Simplifying Scheduler Interfaces

May 25, 2018

Yuta Kitamura <yutak@chromium.org>

This is a supplementary proposal to amend part of Scheduler Architecture 2.0 to address the issue raised while I'm working on the project.

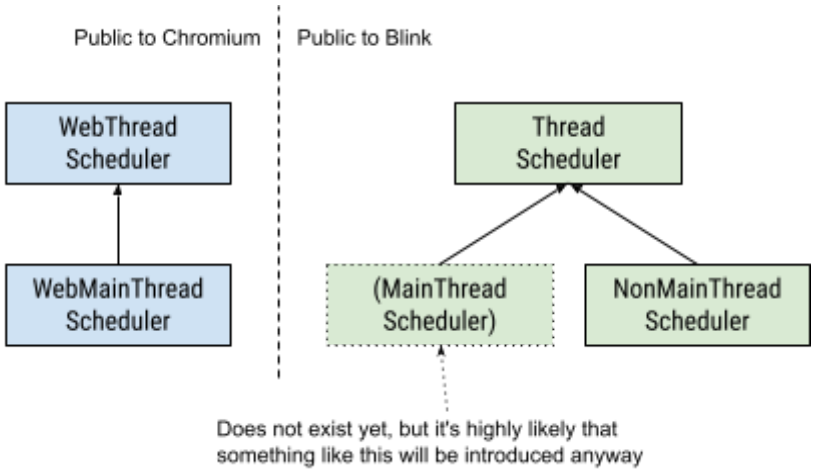
The Issue

Blink's scheduler code is separated into three parts to provide precise access control over different sets of audience:

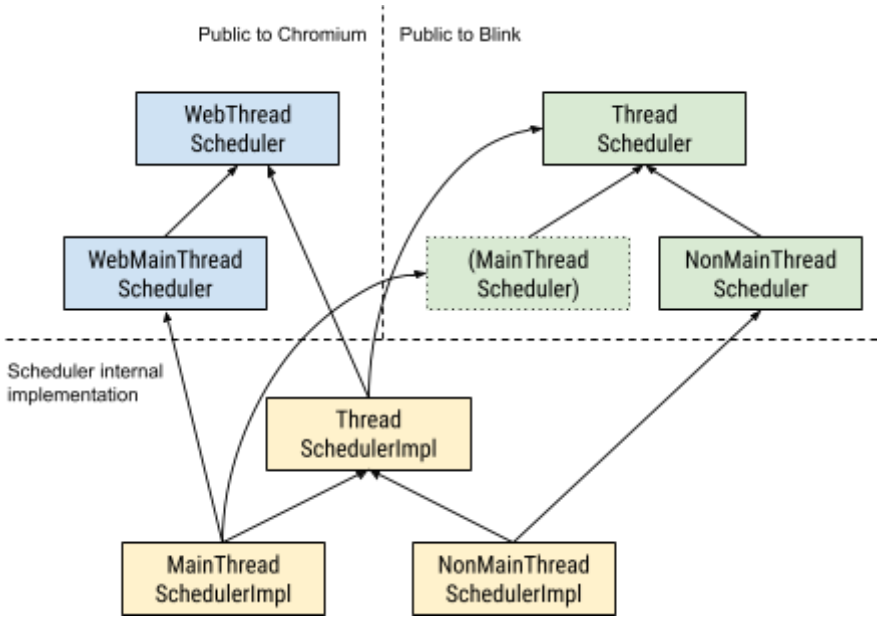
- (A) Public to Chromium (//content mostly) -- at blink/public/platform/scheduler/.
- (B) Public to Blink -- at blink/renderer/platform/scheduler/public/.
- (C) Scheduler-internal implementation -- at blink/renderer/platform/scheduler/<others>.

(A) and (B) are independent to each other, and (C) implements both of them.

Currently, the interface classes in (A) and (B) have the following class hierarchy:



The core question here is: how do we implement those interfaces? A naïve approach would give something like this:



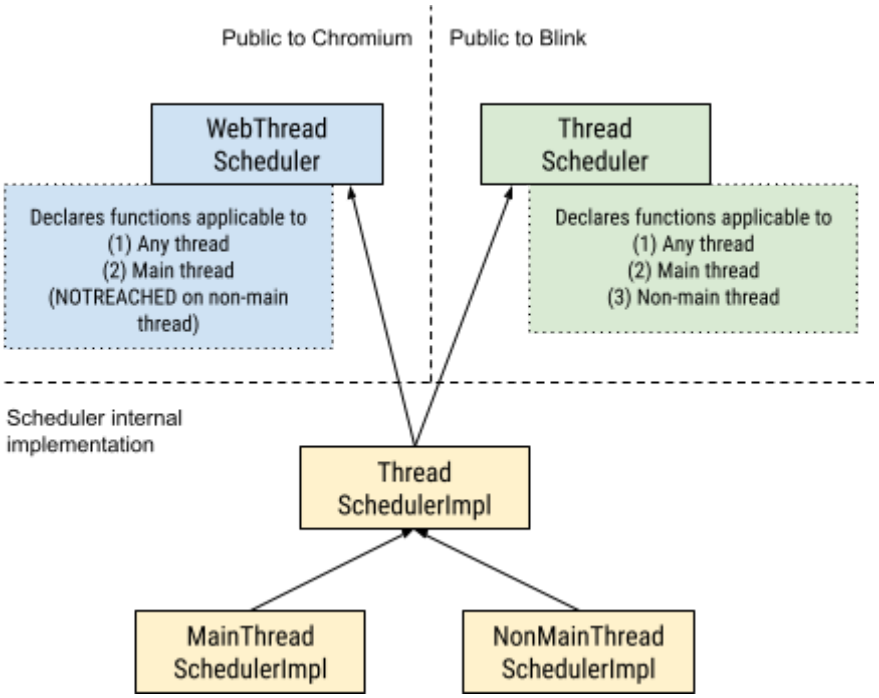
...but you know, this is a classic example of "multiple inheritance hell" -- you should have learned why a diamond inheritance is bad, right?

> For fairness' sake, let me make a defense on this approach: it totally works if we *correctly* make some inheritances "virtual". There won't be any functionality issues. My concern is the mental load of understanding the overall structure and the complexity on the source code.

altimin@ proposed allowing inheritance between (A) and (B). This cuts down a few inheritances but still some of the diamonds remain.

Proposed Solution

My proposal here is to merge interfaces to Chromium to one class, and do the same on the interfaces to Blink, which effectively removes the inheritances within (A) and (B). This will look like this:



All the interface functions now go to either WebThreadScheduler or ThreadScheduler. You can see the inheritance relationship is much simpler. The virtual functions that do not apply to the current thread's scheduler just need to have NOTREACHED() in their definitions.

The reasoning behind this proposal is something like this:

- The thread scheduler object is a thread-local singleton -- you can expect there always exists one thread scheduler on each Blink thread, and its actual instance depends on the kind of the Blink thread (main thread, worker thread and compositor thread).
- That is, you can tell the actual instance of a specific (Web)ThreadScheduler pointer deterministically, if you know the kind of the thread the pointer is on. Generally, you just want to call the scheduler of the current thread, and in most cases you are aware of the type of the current thread. On the context where the target thread is not known, you'd generally call a function of the base class ((Web)ThreadScheduler in the original design), and don't have to detect the type of the thread and do a cast to a child class.
- That means you really don't rely on the dynamic nature of the class hierarchy between classes within (A) or (B). This lets you shrink the class hierarchy and just provide one interface class having all the virtual functions that may used by each audience of the interface (Chromium or Blink).

As far as I can tell, this proposal has no major weakness. (Do you think of any?)

Document History

2018-05-25	Initial proposal posted to scheduler-dev.