Heuristics for Freezing & Discarding

chrisha@chromium.org, panicker@chromium.org
Last updated: Aug 30, 2018

Introduction

This document describes the features used to build heuristics in order to determine if a tab is safe to freeze and / or discard in Chrome.
This list is not intended to be a final exhaustive set but rather what is currently in use, which may change over time.

User Harm

There are two useful distinctions to make in describing features for detecting potential user harm in performing a discard or other similar intervention:

- **Transient features.** Pending form data would be an example of this. This is a transient state that a site can be in, during which time it is not safe to perform the intervention. This state depends on an explicit sequence of user actions, and is not a persistent property of the site.
- **Persistent features.** Usage of certain web platform APIs would be an example of this. These are features of the site itself, that are (usually) independent of user actions.

Transient Features

The features of interest are largely concerned with ways in which a site is continuously doing work on behalf of a user, even while backgrounded. Alternatively, they may also be about ways in which user state is being held by the tab which would be lost if unloaded.

These features do not need to be persistently tracked but only monitored as a tab is alive. The positive presence of any of these features would disallow the tab from being discarded. These features are transient in that they may switch state during any given session of user interaction with a tab.

Unsubmitted User Input

This feature tracks whether or not the user has filled out any form data on the page. Note that this feature blocks discarding from occurring, but does not block suspending. This can currently be detected by observing the PageImportanceSignals associated with a WebContents. Support needs to be added for the *contenteditable* attribute.

Current Use of WebRTC API

Sites can use the Web Real Time Communications APIs to continuously perform useful work on behalf of the user. Many uses of this API will be associated with audio playing, and will be detected by that feature. However, it is possible that the API is being used to stream content to another destination on behalf of the user. These cases can be detected by observing the MediaStreamCaptureIndicator associated with a WebContents.

Current Use of WebSockets API

A website making active use of the WebSockets API is usually continuously sending and/or receiving data. Interrupting this connection can cause loss of user data, loss of work, missed notifications, etc. Tabs using this API should be protected from suspending and discarding.

Current Use of WebUSB API

A website making active use of the WebUSB API is potentially performing useful work on behalf of the user, interacting with external hardware. An example is the Android device flash station web app. Tabs using this API should be protected from suspending and discarding.

Current Use of Audio

Actively playing user-audible audio is a way for a tab to be semantically in the foreground, despite not actually being visible. A common usage pattern for music sites like Google Play and Spotify is for the user to start audio, and then to background or otherwise minimize the tab. This condition is easy to detect using the “RecentlyAudibleHelper” API. Tabs actively playing audio should be protected from suspending and discarding.

Pending state in BeforeUnload Handler

BeforeUnload handlers are a mechanism by which sites are able to do work or communicate with a user when the user navigates away from a page, or otherwise attempts to close the page. Beforeunload handler will be invoked at the time of freezing to determine if there is pending data and consequently risk of data loss. If there is no pending data, then the page will be frozen and can be subsequently discarded. If there is pending data, then the page will be frozen but will not be discarded to avoid data loss.

Persistent Features

Tracking bug: https://bugs.chromium.org/p/chromium/issues/detail?id=731270

The features of interest are largely concerned with ways that a backgrounded tab may continue to communicate with a user, or with ways in which it may perform useful work.

When collecting these features it is generally important to distinguish between sites that do something directly as a result of user interaction, and sites that continue to do something in an automated fashion when they do not have focus. It is also useful to know how often these actions occur, in order to inform how long an observation window is reasonable for newly seen sites.

All of the following features are concerned with observing events that occur when a tab is backgrounded. Tabs can be kept in the background a long time[1] and in order to guide the selection of an appropriate observation window we are proposing using a CUSTOM_TIMES histogram from 1s to 24h, with 100 buckets.

Each feature will be tracked with 3 individual metrics:

- **UKM: TabManager.Heuristics.*FeatureName*.RootFrame**
  Records the cumulative amount of time a page has spent in the background when the

[1] The 75th percentile of time spent in background is 2.5 hours, and the 95th is 24 hours. See the *TabManager.Discarding.InactiveToReloadTime* metric for more information.

signal was observed, when it was associated with the root frame of a tab. This metric will inform the crowd-sourced database. This is tied to the URL of the root frame, and is only observed for backgrounded tabs.

- **UKM: TabManager.Heuristics.*FeatureName*.ChildFrame**
  Records the cumulative amount of time a page has spent in the background when the signal was observed, when it was associated with a child frame of a tab. This metric will inform the crowd-sourced database. It will also be useful in tracking how often the feature observation is due to secondary content. This is tied to the URL of the root frame [2], and is only observed for backgrounded tabs.
- **UMA: TabManager.Heuristics.*FeatureName***
  Records the cumulative amount of time a page has spent in the background when the signal was observed. This distribution will inform the observation period that will be used by the local database heuristic. This is only observed for backgrounded tabs.

Updating Favicon

On the desktop platform sites are able to communicate with the user by modifying the favicon. GMail used to do this (it no longer does), but other sites do. See here for an example of how sites typically accomplish this.

Some sites may do a single favicon update as part of their loading process. Ignoring the first favicon update (whether foreground or background) of a site mitigates against this.

Feature: BackgroundFaviconUpdates

- Measures the cumulative amount of time a tab has spent in background when an update of its favicon has occurred, ignoring the very first favicon update performed on the site. Only observed while a tab is backgrounded.

Updating Title

Much like favicon updates, sites are able to communicate with the user via updates to the title. Most commonly this is used by sites like Gmail and Facebook to indicate a pending number of unread messages or notifications.

Some sites may do a single title update as part of their loading process. Ignoring the first title update (whether foreground or background) of a site mitigates against this.

Feature: BackgroundTitleUpdates

---

[2] UKM's privacy policy forbids linking records to the URL of a child frame. This makes it impossible to know what embedded content is responsible for a feature-positive signal. However, by separating the metric into root- and child-frame variants, it is at least possible to know if the signal originates from embedded secondary content or primary content. This can be used to prioritize which APIs to develop and which heuristic features to retire first.

- Measures the cumulative amount of time a tab has spent in background when an update of its title has occurred, ignoring the very first title update performed by the site. Only observed while a tab is backgrounded.

Background Use of Audio

Sites can use audio to communicate with a user even when they don't have focus. This is commonly used by sites like Facebook in order to provide an aural signal that a new message has arrived.

Feature: BackgroundAudioStarts

- Measures the cumulative amount of time a tab has spent in background when audio starts playing. Audio is considered to have started playing if the tab has never previously played audio, or has been silent for at least one minute. Only observed while a tab is backgrounded.

Use of Notifications

Sites can use the Web Notifications API to display notifications to the user, typically outside of the context of the tab and integrated with the platform native notification system when possible. The API can be accessed directly from running tabs and also via Service Workers. Notifications launched from Service Workers are compatible with tab lifecycles, and thus do not have to be tracked.

Feature: SiteNotifications

- Measures the cumulative amount of time a tab has spent in background when a notification is posted. Only observed while a tab is backgrounded.