GStreamer troubleshooting grid

Contents

1. GStreamer troubleshooting grid 3
2. Category:GStreamer 3
3. Category:Troubleshooting grids 4
4. How to profile video framerate 5

Stable: 23.01.2020 - 10:51 / Revision: 26.11.2019 - 09:44

Stable: 23.01.2020 - 10:01 / Revision: 20.11.2019 - 09:44

A quality version of this page, approved on *23 January 2020*, was based off this revision.

Some typical issues related to the **GStreamer** framework are listed below. Solutions or debugging methods are proposed for these issues.

If your issue is not listed, try also looking in the articles in the Gstreamer or troubleshooting grids categories.

Symptom	Resolution
`There may be a timestamping problem, or this computer is too slow.`	This message is typically displayed when the GStreamer A/V sync drops some frames. This may be due to a decoder that is too slow, badly timestamped buffers, or a badly formed container (.ts, .3gp, .mp4, .avi, ...). To proceed with investigations, refer to the article How to profile video framerate.
`Could not create Wayland display`	HDMI TV not connected, please connect it.
`Could not load BMP file`	Make sure the loader of the bmp file format, libpixbufloader-bmp.so, is present in /usr/lib /gdk-pixbuf-2.0/2.10.0/. If not, remove /usr/lib /gdk-pixbuf-2.0/2.10.0/loaders.cache, copy libpixbufloader-bmp.so file from distrib build to /usr/lib/gdk-pixbuf-2.0/2.10.0/ then run "/usr/lib /gdk-pixbuf-2.0/gdk-pixbuf-query-loaders > /usr/lib/gdk-pixbuf-2.0/2.10.0/loaders.cache" to reload loaders.

High-Definition Multimedia Interface (HDMI standard)

Stable: 17.06.2020 - 15:26 / Revision: 16.01.2020 - 07:49

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to the Linux®**GStreamer** multimedia software framework.

Pages in category "GStreamer"

The following 10 pages are in this category, out of 10 total.

- Gst-discoverer
- Gst-play
- Gst-typefind
- GStreamer overview
- GStreamer troubleshooting grid
- How to get video details
- How to make a camera preview
- How to play a video
- How to profile video framerate
- How to stream camera over network

Stable: 17.06.2020 - 15:27 / Revision: 16.01.2020 - 13:51

A quality version of this page, approved on *17 June 2020*, was based off this revision.

This category groups together all articles related to a troubleshooting grid.

Pages in category "Troubleshooting grids"

The following 9 pages are in this category, out of 9 total.

- ALSA troubleshooting grid
- Audio troubleshooting grid
- Coprocessor management troubleshooting grid
- DRM KMS troubleshooting grid
- GPU troubleshooting grid
- GStreamer troubleshooting grid
- Networking troubleshooting grid
- Visual troubleshooting grid
- Wayland Weston troubleshooting grid

Stable: 24.09.2019 - 09:53 / Revision: 24.09.2019 - 09:52

A quality version of this page, approved on *24 September 2019*, was based off this revision.

This article aims to debug & profile framerate performances of any GStreamer video use-case, including camera preview or video playback use-cases.

Contents

1 Debugging framerate issues 6
1.1 Framerate of played content 6
1.2 Display framerate 6
1.3 Check default traces 7
1.4 Frame drop traces 8
1.5 Frame drop due to display subsytem 8
2 Profiling framerate: fpsdisplaysink 10
3 Disabling frame synchronisation 14

1 Debugging framerate issues

A variety of symptoms related to framerate issues may be observed such as:

- Jerky video
- Video freeze
- Excessively slow Video framerate (typically one frame per second, see below)
- Too slow or too fast video motion
- Audio & video not synchronized
- ...

When such symptoms are observed, one can check below chapters to ease investigations and analysis of the problems.

1.1 Framerate of played content

Before investigating possible framerate issues, check the expected multimedia content framerate.

For a video, refer to gst-discoverer to get the video file framerate:

```
Board $> gst-discoverer-1.0 <my video> -v | grep -i "Frame rate"
     Frame rate: 30/1
```

For this video, the framerate is 30 fps (**f**rames **p**er **s**econd).

For a camera preview use-case, the framerate is set in the pipeline:

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=1280, Height=720, framerate=
(fraction)15/1" ! queue ! autovideosink -e
```

Here the expected framerate is 15 fps.

1.2 Display framerate

When an animation is running on the display, the related framerate can be monitored from the display driver level thanks to the command:

```
Board $> (while true; do export fps=`cat /sys/kernel/debug/dri/0/state | grep fps -m1 |
grep -o '[0-9]\+'`; echo display ${fps}fps; sleep 4; done) &
```

The display framerate is then periodically output in the user console in "fps" (**f**rames **p**er **s**econd):

```
display 50fps
display 50fps
display 50fps
```

Notes:

- Stop monitoring the framerate with the command "kill -9 `ps -o ppid= -C sleep`".

- Adjust the framerate update period by modifying the "sleep" value (4 seconds in the example).
- Use the command "dmesg -n8" to mix both user and kernel console outputs.
- Debugfs configuration needs to be enabled.

It should conform to the expected framerate for the played content. If it is not the case, continue investigations with next chapters.

1.3 Check default traces

By default, the traces show Warnings when the GStreamer video sink receives a **lot of late buffers**:

```
WARNING: from element /GstPipeline:pipeline0/[...] A lot of buffers are being dropped.
```

In that case, GStreamer falls into a recovery mode consisting into displaying **one frame every second**.

If such warning is displayed, it means that some elements before the video sink are not fast enough to sustain the targeted framerate.

Here is a test pipeline illustrating this behaviour:

```
Board $> gst-launch-1.0 videotestsrc ! "video/x-raw, framerate=(fraction)200/1" !
autovideosink
```

```
Setting pipeline to PAUSED ...
Pipeline is PREROLLING ...
Pipeline is PREROLLED ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
WARNING: from element /GstPipeline:pipeline0/GstWaylandSink:waylandsink0: A lot of
buffers are being dropped.
Additional debug info:
../../../../git/libs/gst/base/gstbasesink.c(2901): gst_base_sink_is_too_late ():
/GstPipeline:pipeline0/GstWaylandSink:waylandsink0:
There may be a timestamping problem, or this computer is too slow.
WARNING: from element /GstPipeline:pipeline0/GstWaylandSink:waylandsink0: A lot of
buffers are being dropped.
Additional debug info:
../../../../git/libs/gst/base/gstbasesink.c(2901): gst_base_sink_is_too_late ():
/GstPipeline:pipeline0/GstWaylandSink:waylandsink0:
There may be a timestamping problem, or this computer is too slow.
WARNING: from element /GstPipeline:pipeline0/GstWaylandSink:waylandsink0: A lot of
buffers are being dropped.
```

This test pipeline generates a pattern at 200fps which is not sustainable by the overall system, leading to frames coming to the video sink very late.

Warning

This trace only appears if *lot of frames* are dropped, see following chapters for finer grain analysis.

1.4 Frame drop traces

Fine grain information can be provided around frame drops by enabling video sink traces:

```
--gst-debug=basesink:6 2>&1 | grep drop
```

One trace message will be output for each frame dropped.

With the test pipeline:

```
Board $> gst-launch-1.0 -e videotestsrc ! "video/x-raw, framerate=(fraction)100/1" !
autovideosink --gst-debug=basesink:6 2>&1 | grep drop
```

```
0:00:01.135448709   585   0x1a5a60 DEBUG               basesink gstbasesink.c:3626:
gst_base_sink_chain_unlocked:<autovideosink0-actual-sink-wayland> buffer late, dropping
0:00:01.147437126   585   0x1a5a60 DEBUG               basesink gstbasesink.c:3626:
gst_base_sink_chain_unlocked:<autovideosink0-actual-sink-wayland> buffer late, dropping
0:00:01.157521667   585   0x1a5a60 DEBUG               basesink gstbasesink.c:3626:
gst_base_sink_chain_unlocked:<autovideosink0-actual-sink-wayland> buffer late, dropping
0:00:01.168611
```

If no traces are observed, the GStreamer synchronisation system has not dropped any frame.

Warning

Even if no drop is observed with this trace, frames could nevertheless be dropped by the video sink GStreamer element because of the display subsystem, see next chapter for more details.

.

1.5 Frame drop due to display subsytem

Frames could also be dropped by the video sink GStreamer element because of the display subsystem not being fast enough to sustain the incoming framerate.

This frame dropping strategy is specific to the selected video sink GStreamer element.

Here is a trace that can be enabled to show when wayland subsystem cannot sustain the incoming framerate, and consequently, drop frames:

```
 --gst-debug=waylandsink:6 2>&1 | grep -i "redraw pending"
```

Here is an example:

```
Board $> gst-launch-1.0 -v -e videotestsrc ! video/x-raw, format=I420, framerate=100/1 !
queue ! fpsdisplaysink sync=false video-sink=waylandsink -v --gst-debug=waylandsink:6
2>&1 | grep  -e "redraw pending" -e "dropped"
```

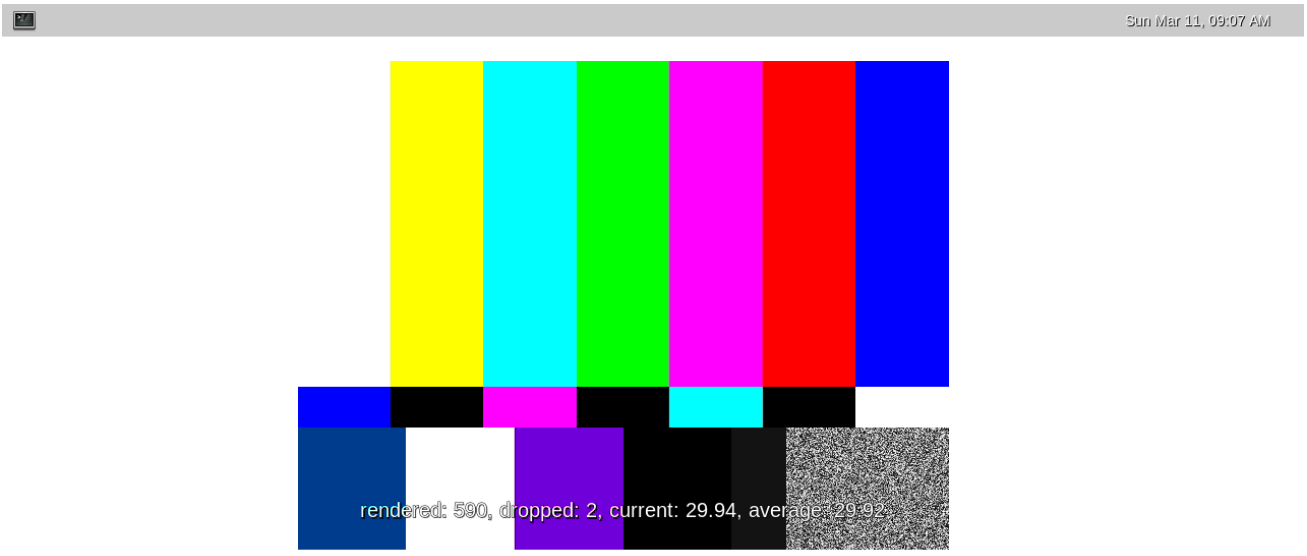
```
0:00:00.392392792  1020   0x19dd50 LOG              waylandsink gstwaylandsink.c:822:
gst_wayland_sink_show_frame:<waylandsink0> buffer 0xb510d860 dropped (redraw pending)
0:00:00.392863376  1020   0x19dd50 LOG              waylandsink gstwaylandsink.c:822:
gst_wayland_sink_show_frame:<waylandsink0> buffer 0xb510d900 dropped (redraw pending)
0:00:00.394748751  1020   0x19dd50 LOG              waylandsink gstwaylandsink.c:822:
gst_wayland_sink_show_frame:<waylandsink0> buffer 0xb510d9a0 dropped (redraw pending)
0:00:00.422420584  1020   0x19dd50 LOG              waylandsink gstwaylandsink.c:822:
gst_wayland_sink_show_frame:<waylandsink0> buffer 0xb510d900 dropped (redraw pending)
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 132, dropped: 0, current: 76.90, average: 87.14
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 132,
dropped: 0, current: 76.90, average: 87.14
```

In this example, the frames are generated fast enough to not be dropped because of their lateness, see trace message **"dropped: 0"**. This is the display subsystem that is responsible of not rendering frames fast enough, see trace message **"dropped (redraw pending)"**.

2 Profiling framerate: fpsdisplaysink

The framerate measurement can be done using **fpsdisplaysink** GStreamer element.

The measure is shown directly on the screen on a display overlay:



Available information:

- the number of rendered frames
- the number of dropped frames
- the current framerate
- the average framerate

fpsdisplaysink can replace any existing video sink into a GStreamer pipeline. To do so, replace:

```
gst-launch-1.0 [...] ! <current video sink>
```

with:

```
gst-launch-1.0 [...] ! fpsdisplaysink video-sink=<current video sink>
```

This could also be done with high level GStreamer bins such as **playbin**. To do so, replace:

```
gst-launch-1.0 playbin [...] video-sink=<current video sink>
```

with:

```
gst-launch-1.0 playbin [...] video-sink="fpsdisplaysink video-sink=<current video sink>"
```

Same can be done for high level GStreamer utility such as gst-play. To do so, replace:

```
gst-play-1.0 [...]
```

with:

```
gst-play-1.0 [...] --videosink="fpsdisplaysink video-sink=autovideosink"
```

Information could also be displayed in the console using the GStreamer "-v" verbose option. Here is a test pipeline illustrating this behaviour:

```
Board $> gst-launch-1.0 videotestsrc ! "video/x-raw, width=640, height=480, framerate=
(fraction)30/1" ! fpsdisplaysink video-sink=autovideosink -v
```

```
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 618, dropped: 3, fps: 25.04, drop rate: 1.93
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 618,
dropped: 3, fps: 25.04, drop rate: 1.93
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 629, dropped: 10, fps: 20.52, drop rate: 13.06
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 629,
dropped: 10, fps: 20.52, drop rate: 13.06
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 644, dropped: 10, current: 29.75, average: 29.55
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 644,
dropped: 10, current: 29.75, average: 29.55
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 660, dropped: 10, current: 30.19, average: 29.57
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 660,
dropped: 10, current: 30.19, average: 29.57
```

More options are also available on fpsdisplaysink, refer to the help:

```
Board $> gst-inspect-1.0 fpsdisplaysink
```

```
Factory Details:
  Rank                     none (0)
  Long-name                Measure and show framerate on videosink
  Klass                    Sink/Video
  Description              Shows the current frame-rate and drop-rate of the videosink as
overlay or text on stdout
  Author                   Zeeshan Ali <zeeshan.ali@nokia.com>, Stefan Kost <stefan.
kost@nokia.com>

Plugin Details:
  Name                     debugutilsbad
  Description              Collection of elements that may or may not be useful for
debugging
  Filename                 /usr/lib/gstreamer-1.0/libgstdebugutilsbad.so
```

```
  Version                  1.12.3
  License                  LGPL
  Source module            gst-plugins-bad
  Source release date      2017-09-18
  Binary package           GStreamer Bad Plug-ins source release
  Origin URL               Unknown package origin

GObject
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstBin
                         +----GstFPSDisplaySink

Implemented Interfaces:
  GstChildProxy

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


Element Flags:
  no flags set

Bin Flags:
  no flags set

Element Implementation:
  Has change_state() function: 0xb6a63d60

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

Pads:
  SINK: 'sink'

Element Properties:
  name                : The name of the object
                        flags: readable, writable
                        String. Default: "fpsdisplaysink0"
  parent              : The parent of the object
                        flags: readable, writable
                        Object of type "GstObject"
  async-handling      : The bin will handle Asynchronous state changes
                        flags: readable, writable
                        Boolean. Default: false
  message-forward     : Forwards all children messages
                        flags: readable, writable
                        Boolean. Default: false
  sync                : Sync on the clock (if the internally used sink doesn't have this
property it will be ignored
                        flags: readable, writable
                        Boolean. Default: true
  text-overlay        : Whether to use text-overlay
                        flags: readable, writable
                        Boolean. Default: true
  video-sink          : Video sink to use (Must only be called on NULL state)
                        flags: readable, writable
                        Object of type "GstElement"
  fps-update-interval : Time between consecutive frames per second measures and update
(in ms). Should be set on NULL state
                        flags: readable, writable
                        Integer. Range: 1 - 2147483647 Default: 500
  max-fps             : Maximum fps rate measured. Reset when going from NULL to READY.-1
```

```
means no measurement has yet been done
                        flags: readable
                        Double. Range:               -1 -   1.797693e+308
Default:               -1
  min-fps              : Minimum fps rate measured. Reset when going from NULL to READY.-1
means no measurement has yet been done
                        flags: readable
                        Double. Range:               -1 -   1.797693e+308
Default:               -1
  signal-fps-measurements: If the fps-measurements signal should be emited.
                        flags: readable, writable
                        Boolean. Default: false
  frames-dropped       : Number of frames dropped by the sink
                        flags: readable
                        Unsigned Integer. Range: 0 - 4294967295 Default: 0
  frames-rendered      : Number of frames rendered
                        flags: readable
                        Unsigned Integer. Range: 0 - 4294967295 Default: 0
  silent               : Don't produce last_message events
                        flags: readable, writable
                        Boolean. Default: false
  last-message         : The message describing current status
                        flags: readable
                        String. Default: null

Element Signals:
  "fps-measurements" :  void user_function (GstElement* object,
                                            gdouble arg0,
                                            gdouble arg1,
                                            gdouble arg2,
                                            gpointer user_data);
```

3 Disabling frame synchronisation

The GStreamer frame dropping mechanism (due to frame lateness) can be disabled using option "sync=false" applied on the video sink.

When frame dropping is disabled, all frames received by the video sink are sent to the display subsystem without any timestamp check.

In this case, the maximum framerate sustainable by the system can be reached.

Here are some typical GStreamer pipelines where video frame synchronization has been disabled:

```
Board $> gst-play-1.0  --videosink="autovideosink sync=false"
```

```
Board $> gst-play-1.0  --videosink="waylandsink sync=false"
```

```
Board $> gst-launch-1.0 playbin ...  video-sink="waylandsink sync=false"
```

```
Board $> gst-launch-1.0 filesrc ... ! waylandsink sync=false
```

```
Board $> gst-launch-1.0 filesrc ... ! kmssink sync=false
```

Using fpsdisplaysink with "sync=false" option allows to get the maximum sustainable framerate value.

Here is an example:

```
Board $> gst-launch-1.0 videotestsrc ! "video/x-raw, width=640, height=480, framerate=
(fraction)100/1" ! fpsdisplaysink sync=false video-sink="autovideosink" -v
```

```
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 51, dropped: 0, current: 22.33, average: 24.67
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 51,
dropped: 0, current: 22.33, average: 24.67
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 63, dropped: 0, current: 22.83, average: 24.30
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 63,
dropped: 0, current: 22.83, average: 24.30
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0/GstTextOverlay:fps-display-text-
overlay: text = rendered: 75, dropped: 0, current: 22.93, average: 24.07
/GstPipeline:pipeline0/GstFPSDisplaySink:fpsdisplaysink0: last-message = rendered: 75,
dropped: 0, current: 22.93, average: 24.07
```

In the above example, the maximum framerate is around 23fps while target is 100fps.

fourcc of YUV420 planar pixel format