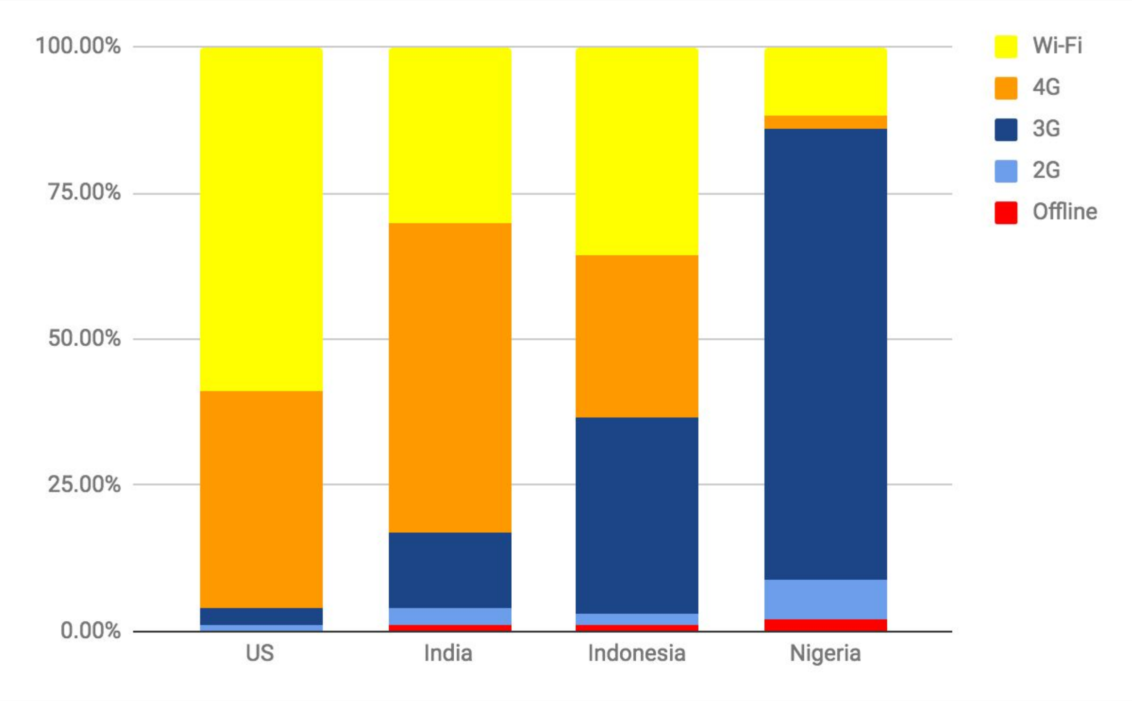
# Speed Focus for 2018

## Load faster, be more responsive, crash less.

tdresser@, sullivan@, bengr@

# Mobile networks are slow



Fraction of mobile browsing sessions on each network technology

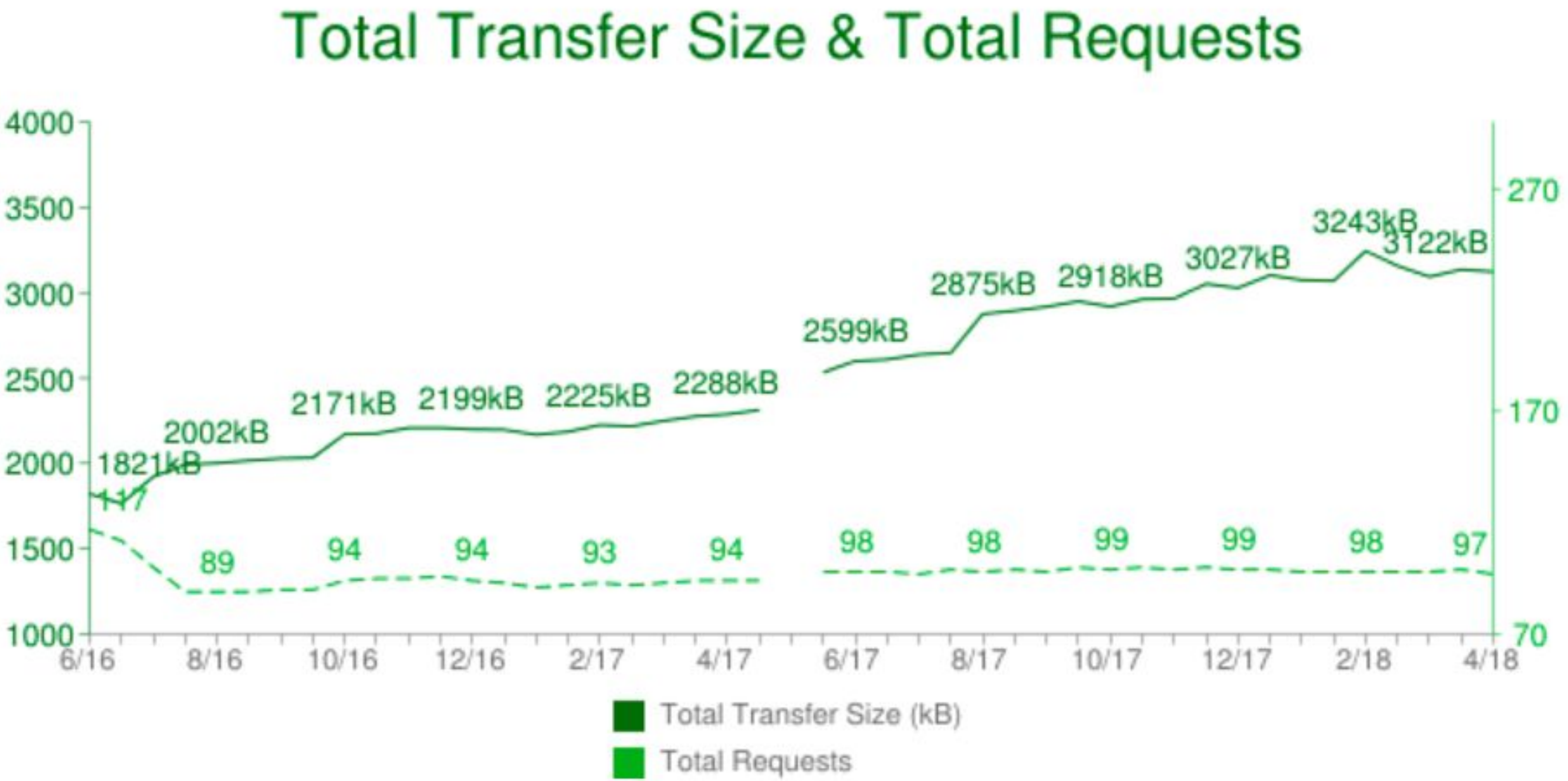
Chrome UMA (Android, Q4 2017)

# Mobile devices are constrained

**24%** of Android devices shipped globally in 2017 H1 are **<$100**, often having **limited memory**.

IDC Quarterly Mobile Phone Tracker

# Pages are bigger and more complicated



httparchive.org

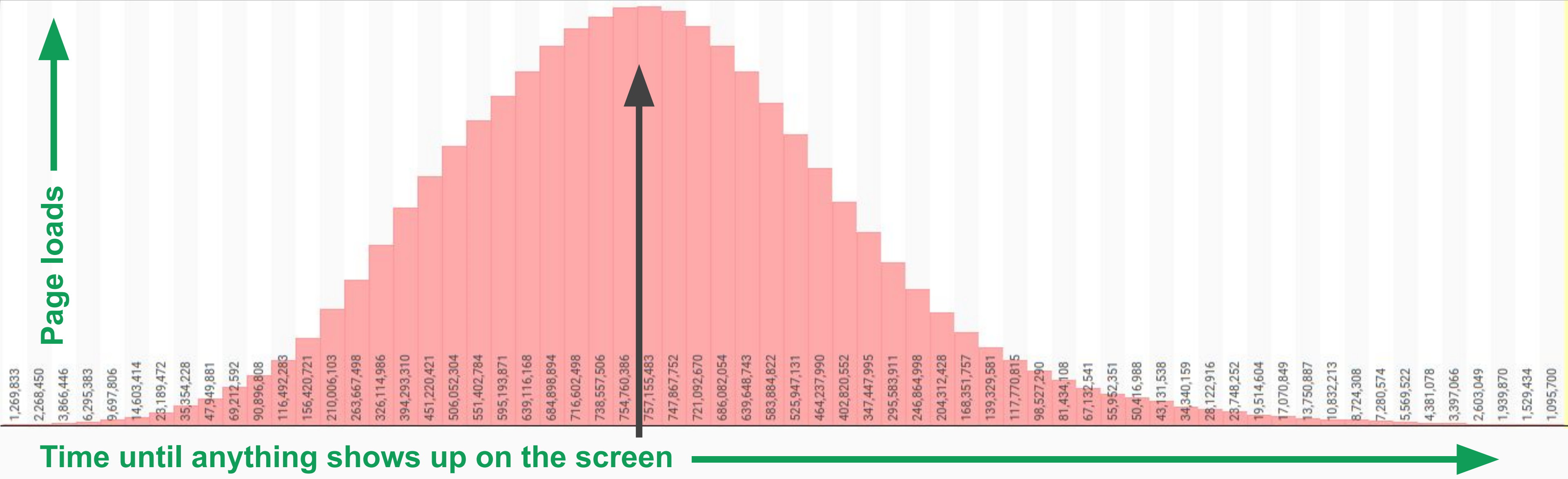
# Pages load slowly, are sluggish, and they crash.

How bad is it?

Example: Load latency in Nigeria (3G)

How much would you use the web if every page took this long to load?

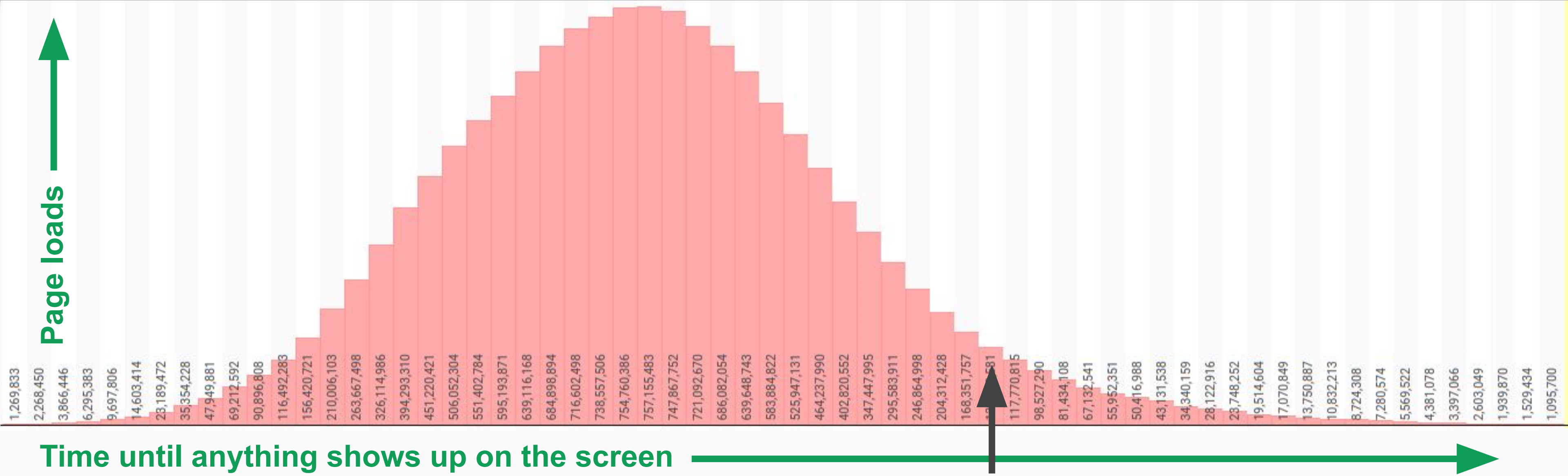
# But I thought the web was fast?



**1.0s median**

Chrome UMA (Android, Stable Channel, 4/18)

# Tail latency is slow

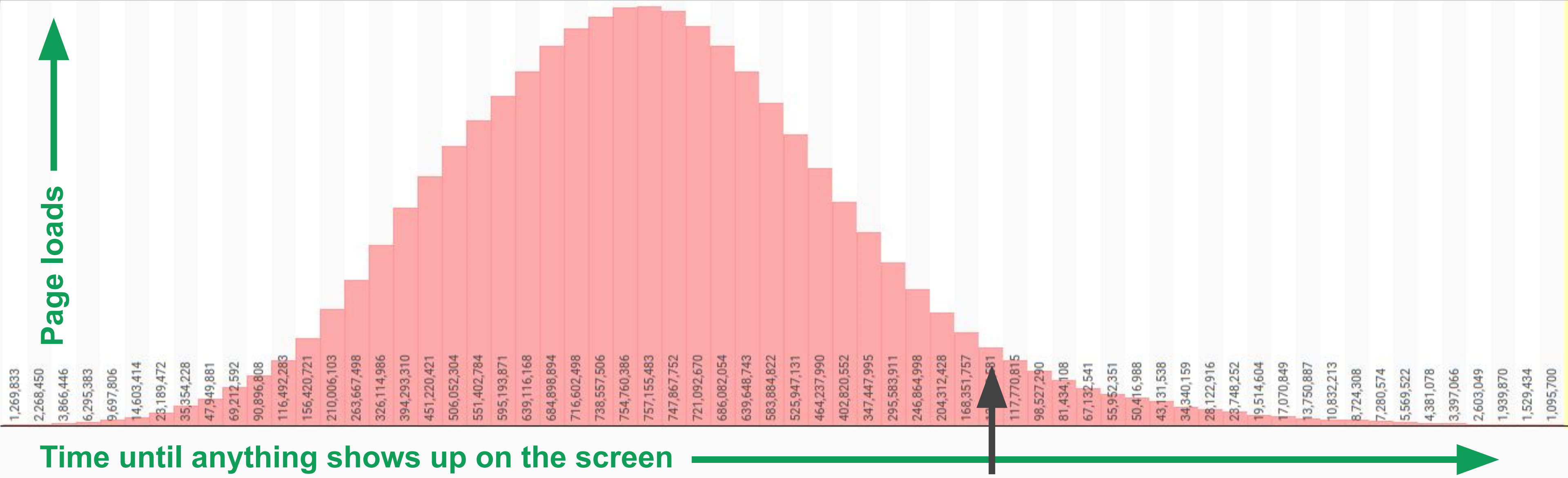


**Page loads**

**Time until anything shows up on the screen**

**6.0s 95th %-ile**

# It’s even slower in emerging markets



12s ~~6.0s~~ 95th %-ile in Nigeria

# What is “Speed”?

“Speed” is all user facing aspects of performance.

How do we ensure we’re tackling the speed issues which impact users most?

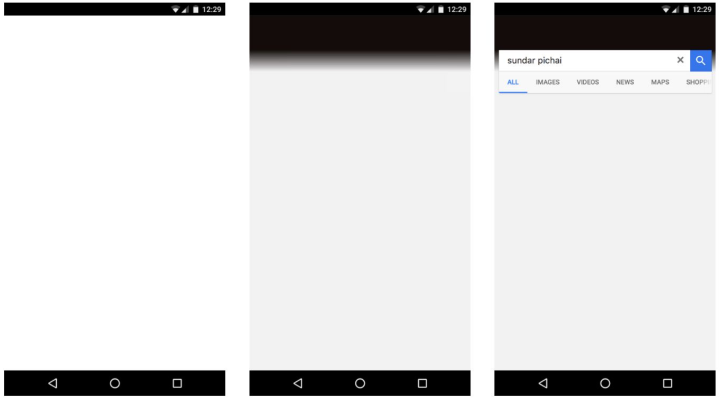
# 2018 Speed Goals

- Load Fast
- Be Responsive
- Don’t run Out Of Memory

See details on metrics in the Speed Launch Metrics Survey.

# Load Fast - **First Contentful Paint**

When the browser first rendered any text, image (including background images), non-white canvas or SVG. This includes text with pending webfonts.

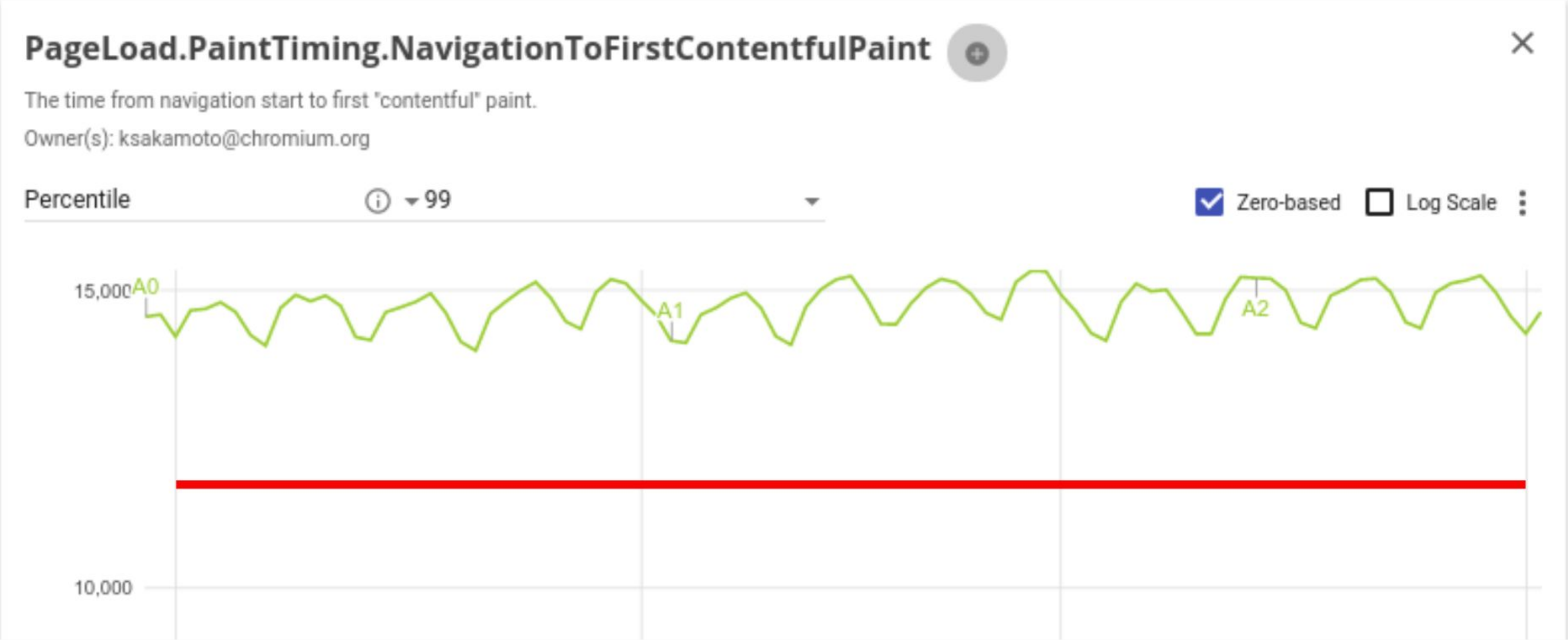


First Paint (FP)

First Contentful Paint (FCP)

# Load Fast - First Contentful Paint

## Overall Android - 99’th percentile First Contentful Paint



### PageLoad.PaintTiming.NavigationToFirstContentfulPaint

The time from navigation start to first "contentful" paint.

Owner(s): ksakamoto@chromium.org

Percentile ⓘ ▾99

☑ Zero-based □ Log Scale

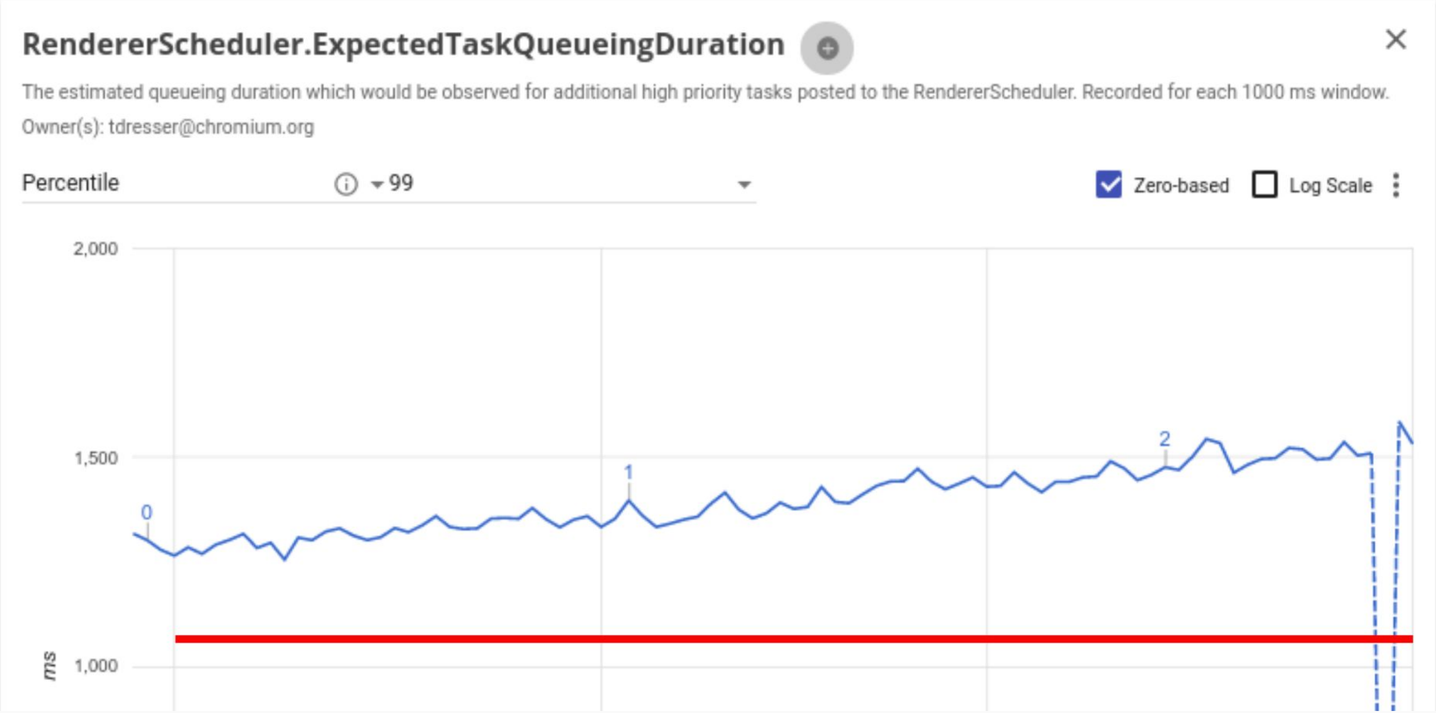
# Be Responsive - **E**xpected **Q**ueueing **T**ime

When a user interacts, we want to respond instantly.

EQT: The expected duration an input event would be queued on the main thread, for an input arriving at a random time within a window.

# Be Responsive - **Expected Queueing Time**

<=512MB Android - 99’th percentile Expected Queueing Time



RendererScheduler.ExpectedTaskQueueingDuration

The estimated queueing duration which would be observed for additional high priority tasks posted to the RendererScheduler. Recorded for each 1000 ms window.

Owner(s): tdresser@chromium.org

Percentile: 99

☑ Zero-based □ Log Scale

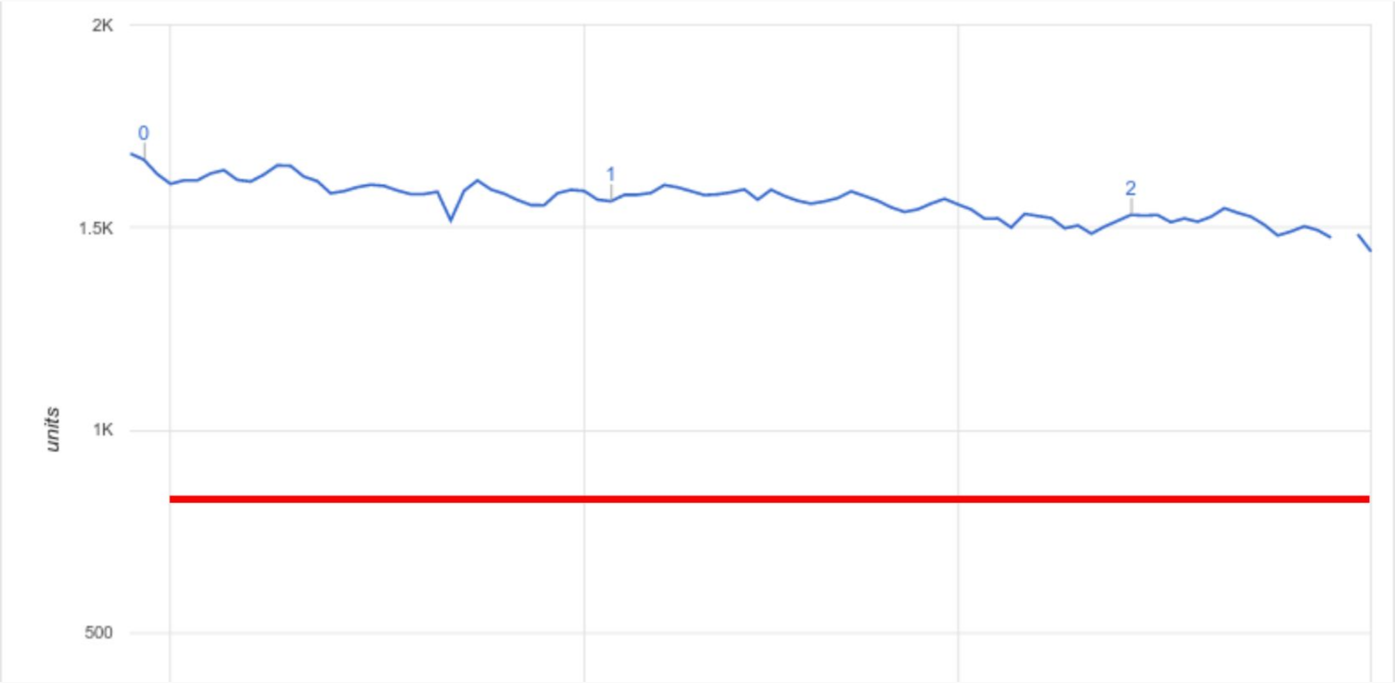
# Don’t Run Out of Memory - **OOMs**

Renderer crashes from running out of memory on Android.

We track OOMs / million page loads.

# Don’t Run Out of Memory - OOMs

<=512MB Android - OOMs / millino page loads



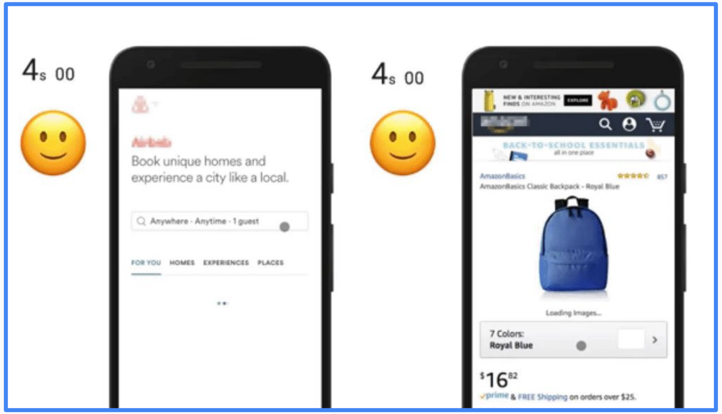
# Metrics - Scroll Latency

- We track Scroll Begin Latency and Scroll Update Latency separately.
- These are measured for touch & wheel input.
- In all cases, we measure the duration from the event's hardware timestamp[1] until we update the display with the scrolled offset.

[1] Except on Windows, where drivers can lie to us, and we just use the time when the browser first saw the event.

# Metrics - Interactivity at Page Load

- Time to Interactive: When the page is ready to process input within 50ms.
    - Can’t be measured accurately in the wild.
- First Input Delay: The queueing delay of the first user input.
    - Correlates fairly well with TTI.


4s 00
Book unique homes and experience a city like a local.
Anywhere · Anytime · 1 guest
FOR YOU
HOMES
EXPERIENCES
PLACES
4s 00
BACK-TO-SCHOOL ESSENTIALS
AmazonBasics
AmazonBasics Classic Backpack - Royal Blue
Loading Images...
7 Colors:
Royal Blue
$16 82
prime & FREE Shipping on orders over $25.

# Types of Speed Projects

There are three main types of speed projects within Blink.

- Optimizations: Making existing pages faster.
- New Primitives: Shipping new features that enable faster pages.
- Activation: Making features fast enough that pages can actually use them.

# Criteria for investing in speed projects

- Optimizations
  - Will this contribute to our overall speed goals?
  - Does an A/B test (via  Finch) show that we're making a real improvement?
- New Primitives
  - Do we have a launch partner for an origin trial?
  - Did they see concrete improvements in metrics we care about during the origin trial?
- Activation
  - Do we have a launch partner?
  - Will they be able to adopt the feature, given the proposed performance improvements?

# Evaluating impact of a performance optimization project

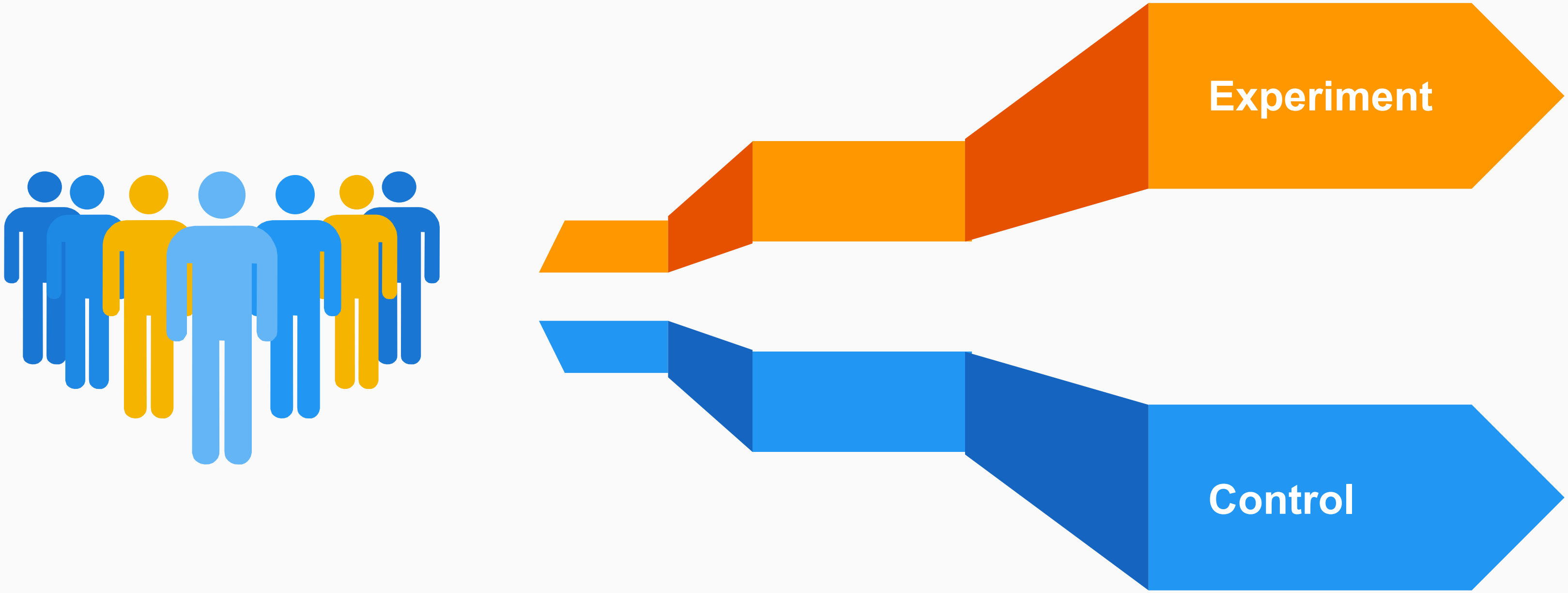
We evaluate the

**end user impact**

on the

**speed launch metrics**.

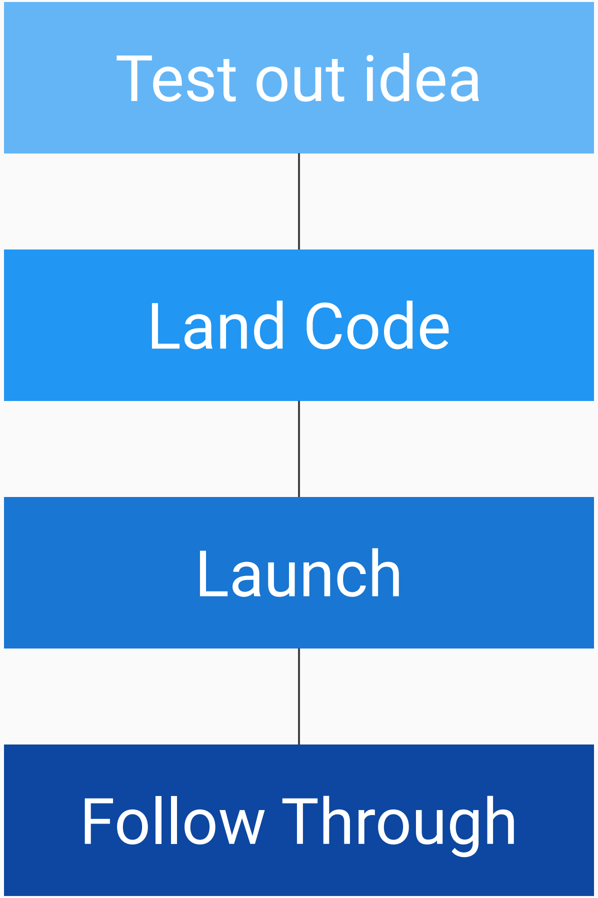
# Start with a Finch Trial



# Finch Trials: Best practices

- Check for statistical significance
- Balance the right amount of volume with enabling for too many users
- Turn on experiment for correctness and performance testing on bots.
- More Info:
  - Googlers: go/finch101
  - Non-Googlers: speed@chromium.org

# Story of a Speed Launch



Test out idea

Land Code

Launch

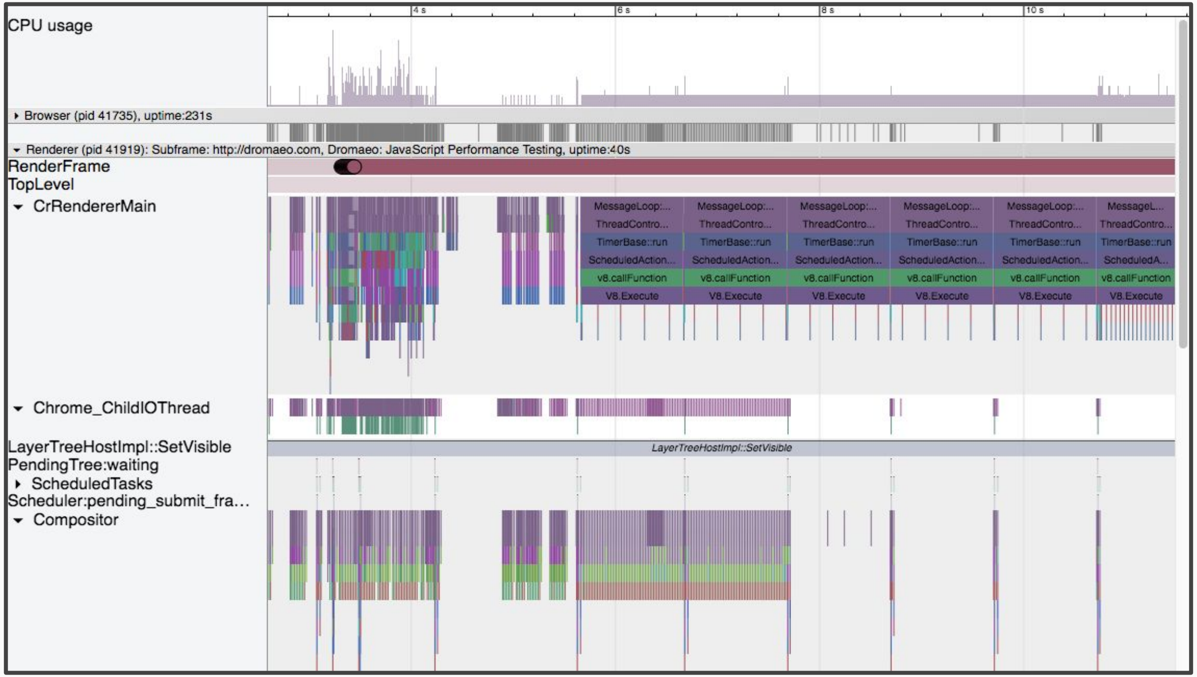
Follow Through

# Before you submit: Trying an idea

# Understanding performance: trace viewer

Use trace-viewer to get an **understanding of Chrome's performance.**

[bit.ly/trace-viewer](bit.ly/trace-viewer)


CPU usage
4 s
6 s
8 s
10 s
Browser (pid 41735), uptime:231s
Renderer (pid 41919): Subframe: http://dromaeo.com, Dromaeo: JavaScript Performance Testing, uptime:40s
RenderFrame
TopLevel
CrRendererMain
MessageLoop:...
ThreadContro...
TimerBase::run
ScheduledAction...
v8.callFunction
V8.Execute
Chrome_ChildIOThread
LayerTreeHostImpl::SetVisible
PendingTree:waiting
ScheduledTasks
Scheduler:pending_submit_fra...
Compositor
LayerTreeHostImpl::SetVisible

# Benchmarks: system health

Run **system health benchmarks** to measure **speed launch metrics** on real web content (90 minutes of popular desktop and mobile use cases).

Use the **perf try bots** to get performance numbers from a variety of real devices across Windows, Mac, Linux, and Android.

You can view tracers either locally or if running on perf try jobs.

bit.ly/chrome-system-health-benchmarks
bit.ly/chrome-perf-trybots

# Expanding to more pages: cluster telemetry

You can get **speed launch metrics** as well as many more fine-grained performance metrics on a **much larger corpus of sites** through cluster telemetry.

Traces are available as well.

Googlers: **go/cluster-telemetry**
Non-Googlers: ping **speed@chromium.org** for help

# Landing the code

# Landing the code: Perf Waterfall

After landing, code goes through a deeper battery of tests:

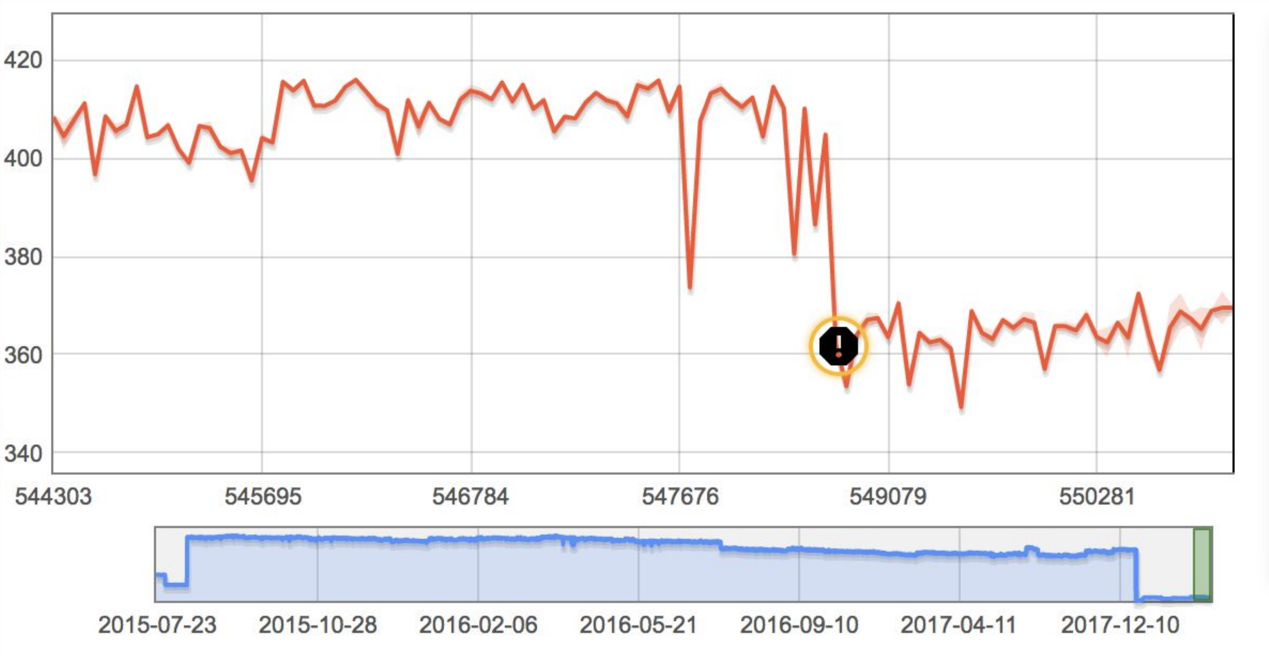
- ~30 hours of tests (bit.ly/chrome-benchmark-harnesses)
- 12 device configurations

Finch trials are turned on/off on perf waterfall via field trial testing config file: bit.ly/field-trial-testing-config

# Landing the code: Perf Dashboard

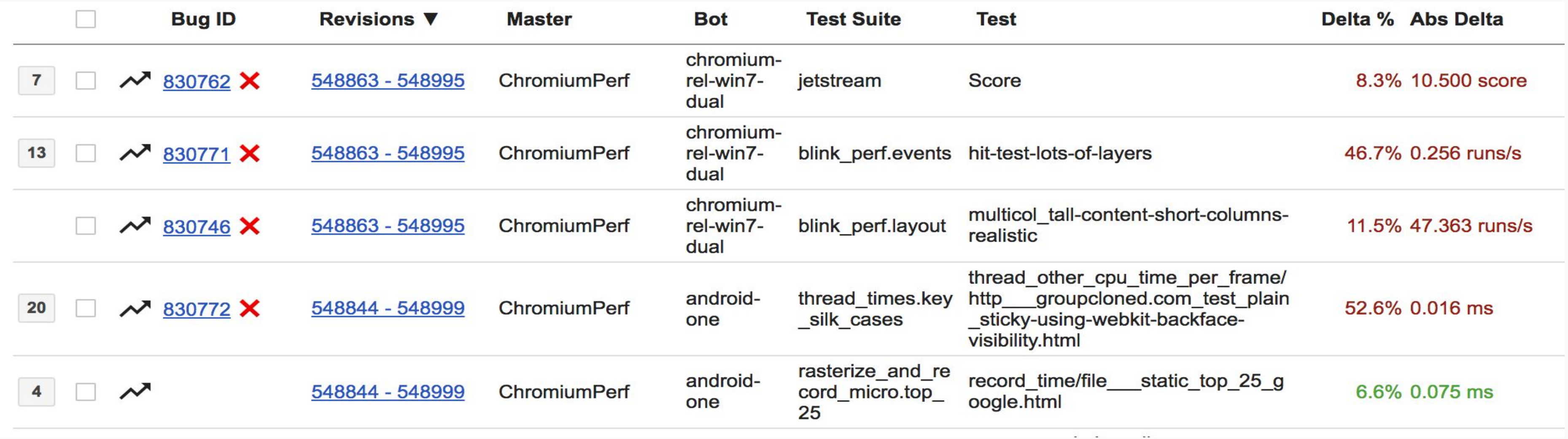
Perf dashboard automatically **detects regressions**.

Perf sheriffs file bugs.



# Landing the code: Perf Dashboard

See the full effects of your change at
https://chromeperf.appspot.com/group_report?rev=COMMIT_POS



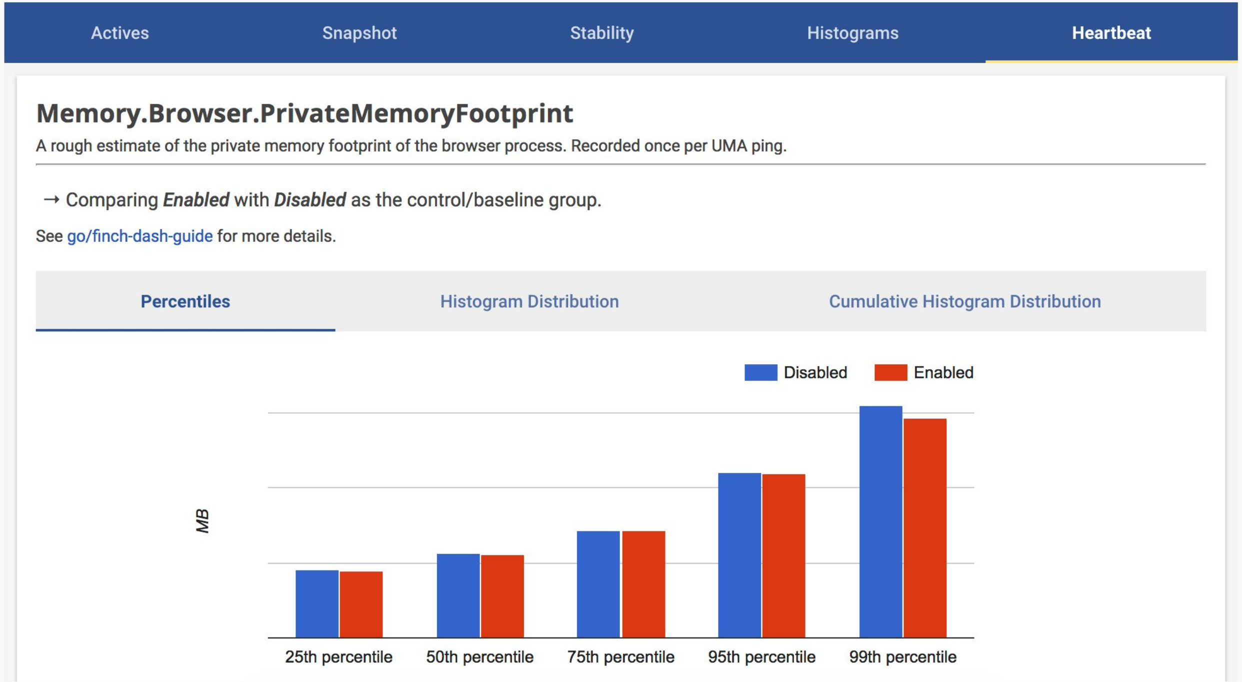
|  | □ |  | Bug ID | Revisions ▼ | Master | Bot | Test Suite | Test | Delta % | Abs Delta |
|---|---|---|---|---|---|---|---|---|---|---|
| 7 | □ | ↗ | 830762 ✗ | 548863 - 548995 | ChromiumPerf | chromium-rel-win7-dual | jetstream | Score | 8.3% | 10.500 score |
| 13 | □ | ↗ | 830771 ✗ | 548863 - 548995 | ChromiumPerf | chromium-rel-win7-dual | blink_perf.events | hit-test-lots-of-layers | 46.7% | 0.256 runs/s |
|  | □ | ↗ | 830746 ✗ | 548863 - 548995 | ChromiumPerf | chromium-rel-win7-dual | blink_perf.layout | multicol_tall-content-short-columns-realistic | 11.5% | 47.363 runs/s |
| 20 | □ | ↗ | 830772 ✗ | 548844 - 548999 | ChromiumPerf | android-one | thread_times.key_silk_cases | thread_other_cpu_time_per_frame/http___groupcloned.com_test_plain_sticky-using-webkit-backface-visibility.html | 52.6% | 0.016 ms |
| 4 | □ | ↗ |  | 548844 - 548999 | ChromiumPerf | android-one | rasterize_and_record_micro.top_25 | record_time/file___static_top_25_google.html | 6.6% | 0.075 ms |

# Launching

# In the wild: Finch trials

The **heartbeat tab** lists speed launch metrics impact of your finch trial.

Can slice by different platforms, device characteristics, etc.



**Memory.Browser.PrivateMemoryFootprint**

A rough estimate of the private memory footprint of the browser process. Recorded once per UMA ping.

→ Comparing ***Enabled*** with ***Disabled*** as the control/baseline group.

See go/finch-dash-guide for more details.

# Follow Through

# In the wild: We need your help!

If performance regressions slip through the cracks, we try to catch them!

- UMA speed launch metric regressions
  - Requires a lot of manual digging.
  - Very difficult to analyze. **Please use finch trials!**
- Out of process heap profiling - anonymized heap dumps from the wild
- Sampling profiler detects increases in execution time during startup

# Directions to improve on Speed

- **Optimize** the loading stack and renderer
- **Intervene** on poor page construction
- Provide **new APIs** (or restrictions) to encourage using **best practices**

**Talks and brainstorms:**

  - **Optimizing image decoding on ARM: 3 gens of silicon w/ software**
  - **Accelerate graphics performance with ozone-gbm on Linux desktop systems**
  - **Web Lifecycle & Interventions for CPU suspension, Throttling & Tab discarding**
  - **near-OOM intervention**
  - **Improving the Loading Experience (in EM)**
  - **How to make developers care about memory?**
  - **...**