LiveVideoStackCon

视频云客户端SDK的设计与实现

展晓凯

2017年10月20日-21日

北京.丽亭华苑酒店

自我介绍

- 展晓凯，曾就职于淘宝，开发机票搜索业务
- 12年加入唱吧，经历了唱吧从上线到拥有4亿用户的整个过程，负责唱吧音视频的开发，其中涉及到多个产品线，包括唱吧、唱吧直播间、火星等
- 目前工作于全民快乐，负责直播产品线业务，主要面向海外市场
- 未来2个月内会有一本关于音视频开发的书籍面市，书中详细介绍了移动平台下音视频开发的整个流程，也是这些年我从事移动平台下音视频开发的一个总结，希望可以帮助到音视频领域内更多的人们。

视频云客户端SDK的发展与意义

- 音视频的架构与开发已经演进了很长的时间，演进流程大约如下：
  - 广电领域
  - PC端的音视频领域
  - 移动端的音视频领域
- 视频云除了提供持续、稳定、高可用的线上服务之外，也提供客户端的SDK以方便客户在不了解音视频细节的条件下也可以快速的构建出自己的音视频App
- 有了SDK之后每个App则可以专注于自己所在的垂直领域业务，而整个音视频领域则呈现出百花齐放的状态，迅速出现了各种短视频社区、游戏直播、秀场直播、体育直播、在线教育等等App

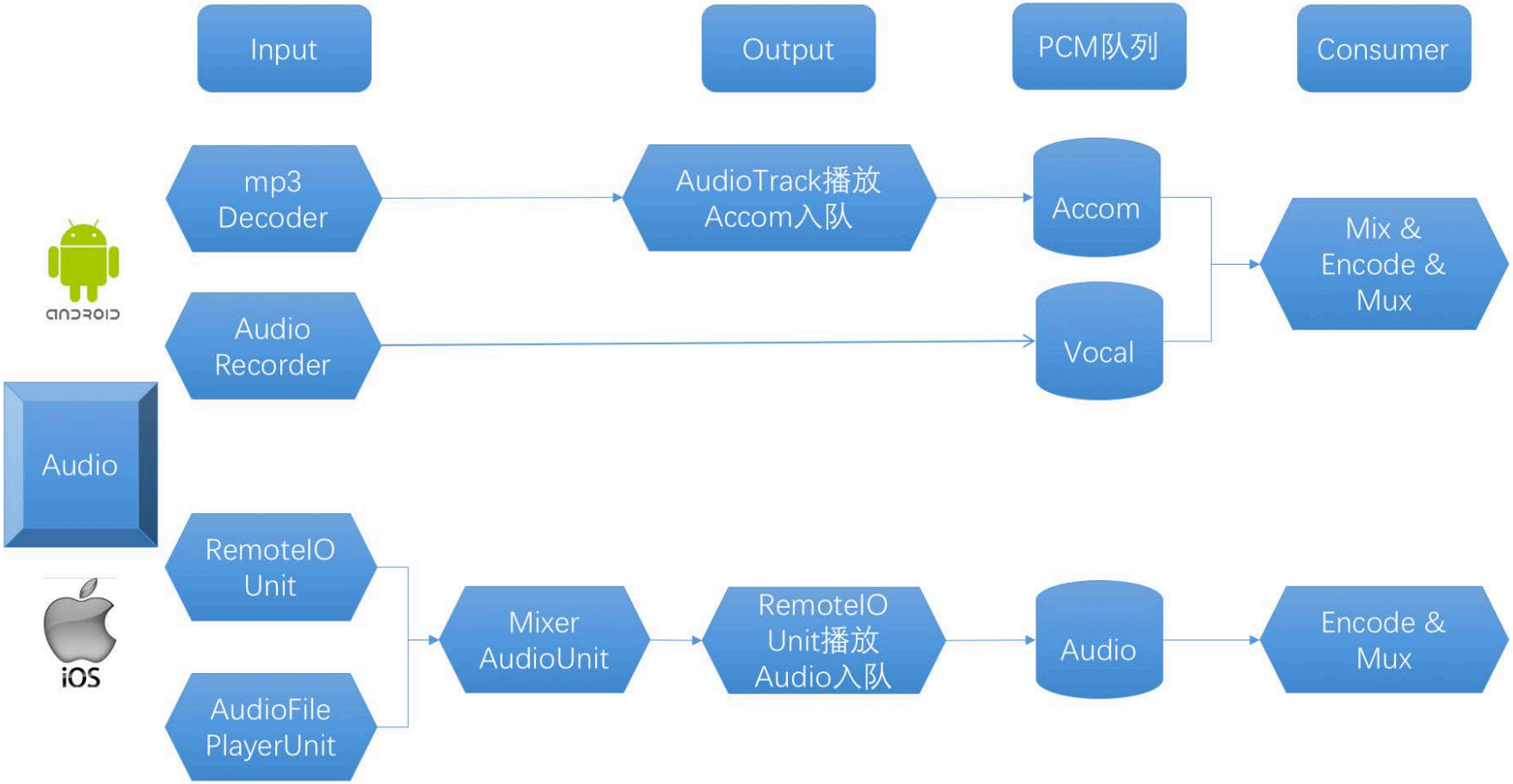
视频云客户端SDK的核心场景

- SDK核心场景如下
  - 录播:
    - 主播端，离线录制视频，增加一些音视频效果，最终上传到服务器;
    - 粉丝端，使用普通播放器就可以播放，然后进行一些社交行为。
  - 直播:
    - 主播端，实时将内容直播出去，并针对于观众的行为完成一些实时互动;
    - 观众端，需要使用定制的播放器观看，然后完成和主播的实时互动。
- 针对于录播和直播的共同点和区别如下：
  - 共同特性就是核心都使用视频录制器与视频播放器;
  - 本质区别在于是否具有实时交互性;
  - 各自场景下需要做一些特殊的配置（比如:直播中推流的稳定性，拉流的秒开；录播中对于视频的后期处理以及上传）。
- 视频录制器与视频播放器的架构设计

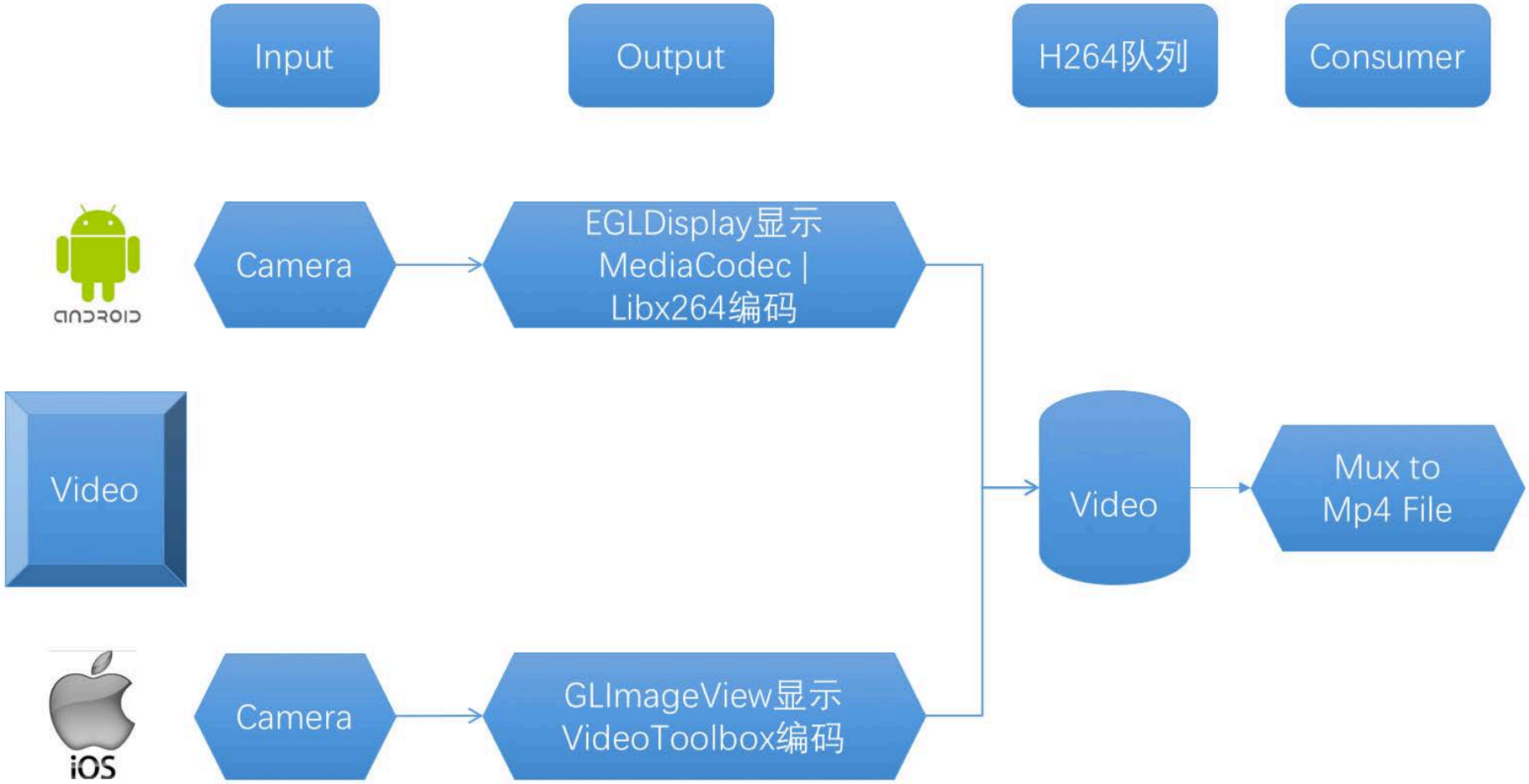
视频录制器模块拆分

- 输入
  - 音频采集
  - 画面采集
- 处理
  - 音频处理
  - 画面处理
- 输出
  - 预览
  - 编码(硬件编码\软件编码)
  - 封装 && IO输出

视频录制器音频部分架构设计



视频录制器视频部分架构设计



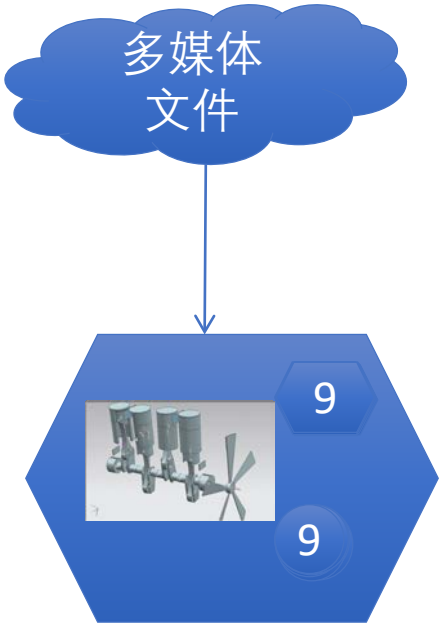
视频录制器总结

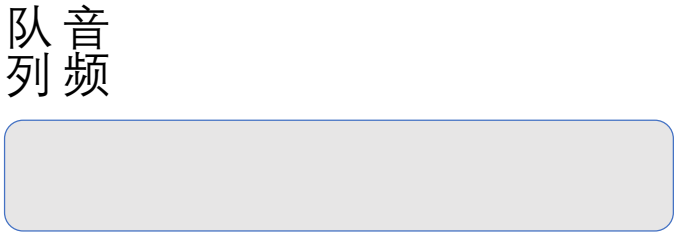
- 当Consumer模块将音视频编码数据Mux到一个Container之后就执行IO操作：
  - 对于录播场景直接写入本地磁盘即可
  - 对于直播场景则需要走对应的协议(rtmp/rtp)上传到流媒体服务器
- 整个视频录制器是一个典型的生产者/消费者模式的实践：
  - 生产者:采集与编码(内部还会有一个子生产者与消费者)
  - 消费者:封装与IO输出

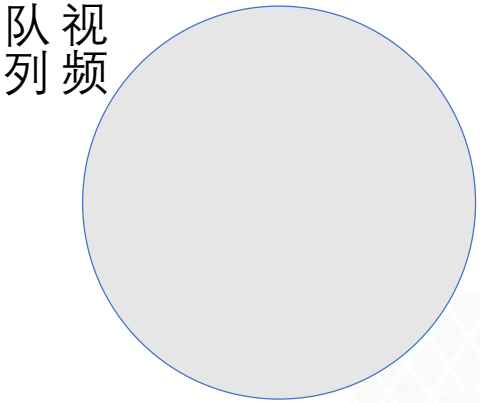
视频播放器模块拆分

- 输入
  - IO输入 && 解封装
  - 解码(硬件解码\软件解码)
- 处理
  - 音频处理(混音)
  - 画面处理(自动对比度)
  - 音视频同步
- 输出
  - 音频渲染
  - 视频渲染

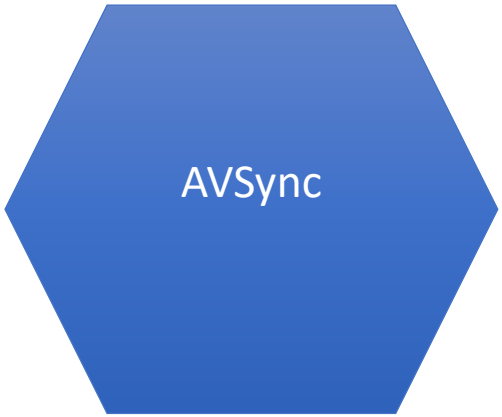
视频播放器-运行流程

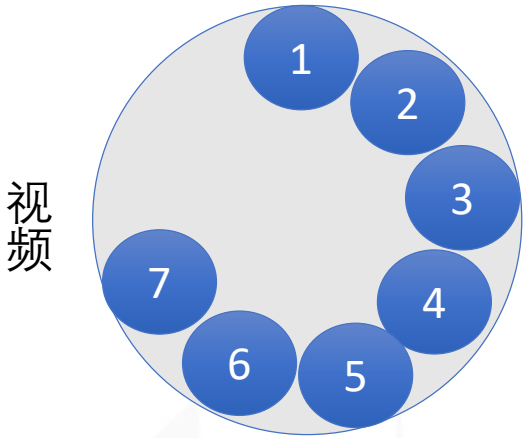






视频播放器-音视频同步策略展示





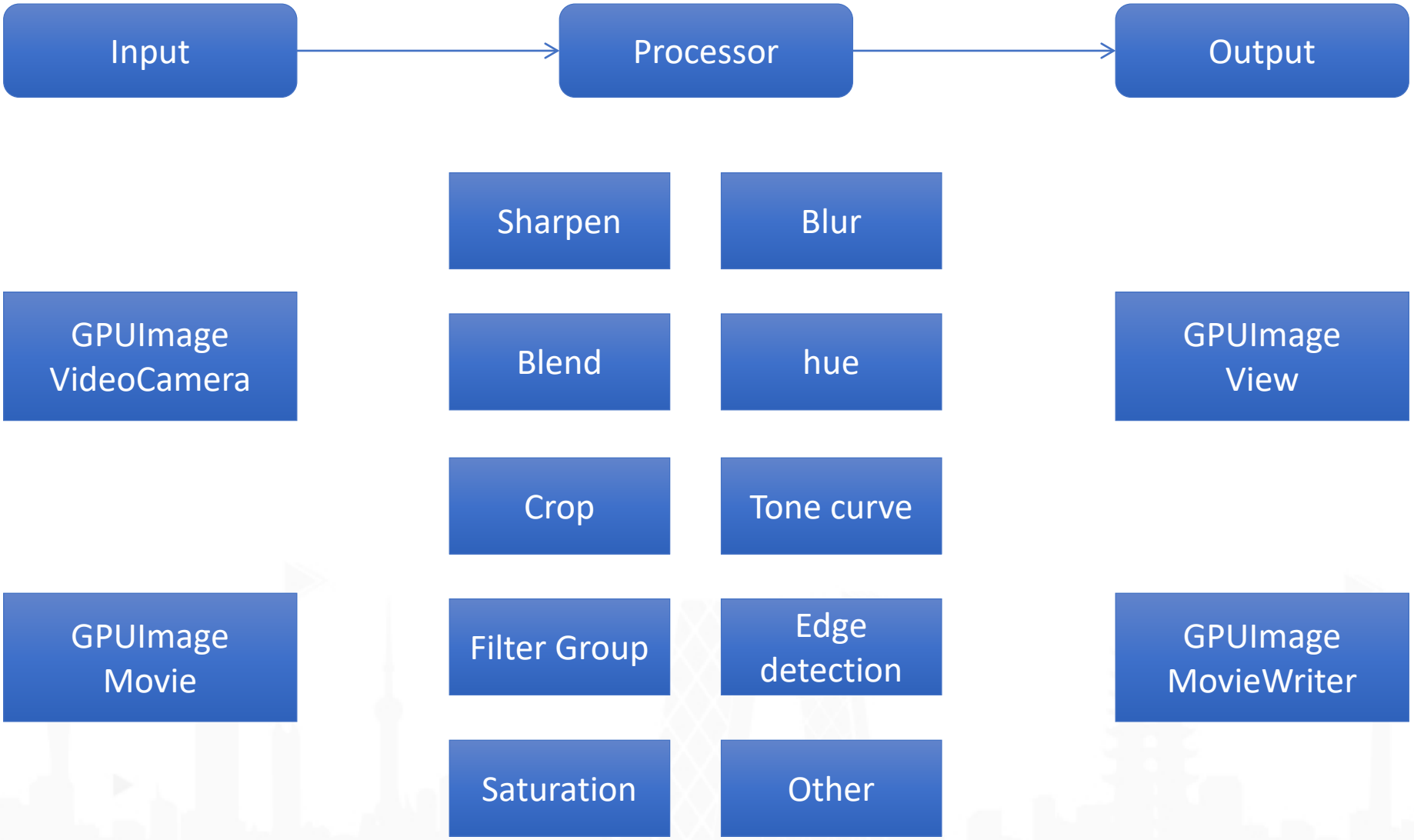
实现每一个模块

- 如果实现每一个细节对于普通开发工程师来讲是一件非常复杂的事情
- 最简单的是我们直接选择一家CDN厂商，它们肯定有一套提供好的SDK，每个客户端接入相应的SDK，然后完成自己的业务逻辑。
- 特殊需求可以基于SDK进行二次开发
- SDK中技术含量比较高的地方我们在这里重点介绍一下：
  - 跨平台的视频处理系统的构建
  - 跨平台的推流系统的构建

跨平台的视频处理系统的构建

- 场景分析
  - 美颜
  - 主题
  - 贴纸
- 输入输出
  - 输入:一个纹理ID与时间戳
  - 输出:处理完毕的纹理ID
- 技术选型: OpenGL ES
- 推荐GPUImage框架作为大家工作以及学习的资料

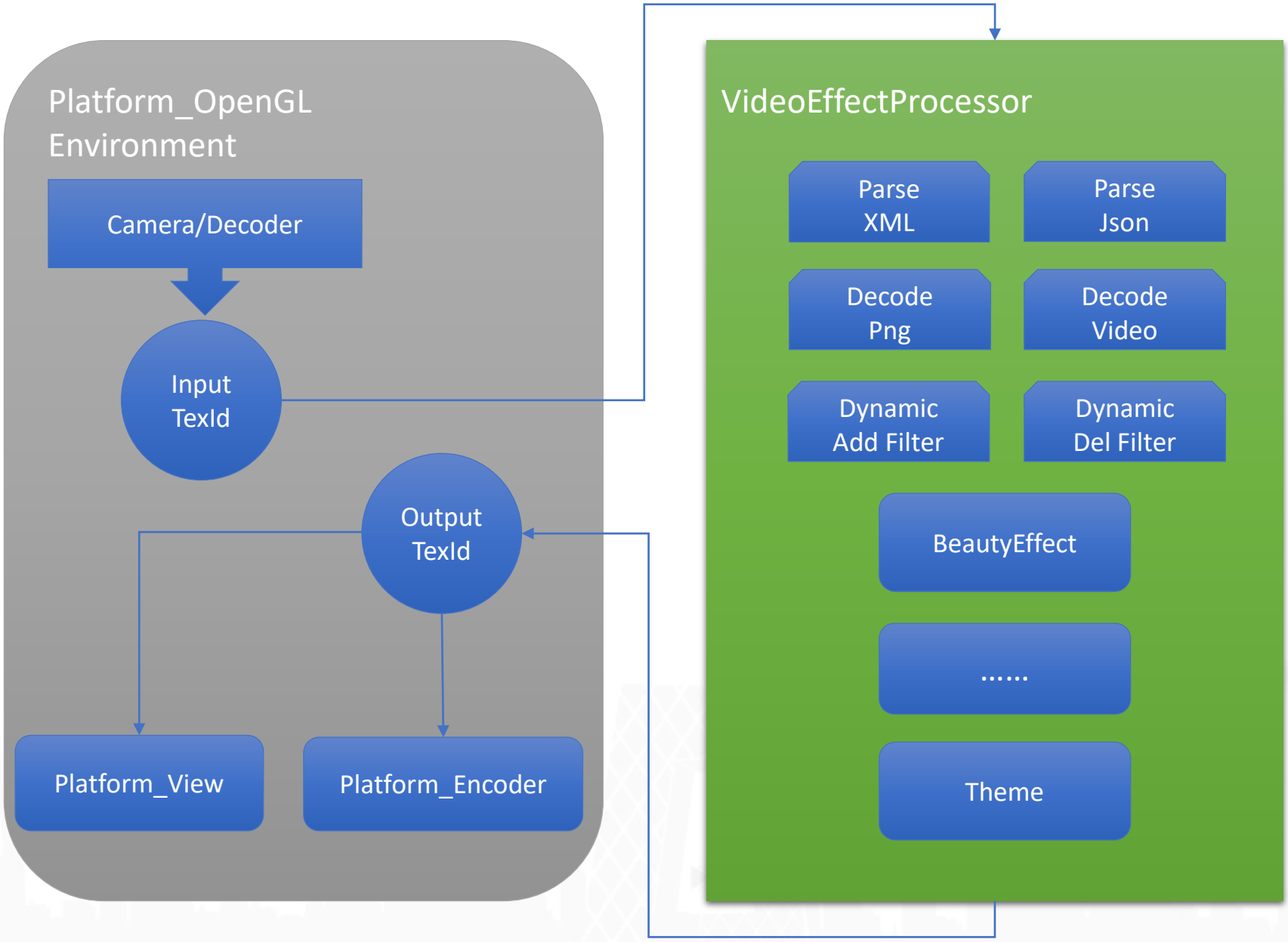
GPUImage框架结构介绍



跨平台的视频处理系统的构建

- 搭建两个客户端的OpenGL环境(包括上下文与窗口管理以及渲染线程)
  - Android:EGL + EGLDisplay（ANativeWindow-Surface-SurfaceView/TextureView）+ PThread
  - iOS:EAGL + UIView（CAEAGLLayer）+ GCD
- 抽象统一接口，服务于两个平台
  - 输入:纹理ID与当前帧的时间戳
  - 输出:处理之后的纹理ID
- 所承担的职责
  - 能够完成磨皮、提亮、色相、饱和度等基础图像处理
  - 能够完成主题的添加功能
  - 能够完成贴纸的功能

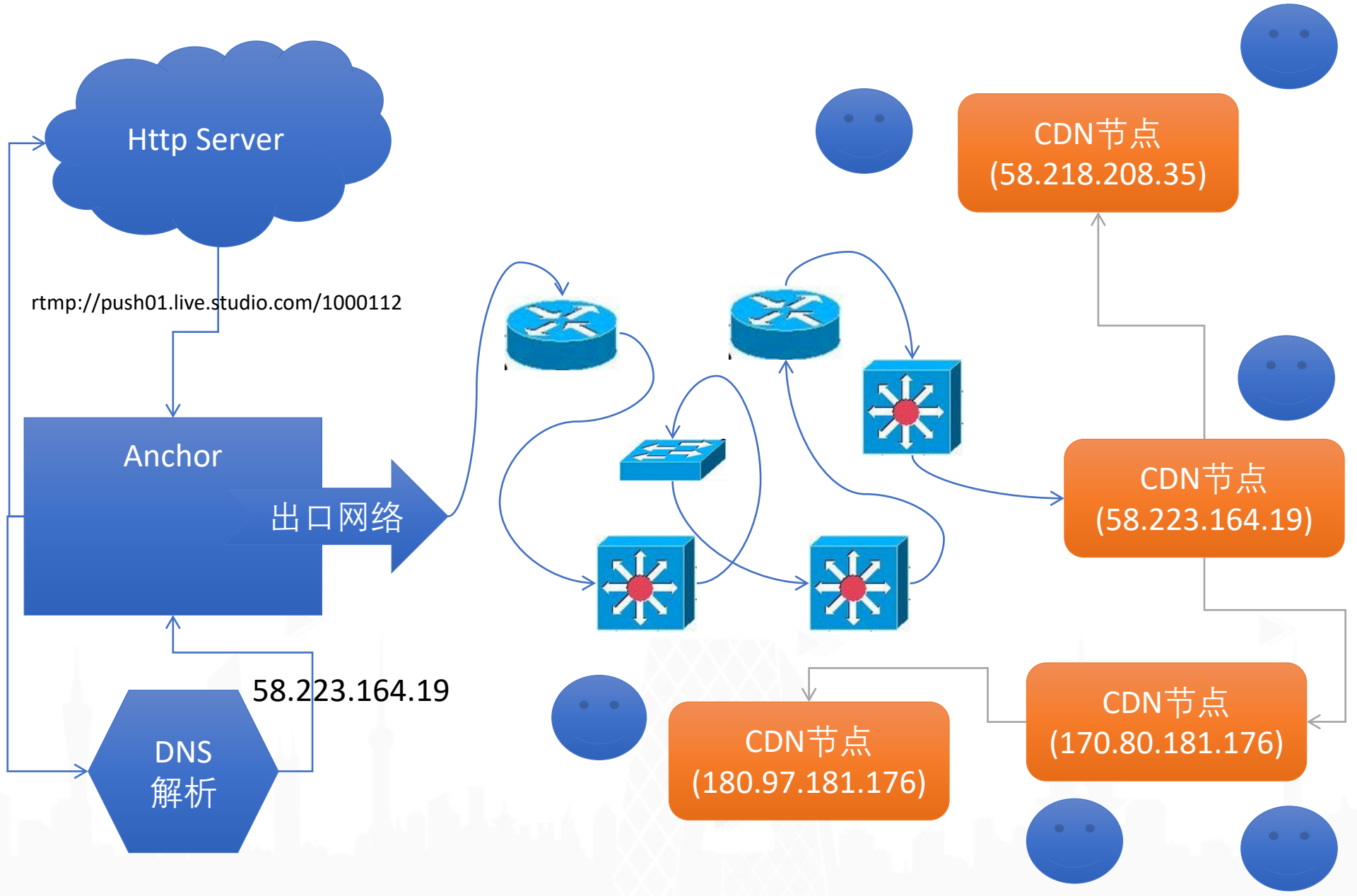
跨平台的视频处理系统的应用



跨平台的推流系统的构建

- 场景分析：
  - 无论网络是否有抖动，要维持交互的实时性；
  - 要保证正常直播的流畅性，能够根据网络条件的好坏来决定清晰度；
  - 要有一些统计数据帮助产品、运营去做一些策略优化(比如提升码率、提升分辨率)。
- 转变为技术实现：
  - 为了维持交互的实时性要在网络抖动的时候做出丢帧；
  - 为了维持直播的流畅性与清晰度要做码率自适应；
  - 为了做到主播端持续直播(比如链路拥塞或者IDC机房节点故障)，要做自动断线重连；
  - 为了方便开发人员持续优化推流策略，要做数据打点统计；
- 为何要跨平台
  - 节省开发成本：开发写一套代码，测试测一套代码
  - 提升开发效率：后期维护维护一套代码

推流的复杂流程



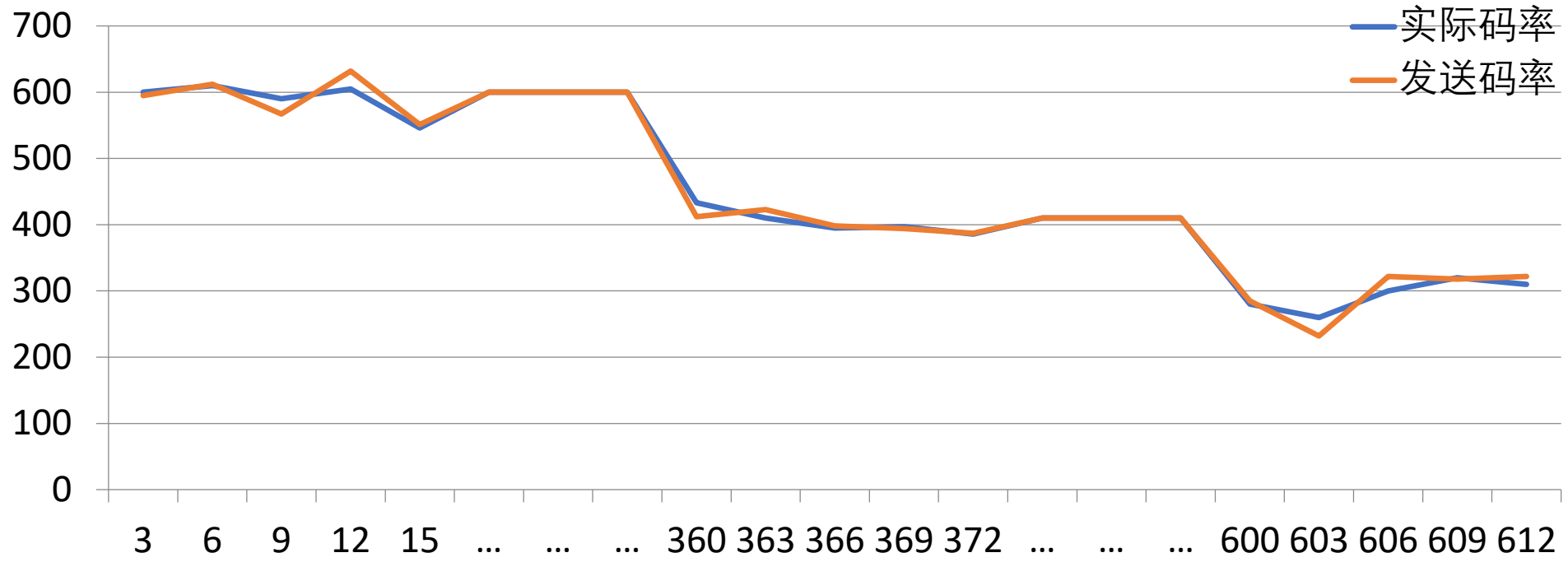
跨平台的推流系统的构建

- 如何实现
  - 输入：编码之后的H264数据与AAC数据
  - 输出：将流数据平稳的推送到流媒体服务器上
  - 实现手段：使用FFmpeg将AAC与H264封装成为flv格式，然后利用RTMP协议推到流媒体服务器
- 弱网丢帧策略
  - 当检测到队列中积攒的视频帧与音频帧到一定阈值的时候，为了保持交互性，我们要做丢帧处理
  - 对于视频帧，要明确丢弃掉的帧是否为I帧或者P帧
  - 对于音频帧，有多种策略，简单处理可以丢弃与丢弃视频帧同等时间长度的音频帧

跨平台的推流系统的构建-码率自适应

- 当前上行带宽速度的监测
  - 通过检测一个时间窗口内发送出去的包大小
  - 根据队列变化曲线来推测是否需要调高码率
- 强制编码器产生关键帧
  - 如果降低了码率可以把之前高码率的帧给丢掉，然后让编码器强制产生关键帧
- 改变编码器的输入帧率
  - 直接根据时间与帧率来过滤掉摄像头采集出多余的视频帧
- 改变编码器的输出码率
  - 对于libx264
  - 对于MediaCodec与VideoToolbox

跨平台的推流系统的构建-码率自适应



这里不单单调整的是码率，还有帧率，当帧率比较低的时候，只是单纯的增高码率，视频质量还是提升不上去，所以这两者会一起调整

链路选择与自动重连策略

- 由于DNS解析找不到最佳链路
  - 接入CDN提供的接口，主播推流之前，我们向CDN厂商请求一个最优的节点，而不是依赖Local DNS来解析
  - 对于主播端，拿到多个推流节点，进行POST一个500KB的flv文件，测试网络链路情况，选择最优的链路
- 自动重连策略
  - 持续推流一段时间之后(1h以上)，网络链路有可能会拥塞，IDC机房节点也有可能会出现问题，所以SDK底层会要有自动重连机制保证会重新分配更优的链路与CDN节点

数据收集

- 连接阶段：主播端发起一次Connect,如果失败直接上报，否则记录Connect时间，单位毫秒
- 发布时长：从主播开播开始,到退出本次推流所持续的时间,单位秒
- 丢帧比例：在推流过程中，由于网络发生抖动或者弱网环境下，本地Queue中视频帧超过阈值,要进行丢帧,因此会有一个丢帧比例，一般以视频帧计算就可以了
- 平均速率：在整个推流过程中，使用发布出去的字节数除以发布时长得到平均速率
- 设置的速率：在整个推流过程中给客户端设置的推流速率 比如:video 500Kbps audio 64Kbps
- 码率自适应的变化曲线:High-360:Middle-3600:Low

机遇与挑战

- 目前我们公司主要做国外的直播业务
- 对于我个人来讲既是机遇又是挑战
  - 机遇在于国外的市场还很大
  - 挑战在于国外的用户需要被教育，用户场景需要被定义
- 我从不怀疑我们在互联网上迈出的任何一步都是人类历史向前迈进的一步，希望我们可以在音视频领域为人类带来进步。

Thank You

手机\微信:15652156892

邮箱:xiaokai.zhan@ushow.media

LiveVideoStackCon

聚音视　研修不止于形

关注**LiveVideoStack**公众号

回复 **展晓凯** 为讲师评分