

# Analyzing Perfetto traces at every scale



Oct 12th, 2022 - Tracing Summit

[lalitm@google.com](mailto:lalitm@google.com)

# About me

lalitm@, 5y in Google

**2015 & 2016:** Interned @ Google

**2017:** Joined Google, worked on memory-infra in Chrome on Android

**2018:** Joined Perfetto as it was founded

**2019-today:** Work on trace analysis aspects of Perfetto (trace processor, batch trace processor) and manage tracing rollouts on

Google populations

# Agenda

- Overview of Perfetto and trace processor
- Introduction to Android app startups
- Analysing traces in local debugging, lab tests and field tracing
- Latest innovations

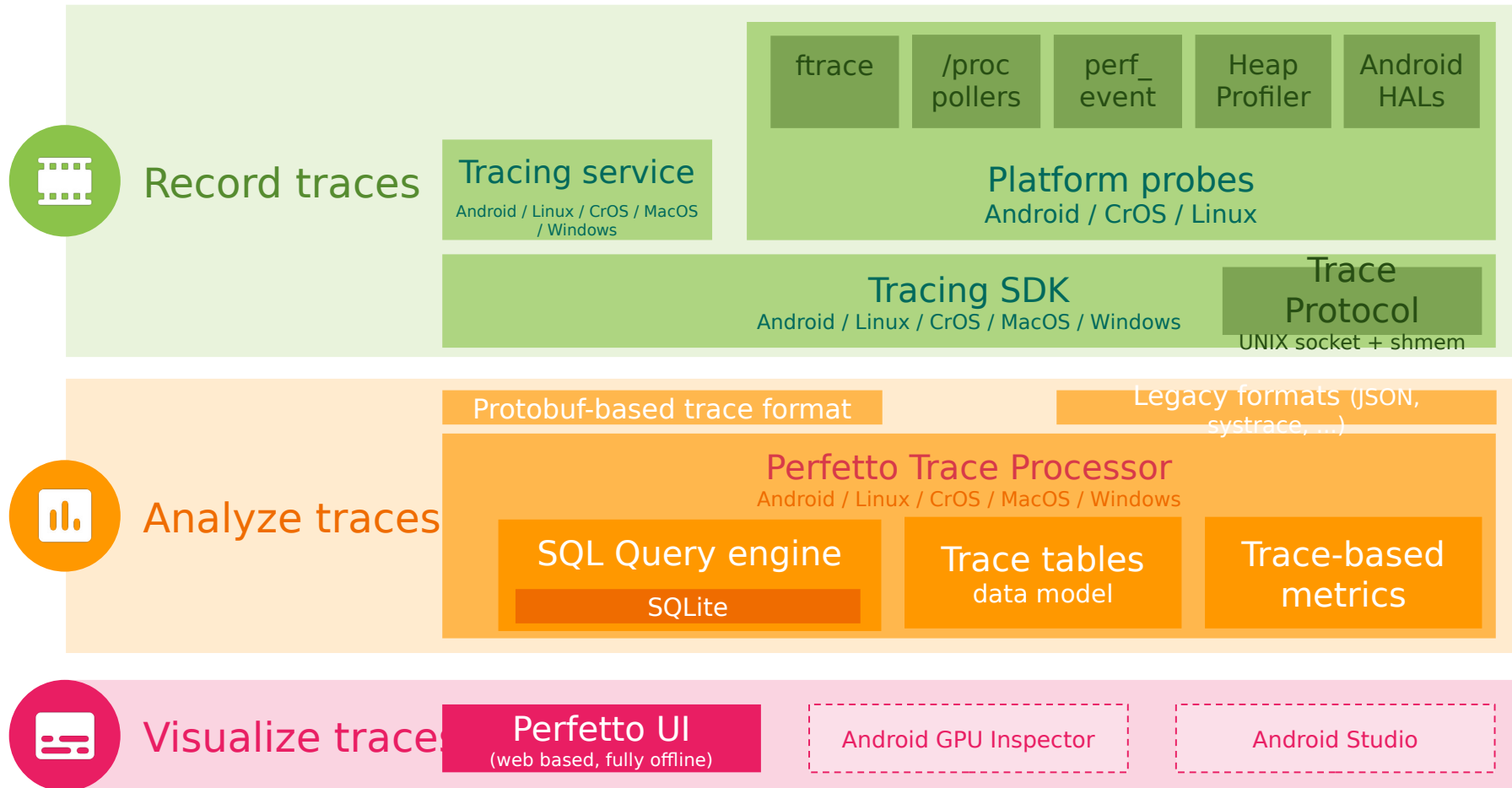
## *The goal*

- Give a sense of the variety of domains we do trace analysis in
  - Similarities and differences in each domain
- Talk is intentionally fast paced
  - Oodles of unconference time allows for discussing any specific topic in more depth
  - Also happy to answer questions/discuss stuff 1:1 – just grab me :)

# Overview of Perfetto & trace processor

Introducing the foundations

# What is Perfetto?



# Primary use-cases for Perfetto



## Local debugging



Device bringup

Local

investigations

Code yellows

## Lab testing



Trace-based

metrics

Integration w/  
dashboards and  
alerting

## Dynamic field tracing



Traces captured  
when  
"problematic"  
events occur

Hard-to-repro  
issues that happen  
only in the field

# Trace Processor

Portable C++ library: SQL engine for trace analysis.

## Efficient

Can ingest multi-GB / hours-long traces

## SQL-powered

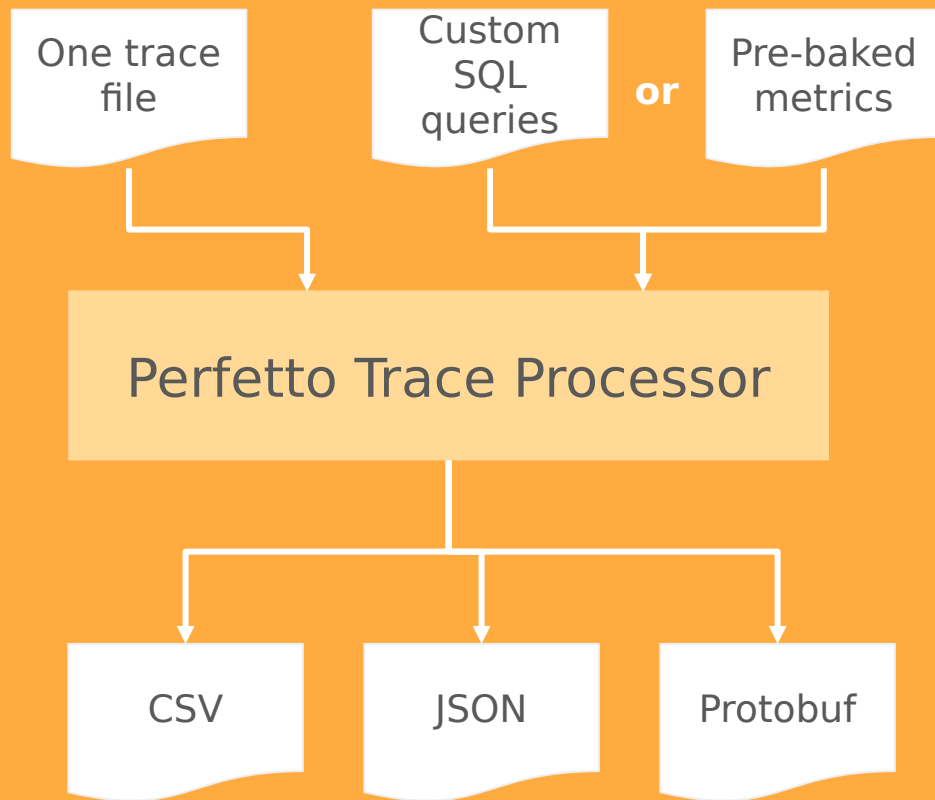
Based on industry standard SQLite engine

## Interoperable

**Runs on servers! Runs on Android! Runs in the browser (via Web Assembly)!**

Runs in other IDEs and tools.

Easy to embed in other apps and integrate with ad-hoc perf test infrastructure



# Feature highlights



**Trace  
parsing and  
“massaging  
”**

Start	<input type="checkbox"/>	<input type="checkbox"/>
Duration	<input type="checkbox"/>	<input type="checkbox"/>
TID	<input type="checkbox"/>	<input type="checkbox"/>
CPU	<input type="checkbox"/>	<input type="checkbox"/>

**Custom, in-  
memory  
columnar  
tables**



**SQL API  
(powered by  
SQLite)**



**Built-in  
metrics for  
critical  
Android and  
Chrome**

USECASES



```
TRACE_B("SocketThread"); ts=100
```

```
...  
TRACE_B("Cleanup"); ts=151
```

```
DoCleanup();
```

```
TRACE_E("Cleanup"); ts=155
```

```
...  
TRACE_E("SocketThread"); ts=160
```



id	ts	slice	
		dur	depth
0	100	60	0
		"SocketThread"	
1	151	4	1
		"Cleanup"	

```
123.0  cpu_frequency: cpu=0  
freq=1000
```

```
145.0  cpu_frequency: cpu=1  
freq=4000
```

```
245.0  rss_stat:          pid=40  
value=512
```

```
345.0  gpu_frequency: cpu=0  
freq=500
```

```
123 sched_switch: p_pid=1 n_pid=2  
end_s=S
```

```
233 sched_waking: pid=1
```

```
354 sched_switch: p_pid=2 n_pid=1  
end_s=D
```



ts	track_name	counter	
		value	track_id
123	CPU Frequency	1000	0
145	CPU Frequency	4000	1
245	RSS Usage	2	512
345	GPU Frequency	500	0
123	110	2	Running
233	121	1	Runnable
354	NULL	1	Running

SPAN\_JOIN  
(span  
intersection)

Table A

Span A

Span B

Table B

Span 1

Span 2

Output

A + 1

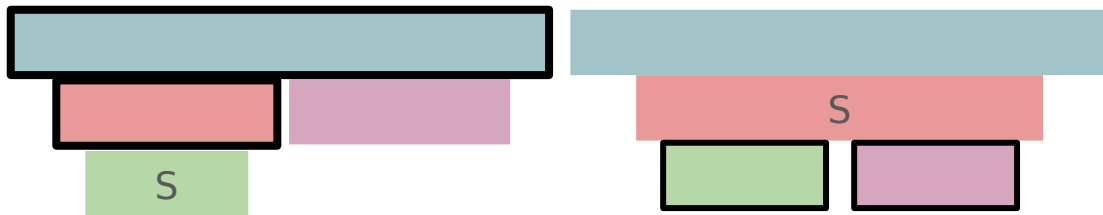
A + 2

B + 2

CREATE\_FUNCTION  
CREATE\_VIEW\_FUNC  
TION

Define functions in  
SQL! (demoed later)

ANCESTOR\_SLICE  
DESCENDENT\_SLICE

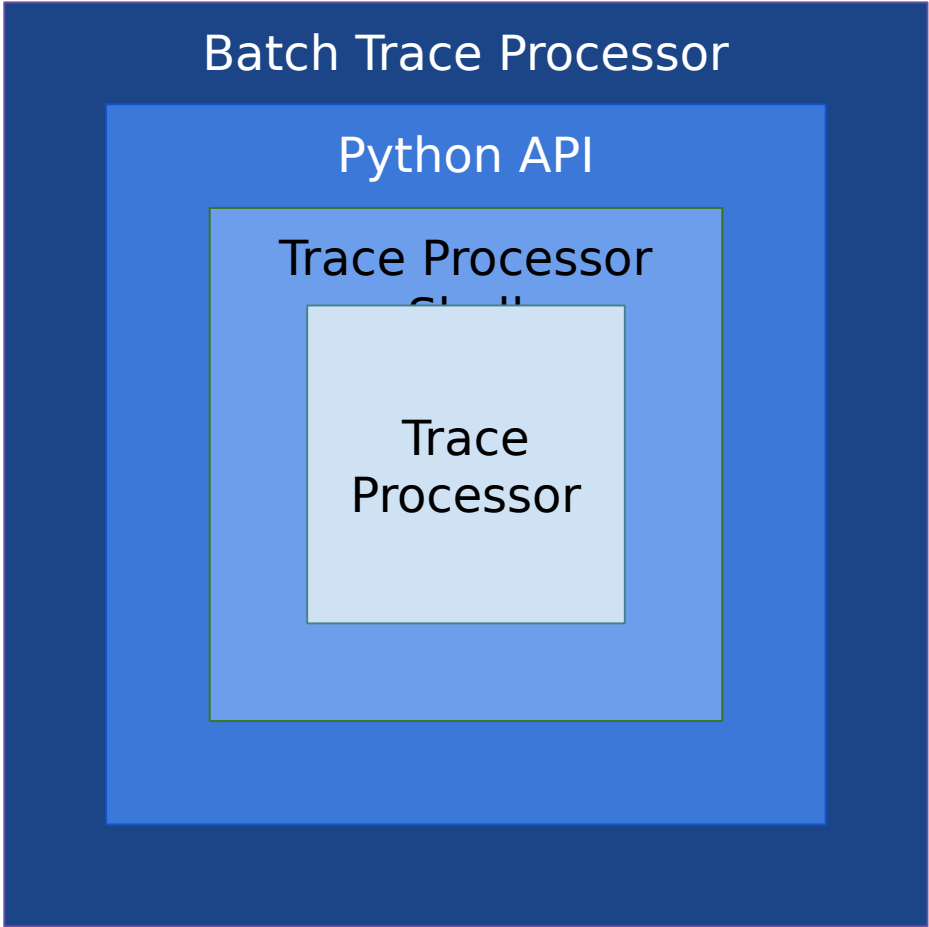


Batch Trace Processor

Python API

Trace Processor

Trace  
Processor



# Why not ...?

## Why not work with the trace directly?

```
interned_data {
  id: 1
  string:
  "loooong"
}
ftrace_event {
  string_iid: 1
}
my_custom_event {
  string_iid: 1
}
track_event {
  string_iid: 1
}
...
```

## Why not expose trace points directly instead of slices/counters?

name	ts	cpu	pid
sched_switch	0	2	100
sched_switch	0	3	150
sched_waking	1	2	300

vs

state	ts	pid
	cpu	dur
Running	2	100
	50	0
Running	3	150
	...	...

## Why not C/C++/Python/<your favourite language> instead of SQL?

```
SELECT *
FROM slice
WHERE
  name LIKE 'startup%'
AND
  dur > 100
```

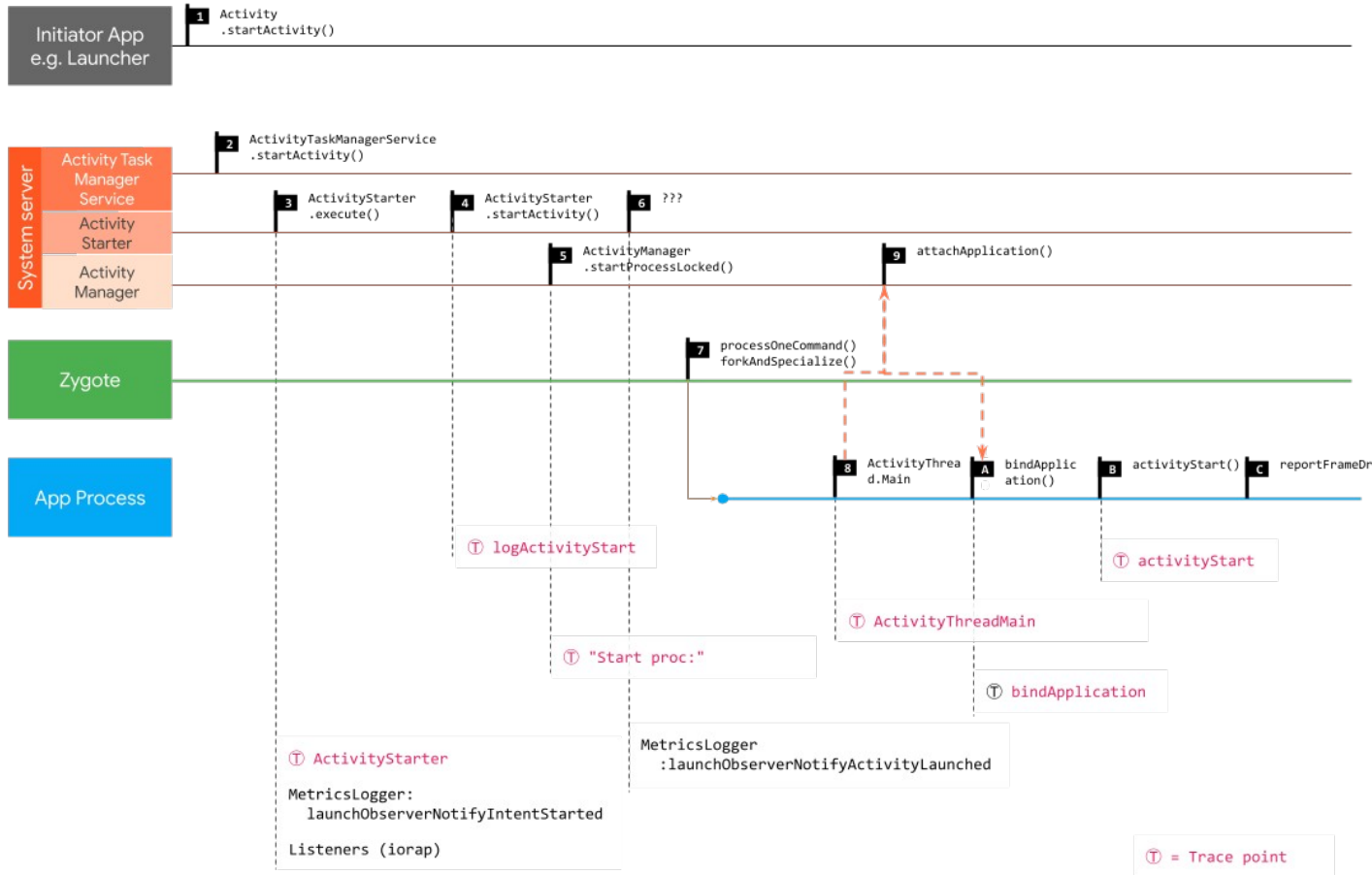
In your chosen language:

- How much code would this take?
- How natural would it feel?
- How big would the library API surface be?

# App Startups

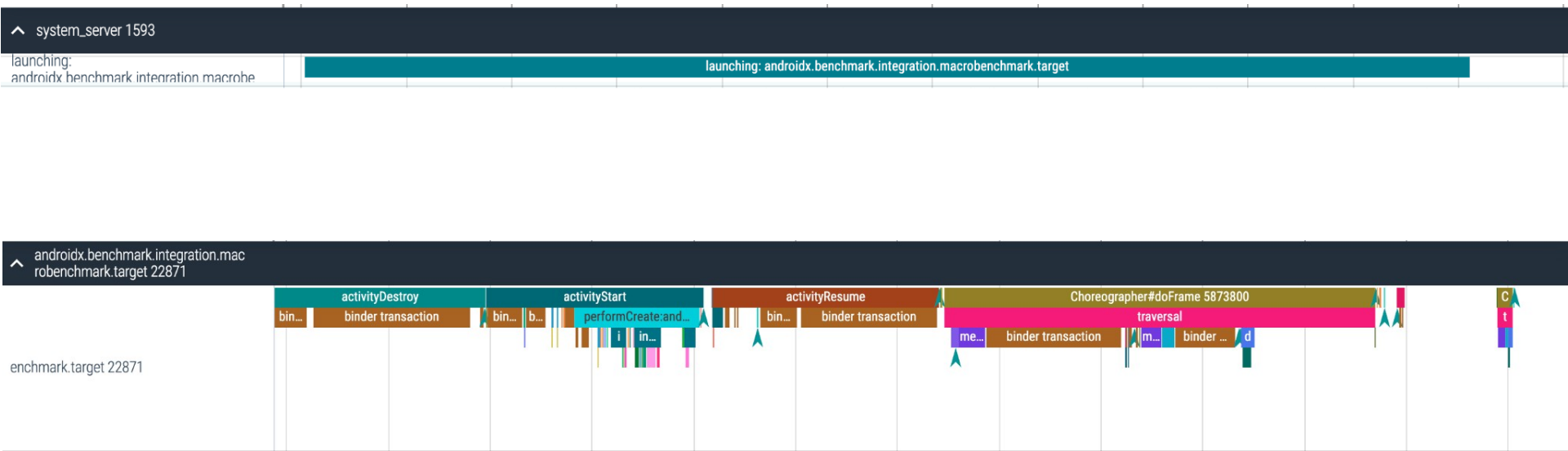
An important use-case for tracing on Android

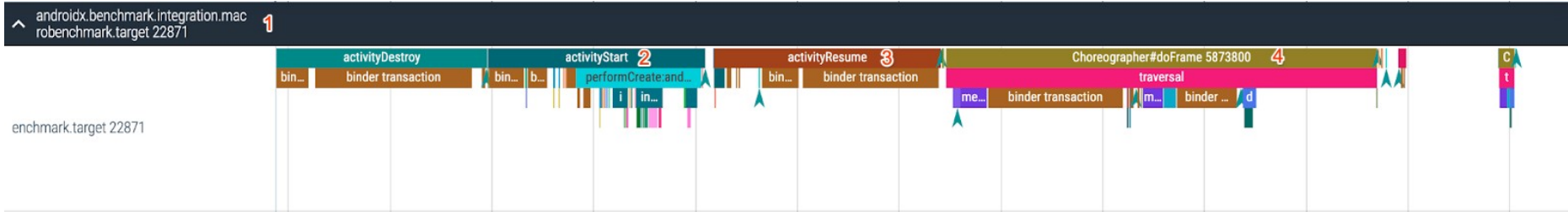
# Journey of an Android cold start



From a 2019 study [go/startup-metrics](#). Most of this changed substantially in the last 2 years.

# Startups in Perfetto traces





```
message StartupProto {  
    optional string process_name = 1;  
    optional int64 time_activity_start = 2;  
    optional int64 time_activity_resume = 3;  
    optional int64 time_choreographer = 4;  
}
```



```
CREATE VIEW launching_slices AS
SELECT ts, ts_end, STR_SPLIT(name, ':', 1) AS launched_package
FROM slice
WHERE name GLOB 'launching: *';
```

```
CREATE VIEW launching_processes AS
SELECT process.name
FROM process
JOIN launching_slices
ON process.name = launching_slices.launched_package;
```

...

```
CREATE VIEW launching_slices AS ...;  
CREATE VIEW launching_processes AS ...;
```

```
SELECT CREATE_FUNCTION(  
    'DUR_FOR_SLICE(process_name STRING, slice_glob STRING)',  
    'INT64',  
    '  
    SELECT dur  
    FROM thread_slice  
    WHERE process_name = $process_name AND name GLOB $slice_glob  
    ,  
);  
...
```

```
CREATE VIEW launching_slices AS ...;
CREATE VIEW launching_processes AS ...;
SELECT CREATE_FUNCTION('STARTUP_SLICE_DUR(...)', ...);
```

```
CREATE VIEW startup_metric_output AS
SELECT StartupProto(
    'process_name',      launching_processes.name,
    'time_activity_start', DUR_FOR_SLICE(name, 'activityStart*'),
    'time_activity_resume', DUR_FOR_SLICE(name, 'activityResume*'),
    'time_choreographer', DUR_FOR_SLICE(name, 'Choreographer#do*')
)
FROM launching_processes;
```

```
startup {  
  process_name: "androidx.benchmark.integration.macrobenchmark.target"  
  time_activity_start: 10.707813  
  time_activity_resume: 11.126928  
  time_choreographer: 21.174429  
}
```

# Local debugging

The entry path to trace analysis

- Navigation
  - Open trace file
  - Open with legacy UI
  - Record new trace
- Current Trace
  - api31\_startup\_warm7481842159456  
015732\_perfetto-trace (4 MB)
  - Show timeline
  - Share
  - Download
  - Query (SQL)
  - Metrics
  - Info and stats
- Convert trace
  - Switch to legacy UI
  - Convert to .json
  - Convert to .systrace
- Example Traces
  - Open Android example
  - Open Chrome example
- Support



```
SELECT *
FROM slice
WHERE name GLOB 'launching: *'
```

Query result - 4 ms SELECT \* FROM slice WHERE name GLOB 'launching: \*'

Copy query Copy result (.tsv) Close

id	type	ts	dur	track_id	category	name	depth	stack_id	parent_stack_id	parent_id	arg_set_id	thread_ts
4805	internal_slice	186982060149946	55378859	1909	NULL	launching: androidx.benchmark.integration.macrobenchmark.target	0	7147770010758995	0	NULL	10875	NULL

Navigation

- Open trace file
- Open with legacy UI
- Record new trace

Current Trace

- api31\_startup\_warm7481842159456  
015732.perfetto-trace (4 MB)
- Show timeline
- Share
- Download
- Query (SQL) ←
- Metrics
- Info and stats

Convert trace

- Switch to legacy UI
- Convert to .json
- Convert to .systrace

Example Traces

- Open Android example
- Open Chrome example

Support

Navigation

- Open trace file
- Open with legacy UI
- Record new trace

Current Trace

- api31\_startup\_warm7481842159456  
015732.perfetto-trace (4 MB)
- Show timeline
- Share
- Download
- Query (SQL)
- Metrics
- Info and stats

Convert trace

- Switch to legacy UI
- Convert to .json
- Convert to .systrace

Example Traces

- Open Android example
- Open Chrome example

Support

Select a metric:

```
android_startup {
  startup {
    startup_id: 1
    startup_type: "warm"
    package_name: "androidx.benchmark.integration.macrobenchmark.target"
    process_name: "androidx.benchmark.integration.macrobenchmark.target"
    process {
      name: "androidx.benchmark.integration.macrobenchmark.target"
      uid: 10246
    }
    activities {
      name: "androidx.benchmark.integration.macrobenchmark.target.TrivialStartupFullyDrawnActivity"
      method: "performCreate"
      ts_method_start: 186982074137030
    }
    activities {
      name: "androidx.benchmark.integration.macrobenchmark.target.TrivialStartupFullyDrawnActivity"
      method: "performResume"
      ts_method_start: 186982080918073
    }
  }
  zygote_new_process: false
  activity_hosting_process_count: 1
  event_timestamps {
    intent_received: 186982050780778
    first_frame: 186982115528805
  }
  to_first_frame {
    dur_ns: 64748027
    dur_ms: 64.748027
    main_thread_by_task_state {
      running_dur_ns: 18635782
      runnable_dur_ns: 3375625
      uninterruptible_sleep_dur_ns: 0
    }
  }
}
```



~

> ~/trace\_processor startup-demo.perfetto-trace 2>/dev/null

> select ts, dur, name from slice where name glob 'launching: \*'

ts	dur	name
186982060149946	55378859	launching: androidx.

Query executed in 2.006 ms

> █

~

```
> cat ~/launching_query.sql  
select ts, dur, name  
from slice  
where name glob 'launching: *'
```

~

```
> ~/trace_processor -q ~/launching_query.sql startup-demo.perfetto-trace 2>/dev/null  
"ts","dur","name"  
186982060149946,55378859,"launching: androidx.benchmark.integration.macrobenchmark.target"
```

~

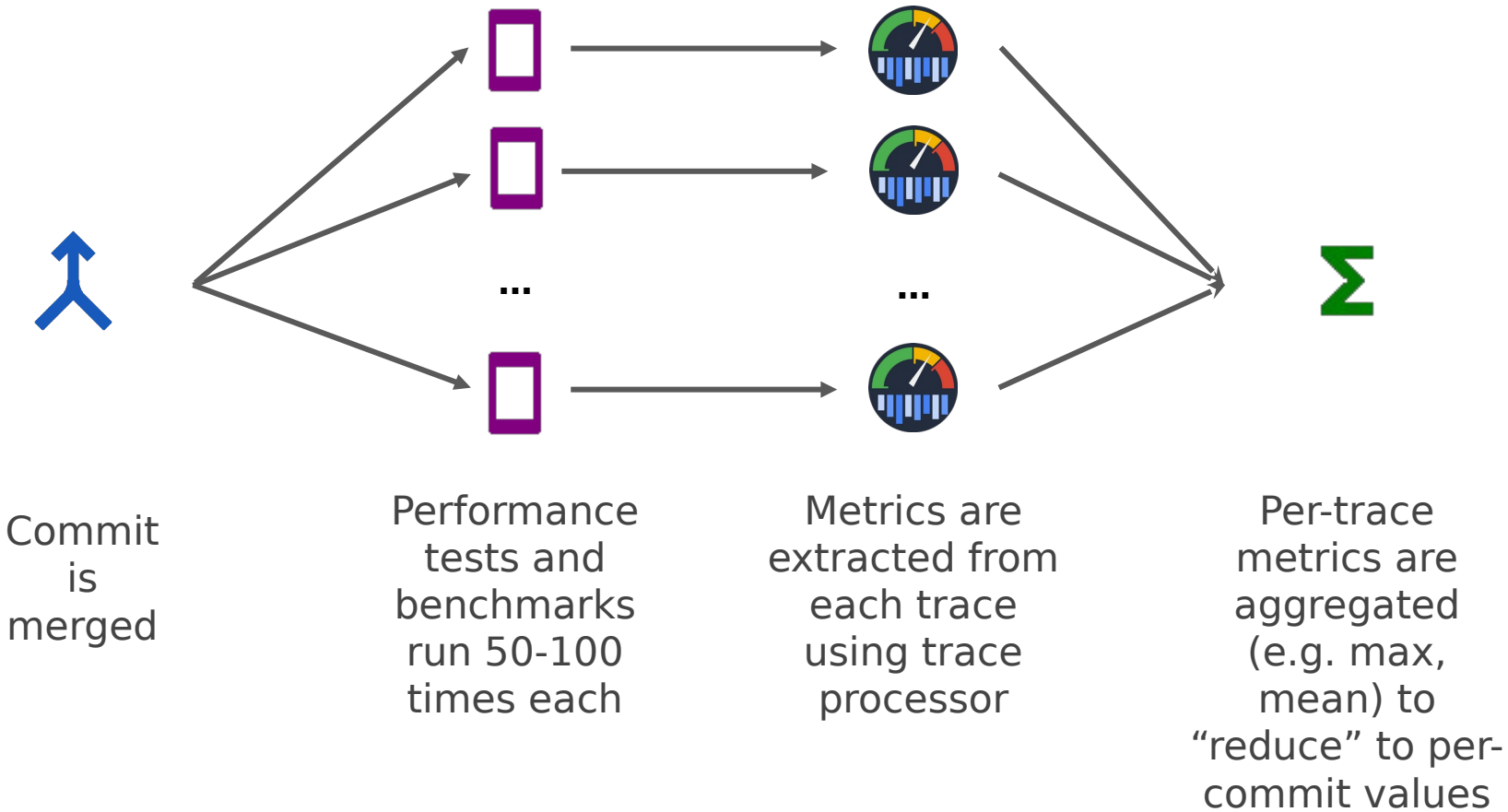
```
> █
```

```
> ~/trace_processor --run-metrics android_startup startup-demo.perfetto-trace 2>/dev/null
```

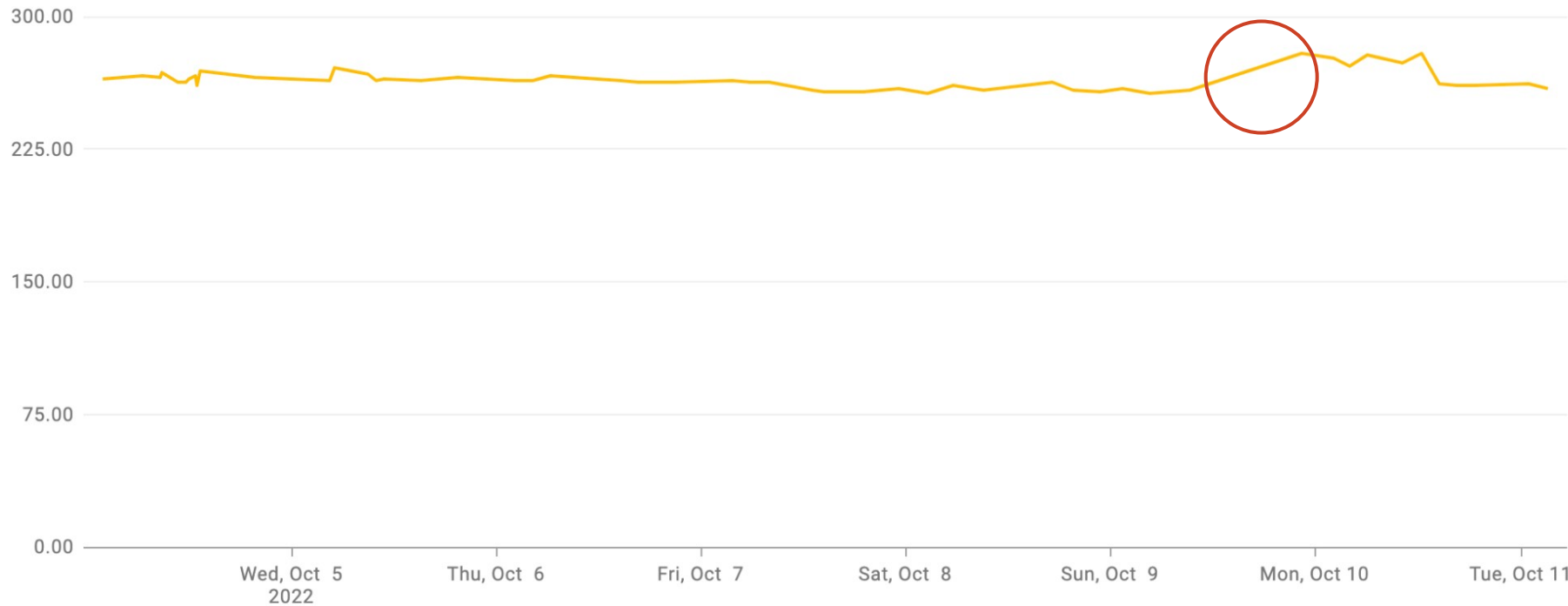
```
android_startup {  
  startup {  
    startup_id: 1  
    startup_type: "warm"  
    package_name: "androidx.benchmark.integration.macrobenchmark.target"  
    process_name: "androidx.benchmark.integration.macrobenchmark.target"  
    process {  
      name: "androidx.benchmark.integration.macrobenchmark.target"  
      uid: 10246  
    }  
    activities {  
      name: "androidx.benchmark.integration.macrobenchmark.target.TrivialStartupFullyDrawnActivity"  
      method: "performCreate"  
      ts_method_start: 186982074137030  
    }  
    activities {  
      name: "androidx.benchmark.integration.macrobenchmark.target.TrivialStartupFullyDrawnActivity"  
      method: "performResume"  
      ts_method_start: 186982080918073  
    }  
  }  
  zygote_new_process: false  
  activity_hosting_process_count: 1  
  event_timestamps {  
    intent_received: 186982050780778  
    first_frame: 186982115528805  
  }  
}
```

# Lab testing

Ensuring performance doesn't regress over time



# Calculator



- git\_master-oriole-userdebug-cold\_startup-com.google.android.calculator-mean
- git\_master-oriole-userdebug-perfetto\_android\_startup-com.google.android.calculator-hsc-full\_startup-dur\_ms-mean
- git\_master-oriole-userdebug-perfetto\_android\_startup-com.google.android.calculator-to\_first\_frame-dur\_ms-mean
- git\_tm-qpr-dev-oriole-us...

# Field tracing

Solving the performance issues faced by real users

# Startup approach in one slide

Break down  
the problem  
space

*In Critical User Journeys*



Cold App Launch



Switch app

...



Return to launcher

Metrics

Measure  
startup

*Measure what users see*

Action

Trigger trace  
collection

*When startup happens*



Analyze  
traces *Observe root  
causes,  
lot of manual work!*



Scale up!  
*Write trace classifiers,  
run them against the  
corpus*



Stack rank  
*GROUP BY root cause,  
ORDER BY COUNT()*



Fix top issues  
*Profit!*



```
SELECT 'dex2oat running during launch' AS slow_cause
WHERE IS_PROCESS_RUNNING_CONCURRENT_TO_LAUNCH(launches.id, '*dex2oat64')

UNION ALL
SELECT 'installd running during launch' AS slow_cause
WHERE IS_PROCESS_RUNNING_CONCURRENT_TO_LAUNCH(launches.id, '*installd')

UNION ALL
SELECT 'Main Thread - Time spent in Running state'
AS slow_cause
WHERE MAIN_THREAD_TIME_FOR_LAUNCH_AND_STATE(launches.id, 'Running') > 2e9

UNION ALL
SELECT 'Main Thread - Time spent in Runnable state'
AS slow_cause
WHERE MAIN_THREAD_TIME_FOR_LAUNCH_AND_STATE(launches.id, 'R') > 1e8

UNION ALL
SELECT 'Main Thread - Time spent in interruptible sleep state'
AS slow_cause
WHERE MAIN_THREAD_TIME_FOR_LAUNCH_AND_STATE(launches.id, 'S') > 2e9

UNION ALL
SELECT 'Main Thread - Time spent in Blocking I/O'
WHERE MAIN_THREAD_TIME_FOR_LAUNCH_STATE_AND_IO_WAIT(launches.id, 'D*', true) > 2e9

















UNION ALL
SELECT 'Time spent in OpenDexFilesFromOat*'
AS slow_cause
WHERE DUR_SUM_FOR_LAUNCH_AND_SLICE(launches.id, 'OpenDexFilesFromOat*') > 1e6
```




Collected slow-startup field traces (last 60 days)



Date	Device Name	App	Apk Version Code	Build Id	Open	Download	Bug	Start Type	Ttff (ms)	Slow Start Reason	Activity onRestart	Running (ms)	Runnable (ms)	Uninterruptible Sleep (ms)	Interruptible Sleep (ms)
2022-10-10	bluejay	com.android.chrome	524907933	TQ1A.220930.003	<a href="#">Link</a>	<a href="#">Download</a>	<a href="#">Create bug</a>	warm	25734.05	Main Thread - Time spent in Runnable state Main Thread - Time spent in interruptible sleep state Time spent in OpenDexFilesFromOat*	false	455.28	138.15	81.39	25045.84
2022-10-10	bramble	com.google.android.apps.messaging	999999999	TQ1A.220930.003	<a href="#">Link</a>	<a href="#">Download</a>	<a href="#">Create bug</a>	cold	6861.00	dex2oat running during launch installd running during launch Main Thread - Time spent in Running state Main Thread - Time spent in Runnable state Time spent in OpenDexFilesFromOat* Time spent in bindApplication Time spent in view inflation JIT Activity Main Thread - Lock contention JIT compiled methods Broadcast dispatched count Broadcast received count	false	3376.51	1437.58	381.35	1363.03
2022-10-10	panther	com.android.settings	33	TQ1A.220930.003	<a href="#">Link</a>	<a href="#">Download</a>	<a href="#">Create bug</a>	cold	6216.06	installd running during launch Main Thread - Time spent in interruptible sleep state Time spent in OpenDexFilesFromOat* Time spent in bindApplication Time spent in ResourceManager#getResources Main Thread - Lock contention Main Thread - Monitor contention Broadcast dispatched count Broadcast received count	false	280.98	43.51	179.69	5331.02
2022-10-10	barbet	com.google.android.apps.messaging	999999999	TP1A.221005.002	<a href="#">Link</a>	<a href="#">Download</a>	<a href="#">Create bug</a>	warm	6180.49	Main Thread - Time spent in Running state Time spent in OpenDexFilesFromOat* Time spent in bindApplication	false	5729.75	53.64	33.50	352.04

- ▶  Summary
- ▶  Stability
- ▶  Battery
- ▶  Startup
- ▶  Memory
- ▶  CPU
- ▶  Stack sampling
- ▶  Binder/looper
- ▶  Jank
  -  Jank Traces
  -  Latency Traces
- ▶  Lab
- ▶  Metric Index
- ▶  Timeline Explorer
- ▶  What's New
- ▶  My alerts

Dogfood: TM QPR1 release

 Values with targets

Experiments 

Devices: All Pixel

Last 7 days (Oct 2 - Oct 8)







## Jank Field Traces

Filter by + Add filter

### Jank Trace Details

Show columns...

Items per page: 20 1 - 20 of 10000 < >

Interaction	Jank cause	Missed app frames	Missed SF frames <span style="font-size: small;">↑</span>	Missed frames %	Bugs
Notification shade row swipe com.android.systemui Pixel 5a 5G - TQ1A.220923.001 Oct 2, 2022  4b889b46-271b-2aa3-62b4-67d7ad9cde42  <a href="#">Open</a>  <a href="#">Download</a>  <a href="#">Open in Colab</a>	MainThread - binder transaction time Long running time ShadeListBuilder Long running time - animation	1	0	16.7%	...
Notification shade scroll fling com.android.systemui Pixel 5a 5G - TQ1A.220923.001 Oct 2, 2022  383c3df2-e41f-8cb5-62b4-67d7ad9cde42  <a href="#">Open</a>  <a href="#">Download</a>  <a href="#">Open in Colab</a>	RenderThread - long shader_compile Long running time Skia - OpsTask::onExecute RenderThread - Skia Drawing call Long running time - DrawFrames	1	0	7.69%	...
	MainThread - binder transaction time GPU completion - long completion time Long running time				

Dogfood: SC release Values with targets Experiments Devices: Standard May 1, 2021 - Jul 24, 2021

### Interactions with jank

Lower is better

Create alert Related distribution → Related traces →

Filter by Device codename: redfin Interaction type: App Start - from icon + Add filter

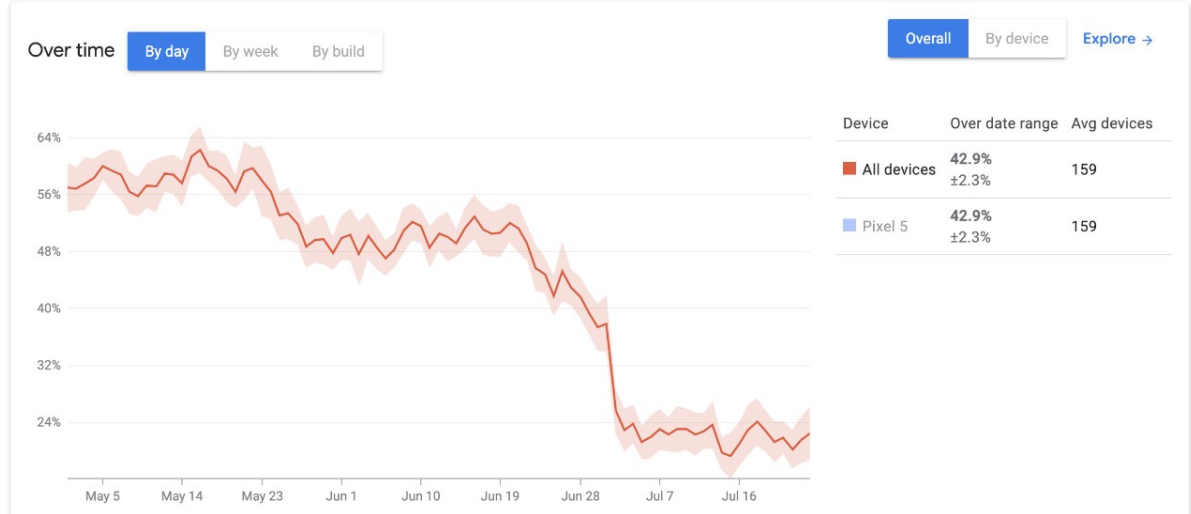
Interactions with jank

**42.9%**  
±2.3%

**Device status**  
1 No target

**Devices**  
159 average

**Metric definition**  
The percentage of interactions that were janky.  
[Read full definition →](#)



Device count

App start from icon

58% -> 21 %

# Latest innovations

At the cutting edge of trace analysis

# Can we interactively query $>1$ trace?



## Perfetto UI / Trace Processor

Pros:

- Power/complexity curve
- UI is visual

Cons:

- Single trace

???

Pros:

- Fast iteration on 100-1000+ traces
- Full power of trace processor

Cons:

- ???

## Map-Reduce Pipeline

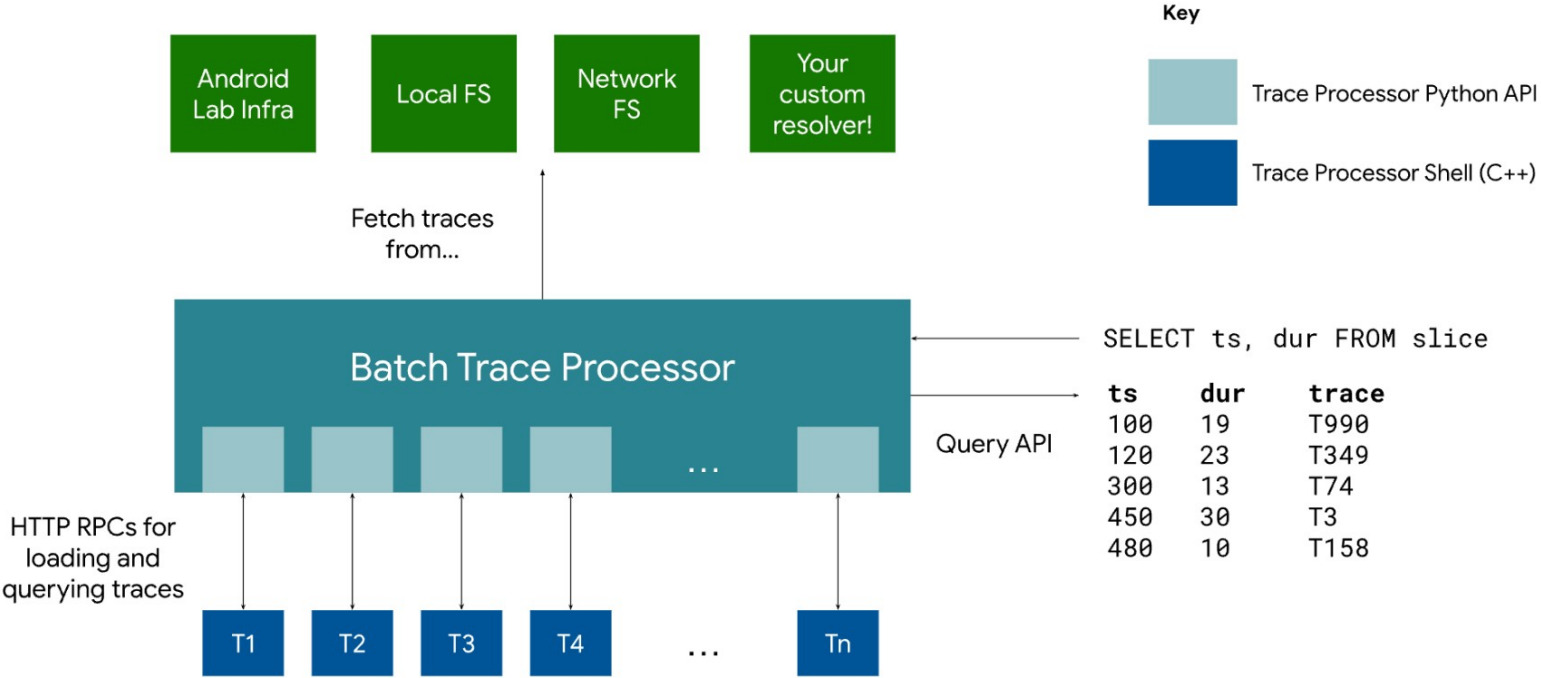
Pros:

- Run on thousands of traces
- Full analysis power

Cons:

- Very complex
- High iteration

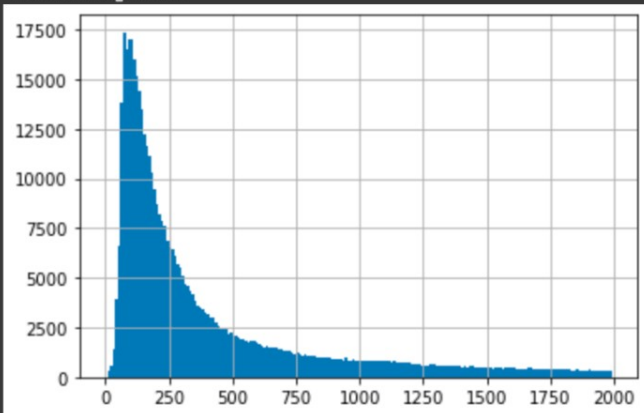
# Batch Trace Processor (BTP)





```
[ ] 1 # Like before, we can plot the duration of binder transactions
2 # but this time across all 100 traces.
3 df = btp.query_and_flatten("select dur/1e3 as dur from slice where name glob 'binder transaction']").astype({'dur': 'float64'})
4 df['dur'].hist(bins=range(0, 2000, 10))
5
6 # What a smooth graph!. By aggregating across all 100 traces, we get a much more representative
7 # sample than if we were to look at a single trace.
```

<AxesSubplot:>



Jank Colab Example  
(Googlers only -  
sorry!)

# Startup Colab Example

(Googlers only -  
sorry!)

Thanks! Questions?

For docs, mailing list and Discord channel see

[perfetto.dev](https://perfetto.dev)