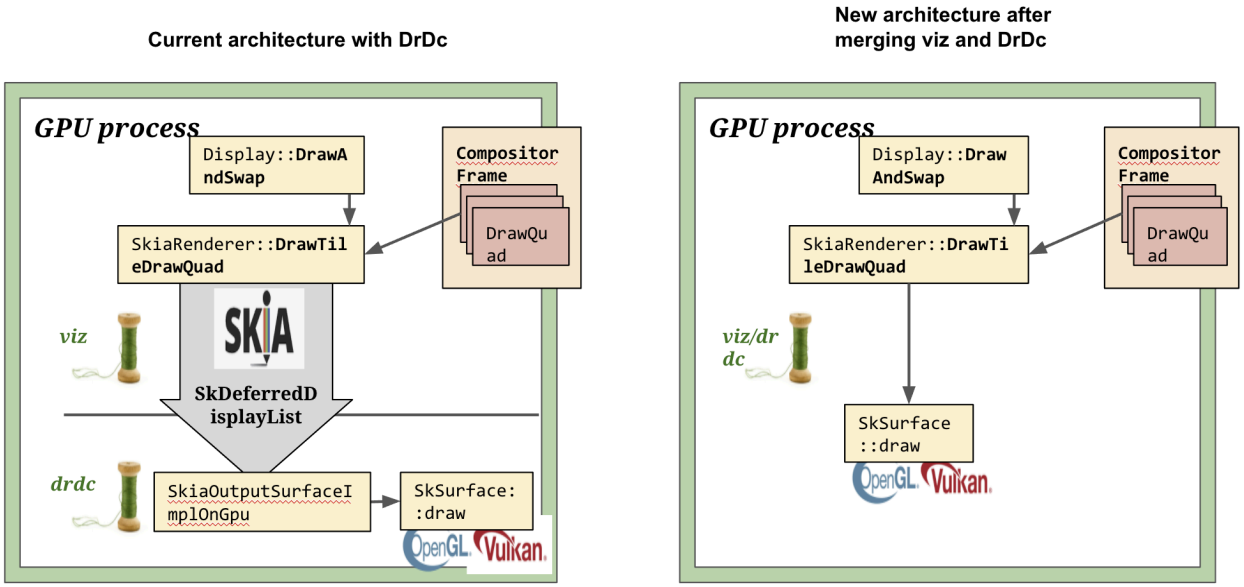
Viz thread as Dr-Dc thread

Author - vikassoni@
Last modified - 02/23/2022
Tracking bug - 1300366

Objective

This document discusses the pros and cons of using Viz display compositor thread as the DrDc gpu thread to directly issue draw commands to skia instead of recording and replaying them in the form of skia DDLs.

Background

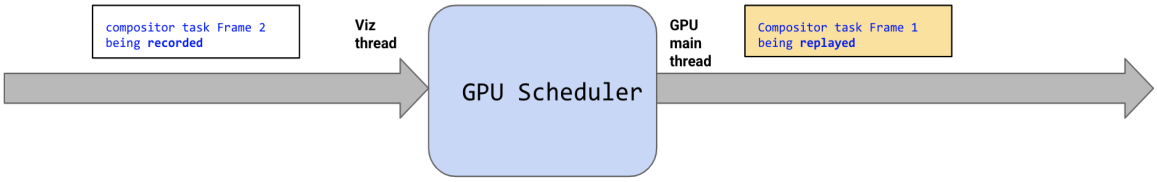


Currently in chrome, display compositor uses viz thread to first record gpu tasks/draw commands in the form of skia DDLs. These DDLs are then replayed back on the gpu thread (gpu main or DrDc thread if it's enabled). There could be added overhead in recording and replaying the draw commands instead of issuing them directly to the skia on the gpu thread. At the same time it has some advantages which currently outweighs the overhead.

In this document we talk about various pros and cons of using DDLs and whether merging viz thread and DrDc thread in future to issue draw commands to skia directly makes more sense for improving performance.

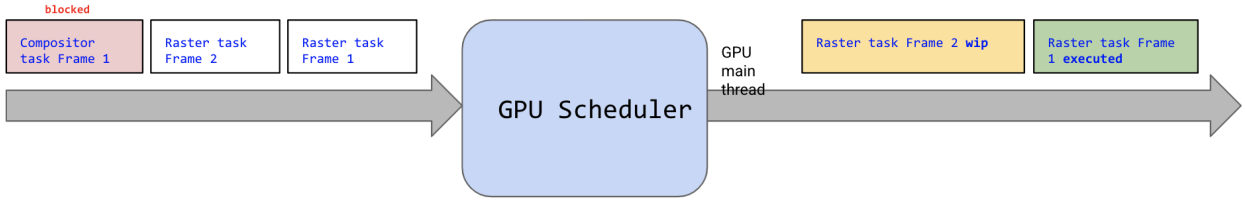
Pros and Cons of merging viz and DrDc thread

Here we look into exploring the option of merging both viz thread and DrDc thread by simply using viz thread itself as DrDc thread which will have the compositor context always current. This way draw commands will be issued directly from the viz thread and there is no need to record and replay DDLs.



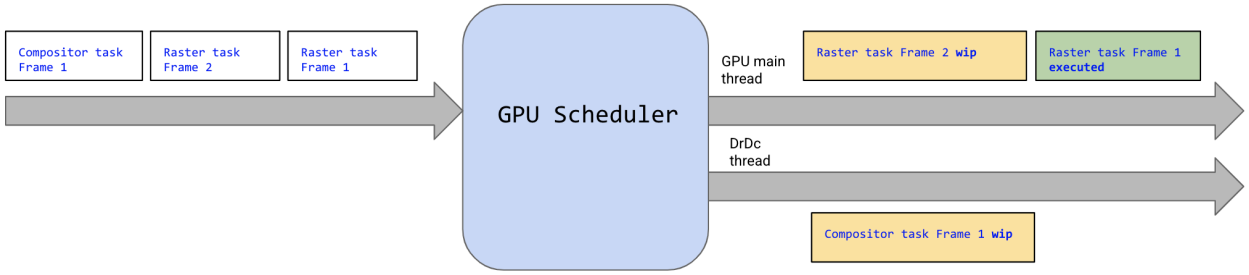
Advantage of separate viz thread to record/replay skia DDLs

One of the primary reasons for using separate viz thread and gpu thread is increased parallelism as shown above which provides the ability to record a future tile on viz thread while continuing to replay a current/past tile on gpu thread. Today the amount of overlap between viz thread and gpu thread is low, i.e, the number of times viz and gpu thread are used concurrently to record and replay different tiles are low and hence the benefit of added parallelism has low advantage here. This is because most of the work is done in skia when the DDLs are actually replayed and record is much light weight comparatively. Both gpu command buffer recording and the submission to gpu driver happens while replay.



Blocked compositor work due to ongoing raster work

Another reason to use a separate viz thread is that the ability to record gpu work on viz thread also allows to unblock some of the compositor work in the cases the compositor is blocked by the raster task happening on gpu thread for example as shown in above diagram.



Compositor unblocked with DrDc even if raster work is ongoing

With DrDc, as shown above, display compositor now has a dedicated gpu thread which means the compositor will not be blocked by any other raster/webgl/canvas etc work now and hence the motivation to use viz thread has low meaning/potential when DrDc is enabled.

Although the above 2 advantages of using separate viz/gpu thread to record/replay DDLs becomes less valuable with DrDc, there are some other factors which point that separate viz thread would be more valuable.

One of the reasons is that it might be useful to not overload viz thread with the gpu work. This is because on desktop platforms, all windows have a single viz instance and hence uses the same viz thread. So heavy animation in one window might affect other windows as well and results in reduced performance for all windows.

Another reason is that with the new skia backend Graphite which is a work in progress, there is no way to issue draw commands directly to skia and the only way to draw is to record and replay although the mechanism is slightly different than DDLs. So even if we merge viz and DrDc thread, we still need to continue to record and replay either on the same or different thread.

Another point to note is that today even if the draw commands are directly issued to skia, skia internally records it in the form of Drawops which are converted into draw calls on flush.

New skia backend graphite also promises more overlap in record and replay which means that the increased parallelism from having separate viz and gpu threads would be more important.

Conclusion

From above analysis, It looks like going forward with the new skia backend Graphite, continuing to have a separate viz and DrDc thread will have higher performance compared to merging viz and DrDc thread.

This document will be updated after gathering additional feedback. Currently a prototype to use viz thread as DrDc thread is also in progress to look into the performance numbers although this will change with new skia backend.

References

1. DrDc design doc.
2. DrDc presentation.
3. Life of a pixel presentation.