多媒体问题分析指南

版本号 2.0

Amlogic, Inc.
3930 Freedom Circle
Santa Clara, CA 95054
U.S.A.
www.amlogic.com

Legal Notices
© 2015 Amlogic, Inc. All rights reserved. Amlogic ® is registered trademarks of Amlogic, Inc. All other registered trademarks, trademarks and service marks are property of their respective owners.

This document is Amlogic Company confidential and is not intended for any external distribution.

Revision history

Revision	Date	Owner	Changes
1.0	Aug 1st, 2015	Lifeng.Cao	
2.0	Jun 30th, 2017	Lifeng.Cao	

1、本地播放问题分析

1.1 播放器闪退

1.1.1 Libplayer

大部分情况是播放器的问题起来，这种情况主要是通过 logcat -s AmSuperPlayer ;logcat -s AmlogicPlayer; logcat -s amplayer 来查看 LOG，看是否有错误信息。如果这些打印都没有出现，那可能是 APK 的问题，或者系统的其他问题，需要具体分析。

1.1.2 Nuplayer

从 Android 7.1 以后的版本，大部分的格式本地文件已经都走 NuPlayer 来播放了。对于 Nuplayer 播放主要用 Logcat -s NU-AmNuPlayer NU-NuPlayerRenderer MediaCodec ACodec AmlogicVideoDecoderAwesome 。可以看到一些错误信息。

先确定解码组件是否加载对，如果发现类似如下打印：

W/ACodec (6158): [OMX.amlogic.avc.decoder.awesome]

说明走到我们的硬解组件里面来了，如果[]里面没有 amlogic 字样，只有类似 google 的字样，这个很可能走到了安卓原生的软解组件里面了。

如果发现走错组件了，先看看 SDK 中 device/amlogic/项目名/files/media_codecs.xml 里面是否有配置我们的硬解组件。

如果发现错误信息带有 AmlogicVideoDecoderAwesome，也走到了我们的硬解组件，可以通过打开 OMX 硬解组件的打印，来获取更加详细的信息。

setprop media.omx.log_levels 255

logcat -s AmlogicVideoDecoderAwesome

这样 OMX 硬解组件的打印就能打出来了。

1.2 有声音和无视频

1.2.1 无视频或者视频不支持

这种情况可以用 PC 软件 mediainfo 看一下片源里面的流的信息。确认文件里面是否有视频流。确定视频流的格式，从而来确认我们播放器是否支持。这种情况下遵照 1.1 里面抓一些 LOG 应该可以看到错误或者不支持的信息。

1.2.2 显示链路阻塞

首先确认显示链路：（通过 cat /sys/class/vfm/map 可以查看 default 关键字来查看显示链路的设置情况）

目前本地播放大致分成如下两种 case:

a、Libplayer

video layer 显示：

default { decoder(1) ppmgr(1) deinterlace(1) amvideo}

osd layer 显示：

default { decoder(1) ionvideo}

b、Nuplayer

video layer 显示：

default { decoder(1) amlvideo(1) deinterlace(1) amvideo}

osd layer 显示：

default { decoder(1) ionvideo}

如何确定是显示链路出问题呢？切换显示链路。如果随着显示链路的切换，就出图像了。这种大部分是显示链路的问题。

a、Libplayer

默认走 video layer 输出，通过 setprop media.amplayer.v4osd.all 1 可以设置成 osd 输出。

b、Nuplayer

强制走 video layer 显示 setprop media.omx.display_mode 1

强制走 OSD 显示 setprop media.omx.display_mode 0

在这种情况下如何确认是哪个后处理模块的问题呢？

a、查看 vfm status:

链路一：default { decoder(1) ppmgr(1) deinterlace(1) amvideo}

cat /sys/class/ppmgr/ppmgr_vframe_states

cat /sys/class/deinterlace/di0/provider_vframe_status

cat /sys/class/video/vframe_states

链路二：default { decoder(1) ionvideo}

cat /sys/class/ionvideo/vframe_states

链路三：default { decoder(1) amlvideo(1) deinterlace(1) amvideo}

cat /sys/class/deinterlace/di0/provider_vframe_status

cat /sys/class/video/vframe_states

这个是后处理的三个模块的状态。

vframe_pool_size=64

vframe buf_free_num=54

vframe buf_recycle_num=0

vframe buf_avail_num=10

如果哪个模块的 vframe buf_avail_num 一直是 0，那么可能它链路上前面一个成员模块阻塞住了。

但是链路二和三有些特别，如果发现

链路二时 cat /sys/class/ionvideo/vframe_states vframe buf_avail_num 一直不动，而且是一个比较大的值。或者是

链路三时 cat /sys/class/deinterlace/di0/provider_vframe_status 一直没有 buffer，这个很大可能是上层 OMX 没有来

取数据。打开 OMX 打印的方法请参照 1.1.2。

OMX 有读取解码后数据的打印如下：

```
05-12 02:23:31.540  3987  6186 D AmlogicVideoDecoderAwesome:            Out PTS: 8120000..vf.fd=39, mOutBufferNo=126
05-12 02:23:31.541  3987  6186 D AmlogicVideoDecoderAwesome:            Out PTS: 8160000..vf.fd=31, mOutBufferNo=127
```

如果没有这个打印就可能是上层没有来取数据。就要往 ACodec 甚至更上层来查。

b、查看试着关闭某些模块：
最常见的就是 bypass di 模块:
echo 1 > /sys/module/di/parameters/bypass_all

1.2.3 视频数据不够

现在驱动分为单路解码驱动和多路解码驱动。
cat /sys/class/vfm/map
单路解码驱动：
default { decoder(1) amlvideo(1) deinterlace(1) amvideo}
多路解码驱动：
vdec-map-0 { vdec.h264.00(1) amlvideo(1) deinterlace(1) amvideo}

如果是单路解码驱动，查看视频数据通过如下命令：
cat /sys/class/amstream/bufs
Video buffer: flag:7(Alloc Used Parser nofirststamp)
buf addr:0000000020c00000
buf size:0x2800000
buf canusesize:0x2800000
buf regbase:0xc40
buf level:0x6a
buf space:0x27fe796
buf read pointer:0x215a4800
buf first_stamp:0xffffffff
buf wcnt:0x153a5
buf max_buffer_delay_ms:0ms
buf current delay:160ms
buf bitrate latest:85140bps,avg:1545405bps
buf time after last pts:15070 ms
buf time after last write data :15070 ms
如果 buf level 为 0x0，就代表没有视频数据，可能是 parser 或者是 demux 的问题。
如果是多路解码驱动，查看视频数据通过如下命令：
cat /sys/class/vdec/dump_vdec_chunks
blocks:vdec-0 id:7,bufsize=3670016,dsize=1919012,frames:34,maxframe:155956
[111:ffffffc052fc1c80]-off=0,size:50849,p:95, pts64=4629625,addr=000000005d5c0000
[112:ffffffc052fc1c80]-off=50944,size:137272,p:72, pts64=4796458,addr=000000005d5cc700
[113:ffffffc052fc1c80]-off=188288,size:53290,p:86, pts64=4713041,addr=000000005d5edf80
[114:ffffffc052fc1c80]-off=241664,size:55809,p:127, pts64=4754750,addr=000000005d5fb000
[115:ffffffc052fc1c80]-off=297600,size:133457,p:111, pts64=4921583,addr=000000005d608a80
[116:ffffffc052fc1c80]-off=431168,size:44389,p:91, pts64=4838166,addr=000000005d629440
[117:ffffffc052fc1c80]-off=475648,size:48062,p:66, pts64=4879875,addr=000000005d634200
[118:ffffffc04c414780]-off=0,size:117931,p:85, pts64=5046708,addr=000000005d640000
[119:ffffffc04c414780]-off=118016,size:40740,p:92, pts64=4963291,addr=000000005d65cd00
[120:ffffffc04c414780]-off=158848,size:52214,p:74, pts64=5005000,addr=000000005d666c80

[121:ffffffc04c414780]-off=211136,size:91449,p:71, pts64=5171833,addr=000000005d6738c0
[122:ffffffc04c414780]-off=302656,size:38191,p:81, pts64=5088416,addr=000000005d689e40
[123:ffffffc04c414780]-off=340928,size:44158,p:66, pts64=5130125,addr=000000005d6933c0
[124:ffffffc04c414780]-off=385152,size:81319,p:89, pts64=5296958,addr=000000005d69e080
[125:ffffffc04c414780]-off=466560,size:32798,p:98, pts64=5213541,addr=000000005d6b1e80
[126:ffffffc0513aff80]-off=0,size:38240,p:96, pts64=5255250,addr=000000005ad00000
这个标红的 frames 就是存储的帧数。

如果是走 Nuplayer 发现数据量很小可以通过打开 OMX 打印来看上层送数据的情况：
打开 OMX 打印的方法请参照 1.1.2。，表明有上层有在送数据，如若没有这些打印，表明没有送数据。

```
05-12 02:23:30.864  3987  6187 D AmlogicVideoDecoderAwesome: input emptebufferdone pBufferHdr->pBuffer=f0d02000, mInPutFrameCount=442
05-12 02:23:30.865  3987  6187 D AmlogicVideoDecoderAwesome: In PTS 9740000,    nFilledLen=4489, bufferHdr.pBuffer=f0e02000
05-12 02:23:30.865  3987  6187 D AmlogicVideoDecoderAwesome: input emptebufferdone pBufferHdr->pBuffer=f0e02000, mInPutFrameCount=443
05-12 02:23:30.872  3987  6187 D AmlogicVideoDecoderAwesome: In PTS 9920000,    nFilledLen=8967, bufferHdr.pBuffer=f0f02000
```

1.2.4 osd 遮挡视频

echo 1 > /sys/class/graphics/fb0/blank
echo 1 > /sys/class/graphics/fb1/blank
如果输入这两个命令之后，视频显示出来了，就说明视频被 OSD 遮住了。
可以再播放时输入如下命令：
#dumpsys SurfaceFlinger

type | handle | hint | flag | tr | blnd | format | source crop (l,t,r,b) | frame | name
----------+----------+------+------+----+------+-------------+--------------------------------+------------------------+------
HWC | f653cd80 | 0002 | 0000 | 00 | 0100 | RGBA_8888 | 0.0, 0.0, 1920.0, 1080.0 | 0, 0, 1920, 1080 | SurfaceView
GLES | f5c3c1f0 | 0000 | 0000 | 00 | 0105 | RGBA_8888 | 0.0, 0.0, 1920.0, 985.0 | 0, 0, 1920, 985 | com.droidlogic.videoplayer/com.droidlogic.videoplayer.VideoPlayer
FB TARGET | f65219c0 | 0000 | 0000 | 00 | 0105 | RGBA_8888 | 0.0, 0.0, 1920.0, 1080.0 | 0, 0, 1920, 1080 | HWC_FRAMEBUFFER_TARGET
查看有没有 SurfaceView，并且看一下 hint 是不是为 0002。如果没有或者 hint 不是 0002，说明 APK 没有创建 videoview。

1.2.5 videolayer 关闭

#cat /sys/class/video/disable_video
a、video layer 被关闭：
返回“1”，需要在代码中查找设置/sys/class/video/disable_video 为 1 的地方。
b、没有视频帧需要输出：
返回“2”
有可能和同步相关：
#echo 0 > /sys/class/tsync/enable

如果可以正常播放，就属于这种情况。
有可能和 deinterlace 或者 ppmgr 相关，可以参见 1.2.2 来确认这种情况。

1.2.6 解码错误

这个一般都会有 decoder error 的打印，查看 kernel 打印就可以。
cat /sys/class/amstream/bufs
Video buffer: flag:7(Alloc Used Parser nofirststamp)
buf addr:0000000020c00000
buf size:0x2800000
buf canusesize:0x2800000
buf regbase:0xc40
buf level:0x17fe796
buf space:0x27fe796
buf read pointer:0x215a4800
buf first_stamp:0xffffffff
buf wcnt:0x153a5
buf max_buffer_delay_ms:0ms
buf current delay:160ms
buf bitrate latest:85140bps,avg:1545405bps
buf time after last pts:15070 ms
看 video buffer 如果 buf level 很高，而且不变化，排除 di、ppmrg 的问题之后就应该是解码的问题。
需要抓取 cat /sys/class/vdec/amrisc_regs 信息。来确定 decoder 出现了什么问题。

从 Android 7.1 的 SDK 之后，添加了多路解码的驱动。（查看是不是多路解码可以参照 1.2.3）
cat /sys/class/vdec/dump_vdec_chunks
blocks:vdec-0 id:7,bufsize=3670016,dsize=1919012,frames:34,maxframe:155956
[111:ffffffc052fc1c80]-off=0,size:50849,p:95, pts64=4629625,addr=000000005d5c0000
[112:ffffffc052fc1c80]-off=50944,size:137272,p:72, pts64=4796458,addr=000000005d5cc700
[113:ffffffc052fc1c80]-off=188288,size:53290,p:86, pts64=4713041,addr=000000005d5edf80
[114:ffffffc052fc1c80]-off=241664,size:55809,p:127, pts64=4754750,addr=000000005d5fb000
[115:ffffffc052fc1c80]-off=297600,size:133457,p:111, pts64=4921583,addr=000000005d608a80
[116:ffffffc052fc1c80]-off=431168,size:44389,p:91, pts64=4838166,addr=000000005d629440
[117:ffffffc052fc1c80]-off=475648,size:48062,p:66, pts64=4879875,addr=000000005d634200
[118:ffffffc04c414780]-off=0,size:117931,p:85, pts64=5046708,addr=000000005d640000

如果里面帧一直没有变化，那就可能是解码有问题。这个时候可以试着换成单路解码驱动看还是不是有问题，切换的命令如下：
setprop media.omx.single_mode 1

1.2.7 OSD 上显示 Render 问题

Android 6.0 及以下平台：
dumpsys SurfaceFlinger --latency SurfaceView

Android 7.1 及以上平台：

dumpsys SurfaceFlinger --latency "SurfaceView - com.android.gallery3d/com.android.gallery3d.app.MovieActivity"

（这个 SurfaceView - 跟着的是正在播放视频的 activity 的名字）

会出现类似的打印，中间这一列标示 surfaceview 的刷新到 frame buffer 的实际时间。

如果中间这一列一直为 0，或者一直保持不变。那就是说没有 frame 没有被刷新到 frame buffer 上去。

16666666

0 8493743536000 0
0 8493760355000 0
0 8493776867000 0
0 8493793527000 0
0 8493810194000 0
0 8493826868000 0
0 8493843532000 0
0 8493860188000 0
0 8493910228000 0
0 8493926875000 0
0 8493943537000 0
0 8493976867000 0
0 8493993531000 0
0 8494010219000 0
0 8494026876000 0
0 8494043559000 0
0 8494076941000 0
0 8494093579000 0
0 8494110209000 0
0 8494160210000 0

1.3 视频位置问题：

目前一般只有在 video layer 上显示的时候才会碰到有视频位置问题。

#cat /sys/class/video/axis （x0,y0,x1,y1）

#cat /sys/class/video/screen_mode（0：按片源比例显示；1：全屏；2:4 比 3；3:16 比 9）

#cat /sys/class/video/crop 视频的剪切范围

1.3.1 SurfaceView 的位置

#dumpsys SurfaceFlinger

type | handle | hint | flag | tr | blnd | format | source crop (l,t,r,b) | frame | name

----------+----------+------+------+----+------+------------+--------------------------------+----------------------+------

HWC | f653cd80 | 0002 | 0000 | 00 | 0100 | RGBA_8888 | 0.0, 0.0, 1920.0, 1080.0 | 0, 0, 1920, 1080 | SurfaceView

GLES | f5c3c1f0 | 0000 | 0000 | 00 | 0105 | RGBA_8888 | 0.0, 0.0, 1920.0, 985.0 | 0, 0, 1920,

985 | com.droidlogic.videoplayer/com.droidlogic.videoplayer.VideoPlayer
FB TARGET | f65219c0 | 0000 | 0000 | 00 | 0105 | RGBA_8888 | 0.0, 0.0, 1920.0, 1080.0 | 0, 0, 1920, 1080 | HWC_FRAMEBUFFER_TARGET
标红的为 video view 的位置

1.3.2 窗口位置换算

如果 video view 的位置正确，而/sys/class/video/axis 的值不对，可能是换算出现问题。

```
logcat -s amavutils
E/amavutils( 2785): unable to open file /sys/class/graphics/fb2/clone,err: No such file or directory
I/amavutils( 2785): amvideo_utils_set_virtual_position :: x=0 y=0 w=1921 h=1081
I/amavutils( 2785): device resolution 1280x720
I/amavutils( 2785): disp resolution 1920x3240
I/amavutils( 2785): amvideo_utils_set_virtual_position:: disp_w=1920, disp_h=1080
I/amavutils( 2785): video global_offset 0
I/amavutils( 2785): set ppmgr angle :0
I/amavutils( 2785): /sys/class/graphics/fb0/free_scale_axis axis: 0 0 1919 1079
I/amavutils( 2785): after scaled, screen position3: 0 0 1280 720
I/amavutils( 2785): amvideo_utils_set_virtual_position (corrected):: x=0 y=0 w=1280 h=720
I/amavutils( 2785): /sys/class/graphics/fb0/free_scale_axis axis: 0 0 1919 1079
D/amavutils( 2785): amvideo_setscreenmode as 1.778499
```

蓝色字体为 video view 的位置，红色字体为换算之后的 video axis 的位置。

1.4 音视频卡顿

1.4.1 音视频同时卡顿

如果音视频同时卡顿，可能与如下原因：

a、 buffer 下溢

这五项分别代表 video、audio、subtitle、user data、h265video buffer 的情况，如果 buf level 比较低，表示 buffer 下溢。有可能是 player 送数据不够。

```
cat /sys/class/amstream/bufs
Video buffer:     flag:7( Alloc Used Parser nofirststamp )
        buf addr:0000000020c00000
        buf size:0x2800000
        buf canusesize:0x2800000
        buf regbase:0xc40
        buf level:0x65556a
        buf space:0x21a9296
        buf read pointer:0x20f4f300
        buf first_stamp:0xffffffff
        buf wcnt:0x1a227
```

buf max_buffer_delay_ms:0ms
buf current delay:10520ms
buf bitrate latest:85140bps,avg:1545405bps
buf time after last pts:4060 ms
buf time after last write data :4060 ms
Audio buffer: flag:7(Alloc Used Parser nofirststamp)
buf addr:0000000027400000
buf size:0x180000
buf canusesize:0x180000
buf regbase:0x1584
buf level:0x275aa
buf space:0x157256
buf read pointer:0x27413e80
buf first_stamp:0xffffffff
buf wcnt:0x700
buf max_buffer_delay_ms:0ms
buf current delay:10496ms
buf bitrate latest:43680bps,avg:76144bps
buf time after last pts:4060 ms
buf time after last write data :4060 ms
Subtitle buffer: flag:0(Unalloc Noused noParser nofirststamp)
buf addr: (null)
buf size:0x40000
buf canusesize:0x40000
buf start:0x0
buf write pointer:0x0
buf read pointer:0x0
buf level:0x0
buf first_stamp:0xffffffff
buf wcnt:0x0
buf max_buffer_delay_ms:0ms
UserData buffer: flag:0(Unalloc Noused noParser nofirststamp)
buf addr: (null)
buf size:0x0
buf canusesize:0x0
buf regbase:0x0
buf no used.
buf write pointer:0x0
buf read pointer:0x0
buf first_stamp:0xffffffff
buf wcnt:0x0
buf max_buffer_delay_ms:0ms
HEVC buffer: flag:0(Unalloc Noused noParser nofirststamp)
buf addr: (null)
buf size:0x2800000
buf canusesize:0x2800000

buf regbase:0x3102
buf no used.
buf first_stamp:0xffffffff
buf wcnt:0x0
buf max_buffer_delay_ms:0ms

增加多路解码驱动之后，如果是走的多路解码驱动，而且用的是 frame base 模式（stream base 还是看 /sys/class/amstream/bufs）
则用如下命令看 buffer level
cat /sys/class/vdec/dump_vdec_chunks
blocks:vdec-0 id:7,bufsize=3670016,dsize=1919012,frames:34,maxframe:155956
[111:ffffffc052fc1c80]-off=0,size:50849,p:95, pts64=4629625,addr=000000005d5c0000
[112:ffffffc052fc1c80]-off=50944,size:137272,p:72, pts64=4796458,addr=000000005d5cc700
[113:ffffffc052fc1c80]-off=188288,size:53290,p:86, pts64=4713041,addr=000000005d5edf80
[114:ffffffc052fc1c80]-off=241664,size:55809,p:127, pts64=4754750,addr=000000005d5fb000
[115:ffffffc052fc1c80]-off=297600,size:133457,p:111, pts64=4921583,addr=000000005d608a80
[116:ffffffc052fc1c80]-off=431168,size:44389,p:91, pts64=4838166,addr=000000005d629440
[117:ffffffc052fc1c80]-off=475648,size:48062,p:66, pts64=4879875,addr=000000005d634200
[118:ffffffc04c414780]-off=0,size:117931,p:85, pts64=5046708,addr=000000005d640000
这个 frames 一般大概存两秒左右的数据，大致帧率*2 这么多帧。如果非常小，只有几个 frame 就有可能会下溢。

buffer 下溢大部分是送数据的速度比较慢，对于 OMX 而言，这个可以通过打开 OMX 打印（见 1.2.3），看一下 input 的速度是不是比较慢。一般要稍微大于帧率才能收支平衡。对于 libplayer 可以查看用如下命令：
echo 1 > /sys/class/tsync/debug_pts_checkin 查看 checkin 的 pts
echo 1 > /sys/class/tsync/debug_video_pts 查看 video pts
然后查看 pts checkin 的情况，看是不是 checkin 比较慢，如果是，需要查 codec write 为什么会这么慢。
还有一种特例就是 video buffer 很满，audio buffer 很空，导致 audio pts reset pcr。然后造成卡顿。这种情况下需要加大 video buffer 来试试，也可以在写数据的时候先写多一些 audio 包，然后再来写 video 包。

b、同步相关卡顿

Libplayer:

一般同步引起卡顿可以有如下方法可以确认。
查看同步模式，关闭同步，或者关闭声音，如果此时视频没有卡顿了，说明是同步引起的卡顿。
cat /sys/class/tsync/mode 查看同步模式
echo 0 > /sys/class/tsync/enable 关闭同步
setprop media.amplayer.noaudio true disable audio

或者是设置 free run 模式：
echo 1 > /sys/class/video/freerun_mode

还有可能是 pts 有跳变等情况，这个时候就需要打开 pts 的打印。
查看 checkin checkou pts
echo 1 > /sys/class/tsync/debug_audio_pts 查看 audio pts

echo 1 > /sys/class/tsync/debug_pts_checkin　　查看 checkin 的 pts
echo 1 > /sys/class/tsync/debug_pts_checkout　查看 checkout 的 pts
echo 1 > /sys/class/tsync/debug_video_pts　　查看 video pts

Nuplayer

同步是 Nuplayer 里面在做，所以需要查看 Nuplayer 的打印。
logcat | grep NU 可以看到 Nuplayer 的打印。比较关键的同步信息如下：
29448 I NU-NuPlayerRenderer: PTS: AV sync info:AV SYNCED
29448 I NU-NuPlayerRenderer: PTS:　SystemTimeStamp:9438303
29448 I NU-NuPlayerRenderer: PTS:　Last AudioTimeStamp:9891700
29448 I NU-NuPlayerRenderer: PTS:　Last VideoTimeStamp:9440000
29448 I NU-NuPlayerRenderer: PTS:　AV diff:-1697 ,Ajump:0,Vjump:0
29448 I NU-NuPlayerRenderer: PTS:　mAjumpedNum:0, mAudioJumped till now=86701582067
29448 I NU-NuPlayerRenderer: PTS:　mVjumpedNum:0, mVideoJumped till now=86701582067
29448 I NU-NuPlayerRenderer: PTS:　mTotalAudioJumpedTimeUs:0
29448 I NU-NuPlayerRenderer: PTS:　mLastVideoUs:9520000
29448 I NU-NuPlayerRenderer: PTS:　mLastAudioUs:9961360
这就是 Nuplayer 的同步信息。
如果发现打印中出现“video late by”这可能就是由于同步引起的卡顿的情况。
关闭 audio 或者 video 来看看是否不卡顿了：
setprop media.amplayer.novideo 1　nuplayer 本地播放关闭 音频
setprop media.amplayer.noaudio 1　nuplayer 本地播放关闭 视频
setprop media.amplayer.nosubtitle 1　nuplayer 本地播放关闭字幕

1.4.2 只有视频卡

a、视频解码出错

这种情况只需要查看 kernel 打印即可，如果解码出错会有错误打印。

b、后处理相关卡顿

这类情况一般是走 video layer 显示才会出现这种情况：

如果上述情况排除之后，还有可能是后处理模块或者是带宽造成的。
关闭 DI：
echo 1 > /sys/module/di/parameters/bypass_all
或者
echo rm default > /sys/class/vfm/map
echo add default decoder ppmgr amvideo > /sys/class/vfm/map
DI 关闭后如果不卡顿，也有可能不是模块本身的问题，有可能是带宽问题。此时需要按照 1.4.5 来确认。

关闭 PPMGR：
echo rm default > /sys/class/vfm/map
echo add default decoder deinterlace amvideo > /sys/class/vfm/map

通过 Underflow 查看：

cat /sys/module/amvideo/parameters/underflow
cat /sys/class/video/freerun_mode
卡顿时看这个值是否一直增长,如果是则是 amvideo 前边的节点数据送过来的慢导致的,

c、视频解码慢

由于 nuplayer 只会送 video es 数据到 stream buffer 里面来，所以不存在因为 video buffer 慢而造成 audo buffer 满而引起卡顿。所以此时卡顿可能有两种情况，一种是 video 数据送得慢，一种是解码慢。
确认是不是数据送得慢，可以根据（1.4.1-a）来看 buffer 状态，看 buffer 是否为空。
如果确认是不是解码慢，可以根据如下命令：

echo 1 > /sys/class/tsync/debug_pts_checkout 查看 checkout 的 pts
echo 1 > /sys/class/tsync/debug_video_pts 查看 video pts

关闭 printk，在播放的时候出现卡顿时，dmesg。然后统计一下每秒 checkout pts 的次数，如果发现这个数据一直小于视频的帧率。就说明视频解码慢。这个可以修改 dpb buffer，vdec 的频率等手段来改善。

d、效能不足引起卡顿

cpu 效能不够，可以提高 CPU 频率：
cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_cur_freq
echo performance > scaling_governor

DDR 带宽问题：
1.vdec urgent 补丁
2.skip B 补丁
3.抽线：echo 1 > /sys/module/amvideo/parameters/force_vskip_cnt
抽掉一半
4.MALI 降核:
echo 2 > /sys/class/mpgpu/scale_mode
chmod 600 /sys/class/mpgpu/scale_mode
echo PP_NUM > /sys/kernel/debug/mali/pp/num_cores_enabled
PP_NUM = 1,2,3

查看带宽信息可以找平台组要带宽分析工具，可以查看各个 channel 的带宽占用情况。

1.5 音视频不同步

Libplayer:
如果出现音视频不同步，先确认如下状态：
同步是否打开：cat /sys/class/tsync/enable
同步的状态：cat /sys/class/tsync/mode
0: vmaster //视频为准，一般这个时候表示音视频不做同步；
1：amaster //以音频作为同步基准；
2：pcrmaster //以独立的系统时间为基准；

查看 APTS、PCR、VPTS：

cd /sys/class/tsync/
cat pts_audio pts_pcrscr pts_video
pts 的单位是 1/90000，也就是，数值 90000 表示一秒，一般比较明显的看到不同步在 100ms 左右，也就 9000；如果 pts 不对就查看 checkin checkout pts。

Nuplayer:

logcat | grep NU 可以看到 Nuplayer 的打印。比较关键的同步信息如下：

29448 I NU-NuPlayerRenderer: PTS: AV sync info:AV SYNCED
29448 I NU-NuPlayerRenderer: PTS: SystemTimeStamp:9438303
29448 I NU-NuPlayerRenderer: PTS: Last AudioTimeStamp:9891700
29448 I NU-NuPlayerRenderer: PTS: Last VideoTimeStamp:9440000
29448 I NU-NuPlayerRenderer: PTS: AV diff:-1697 ,Ajump:0,Vjump:0
29448 I NU-NuPlayerRenderer: PTS: mAjumpedNum:0, mAudioJumped till now=86701582067
29448 I NU-NuPlayerRenderer: PTS: mVjumpedNum:0, mVideoJumped till now=86701582067
29448 I NU-NuPlayerRenderer: PTS: mTotalAudioJumpedTimeUs:0
29448 I NU-NuPlayerRenderer: PTS: mLastVideoUs:9520000
29448 I NU-NuPlayerRenderer: PTS: mLastAudioUs:9961360

1.5.1 Libplayer 不同步分析：

音频比视频早：
setprop sys.amplayer.drop_pcm 1
看是否还有不同步出现。
音频比视频晚：
a、查看 checkin checkou pts

echo 1 > /sys/class/tsync/debug_audio_pts 查看 audio pts
echo 1 > /sys/class/tsync/debug_pts_checkin 查看 checkin 的 pts
echo 1 > /sys/class/tsync/debug_pts_checkout 查看 checkout 的 pts
echo 1 > /sys/class/tsync/debug_video_pts 查看 video pts

b、确认是 audio 还是 video 的问题

echo 0 > /sys/class/tsync/enable
对比在 PC 上播放该视频，确认是 audio 还是 video 播放的问题。
如果是 audio pts 有问题，一般是 pts 计算错误，这个一般出现在音频那边；以为音频 pts 计算复杂；

c、设置 freerun_mode

echo 1 > /sys/class/video/freerun_mode
看看是否还会不同步。

d、确认音频采样率

音频流畅，视频时快时卡住：
这种情况可能是 vsync 里面算 pcr 有问题。
cat /sys/class/video/frame_rate
VF.fps=0.00 panel fps 50, dur/is: 93,v/s=49.46,inc=1800
panel fps 为输出的刷新率。

v/s 为实时刷新率。

一直流畅播放，但是不同步：

设置控制同步的阀值

最小不同步时间：RW: /sys/class/tsync/av_threshold_min

最大不同步时间：RW:/sys/class/tsync/av_threshold_max

当音视频 pts 差距跳入：<min 区间： ?音频或者视频会暂停或者加速同步上；

当音视频 pts 差距>min,Max 是，音视频会暂时不强制同步，等待超时 60 秒后强制同步；?如果视频和音频差距>max, 同步系统放弃同步处理；

1.5.2 Nuplayer 不同步分析：

a、查看每一帧的 pts

setprop media.hls.ptsdebug 4 dump pts 能显示音视频每帧的 pts

b、走 video layer 显示的时候，可以调整 omx_pts 纠 pcr 的阈值

echo xxx > /sys/module/amvideo/parameters/omx_pts_interval_lower

echo xxx > /sys/module/amvideo/parameters/omx_pts_interval_upper

把阈值调小，可以让 omx_pts 和 pcr 很小的差异就调整，看是否能纠正不同步。

c、打开 exffmpeg 的打印

setprop media.ffmpegextractor.loglevel

d、起播慢同步开关

setprop media.hls.slowsync 1

1.6 视频抖动、闪烁、锯齿

视频抖动、闪烁和锯齿大部分和 Deinterlace，RDMA 和 VPP 设置相关。

1.6.1 Deinterlace

a、确认 Deinterlace 模块是否在 vfm/map 链里

cat /sys/class/vfm/map

default { decoder(1) ppmgr(1) deinterlace(1) amvideo}

如果 default 里面没有 deinterlace，执行如下指令添加 deinterlace 模块。

echo rm default > /sys/class/vfm/map

echo add default decoder ppmgr deinterlace amvideo > /sys/class/vfm/map

b、确认 Deinterlace 模块是否被 bypass

播放节目过程中：cat /sys/module/di/parameters/bypass_state

如果为 1 则被 bypass

那要根据片源的情况，然后查看

cat /sys/module/di/parameters/

bypass_1080p	1080p 片源 bypass
bypass_3d	3D 片源是否 bypass
bypass_all	强制所有的都 bypass
bypass_dynamic	是否动态 bypass
bypass_hd	高清片源 bypass
bypass_hd_prog	高清逐行片源 bypass
bypass_interlace_output	I 模式输出的时候所有片源 bypass
bypass_prog	逐行片源 bypass

看是哪个结点造成 di bypass 的。把它设成 0 之后，然后再播放，看是否还抖动。

c、开关 nr

echo 0/1 >/sys/module/di/parameters/nr2_en

d、查看 Deinterlace 模块工作状态

echo state > /sys/class/deinterlace/di0/debug

e、确认送到 DI 模块的帧是否是 top/bottom 均匀穿插

cat /sys/module/di/parameters/same_field_bot_count

cat /sys/module/di/parameters/same_field_top_count

如果这两个值有一个一直在增加就说明前边解码器解码有问题，此时会出现锯齿。

f、小窗口视频闪烁

cat /sys/module/di/parameters/di_vscale_skip_count_real

是否真实在小窗口情况下

cat /sys/module/di/parameters/di_vscale_skip_count

是否是在小窗口情况下

cat /sys/module/di/parameters/di_vscale_skip_enable

是否 enable 小窗口时 bypass di

1.6.2 VPP 设置

在小窗口出现花屏或者闪烁的时候，还有可能是因为带宽问题。我们现在解决的方式就是在小窗口播放的时候，先抽掉一部分视频帧的线，然后再送到 VPP 去显示。

通过如下命令可以看一下目前是否有抽线处理：

cat /sys/class/video/video_state

zoom_start_x_lines:0.zoom_end_x_lines:1279.

zoom_start_y_lines:0.zoom_end_y_lines:719.

frame parameters: pic_in_height 720. //标示抽线后的高度

frame parameters: VPP_line_in_length_ 1280.

vscale_skip_count 0. //标示抽线条数

hscale_skip_count 0.

hscale phase step 0x1000000.
vscale phase step 0x1000000.
根据这些状态信息可以确定输入的大小,是否抽线,是否在做 scale,是否打开 3d 等,一般花屏等问题需要这些信息。
如果发现窗口很小，而没有抽线，可以修改如下结点：
cat /sys/module/amvideo/parameters/bypass_ratio
Vpp 的处理能力会根据这个值算出来一个结果,根据结果来判断是否抽线,越大越不容易抽线,越小越容易抽线。

Dump vpp 相关寄存器：

```
echo 0xd0106800    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0106900    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0106a00    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0106b00    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0107400    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0107500    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0107600    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0107700    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0105c00    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0105d00    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0105e00    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
echo 0xd0105f00    64 >    /sys/kernel/debug/aml_reg/dump;
cat /sys/kernel/debug/aml_reg/dump
```

1.6.3 RDMA

cat /sys/module/rdma/parameters/enable
如果为 0，则为关闭，echo 1 > /sys/module/rdma/parameters/enable 看是否还抖动。

2、网络流播放问题分析

2.1 Dump 音视频流

2.1.1 APK 用 MediaPlayer 接口实现：

网络流遇到各种问题，首先需要 Dump 流，确认在本地能否复现问题。Dump 方法如下：
播放在线视频 dump 数据
1.创建 dump 文件夹 mkdir /data/tmp
2.修改 temp 文件夹属性 chmod 777 /data/tmp
3.设置 dump 数据类型 setprop media.libplayer.dumpmode x
x 的值:
1 ts,ps,rm 几种流的 raw data 从文件或网络读到的数据 dump
2 ts,ps,rm 几种流的 raw data 写入解码 buffer 的数据 dump
4 es 码流 video 数据，读数据 dump
8 es 码流 video 数据，写数据 dump
16 es 码流 audio 数据，读数据 dump
32 es 码流 audio 数据，写数据 dump
dump 到的数据存放在/data/tmp/pidn_dump_xx.dat 以 pidn 开头，可以区分是哪个线程 dump 出来的数据。
播放 ts 如果开 softdemux 后 dump 不到 ts 流，需要按 es 流 dump

2.1.2 APK 用 MediaCodec 或者 OMX 接口实现：

如果播放 APK 用的是 MediaCodec 或者 OMX 接口来实现的，那只能 dump 到 ES 数据，dump 接口如下：

media.omx.dumpRecv	是否 dump Omx codec 接收到的数据
media.omx.dumpCodec	是否 dump Omx codec 往解码 buffer 里面写的数据
media.omx.RecvDir	设置 dump Omx codec 接收到的数据写到哪个目录
media.omx.CodecDir	设置 dump Omx codec 往解码 buffer 数据写到哪个目录
media.omx.RecvName	设置 dump Omx codec 接收到的数据写到哪个文件里
media.omx.CodecName	设置 dump Omx codec 往解码 buffer 数据写到哪个文件里

例如：如果需要 dump Omx Codec 接收到的数据到/data/tmp/es_aml_recv.0 同时还要 dump Omx Codec 往解码 buffer 数据到/data/tmp/es_aml_codec.0，对应的设置如下：

```
setprop media.omx.dumpRecv true
setprop media.omx.dumpCodec true
setprop media.omx.RecvDir /data/tmp
setprop media.omx.CodecDir /data/tmp
setprop media.omx.RecvName es_aml_recv.0
setprop media.omx.CodecName es_aml_codec.0
```

如果目录不存在记得创建，然后修改目录权限为 777。

2.1.3 APK 用 Nuplayer 接口 dump 码流：

HLS 数据
在音视频分离的情况下用如下命令，会分别 dump audio video sub 的数据。
setprop media.hls.dumpmode 1

如果音视频数据是同一个流，用如下命令：
setprop media.hls.dumpmode 2

dump 到的数据在/data/tmp/目录下，记得如果目录不存在，创建一下，然后修改目录权限为 777。
UDP 数据：
setprop media.udp.dump 1

2.2 抓取网络包

其次还可以通过抓包来分析数据：
可以通过 tcpdump 命令抓包
默认升级包里面没有这个命令，可以从下面编译路径拷贝
out\target\product\xxxref\symbols\system\xbin
用法：
tcpdump -s 0 -w 文件名