RendererMainThreadLoad metrics

altimin@chromium.org

July 11, 2017

This document describes the family of metrics measuring the load of the renderer main thread. At the moment this family consists of RendererScheduler.RendererMainThreadLoad3.* metrics.

Metric design

This metric is used for monitoring how busy is a renderer in a particular situation. The original motivation was to measure activity of background renderers to understand how effective our background throttling efforts are.

This metric is reported as an average load (0% - 100%) in blocks of a fixed length (1 second for current version). This metric is rounded down to nearest integer and uploaded to UMA.

Implementation of this metric is based on a concept of not increasing a number of renderer wake-ups, which rules out running a timer to report this metric. Each time a task processed, a load in current block is updated, and last known time is increased. If last known time exceeds right boundary of the current block, a result is added to UMA. Note that per this design, a long idle period will result in a series of notifications after the first task after this idle period is processed.

There is a one big known problem affecting this metric — system falling asleep while the timer continues to increase. This can lead to hour-long tasks (if the system fell asleep in a middle of a task) or hour-long idle periods. This can seriously distort data, especially on Canary channel, especially in higher percentiles. To work around this, two heuristics are employed:

- If a task takes longer than 30 seconds, it's considered to be a fake one and discarded. This threshold matches other metrics employing the same idea.
- If an idle period is longer than N+1 minutes, it's considered to be a fake one and discarded. This is based on an idea of the browser pinging renderer every N minutes to check if it's alive (handling this ping IPC will stop an idle period).

Note

First two versions are known to contain garbage data coming from a handful of clients, especially on Android. Last version (RendererMainThreadLoad3) should fare better in this regard, but users should still be aware of this and manually filter out clients with anomalously large number of samples if needed.

List of metrics

- RendererScheduler.RendererMainThreadLoad4

- RendererScheduler.RendererMainThreadLoad4.Foreground
- RendererScheduler.RendererMainThreadLoad4.Background
- RendererScheduler.RendererMainThreadLoad4.Foreground.AfterFirstMinute
- RendererScheduler.RendererMainThreadLoad4.Background.AfterFirstMinute

Version history

RendererMainThreadLoad5

Special support for extensions renderers, with Extension, Extension.Foreground and Extension.Background suffixes.

RendererMainThreadLoad4

Switched from 1-minute reporting intervals to 1-second reporting intervals.

RendererMainThreadLoad3

Added discarding of long idle periods to prevent overreporting idle tiime. Naming scheme was refactored to have `RendererScheduler.RendererMainThreadLoad` prefix for all histograms.

RendererScheduler.{Foreground,Background}RendererMainThreadLoad2

Added discarding of long idle tasks to prevent reporting unrealistic numbers when system fell asleep for a long time in a middle of a task.

These metrics discarded first minute after foregrounding/backgrounding as they were trying to measure usage in idle state and discard loading stage. Compare them with RendererScheduler.RendererMainThreadLoad3.{Foreground,Background}.AfterFirstMinute.

This metric is known to contain some strange data coming from isolated number of clients. It's recommended to filter out manually clients with anomalously high number of samples.

RendererScheduler.{Foreground,Background}RendererMainThreadLoad

Initial version.

There's been a number of bugs in supporting measurement infrastructure that have been tweaked during the lifetime of this metric. It's not recommended to use this metric, in particularly you shouldn't aggregate it across different Chrome versions.