**FFMPEG多线程解码**

FFMPEG多线程编码器一般以在Slice内分功能模块进行多线程编码，如h263，h263P，msmpeg(v1, v2, v3)，wmv1。包含以下几个线程：(1)Pre\_estimation\_motion\_thread运动估计前的准备;(2)Estimation\_motion\_thread运动估计;(3)Mb\_var\_thread宏块其他变量;(4)Encode\_thread编码主线程。当然也有例外，如FFV1编码器按Slice为线程单位进行多线程编码。

FFMPEG多线程解码器分为Frame级和Slice级两种，Slice级多线程同时解码一帧中不同的部分。Frame级多线程同时接受多帧码流，实现并行解码，当前帧处于显示状态时，未来的几帧已经在其他线程中被解码。

**1.         Slice Threading**

FFmpeg中，dvvideo\_decoder, ffv1\_decoder, h264\_decoder, mpeg2\_video\_decoder和mpeg\_video\_decoder均支持了Slice Threading。

实现方法是：首先为codecContext注册注册多线程处理函数excute()，Codec解码过程中处理Slice时调用avctx->excute()。excute()启动Slice解码工作线程开始多线程解码，同时快速返回开始下一Slice的解析和解码。

Frame Threading主线程和解码线程的同步如图1所示。

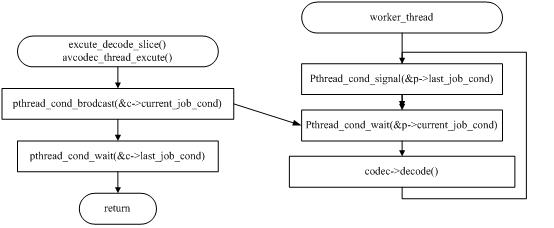


图1 Frame Threading主线程和解码线程的同步

**2.         Frame Threading**

目前为止支持Frame Threading的解码器有h264\_decoder, huffyuv\_decoder, ffvhuff\_decoder, mdec\_decoder, mimic\_decoder, mpeg4\_decoder, theora\_decoder, vp3\_decoder和vp8\_decoder。

Frame Threading有如下限制：用户函数draw\_horiz\_band()必须是线程安全的；为了提升性能，用户应该为codec提供线程安全的get\_buffer()回调函数；用户必须能处理多线程带来的延时。另外，支持Frame Threading的codec要求每个包包含一个完整帧。Buffer内容在ff\_thread\_await\_progress()调用之前不能读，同样，包括加边draw\_edges()在内的处理，在ff\_thread\_report\_progress()调用之后，Buffer内容不能写。

每个线程都有以下四个状态。如图2所示，为了保证线程安全，若Codec未实现update\_thread\_context()和线程安全的get\_buffer()，则必须在解码完成后才能将状态转换为STATUS\_SETUP\_FINISHED，意味着下一个线程只能在当前线程解码完成后才能开始解码。

而如图3所示，如果Codec实现update\_thread\_context()和线程安全的get\_buffer()，线程状态可以在解码开始之前转换为STATUS\_SETUP\_FINISHED，这样，下一个线程就可能与当前线程并行。

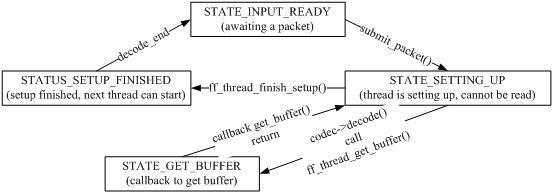


图2 Codec未实现update\_thread\_context()和线程安全的get\_buffer()，线程状态转换

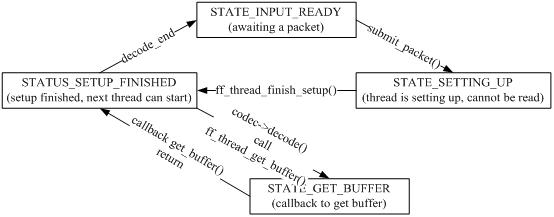


图3 Codec实现update\_thread\_context()和线程安全的get\_buffer()，线程状态转换

解码主线程通过调用submit\_packet()将码流交给对应的解码线程。主线程和解码线程的同步如图4所示。

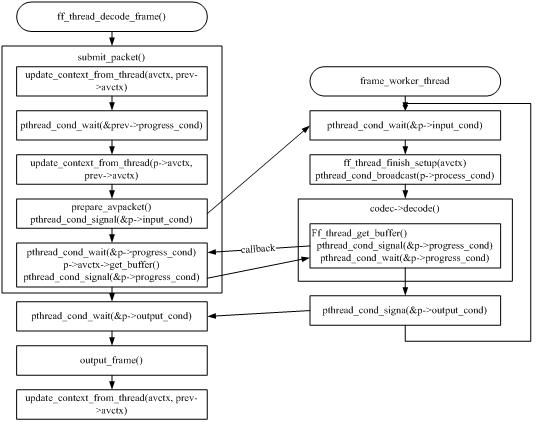


图4 Frame Threading主线程和解码线程的同步