README

[Public]

PreloadServingMetrics: A Unified Logging Framework for Preloading Serving Metrics

Author: kenoss@
Input: hiroshige@, nhiroki@, taiyo@, kouhei@, <your-name>@,
Status: Draft
Created: Aug 5, 2025
Updated: Aug 5, 2025

About this document

This document proposes a unified logging framework called `PreloadServingMetrics` to capture detailed information about how preloaded resources (specifically from prefetch and prerender) are used by subsequent navigations. This will provide crucial data for performance analysis and debugging.

Previous document: 2024Q4 Prefetch Idea Dumps
WIP CL: https://chromium-review.googlesource.com/c/chromium/src/+/6814289

Motivation

There are three view points of existing metrics of preloads (not exclusive):

- A. Metrics of execution layer of preloads
  - Used to get knowing relatively rough statistics of the components.
  - Preloading.Prefetch.PrefetchStatus: Records prefetch status of `PrefetchContainer`.
  - Preloading.Prefetch.PrefetchDataPipeTeeCloneFailed.<state>: Records failures of `PrefetchDataPipeTee` in <state>. Used for launch of `PrefetchResubale`.
  - Prerender.Experimental.MatchableHostCountOnActivation: Records counts of candidates of prerender activation.
- B. Per-trigger (trigger suffix)
  - Used to analyze the triggered preloads and the behavior of the execution stack with on a per-trigger.
  - Prefetch.PrefetchContainer.AddedToPrefetchStarted.<trigger-type-and-eagerness>: Records duration from prefetch triggered to prefetch started. Used for `PrefetchScheduler` and CCT.

    - Prerender.Experimental.ActivationNavigationParamsMatch.<suffix>: Records reason for prerender header mismatch.
- C. Per-navigation
    - We want to analyze the behavior of navigation that uses preloads and compare it to ones that don't.
    - PrerenderPageLoadMetricsObserver
    - PrefetchServingPageMetrics, but we believe that it is broken (when the PrefetchServingPageMetricsContainer is registered to PrefetchContainer that is already entangled with PSPMContainer); and more broken by recent updates of prefetch stack, e.g. PrefetchReusable.

Note that A vs B is also discussed in
Unified Prefetch Cache: Prefetch Time or Serving Time?. In this document, tentatively, we follow the term "prefetch time" and "serving time".

We daily debug the preloading stack for experiments. We mainly use A and B with great ingenuity, but inherently they are **point**s of data. In a difficult situation, we sometimes need a **history** of events/data, and want to collect/analyze data with a bias of target situation.

Simple examples:

- Investigating what is happening when prefetch matching is timed out, or very slow. (Bad effect on FCP/LCP.)
- Investigating what is happening when prerender initial navigation failed, but prefetch ahead of prerender looks healthy.

Currently, we lack C for navigation used prefetch, navigation used prerender with prefetch ahead of prerender, and failed prerender initial navigation.
This document proposes a mechanism to collect logs for them.

Relation to the other projects

Chrome Navigation Metrics Improvements

1. These metrics improvements are breakdown of periods of a navigation. Some of them work for navigation served by preloads, but some of them don’t. It’s because

- Loading of a resource and serving for a real navigation is decoupled.
- Navigation can be served by multiple preloads.
    - Prefetch ahead of prerender and prerender: Prerender initial navigation is served by prefetch, and a real navigation is served by prerender.
- Navigation can involve multiple prefetches.
    - Prefetch matching is tried per redirect.

- A prefetch can serve multiple navigation.

2. The purposes are different. PrefetchServingLog aims for comparison and debugging.

So, at least in the short-term, we'll focus on providing a logging mechanism and variants of UMAs that work for navigation served by preloads, e.g. NavigationToFirstContentfulPaint.

Future work:

- We might add such variants, e.g. NavigationStartToFinish.
  - For comparison.
- We might add variants of type A (in Motivation), e.g. LoaderStartToReceiveResponse.
  - We make them as variants because the code paths are different. We will use them to improve loading of preloads.

We don't have a clear plan to unify them so far.

Requirements

- We want to record logs of navigation of these types:
  - Navigation without preloads, committed.
  - Navigation with prefetch, committed.
  - Navigation with prerender (prerender activation), committed.
  - Prerender initial navigation, failed.
- We want to distinguish logs of navigation when:
  - (Prefetch loading is slow.)
  - Prefetch matching is slow.
  - Using prefetch failed somewhere after prefetch matching.
  - etc.
- We want to record
  - Duration from prefetch is triggered to start of prefetch matching.
  - etc.

Philosophy

We should distinguish between navigation and `NavigationHandle`/`NavigationRequest`:

- (Main frame) navigation (in this document)
  - (Very roughly) A unit having an effect to refresh the contents of WebContents shown.
  - Non prerender navigation or (prerender initial navigation +) prerender activation navigation.
- `NavigationHandle`/`NavigationRequest`
  - A unit having an effect to refresh the contents of `FrameTreeNode`.

    - If a navigation “uses” prerender, there are two `NavigationHandle`/`NavigationRequest` for prerender initial navigation and prerender activation navigation.

The philosophy of `PreloadServingMetrics` are

- 1. Provide a way to attach data to navigation.
    - Currently, we have `NavigationHandleUserData`, but it lacks prerender support. See below.
- 2. Use it for metrics of prefetch/prerender.

Design overview

The core idea is to generate logs within the preloading components at prefetch/prerender time, and attach them to the `NavigationHandle`, and then record them as UMA metrics at serving time.

We can use `NavigationHandleUserData` (`NavigationPreloadLogHolder` for our purpose) to pass them at serving time.

```
None
void PrefetchMatchResolver::UnblockInternal(...) {
...
  auto* ftn = FrameTreeNode::GloballyFindByID(ftn_id_); // (*)
  if (!ftn) {
    return;
  }

  NavigationRequest* navigation_request =
ftn->navigation_request(); // (**)
  if (!navigation_request) {
    return;
  }

  auto& nav_preload_log_holder =
    *NavigationPreloadLogHolder::GetOrCreateForNavigationHandle(
        *navigation_request);

  nav_preload_log_holder.AddPrefetchMatchLog(
```

```
    std::move(prefetch_match_log_));
...
}
```

None

```
+--------------------------+
|           UMA            |
+--------------------------+
```

Key classes and structs

We'll describe the details in the following sections. Here, we'll only show an overview.

- Logs
  - `NavigationPreloadLog`: Log of preloads related to a navigation. May hold `PrefetchMatchLog`, `NavigationPreloadLog` for prerender initial navigation (if the navigation is prerender activation navigation).
  - `PrefetchMatchLog`: Log of prefetch matching. May hold `PrefetchContainerLog`.
  - `PrefetchContainerLog`: Log of `PrefetchContainer`.
- `NavigationPreloadLogHolder`: Holds `NavigationPreloadLog` to collect logs in navigation.
- `NavigaitonPreloadLogCapsule` (//content public): Allows `PageLoadMetricsObserver` to get/hold/record `NavigationPreloadLog`.
- `NavigationPreloadLogPageLoadMetricsObserver`: Record logs when a navigation is **committed**.

Handling Prerender

There are two roadblocks around prerender.

Passing logs from initial navigation to activation

The prerender initial navigation and the activation navigation are two distinct navigations with separate `NavigationRequest` objects. To connect them, we will store the `NavigationPreloadLog` from the initial navigation in the `PrerenderHost`. When the page is activated, the log is moved to the `NavigationPreloadLogHolder` of the activation navigation.

None

```
void PrerenderHost::DidFinishNavigation(NavigationHandle*
navigation_handle) {
  auto* navigation_request =
NavigationRequest::From(navigation_handle);

  auto& initial_navigation_preload_log_holder =

*NavigationPreloadLogHolder::GetOrCreateForNavigationHandle(
          *navigation_handle);
  prerender_initial_navigation_preload_log_ =
      initial_navigation_preload_log_holder.Take();

  ...
}

std::unique_ptr<StoredPage> PrerenderHost::Activate(
    NavigationRequest& navigation_request) {
  ...

  // Before cut-and-pasting frame trees, move `PreloadLog`.
  auto& activation_navigation_preload_log_holder =

*NavigationPreloadLogHolder::GetOrCreateForNavigationHandle(
          navigation_request);

activation_navigation_preload_log_holder.SetLogOfPrerenderInitial
Navigation(
      std::move(prerender_initial_navigation_preload_log_));

  ...
}
```

Recording logs for failed prerender initial navigation

PageLoadMetricsObserver (PLMO) can observe only the prerender that is used for a navigation. So, we need another path for failed prerender initial navigation.

We will introduce a new method, `PrerenderHost::OnWillBeCancelled()`, which will be called from `PrerenderHostRegistry::CancelHost()`.
Also, we will add an additional argument `NavigationHandle*` to them, which is non null if we can get it.

Visibility and layering

PLMO is in outside //content. So, we need to make something //content public.

Log structure is relatively large and may contain //content non public data. So, we encapsulate them into //content public `NavigationPreloadLogCapsule`.

All recording logic is in `NavigationPreloadLog`, including FCP.

~~Use of `FrameTreeNodeId` and `FrameTreeNode`~~

~~To get `NavigationRequest`, e.g. in (*) and (**), we use frame tree node id and frame tree node.~~

~~We use this pattern in `PrefetchMatchResolver`. It is safe because `PrefetchMatchResolver` is created per navigation; and `PrerenderURLLOaderThrottle` works if the navigation is prerender initial navigation and it has a dedicated frame tree node.~~

Alternative considered

NavigationHandleUserData-like object that supports prerender

For Philosophy-1, we can consider providing a `NavigationHandleUserData`-like object. This might be nice for the long-term, but it's too generic for the short-time. We limit the scope to logs for preloads.

Linear, unstructured log and then parse it

`NavigationPreloadLog` is structured. Alternatively, we can consider **true history**, linear and unstructured log like the following:

```
None
{time = 2025-01-01T01:00:01Z, event = PrefetchContair::Added, value = ...}
{time = 2025-01-01T01:00:05Z, event = PrefetchMatchResolver::MatchStart, value
= ...}
{time = 2025-01-01T01:00:08Z, event = PrefetchContainer::ReceivedHeader, value
= ...}
```

```
...
```

It is too complicated. Logging is easy but we need to parse it before recording UMAs.

Rejected. We use structs like `PrefetchMatchLog` that are enough to log UMAs.

Shared handle to add data to NavigationPreloadLog

Consider adding some logs of `PrefetchStreamingURLLoader` or `PrefetchDataPipeTee`. These components are far from prefetch matching and lifetimes are not bounded by PrefetchMatchResolver. We'd like to pass data to these components to add logs, but also like to avoid using `FrameTreeNodeId` to log it [section].

So, it would be nice to add a handle containing `base::scoped_refptr<NavigationPreloadLog>`. Pass it to potentially matched `PrefetchContainer` and `PrefetchStreamingURLLoader` and to actually matched ones and `PrefetchDataPipeTee`, and then add logs using the handle.

Initially, we don't need it, but we might add it in the future.

Version suffix of UMAs

Almost all UMAs are for debugging purposes. The users are the core developers of preloads. So, we don't need to add a suffix for minor changes (with notice to the team).

An exception is FCP:

- PreloadLog.PageLoad.Clients.PaintTiming.NavigationToFirstContentfulPaint.WithoutPreload
- PreloadLog.PageLoad.Clients.PaintTiming.NavigationToFirstContentfulPaint.WithPrefetch
- PreloadLog.PageLoad.Clients.PaintTiming.NavigationToFirstContentfulPaint.WithPrerender

Naming

- **Navigation**PreloadLog
  - "Navigation" comes from `NavigationHandle`.
- **Serving**PreloadLog
  - We use this prefix tentatively.
- Preload**Activation**Log

    - We'd like to call "serving" as "activation" for both prefetch/prerender.

- HogePreload**Metrics**

Kouhei解釈(どうでもいいかも)
Navigation - NavigationRequestがつくるもの

PrefetchActivation - 1このPrefetchEntryが当該Navigationでつかわれること
PrerenderActivation - 1このPrerenderEntryが当該Navigationでつかわれること

XServing - 0こ以上のXEntryが当該Navigationでつかわれること

Note

- We need to support prerender initial navigation that is committed but not used by activation.
- Add a section to discuss what case is recorded. Compatibility with current PMR metrics.
- Metrics and contents of log.
- Send a mail to notice deprecation of PrefetchServingPageMetrics.
- About prefetch trigger-type-and-eagerness suffix.

CL review

std::unique_ptr<LogObject>

Discussion [doc]

- A. Use `std::unique_ptr<LogStruct>` everywhere.
    - Pros (kenoss@)
        - Simple.
        - No need to judge what we should use.
        - Minimal move cost.
        - The move is guessable. (`std::unique_ptr`)
    - Cons
        - It is unclear whether it is nullable (lazily filled, optional because a condition is not satisfied, might be moved out) or not.
            - It is mitigated by detailed comments and `CHECK`.
- B. Use `LogStruct` if owned and not moved, `std::optional<LogStruct>` if optional and copied from another place, `std::unique_ptr<LogStruct>` if `LogStruct` is large or not copyable.
    - Pros
        - It is unclear whether it is nullable or not.
    - Cons (kenoss@)

    - It is still unclear why nullable (lazily filled, optional because a condition is not satisfied, might be moved out)
    - Move/copy cost depends on size and contents.
    - Move/copy is non trivial nor guessable.
    - Stack/owner class size is not guessable.

So, we will use A for log objects (for `PreloadServingMetrics`).

Trailing period for comments

We are not sure about concrete rules. We'll leave it and fix it later.

PreloadServingMetricsCapsuleImpl

Private chat [doc]

There are pros/cons. Either is OK for the short-term. We'll land it as is so far.

Using `ContentBrowserClient` instead of //content public feature flag

We tried the latter in the other situation. We concluded that the former is better for preloads, because multiple triggers can depend on a new mechanism in //content.

(Known limitation: We can't use a feature flag in two experiments.)

What we tried in enabling prefetch/prerender integration:

- Originally, `kPrerender2FallbackSpecRules` had two meanings: 1. Triggers prefetch ahead of prerender. (`PrerendererImpl`) 2. Enables paths of handling it in prefetch execution stack (e.g. `PrefetchService`, `PrerenderURLLoaderThrottle`).
- Started to implement `kDsePreload2`. We needed 2 for it. Split 1 and 2 and added //content public feature flag `kPrefetchPrerenderIntegration`. [cl]
- Started to implement Bookmarkbar trigger prefetch. We needed `kPrefetchPrerenderIntegration`. Options: 1. Add a variant of `kPrefetchPrerenderIntegration`. 2. Use `ContentBrowserClient`. We did 2. [cl]
- Started to implement NewTabPage trigger prefetch. Ditto. [cl]

So, we use `ContentBrowserClient` for `PreloadServingMetrics`.

TODO left

We will leave some tasks as TODO as the CL chain is huge and rebasing is hard.

Urgent (within a week or so)

- Approved [thread][fix] Refactor PrefetchMatchResolver::FindPrefetch and ForTesting
- Under review [thread][fix] Use PreloadServingMetricsCapsule::IsFeatureEnabled()
- Approved [thread][fix] Mark structs as `final`
- Done [thread][fix] Use methods for common part of UMAs, e.g. `is_prefetch_potential_match`
- Approved [thread][fix] Check necessity of WeakPtrFactory
- Approved [thread][fix] Fix typo: MakeSkeltonPreloadServingMetrics (skeleton)
- Wantfix [thread] Remove MakeSkeltonPreloadServingMetricsArgs.
    - Reason: Without this, I want to add comment `MakeSkeletonPreloadServingMetrics(/*n_prefetch_match_metrics=*/n)`. Efficiency/binary size are not problems as this is in tests. I prefer using struct as “with name call” is enforced.
- Wantfix [thread][fix] s/prerender_host/prerender_host_for_metrics/
    - As this is an argument.
- Not started #ideal-form Discuss ideal form of `PreloadServingMetrics/PreloadServingMetricsHolder/PreloadServingMetricsCapsule`
    - [thread] Move `PreloadServingMetricsCapsule` into the internal of content/public/browser/preload_serving_metrics_capsule.cc if it compiles.
    - [thread] Describe the policy of logs.
    - [thread] Rethink structure of PreloadServingMetrics and holder/capsule

Non-urgent

- Not started [thread] Mark `std::unique_ptr` const if available.
    - Blocked by #ideal-form
- Not started [thread] Centralize Take() call paths.
    - Blocked by #ideal-form
- Approved [thread][fix] Consider to remove no-op test fixture