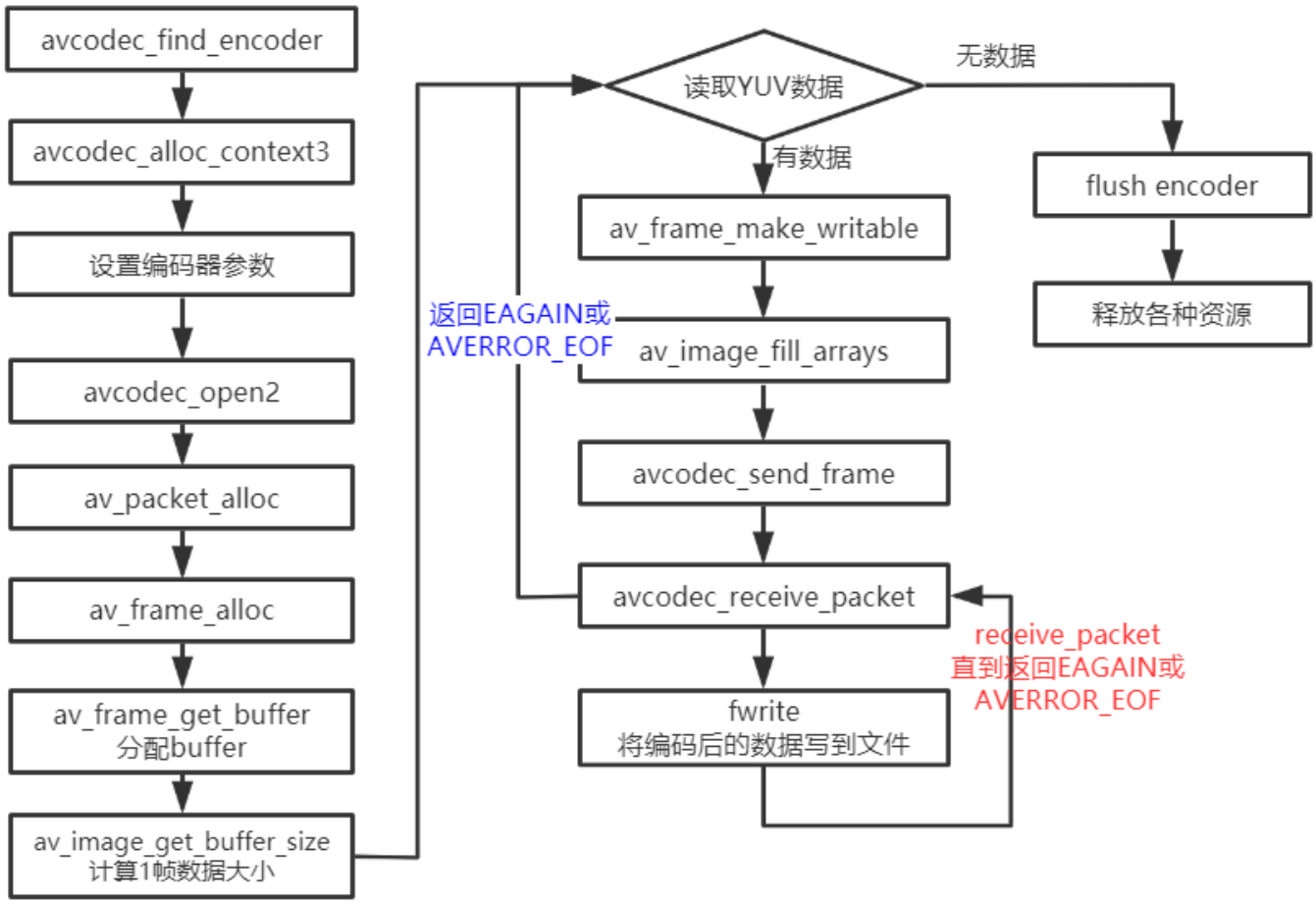
视频编码实战

FFmpeg流程



从本地读取YUV数据编码为h264格式的数据，然后再存入到本地，编码后的数据有带startcode。

与FFmpeg 示例音频编码的流程基本一致。
函数说明：

- avcodec_find_encoder_by_name：根据指定的编码器名称查找注册的编码器。
- avcodec_alloc_context3：为AVCodecContext分配内存。
- avcodec_open2：打开编解码器。
- avcodec_send_frame：将AVFrame非压缩数据给编码器。。
- avcodec_receive_packet：获取到编码后的AVPacket数据。
- av_frame_get_buffer: 为音频或视频数据分配新的buffer。在调用这个函数之前，必须在AVFame上设置好以下属性：format(视频为像素格式，音频为样本格式)、nb_samples(样本个数，针对音频)、channel_layout(通道类型，针对音频)、width/height(宽高，针对视频）。
- av_frame_make_writable：确保AVFrame是可写的，尽可能避免数据的复制。
  如果AVFrame不是是可写的，将分配新的buffer和复制数据。
- av_image_fill_arrays: 存储一帧像素数据存储到AVFrame对应的data buffer。

编码出来的h264数据可以直接使用ffplay播放，也可以使用VLC播放。

av_image_get_buffer_size

int av_image_get_buffer_size(enum AVPixelFormat pix_fmt, int width, int height, int align);
函数的作用是通过指定像素格式、图像宽、图像高来计算所需的内存大小
重点说明一个**参数align**:此参数是设定内存对齐的对齐数，也就是按多大的字节进行内存对齐：

- 比如设置为1，表示按1字节对齐，那么得到的结果就是与实际的内存大小一样。3
- 再比如设置为4，表示按4字节对齐。也就是内存的起始地址必须是4的整倍数。

av_image_alloc

av_image_alloc()是这样定义的。此函数的功能是按照指定的宽、高、像素格式来分配图像内存。
int av_image_alloc(uint8_t *pointers[4], int linesizes[4], int w, int h, enum AVPixelFormat pix_fmt, int align);

- pointers[4]：保存图像通道的地址。如果是RGB，则前三个指针分别指向R,G,B的内存地址。第四个指针保留不用 linesizes[4]：保存图像每个通道的内存对齐的步长，即一行的对齐内存的宽度，此值大小等于图像宽度。
- w: 要申请内存的图像宽度。
- h: 要申请内存的图像高度。
- pix_fmt: 要申请内存的图像的像素格式。
- align: 用于内存对齐的值。
- 返回值：所申请的内存空间的总大小。如果是负值，表示申请失败。

av_image_fill_arrays

int av_image_fill_arrays(uint8_t *dst_data[4], int dst_linesize[4], const uint8_t *src, enum AVPixelFormat pix_fmt, int width, int height, int align);
av_image_fill_arrays()函数自身不具备内存申请的功能，此函数类似于格式化已经申请的内存，即通过av_malloc()函数申请的内存空间，或者av_frame_get_buffer()函数申请的内存空间。
再者，av_image_fill_arrays()中参数具体说明：

- dst_data[4]: [out]对申请的内存格式化为三个通道后，分别保存其地址
- dst_linesize[4]: [out]格式化的内存的步长（即内存对齐后的宽度)
- *src: [in]av_alloc()函数申请的内存地址。
- pix_fmt: [in] 申请 src内存时的像素格式
- width: [in]申请src内存时指定的宽度
- height: [in]申请scr内存时指定的高度
- align: [in]申请src内存时指定的对齐字节数。

H.264 码率设置

1 什么是视频码率

- 视频码率是视频数据（包含视频色彩量、亮度量、像素量）每秒输出的位数。一般用的单位是kbps。

2 设置视频码率的必要性

- 在网络视频应用中，视频质量和网络带宽占用是相矛盾的。通常情况下，视频流占用的带宽越高则视频质量也越高，需要的网络带宽也越大，解决这一矛盾的钥匙当然是视频编解码技术。评判一种视频编解码技术的优劣，是比较在相同的带宽条件下，哪个视频质量更好；在相同的视频质量条件下，哪个占用的网络带宽更少（文件体积小）。
- 是不是视频码率越高，质量越好呢？理论上是这样的。然而在我们肉眼分辨的范围内，当码率高到一定程度时，就没有什么差别了。所以码率设置有它的最优值，H.264（也叫AVC或X264）的文件中，视频的建议码率如下：

视频大小	分辨率	推荐码率
480P	720X480	1800Kbps
720P	1280X720	3500Kbps
1080P	1920X1080	8500Kbps

3 手机设置码率建议

- 通过上面的介绍，结合我做过的一些手机项目，我总结了一套设置码率的公式，分享给大家如下：

项目	计算公式	192X144	320X240	480X360	640X480	1280X720	1920X1080
极低码率	(宽X高X3)/4	30kbps	60kbps	120kbps	250kbps	500kbps	1000kbps
低码率	(宽X高X3)/2	60kbps	120kbps	250kbps	500kbps	1000kbps	2000kbps
中码率	(宽X高X3)	120kbps	250kbps	500kbps	1000kbps	2000kbps	4000kbps
高码率	(宽X高X3)X2	250kbps	500kbps	1000kbps	2000kbps	4000kbps	8000kbps
极高码率	(宽X高X3)X4	500kbps	1000kbps	2000kbps	4000kbps	8000kbps	16000kbps

FFmpeg与H264编码指南

鉴于x264的参数众多，各种参数的配合复杂，为了使用者方便，x264建议如无特别需要可使用preset和tune设置。这套开发者推荐的参数较为合理，可在此基础上在调整一些具体参数以符合自己需要，手动设定的参数会覆盖preset和tune里的参数。

使用**ffmpeg -h encoder=libx264** 命令查询相关支持的参数

```
1 libx264 AVOptions:
```

```
  -preset           <string>     E..V..... Set the encoding preset (cf. x
264 --fullhelp) (default "medium")
  -tune             <string>     E..V..... Tune the encoding params (cf.
x264 --fullhelp)
  -profile          <string>     E..V..... Set profile restrictions (cf.
x264 --fullhelp)
  -fastfirstpass    <boolean>    E..V..... Use fast settings when encodin
g first pass (default true)
  -level            <string>     E..V..... Specify level (as defined by A
nnex A)
  -passlogfile      <string>     E..V..... Filename for 2 pass stats
  -wpredp           <string>     E..V..... Weighted prediction for P-fram
es
  -a53cc            <boolean>    E..V..... Use A53 Closed Captions (if av
ailable) (default true)
  -x264opts         <string>     E..V..... x264 options
  -crf              <float>      E..V..... Select the quality for constan
t quality mode (from -1 to FLT_MAX) (default -
1)
  -crf_max          <float>      E..V..... In CRF mode, prevents VBV from
lowering quality beyond this point. (from -1
to FLT_MAX) (default -1)
  -qp               <int>        E..V..... Constant quantization paramete
r rate control method (from -1 to INT_MAX) (de
fault -1)
  -aq-mode          <int>        E..V..... AQ method (from -1 to INT_MAX)
(default -1)
     none                      E..V.....
     variance                  E..V..... Variance AQ (complexity mask)
     autovariance              E..V..... Auto-variance AQ
     autovariance-biased           E..V..... Auto-variance AQ with bias
to dark scenes
  -aq-strength      <float>      E..V..... AQ strength. Reduces blocking
and blurring in flat and textured areas. (from
 -1 to FLT_MAX) (default -1)
  -psy              <boolean>    E..V..... Use psychovisual optimizations
. (default auto)
  -psy-rd           <string>     E..V..... Strength of psychovisual optim
ization, in <psy-rd>:<psy-trellis> format.
  -rc-lookahead     <int>        E..V..... Number of frames to look ahead
```

```
for frametype and ratecontrol (from -1 to INT
_MAX) (default -1)
  -weightb           <boolean>    E..V..... Weighted prediction for B-fram
es. (default auto)
  -weightp           <int>        E..V..... Weighted prediction analysis m
ethod. (from -1 to INT_MAX) (default -1)
     none                         E..V.....
     simple                       E..V.....
     smart                        E..V.....
  -ssim              <boolean>    E..V..... Calculate and print SSIM stats
. (default auto)
  -intra-refresh     <boolean>    E..V..... Use Periodic Intra Refresh ins
tead of IDR frames. (default auto)
  -bluray-compat     <boolean>    E..V..... Bluray compatibility workaroun
ds. (default auto)
  -b-bias            <int>        E..V..... Influences how often B-frames
are used (from INT_MIN to INT_MAX) (default IN
T_MIN)
  -b-pyramid         <int>        E..V..... Keep some B-frames as referenc
es. (from -1 to INT_MAX) (default -1)
     none                         E..V.....
     strict                       E..V..... Strictly hierarchical pyramid
     normal                       E..V..... Non-strict (not Blu-ray compat
ible)
  -mixed-refs        <boolean>    E..V..... One reference per partition, a
s opposed to one reference per macroblock (def
ault auto)
  -8x8dct            <boolean>    E..V..... High profile 8x8 transform. (d
efault auto)
  -fast-pskip        <boolean>    E..V..... (default auto)
  -aud               <boolean>    E..V..... Use access unit delimiters. (d
efault auto)
  -mbtree            <boolean>    E..V..... Use macroblock tree ratecontro
l. (default auto)
  -deblock           <string>     E..V..... Loop filter parameters, in <al
pha:beta> form.
  -cplxblur          <float>      E..V..... Reduce fluctuations in QP (bef
ore curve compression) (from -1 to FLT_MAX) (d
efault -1)
  -partitions        <string>     E..V..... A comma-separated list of part
```

```
itions to consider. Possible values: p8x8, p4x
4, b8x8, i8x8, i4x4, none, all
  -direct-pred       <int>        E..V..... Direct MV prediction mode (fro
m -1 to INT_MAX) (default -1)
     none                         E..V.....
     spatial                      E..V.....
     temporal                     E..V.....
     auto                         E..V.....
  -slice-max-size    <int>        E..V..... Limit the size of each slice i
n bytes (from -1 to INT_MAX) (default -1)
  -stats             <string>     E..V..... Filename for 2 pass stats
  -nal-hrd           <int>        E..V..... Signal HRD information (requir
es vbv-bufsize; cbr not allowed in .mp4) (from
 -1 to INT_MAX) (default -1)
     none                         E..V.....
     vbr                          E..V.....
     cbr                          E..V.....
  -avcintra-class    <int>        E..V..... AVC-Intra class 50/100/200 (fr
om -1 to 200) (default -1)
  -me_method         <int>        E..V..... Set motion estimation method (
from -1 to 4) (default -1)
     dia                          E..V.....
     hex                          E..V.....
     umh                          E..V.....
     esa                          E..V.....
     tesa                         E..V.....
  -motion-est        <int>        E..V..... Set motion estimation method (
from -1 to 4) (default -1)
     dia                          E..V.....
     hex                          E..V.....
     umh                          E..V.....
     esa                          E..V.....
     tesa                         E..V.....
  -forced-idr        <boolean>    E..V..... If forcing keyframes, force th
em as IDR frames. (default false)
  -coder             <int>        E..V..... Coder type (from -1 to 1) (def
ault default)
     default                      E..V.....
     cavlc                        E..V.....
     cabac                        E..V.....
```

```
      vlc                         E..V.....
      ac                          E..V.....
  -b_strategy        <int>        E..V..... Strategy to choose between I/P/B-frames (from -1 to 2) (default -1)
  -chromaoffset      <int>        E..V..... QP difference between chroma and luma (from INT_MIN to INT_MAX) (default -1)

  -sc_threshold      <int>        E..V..... Scene change threshold (from INT_MIN to INT_MAX) (default -1)
  -noise_reduction   <int>        E..V..... Noise reduction (from INT_MIN to INT_MAX) (default -1)
  -x264-params       <string>     E..V..... Override the x264 configuration using a :-separated list of key=value parameters
```

英文地址：https://trac.ffmpeg.org/wiki/Encode/H.264。内容有一定出入，但是可以借鉴学习。

x264是一个 H.264/MPEG4 AVC 编码器，本指南将指导新手如何创建高质量的H.264视频。
对于普通用户通常有两种码率控制模式：CRF（Constant Rate Factor)和Two pass ABR。码率控制是一种决定为每一个视频帧分配多少比特数的方法，它将决定文件的大小和质量的分配。
如果你在编译和安装libx264 方面需要帮助，请查看ffmpeg和x264编译指南：
http://ffmpeg.org/trac/ffmpeg/wiki/CompilationGuide

参考：https://www.jianshu.com/p/b46a33dd958d

CRF（Constant Rate Factor)：

1 选择一个CRF值

量化比例的范围为0~51，其中0为无损模式，**23为缺省值**，51可能是最差的。该数字越小，图像质量越好。从主观上讲，18~28是一个合理的范围。18往往被认为从视觉上看是无损的，它的输出视频质量和输入视频一模一样或者说相差无几。但从技术的角度来讲，它依然是有损压缩。
若CRF值加6，输出码率大概减少一半；若CRF值减6，输出码率翻倍。通常是在保证可接受视频质量的前提下选择一个最大的CRF值，如果输出视频质量很好，那就尝试一个更大的值，如果看起来很糟，那就尝试一个小一点值。
注释：本文所提到的量化比例只适用于8-bit x264（10-bit x264的量化比例 为0~63），你可以使用x264 --help命令在Output bit depth选项查看输出位深，在各种版本中，8bit是最常见的。

2 选择一个preset和tune

preset

预设是一系列参数的集合，这个集合能够在编码速度和压缩率之间做出一个权衡。一个编码速度稍慢的预设会提供更高的压缩效率（压缩效率是以文件大小来衡量的)。这就是说，假如你想得到一个指定大小的文件或者采用恒定比特率编码模式，你可以采用一个较慢的预设来获得更好的质量。同样的，对于恒定质量编码模式，你可以通过选择一个较慢的预设轻松地节省比特率。
如果你很有耐心，通常的建议是使用最慢的预设。目前所有的预设按照编码速度降序排列为：

- ultrafast
- superfast
- veryfast
- faster
- fast
- medium – default preset
- slow
- slower
- veryslow
- placebo - ignore this as it is not useful (see FAQ)

默认为medium级别。

你可以使用--preset来查看预设列表，也可以通过x264 --fullhelp来查看预设所采用的参数配置。

针对 libx264做过简单的各选项对比测试，结果如下图

crf	avg_duration	preset	avg_bitrate	avg_time	bitrate1	bitrate2	bitrate3	bitrate4	bitrate5	time1	time2	time3	time4	time5
-	12.6	-	878.8	0	555	912	791	585	1551	-	-	-	-	-
28	12.6	veryfast	878.4	1.1904	462	908	811	574	1637	1.003	0.805	0.800	0.921	2.423
28	12.6	faster	958.4	1.7376	604	1058	845	646	1639	1.565	1.239	1.296	1.479	3.109
28	12.6	fast	917.6	2.2948	589	1028	828	624	1519	1.925	1.631	1.789	1.788	4.341
28	12.6	medium	871.8	2.8308	520	961	793	585	1500	2.065	2.052	2.013	2.205	5.819
28	12.6	slow	811.8	4.1892	470	890	755	538	1406	2.713	2.699	2.992	3.169	9.373
28	12.6	slower	809.4	8.056	473	877	759	535	1403	5.129	4.801	5.272	5.844	19.234
28	12.6	veryslow	739.8	17.2842	438	801	709	492	1259	10.290	10.010	9.950	11.269	44.902

从图中可以看出，当其他参数固定时，选择不同的preset，对应的码率和编码时间都不一样，

tune

tune是x264中重要性仅次于preset的选项，它是视觉优化的参数，tune可以理解为视频偏好（或者视频类型），tune不是一个单一的参数，**而是由一组参数构成-tune来改变参数设置**。当前的 tune包括：

- film：电影类型，对视频的质量非常严格时使用该选项
- animation：动画片，压缩的视频是动画片时使用该选项
- grain：颗粒物很重，该选项适用于颗粒感很重的视频
- stillimage：静态图像，该选项主要用于静止画面比较多的视频
- psnr：提高psnr，该选项编码出来的视频psnr比较高

- ssim：提高ssim，该选项编码出来的视频ssim比较高
- fastdecode：快速解码，该选项有利于快速解码
- **zerolatency**：零延迟，该选项主要用于视频直播

如果你不确定使用哪个选项或者说你的输入与所有的tune皆不匹配，你可以忽略--tune 选项。
你可以使用-tune来查看tune列表，也可以通过x264 --fullhelp来查看tune所采用的参数配置。

profile

另外一个可选的参数是-profile:v，它可以将你的输出限制到一个特定的 H.264 profile。一些非常老的或者要被淘汰的设备仅支持有限的选项，比如只支持baseline或者main。

所有的profile 包括：

1. baseline profile：基本画质。支持I/P 帧，只支持无交错（Progressive）和CAVLC；
2. extended profile：进阶画质。支持I/P/B/SP/SI 帧，只支持无交错（Progressive）和CAVLC；
3. main profile：主流画质。提供I/P/B 帧，支持无交错（Progressive）和交错（Interlaced），也支持CAVLC 和CABAC 的支持；
4. high profile：高级画质。在main Profile 的基础上增加了8x8内部预测、自定义量化、 无损视频编码和更多的YUV 格式；

想要说明H.264 high profile与H.264 main profile的区别就要讲到H.264的技术发展了。JVT于2003年完成H.264基本部分标准制定工作，包含baseline profile、extended profile和main profile，分别包括不同的编码工具。之后JVT又完成了H.264 FREXt（即：Fidelity Range Extensions）扩展部分（Amendment）的制定工作，包括high profile（HP）、high 10 profile（Hi10P）、high 4:2:2 profile（Hi422P）、high 4:4:4 profile（Hi444P）4个profile。

H.264 baseline profile、extended profile和main profile都是针对8位样本数据、4:2:0格式的视频序列，FREXt将其扩展到8～12位样本数据，视频格式可以为4:2:0、4:2:2、4:4:4，设立了high profile（HP）、high 10 profile（Hi10P）、high 4:2:2 profile（Hi422P）、high 4:4:4 profile（Hi444P） 4个profile，这4个profile都以main profile为基础。

在相同配置情况下，High profile（HP）可以比Main profile（MP）节省10%的码流量，比MPEG-2 MP节省60%的码流量，具有更好的编码性能。根据应用领域的不同：

- **baseline profile多应用于实时通信领域；**
- **main profile多应用于流媒体领域；**
- **high profile则多应用于广电和存储领域。**

扩展阅读：H264编码系列之profile & level控制

低延迟

x264提供了一个 -tune zerolatency 选项。

兼容性

如果你想让你的视频最大化的和目标播放设备兼容(比如老版本的的ios或者所有的android 设备），那么你可以这做：

-profile:v baseline

这将会关闭很多高级特性，但是它会提供很好的兼容性。也许你可能不需要这些设置，因为一旦你用了这些设置，在同样的视频质量下与更高的编码档次相比会使比特率稍有增加。

关于profile列表和关于它们的描述，你可以运行x264 --fullhelp

要牢记apple quick time 对于x264编码的视频只支持 YUV 420颜色空间，而且不支持任何高于 mian profile编码档次。这样对于quick time 只留下了**两个兼容选项baseline和 main**。其他的编码档次qucik time均不支持，虽然它们均可以在其它的播放设备上回放。

问题与解答：

1 两遍编码模式能够比CRF模式提供更好的质量吗?

不能，但它可以更加精确地控制目标文件大小。

2 为什么 placebo 是一个浪费时间的玩意儿?

与 veryslow相比，它以极高的编码时间为代价换取了大概1%的视频质量提升，这是一种收益递减准则，veryslow 与 slower相比提升了3%；slower 与 slow相比提升了5%；slow 与 medium相比提升了5%~10%。

3 为什么我的无损输出看起来是无损的?

这是由于rgb->yuv的转换，如果你转换到yuv444,它依然是无损的。

4 显卡能够加速x264的编码吗?

不，x264没有使用（至少现在没有），有一些私有编码器使用了GPU加快了编码速度，但这并不意味着它们经过良好的优化。也有可能还不如x264，或许速度更慢。总的来说，ffmpeg到目前为止还不支持GPU。

翻译注释：x264在2013版中已经开始支持基于opencl的显卡加速，用于帧类型的判定。

5 为Quick time 播放器压制视频

你需要使用-pix_fmt yuv420p来是你的输出支持QT 播放器。这是因为对于H.264视频剪辑苹果的Quick time只支持 YUV420颜色空间。否则ffmpeg会根据你的视频源输出与Quick time 不兼容的视频格式或者不是基于ffmpeg的视频。

X264参数之zerolatency的分析

最近在测试服务器的在线转码性能时发现，加入zerolatency之后，转码路数会明显降低。

我们都知道，加入zerolatency的目的是为了降低在线转码的编码延迟，那么，该参数是如何影响到x264的转码性能了呢?

首先，先来看看代码中编码延迟的影响因素：

h->frames.i_delay = max(h->param.i_bframe, h->param.rc.i_lookahead)

```
+  h->i_thread_frames - 1
+  h->param.i_sync_lookahead
+ h->param.b_vfr_input
```

设置zerolatency后，相应的参数配置如下：

```
if( !strncasecmp( s, "zerolatency", 11 ) )
{
    param->rc.i_lookahead = 0;
    param->i_sync_lookahead = 0;
    param->i_bframe = 0;
    param->b_sliced_threads = 1;
    param->b_vfr_input = 0;
    param->rc.b_mb_tree = 0;
}
```

下面我们来看一下zerolatency设置中各个参数的意义：

rc_lookahead: Set number of frames to look ahead for frametype and ratecontrol.

该参数为mb-tree码率控制和vbv-lookahead设置可用的帧数量，最大值为250。对于mbi-tree来说，rc_lookahead值越大，会得到更准确的结果，但编码速度也会更慢，因为编码器需要缓存慢rc_lookahead帧数据后，才会开始第一帧编码，增加编码延时，因此在实时视频通信中将其设置为0。

sync_lookahead: 设置用于线程预测的帧缓存大小，最大值为250。在第二遍及更多遍编码或基于分片线程时自动关闭。sync_lookahead = 0为关闭线程预测，可减小延迟，但也会降低性能。

bframes: I帧和P帧或者两个P帧之间可用的最大连续B帧数量，默认值为3。B帧可使用双向预测，从而显著提高压缩率，但由于需要缓存更多的帧数以及重排序的原因，会降低编码速度，增加编码延迟，因此在实时编码时也建议将该值设置为0。

sliced_threads: 基于分片的线程，默认值为off，开启该方法在压缩率和编码效率上都略低于默认方法，但没有编码延时。除非在**编码实时流或者对低延迟要求较高**的场合开启该方法，一般情况下建议设为off。

vfr_input： 与force-cfr选项相对应：

```
OPT("force-cfr")
    p->b_vfr_input = !atobool(value);
```

vfr_input= 1时，为可变帧率，使用timebase和timestamps做码率控制；vfr_input = 0时，为固定帧率，使用fps做码率控制。

mb_tree: 基于宏块树的码率控制。对于每个MB，向前预测一定数量的帧(该数量由rc_lookahead和keyint中的较小值决定)，计算该MB被引用的次数，根据被引用次数的多少决定为该MB分配的量化QP值。该方法会生成一个临时stats文件，记录了每个P帧中每个MB被参考的情况。使用mb_tree的方法能够节约大概30%的码率，但同时也会增加编码延迟，因此实时流编码时也将其关闭。

从这里分析可以看出来，不单是B帧导致延迟，当

原文地址：https://blog.csdn.net/DeliaPu/article/details/77004392

更多参考

CRF Guide (Constant Rate Factor in x264, x265 and libvpx)