# Better scheduling metrics

altimin@chromium.org
Jun 11, 2018

This document describes a proposal to reduce the overhead for scheduling metrics and expand the usage of thread time in these metrics.

## Scheduling metrics

See MainThreadMetricsHelper for the details.

Scheduling metrics can be divided into two categories:

- Per-task metrics (e.g. RendererScheduler.TaskDurationPer* family of metrics).
- Integral metrics, which take into account multiple tasks (e.g. RendererScheduler.RendererMainThreadLoad5.* family).

The majority of the metrics belong to the first category.

All integral metrics and the majority of per-task metrics use wall time. Thread time is used for a small number of metrics (RendererScheduler.TaskCPUDurationPer*) and these tasks are sampled with 0.01 rate, all other metrics are recorded for each task.

## Current state

Benchmark results from the waterfall suggest that the cost of posting and running a scheduling task is 20-30us on desktop and 60-130us for Android, with ~12-13% of this being spent on recording metrics.

Sampling profiler suggests that cpu time spent on recording metrics is 0.5% for process startup and 1.7% for periodic collection.

## Acceptable overhead

It is proposed to set acceptable overhead as 1% of total scheduling cost to run a task, therefore a 10x reduction in the overhead is desired compared to the current situation.

## Per-task metrics

Given that recording metrics for one task is not prohibitively expensive, it is proposed to just reduce the number of tasks the metric is recorded for to 1% through random sampling with corresponding rate. Thread time sampling rate won't change and thread time will be recorded for all tasks for which per-task metrics are recorded.

Note: thread-time measurement cost is very high on OS X, but 20us -> 30us increase for 1% of tasks seems acceptable, but closer monitoring is needed.

### Integral metrics

Existing integral metrics (RendererScheduler.RendererMainThreadLoad5) are sufficiently lightweight to be always recorded (and per-task wall time measurement is needed anyway for other scheduling logic like throttling).

It is proposed to add a thread-time-based counterpart, which required to record thread time for each task. This also has the benefit of exposing the total amount of work not tracked by the blink scheduler, including task time observers, microtasks, message loop overhead etc, which can serve as a power usage proxy and can help with understanding regressions (e.g. crbug.com/849758).

Thread time measurement can be expensive, so it is proposed to enable this for 0.01% of renderers (and maybe 1% for Canary&Dev if the amount of data seems to be insufficient).

Note: this may be too expensive on OS X, a closer monitoring is needed.

### Miscellaneous thoughts

#### EQT

At the moment EQT is calculated outside MainThreadMetricsHelper and was not included in the “metrics” section and it remains a follow-up.

#### ResourceCoordinator

EQT and RendererMainThreadLoad metrics are used by RC. EQT is forwarded to the browser process where it’s used for further metrics recording. RendererMainThreadLoad is used for PageAlmostIdle signal calculation.