1.主题

awplayer通路播放卡顿分析方法

2.问题背景

产品：多媒体相关的解码播放产品
硬件：VE
软件：多媒体模块+VE驱动

3.问题描述

3.1复现步骤

使用fireplayer、TVvideo等应用走mediaplayer-> awplayer通路（全志多媒体框架）播放视频，出现卡顿问题

3.2具体表现

视频播放卡顿。表现为：播放视频时，出现视频卡顿或者声音卡顿的情况，并且从打印中可以看到大量的 drop this frame 或者reset the timer 的打印

4.问题分析

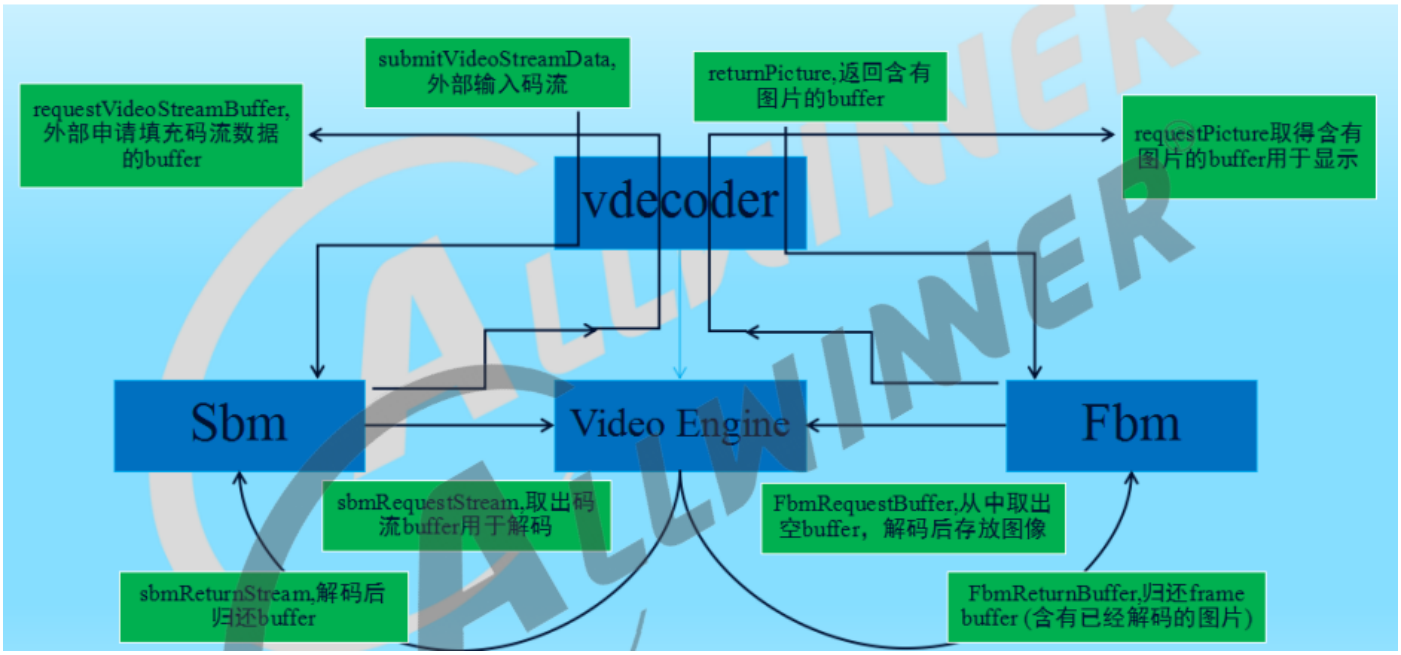
对于播放一个视频来说，造成卡顿的原因有很多，常见的原因有如下几种：
· 播放的视频超过解码器解码性能，也就是超规格了（通过规格书确认即可）
· 网络视频，当网络情况较差时，会影响数据的获取，进而导致卡顿
· 当播放本地视频，根据播放通路来说，分为解封装、解码和显示，由这三个流程，当然，对于网络视频来说，当确定网络不会影响（确定网络影响的方法下文会给出），也可以得出卡顿的原因如下：
· 1. 解封装给的数据过慢，导致卡顿
· 2. 解码耗时严重，导致播放视频卡顿
· 3. 显示耗时，没有及时返回，导致视频播放卡顿

5.根本原因

第一个原因：网络较差导致的卡顿
在线播放就是流媒体，服务器将一连串的媒体数据压缩后，经过分段发送数据，在网上即时传输影

音以供观赏的一种技术与过程，此技术使得数据包得以像流水一样发送。有了该技术，用户可以不必下载整个媒体文件后再观看，而是可以变下载边观看。但是在网络条件较差时，媒体文件播放就会卡顿，体验就会变得很差。那么如何确定此时是由于网络差导致的卡顿呢？
以下介绍一个简易的方法，对于多媒体调试人员来说可以将小机连接上 adb 或者串口之后，通过*** ping baidu.com***, 通过与百度服务器的包交换时间来看，如下所示，耗时越少说明网络情况越好。当网络较差时，time 的耗时会达到几百毫秒，此时建议先保证网络良好，如果网络保证无问题后，视频还卡顿，那么继续往下分析。

解封装解码和显示流程
在这里给出视频播放的整个流程，方便理解。为了保证传输和储存的方便，媒体文件一般都是经过压缩和封装的，那么对于媒体播放来说就需要先进行解封装，将解封装后的数据通过 submitVideoStreamData 接口传输到解码器中，然后解码器进行解码，解码器将解码完成后的 buffer 通过 requestVideoStreamBuffer 接口还给解封装模块使用，解码出来的数据通过requestPicture 接口给到显示，同样，显示模块将已经显示后的 buffer 通过 returnPicture 给到解码器继续使用。当然底层还涉及到 SBM 和 FBM 的问题，原理也是一致的。



第二个原因：解封装给数据慢导致的卡顿
解封装模块将解封装后的音频和视频裸数据分别给到音视频解码后，解码器才能正常解码，如果解封装模块给数据慢了，那么就会导致整个音视频播放卡顿。那么如何确定是否是由于解封装慢了导致的卡顿呢？可以按照以下方法进行确认。

2.1 submit 数据的时间间隔
可以在 submitVideoStreamData 函数中，将每一笔 submit 数据的 pts 打印出来，并且注意每一笔 submit 数据的时间间隔，如果相邻的两笔数据，相差超过理论值的话，那么则表示这笔数据送慢了。举个例子来说，对于帧率为 25fps 的视频来说，那么按照计算，每两笔数据的 pts相差 40ms，也就是说，将如当前帧的 pts 为 1200ms，系统时间为 10:25:400，那么 submit的下一笔数据的 pts 为 1240ms，并且系统时间不超过 40ms，即系统时间不晚于 10:25:440，否则即为 submit 晚了，这时候需要排查解封装是否异常。

2.2 解码器返回 NO BIT STREAM

还有另一种更快捷的办法确定是否是解封装出现异常导致 submit 数据慢，导致视频卡顿。可以按照如下方法，在 vdecoder.c 文件的 DecodeVideoStream（）函数中添加打印，那就是将解码器的返回值打印出来，查看是否是“ 出现大量的 5 ”，5 代表的是 VDECODE_RESULT_NO_BITSTREAM, 如果有大量连续的“ 5 ”打印，那么说明解码器没有收到足够的码流，即可以往解封装模块查看原因，否则继续往下排查。

```
ret = VideoEngineDecode(p->pVideoEngine,
                        bEndOfStream,
                        bDecodeKeyFrameOnly,
                        bDropBFrameIfDelay,
                        nCurrentTimeUs);
logd("the decoder return %d",ret);
```

第三个原因：解码器解码异常导致的卡顿

当通过以上方法判断到视频卡顿原因不是由于与解封装模块导致的问题，那么接下来可以往解码器方向排查。查看解码器解码状态最直接的方法就是通过解码器返回值进行判断，方法上文已经提到，解码器返回值的含义如下所示：

VDECODE_RESULT_UNSUPPORTED = -1,
VDECODE_RESULT_OK = 0,
VDECODE_RESULT_FRAME_DECODED = 1,
VDECODE_RESULT_CONTINUE = 2,
VDECODE_RESULT_KEYFRAME_DECODED = 3,
VDECODE_RESULT_NO_FRAME_BUFFER = 4,
VDECODE_RESULT_NO_BITSTREAM = 5,
VDECODE_RESULT_RESOLUTION_CHANGE = 6,

解码器解码异常时，主要有以下几种情况：

· 解码一帧过于耗时，导致视频播放卡顿。解码耗时过大，这一点可以直接通过打印进行确认，当日志中出现“ 大量 ”如下打印则说明 SBM 获取不到 streambuffer 了，也就说明解码器解码慢了这里主要有两个原因：

W cedarc : <RequestVideoStreamBuffer:1272>: request stream buffer fail,8302286 bytes valid data in SBM[0], total buffer size is 8388608 bytes.

3.1 带宽不足导致的解码器异常卡顿

为了确认是不是带宽导致的解码器解码过慢，那就将带宽给足 VE，可以通过以下方法调高 VE 的带宽优先级，确保 VE 带宽足够，如果带宽足够后，解码速度正常，视频播放不再卡顿，则说明视频卡顿是由于带库不足引起的。

查看VE实时带宽,查看VE的带宽是否是较低
adb shell su
mtop -m

调整带宽优先级，优化VE带宽的方法

adb shell

cd /sys/devices/platform/soc@2900000/3100000.nsi-controller/nsi-pmu/hwmon0

cat port_prio (将各个模块的参数列出，例如VE是13)

echo 13 3 > port_prio （将VE的带宽优先级调整到3， 0是最低）

3.2 解码器解码性能不足

在保证带宽无异常后，可以将 vdecoder.c 文件中的 AW_VDECODER_SPEED_INFO宏打开，通过实时打印进行解码性能确认。主要关注 average 即可。如果确认解码性能不足则需要从解码器方向排查。

player speed: %.2ffps, average: %.2ffps;slowest frame cost time: %.2f ms;hardware speed:%.2ffps, average: %.2ffps

3.3 解码buffer数量不足

解码器返回大量的 4，也就是 VDECODE_RESULT_NO_FRAME_BUFFER，表示解码器没有足够的 FBM buffer 进行解码，这时候需要排查为什么 FBM buffer 轮转不及时，是不是显示模块占用异常，这时候往下继续排查

第四个原因：显示过于耗时导致的卡顿

如果显示模块通过 requestPicture() 获取到一帧待显示的图片进行显示后，需要将显示完成后的buffer 通过 returnPicture() 还给解码器，解码器才能正常获取到 FBM buffer 进行解码。如果显示通路过于耗时，那么就会导致解码器获取不到足够的 buffer 进行解码，这一点即上文提到的解码器返回“ 大量 ”的 4，即表示解码器没有足够的 FBM buffer，这是就需要往显示模块进行排查，确认没有及时返回 FBM buffer 的原因。

6.解决办法

由此，通过以上四个方法便可以定位到视频播放卡顿的原因。