Redefine concepts of Worker(Backing)Thread

Author: nhiroki@chromium.org
Last update: Jul 6, 2017
Status: Request for comments / making POC
Issue: crbug.com/710364
Visibility: PUBLIC

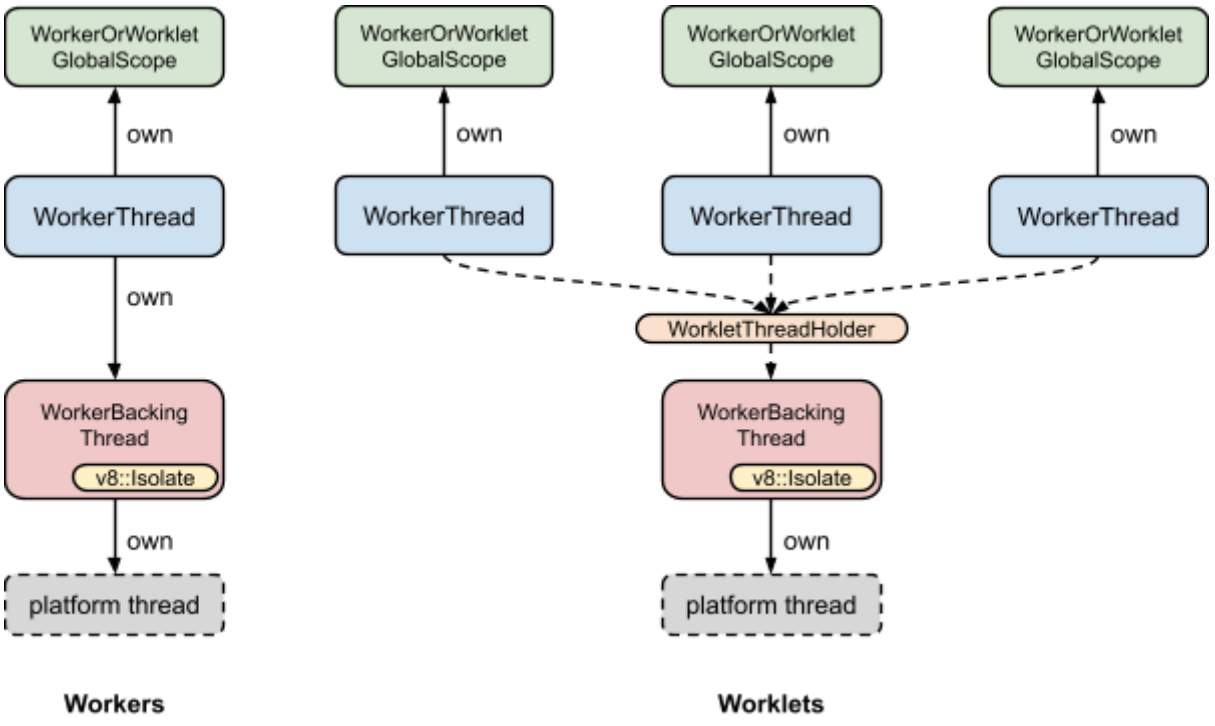
TL;DR This document proposes to…

- **redefine WorkerThread as a per-global-scope object** and rename it to something more appropriate name like **GlobalScopeController**, and
- **redefine WorkerBackingThread as a per-platform-thread object** and move per-platform-thread things from WorkerThread to WorkerBackingThread.

What's WorkerThread?

WorkerThread is a key class in the worker infrastructure. Interestingly, **WorkerThread is actually not a thread**, but a kind of a client of a platform thread and a holder/controller of WorkerOrWorkletGlobalScope.

Basically, there is one-to-one relationship between WorkerThread and WorkerBackingThread. However, for worklets, one WorkerBackingThread can be shared by multiple WorkerThreads. This means multiple execution contexts (WorkerOrWorkletGlobalScopes) can run on the same platform thread (see the following figure).



Problems

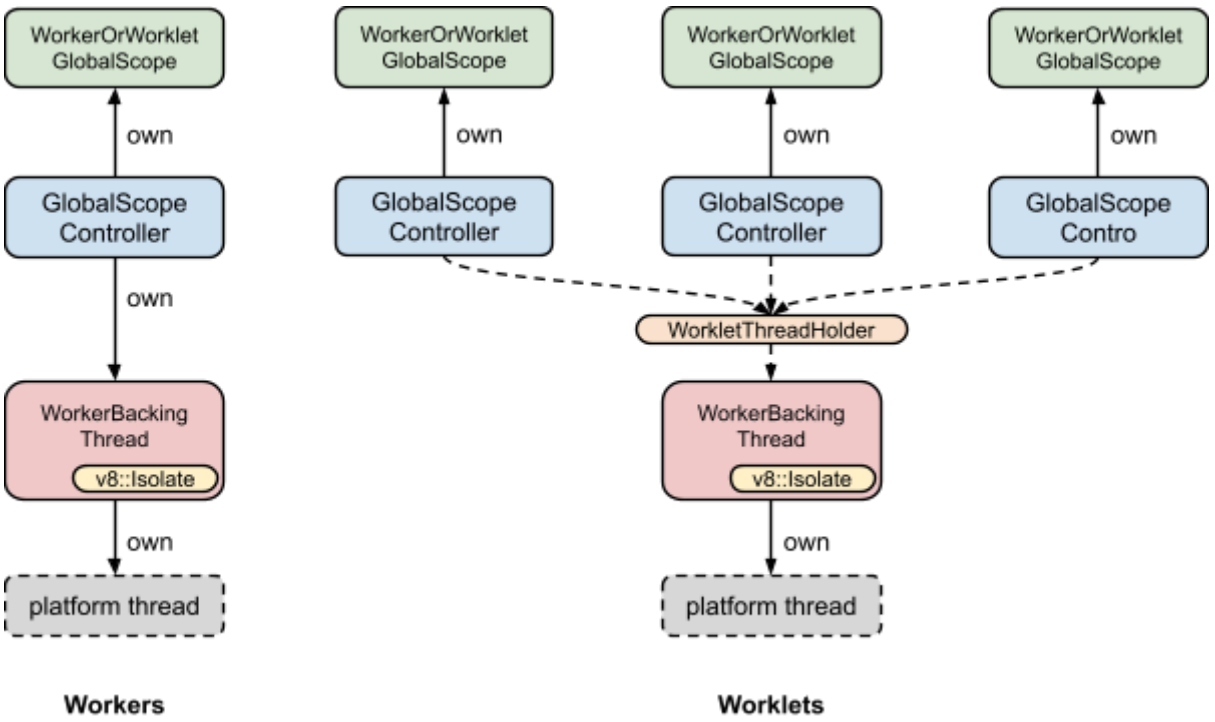
WorkerThread becomes a kind of the God object and its concept is getting unclear. It has a bit too much functionalities as follows and some of them would be misplaced or broken because worklets were introduced.

- Starts and closes a global scope.
- Derived classes of WorkerThread provide functions that depend on each worker/worklet type (e.g., creating a global scope, initializing a shared thread).
- ~~Provides postTask interfaces to the worker thread.~~
    - This was already fixed by the per-global-scope scheduller (crbug/694914)
- Notifies WorkerThread shutdown via WorkerThreadLifecycleObserver
    - The concept of WorkerThreadLifecycleObserver would be unclear. The name implies that this notifies thread shutdown. This is correct on workers, but may not be correct on worklets because it's notified every global-scope close.
- Manages inspector tasks
- Counts # of worker threads (see WorkerThread::WorkerThreadCount)
    - This could be broken for workelts because it counts not # of WorkerBackingThread but # of WorkerThreads.

Also, WorkerThread is a confusing name because it's not a real thread.

Proposals

Our proposals are to redefine WorkerThread class as a per-global-scope object to mediate between a global scope and an underlying platform thread, and to rename it to GlobalScopeController.



Specifically, we will…

- rename per-global-scope classes as follows
    - WorkerThread to GlobalScopeController
    - WorkerThreadLifecycleObserver to GlobalScopeLifecycleObserver
    - WorkerThreadStartupData to GlobalScopeStartupData
- **[DONE in 2017Q2]** factor out postTask interfaces into the per-global-scope scheduler (CL)
- make GlobalScopeController thread-agnostic
    - Current implementation and comments were written based on an assumption that WorkerThread/GlobalScope run on a worker thread, but it could be on a compostor thread etc. We should rename variable and method names, and update comments.
- identify non-per-global-scope things and move them into WorkerBackingThread.
- move inspector tasks from GlobalScopeController to WorkerBackingThread
    - nhiroki@ is still not really sure if this work well and need more investigation (crbug/662870)
- write markdown documents about an overview of the class structure, concepts of GlobalScope / GlobalScopeController / WorkerBackingThread, and links to design documents like scheduler/README.md.