Chromium GSoC 2024 Project Proposal:

Interaction to Next Paint (INP) "subparts"

Public - Non-confidential

Mentors: mmocny@, sullivan@google.com

Note#1: Optional equipment to send to the intern

- Testing devices for project: *None*
- Require GCP instance for faster builds: *Yes*
- Tryjob Access: *Yes*
- GOMA Access: *Yes*

Note#2: Please refer to FAQs at the end of this document. If you still have questions, please send an email to chromium-gsoc@chromium.org.

Project Overview

The web performance specifications define how browsers are supposed to be behaving when it comes to the various web performance APIs, and the Core Web Vitals (CWV) program uses those apis to automatically measure the performance and UX of most websites in the world, and shares the data publicly via the Chrome User Experience Report (CrUX).

Last year a new metric, Interaction to Next Paint (INP), was added to the CWV. This year, we would like your help to give developers additional insights into INP latency issues by also reporting INP "subparts", similar to what we did for LCP last year.

For example: a developer should know if any INP responsiveness issues are caused by:

- long input delay (maybe the page as blocked and busy)
- event processing times (maybe the interaction event listeners were too complex)
- rendering/pixel presentation delays (maybe the page content is too complex/bloated)
- …or other task scheduling issues.

We've recently instrumented the measurement code (i.e. the Event Timing API) to measure these time points, but we need your help to finish the job: moving the data from Renderer code to Browser code (using Mojo IPC), adding it to field metrics reporting (UKM), and eventually helping teammates integrate into CrUX experimental data – while also writing tests for the above.

This is an open source project and some of the contributions will be useful to developers for local lab testing, as well (as for other downstream browsers). Finally, there are also several

stretch technical opportunities related to the Event Timing API / INP metric.

Goals

- Work with GSoC mentors to clearly define the specific timestamps to be used for implementing INP subparts (using existing measurement data)
    - Start here to see how we use timings in existing lab-only tracing code.
- Move the "INP algorithm" from Blink (Renderer process) to the Browser process
    - This is how we currently report the page INP metric candidates, and eventually sent from Renderer to Browser using this mojo message struct. This format is not sufficient for reporting sub-parts, so:
    - Update PageLoadMetricsSender mojo struct:
        - Change to report **each part of each event timing** (with non-0 interactionID), rather than just a single duration value.
        - Update blink/Renderer side "plumbing" to migrate these values from window_performance.cc / responsiveness_metrics.cc to the PageTimingMetricsSender.
    - From Browser, update the UkmPageLoadMetricsObserver:
        - Change PageLoadMetricsUpdateDispatcher::UpdatePageInputTiming and ResponsivenessMetricsNormalization helper to use this new mojo struct format in order to re-implement the existing UKM reporting of INP and NumInteractions.
        - (There are other PageLoadMetricsObserver types which may require similar updates.)
    - Ensure existing tests continue to pass.
    - (Stretch: we also report each Interaction candidate from renderer and might want to make similar changes.)
- Add Subparts to UKM
    - Similar to above, but now add more timestamps and new reporting mechanisms.
    - Update mojo struct, plumbing, and UkmPageLoadMetricsObserver to report these values.
    - Add relevant tests.
    - Update UKM.xml to add this new data.
        - If you are not familiar with UKM, there are some public docs
        - You can access your own UKM locally with chrome://ukm (you will be prompted to enable debug pages first)
- Stretch: Work with GSoC mentors to add these INP subparts to CrUX (as part of an "experimental" dataset)

Non-goals

- Changing Event Timing or INP definition itself.

Platforms

All platforms are supported, but the implementation is platform agnostic. GCP machine will likely be granted to improve Chromium compile performance, which is based on linux, but it is possible to build Chromium also on Windows/Mac if needed.

Technology Stack

C++ for the Chromium codebase. JavaScript needed for the tests (and for general testing). Chromium DevTools performance profiler likely useful for local experimentation.

Team

Chrome Speed Metrics.

Bugs

- https://issues.chromium.org/issues/40858679
-

Difficulty Level

Medium

Code affected

- blink/renderer/core/timing (window_performance.cc, responsiveness_metrics.cc)
- components/page_load_metrics/renderer/page_timing_metrics_sender.cc
- chrome/browser/page_load_metrics/observers/core/ukm_page_load_metrics_observer.cc

Requirements

GSoC contributors for this project would need to read C++ code, and be able to debug it. Some familiarity with HTML and JS is also required, as those are the languages in which the tests are built.

Expectations from the GSoC Contributor

The contributor is expected to be committed to make consistent progress towards the goal. The mentors would be happy with any method of periodic communication the contributor would find useful, and would be happy to help with pointers, explanations, etc to avoid the contributor getting stuck. No TZ restrictions, but the hosts are all in Eastern time zone (EST).

FAQs

Q: I am interested in this project, what should I do now?
A: Start by reading the documents for INP and CWV and trying to use the DevTools performance panel to measure responsiveness on any live website. When you identify a slow interaction, attempt to use the performance profiler to measure the interaction. Even if you cannot find the source of the responsiveness issue or fix the performance problem, at least see if you can understand what the long duration represents (using tools like the Interactions track, the Main thread track, and Screenshots) and/or where the time was spent.

Q: Would I need to build Chromium for that?
A: I'm afraid so :) See https://www.chromium.org/developers/how-tos/get-the-code/ for the official guide, or https://meowni.ca/posts/chromium-101/ for a more fun one (which may be a bit outdated)

Q: I made some progress on local benchmarking and I like the problem space… now what?
A: Great work! Now, it is time to write your proposal. Do not forget to include the link for the work you did (either code, docs or both) in the proposal.