C/C++Linux服务器开发

高级架构师课程

三年课程沉淀 | 五次精益升级 | 十年行业积累 | 百个实战项目 | 十万内容受众

讲师:darren/326873713

扫一扫 升职加薪

班主任:柚子/2690491738

讲师介绍--专业来自专注和实力

King老师

系统架构师，曾供职著名创业公司系统架构师，微软亚洲研究院、创维集团全球研发中心。国内第一代商业Paas平台开发者。著有多个软件专利，参与多个开源软件维护。在全球化，高可用的物联网云平台架构与智能硬件设计方面有丰富的研发与实战经验。

Lee老师

曾供职于华为和诺基亚通信长达7年时间，曾在某上市公司担任技术总监。 10年（C/C++）开发经验和产品管理经验，研究过过多款C/C++优秀开源软件的框架，开发过大型音频广播云平台，深入理解需求分析、架构分析和产品管理，精通敏捷开发流程和项目管理。

Darren老师

曾供职于国内知名半导体公司（珠海扬智/深圳联发科），曾在某互联网公司担任音视频通话项目经理。主要从事音视频驱动、多媒体中间件、流媒体服务器的开发，开发过即时通讯+音视频通话的大型项目，在音视频、C/C++/GO Linux服务器领域有丰富的实战经验。

课题：RTMP流媒体实战3

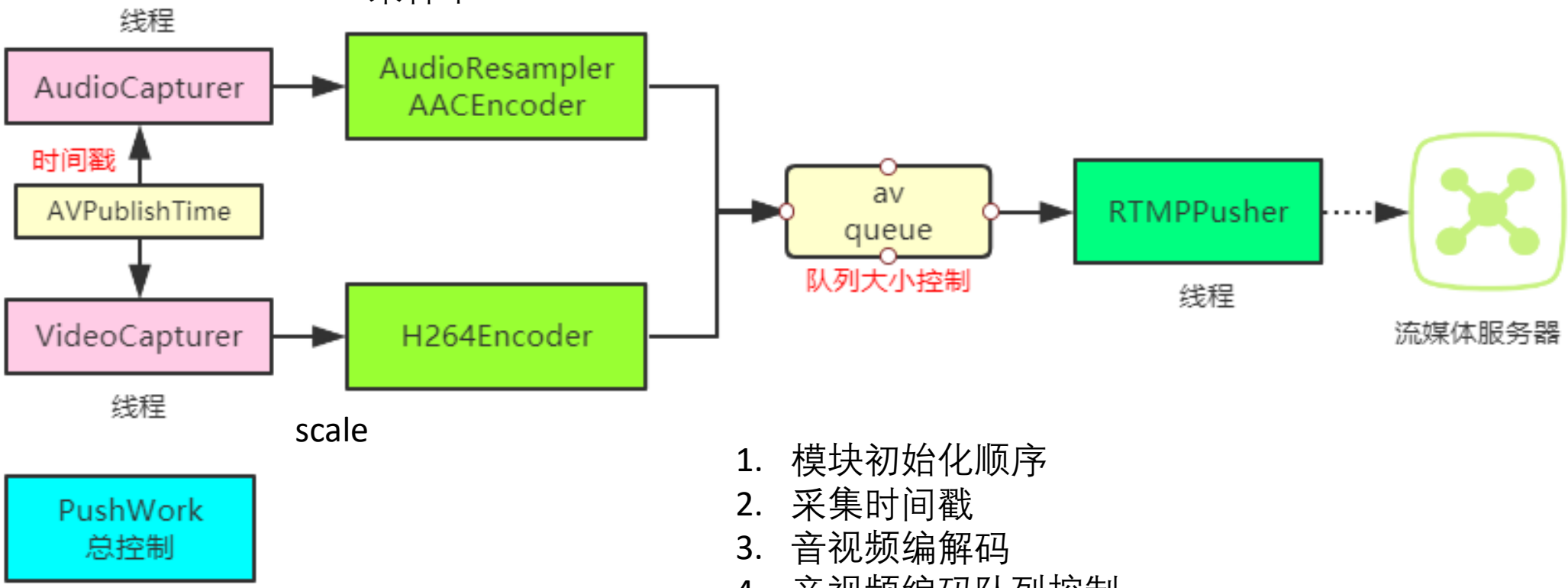
2020年3月10日 20:05正式上课

1. 推流实战分析
2. 拉流实战分析

第一章 推流实战

1 推流框架

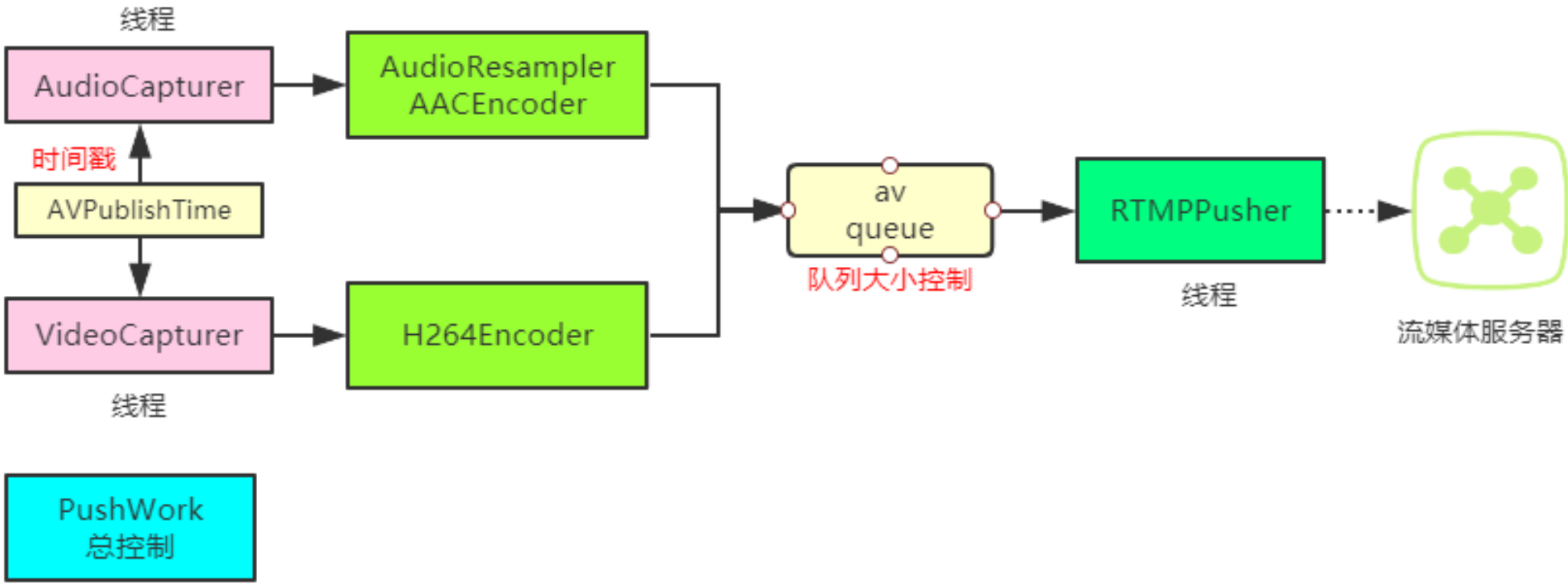
PCM S16交错 -> float 棋盘格类型
采样率



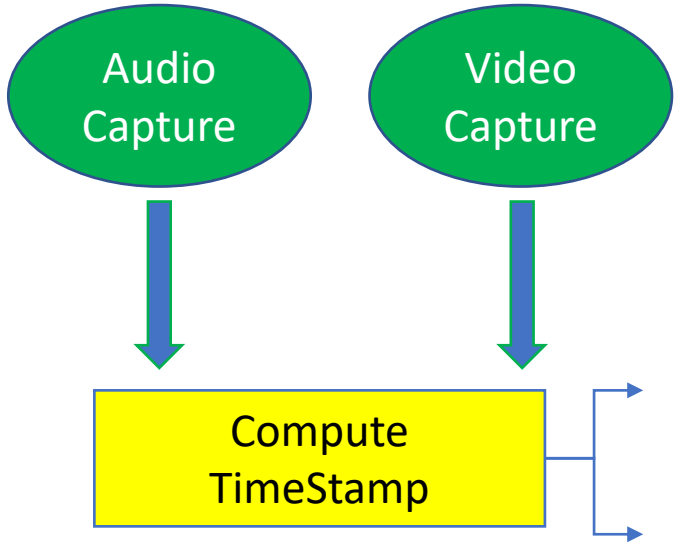
1. 模块初始化顺序
2. 采集时间戳
3. 音视频编解码
4. 音视频编码队列控制
5. 关键时间点

1.1 模块初始化顺序

推流模块（网络连接耗时） > 音视频编码模块 >音视频采集模块（）



1.2 采集时间戳—帧间隔模式



frameTimeStamp = lastFrameTimeStamp + frameDuration。

起始时间 =系统时间

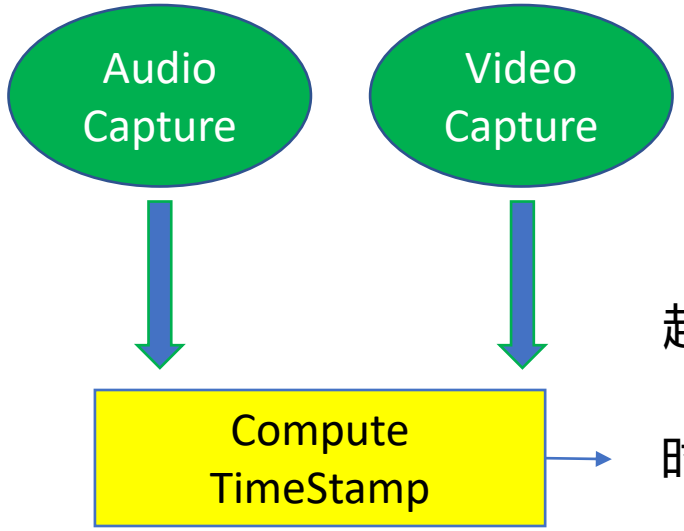
第一帧时间戳 = 系统时间 –起始时间

后续帧 = 上一帧时间戳 + **帧间隔**

优点：能输出frame duration稳定的音视频时间戳。

风险：**(1)**当系统负载比较重时有可能初学采集间隔不稳定的情况，比如预计1秒采集25帧图像，但实际采集了20帧，而音视频的时间戳是通过帧间隔累计的，这样计算出来的时钟就不正确了。
(2)帧间隔涉及到无限小数时，比如预计30帧，通常按帧间隔33毫秒处理，但实际是33.3333333毫秒。累积3333帧（约111秒）就出现1秒的误差。

1.2 采集时间戳-直接系统时间模式



frameTimeStamp = systemTime -startTime

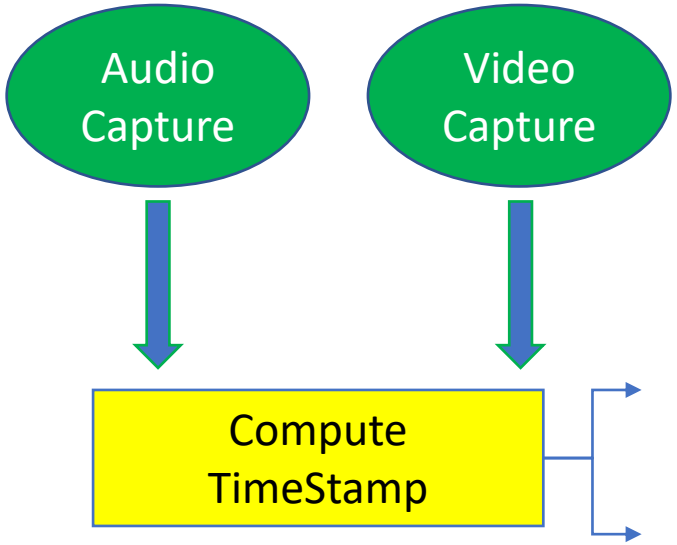
起始时间 =系统时间

时间戳 = 系统时间 – 起始时间

优点：能够实时纠正时间戳，只要系统正常运转，就能立即恢复正确的时间戳。

风险：帧间隔不均匀，能否正常播放依赖于终端。
比如，假如音频一帧间隔为24毫秒，被采集的回调时间可能为20毫秒，28毫秒，27毫秒，21毫秒。

1.2 采集时间戳–帧间隔+直接系统时间模式



frameTimeStamp = lastFrameTimeStamp + frameDuration。

起始时间 =系统时间

第一帧时间戳 = 系统时间 – 起始时间

后续帧 =上一帧时间戳 + 帧间隔
T = 当前系统时间 –起始时间
If(|后续帧 – T|>=帧间隔/2)后续帧 =T

优点：大部分音视频帧的frame duration都稳定。

风险：纠正时间戳时可能会造成画面卡顿的感觉。

1.3 音视频编解码模块

关键点：

1. 编码前：pts
2. 编码后：dts (发送的时间戳)

3. 封装成avframe送编码器 后frame的释放

4. 接收packet后数据的释放

1.4 音视频队列的控制

1. 音频帧数量
2. 视频帧数量
3. 视频超过阈值时drop帧
    1. 直到检测到I帧停止
4. 音频超过阈值时：
    1. drop帧；
    2. 重采样提高缩短播放时间再发送（不能简单进行重采样，需要变速不变调 sonic）

1.5 关键时间点

关键点：

1. 采集到的第一帧视频时间
2. 采集到的第一帧音频时间
3. 视频第一帧编码时间
4. 音频第一帧编码时间
5. rtmp发送第一帧视频完成时间
    1. sequence 帧
    2. i帧
6. rtmp发送第一帧音频完成时间
    1. sequence帧
    2. 数据帧

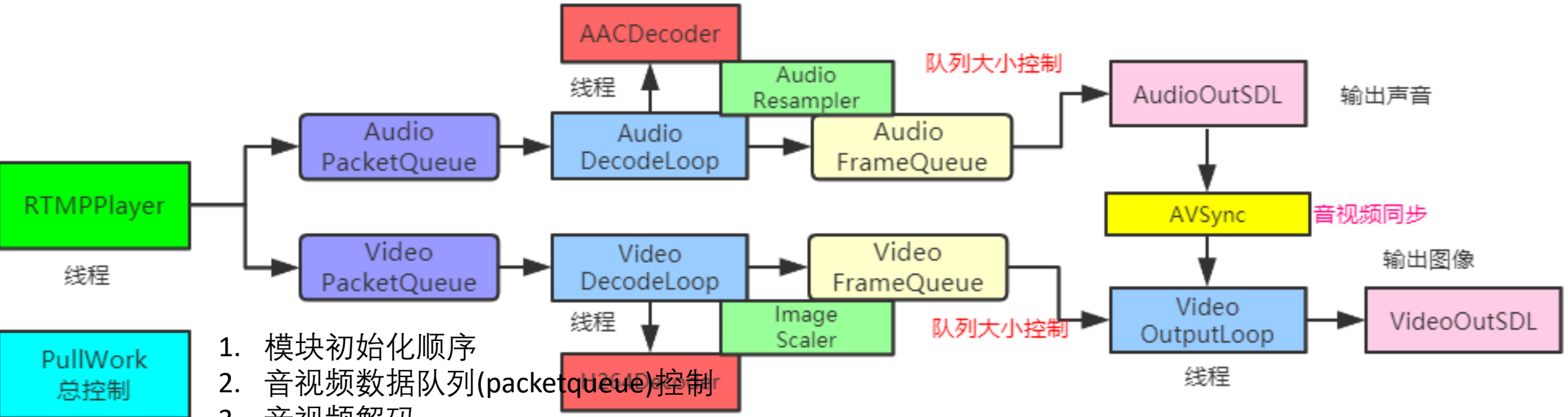
1.6 其他

1. 发送数据阻塞检测，在RTMP_SendPacket函数前后加打印监测发送耗时，比如耗时超过200ms打印警告。
2. av队列 队列的监测，帧数， 队列数据持续播放的时间

1. 推流后，拉流端拉流不正常：
   A. 使用ffplay或者vlc进行拉流double check，根据不同的错误信息进行排查
      ① 如果是编码错误则需要dump编码前和编码后的数据，用ffplay进行播放；
      ② 如果是时间戳的问题则需要调整发送时间戳，检测时间戳是否是递增

第二章 拉流实战

2 拉流框架



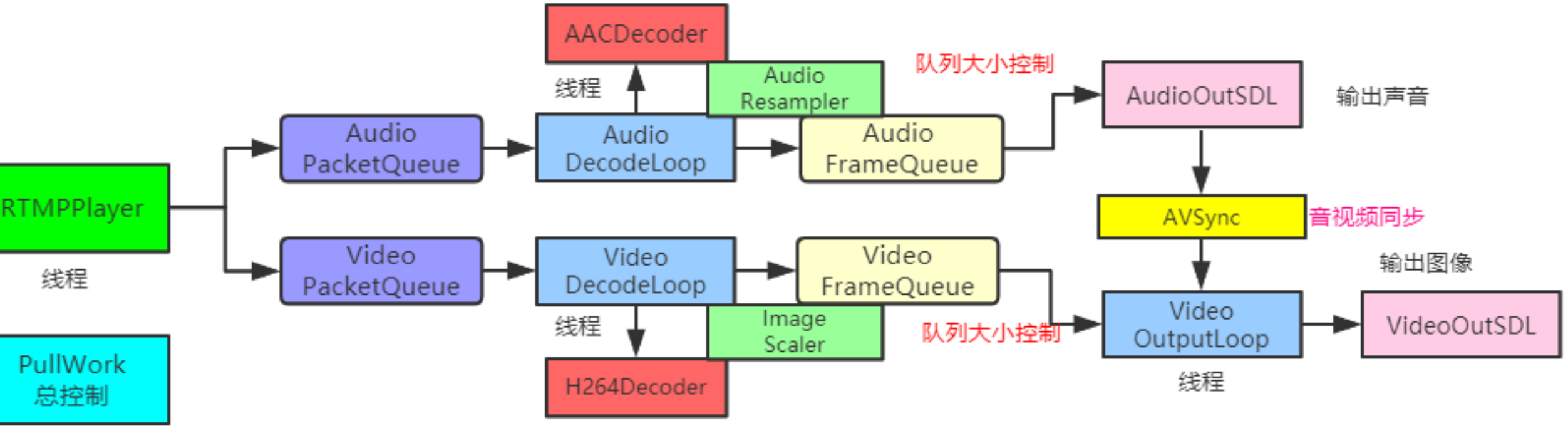
1. 模块初始化顺序
2. 音视频数据队列(packetqueue)控制
3. 音视频解码
4. 音频重采样
5. 视频尺寸变换
6. 音视频帧队列
7. 音视频同步
8. 关键时间点check
9. 其他

2.1 模块初始化顺序

推流模块（网络连接耗时） > 音视频编码模块 >音视频采集模块（）

音视频输出模块 >音视频解码模块 > 拉流模块

本质上来讲，就是在数据到来之前准备好一切工作



2.2 音视频数据队列

音视频队列涉及到
1. Audio PacketQueue 还没有解码的
2. Video PacketQueue

两者独立

队列设计要点：
1. 可控制队列大小
    1. 根据packet数进行控制
    2. 根据总的duration进行控制 音频48khz，21.3毫秒, 21.3*20 = 426ms
    3. 支持packet的size进行入队列累加，出队列**则减,** 300,200,400, 字节累加
2. 支持packet数量统计
3. 支持packet的duration进行入队列累加，出队列则减
4. 支持阻塞和唤醒

目的：
1. 统计延迟（缓存时间长度）

2.2 音视频数据队列

音视频队列涉及到
1. Audio PacketQueue
2. Video PacketQueue

两者独立

队列设计要点：
1. 可控制队列大小
    1. 根据packet数进行控制
    2. 根据总的duration进行控制
2. 支持packet数量统计
3. 支持packet的size进行入队列累加，出队列则减
4. 支持packet的duration进行入队列累加，出队列则减
5. 支持阻塞和唤醒

2.3 音视频解码

关键点：

1. 编码前： dts
2. 编码后： pts
3. packet释放
4. frame释放
5. 返回值处理

2.4 音频重采样

音频重采样模块AudioResampler：

注意重采样后不能直接使用linesize进行大小判断，需要使用

```
int size = av_get_bytes_per_sample((AVSampleFormat)(dstframe->format))
        * dstframe->channels * dstframe->nb_samples ;
```

2.5 视频尺寸变换

图像尺寸变换模块ImageScaler：变换尺寸大小

性能的提升可以考虑 libyuv

2.6 音视频解码后帧队列

FrameQueue

解码后的数据量比较大，需要要控制解码后帧队列的大小

参考ffplay固定大小

```
#define VIDEO_PICTURE_QUEUE_SIZE  3      // 图像帧缓存数量
#define SUBPICTURE_QUEUE_SIZE     16     // 字幕帧缓存数量
#define SAMPLE_QUEUE_SIZE         9      // 采样帧缓存数量
#define FRAME_QUEUE_SIZE FFMAX(SAMPLE_QUEUE_SIZE,
FFMAX(VIDEO_PICTURE_QUEUE_SIZE, SUBPICTURE_QUEUE_SIZE))
```

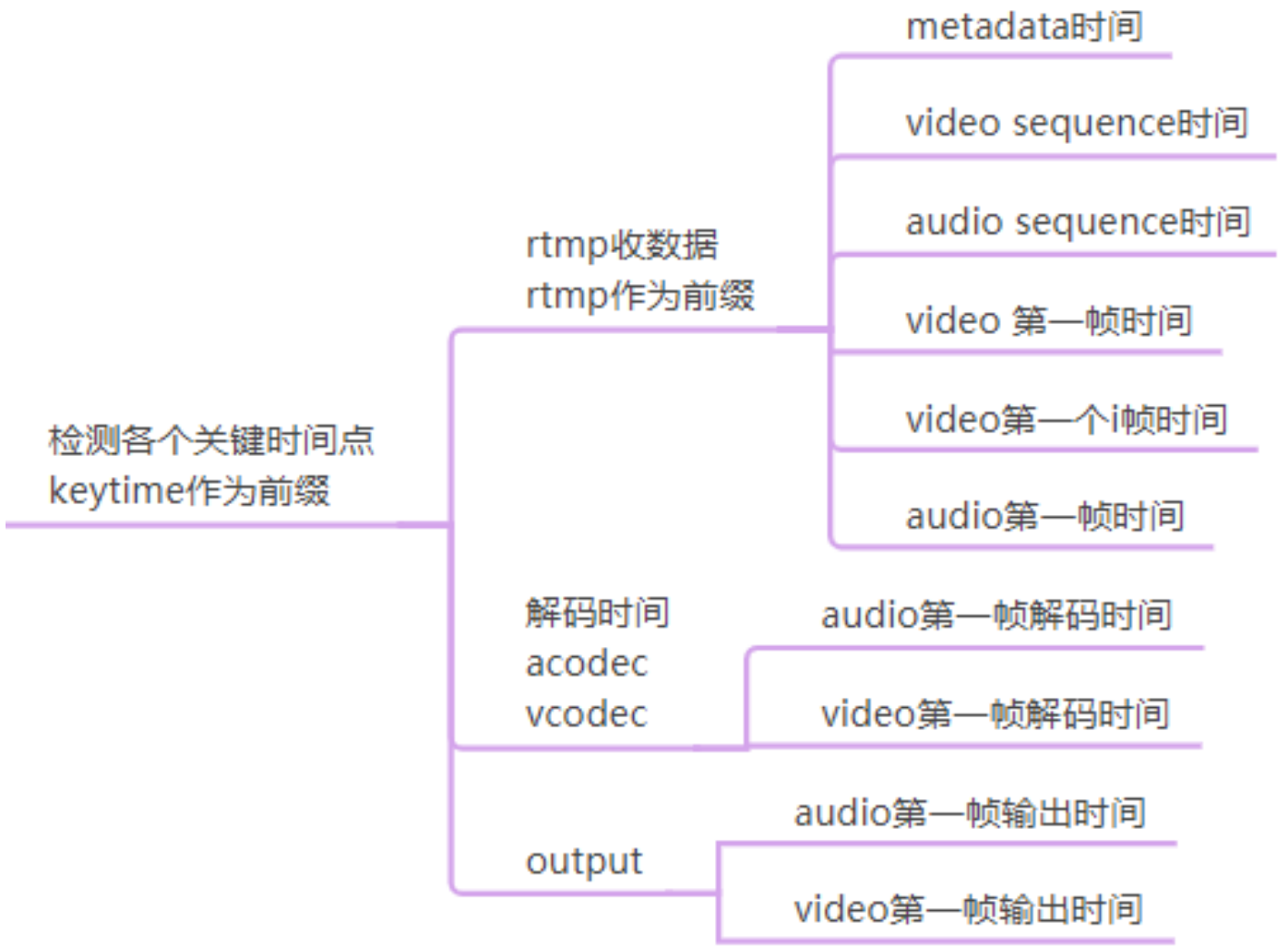
```
typedef struct Frame {
    AVFrame      *frame;          // 指向数据帧
    AVSubtitle   sub;             // 用于字幕
    int      serial;              // 帧序列，在seek的操作时serial会变化
    double       pts;             // 时间戳，单位为秒
    double       duration;        // 该帧持续时间，单位为秒
    int64_t      pos;             // 该帧在输入文件中的字节位置
    int      width;               // 图像宽度
    int      height;              // 图像高读
    int      format;              // 对于图像为(enum AVPixelFormat),
                                  // 对于声音则为(enum AVSampleFormat)
    AVRational   sar;             // 图像的宽高比，如果未知或未指定则为0/1
    int      uploaded;            // 用来记录该帧是否已经显示过?
    int      flip_v;              // =1则旋转180,  = 0则正常播放
} Frame;
```

2.7 音视频同步

Avsync模块

目前只支持audio master的方式。

2.8 各个模块关键时间点的监测



2.9 其他

1. 客户端的首帧秒开，本质上就是不做同步先把第一帧显示出来。

2. 推流没有问题时，如果拉流不能正常播放：
    1. 没有声音：dump rtmp拉流后的数据是否可以正常播放
    2. 声音异常：是否有解码错误报告，重采样前的pcm数据是否正常
    3. 没有图像： dump rtmp拉流后的数据是否可以正常播放
    4. 画面异常：是否有解码错误报告，scale前的数据是否正常

服务器首帧秒开：这个功能不能降低延迟