Tracking memory leaks

GStreamer Conference 2016 - Berlin

Open First

COLLABORA

Dead to all leaks

- Increase memory consumption
- OOM
- Crashes

COLLABORA

Existing tools

COLLABORA

Valgrind: pros

- Most common tool
- Detect all type of leaks
- Full stack trace
- gst-validate support

COLLABORA

Valgrind: cons

- Low level
- Very slow
- CPU
- Tricky to use on some systems
- False positives
- “Noise”
- System/distro specific supp files

COLLABORA

Refcounting logs

- *GST_DEBUG=GST_REFCOUNTING:5*

COLLABORA

GST_TRACE

- Internal debugging tool (until 1.10)
- Track GstObject and GstMiniObject
- Fase positives
- Not great for QA tools

COLLABORA

GST_TRACE

GST_TRACE=all gst-launch-1.0 fakesrc num-buffers=1 ! fakesink

- GstMiniObject : 4
- GstCaps : (2) 0x56270cc344a0 ("ANY")
- GstCaps : (2) 0x56270cc34450 ("ANY")
- GstCaps : (1) 0x56270cad9050 ("EMPTY")
- GstCaps : (1) 0x56270cad9000 ("ANY")
- GstObject : 4
- GstTaskPool : (1) 0x56270ccdb430
- GstPadTemplate : (1) 0x56270cae1410
- GstPadTemplate : (1) 0x56270cae12f0
- GstAllocatorSysmem : (3) 0x56270cae3040
-

COLLABORA

gst-leaks

“Leaked” flag

- GST_OBJECT_FLAG_MAY_BE_LEAKED
- GST_MINI_OBJECT_FLAG_MAY_BE_LEAKED

GstTracer

- Tracing module loaded at run time
- Monitoring hooks
- Formatted output

Hooks

- Gst(Mini)Object created
- Gst(Mini)Object destroyed

COLLABORA

gstleaks

- Implemented as a GstTracer
- Track GstObject and GstMiniObject
- Raise a GLib warning if leaks are detected
- GST_DEBUG="GST_TRACER:7" GST_TRACERS="leaks" gst-launch-1.0 videotestsrc num-buffers=10 ! Fakesink

COLLABORA

Filtering support

- GST_DEBUG="GST_TRACER:7" GST_TRACERS="leaks(GstTaskPool,GstCaps)" gst-launch-1.0 videotestsrc num-buffers=10 ! Fakesink

- object-alive, type-name=(string)GstTaskPool, address=(gpointer)0xf3ed90, description=(string)<taskpool1>, ref-count=(uint)1, trace=(string);
- object-alive, type-name=(string)GstCaps, address=(gpointer)0x1899de0, description=(string)ANY, ref-count=(uint)2, trace=(string);

COLLABORA

Gstleaks: pro

- Much lighter/faster than valgrind
- Integrated in core (1.10)
- QA friendly
- Only track leaks in gst code
- No false positives
- Helped fixing loads of leaks

COLLABORA

Stack trace

- Libunwind (optional)
- GST_LEAKS_TRACER_STACK_TRACE=1 GST_DEBUG="GST_TRACER:7" GST_TRACERS="leaks" gst-launch-1.0 videotestsrc num-buffers=10 ! Fakesink
- object-alive, type-name=(string)GstCaps, address=(gpointer)0xc39cf0, description=(string)ANY, ref-count=(uint)2, trace=(string)handle_object_created.part.0
- gst_mini_object_init
- gst_caps_new_empty
- gst_caps_from_string
- gst_static_caps_get
- gst_static_pad_template_get
- gst_element_class_add_static_pad_template
- gst_fake_sink_class_intern_init
- g_type_class_ref
- g_object_newv
- gst_element_factory_create
- gst_element_factory_make
- priv_gst_parse_yyparse
- priv_gst_parse_launch
- gst_parse_launch_full
- gst_parse_launchv_full

COLLABORA

Signal support

- Borrowed from gobject-list
- GST_LEAKS_TRACER_SIG
- SIGUSR1: log alive objects
- SIGUSR2: log objects created/destroyed since the previous checkpoint

COLLABORA

Debugging leaks

- gst_deinit()
- Find the “top” leak
- Only Gst(Mini)Object are tracked
- Filter when using GST_LEAKS_TRACER_STACK_TRACE

COLLABORA

gdb

- Track refcount changes
  - b gst_mini_object_ref if (mini_object == 0xdeadbeef)
  - b gst_mini_object_unref if (mini_object == 0xdeadbeef)
  - commands 1 2
  - bt
  - cont
  - end
- Ignore GObject signal and property code paths

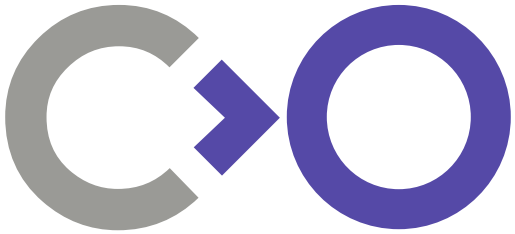
COLLABORA

Refcount transfers

- (transfer full) API
- Tricky to debug (GST_PAD_PROBE_HANDLED)

Future improvements

- Better stack trace (file + line)
- GUI tracking alive objects? gst-debugger?
- Suggestions?



Thank you!