Further per-frame scheduler work

altimin@chromium.org, hajimehoshi@chromium.org

crbug/780785, crbug/786332

19 October 2017

This document describes some initiatives around per-frame scheduler effort which are different from converting ThreadTaskRunnerHandle::Get callsites to GetTaskRunner(task_type). The general goal is to avoid the default task queue ThreadTaskRunnerHandle::Get() and replace them with GetTaskRunner(task_type) so that we can get more flexible task scheduling to improve responding.

Introduce IPC task queue

At the moment ipc tasks are believed to be the biggest source of tasks for default_tq. It is proposed to introduce a special IPC task queue to reduce the number of tasks in default queues and determine the actual real-world percentage of IPC tasks. This will enable further work for prioritising different IPCs differently and moving some IPC to per-frame task queues.

There are two places where there tasks are posted from:

- ipc_channel_proxy.cc:OnMessageReceivedNoFilter (~25% of all default tasks)
  Status: Done (CL)
  Listener task runner is captured via ThreadTaskRunnerHandle::Get when a ChannelProxyContext is created. We need to add an option to the constructor to specify a listener task runner and plumb it from content::ChildThreadImpl::Init via IPC::SyncChannel::Create.
- ipc_mojo_bootstrap.cc:Accept (~20% of all default tasks)
  Status: working (CL, CL)
  Proxy task runner is captured via ThreadTaskRunnerHandle::Get in a ChannelAssociatedGroupController constructor. An option to specify task runner is needed, and the relevant task runner should be plumbed from ChildThreadImpl::ConnectChannel via ChannelProxy::Init, ChannelProxy::Context::CreateChannel, PlatformChannelFactory::BuildChannel, ChannelMojo::Create and ChannelMojo constructor

Expose ChildSheduler on ChildThreadImpl

Status: Done (CL, CL)
IPC::SyncChannel is created at ChildThreadImpl::Init, where a new (listener) task queue will be passed. Now ChildThreadImpl doesn't refer a ChildScheduler or RendererScheduler. The problem is that we can't get a task queue for IPC from ChildThreadImpl.

Fix ChildThreadImpl to ref the scheduler and use the task runner for IPC.

Use TaskType for scheduler metrics

Status: Done (CL)

Record RendererScheduler.TaskDurationPerTaskType similar to already-existing RendererScheduler.TaskDurationPerQueueType2, The correct place to do this is RendererSchedulerImpl::OnTaskCompleted (we also need to plumb Task reference to OnTaskCompleted, which is easy given that it RendererSchedulerImpl::OnTaskCopleted is the only non-test caller).

Relevant patch introducing per-frame type metrics.

Using this information, replace RendererScheduler.TaskDurationPerQueueType.* with RendererScheduler.TaskDurationPerWorkType.*, surfacing most expensive task types and grouping the long tail of task types per TaskQueue. Exact members of WorkType are to be determined based on the previous results. Most likely it will be per-renderer task queues plus kJavascriptTimers and kNetworking task types (and maybe some more) split by frame status plus per-frame task queues.

Breaking down IPC tasks

Prioritising TTRH::Get callsite conversion

Status: hajimehoshi is working on this (CL)

(This is just a local experiment and we don't have a plan to ship this.) In order to get the maximal benefit from converting individual ThreadTaskRunnerHandle::Get conversion it is proposed to collect information about amount of work originating from each callsite. It is proposed to locally modify Chrome to collect and output this information and run modified version manually to collect some statistics.

Results of the previous iteration: spreadsheet.

Suggested steps to get this data:

- Modify ThreadTaskRunnerHandle::Get to take Location as a parameter (example).
- Autoreplace ThreadTaskRunnerHandle::Get with ThreadTaskRunnerHandle::Get(FROM_HERE).
    - Fix crashes (and upstream) from people who use raw pointers to TaskRunners.
- Add base::Optional<Location> task_runner_obtained_from to TaskQueue::PostedTask and TaskQueue::Task (similar to blink::TaskType).
- Record metadata for each task including posted_from and task_runner_obtained_from
    - First option: record it to an external file (prototype). Requires --no-sandbox flag to write contents to the disk.

    - Second option: tracing. Unfortunately, at the moment there is no option to trace for the whole duration of running the browser (most similar is --no-
    - My intuition would be to play with the first option and explore the second one if we decide to land it in some form.
- Collect data
    - First great source of data is just running browser locally and open and navigate a number of sites.
    - Second option is running layout tests and collect the data. It will provide us with great repeatability of the data.

Note: the long-term plan is to make ThreadTaskRunnerHandle::Get to post tasks to the current task queue. Now we can't do that because in many places where an appropriate task runner is not specified a wrong task queue will be inherited and this will propagate across multiple TTRH::Get calls (we call this problem "colouring"). So the plan is to properly annotate appropriate calls (places where ThreadTaskRunnerHandle::Get result is stored in member variable) first and change it later.

2017-11-30
Found most calls are at SequencedTaskRunnerHandler::Get().
Result
IPC seems the heaviest, but as we introduced IPC task runner, that's strange.

2017-11-30 (second attempt)
Found that IPC task runner was not used when needed. CL to fix
Result
GetTaskRunnerToUseFromUserProvidedTaskRunner and render widget seem the heaviest.

2017-12-01
Did the same experiment (with posted locations) with all the layout tests
Result
The result seems the same as above. GetTaskRunnerToUseFromUserProvidedTaskRunner is the heaviest.

--

What we want to do:

- We want RendererScheduler work not only on the main thread but also on other threads.
- We want ChildThreadImpl to own a ChildScheduler