Better Threading Affinity in Blink

Assign Thread Affinity to Task-related classes

hiroshige@chromium.org

STATUS: Ongoing, anyone can comment. Last Update: Feb 26, 2016.

Update (Feb 26):

Some CLs are landed. Updating working memo in Step 1.

Update (Feb 19):

Added FROM_HERE for clarification. Made public.

Added draft CLs:
1. https://codereview.chromium.org/1713143002 (landed)
2. https://codereview.chromium.org/1549143002

Abstract (from point of view of bind/Bind merging)

To prepare merging WTF::bind() and base::Bind(), I'm going to assign thread affinity to WTF::Function. However, WTF::Function is wrapped by WebTaskRunner::Task and many classes before reaching Chromium, so just modifying WTF::Function is not sufficient.

I propose to:

- Use WTF::Function (rather than WebTaskRunner::Task) in Blink (except for inside WebTaskRunner) directly, rather than using/wrapping by WebTaskRunner::Task and blink::Task, and
- Split WTF::Function into WTF::Function (for same thread tasks) and WTF::CrossThreadFunction.

For example,

postTask(BLINK_FROM_HERE, new Task(WTF::bind(...)))

will be

postTask(BLINK_FROM_HERE, WTF::bind(...))

I expect changes will be completed in a rock step, but I created this doc because this changes the style of Blink's postTask() -- reverting postTask + new Task() into postTask() + WTF::bind().

Problems

Clean world of cross-thread task posting would be:

- Every object either:
  - belongs to a thread, or
  - is detached from any thread.
- Every object is taken a deep copy (or explicitly detached from threads) when passed across threads.
  - We create data X on Thread A. X belongs to Thread A.

    - We take a deep copy of X -- let this deep copy Y. Y is detached from any thread.
    - We post a task that owns Y. the task and Y is passed to Thread B.
    - We attach Y to Thread B, and use and destruct Y on Thread B.

Good. However, current problems are:

- **Problem 1**: Blink uses same classes/functions to post tasks to the current thread (*SameThread*) and to a different thread (*CrossThread*).
- **Problem 2**: Blink passes many objects without deep copying.
- **Problem 3**: Deep copy functions are quite error-prone.
- **Problem 4**: There are no explicit detaching from thread.

This proposal covers Problems 1 and a part of Problem 2.
(I'm also working for Problems 3 and 4 but they will be covered by another proposal.)

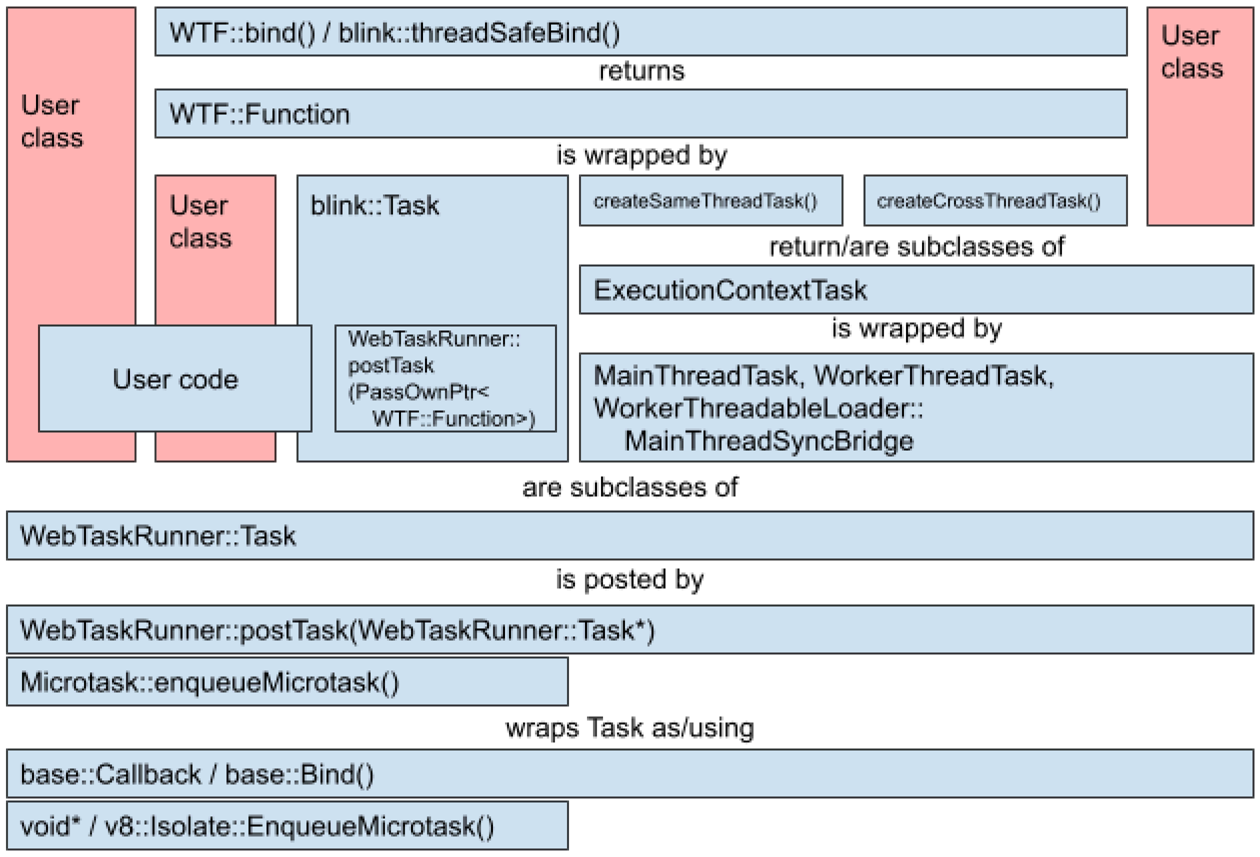
Goals

Here, I propose a refactoring plan that achieves:

- **Goal 1**: Use separate classes for SameThread/CrossThread tasks.
    - Thread affinity is clearer, and checked more statically.
    - Preparation for merging WTF::bind() into base::Bind().
    - Tracking bug: https://bugs.chromium.org/p/chromium/issues/detail?id=**588224**
    - Draft CLs:
        - 1. https://codereview.chromium.org/1713143002
        - 2. https://codereview.chromium.org/1549143002
- **Goal 2**: Reduce paths that bypasses CrossThreadCopier.
    - Tracking bug: https://crbug.com/**478194**
    - Draft CLs: many that will be tracked by Issue **478194**.

Current Implementation: How tasks are created and posted

(data are flowing from top to bottom)


WTF::bind() / blink::threadSafeBind()
returns
WTF::Function
is wrapped by
User class
User class
User class
User class
blink::Task
createSameThreadTask()
createCrossThreadTask()
return/are subclasses of
ExecutionContextTask
is wrapped by
User code
WebTaskRunner:: postTask (PassOwnPtr< WTF::Function>)
MainThreadTask, WorkerThreadTask, WorkerThreadableLoader:: MainThreadSyncBridge
are subclasses of
WebTaskRunner::Task
is posted by
WebTaskRunner::postTask(WebTaskRunner::Task*)
Microtask::enqueueMicrotask()
wraps Task as/using
base::Callback / base::Bind()
void* / v8::Isolate::EnqueueMicrotask()

Problems:

Problem 1: ThreadAffinity (Same/CrossThread) are not explicit.

- WTF::Function, WebTaskRunner::Task, ExecutionContextTask has unknown thread affinity -- whether SameThread or CrossThread.

Problem 2: There are several paths that bypasses CrossThreadCopier:

- **Problem 2a**: Subclasses of WebTaskRunner::Task or ExecutionContextTask can contain data members, which are not checked by CrossThreadCopier (red boxes above).
- **Problem 2b** (Not handled in this proposal):
  If we pass pointers by AllowCrossThreadAccess(), PassOwnPtr, or raw pointers to ThreadSafeRefCounted/GarbageCollected, the data members of the object pointed by those pointers are not checked by CrossThreadCopier.

Proposal: Step 1 (2016Q1)

Solution 1: Use separate types for same-thread/cross-thread WTF::Functions, and pass them to WebTaskRunner without wrapping by WebTaskRunner::Task.

Before	After		Progress
	SameThread	CrossThread	
Types			
WTF::Function<T>	WTF::Function<T, SameThreadAffinity> ThreadAffinity is the last argument to make it default to SameThreadAffinity.	WTF::Function<T, CrossThreadAffinity>	[2]
WTF::Closure	WTF::Closure	WTF::CrossThreadClosure	[2]
WTF::bind()	WTF::bind()	blink::threadSafeBind()	[2]
blink::Task	Use WTF::bind() directly. blink::Task -> make this an internal class	Use threadSafeBind() directly. blink::CrossThreadTask -> make this an internal class	Landed. [1]
Other WebTaskRunner::Task subclasses in Blink	To be replaced with WTF::bind().	To be replaced with threadSafeBind().	Landed. [10] [11] [12] [13] [14] [15] [16][17]
Other WebTaskRunner::Task subclasses in chromium	Untouched, remain unannotated.		N/A
WebThread::IdleTask subclasses in Blink	To be replaced with WTF::bind().	To be replaced with WTF::threadSafeBind().	Not started. IdleFence Task only?
WebThread::IdleTask subclasses in Chromium	Untouched, remain unannotated.		N/A
CallClosureTaskBase<T >	CallClosureTaskBase<T, SameThreadAffinity>	CallClosureTaskBase<T, CrossThreadAffinity>	[2]
Other ExecutionContextTask	To be replaced with createSameThreadTask	To be replaced with createCrossThreadTask().	Step 2.

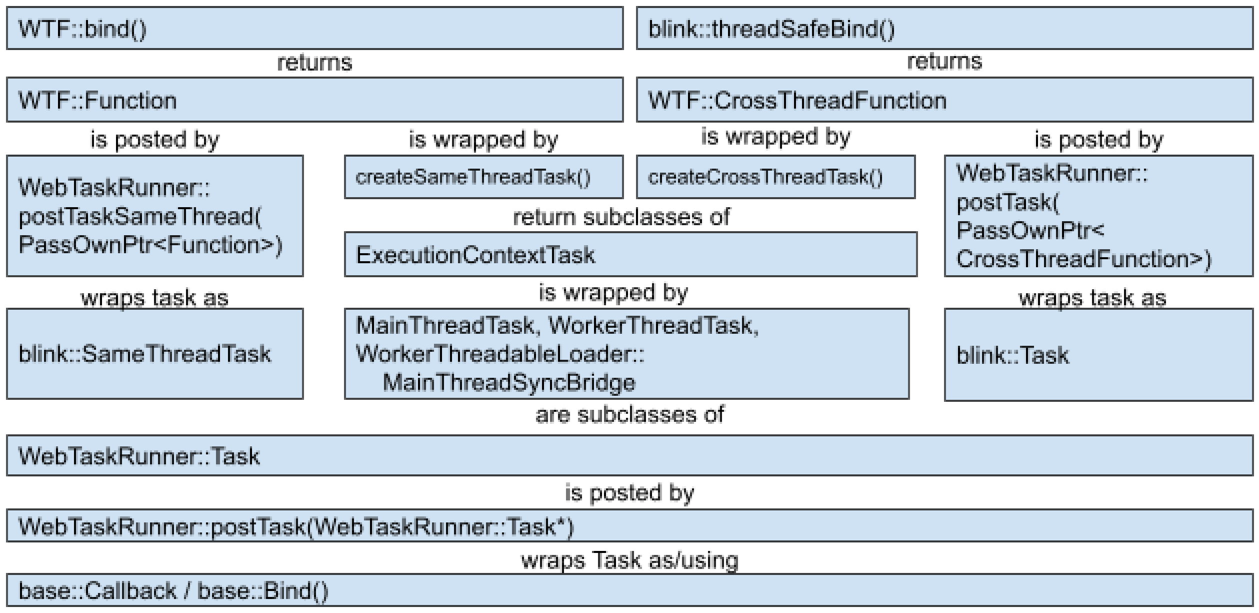
subclasses	().		
postTask()			
WebTaskRunner	postTask() with WebTaskRunner::Task remains for chromium.		[2]
WebThreadSupporting GC	postTask() with Closure/CrossThreadClosure only.		[2]
WebScheduler	postTask() with WebTaskRunner::Task remains, but most of them can be protected, because it is not called from Blink outside WebScheduler.		Not started.

memo: add assert and comment, make crash distinguishable

- WebTaskRunnerImpl wraps them by WebTaskRunner::Task just before crossing Chromium/Blink boundary.

Solution 2: Reduce subclasses of tasks that can be used for bypassing CrossThreadCopier.

- Remove subclasses of WebTaskRunner::Task and use bind()/threadSafeBind().
- Remove subclasses of ExecutionContextTask and use createSame/CrossThreadTask().



(Perhaps it would be better to rename blink::Task to something like blink::CrossThreadTask)

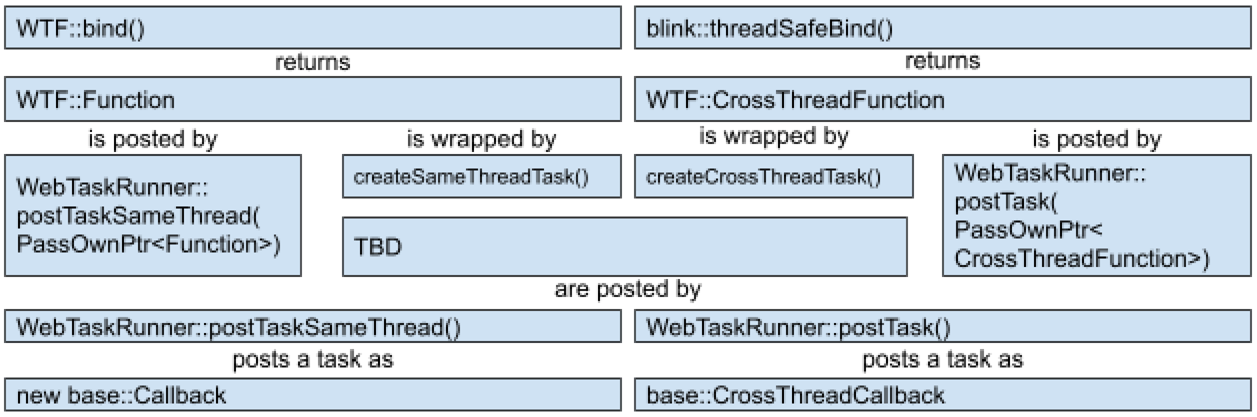
Proposal: Step 2 (2016Q1)

- Use separate classes for same-thread/cross-thread ExecutionContextTask (TBD).

Proposal: Step 3 (2016Q2)

- Use new Callbacks in Chromium side.
    - WTF::Function/CrossThreadFunction will be (thin wrapper of) new base::Callback/CrossThreadCallback, directly passed to Chromium.

- (WebTaskRunner::Task or something similar might be needed for crossing Chromium/Blink boundaries though, but it would be just a thin wrapper)


WTF::bind()
returns
WTF::Function
is posted by
WebTaskRunner:: postTaskSameThread(PassOwnPtr<Function>)
is wrapped by
createSameThreadTask()
blink::threadSafeBind()
returns
WTF::CrossThreadFunction
is wrapped by
createCrossThreadTask()
is posted by
WebTaskRunner:: postTask(PassOwnPtr< CrossThreadFunction>)
TBD
are posted by
WebTaskRunner::postTaskSameThread()
posts a task as
new base::Callback
WebTaskRunner::postTask()
posts a task as
base::CrossThreadCallback