自动化课程制作服务在 π 项目中的应用实践

通过以动作为基本单元进行课程自动化制作是我们在课程制作提效这件事情上一直在努力的方向。之前在瑜伽项目也有过相应的尝试，但是也仅仅是浅尝辄止的状态，我觉得原因有二：

- 一是既有课程体系的兼容性没有得到解决
- 二是技术实现的思路上还有限制，没有突破关键节点

与此同时，我们还面临着用户播放卡顿、音画不同步、无法播放课程等用户课程播放上的问题，导致这些问题的原因不尽一致，但归根结底还是视频与用户的终端播放设备不匹配。借着新项目没有历史包袱的东风下，重启课程制作自动化的工作，同时将用户播放问题也纳入这个工作需要解决的问题列表中。

上面我们说在瑜伽项目中自动化课程制作的浅尝辄止的原因有技术实现思路上的限制没有突破的因素在，同样的在π项目中实践时，也走了一些弯路，碰到了一些棘手的问题。

标准问题

这个问题在处理课程期间是碰到最多的问题，具体表现有以下几个方面：

1. 动作视频的分辨率、码率、帧率这些参数不统一
2. 单个动作的标准交付物有哪些，规则是怎样的，哪些信息是可更新可变的，哪些信息是不能更改的
3. 输出流没有统一标准，要输出哪些分辨率，这些分辨率对应的码率、帧率是怎样的
4. 课程播放时的规则是怎样的，客户端需要怎样的分辨率以支持业务需求，客户端播放器能支持到的时间轴精度是怎样的

所以最重要的第一步就是要定标准，让一切都在标准下执行，这样方案使用的技术不会出现大的偏差：

- 动作标准交付物
    - 动作视频标准：
        - `H.264 编码，其中 H.264 的 Profile Level 参数建议统一为 Main 或 High`
        - 仅有一个视频轨道

        - 同一个应用的动作库分辨率、色值、结构最好是一致的
    - 动作音频标准：
        - 仅有一个 MP3 编码 128k 比特率的音频轨道
        - 采样使用 fltp 44100Hz 的参数
        - 采用 stereo 立体声声道，音量统一为一个值
    - 动作视频名称与音频名称需要保持一致，且不能更改
    - 动作的播放信息，如倒计次的时间节点、倒计时多少秒等
- 输出流标准
    - 支持四种分辨率的输出，并且指定不同分辨率的码率、帧率参数

Resolution	bit rates	frame rates	audio bit rates	remark
480p (854x480)	625 k	25	128 k	
720p (1280x720)	750 k	25	128 k	
1080p (1920x1080)	1500 k	25	128 k	
1080pvh (1920x1080)	不定	50	128 k	原始流

    - π 项目由于支持了视屏播放的横竖屏转换，所以没有采用标准的 16:9 分辨率，而是采用了 1:1 的分辨率，输出流以窄边为准，如 720p 的分辨率就是 720 x 720，而不是 1280 x 1280
    - 单个动作时长必须为整数，即时间轴精度为秒，这个是由客户端播放器的限制导致，如果精度为毫秒，客户端则会出现吞音等视频播放问题

课程合成方案问题

在前期的预研阶段，我们基本确定了输出流为 HLS 多码率的方式来提升播放设备的兼容性的技术方向，所以课程的合成方案也是介于此，形成了三种预期可行的方案：

- 先切片，后合并（A）；这种方式的主要流程是先将基本单元动作先切分为支持所有分辨率的 mpgets 格式文件与 HLS 播放列表，然后根据课程的编排规则，将 ts 格式文件组装到一个新的 HLS 播放列表中，从而形成课程的多码率播放列表
- 先合并，后切片（B）；这种方式的流程是使用动作的原始视频进行课程单节动作的构造，然后将课程编排的动作列表合并为一个课程视频，然后将这个视频进行切分，并生成 HLS 播放列表的生成

- HLS 课程包（C）；这种方式就是瑜伽之前使用的课程包方式的变体；将单个动作首先切分并生成 HLS 多码率播放列表，然后将播放列表链接与动作播放配置打包为一个配置文件，客户端拉取这个配置文件解析后本地进行播放

下面从灵活性、扩展性、准确性、播放体验、耗时等几个维度对三种方案进行对比：

plan	灵活性/扩展性	准确性	生成耗时	播放体验
Plan A	中	低	低	中
Plan B	高	高	高	好
Plan C	低	高	低	好

A 方案由于将动作支持的多码率切片文件与播放列表提前生成了，所以在多分辨率、多码率的扩展性上表现一般，每次都要重新对每个动作进行处理以支持后续的课程编排工作；虽然该方案有着灵活性高、耗时低的优点，但是在准确性与播放体验的表现不尽人意，此外，该方案还有其他缺点：

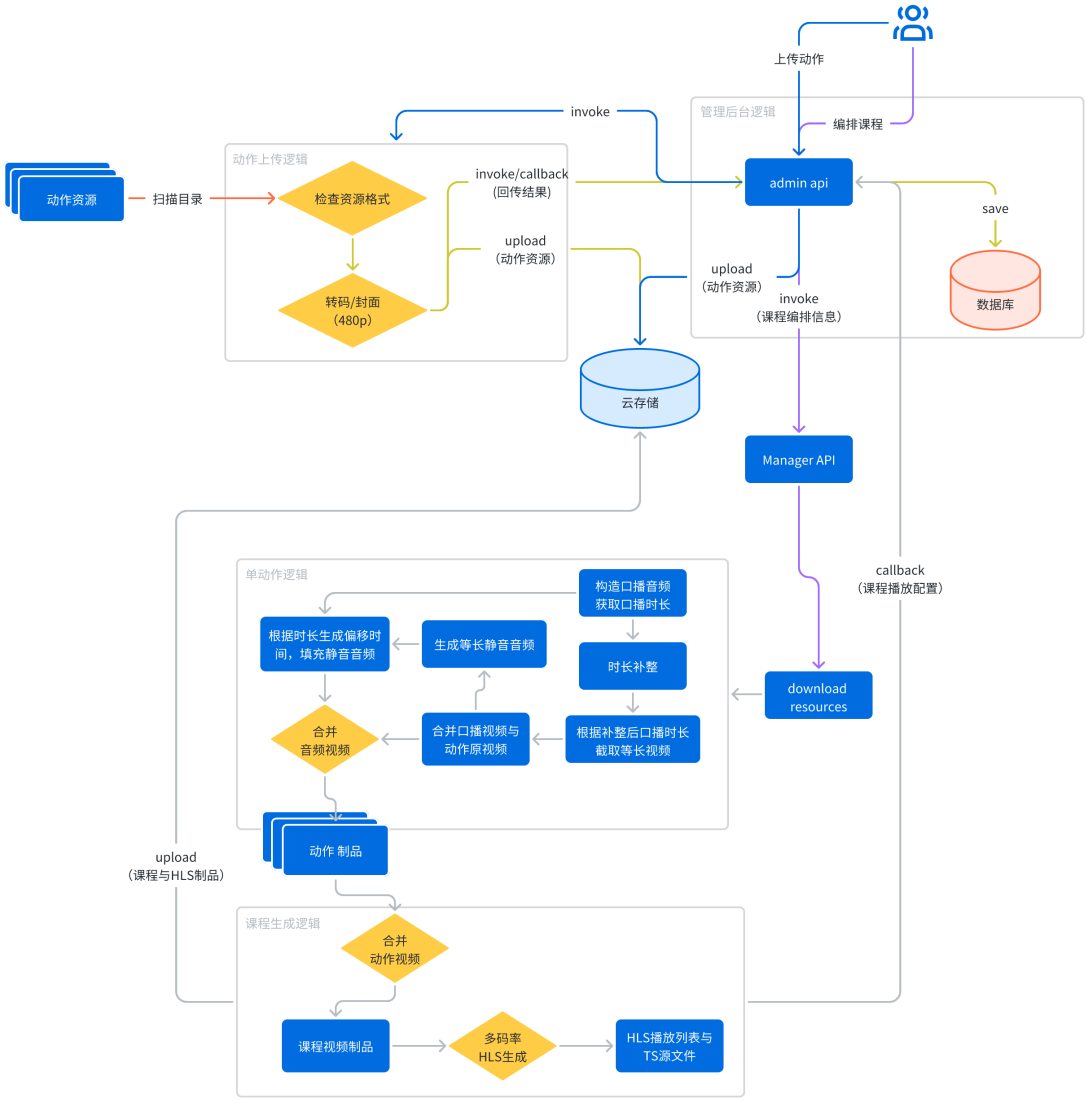
- 无法精确控制时间轴，只能精细化到秒，而且该精度也是不够准确的
- 由于切片提前生成，导致后续的所有的操作，引导语等都必须配合切片的时长
- 在生成播放列表时动作间的切换可能不受控制，在部分播放器上出现跳帧的现象
- 无法支持下载到本地与投屏的功能

C 方案由终端设备进行视频的播放，且动作仅做了切片处理，所以在播放体验与准确性上是非常有优势的，并且耗时也是最低的，可以直接在后台进行配置的生成，不需要单独的程序处理。但这个方案也与 Plan A 一样，在多分辨率、多码率的扩展性与后期配置的灵活性上是比较差的，并且由于客户端进行播放，配置一开始是需要约定好所有情况的，这并不现实，后续的更新需要发版来解决，还需要兼容存量版本的播放，灵活性非常低。

B 方案是我们最终选择的执行方案，这个方案为了做到准确性、灵活性与扩展性，对动作视频进行了重新编码，同时每次动作音频也是每次生成课程时重新组织、编码，这就导致了该方案在课程生成时耗时最久的。优点有以下几点：

- 多分辨率、多码率可配置或改动较少代码就可以支持很好的灵活性与扩展性
- 因为重新进行了编码，所以在音频、视频的时间轴精确性上有很好的保证，如需要一个 7.6s 的音频，不会出现为一个 8.0s 的音频
- 客户端播放仅需要处理 UI 相关的播放配置即可，不影响课程视频的播放，播放体验也能得到一定的保证

总体流程可查看以下流程图：



同时上面的流程图也说明了，我们怎么处理“冷启动”的问题。一般情况下，动作的交付都是批量交付的，而不是一个动作一动作的交付，这就需要我们有一个批量处理动作视频的功能以提高动作交付的效率。主要流程有：

- 动作按照交付标准内容，交付至共享盘中，方便获取
- 获取到该批次全部动作列表，检测交付资源是否符合标准，如果服务上传到云存储中
- 将动作配置信息提交入库
- 触发动作处理服务，生成动作播放需要的内容，如低分辨率的视频、封面图等

流畅性问题

流畅性问题主要指的是设备的适配性。为了提高终端设备的适配性，需要服务端与客户端共同努力才可能做的不错。首先客户端指定视频播放标准，包括分辨率、码率、帧率等参数标准，然后服务根据客户端给的参数标准，输出不同的视频流供客户端选择。这样就可以解决大部分设备的播放体验问题，同时我们也需要考虑秒开的问题，让首次加载的数据更少、更快、更合适，鉴于此，我们有两个选择：

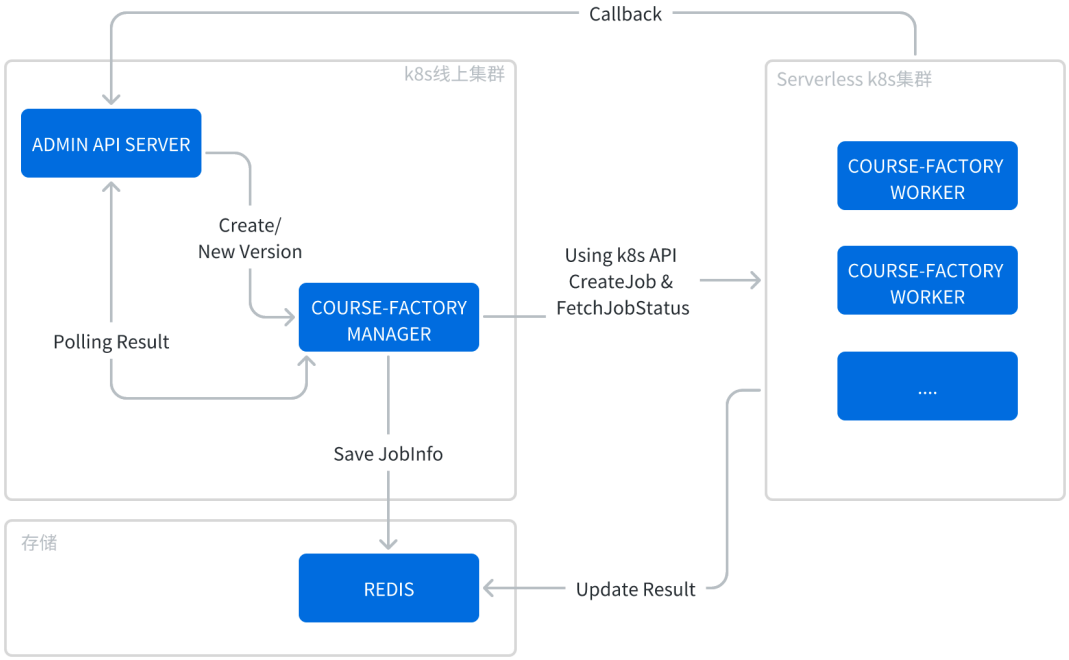
- HLS 是一个由 Apple 公司提出的基于 HTTP 的媒体流传输协议，用于实时音视频流的传输。目前HLS协议被广泛的应用于视频点播和直播领域，在移动端、H5端、服务端的相关软件支持是比较全面，比较容易接入。HLS 把整个流分成一个个小的基于 HTTP 的文件来下载，每次只下载一些。HLS 协议由三部分组成：HTTP、M3U8、TS。这三部分中，HTTP 是传输协议，M3U8 是索引文件，TS 是音视频的媒体信息。
- DASH 是一个相对比较新的流媒体协议。也是基于 HTTP 的动态自适应的比特率流技术，和 HLS，HDS 技术类似， 都是把视频分割成一小段一小段， 通过 HTTP 协议进行传输，客户端得到之后进行播放，不同的是，DASH 支持多种编码。但是配置相对 HLS 来说是比较复杂的，而且各端的支持性也不是特别好

综上，我们选择 HLS 作为最终的协议支持，可以在用户不同网络条件下自动切换分辨率以达到较好的播放体验，同时配置简单，还可以支持视频加密，可为后续数字版权的实施做支撑。

项目部署调用问题

最后一个问题是这个服务怎么部署，各个项目怎么接入。如果各个项目不能接入，那么自动化就仅只做了一半，所以，我们需要一个相对方便的方案，让不同项目可以快速接入课程制作自动化功能。

我们需要一个 Manager 管理服务，这个服务是接入层，主要服务接受各项目的课程制作请求及结果查询接口，并对负责课程生成的 Worker 进行调度，整体逻辑如下：



以上就是自动化课程制作服务在 π 项目中实践中遇到的一些问题，该服务已经完整的支持了 π 项目的首批课程上线，带来了不错的收益，视频成本大约可节省 45%，同时课程制作的中心也转移到了课程编排上，整体步骤由原来的 9 步缩减到了 3 步，效率也大大提升，由原来的 7 节课程需要3个工作日（即 24 小时）提高到 31 节课程仅需 2 个小时即可。