Capturing task type metadata

altimin@chromium.org, hajimehoshi@chromium.org
5 September 2017
crbug.com/762453

At the moment most of tasks originating from Blink are posted through TaskRunnerHelper API and annotated with task type metadata. Unfortunately, this information is lost when an appropriate task queue is selected for a task type. It is proposed to rework WebTaskRunner to contain a reference to a task type and forward it to scheduler's TaskQueue where it will be stored along other metadata.

WebTaskRunner

WebTaskRunner is a Blink counterpart to SingleThreadTaskRunner interface with the only non-test implementation (WebTaskRunnerImpl). Currently WebTaskRunners are created by the scheduler, with each non-internal TaskQueue having exactly one WebTaskRunner which is exposed to the rest of Blink. It is proposed to change this relationship to one-to-many, with TaskRunnerHelper creating new WebTaskRunners, every TaskRunnerHelper containing TaskType and each call to WebTaskRunner::PostTask being forwarded to TaskQueue::PostTaskWithType and the task type should be stored in TaskQueue::Task.

Milestones

Move TaskType to public/platform

Status: done, CL.
Move TaskType definition from TaskRunnerHelper.h to public/platform/TaskType.h to allow both content/ layer to specify task type and platform/ layer (including scheduler to use it).

Move TaskRunnerHelper inside scheduler

Status: Done (CL, CL)
Instead of exposing individual task runners (throttleable/deferrable/etc) WebFrameScheduler should take TaskType as an argument and return an appropriate task runner (WebFrameScheduler::GetTaskRunner(TaskType type)).

Given that CL 644325 guaranteed that the second argument to TaskRunnerHelper::Get isn't null, similar methods can be implemented in other relevant objects (LocalFrame, Document, ExecutionContext, etc). The biggest benefit of this approach is that the same approach can work with content/ layer with GetTaskRunner method being added to RenderFrame and similar objects.

Create a new WebTaskRunner for every WebFrameScheduler::GetTaskRunner call and save task type

Status: Done (CL, CL, CL)

Now instead of returning a saved web task runner web frame scheduler should create a new one for each task type. WebTaskRunnerImpl::Create should take a task type as an argument and store it inside WebTaskRunnerImpl. There can be problems with the WebTaskRunner being refcounted and some callers not saving references and using raw pointers and relying on WebFrameScheduler holding references to WebTaskRunners for them. This can lead to unexpected crashes, but they should be easy to fix. List of suspicious places (WebTaskRunner*).

Introduce TaskQueue::PostTaskWithMetadata and use it in WebTaskRunner

Status: done: CL

Change all major methods in WebTaskRunner to be virtual and implement them in WebTaskRunnerImpl instead of relying on forwarding calls to single thread task runner obtained from WebTaskRunner::ToSingleThreadTaskRunner.

This will allow us to implement PostDelayedTask inside WebTaskRunnerImpl, which means it can use TaskQueue::PostTaskWithMetadata.

Plumb metadata from TaskQueue::PostTaskWithMetadata to TaskQueue::Task

Status: Done, TaskQueue::PostedTask is introduced (1, 2) and used internally in TaskQueue/TaskQueueImpl.

Due to having to have a sequence number when creating a TaskQueueImpl::Task, it is created somewhat deep in the scheduler and the task (base::OnceClosure) is plumbed. It is proposed to introduce TaskQueue::PendingTask which will contain both callback (base::OnceClosure) and metadata (task type).

Plumb TaskType to TaskQueue::Task via TaskQueue::PostedTask

Status: Done (CL, CL)

Due to having to have a sequence number when creating a TaskQueueImpl::Task, it is created somewhat deep in the scheduler and the task (base::OnceClosure) is plumbed. It is proposed to introduce TaskQueue::PendingTask which will contain both callback (base::OnceClosure) and metadata (task type).

(Stretch) WebTaskRunner::ToSingleThreadTaskRunner to return a wrapper

Status: Obsolete in favour of merging WebTaskRunner into SingleThreadTaskRunner.

Now WebTaskRunner::ToSingleThreadTaskRunner underlying task queue, which means that task type information will be lost when the returned task queue is used. It is proposed to return a wrapper — custom implementation of SingleThreadTaskRunner which will store metadata (including task type) inside and forward PostDelayedTask calls to PostTaskWithMetadata.

(Stretch) Remove TaskRunnerHelper

Status: Done (crbug.com/777775)

Now TaskRunnerHelper's implementation is just to redirect the task runner getter (or return the default task runner as fallback), let's move the implementation to the corresponding classes and remove TaskRunnerHelper.