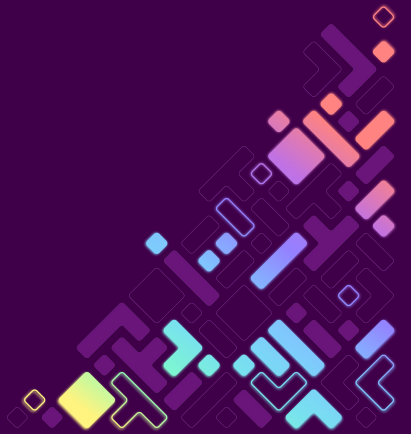
EMBEDDED LINUX CONFERENCE @ THE LINUX FOUNDATION OPEN SOURCE SUMMIT EUROPE

Advanced System Profiling, Tracing and Trace Analysis with Perfetto

in Android and Yocto

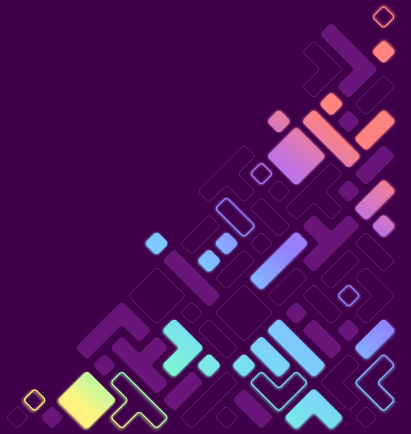


Anna-Lena Marx & Stefan Lengfeld, inovex

EMBEDDED LINUX CONFERENCE @ THE LINUX FOUNDATION OPEN SOURCE SUMMIT EUROPE

Advance~~d~~ **your** System Profiling, Tracing and Trace Analysis with Perfetto

in Android and Yocto



Anna-Lena Marx & Stefan Lengfeld, inovex

Anna-Lena Marx

Anna-Lena Marx

anna-lena.marx
@inovex.de

marx.engineer

Embedded Systems Developer

- since 2015 with inovex
- has a Master's degree in Embedded Systems
- studies Electrical Engineering as a hobby

Main Topics

- Embedded Systems
- Yocto Linux
- Linux Kernel
- AOSP/AAOS
- IoT

inovex

Stefan Lengfeld

Stefan Lengfeld

stefan.lengfeld@inovex.de

stefan.lengfeld.xyz

lengfeld

Android and Linux Embedded Developer

- since 2017 with inovex
- since 2014 a professional embedded software developer
- many more years a linux enthusiast

Main Topics

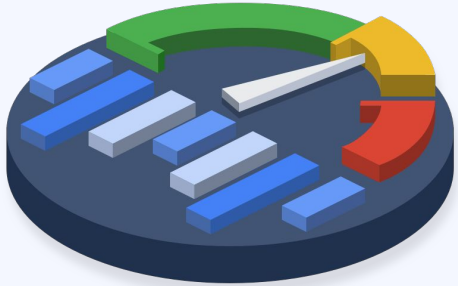
- Embedded Systems (Linux and Android)
- Linux Kernel
- Build systems
- Linux Graphics Stack

inovex

Agenda for today

- **Perfetto**
- **Record traces** *in Linux in general and Android*
- **Perfetto SDK** *add custom trace events in C++*
- **Analyzing** *with Perfetto UI (and command line tooling)*
- **Recap** *summarize it up!*

Perfetto



inovex

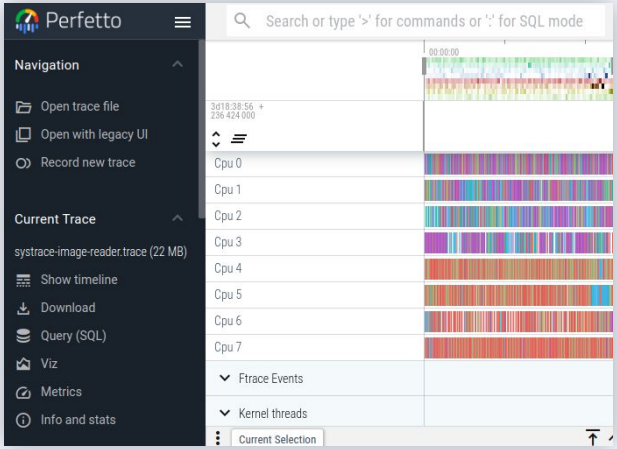
Why do we speak about an Android tool?

We learned to love ❤️ systrace in our Android Embedded Projects. Having the successor available for the Linux kernel, the system and application use-cases in general is great!

Thus, we want to share how perfetto and it’s powerful UI can advance tracing and analyzing.

From Catapult to Perfetto

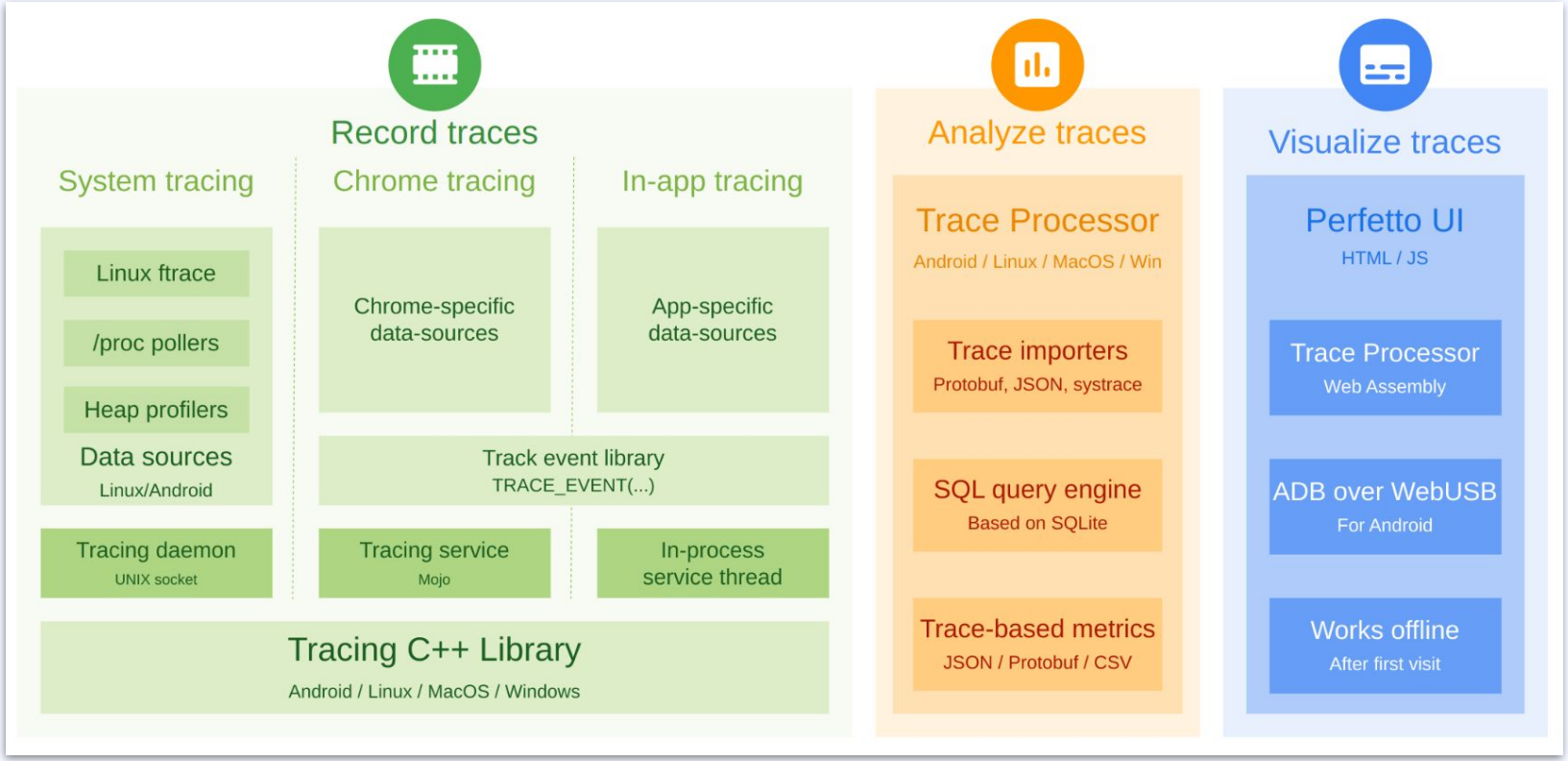
Catapult



Perfetto

inovex

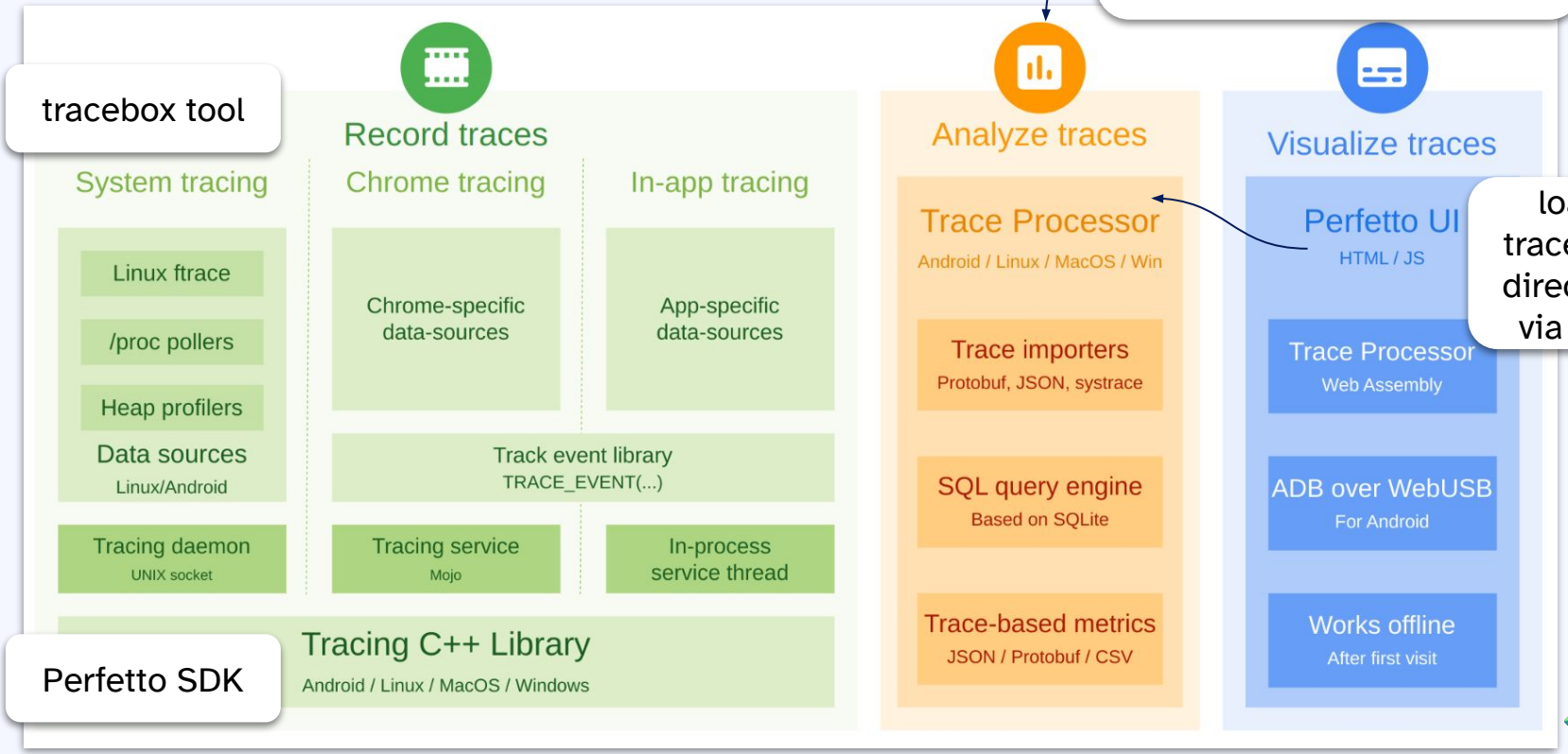
What is Perfetto?



https://perfetto.dev/docs/

inovex

What is Perfetto?



https://perfetto.dev/docs/

inovex

Record traces

in Linux

Setup on Yocto – install the tracebox tool

```
IMAGE_INSTALL:append = " perfetto”  # v31.0
```

v31.0 is from Nov 2023, current version is v47.0!

```
# strace, gdb and debug symbols
IMAGE_FEATURES:append = " tools-debug dbg-pkgs"
```

```
# oprofile, exmap, lttng, valgrid -> x86 only
IMAGE_FEATURES:append = " tools-profile"
```

Additional information:
https://perfetto.dev/docs/quickstart/linux-tracing
https://docs.yoctoproject.org/profile-manual/index.html

inovex

```
root@raspberrypi3-64:~# tracebox -t 10s -o trace_file.perfetto-trace sched/sched_switch
[041.744]          service.cc:239 Started traced, listening on @traced-p-493 @traced-c-493
[041.809]           probes.cc:104 Starting traced_probes service
[041.815]  probes_producer.cc:345 Connected to the service[   41.825220] tracebox[494]: memfd_create()
called without MFD_EXEC or MFD_NOEXEC_SEAL set

[041.822]     perfetto_cmd.cc:999 Connected to the Perfetto traced service, TTL: 10s
[041.832] ing_service_impl.cc:945 Configured tracing session 1, #sources:2, duration:10000 ms, #buffers:1,
total buffer size:32768 KB, total sessions:1, uid:0 session name: ""
```

```
if (!InferFtraceType(ftrace_field.type_and_name, ftrace_field.size,
                     ftrace_field.is_signed, &field->ftrace_type)) {
  PERFETTO_FATAL(
      "Failed to infer ftrace field type for \"%s.%s\" (type:\"%s\" "
      "size:%d "
      "signed:%d)",
      event_name_for_debug, field->ftrace_name,
```

```
124
125  if (!InferFtraceType(ftrace_field.ty
126                       ftrace_field.is
127    PERFETTO_DFATAL(
128        "Failed to infer ftrace field
129        "size:%d "
130        "signed:%d)",
131        event_name_for_debug, field->f
```

```
-----END PERFETTO PRE-CRASH LOG-----
[051.834] ng_service_impl.cc:1888 FlushAndDisableTracing(1) done, success=1
[051.835]    perfetto_cmd.cc:1144 Wrote 689 bytes into trace_file.perfetto-trace
```

https://android-review.googlesource.com/c/platform/external/perfetto/+/2583173

-> just patch with .bbappend!

inovex

Using tracebox

```
root@raspberrypi3-64:~# tracebox
Welcome to Perfetto tracing!

Tracebox is a bundle containing all the tracing services and the perfetto
cmdline client in one binary. It can be used either to spawn manually the
various subprocess or in "autostart" mode, which will take care of starting
and tearing down the services for you.

Usage in autostart mode:
  tracebox -t 10s -o trace_file.perfetto-trace sched/sched_switch
  See tracebox --help for more options.

Usage in manual mode:
  tracebox applet_name [args ...]  (e.g. ./tracebox traced --help)
  Applets: traced traced_probes perfetto trigger_perfetto websocket_bridge

See also:
  * https://perfetto.dev/docs/
  * The config editor in the record page of https://ui.perfetto.dev/
```

Using tracebox - autostart mode

```
Light configuration flags: (only when NOT using -c/--config)
  --time            -t      : Trace duration N[s,m,h] (default: 10s)
  --buffer          -b      : Ring buffer size N[mb,gb] (default: 32mb)
  --size            -s      : Max file size N[mb,gb]
                               (default: in-memory ring buffer only)
  --app             -a      :
  FTRACE_GROUP/FTRACE_NAME  :

cat /sys                                                    events
```

Caution:
<group>:<name> does not work!
use <group>/<name>

```
$ tracebox -t 10s -o trace_file.perfetto-trace sched/sched_switch
```


inovex

tracebox - trace configuration files

example.cfg

```
duration_ms: 10000

buffers {
  size_kb: 65536
  fill_policy: RING_BUFFER
}

data_sources {
  config {
    name: "linux.ftrace"
    target_buffer: 0
    ftrace_config {
      ftrace_events: "sched_switch"
      ftrace_events: "start_task_reaping"
      ftrace_events: "kmalloc"
    }
  }
}
```

```
tracebox --txt -c example.cfg -o example.perfetto-trace
```


inovex

tracebox - trace configuration files

```
duration_ms: 10000

buffers {
  size_kb: 65536
  fill_policy: RING_BUFFER
}

data_sources {
  config {
    name: "linux.ftrace"
    target_buffer: 0
    ftrace_config {
      ftrace_events: "sched_switch"
      ftrace_events: "start_task_reaping"
      ftrace_events: "kmem/kmalloc"
    }
  }
}
```

recording duration

at least one buffer needed!

RING_BUFFER or DISCARD (not recommended)

linux.ftrace, linux.sys_stats, linux.process_stats, linux.inode_file_map, linux.perf, ...

plus Android specific!

PBTX format! Pass with --txt flag!

Convert to binary format for benchmarking on different machines, ...

Examples:
https://github.com/google/perfetto/tree/main/test/configs

File schema:
https://github.com/google/perfetto/blob/main/protos/perfetto/config/trace_config.proto

https://perfetto.dev/docs/concepts/config

Record traces

in Android

Setup in Android

Much simpler than the Yocto/Linux setup!

- perfetto tools and daemon are already installed
- Enable adb on your device and visit ui.perfetto.dev

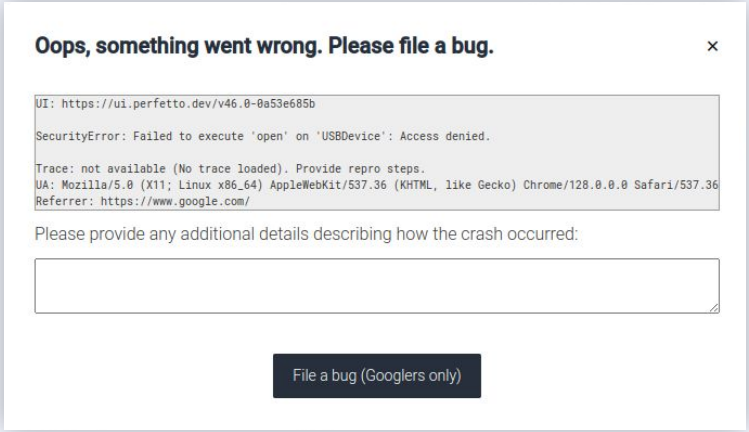
See documentation

https://perfetto.dev/docs/quickstart/android-tracing

```
# Needed only on Android 9 (P) and 10 (Q) on non-Pixel phones.
adb shell setprop persist.traced.enable 1
```

Perfetto and Chromium as snap

If the perfetto UI cannot connect
to your Android device,
you may see the error message:




```
SecurityError: Failed to execute ‘open’ on ‘USBDevice’:
Access denied.
```

Solution:

```
$ snap connect chromium:raw-usb
```


inovex

Record traces in Android

There are three ways to record traces

- via the Perfetto UI in the browser
- with the `perfetto` commandline tool on the device
- with the `record_android_trace` helper scripts

The python script systrace.py is not available anymore. See

> *This package used to contain systrace, but that has been obsoleted in favor of Studio Profiler, gpuinspector.dev, or Perfetto.*

Use `record_android_trace` instead.

See https://stackoverflow.com/a/74005757

Perfetto SDK

add custom trace events in C++ applications

Include SDK

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.13)
project(PerfettoExample)
find_package(Threads)

# Define a static library for Perfetto.
include_directories(perfetto/sdk)
add_library(perfetto STATIC perfetto/sdk

# Link the library to your main executab
add_executable(example example.cc)
target_link_libraries(example perfetto $
```

example.cc

```
#include <perfetto.h>

int main(int argc, char** argv) {
  perfetto::TracingInitArgs args;

  // The backends determine where trace events are recorded. You may select one
  // or more of:

  // 1) The in-process backend only records within the app itself.
  args.backends |= perfetto::kInProcessBackend;

  // 2) The system backend writes events into a system Perfetto daemon,
  //    allowing merging app and system events (e.g., ftrace) on the same
  //    timeline. Requires the Perfetto `traced` daemon to be running (e.g.,
  //    on Android Pie and newer).
  args.backends |= perfetto::kSystemBackend;

  perfetto::Tracing::Initialize(args);
}
```


https://perfetto.dev/docs/instrumentation/tracing-sdk

Track events

-> easiest way for custom events

- slices

Scenario	Runtime on Pixel 3 XL	Runtime
`TRACE_EVENT(...)` (disabled)	2 ns	1 ns
`TRACE_EVENT("cat", "name")`	285 ns	630 ns
`TRACE_EVENT("cat", "name", <lambda>)`	304 ns	663 ns
`TRACE_EVENT("cat", "name", "key", value)`	354 ns	664 ns
`DataSource::Trace(<lambda>)` (disabled)	2 ns	1 ns
`DataSource::Trace(<lambda>)`	133 ns	58 ns

- flows

  -> link two or more events and mark them as related

- counters

```
TRACE_COUNTER("category", "SheepCounter", 42);
```


inovex

```
#include "perfetto.h"

PERFETTO_DEFINE_CATEGORIES(
    perfetto::Category("console")
        .SetDescription("Interaction with the console, lik

PERFETTO_TRACK_EVENT_STATIC_STORAGE();
```

example.cc

```
void write_to_console() {
    TRACE_EVENT("console", "write_to_console");
    printf("ping\n");
}

int main() {
    perfetto::TracingInitArgs args;

    args.backends |= perfetto::kInProcessBackend;
    args.backends |= perfetto::kSystemBackend;

    perfetto::Tracing::Initialize(args);
    perfetto::TrackEvent::Register();

    printf("Example has started.\n");

    while (true) {
        sleep(1);
        write_to_console();
    }

    return 0;
}
```

example.cfg

```
data_sources {
  config {
    name: "track_event"
    track_event_config {
        enabled_categories: "console"
        disabled_categories: "ui"
    }
  }
}
```

Flows

```
uint64_t request_id = GetRequestId();

{
  TRACE_EVENT("rendering", "HandleRequestPhase1", perfetto::Flow::ProcessScoped(request_id));
}

std::thread t1([&] {
  TRACE_EVENT("rendering", "HandleRequestPhase2", perfetto::TerminatingFlow::ProcessScoped(request_id));
});
```

non-terminating

inovex

Custom data sources

Powerful tool in certain situations,
but needs corresponding **changes in trace processor**!

https://perfetto.dev/docs/instrumentation/tracing-sdk#custom-data-sources

Perfetto SDK for Android?

Answer: still use atrace!

> *For Android-only instrumentation, the advice is to keep using the existing **android.os.Trace (SDK) / ATrace_* (NDK)** if they are sufficient for your use cases. Atrace-based instrumentation is fully supported in Perfetto. See the Data Sources -> Android System -> Atrace Instrumentation for details."*

See https://perfetto.dev/docs/instrumentation/tracing-sdk
and https://perfetto.dev/docs/tracing-101

Analyzing

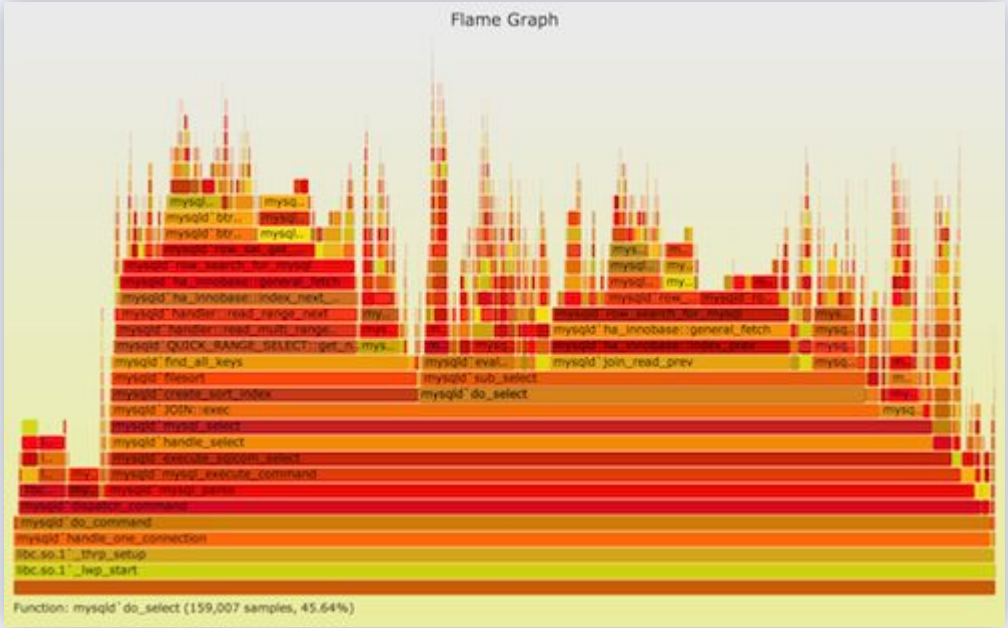
with Perfetto UI

(and command line tooling)

Perfetto – CPU flame graphs extended

You maybe know flamegraph from other profile tools:

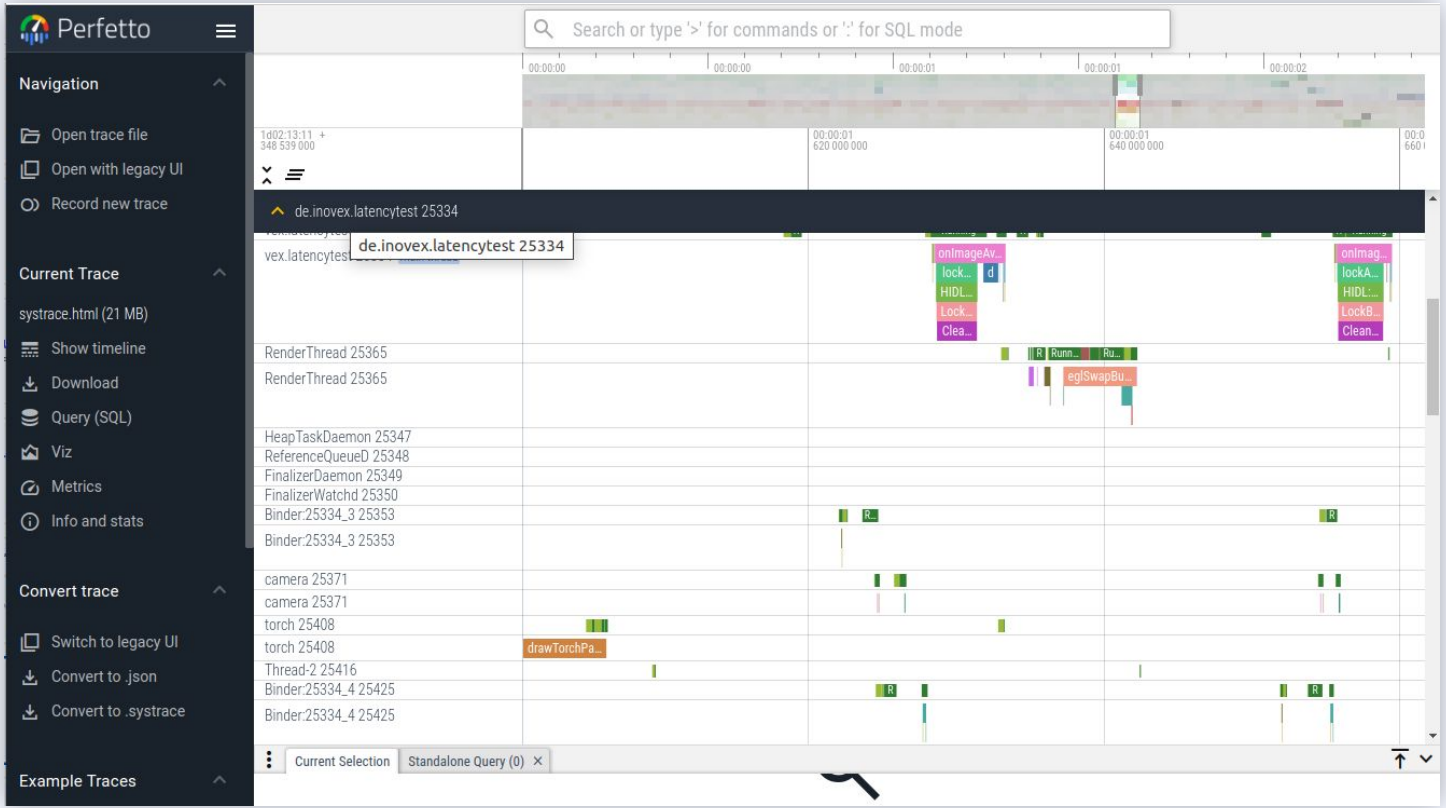
The Catapult TraceViewer and the Perfetto UI are a flame graphs on steroids!



https://www.brendangregg.com/FlameGraphs/cpuflamegraphs.html

inovex

Perfetto UI Overview

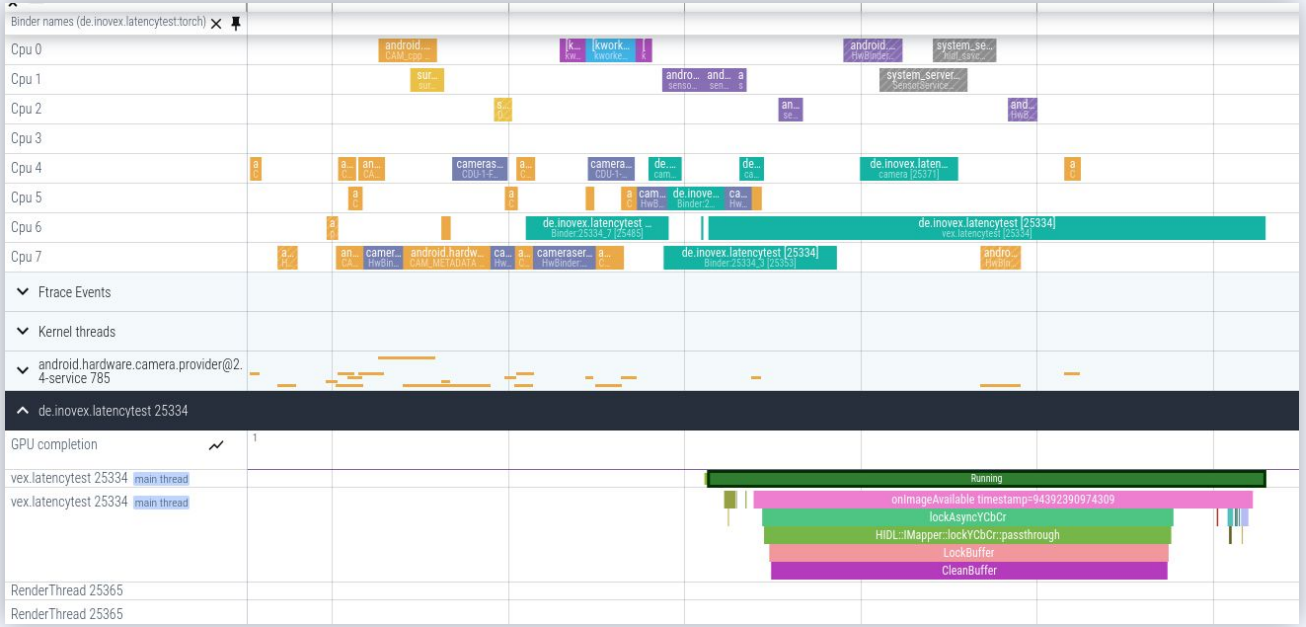


inovex

Everything as Catapult, but nicer!

Time

Track



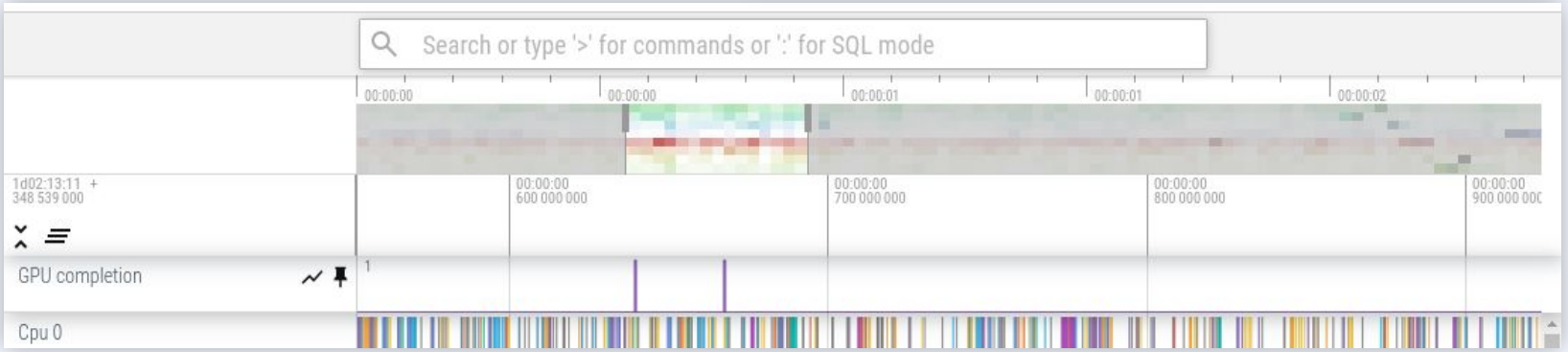
inovex

Keyboard shortcuts and navigation

- keyboard shortcuts
- mouse navigation
- track summary (*New!*)

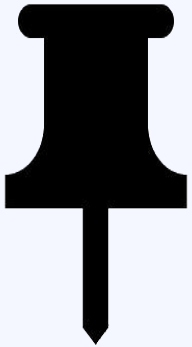
New!

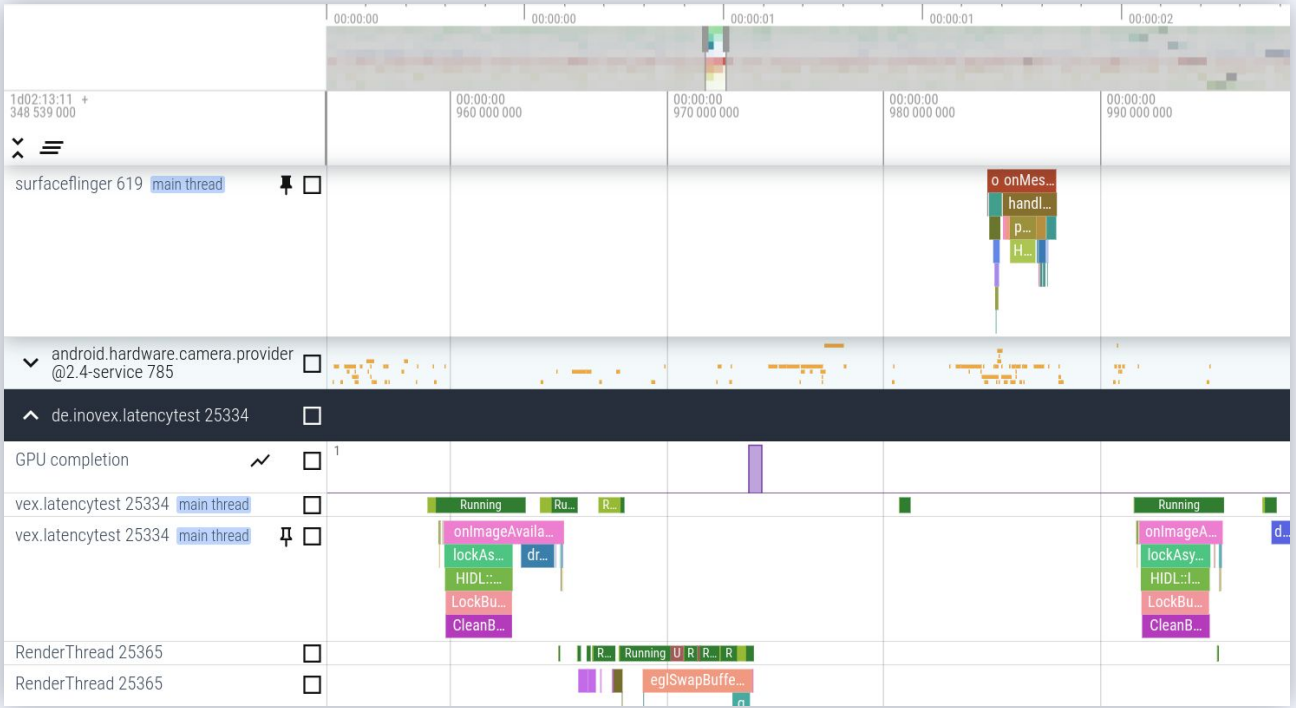
- New updated tabs are extensible and user friendly.
- Use W A S D to navigate the trace.
- Try the command palette, press Ctrl ⇧ P.



inovex

Pin threads to the top





inovex

SQL for trace events

inovex

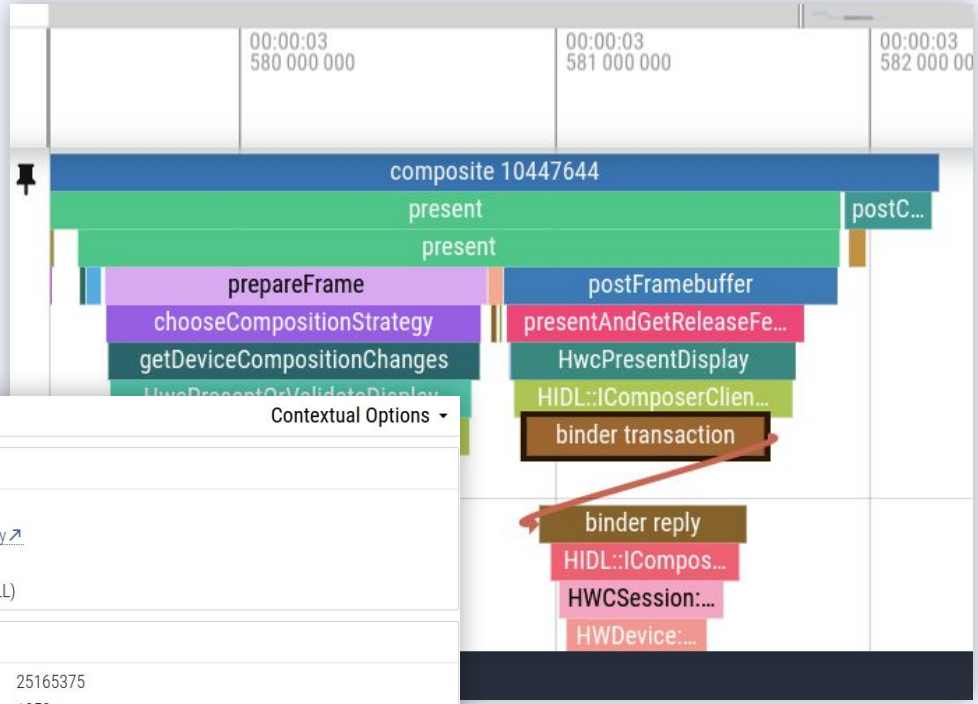
On the command line – the trace_processor

```
$ ./trace_processor systrace.html
[340.472] processor_shell.cc:1658 Trace loaded: 20.22 MB in 1.81s (11.2 MB/s)
> SELECT   min(dur/100000) as "dur in 100ns", count(dur) as count \
     FROM slice WHERE name like '%onImageAvailable%' GROUP BY dur/100000;
dur in 100ns      count
------------------- -------------------
                  13                   1
                  14                   1
                  15                   1
                  16                   1
                  17                   2
                  18                   1
                  20                   4
                  21                   3
                  22                   4
                  23                   5
                  24                   9
                  25                   4
                  26                   5
[...]
```

https://perfetto.dev/docs/quickstart/trace-analysis

Flows

Example: binder transactions
(IPC on the Android platform)



Slice binder transaction — Contextual Options

Details

Name	binder transaction
Category	binder
Start time	00:00:03.580 901 978
Absolute Time	2024-09-11T23:28:32.044507700
Duration	773us 958ns
Thread	708
Process	708
SQL ID	slice[32490]

Following Flows

Flow
- Slice: binder reply
- Delay: 0s
- Thread: NULL (NULL)

Arguments

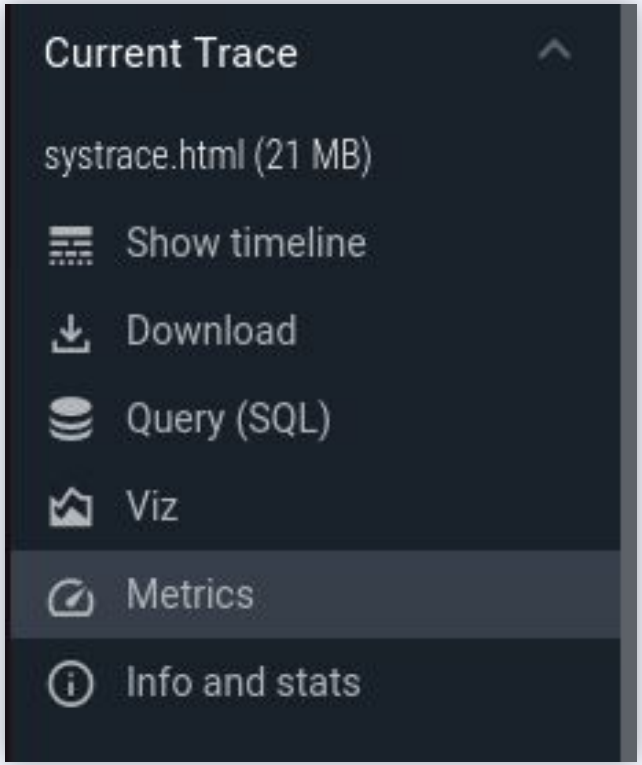
transaction id	25165375
destination node	1053
destination process	649
reply transaction?	false
flags	0x10 allow replies with file descriptors;
code	0x17 Java Layer Dependent
calling tid	708
data size	140
offsets size	16
destination thread	983

inovex

Many more things to discover!

Not yet tried:

- Viz
- Metric
- Info and stats



inovex

Recap

summarize it up!

Perfetto for Yocto and Android

Perfetto is ...

- ... a very powerful and good successor of the Catapult Tracer
- ... really well integrated into Android ecosystem (It works just out of the box)
- ... not (yet) well integrated into the Yocto ecosystem (but it’s promising)

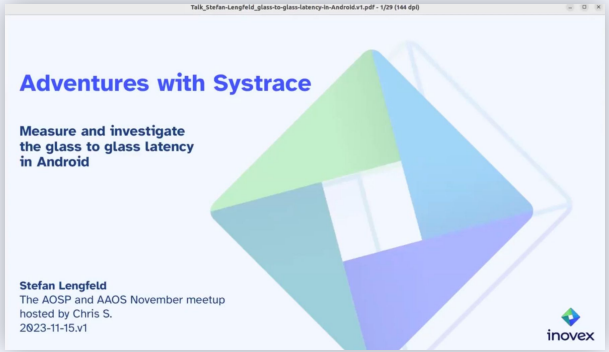
What can perfetto do for you!

Perfetto helps you, because it ...

- is a really powerful ecosystem
  - advancing the use of existing tools,
  - without replacing them!
- is battled tested in the Android and Chromium ecosystem
- is easy to use and powerful Tracing GUI
- has nice graphics
  - making it easy to see relations
  - directly suitable for the management level 😉

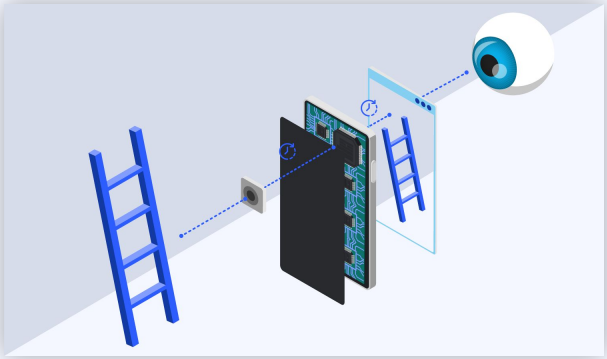
Tracing by example – Glass-to-glass latency in Android

Talk:



www.youtube.com/watch?v=NKP4JcVegbY

Blogpost:



www.inovex.de/de/blog/the-glass-to-glass-latency-on-android/

(still with systrace)

inovex

Thank you!

Time for questions.

inovex is an IT project center driven by innovation and quality, focusing its services on ‘Digital Transformation’.

- founded in 1999
- 500 employees
- 8 offices across Germany

www.inovex.de

Anna-Lena Marx
Embedded Systems Developer

anna-lena.marx@inovex.de

Stefan Lengfeld
Android and Linux Embedded Developer

stefan.lengfeld@inovex.de

inovex