Introducción a GStreamer

Diego Nieto Muñoz @ Fluendo

21 de febrero de 2024

Quien soy?

- Senior multimedia engineer at Fluendo
  - Audio/Video codecs
  - Digital microscopes
  - Drones
- Mirada PLC
  - Native C++ with Qt Streaming player and FFmpeg
  - Integration of Netflix, Disney+ as embedded applications
  - Screencapture for videogames
- BSC-CNS
  - OpenCL and CUDA offloading algorithms
  - OpenMP and MPI parallelizations
- USC
- ULPGC

Agenda

- ¿Qué es GStreamer?
- ¿Quién lo usa?
- ¿Para qué lo podemos usar?
- Conceptos básicos de GStreamer y multimedia
- Cómo se hacen pipelines
- Cómo hacer nuestra primera aplicación
  - C
  - Python
- Dónde buscar referencias

¿Que es GStreamer?

- Multimedia framework
- Based on plugins
- Data agnostic
- Multiplatform

¿Quién lo usa?

Big companies:

- DELL
- HP
- IGEL
- Citrix
- HbbTV

Fluendo CS clients:

- Partner Electronics
- Vicon
- ScoutDI
- Brightsign
- TZ Electronics

¿Para qué lo podemos usar?

Use cases:

- Screencapture for videogames
- Microscopes
- Webcams
- Virtual environments
- Video streaming
- DVB TV
- ...

GStreamer History

- GStreamer 0.1-0.9 - since 1999
- GStreamer 0.10 - December 2005
- GStreamer 1.0 - September 24, 2012
  - HW dec/enc with GPU
  - Dynamic pipelines
  - Zero copy
  - API enhancements

Conceptos básicos de GStreamer y multimedia I

- video codecs: h265, h265, AV1
- audio codecs: AAC, MP3
- subtitles: closedcaptions, TTML
- images: jpeg, png
- containers: mp4, mpegts
- demuxers: qtdemux(mp4),tsdemux(ts)
- tcp streaming protocols: DASH, HLS
- udp protocols: udp, WebRTC
- formats: RGBA, YUV, NV12
- ...

Conceptos básicos de GStreamer y multimedia II

¿Qué es YUV?

- Y (Luminancia): Esta componente lleva la información de brillo de la imagen y representa la escala de grises. Es decir, determina la intensidad luminosa de un píxel y define el contraste y la claridad de la imagen.
- U (Crominancia azul-diferencia): Esta componente representa la diferencia entre la luminancia y la componente azul (B) de la imagen. Lleva información sobre la cantidad de color azul en la imagen.
- V (Crominancia rojo-diferencia): Similar a la componente U, la componente V representa la diferencia entre la luminancia y la componente roja (R) de la imagen. Lleva información sobre la cantidad de color rojo en la imagen.

Compresión con pérdida de información no apreciable

Conceptos básicos de GStreamer y multimedia III

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{array}{l} Y \in [0, 255] \\ U \in [-111, 111] \\ V \in [-157, 157] \end{array} \tag{F.10}$$

Figura: YUV conversion

Conceptos básicos de GStreamer y multimedia IV

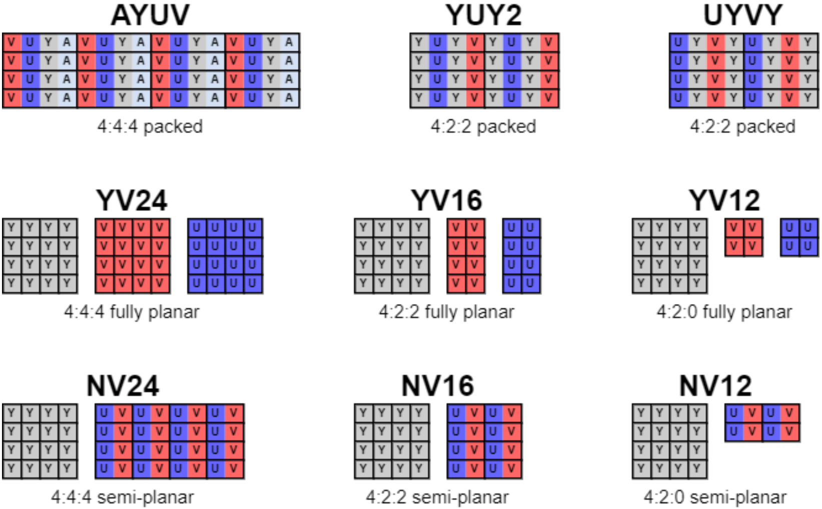

Figura: YUV compressed formats

GStreamer architecture

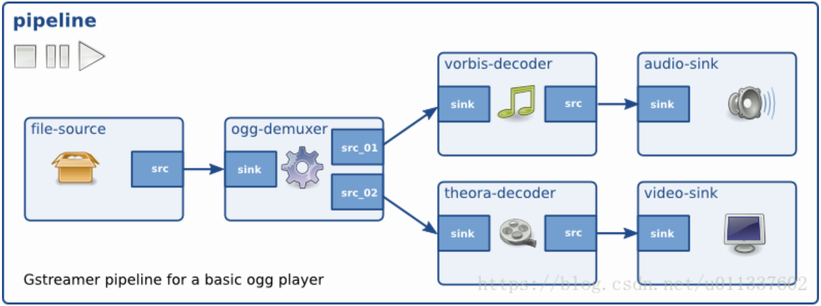

Figura: GStreamer architecture

GStreamer architecture - Elements

- The most important object in GStreamer for the application programmer
- An element is the basic building block for a media pipeline
- Normally an element has one specific function
- By chaining together several elements, you create a pipeline that can do a specific task, for example media playback or recording
- Can be visualized as black boxes - on the one end, you might put something in, the element does something with it and something else comes out at the other side.

GStreamer architecture - Main elements

- Source elements generate data for use by a pipeline
- Source elements do not accept data, they only generate data
- Sink elements are end points in a media pipeline.
- They accept data but do not produce anything.
- Filters and filter-like elements (convertors, demuxers, muxers and encoders/decoders have both input and outputs pads)

GStreamer architecture - Pads

- The pads are the element's interface to the outside world
- GStreamer defines two pad directions: source pads and sink pads
- Data streams from one element's source pad to another element's sink pad
- The specific type of media that the element can handle will be exposed by the pad's capabilities (caps)

GStreamer architecture - Caps

- Caps - capabilities of a pad
- Describe the type of data that is streamed between two pads, or that one pad (template) supports
- Purposes:
  - Autoplugging
  - Compatibility detection
  - Metadata
  - Filtering

GStreamer architecture - Bins and pipelines

- A bin is a container for a collection of elements
- Bins are elements
- A pipeline is a top-level bin. It provides a bus for the application and manages the synchronization for its children
- (A bus is a simple system that takes care of forwarding messages from the streaming threads to an application in its own thread context.)

GStreamer architecture - communication

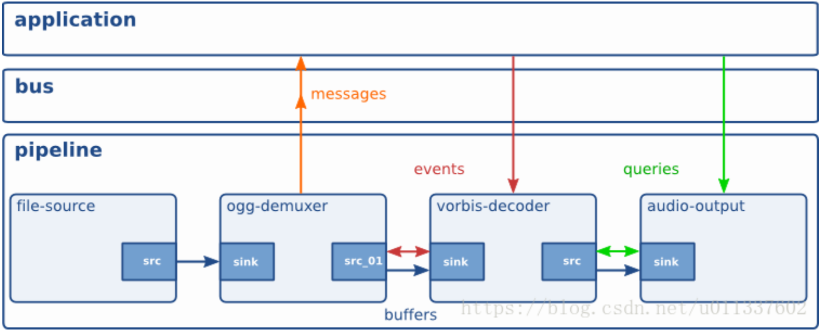

Figura: GStreamer architecture

GStreamer architecture - communication

- Downstream and upstream communication
- Buffers - objects for passing streaming data between elements in the pipeline (downstream)
- Events - objects sent between elements or from the application to elements (downstream/upstream)
- Messages - objects posted by elements on the pipeline's message bus, where they will be held for collection by the application (eof, errors, tags, state changes, bufferingstate, redirects etc.)
- Queries - allow applications to request information such as duration or current playback position from the pipeline (downstream/upstream)

GStreamer architecture - playback states

- NULL: This is the initial state of an element.
- READY: The element should be prepared to go to PAUSED.
- PAUSED: The element should be ready to accept and process data. Sink elements, however, only accept one buffer and then block.
- PLAYING: The same as PAUSED except for live sources and sinks. Sinks accept and render data. Live sources produce data.

Cómo se hacen pipelines I

- gst-launch-1.0: pipeline to run
  - source element: file, webcam, network
  - n elements: filter, demuxer
  - sink: display, file
- gst-inspect-1.0: plugins inspection
- gst-discoverer-1.0: media analysis

Cómo se hacen pipelines II

```
gst-launch-1.0 videotestsrc num-buffers=1 !
    "video/x-raw,width=640,height=480,format=I420"
    ! jpegenc ! filesink location=test.jpeg
gst-launch-1.0 filesrc location= test.jpeg !
    jpegdec ! video/x-raw,format=I420 ! filesink
    location = test-i420
gst-launch-1.0 filesrc location= test-i420 !
    videoparse format=i420 width=640 height=480
    ! imagefreeze ! decodebin ! videoconvert !
    ximagesink
gst-launch-1.0 filesrc location= test.jpeg !
    jpegdec ! videoconvert !
    video/x-raw,format=RGB ! filesink location =
    test-rgb
gst-launch-1.0 filesrc location= test-rgb !
    videoparse format=rgb width=640 height=480 !
    imagefreeze ! decodebin ! videoconvert !
    ximagesink
```

¿Existen alternativas?

FFmpeg

- ¿Qué tiene GStreamer que no tiene FFmpeg?
- ¿Para que se usa FFmpeg?

Cómo hacer nuestra primera aplicación

Ejercicios!

¿Dónde buscar referencias?

- Basic tutorials
- Playback tutorials
- Plugins Writer's guide

¿Preguntas?

QA?