[PUBLIC] Freeze in Background on Android

Authors: panicker@chromium.org

One-page overview

Summary

Currently timer tasks (on stable) and loading (on stable) is stopped in background after 5 minutes grace time, on Android.
This feature extends this to all (per-frame) task-queues -- well beyond timers and loading.

Link to Intent to Ship:

https://groups.google.com/a/chromium.org/d/msg/blink-dev/NKtuFxLsKgo/brL3bfS5CAAJ

Platforms

Mobile: Android.

Team

panicker@google.com, shaseley@google.com

Bug

https://bugs.chromium.org/p/chromium/issues/detail?id=822954

Code affected

The condition is triggered in Blink Scheduler and additional task queues (i.e. deferrable and pauseable task queues) are stopped.

Motivation

Many sites do not stop activity, even after their app has been in background for over 5 minutes (a generous grace time), this consumes non-trivial amount of CPU and hurts the responsiveness of foreground app.
After five minutes in the background, FrameDeferrable and FramePausable account for 15.44% of background work: see UMA.

In background:

RendererSchedulerTaskQueueType	Count	PDF
Default	2,234,800	07.73%
FrameLoading	7,135,117	24.68%
Compositor	1,331,790	04.61%
Idle	172,596	00.60%
FrameLoadingControl	20,647	00.07%
FrameThrottleable	5,402,092	18.69%
FrameDeferrable	1,047,436	03.62%
FramePausable	1,973,843	06.83%
FrameUnpausable	986	00.00%
V8	4,482,759	15.51%
IPC	4,873,095	16.86%
Input	210,996	00.73%
Detached	22,277	00.08%

After five minutes in background:

RendererSchedulerTaskQueueType	Count	PDF
Default	1,251,999	16.48%
FrameLoading	364,210	04.79%
Compositor	957,466	12.61%
Idle	54,001	00.71%
FrameLoadingControl	154	00.00%
FrameThrottleable	20,170	00.27%
FrameDeferrable	243,330	03.20%
FramePausable	929,924	12.24%
FrameUnpausable	67	00.00%
V8	2,176,617	28.66%
IPC	1,532,195	20.17%
Input	65,621	00.86%
Detached	151	00.00%

Under the umbrella of Web Lifecycle we will stop background work in cases where it would improve the foreground user experience -- starting with mobile (and eventually followed by desktop).
The motivation for this is to:
- improve the responsiveness of the foreground tab
- conserve data (on data connections)
- conserve device battery life

Design

Implementation:

Freeze non-timer task queues in Blink Scheduler:
when threshold (5 mins) have elapsed in background, stop the *deferrable* and *pauseable* task queues in FrameSchedulerImpl:
https://cs.chromium.org/chromium/src/third_party/blink/renderer/platform/scheduler/main_thread/frame_scheduler_impl.cc?q=StopNonTimers&l=508

Metrics

Success metrics

We will watch the following success metrics:

Reduced CPU usage in background

RendererScheduler.TaskDurationPerQueueType2.Background.AfterFifthMinute (M68 Beta)
RendererScheduler.TaskDurationPerQueueType3.Background.AfterFifthMinute (M69)
RendererScheduler.TaskDurationPerTaskType2.Background.AfterFifthMinute (M69)

Note July 23, 2018:
UMAs in Canary and Dev look good. Pausable and deferrable tasks have indeed significantly decreases, as has IPC. Total count (all task types) showing 45% decrease (per queue type) and 61% decrease (per task type). Note: RendererScheduler.TaskDurationPerTaskType2 was added after per-queue type metrics.

Note August 22, 2018:
UMAs in stable (M68) and beta (M69) show significant decreases in pausable, deferrable, and IPC task queues after 5th minute in background, which corresponds to a decrease in CPU. Total count (all task types) is showing -39.13% on stable and -72.58% on beta. The

stark difference between beta and stable is attributed to significantly more compositor background work in M68 (fixed in M69). This intervention does not have an impact on compositor work, so the percentage of total task duration that it can affect in M68 is lower than M69 resulting in this discrepancy.

"Multi-tab loading" for foreground

FCP:
PageLoad.Clients.MultiTabLoading.2OrMore.PaintTiming.NavigationToFirstContentfulPaint
PageLoad.Clients.MultiTabLoading.5OrMore.PaintTiming.NavigationToFirstContentfulPaint
FMP:
PageLoad.Clients.MultiTabLoading.2OrMore.PaintTiming.NavigationToFirstMeaningfulPaint
PageLoad.Clients.MultiTabLoading.5OrMore.PaintTiming.NavigationToFirstMeaningfulPaint

Note July 23, 2018:
UMAs in Canary and Dev show an ostensibly significant decrease in the higher percentiles (13% - 17% for 99th percentile across all metrics). However, for some there is a significant difference in counts between the control and experiment groups. Will continue to monitor.

Note August 22, 2018:
Stable: For 2 or more tabs, the UMAs show a significant decrease in the 25th (-8.60%) and 50th (-6.12%) percentiles, small decreases in the 75th and 95th percentiles, and a noisy 99th percentile. For 5 or more tabs, we see significant decreases in the 50th (-19.34%) and 75th (-17.65%) percentiles, and decreases (noisy) in the other percentiles. There is a rather high (25.84%) difference in counts, which are rather low to begin with, be cautious in drawing any conclusions here.

Beta: On beta, the 2 or more case does not show any movement and is fairly noisy. The 5 or more tab case shows (noisy) decreases in most cases, but here also there is a high difference in counts which are also rather low to begin with.

Scroll Latency and Responsiveness for foreground renderer

Event.Latency.ScrollBegin.TimeToScrollUpdateSwapBegin2
Event.Latency.ScrollUpdate.TimeToScrollUpdateSwapBegin2
PageLoad.PaintTiming.NavigationToFirstContentfulPaint
PageLoad.Experimental.PaintTiming.NavigationToFirstMeaningfulPaint
RendererScheduler.ExpectedQueueingTimeByFrameStatus2.MainFrameVisible

Note July 23, 2018:
These UMAs on Canary and Dev do not appear to show any statistically significant movement. Some percentiles for EQT regress a bit (50th +5.55%), but improve slightly in 99th percentile, which is most likely noise. Will continue to monitor.

Note August 22, 2018:
Stable: No significant movement in any of these metrics.
Beta: No significant movement in any of these metrics.

Regression metrics

If the user were to switch from foreground to a background tab (that underwent this intervention) their time to interaction should not regress much.
PageLoad.Experimental.PaintTiming.ForegroundToFirstMeaningfulPaint
PageLoad.PaintTiming.ForegroundToFirstContentfulPaint

Note July 23, 2018:
The UMAs on Canary and Dev do not show any regressions.

Note August 22, 2018:
Stable: No significant movement in any of these metrics.
Beta: No significant movement in any of these metrics.

Experiments

Flag addition CL
Finch study CL

Rollout plan

Standard experiment-controlled rollout with flag:
“stop-non-timers-in-background”

Intervention Guidelines

Predictability:
The mechanism is well defined and predictable, same as stopping timers and loading in background. onfreeze / onresume Lifecycle callbacks will be called to notify apps.

Avoidability:
Continuing to do work in background after 5 minutes, is not a good practice, especially on mobile -- for data and battery, in addition to hurting responsiveness of foreground app.

Transparency:
onfreeze / onresume Lifecycle callbacks will be called to notify apps. We are also looking into notification via Reporting API.

Data-driven:
We will use our success and regression metrics to inform moving forward with this intervention.

Privacy considerations

N/A

Testing plan

Added unit-tests.
Manually tested.
Also the experiment has been vetted by running on canary, dev, beta, stable over last months.