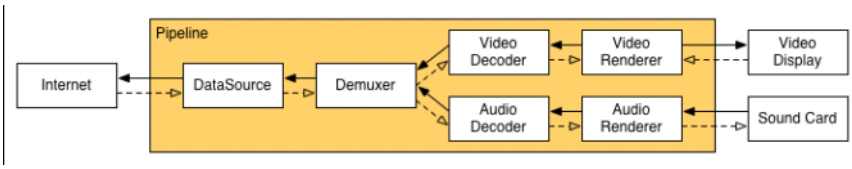
Video Render

2019年11月15日 15:16

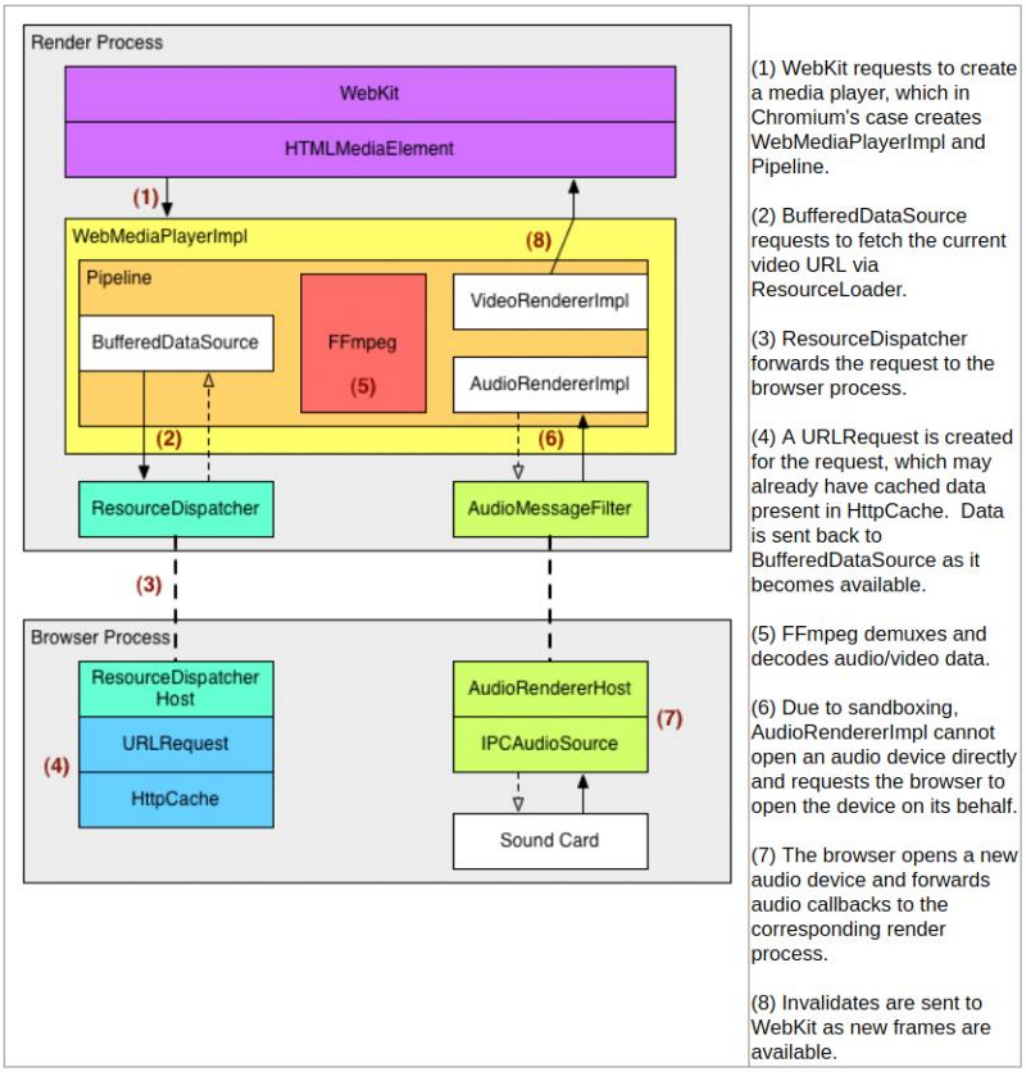
https://www.chromium.org/developers/design-documents/video
https://www.chromium.org/audio-video



过一个步骤称为一个filter，合起来称为 Pipeline

Integration

The following diagram shows the current integration of the media playback pipeline into WebKit and Chromium browser; this is slightly out of date, but the gist remains the same.



media::WebMediaPlayerImpl 用来和blink::HTMLMediaElement对接，它包含一个media::PipelineController；
media::PipelineController 来管理各种filters，包括 media::DataSource,media::Demuxer,media::Renderer；

media::Demuxer 有2种：
media::FFmpegDemuxer：用来播放那些由src直接提供视频地址的视频
media::ChunkDemuxer：用来播放那些由js提供视频内容的视频

media::Renderer 一般是 media::RendererImpl，它包含media::AudioRenderer和media::VideoRenderer实例。
它们又分别包含了media::AudioDecoder和media::VideoDecoder，这些decoder又从media::Demuxer暴露出来的media::DemuxerStream接口。

media/gpu 包含硬件加速的视频解码
media/filters 包含了软件解码，由FFmpeg和libvpx驱动
解码器是由media::RendererFactory提供的，它会按照内部定义的顺序去尝试，第一个成功的会被使用，通常会是硬件解码器。

renderer（AudioRenderer和VideoRenderer）通过 media::AduioRenderSink和media::VideoRendererSink接口来管理音视频的时序和渲染。这些接口会通过定时回调的方式来获取新的音视频帧。

media::AudioRendererSink 通过base::SyncSocket和由browser进程拥有的共享内存来实现。 base::SyncSocket由位于meida/audio 中的media::AudioOutputStream驱动。
meida::VideoRendererSink 由 media::VideoFrameCompositor 的异步回调驱动。media::VideoRendererSink会通过media::TimeSource和media::AudioRenderer通信，以便进行音画同步。

可以查看 chrome://media-internals 来协助调试。
media部分使用DVLOG比较多，
MeidaLog可以发送logs到 chrome://media-internals 中。详见 media/base/media_log.h

WebMediaPlayerImpl

启动调试命令：
```
out/v71.3578/content_shell --enable-logging=stderr --no-sandbox --single-process /media/keyou/backup/UbtData/H_VIDEO_CODECS/VID-4H.mp4
```
视频转码：
```
ffmpeg -i 14.mp4 -c:v h264 14.h264.mp4
```

以下运行于“Media”线程中
获取VideoFrame并存入VRA中。
```
#0  0x00007fffeea34e76 in media::VideoRendererAlgorithm::EnqueueFrame(scoped_refptr<media::VideoFrame> const&) (this=0x318af5c98440, frame=...)
at ../../src/media/filters/video_renderer_algorithm.cc:339
#1  0x00007fffeeb52521 in media::VideoRendererImpl::AddReadyFrame_Locked(scoped_refptr<media::VideoFrame> const&) (this=0x318af534fa20, frame=...)
at ../../src/media/renderers/video_renderer_impl.cc:642
#2  0x00007fffeeb51c8c in media::VideoRendererImpl::FrameReady(media::DecoderStream<(media::DemuxerStream::Type)2>::Status,
scoped_refptr<media::VideoFrame> const&) (this=0x318af534fa20, status=media::DecoderStream<media::DemuxerStream::VI
DEO>::OK, frame=...) at ../../src/media/renderers/video_renderer_impl.cc:530
```
从以下发位置起异步调用
```
#0  0x00007fffee95d7d7 in
media::DecoderStream<(media::DemuxerStream::Type)2>::SatisfyRead(media::DecoderStream<(media::DemuxerStream::Type)2>::Status,
scoped_refptr<media::VideoFrame> const&) (this=0x318af5a7e520, status=media::DecoderStr
eam<media::DemuxerStream::VIDEO>::OK, output=...) at ../../src/media/filters/decoder_stream.cc:415
#1  0x00007fffee966d83 in media::DecoderStream<(media::DemuxerStream::Type)2>::OnPreparedOutputReady(scoped_refptr<media::Vid eoFrame> const&)
(this=0x318af5a7e520, output=...) at ../../src/media/filters/decoder_stream.cc:977
…
#7  0x00007fffeeb6ce54 in media::GpuMemoryBufferVideoFramePool::PoolImpl::CompleteCopyRequestAndMaybeStartNextCopy(scoped_ref ptr<media::VideoFrame>
const&) (this=0x318af7dc5020, video_frame=...) at ../../src/media/video/gpu_memory_buffer_v
ideo_frame_pool.cc:1128
#8  0x00007fffeeb67ff6 in
media::GpuMemoryBufferVideoFramePool::PoolImpl::BindAndCreateMailboxesHardwareFrameResources(scoped_refptr<media::VideoFrame>  const&,
media::GpuMemoryBufferVideoFramePool::PoolImpl::FrameResources*) (this=0x318af7
dc5020, video_frame=..., frame_resources=0x318af47de2f0) at ../../src/media/video/gpu_memory_buffer_video_frame_pool.cc:1030
```
从以下位置发起异步调用，运行在 "CompositorTileW" 线程
```
#0  media::GpuMemoryBufferVideoFramePool::PoolImpl::OnCopiesDone (this=0x318af7dc5020, video_frame=..., frame_resources=0x318 af7965020)
at ../../src/media/video/gpu_memory_buffer_video_frame_pool.cc:766
```

这里并不是真正的Render，只是从VRA取出指定时间范围中的最优的VideoFrame，将它保存在VideoFrameCompositor的current_frame_
```
#0  0x00007fffeeb4f820 in media::VideoRendererImpl::Render(base::TimeTicks, base::TimeTicks, bool) (this=0xd1f5b89ba20, deadl ine_min=...,
deadline_max=..., background_rendering=true) at ../../src/media/renderers/video_renderer_impl.cc:278
#1  0x00007fffeeb502ae in non-virtual thunk to media::VideoRendererImpl::Render(base::TimeTicks, base::TimeTicks, bool) ()
at ../../src/base/bind_internal.h:658
#2  0x00007fffd0796e9c in media::VideoFrameCompositor::CallRender(base::TimeTicks, base::TimeTicks, bool) (this=0xd1f59782e20 , deadline_min=...,
deadline_max=..., background_rendering=true) at ../../src/media/blink/video_frame_compositor.c
```
c:316 在这里将VideoFrame放入current_frame_
```
#3  0x00007fffd0795104 in media::VideoFrameCompositor::BackgroundRender() (this=0xd1f59782e20)
at ../../src/media/blink/video_frame_compositor.cc:289
#4  0x00007fffd0796249 in media::VideoFrameCompositor::OnRendererStateUpdate(bool) (this=0xd1f59782e20, new_state=true)
at ../../src/media/blink/video_frame_compositor.cc:127
```
从下面发起上述请求
```
#0  0x00007fffd0797292 in media::VideoFrameCompositor::Start(media::VideoRendererSink::RenderCallback*) (this=0xd1f59782e20,  callback=
0xd1f5b89ba28) at ../../src/media/blink/video_frame_compositor.cc:194
#1  0x00007fffeeb50e40 in media::VideoRendererImpl::StartSink() (this=0xd1f5b89ba20) at ../../src/media/renderers/video_rende rer_impl.cc:727
#2  0x00007fffeeb50c4b in media::VideoRendererImpl::OnTimeProgressing() (this=0xd1f5b89ba20)
at ../../src/media/renderers/video_renderer_impl.cc:435
#3  0x00007fffeeb45032 in media::RendererImpl::StartPlayback() (this=0xd1f59ffc020) at ../../src/media/renderers/renderer_imp l.cc:828
#4  0x00007fffeeb44843 in media::RendererImpl::OnBufferingStateChange(media::DemuxerStream::Type, media::BufferingState) (thi s=0xd1f59ffc020,
type=media::DemuxerStream::AUDIO, new_buffering_state=media::BUFFERING_HAVE_ENOUGH) at ../../src/
media/renderers/renderer_impl.cc:758
#5  0x00007fffeeb47ce5 in media::RendererImpl::RendererClientInternal::OnBufferingStateChange(media::BufferingState) (this=0x d1f5d3a1750,
state=media::BUFFERING_HAVE_ENOUGH) at ../../src/media/renderers/renderer_impl.cc:49
#6  0x00007fffeeb241d9 in media::AudioRendererImpl::OnBufferingStateChange(media::BufferingState) (this=0xd1f5a974820,
state=media::BUFFERING_HAVE_ENOUGH) at ../../src/media/renderers/audio_renderer_impl.cc:629
#6  media::VideoRendererImpl::OnBufferingStateChange (this=0xd1f5b89ba20, state=media::BUFFERING_HAVE_NOTHING)
at ../../src/media/renderers/video_renderer_impl.cc:387
```

Media线程通过VideoFrameCompositor的成员变量传输VideoFrame到Compositor线程。

==========

以下运行于“Compositor”线程中，从Media线程获取到VideoFrame然后创建CompositorFrame，并且将该CF发送到其它线程创建Surface。

从VideoFrameCompositor中获取当前的VideoFrame，把它填充到FrameData对象。这里使用锁进行同步。

```
#0  0x00007fffd0796569 in media::VideoFrameCompositor::GetCurrentFrame() (this=0x318af4c4f020)
at ../../src/media/blink/video_frame_compositor.cc:156
#1  0x00007fffd0796696 in non-virtual thunk to media::VideoFrameCompositor::GetCurrentFrame() () at ../../src/base/bind_internal.h:516
#0  0x00007fffedcaf1e1 in cc::VideoFrameProviderClientImpl::AcquireLockAndCurrentFrame() (this=0xd1f596902a0)
at ../../src/cc/layers/video_frame_provider_client_impl.cc:83
#1  0x00007fffedcb1d9b in cc::VideoLayerImpl::WillDraw(cc::DrawMode, viz::ClientResourceProvider*) (this=0xd1f59f67b20,
draw_mode=cc::DRAW_MODE_HARDWARE, resource_provider=0xd1f5a4bea90) at ../../src/cc/layers/video_layer_impl.cc:95
#2  0x00007fffede38ba5 in cc::LayerTreeHostImpl::CalculateRenderPasses(cc::LayerTreeHostImpl::FrameData*) (this=0xd1f5a4be820, frame=
0x7fffa0448580) at ../../src/cc/trees/layer_tree_host_impl.cc:1100
#3  0x00007fffede3dd3b in cc::LayerTreeHostImpl::PrepareToDraw(cc::LayerTreeHostImpl::FrameData*) (this=0xd1f5a4be820, frame=0x7fffa0448580)
at ../../src/cc/trees/layer_tree_host_impl.cc:1388
#4  0x00007fffedf04c52 in cc::ProxyImpl::DrawInternal(bool) (this=0xd1f597b3de0, forced_draw=false) at ../../src/cc/trees/proxy_impl.cc:666 在这里创
```

建FrameData

```
#5  0x00007fffedf048aa in cc::ProxyImpl::ScheduledActionDrawIfPossible() (this=0xd1f597b3de0) at ../../src/cc/trees/proxy_imp l.cc:537
#6  0x00007fffedd0fd98 in cc::Scheduler::DrawIfPossible() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:704
#7  0x00007fffedd09961 in cc::Scheduler::ProcessScheduledActions() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:805
#8  0x00007fffedd0fbba in cc::Scheduler::OnBeginImplFrameDeadline() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:692
```

使用VideoFrame创建Resource对象，可能是Hardware也可能是Software

```
#0  0x00007fffeeb55b50 in media::VideoResourceUpdater::CreateExternalResourcesFromVideoFrame(scoped_refptr<media::VideoFrame> ) (this=0xd1f5d7aa8e0,
video_frame=...) at ../../src/media/renderers/video_resource_updater.cc:543
#1  0x00007fffeeb553f7 in media::VideoResourceUpdater::ObtainFrameResources(scoped_refptr<media::VideoFrame>) (this=0xd1f5d7a a8e0, video_frame=...)
at ../../src/media/renderers/video_resource_updater.cc:372
#2  0x00007fffedcb24cf in cc::VideoLayerImpl::WillDraw(cc::DrawMode, viz::ClientResourceProvider*) (this=0xd1f59f67b20,
draw_mode=cc::DRAW_MODE_HARDWARE, resource_provider=0xd1f5a4bea90) at ../../src/cc/layers/video_layer_impl.cc:116
#3  0x00007fffede38ba5 in cc::LayerTreeHostImpl::CalculateRenderPasses(cc::LayerTreeHostImpl::FrameData*) (this=0xd1f5a4be820, frame=
0x7fffa0448580) at ../../src/cc/trees/layer_tree_host_impl.cc:1100
#4  0x00007fffede3dd3b in cc::LayerTreeHostImpl::PrepareToDraw(cc::LayerTreeHostImpl::FrameData*) (this=0xd1f5a4be820, frame=0x7fffa0448580)
at ../../src/cc/trees/layer_tree_host_impl.cc:1388
#5  0x00007fffedf04c52 in cc::ProxyImpl::DrawInternal(bool) (this=0xd1f597b3de0, forced_draw=false) at ../../src/cc/trees/proxy_impl.cc:666
#6  0x00007fffedf048aa in cc::ProxyImpl::ScheduledActionDrawIfPossible() (this=0xd1f597b3de0) at ../../src/cc/trees/proxy_imp l.cc:537
#7  0x00007fffedd0fd98 in cc::Scheduler::DrawIfPossible() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:704
#8  0x00007fffedd09961 in cc::Scheduler::ProcessScheduledActions() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.c c:805
#9  0x00007fffedd0fbba in cc::Scheduler::OnBeginImplFrameDeadline() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler. cc:692
```

将VideoFrame作为资源添加到viz::DrawQuad中，DrawQuad又被添加到RenderPass中，RenderPass被添加到FrameData中，FrameData被用来创建CF。一个DrawQuad可以有很多种Material，其中kSurfaceContent类型的Material表示该DrawQuad是Surface的quad，通过它可以获取到它对应的surface，因此最终可以通过CF获取到它的子Surface。

```
#1  0x00007fffedcb29c3 in cc::VideoLayerImpl::AppendQuads(viz::RenderPass*, cc::AppendQuadsData*) (this=0xd1f59f67b20, render _pass=0xd1f5e9b49c0,
append_quads_data=0x7fffa0446310) at ../../src/cc/layers/video_layer_impl.cc:156
#2  0x00007fffede38e90 in cc::LayerTreeHostImpl::CalculateRenderPasses(cc::LayerTreeHostImpl::FrameData*) (this=0xd1f5a4be820, frame=
0x7fffa0448580) at ../../src/cc/trees/layer_tree_host_impl.cc:1107
#3  0x00007fffede3dd3b in cc::LayerTreeHostImpl::PrepareToDraw(cc::LayerTreeHostImpl::FrameData*) (this=0xd1f5a4be820, frame=0x7fffa0448580)
at ../../src/cc/trees/layer_tree_host_impl.cc:1388
#4  0x00007fffedf04c52 in cc::ProxyImpl::DrawInternal(bool) (this=0xd1f597b3de0, forced_draw=false) at ../../src/cc/trees/proxy_impl.cc:666
#5  0x00007fffedf048aa in cc::ProxyImpl::ScheduledActionDrawIfPossible() (this=0xd1f597b3de0) at ../../src/cc/trees/proxy_imp l.cc:537
#6  0x00007fffedd0fd98 in cc::Scheduler::DrawIfPossible() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:704
#7  0x00007fffedd09961 in cc::Scheduler::ProcessScheduledActions() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.c c:805
#8  0x00007fffedd0fbba in cc::Scheduler::OnBeginImplFrameDeadline() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler. cc:692
```

使用FrameData创建CompositorFrame并通过CompositorFrameSink这个mojo接口发送到Browser进程的content_shell线程。CF中会包含一个RenderPass列表。

```
#0  0x00007fffd03dc899 in cc::mojo_embedder::AsyncLayerTreeFrameSink::SubmitCompositorFrame(viz::CompositorFrame) (this=0xd1f5c1671a0, frame=...)
at ../../src/cc/mojo_embedder/async_layer_tree_frame_sink.cc:165
#1  0x00007fffede45ae4 in cc::LayerTreeHostImpl::DrawLayers(cc::LayerTreeHostImpl::FrameData*) (this=0xd1f5a4be820, frame=0x7 fffa0448580)
at ../../src/cc/trees/layer_tree_host_impl.cc:2085
#2  0x00007fffedf04cf4 in cc::ProxyImpl::DrawInternal(bool) (this=0xd1f597b3de0, forced_draw=false) at ../../src/cc/trees/proxy_impl.cc:673
#3  0x00007fffedf048aa in cc::ProxyImpl::ScheduledActionDrawIfPossible() (this=0xd1f597b3de0) at ../../src/cc/trees/proxy_imp l.cc:537
#4  0x00007fffedd0fd98 in cc::Scheduler::DrawIfPossible() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:704
#5  0x00007fffedd09961 in cc::Scheduler::ProcessScheduledActions() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.c c:805
#6  0x00007fffedd0fbba in cc::Scheduler::OnBeginImplFrameDeadline() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler. cc:692
```

scheduler中维护了一个SchedulerStateMachine

```
#0  0x00007fffedf0562a in cc::ProxyImpl::ScheduledActionCommit() (this=0xd1f597b3de0) at ../../src/cc/trees/proxy_impl.cc:567
#1  0x00007fffedd09867 in cc::Scheduler::ProcessScheduledActions() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:790
#2  0x00007fffedd0ade8 in cc::Scheduler::NotifyReadyToCommit() (this=0xd1f5a4adc20) at ../../src/cc/scheduler/scheduler.cc:16 3
#3  0x00007fffedf0092e in cc::ProxyImpl::NotifyReadyToCommitOnImpl(cc::CompletionEvent*, cc::LayerTreeHost*, base::TimeTicks, bool) (this=
0xd1f597b3de0, completion=0x7fffa265ed38, layer_tree_host=0xd1f5a3f8020, main_thread_start_time=...,
hold_commit_for_activation=false) at ../../src/cc/trees/proxy_impl.cc:268
```

这个状态机有以下状态：

```
122   enum class Action {
123     NONE,
124     SEND_BEGIN_MAIN_FRAME,
125     COMMIT,
126     ACTIVATE_SYNC_TREE,
127     PERFORM_IMPL_SIDE_INVALIDATION,
128     DRAW_IF_POSSIBLE,
129     DRAW_FORCED,
130     DRAW_ABORT,
131     BEGIN_LAYER_TREE_FRAME_SINK_CREATION,
132     PREPARE_TILES,
133     INVALIDATE_LAYER_TREE_FRAME_SINK,
134     NOTIFY_BEGIN_MAIN_FRAME_NOT_SENT,
135   };
```

Compositor线程通过CompositorFrameSink::SubmitCompositorFrame这个mojo接口将CF传输给content_shell线程。

==========

在单进程中，以下运行在“content_shell”线程中，他是单进程中的1号线程。

在这个地方通过mojo接口获取到由Compositor线程发送的CompositorFrame，根据需要创建Surface（该Surface可以通过root surface获取到），并且通过Surface::QueueFrame将它设置为Pending Frame或者Active Frame。一个Surface最多同时拥有2个Frame。这样CF被存储在了Surface中，而Surface又被存储在了SurfaceManager中，其它地方会根据SurfaceId获取该Surface。根据前面CF的创建过程可知，可以用CF获取到子Surface，因此通过这种方式形成了一个Surface树。

```
#0  0x00007fffd8ee3531 in viz::Surface::ActivateFrame(viz::Surface::FrameData, base::Optional<base::TimeDelta>) (this=0xd1f5a be81a0,
frame_data=..., duration=...) at ../../src/components/viz/service/surfaces/surface.cc:467
#1  0x00007fffd8ee2812 in viz::Surface::QueueFrame(viz::CompositorFrame, unsigned long, base::ScopedClosureRunner, base::Once Callback<void
(gfx::PresentationFeedback const&)>) (this=0xd1f5abe81a0, frame=..., frame_index=237, frame_rejected
_callback=..., presented_callback=...) at ../../src/components/viz/service/surfaces/surface.cc:252
#2  0x00007fffd8e65df9 in viz::CompositorFrameSinkSupport::MaybeSubmitCompositorFrameInternal(viz::LocalSurfaceId const&, viz::CompositorFrame,
base::Optional<viz::HitTestRegionList>, unsigned long, base::OnceCallback<void (std::__1::vecto
r<viz::ReturnedResource, std::__1::allocator<viz::ReturnedResource> > const&)>) (this=0xd1f5989c260, local_surface_id=..., fr ame=...,
hit_test_region_list=..., submit_time=0, callback=...) at ../../src/components/viz/service/frame_sinks/co
mpositor_frame_sink_support.cc:451
```

这个方法会根据需要为当前CF创建新的Surface，Surface存储在SurfaceManager中。

```
#3  0x00007fffd8e641a6 in viz::CompositorFrameSinkSupport::MaybeSubmitCompositorFrame(viz::LocalSurfaceId const&, viz::Compos itorFrame,
base::Optional<viz::HitTestRegionList>, unsigned long, base::OnceCallback<void (std::__1::vector<viz::R
eturnedResource, std::__1::allocator<viz::ReturnedResource> > const&)>) (this=0xd1f5989c260, local_surface_id=..., frame=...,
hit_test_region_list=..., submit_time=0, callback=...) at ../../src/components/viz/service/frame_sinks/compositor
_frame_sink_support.cc:608
#4  0x00007fffd8e63fba in viz::CompositorFrameSinkSupport::SubmitCompositorFrame(viz::LocalSurfaceId const&, viz::CompositorF rame,
base::Optional<viz::HitTestRegionList>, unsigned long) (this=0xd1f5989c260, local_surface_id=..., frame=...,
 hit_test_region_list=..., submit_time=0) at ../../src/components/viz/service/frame_sinks/compositor_frame_sink_support.cc:277
#5  0x00007ffff4d06582 in content::DelegatedFrameHost::SubmitCompositorFrame(viz::LocalSurfaceId const&, viz::CompositorFrame ,
base::Optional<viz::HitTestRegionList>) (this=0xd1f5a20b380, local_surface_id=..., frame=..., hit_test_region_li
st=...) at ../../src/content/browser/renderer_host/delegated_frame_host.cc:279
#6  0x00007ffff47a8f0d in content::RenderWidgetHostViewAura::SubmitCompositorFrame(viz::LocalSurfaceId const&, viz::Composito rFrame,
base::Optional<viz::HitTestRegionList>) (this=0xd1f5a5278a0, local_surface_id=..., frame=..., hit_test_reg
ion_list=...) at ../../src/content/browser/renderer_host/render_widget_host_view_aura.cc:904
#7  0x00007ffff477ea0f in content::RenderWidgetHostImpl::SubmitCompositorFrame(viz::LocalSurfaceId const&, viz::CompositorFrame,
base::Optional<viz::HitTestRegionList>, unsigned long) (this=0xd1f5c1d5e20, local_surface_id=..., frame=..., h
it_test_region_list=..., submit_time=0) at ../../src/content/browser/renderer_host/render_widget_host_impl.cc:2870
```

在单进程中，以下运行在“content_shell”线程中，在多进程中可能运行于GPU进程（TODO:需要验证）

在下面取出之前创建的Surface并进行合成和渲染。最终将渲染的指令转换成数据存入共享内存中，该内存作为ring buffer来使用。

```
#0  0x00007fffd95e0af1 in gpu::gles2::cmds::Uniform2f::Init(int, float, float) (this=0x7fff9acc58b8, _location=1, _x=0, _y=0)
at ../../src/gpu/command_buffer/common/gles2_cmd_format_autogen.h:8643
#1  0x00007fffd95cc0e7 in gpu::gles2::GLES2CmdHelper::Uniform2f(int, float, float) (this=0xd1f59e0dca0, location=1, x=0, y=0)
at ../../src/gpu/command_buffer/client/gles2_cmd_helper_autogen.h:1732
```

在这个类中为指令申请空间。

```
#2  0x00007fffd95aee80 in gpu::gles2::GLES2Implementation::Uniform2f(int, float, float) (this=0xd1f59fe8520, location=1, x=0, y=0)
at ../../src/gpu/command_buffer/client/gles2_implementation_impl_autogen.h:2106
#0  0x00007fffd8d79544 in viz::GLRenderer::DrawYUVVideoQuad(viz::YUVVideoDrawQuad const*, gfx::QuadF const*) (this=0xd1f59fe3c20, quad=
0xd1f5eebd660, clip_region=0x0) at ../../src/components/viz/service/display/gl_renderer.cc:2095
```

在这个地方取出包装有VideoFrame的DrawQuad，调用GL方法进行绘制。并且在第一次进入这里的时候创建相应的Shader和Program。

```
#1  0x00007fffd8d7550b in viz::GLRenderer::DoDrawQuad(viz::DrawQuad const*, gfx::QuadF const*) (this=0xd1f59fe3c20, quad=0xd1 f5eebd660,
clip_region=0x0) at ../../src/components/viz/service/display/gl_renderer.cc:511
#2  0x00007fffd8d0f815 in viz::DirectRenderer::DrawRenderPass(viz::RenderPass const*) (this=0xd1f59fe3c20, render_pass=0xd1f5 bfb8180)
at ../../src/components/viz/service/display/direct_renderer.cc:642
#3  0x00007fffd8d0e1d6 in viz::DirectRenderer::DrawRenderPassAndExecuteCopyRequests(viz::RenderPass*) (this=0xd1f59fe3c20, re nder_pass=
0xd1f5bfb8180) at ../../src/components/viz/service/display/direct_renderer.cc:527
#0  0x00007fffd8d0ce23 in viz::DirectRenderer::DrawFrame(std::__1::vector<std::__1::unique_ptr<viz::RenderPass, std::__
1::default_delete<viz::RenderPass> >, std::__1::allocator<std::__1::unique_ptr<viz::RenderPass, std::__1::default_delete
<viz::RenderPass> > > >*, float, gfx::Size const&) (this=0xd1f59fe3c20, render_passes_in_draw_order=0x7fffffff89f8, device_sc ale_factor=1,
device_viewport_size=...) at ../../src/components/viz/service/display/direct_renderer.cc:321
#1  0x00007fffd8d24ad5 in viz::Display::DrawAndSwap() (this=0xd1f5a5e9860) at ../../src/components/viz/service/display/display.cc:400
```

这个函数中调用aggregator进行合成，aggregator中存有viz::DisplayResourceProvider类型的resource provider，它用于维护跨进程/线程的资源传递，它包装了不同的传递方式，比如GL Textures,GpuMemoryBuffers以及software bitmaps。资源通过ResourceId来取用。Display中还存储有Root Surface。这个函数会使用硬件渲染或者软件渲染或者skia渲染。

```
#2  0x00007fffd8d5e92f in viz::DisplayScheduler::DrawAndSwap() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:212
```

该方法会将所有的RenderPass转换成GL命令，然后发送一个GpuChannelMsg_FlushDeferredMessages IPC命令，这个命令会驱动最终的渲染。

```
#3  0x00007fffd8d5d7b5 in viz::DisplayScheduler::AttemptDrawAndSwap() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:485
#4  0x00007fffd8d5d0c0 in viz::DisplayScheduler::OnBeginFrameDeadline() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:502
```

启动Frame的Draw，它有很多触发逻辑。

```
#5  0x00007fffd8d5ef26 in viz::DisplayScheduler::OnBeginFrameDerivedImpl(viz::BeginFrameArgs const&) (this=0xd1f5a0538e0, arg s=...)
at ../../src/components/viz/service/display/display_scheduler.cc:253
#6  0x00007fffef6a3509 in viz::BeginFrameObserverBase::OnBeginFrame(viz::BeginFrameArgs const&) (this=0xd1f5a0538e0, args=... )
at ../../src/components/viz/common/frame_sinks/begin_frame_source.cc:66
#7  0x00007fffef6a8559 in viz::(anonymous namespace)::FilterAndIssueBeginFrame(viz::BeginFrameObserver*, viz::BeginFrameArgs const&) (observer=
0xd1f5a0538e0, args=...) at ../../src/components/viz/common/frame_sinks/begin_frame_source.cc:41
#8  0x00007fffef6aafb3 in viz::DelayBasedBeginFrameSource::OnTimerTick() (this=0xd1f59f9e200)
at ../../src/components/viz/common/frame_sinks/begin_frame_source.cc:267
#9  0x00007fffef6bbf08 in viz::DelayBasedTimeSource::OnTimerTick() (this=0xd1f59546de0)
at ../../src/components/viz/common/frame_sinks/delay_based_time_source.cc:78
```

这个Timer一旦启动就会自己在内部循环，按照需要的帧率定时触发。

下面是另一个触发逻辑：

```
#0  0x00007fffd8d0ce23 in viz::DirectRenderer::DrawFrame(std::__1::vector<std::__1::unique_ptr<viz::RenderPass, std::__
1::default_delete<viz::RenderPass> >, std::__1::allocator<std::__1::unique_ptr<viz::RenderPass, std::__1::default_delete
<viz::RenderPass> > > >*, float, gfx::Size const&) (this=0xd1f59fe3c20, render_passes_in_draw_order=0x7fffffff98e8, device_sc ale_factor=1,
device_viewport_size=...) at ../../src/components/viz/service/display/direct_renderer.cc:321
#1  0x00007fffd8d24ad5 in viz::Display::DrawAndSwap() (this=0xd1f5a5e9860) at ../../src/components/viz/service/display/display.cc:400
#2  0x00007fffd8d5e92f in viz::DisplayScheduler::DrawAndSwap() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:212
#3  0x00007fffd8d5d7b5 in viz::DisplayScheduler::AttemptDrawAndSwap() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:485
#4  0x00007fffd8d5d0c0 in viz::DisplayScheduler::OnBeginFrameDeadline() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:502
```

在这里创建GpuCommandBufferMsg_AsyncFlush消息，并将它包装进GpuChannelMsg_FlushDeferredMessages，然后发送给Chrome_InProcGp线程，从而触发真正是渲染。

```
#0  0x00007ffff120f67f in gpu::GpuChannelHost::EnqueuePendingOrderingBarrier() (this=0xd1f5a5914a0)
at ../../src/gpu/ipc/client/gpu_channel_host.cc:156
```

#1 0x00007ffff120fd1f in gpu::GpuChannelHost::InternalFlush(unsigned int) (this=0xd1f5a5914a0, deferred_message_id=4294967295)
at ../../src/gpu/ipc/client/gpu_channel_host.cc:169
#2 0x00007ffff121019b in gpu::GpuChannelHost::VerifyFlush(unsigned int) (this=0xd1f5a5914a0, deferred_message_id=4294967295)
at ../../src/gpu/ipc/client/gpu_channel_host.cc:139
#3 0x00007ffff11ff590 in gpu::CommandBufferProxyImpl::EnsureWorkVisible() (this=0xd1f5a4a3fa0)
at ../../src/gpu/ipc/client/command_buffer_proxy_impl.cc:512
#4 0x00007fffd958d44b in gpu::gles2::GLES2Implementation::VerifySyncTokensCHROMIUM(signed char**, int) (this=0xd1f59fe8520, sync_tokens=
0xd1f5c7353e0, count=1) at ../../src/gpu/command_buffer/client/gles2_implementation.cc:6291
#5 0x00007fffd8d3dcf0 in viz::DisplayResourceProvider::DeleteAndReturnUnusedResourcesToChild(std::__1::__hash_map_iterator<std::__1::
__hash_iterator<std::__1::__hash_node<std::__1::__hash_value_type<int, viz::DisplayResourceProvider::Child
d>, void*>*> >, viz::DisplayResourceProvider::DeleteStyle, std::__1::vector<unsigned int, std::__1::allocator<unsigned int> > const&) (this=
0xd1f59fe6ca0, child_it=..., style=viz::DisplayResourceProvider::NORMAL, unused=...) at ../../src/c
omponents/viz/service/display/display_resource_provider.cc:739
#6 0x00007fffd8d40f98 in viz::DisplayResourceProvider::SetBatchReturnResources(bool) (this=0xd1f59fe6ca0, batch=false)
at ../../src/components/viz/service/display/display_resource_provider.cc:798
#7 0x00007fffd8d4233a in viz::DisplayResourceProvider::ScopedBatchReturnResources::~ScopedBatchReturnResources() (this=0x7ff fffff8a28)
at ../../src/components/viz/service/display/display_resource_provider.cc:959
#8 0x00007fffd8d26223 in viz::Display::DrawAndSwap() (this=0xd1f5a5e9860) at ../../src/components/viz/service/display/display.cc:487
#9 0x00007fffd8d5e92f in viz::DisplayScheduler::DrawAndSwap() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:212
#10 0x00007fffd8d5d7b5 in viz::DisplayScheduler::AttemptDrawAndSwap() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:485
#11 0x00007fffd8d5d0c0 in viz::DisplayScheduler::OnBeginFrameDeadline() (this=0xd1f5a0538e0)
at ../../src/components/viz/service/display/display_scheduler.cc:502

content_shell线程通过GpuChannelMsg_FlushDeferredMessages这个IPC命令将存有GL命令的command buffer数据通过共享内存的方式传给Chrome_InProcGp线程。

==========

在单进程模式，以下在"Chrome_InProcGp"线程中执行

gpu channel的IPC机制收到其它端发送的GpuChannelMsg_FlushDeferredMessages之后会驱动command buffer模块的Scheduler进行调度，最终调用真实GL函数进行绘制过程。

#0 0x00007fffd973e784 in gl::DebugGLApi::glDrawElementsFn(unsigned int, int, unsigned int, void const*) (this=0xd1f5a2a8840, mode=4, count=6,
type=5123, indices=0x0) at ../../src/ui/gl/gl_bindings_autogen_gl.cc:10255
#1 0x00007fffd9c4025b in gpu::gles2::GLES2DecoderImpl::DoDrawElements(char const*, bool, unsigned int, int, unsigned int, int t, int) (this=
0xd1f59ff9320, function_name=0x7fffd9ac83bc "glDrawElements", instanced=false, mode=4, count=6, type
=5123, offset=0, primcount=1) at ../../src/gpu/command_buffer/service/gles2_cmd_decoder.cc:10931
#2 0x00007fffd9bdb0f1 in gpu::gles2::GLES2DecoderImpl::HandleDrawElements(unsigned int, void const volatile*) (this=0xd1f59f f9320,
immediate_data_size=0, cmd_data=0x7fff98f34eec) at ../../src/gpu/command_buffer/service/gles2_cmd_decoder.c
c:10968
#3 0x00007fffd9c621b9 in gpu::gles2::GLES2DecoderImpl::DoCommandsImpl<true>(unsigned int, void const volatile*, int, int*) (this=0xd1f59ff9320,
num_commands=20, buffer=0x7fff98f34e44, num_entries=746, entries_processed=0x7fffba748cd4) at
../../src/gpu/command_buffer/service/gles2_cmd_decoder.cc:5664 在这个地方从CommandBuffers中取出GL command，然后转换成真实的GL调用
#4 0x00007fffd9c23c46 in gpu::gles2::GLES2DecoderImpl::DoCommands(unsigned int, void const volatile*, int, int*) (this=0xd1f 59ff9320,
num_commands=20, buffer=0x7fff98f34e44, num_entries=746, entries_processed=0x7fffba748cd4) at ../../src/
gpu/command_buffer/service/gles2_cmd_decoder.cc:5721
#5 0x00007ffff11539fd in gpu::CommandBufferService::Flush(int, gpu::AsyncAPIInterface*) (this=0xd1f5969aa20, put_offset=2033 87, handler=
0xd1f59ff9320) at ../../src/gpu/command_buffer/service/command_buffer_service.cc:69 这个类中存有用于存储command的共享内存。
#6 0x00007fffd9862fad in gpu::CommandBufferStub::OnAsyncFlush(int, unsigned int) (this=0xd1f59ef92a0, put_offset=203387, flu sh_id=527)
at ../../src/gpu/ipc/service/command_buffer_stub.cc:538
……
#11 0x00007fffd9860c4d in gpu::CommandBufferStub::OnMessageReceived(IPC::Message const&) (this=0xd1f59ef92a0, message=...)
at ../../src/gpu/ipc/service/command_buffer_stub.cc:199 在这里通过IPC_BEGIN_MESSAGE_MAP()宏根据message.type()的值进行message的分发
#12 0x00007ffff7fe7b18 in IPC::MessageRouter::RouteMessage(IPC::Message const&) (this=0xd1f59577ea0, msg=...) at ../../src/ip c/message_router.cc:56
#13 0x00007fffd9884032 in gpu::GpuChannel::HandleMessageHelper(IPC::Message const&) (this=0xd1f59577e20, msg=...)
at ../../src/gpu/ipc/service/gpu_channel.cc:513
#14 0x00007fffd987fc7b in gpu::GpuChannel::HandleMessage(IPC::Message const&) (this=0xd1f59577e20, msg=...)
at ../../src/gpu/ipc/service/gpu_channel.cc:489
……
#20 0x00007ffff11670d8 in gpu::Scheduler::RunNextTask() (this=0xd1f59cfbec0) at ../../src/gpu/command_buffer/service/schedule r.cc:526
由以下部分异步调用进来，以下运行在“Chrome_ChildIOT”线程。
#0 0x00007ffff1162031 in gpu::Scheduler::TryScheduleSequence(gpu::Scheduler::Sequence*) (this=0xd1f59cfbec0, sequence=0xd1f5 b011f00)
at ../../src/gpu/command_buffer/service/scheduler.cc:467
#1 0x00007ffff116525c in gpu::Scheduler::ScheduleTaskHelper(gpu::Scheduler::Task) (this=0xd1f59cfbec0, task=...)
at ../../src/gpu/command_buffer/service/scheduler.cc:403
#2 0x00007ffff1165425 in gpu::Scheduler::ScheduleTasks(std::__1::vector<gpu::Scheduler::Task, std::__1::allocator<gpu::Sched uler::Task> >) (this=
0xd1f59cfbec0, tasks=...) at ../../src/gpu/command_buffer/service/scheduler.cc:376 它会把task加入自己维护的一个sequence队列中
#3 0x00007fffd987f4a4 in gpu::GpuChannelMessageFilter::OnMessageReceived(IPC::Message const&) (this=0xd1f59cfe260, message=...)
at ../../src/gpu/ipc/service/gpu_channel.cc:270 在这个地方收到GpuChannelMsg_FlushDeferredMessages，它会把包装的message通过GpuChannel::HandleMessage包装
成Task传给Scheduler进行调度

==========

Thread: Media

创建VideoFrame

#0 0x00007fffee897866 in media::VideoFrame::WrapVideoFrame(scoped_refptr<media::VideoFrame> const&, media::VideoPixelFormat, gfx::Rect const&,
gfx::Size const&) (frame=..., format=media::PIXEL_FORMAT_I420, visible_rect=..., natural_size=.
..) at ../../src/media/base/video_frame.cc:475
#1 0x00007fffee8a75b2 in media::VideoFramePool::PoolImpl::CreateFrame(media::VideoPixelFormat, gfx::Size const&, gfx::Rect c onst&, gfx::Size
const&, base::TimeDelta) (this=0x318af8495390, format=media::PIXEL_FORMAT_I420, coded_size=..., v
isible_rect=..., natural_size=..., timestamp=...) at ../../src/media/base/video_frame_pool.cc:109
#2 0x00007fffee8a7bbd in media::VideoFramePool::CreateFrame(media::VideoPixelFormat, gfx::Size const&, gfx::Rect const&, gfx ::Size const&,
base::TimeDelta) (this=0x318af74ad3e0, format=media::PIXEL_FORMAT_I420, coded_size=..., visible_rec
t=..., natural_size=..., timestamp=...) at ../../src/media/base/video_frame_pool.cc:155
#3 0x00007fffeea88da5 in media::FFmpegVideoDecoder::GetVideoBuffer(AVCodecContext*, AVFrame*, int) (this=0x318af74ad2e0, cod ec_context=
0x318af7d86040, frame=0x318af55980c0, flags=1) at ../../src/media/filters/ffmpeg_video_decoder.cc:175
#4 0x00007fffeea8b863 in media::GetVideoBufferImpl(AVCodecContext*, AVFrame*, int) (s=0x318af7d86040, frame=0x318af55980c0, flags=1)
at ../../src/media/filters/ffmpeg_video_decoder.cc:104
#5 0x00007fffd7959eb5 in get_buffer_internal (avctx=0x318af7d86040, frame=0x318af55980c0, flags=1)
at ../../src/third_party/ffmpeg/libavcodec/decode.c:1899
#6 0x00007fffd7959eb5 in ff_get_buffer (avctx=0x318af7d86040, frame=0x318af55980c0, flags=1)
at ../../src/third_party/ffmpeg/libavcodec/decode.c:1924
#7 0x00007fffd7976d06 in thread_get_buffer_internal (avctx=0x318af7d86040, f=0x318af7d015b8, flags=1)
at ../../src/third_party/ffmpeg/libavcodec/pthread_frame.c:890

```
#8  0x00007fffd7976d06 in ff_thread_get_buffer (avctx=0x318af7d86040, f=0x318af7d015b8, flags=1)
at ../../src/third_party/ffmpeg/libavcodec/pthread_frame.c:966
#9  0x00007fffd7a7ee9e in alloc_picture (h=<optimized out>, pic=<optimized out>) at ../../src/third_party/ffmpeg/libavcodec/h 264_slice.c:195
#10 0x00007fffd7a7ee9e in h264_frame_start (h=0x318af7d01040) at ../../src/third_party/ffmpeg/libavcodec/h264_slice.c:505
#11 0x00007fffd7a7b741 in h264_field_start (h=<optimized out>, sl=<optimized out>, nal=<optimized out>, first_slice=<optimize d out>)
at ../../src/third_party/ffmpeg/libavcodec/h264_slice.c:1602
#12 0x00007fffd7a7b741 in ff_h264_queue_decode_slice (h=<optimized out>, nal=<optimized out>) at ../../src/third_party/ffmpeg /libavcodec/h264
_slice.c:2131
#13 0x00007fffd7a8440e in decode_nal_units (h=<optimized out>, buf=<optimized out>, buf_size=<optimized out>)
at ../../src/third_party/ffmpeg/libavcodec/h264dec.c:670
#14 0x00007fffd7a8440e in h264_decode_frame (avctx=0x318af7d86040, data=<optimized out>, got_frame=0x7fff95f34808, avpkt=<opt imized out>)
at ../../src/third_party/ffmpeg/libavcodec/h264dec.c:994
#15 0x00007fffd795739c in decode_simple_internal (avctx=0x318af7d86040, frame=0x318af4854040)
at ../../src/third_party/ffmpeg/libavcodec/decode.c:437
#16 0x00007fffd795739c in decode_simple_receive_frame (avctx=0x318af7d86040, frame=0x318af4854040)
at ../../src/third_party/ffmpeg/libavcodec/decode.c:633
#17 0x00007fffd795739c in decode_receive_frame_internal (avctx=<optimized out>, frame=0x318af4854040)
at ../../src/third_party/ffmpeg/libavcodec/decode.c:651
#18 0x00007fffd795720f in avcodec_send_packet (avctx=0x318af7d86040, avpkt=0x7fff95f34de0) at ../../src/third_party/ffmpeg/li bavcodec/decode.c:709
#19 0x00007fffeeb97818 in media::FFmpegDecodingLoop::DecodePacket(AVPacket const*, base::RepeatingCallback<bool (AVFrame*)>) (this=0x318af8096c80,
packet=0x7fff95f34de0, frame_ready_cb=...) at ../../src/media/ffmpeg/ffmpeg_decoding_loop.cc
:26
#20 0x00007fffeea8a65b in media::FFmpegVideoDecoder::FFmpegDecode(media::DecoderBuffer const&) (this=0x318af74ad2e0, buffer=. ..)
at ../../src/media/filters/ffmpeg_video_decoder.cc:372
#21 0x00007fffeea8a2cc in media::FFmpegVideoDecoder::Decode(scoped_refptr<media::DecoderBuffer>, base::RepeatingCallback<void
(media::DecodeStatus)> const&) (this=0x318af74ad2e0, buffer=..., decode_cb=...) at ../../src/media/filters/ffmpeg
_video_decoder.cc:322
#22 0x00007fffee9639e7 in media::DecoderStream<(media::DemuxerStream::Type)2>::DecodeInternal(scoped_refptr<media::DecoderBuf fer>) (this=
0x318af5a7e520, buffer=...) at ../../src/media/filters/decoder_stream.cc:471
#23 0x00007fffee962f04 in media::DecoderStream<(media::DemuxerStream::Type)2>::Decode(scoped_refptr<media::DecoderBuffer>) (t his=0x318af5a7e520,
buffer=...) at ../../src/media/filters/decoder_stream.cc:438
#24 0x00007fffee965f28 in media::DecoderStream<(media::DemuxerStream::Type)2>::OnBufferReady(media::DemuxerStream::Status,
scoped_refptr<media::DecoderBuffer>) (this=0x318af5a7e520, status=media::DemuxerStream::kOk, buffer=...) at ../../sr
c/media/filters/decoder_stream.cc:791
```

Decode完了之后会调用进CreateHardwareFrame，在这里将内存中的像素数据拷贝到GPU内存中。

```
#0  0x00007fffeeb66575 in media::GpuMemoryBufferVideoFramePool::PoolImpl::CreateHardwareFrame(scoped_refptr<media::VideoFrame > const&,
base::OnceCallback<void (scoped_refptr<media::VideoFrame> const&)>) (this=0x318af7dc5020, video_frame=..
., frame_ready_cb=...) at ../../src/media/video/gpu_memory_buffer_video_frame_pool.cc:640
#1  0x00007fffeeb6ed8c in media::GpuMemoryBufferVideoFramePool::MaybeCreateHardwareFrame(scoped_refptr<media::VideoFrame> con st&,
base::OnceCallback<void (scoped_refptr<media::VideoFrame> const&)>) (this=0x318af77b59d0, video_frame=..., fr
ame_ready_cb=...) at ../../src/media/video/gpu_memory_buffer_video_frame_pool.cc:1237
……
#7  0x00007fffee95f111 in media::DecoderStream<(media::DemuxerStream::Type)2>::MaybePrepareAnotherOutput() (this=0x318af5a7e5 20)
at ../../src/media/filters/decoder_stream.cc:950
#8  0x00007fffee9625da in media::DecoderStream<(media::DemuxerStream::Type)2>::OnDecodeOutputReady(scoped_refptr<media::Video Frame> const&) (this=
0x318af5a7e520, output=...) at ../../src/media/filters/decoder_stream.cc:612
……
#20 0x00007fffeeb97b2d in media::FFmpegDecodingLoop::DecodePacket(AVPacket const*, base::RepeatingCallback<bool (AVFrame*)>) (this=0x318af8096c80,
packet=0x7fff95f34de0, frame_ready_cb=...) at ../../src/media/ffmpeg/ffmpeg_decoding_loop.cc
:60
#21 0x00007fffeea8a65b in media::FFmpegVideoDecoder::FFmpegDecode(media::DecoderBuffer const&) (this=0x318af74ad2e0, buffer=. ..)
at ../../src/media/filters/ffmpeg_video_decoder.cc:372
#22 0x00007fffeea8a2cc in media::FFmpegVideoDecoder::Decode(scoped_refptr<media::DecoderBuffer>, base::RepeatingCallback<void
(media::DecodeStatus)> const&) (this=0x318af74ad2e0, buffer=..., decode_cb=...) at ../../src/media/filters/ffmpeg
_video_decoder.cc:322
#23 0x00007fffee9639e7 in media::DecoderStream<(media::DemuxerStream::Type)2>::DecodeInternal(scoped_refptr<media::DecoderBuf fer>) (this=
0x318af5a7e520, buffer=...) at ../../src/media/filters/decoder_stream.cc:471
#24 0x00007fffee962f04 in media::DecoderStream<(media::DemuxerStream::Type)2>::Decode(scoped_refptr<media::DecoderBuffer>) (t his=0x318af5a7e520,
buffer=...) at ../../src/media/filters/decoder_stream.cc:438
#25 0x00007fffee965f28 in media::DecoderStream<(media::DemuxerStream::Type)2>::OnBufferReady(media::DemuxerStream::Status,
scoped_refptr<media::DecoderBuffer>) (this=0x318af5a7e520, status=media::DemuxerStream::kOk, buffer=...) at ../../sr
c/media/filters/decoder_stream.cc:791
```

// vertex shader

```
#define TexCoordPrecision highp
attribute TexCoordPrecision vec4 a_position;
uniform mat4 matrix;
varying TexCoordPrecision vec2 v_uvTexCoord;
varying TexCoordPrecision vec2 v_yaTexCoord;
attribute TexCoordPrecision vec2 a_texCoord;
uniform TexCoordPrecision vec2 uvTexOffset;
uniform TexCoordPrecision vec2 uvTexScale;
uniform TexCoordPrecision vec2 yaTexOffset;
uniform TexCoordPrecision vec2 yaTexScale;
void main() {
  // Compute the position.
  vec4 pos = a_position;
  gl_Position = matrix * pos;
  // Compute texture coordinates.
  vec2 texCoord = a_texCoord;
  v_yaTexCoord = texCoord * yaTexScale + yaTexOffset;
  v_uvTexCoord = texCoord * uvTexScale + uvTexOffset;
}
```

// YUV格式的 fragment shader

```
#define LutLookup texture2D
#define TexCoordPrecision mediump
#define SamplerType sampler2D
#define TextureLookup texture2D
precision mediump float;
uniform SamplerType y_texture;
uniform Sampler
Type uv_texture;
```

```
uniform vec4 ya_clamp_rect;
uniform vec4 uv_clamp_rect;
uniform float resource_multiplier;
uniform float resource_offset;
varying TexCoordPrecision vec2 v_yaTexCoord;
varying TexCoordPrecision vec2 v_uvTexCoord;
vec3 DoColorConversion(vec3 color) {
  color = mat3( 1.16438353e+00, 1.16438353e+00, 1.16438353e+00,
                -2.28029018e-09, -2.13248596e-01, 2.11240172e+00,
                1.79274118e+00, -5.32909274e-01, -5.96049432e-10) * color;
  color += vec3(-9.69429970e-01, 3.00019622e-01, -1.12926030e+00);
  return color;
}
uniform float alpha;
void main() {
  // YUV texture lookup and conversion to RGB.
  vec2 ya_clamped = max(ya_clamp_rect.xy, min(ya_clamp_rect.zw, v_yaTexCoord));
  vec2 uv_clamped = max(uv_clamp_rect.xy, min(uv_clamp_rect.zw, v_uvTexCoord));
  vec4 texColor;
  texColor.w = 1.0;
  texColor.x = TextureLookup(y_texture, ya_clamped).x;
  texColor.yz = TextureLookup(uv_texture, uv_clamped).xy;
  texColor.xyz -= vec3(resource_offset);
  texColor.xyz *= resource_multiplier;
  // un-premultiply alpha
  if (texColor.a > 0.0) texColor.rgb /= texColor.a;
  texColor.rgb = DoColorConversion(texColor.xyz);
  texColor.rgb *= texColor.a;
  // Apply alpha from uniform, varying, aa, and mask.
  texColor = texColor * alpha;
  // Write the fragment color
  gl_FragColor = texColor;
}
```

播放本地视频的火焰图：

生成火焰图的命令：

使用bcc中的profile命令生成堆栈：

```
sudo profile -p 16675 -F 199  -U -f 10 > content_shell.log
```

将堆栈生成图形：

```
./flamegraph.pl --countname=cpu --width=1900 --title="conent_shell play local video(199Hz/5ms)"
< content_shell.log > content_shell.svg
```

content_sh
ell