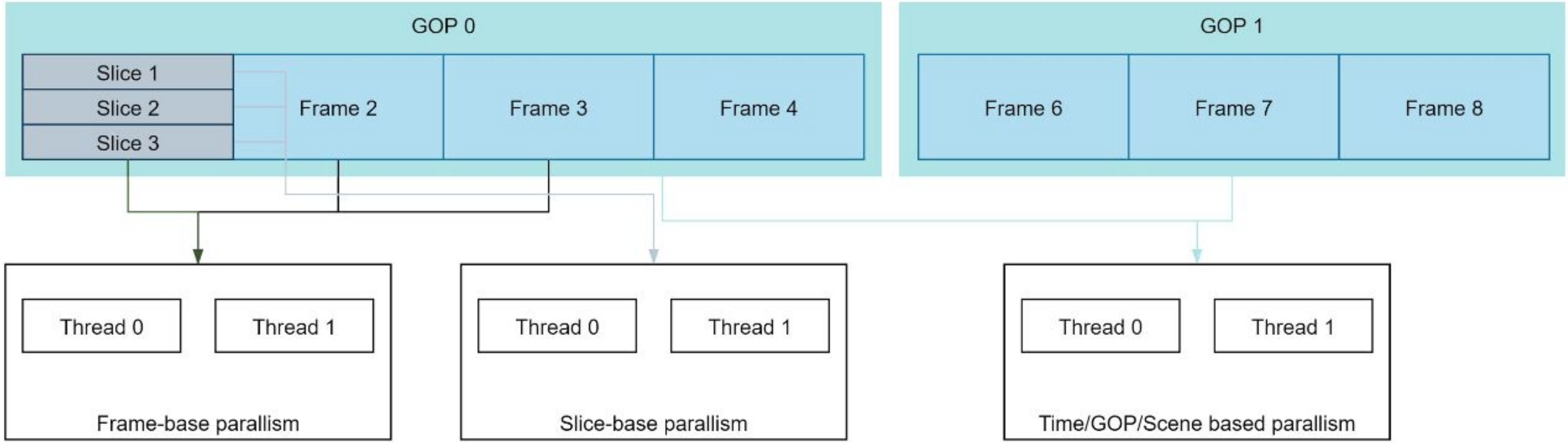
Let's use all together in parallel !!!

Encoder Parallelization types

Parallelism	Pros and Cons
Frame level	Can only be used for certain GOP structures (non b-frames)
Slice level	Scaling limited to a single frame
Tile level	Similar parallelization level as slice level
Wavefront Parallel Processing (WPP)	Only available in HEVC, VVC…
Time-slicing (fixed duration)	Sub-optimal rate-control due to IDR frames forced at a given internal
Time-slicing (scene/GOP)	Extremely high parallelization, constant quality per scene, convex-hull encoding

Encoder Parallelization types



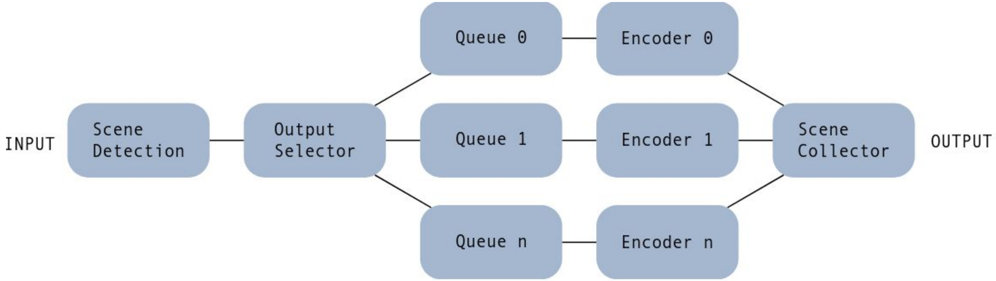


Fluendo Hype: Hybrid parallel encoder

- Hype is a Meta encoder
- It’s codec agnostic and can support a wide variety of codecs.
- It’s hybrid, supporting hardware and software encoders.
- It can use all the resources of a machine, mixing GPUs and CPU
- It parallelizes VOD encoding increasing encoding speeds.
- Based on time-slicing parallelization (fixed duration and scene/GOP)

Fluendo Hype: Hybrid parallel encoder

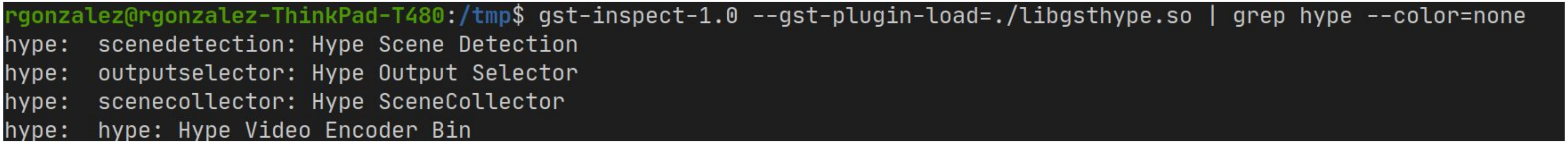
- Hype is a Meta encoder
- It’s codec agnostic and can support a wide variety of codecs.
- It’s hybrid, supporting hardware and software encoders.
- It can use all the resources of a machine, mixing GPUs and CPU
- It parallelizes VOD encoding increasing encoding speeds.
- Based on time-slicing parallelization (fixed duration and scene/GOP)



Similar ideas in the market

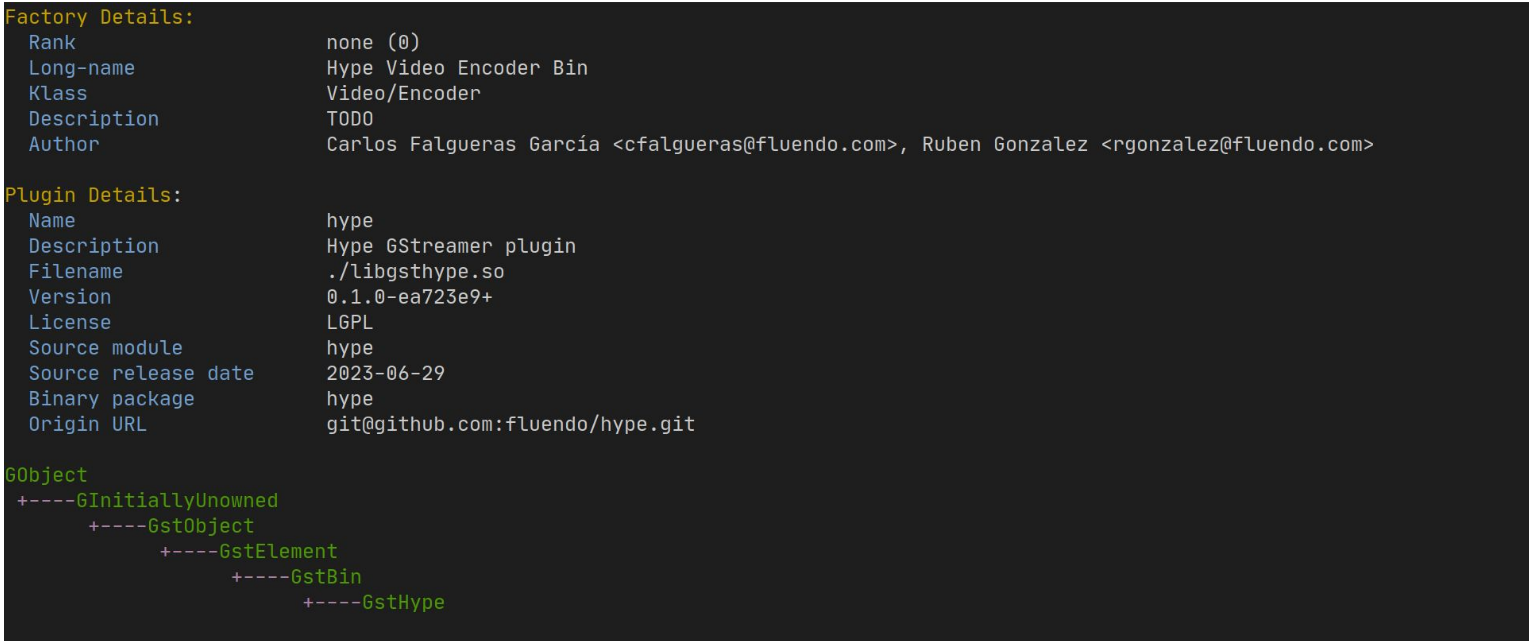
- Multi-GPU video encoding, transcoding and processing library from multicamera.systems
- Intel® Deep Link Hyper Encode (link)

GStreamer elements



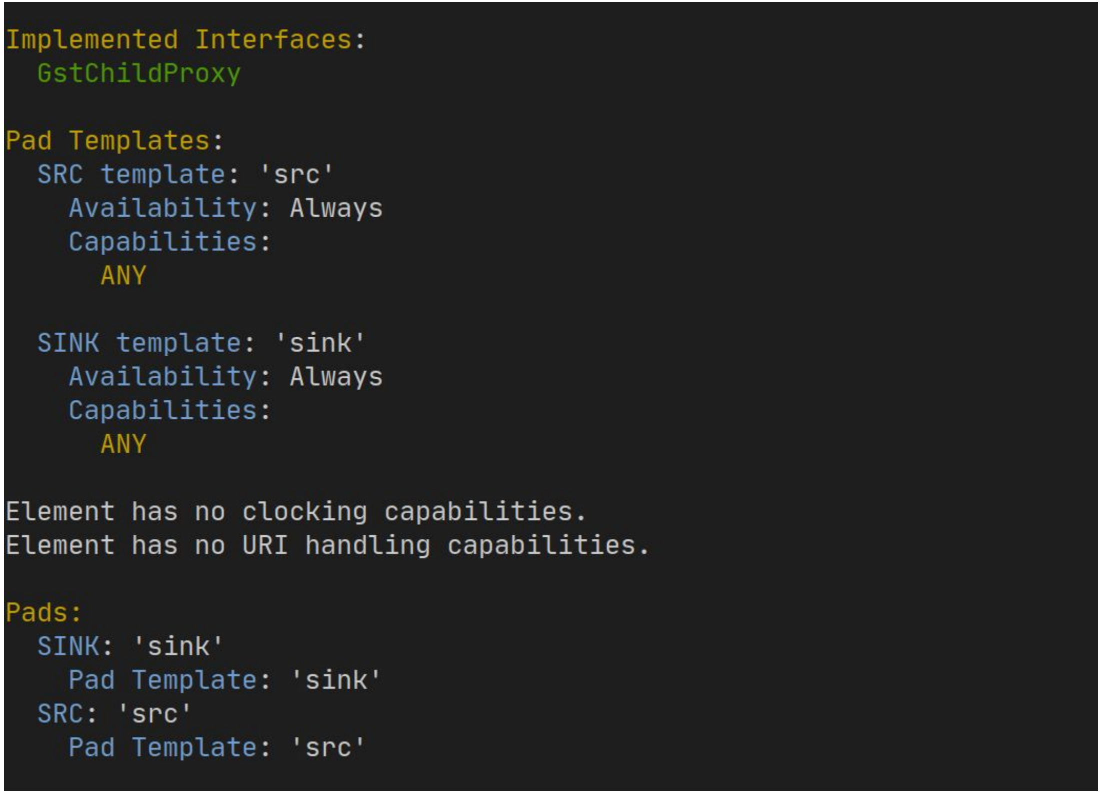
```
rgonzalez@rgonzalez-ThinkPad-T480:/tmp$ gst-inspect-1.0 --gst-plugin-load=./libgsthype.so | grep hype --color=none
hype:  scenedetection: Hype Scene Detection
hype:  outputselector: Hype Output Selector
hype:  scenecollector: Hype SceneCollector
hype:  hype: Hype Video Encoder Bin
```

gst-inspect output



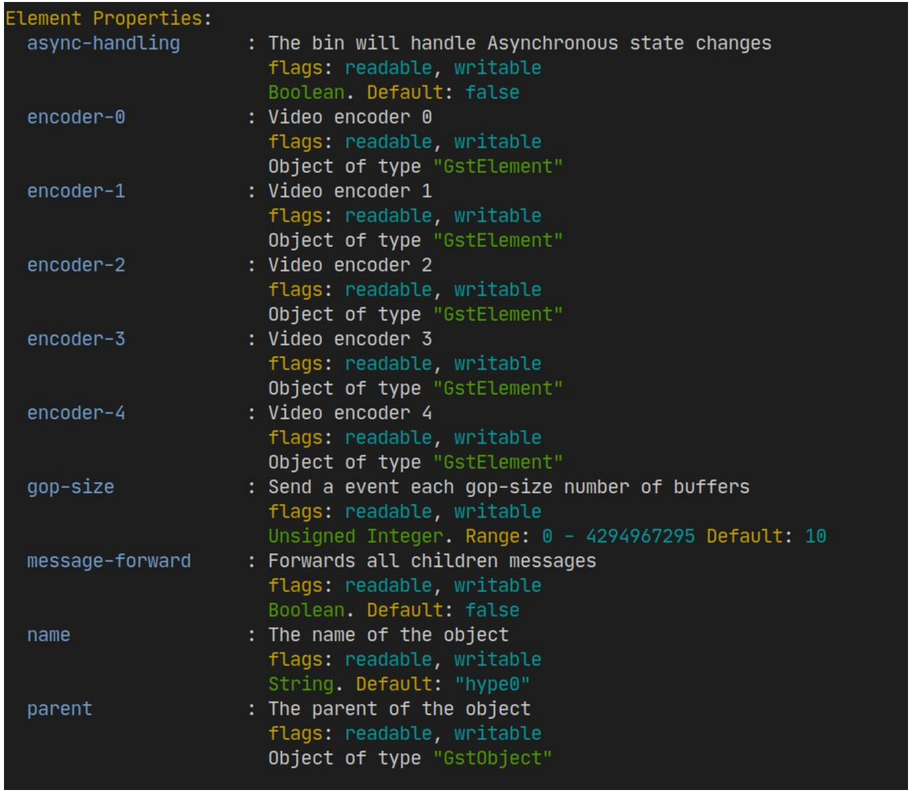
```
Factory Details:
  Rank                     none (0)
  Long-name                Hype Video Encoder Bin
  Klass                    Video/Encoder
  Description              TODO
  Author                   Carlos Falgueras García <cfalgueras@fluendo.com>, Ruben Gonzalez <rgonzalez@fluendo.com>

Plugin Details:
  Name                     hype
  Description              Hype GStreamer plugin
  Filename                 ./libgsthype.so
  Version                  0.1.0-ea723e9+
  License                  LGPL
  Source module            hype
  Source release date      2023-06-29
  Binary package           hype
  Origin URL               git@github.com:fluendo/hype.git

GObject
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstBin
                         +----GstHype
```

gst-inspect output



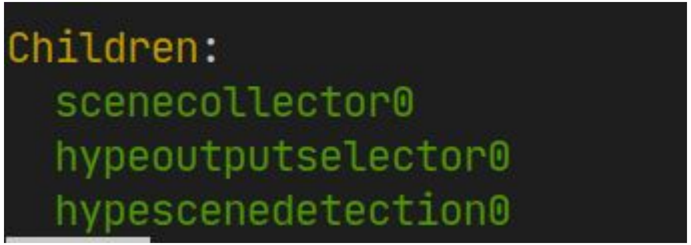
```
Implemented Interfaces:
  GstChildProxy

Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      ANY

  SINK template: 'sink'
    Availability: Always
    Capabilities:
      ANY

Element has no clocking capabilities.
Element has no URI handling capabilities.

Pads:
  SINK: 'sink'
    Pad Template: 'sink'
  SRC: 'src'
    Pad Template: 'src'
```

gst-inspect output

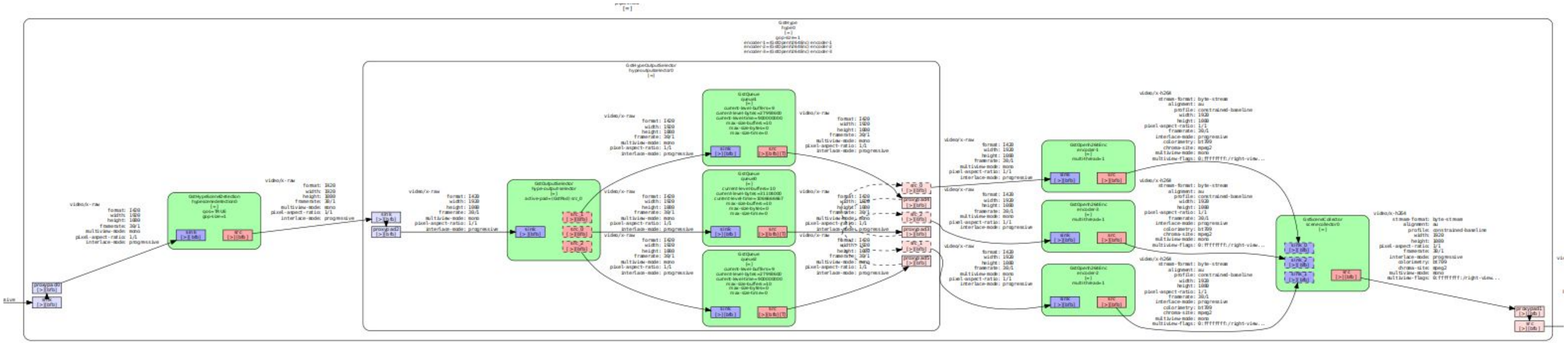


```
Element Properties:
  async-handling      : The bin will handle Asynchronous state changes
                        flags: readable, writable
                        Boolean. Default: false
  encoder-0           : Video encoder 0
                        flags: readable, writable
                        Object of type "GstElement"
  encoder-1           : Video encoder 1
                        flags: readable, writable
                        Object of type "GstElement"
  encoder-2           : Video encoder 2
                        flags: readable, writable
                        Object of type "GstElement"
  encoder-3           : Video encoder 3
                        flags: readable, writable
                        Object of type "GstElement"
  encoder-4           : Video encoder 4
                        flags: readable, writable
                        Object of type "GstElement"
  gop-size            : Send a event each gop-size number of buffers
                        flags: readable, writable
                        Unsigned Integer. Range: 0 - 4294967295 Default: 10
  message-forward     : Forwards all children messages
                        flags: readable, writable
                        Boolean. Default: false
  name                : The name of the object
                        flags: readable, writable
                        String. Default: "hype0"
  parent              : The parent of the object
                        flags: readable, writable
                        Object of type "GstObject"
```

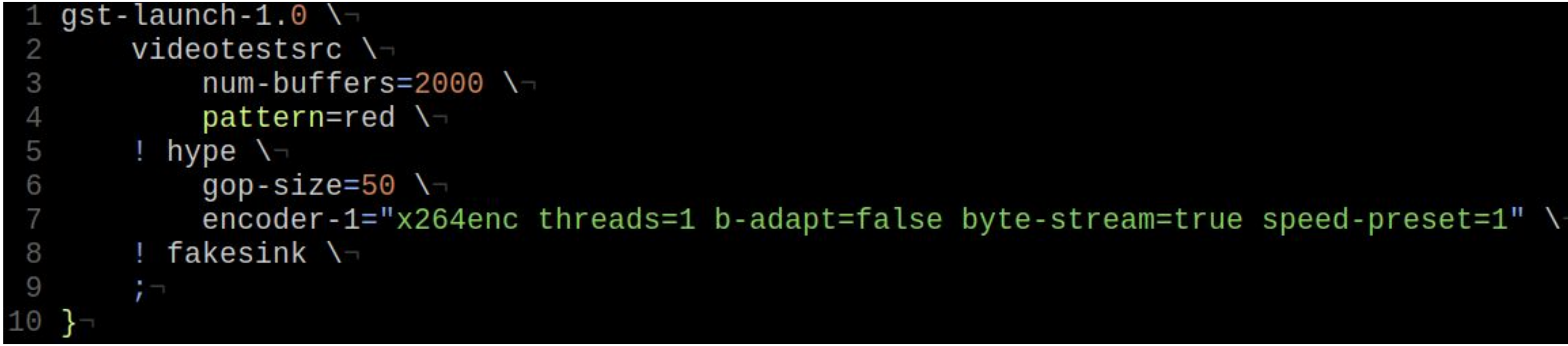
Hype graph



```
Children:
  scenecollector0
  hypeoutputselector0
  hypescenedetection0
```



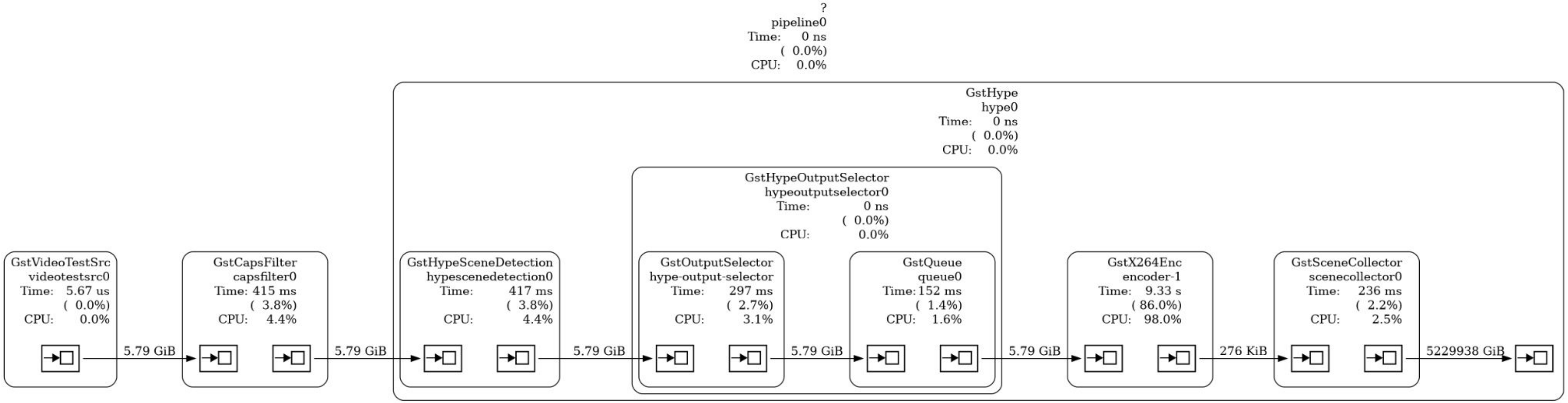
Hype with 1 encoder



```
gst-launch-1.0 \
    videotestsrc \
        num-buffers=2000 \
        pattern=red \
    ! hype \
        gop-size=50 \
        encoder-1="x264enc threads=1 b-adapt=false byte-stream=true speed-preset=1" \
    ! fakesink \
    ;
}
```

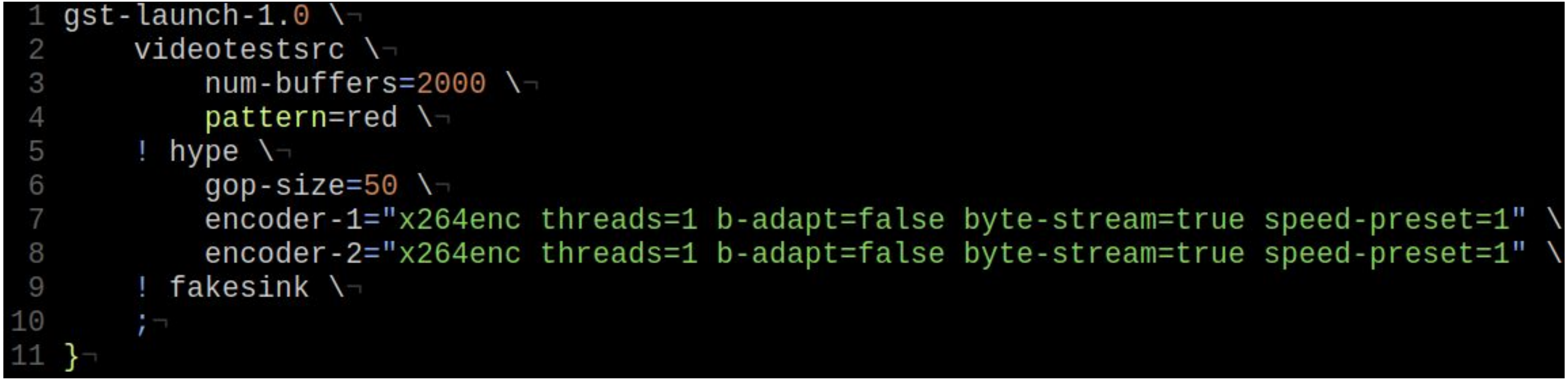
9.768s

Hype with 1 encoder



9.768s

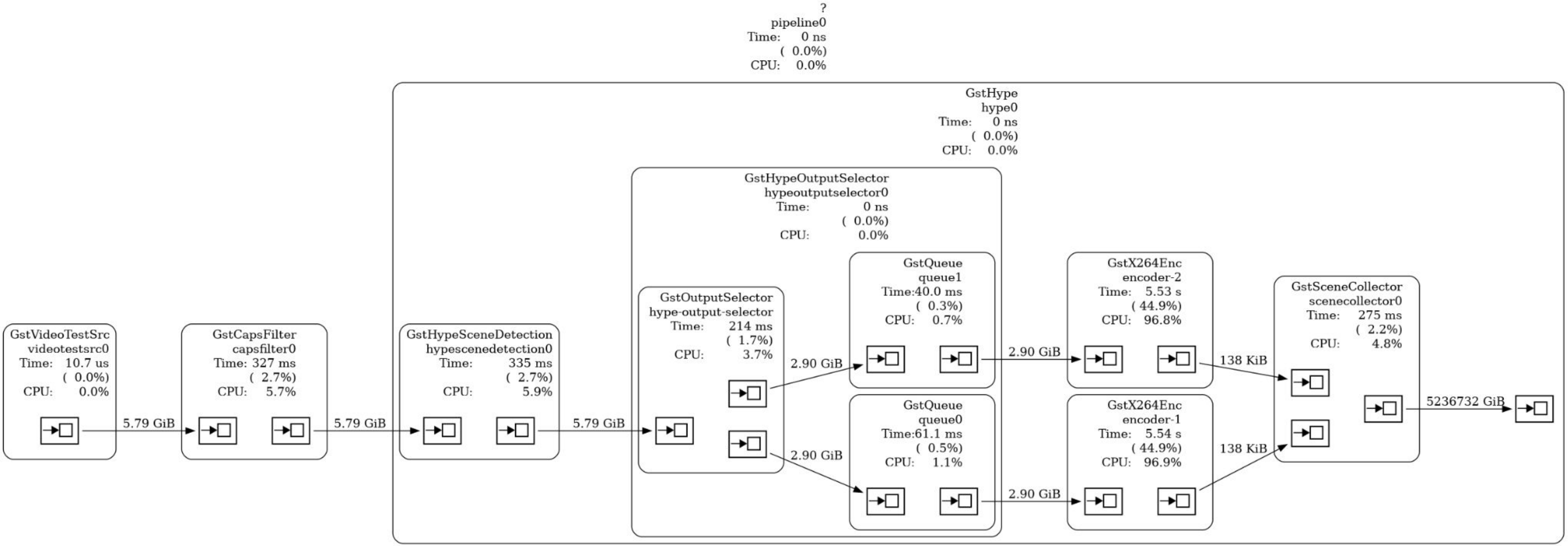
Hype with 2 encoders



```
gst-launch-1.0 \
    videotestsrc \
        num-buffers=2000 \
        pattern=red \
    ! hype \
        gop-size=50 \
        encoder-1="x264enc threads=1 b-adapt=false byte-stream=true speed-preset=1" \
        encoder-2="x264enc threads=1 b-adapt=false byte-stream=true speed-preset=1" \
    ! fakesink \
    ;
}
```

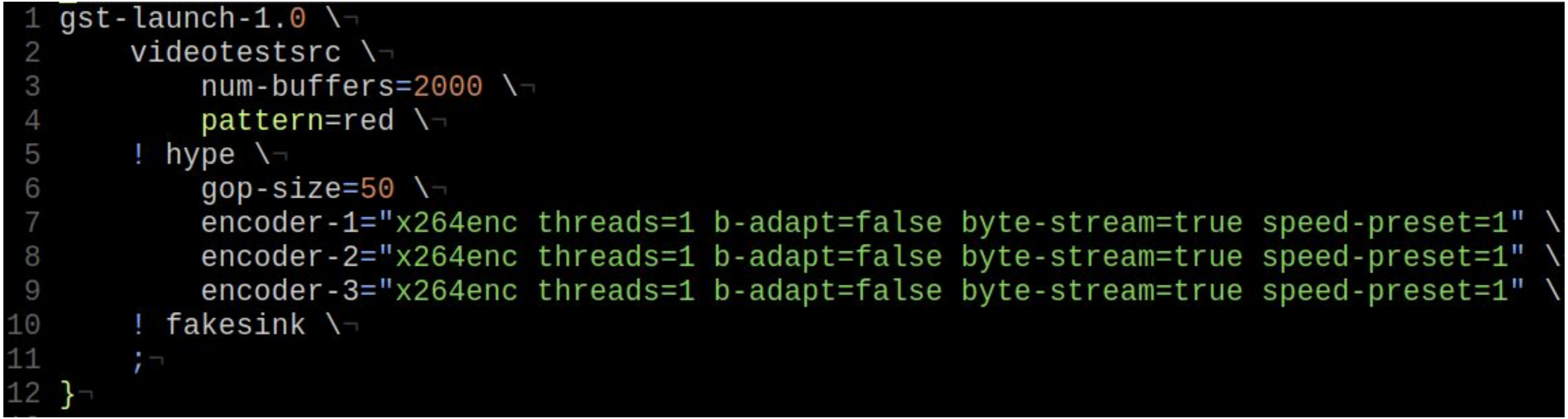
5.787s

Hype with 2 encoders



5.868s

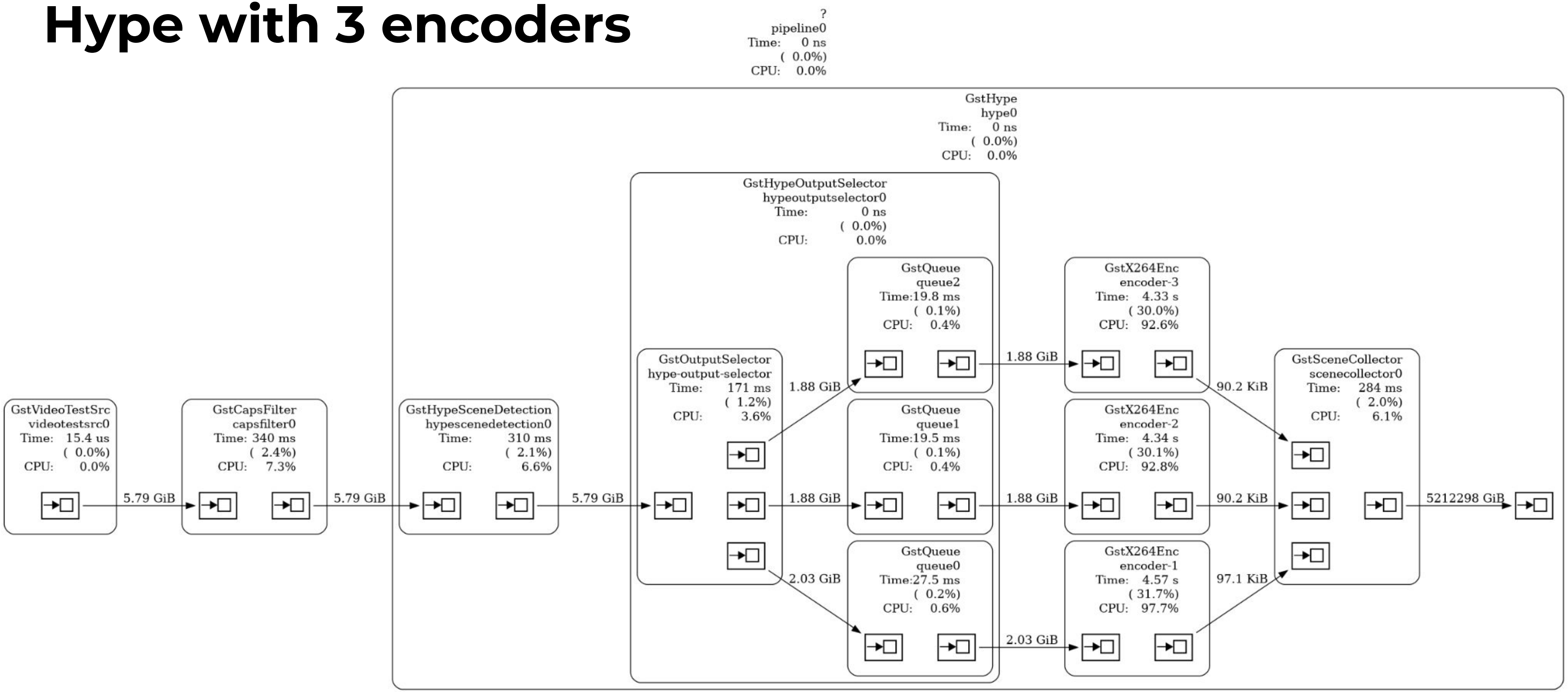
Hype with 3 encoders



```
gst-launch-1.0 \¬
    videotestsrc \¬
        num-buffers=2000 \¬
        pattern=red \¬
    ! hype \¬
        gop-size=50 \¬
        encoder-1="x264enc threads=1 b-adapt=false byte-stream=true speed-preset=1" \
        encoder-2="x264enc threads=1 b-adapt=false byte-stream=true speed-preset=1" \
        encoder-3="x264enc threads=1 b-adapt=false byte-stream=true speed-preset=1" \
    ! fakesink \¬
    ;¬
}¬
```

4.855s

Hype with 3 encoders



4.855s

Summary

with “openh264enc multi-thread=1”

# encs	Elapsed (wall clock) time	Percent of CPU this job got	Maximum resident set size (kbytes)
1	1:03.73	103%	80_612
2	0:40.89	202%	139_984
3	0:24.09	302%	199_552
4	0:20.34	395%	258_952

Conclusions

- Based on `force-keyunit` events: But not supported by all the GStreamer encoders.
- Equaling the output quality for different encoders may be very challenging.
- Transcoding better than only encoding
  - Less memory required.
  - Take advantage of zero-copy transcoder by GOP.
- Fluendo will Open Source the project during the hackfest.
- Implemented in Rust.

Conclusions

- Based on `force-keyunit` events: But not supported by all the GStreamer encoders.
- Equaling the output quality for different encoders may be very challenging.
- Transcoding better than only encoding
  - Less memory required.
  - Take advantage of zero-copy transcoder by GOP.
- Fluendo will Open Source the project during the hackfest.
- Implemented in Rust.

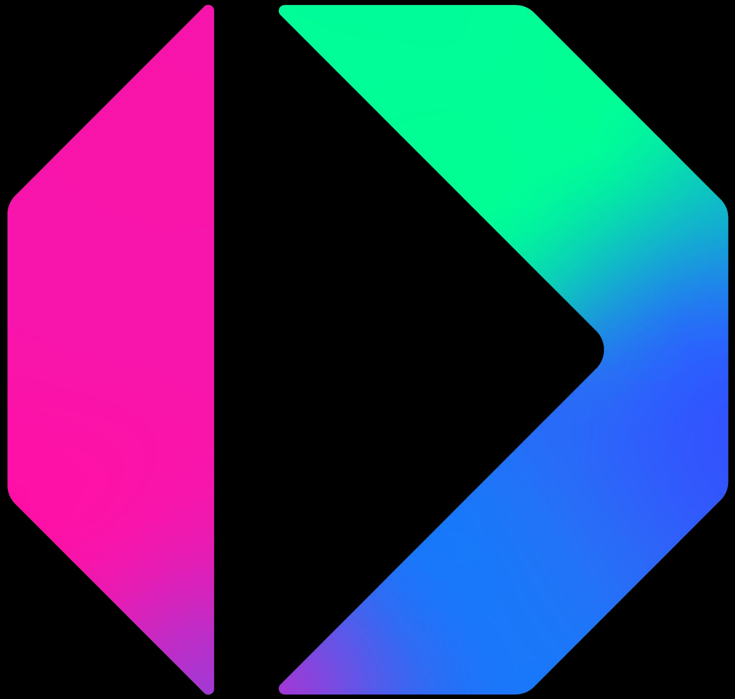
Acknowledgments

- Andoni and Nacho form Fluendo R&D team.
- Carlos Falgueras from Fluendo Eng team

Moitas Grazas

Questions time

fluendo



#161616

#F915AB

#00FD98

#2671F7