Composited Worklet Animations – Updating Local Time on Main Thread

The Problem

The current approach for updating local time on the main thread is as follows:

- Every frame the current time is updated on the main thread, “peek” request is made to the worklet.
- The worklet returns the last local time produced based on the current time supplied by the compositor thread.
- The local time is assigned to the keyframe effect of the main thread.

The problems with this approach are:

- Since ticking is not synchronized between the main and compositor threads, the local time returned by the worklet might not have been produced for the latest main thread current time. As a result, the local time on the main thread does not correspond to the latest current time.
- Scroll-linked animations may produce different dynamics of ticking between the main and compositor threads. An extreme case of this is when a user scrolls rapidly up and down to the same position, but the change is only detectable by the compositor and not the main thread. As a result, the local time is never updated on the main thread.

Solution

Instead of polling local times on the main thread, use a push model, where local times are pushed from the compositor to main thread for each BeginMainFrame. Similar approach is used to synchronize scroll deltas from the compositor to main thread.

Note, that another mechanism is implemented to sync animation events, which is posting a task to main thread with the events generated on the compositor thread. This approach is too expensive for local time updates because in most cases the sync is required for each frame.

Objective

The goals of the proposed design are:

1. Fix synchronizing worklet animation local times from the compositor to main thread.
2. Implementing unified animation updates path that will be used to synchronize animation events and worklet animation local times from the compositor to main thread.

Design

New animation event of type WA_LOCAL_EFFECT_UPDATE will be introduced to encapsulate local time update information.

Generating and Collecting Events on the Compositor Thread

New member of cc:AnimationHost, animation_events_, will keep a collection of animation events to be pushed to the main thread as part of next BeginMainFrame.

Existing Animation Events

The logic of generating existing animation events will not change with the exception that the events will be added to animation_events_ collection and no task posted to the main thread.

Local Time Updates Events

cc:WorkletAnimation will keep the most recent local time to be synced to the main thread.

Sending Animation Events from Compositor to Main Thread

Animation events will be included into the begin main frame arguments that are provided from the compositor to main thread as part of BeginMainFrame call:
cc: BeginMainFrameAndCommitState::mutator_events.

mutator_events will be populated from cc:AnimationHost::animation_events_ and worklet animations local times. Upon populating the mutator_events, the sources of these events will be emptied to prevent from sending duplications.

Processing Events on the Main Thread

Existing mechanism of AnimationHost::SetAnimationEvents will be expanded to propagate the events to their listeners. In case of WA_LOCAL_EFFECT_UPDATE event, the listener will be blink:WorkletAnimation. The callback will be similar to WorkletAnimation::SetOutputState, where the local_times_ is updated with the new value. The local effect will be updated during normal Update flow.

Implementation Details

The implementation can be achieved in 2 stages:

1. Introduce a new AnimationEvent of TIME_UPDATE type and include this event as part of the existing sync mechanism (e.g. post task to main thread).
2. Introduce Begin Main Frame synchronization path and move all animation events to that path.

Prototype:

https://chromium-review.googlesource.com/c/chromium/src/+/1620933

Alternative Solutions

1. Enhance existing peek solution to track local time updates on main and worklet animation threads. However this makes the current solution complicated and defeats the purpose of selecting “peek optimization” solution at the first place for its simplicity.
2. Keep separate mechanisms for syncing animation events, which is having posting tasks for the existing animation events and local time updates included into the begin main frame arguments. This is the safest solution since it doesn’t introduce risk of breaking the existing functionality, however in the long term it requires maintaining multiple solutions for similar asks.
3. Reuse the existing posting updates mechanism for local times updates. This solution is not optimal since it may send multiple events per single main frame and introduces unnecessary overhead of posting a task.