幻影时移系统

——基于切片模式的时移系统

一、概览：

随着网络视频技术的发展，直播从专业媒体已经开始在互联网上迅速的普级，越来越多的互联网公司开始搭建网络直播平台，也有越来越多人参与到网络直播中，基于直播形式的的商业模式也不断的涌现，网红直播、游戏直播、直播带货等等吸引了大量网民的参与，在为网络直播不断的赋予新的功能和技术的同时，做为电视台的传统业务形态“时移”的功能，大家却很少能够看到；时移是数字电视最重要的功能之一，在推出之日起就受到观众的普遍欢迎和喜爱，目前很多的数字电视和IPTV已经把“时移”做为一个标准服务，为广大的观众进行使用，可见其在电视屏幕上的重要性；随着越来越多的专业媒体在进行业务转型的同时，互联网的运营和推广也已经成为了业务互联网化的重要组成部分，提供专业的、高品质的内容是专业媒体的优势，所以基于互联网络的时移功能也是体现专业媒体必须提供的基本服务；

时移(TSOC), 是指观众可以任意回放过去时间的直播内容。时移是直播与点播相结合而形成的一种新业务，是直播业务的补充。用户在观看直播节目时，中途可以暂停，过后可以从暂停处继续收看，以免错过某些重要情节。从技术上讲，在暂停请求时，播放服务器把节目录下来，在继续播放时，再把录下的节目采用单播形式重新播放。

目前互联网时移技术实现大都是沿用数字电视技术框架，是基于流媒体方式，实现的成本非常高，并且架构复杂，不易于管理，也不适合规模化的发展，对于用户的高并发量访问也存在很严重的瓶颈问题；这也是目前时移业务没有在互联网上普遍使用的重要原因，寻找一种简单的、低成本、支持海量用户的时移功能的技术实现架构，是必需要解决的问题；基于切片模式的时移系统-幻影时移系统，是基于HTTP+HLS协议实现的时移系统，具有低成本、高可用、高并发等特点，系统的对接工作很低，后期的运营成本与直播基本相同，不会业务运营后成本的大幅度提高，是目前较有竞争力的产品；

二、设计原则：

- **可靠性**

用软件系统规模越做越大越复杂，其可靠性越来越难保证。应用本身对系统运行的可靠性要求越来越高，软件系统的可靠性也直接关系到设计自身的声誉和生存发展竞争能力。软件可靠性意味着该软件在测试运行过程中避免可能发生故障的能力，且一旦发生故障后，具有解脱和排除故障的能力。本系统在设计阶段就考虑到这些因素，充分利用各种成熟的技术的产品，并实现集群化布署，能够实现对于高并发、高吞吐量的支撑，可以实现7x24小时不间断运行，具备很高的可靠性；

- **健壮性**

健壮性又称鲁棒性，是指软件对于规范要求以外的输入能够判断出这个输入不符合规范要求，并能有合理的处理方式。对于系统容错及兼容性做了非常多的处理，对于日常使用过程的数据错误，超时错误等都可以及时的进行解决。

- **可修改性**

提供完善和详细的设计与维护文档，供后期的人员进行维护和管理。

- **可测试性**

可以通过模拟直播环境进行功能的测试，也可以通过专用的手段进行大并发量的用户访问测试，产可以通过专用工具对于系统的关键模块进行极限测试。

- **效率性**

系统在支撑日常访问时，只受带宽和磁盘读写效率的硬件环境影响，对于CPU和内存的使用率非常低，能够大大的降低使用成本。

- **先进性**

系统基于成熟的协议及linux环境开发，并且经过算法和流程的优化，结构清晰，流程简单，具有非常高的可维护性；

- **可扩展性**

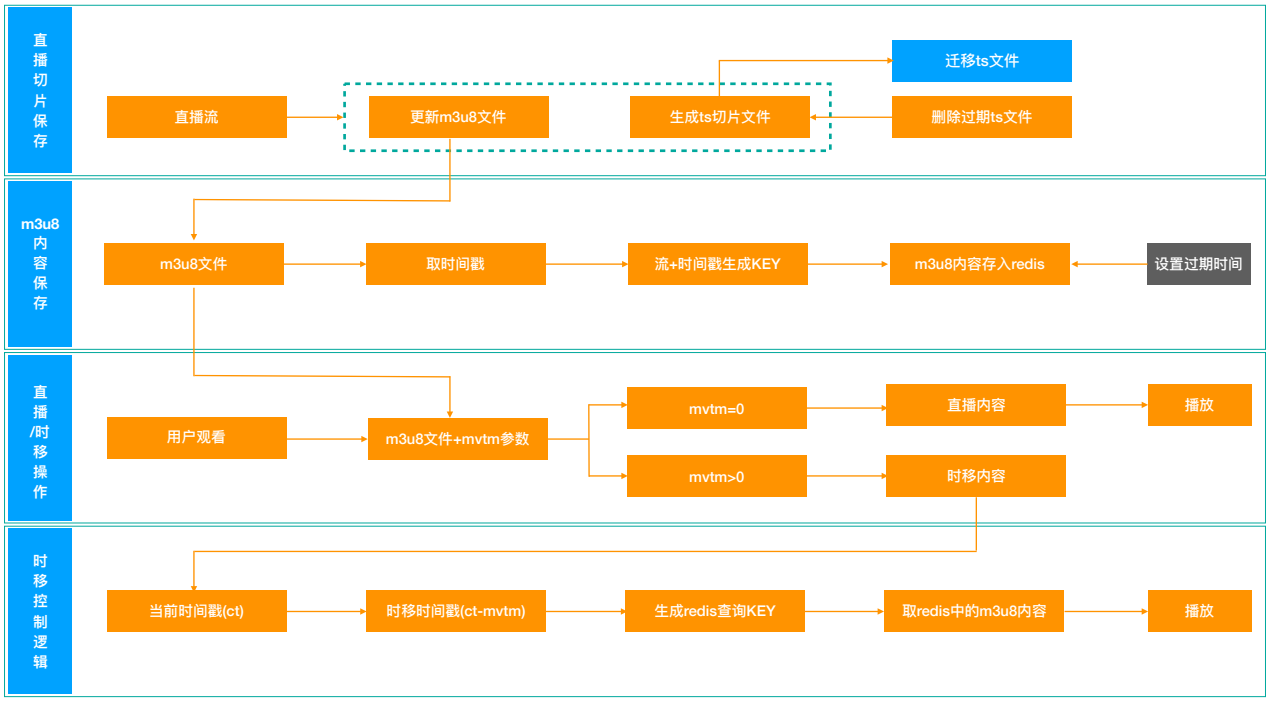
系统支持脚本开发环境，可以根据自已的需求进行调整和配置，能够随时扩展新的服务功能；

- **安全性**

系统安装的系统环境和开发的代码都经过严格的安全检测，并达到安全的要求；并可以对于用户鉴权系统进一步提升业务的安全级别，并可以随时进行软件的安全更新工作，不会影响业务的正常处理；

三、系统设计：

1) 系统架构



基于切片模式的时移系统-幻影时移系统

2) 架构说明

系统共划分为四个模块：直播切片保存模块、m3u8内容保存模块、直播/实移切换控制模块、时移控制逻辑模块；

直播切片保存模块：

负责RTMP流接收并基于HLS协议生成TS切片，直播切片TS文件生成后，默认保存在本地，过期的内容会删除，为了保证时移内容的可访问性，需要对于生成的TS切片进行全部保存，所以需要从本地迁移到永久存储中；直播切片保存模块就是实时的把切片内容同步到永久存储中；

m3u8内容保存模块：

HLS直播流访问的m3u8文件内容会定时进行更新，为了实现时移功能，需要把m3u8的文件内容每5秒钟(可调整)做一次保存，保存的内容放入到redis中；因为时移的内容不需要进行永久保存，所以对于m3u8内容存入redis中设置过期时间(默认20小时，可进行调整),过期后由redis进行自动删除；

直播/时移切换控制模块：

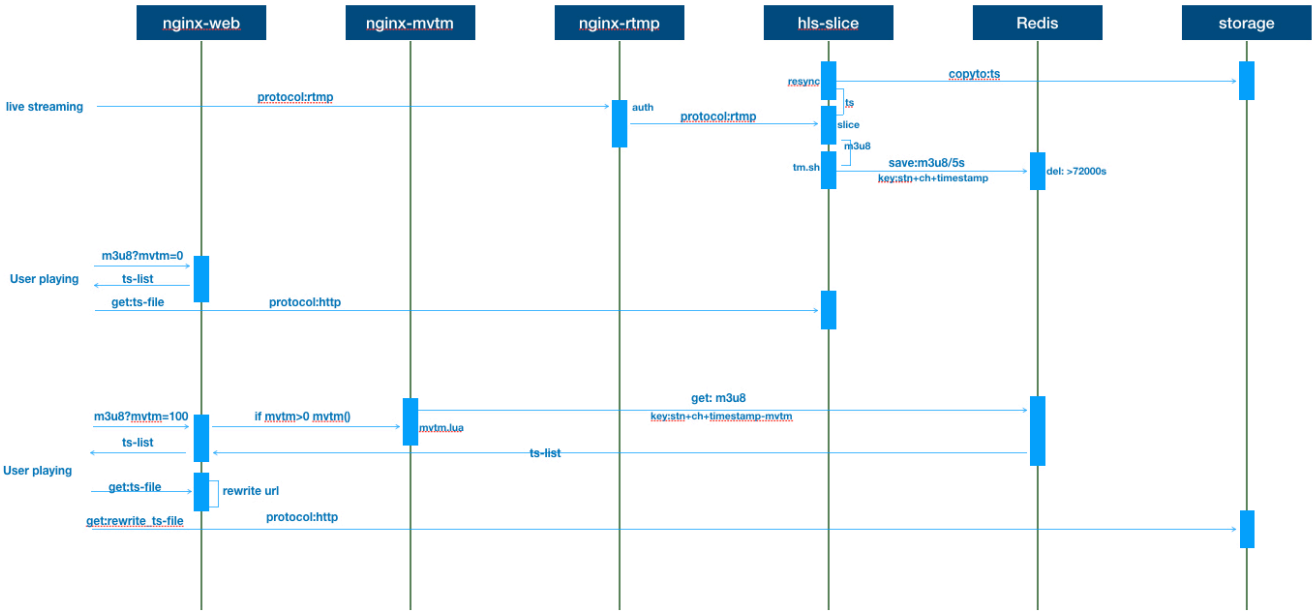
用户在使用播放器观看直播内容的过程中，如果向后拖动播放位置，则认为用户要使用时移功能，播放器需要在直播访问链接(.m3u8)后加入mvtm参数来重新请求播放内容。服务器接收到请求，如果mvtv>0则触发时移处理流程；如果mvtm=0则返回到实时直播内容；

时移控制逻辑模块：

进入时移流程后，根据时移时间计算获取时移m3u8内容的key值，根据Key值redis中获取m3u8内容后，需要根据实际情况对于TS片的访问路径做rewrite，重写的地址为永久存储区的访问地址或者是CDN可访问的资源路径；

四、主要流程介绍

1) 主要流程



2) 主要流程说明

直播流处理流程：

编码器推流进入nginx-rtmp模块，并通过流认证(auth)后，服务器进行流切片处理，系统的默认切片大小为5s (可根据实际需要调整，不要小于2秒)，hls-slice模块进行切片的同时会生成直播m3u8文件, m3u8文件中含5片ts记录，为了保证磁盘的可用性，hls-slice会根据m3u8文件的内容，不断的删除过期的ts文件内容(也可设置为不删除，但会增加磁盘的存储空间)；为了增加直播时移功能，需要对于ts内容和m3u8文件进行保存，在hls-slice模块中增加m3u8保存和ts的迁移工作，利用nginx-rtmp中的record功能来实现m3u8的内容定时存入redis中；该方法系统占用低，效率高，并且可靠性好；ts的迁移工作依靠inotify+resync来进行处理，该方式可以实时的把ts文件迁移到云存储或者NAS存储中，供时移过程中对于ts文件的正常访问；

观看直播处理流程：

用户通过播放器观看增加时移功能的直播与其它直播没有任何区别，通过请求直播的m3u8文件，加载相应的ts文件进行正常的播放；对于增加时移功能的直播服务来说，正常的直播内容观看就相当于时移的参数mvtm=0情况的处理；如果mvtm=0则认为是正常的直播内容，如果mvtm>0则认为是观看时移的内容；

时移处理流程

用户观看直播内容过程中如果向后拖动播放条，则认为用户要进行时移观看，播放器计算用户拖动的偏移距离的秒数（以秒为单位，如1小时=3600秒，mvtm=3600），在原直播观看的地址后加入mvtm=偏移时间，即可进行时移的观看；因为系统通过hls协议进行时移的实现。hls协议本身对于实时性的响应有一些延时，所以时移的定位点与正常的点会有几秒的误差，<5秒的误差都是在设计范围之内，如果时移业务对于误差率要求极高，则不建议使用本系统实现；

3) 关键程序说明

redis数据同步程序tm.sh

```
#!/bin/bash
path=$1
channel=$2
value=`sed ':a ; N;s/\n/\\n/; t a ;' $1`
##取当前时间戳, 每key为5秒; key=app_channel_timestamp； 每个ky期间5小时
p1=`date +%s`
p2=$(((${p1}/5)*5))
key=$2_$p2
p3=`redis-cli set $key "$value" ex 72000`
```

path= $1 ： 第一个参数是路径，就是生成m3u8的路径名；

channel=$2：第二个参数是通道名称(也可加流名称)，就是为了标识是那一个通道生成的m3u8文件，只要是唯一的就可以，这个参数和后面的mvtm.lua中的取redis的key匹配就可以；

value=`sed ':a ; N;s/\n/\\n/; t a ;' $1`：对于m3u8的多行内容，转换成一行内容，以方便写入到redis中；

p1=`date +%s`: 取当前的系统时间戳；

p2=$(((${p1}/5)*5)): 根据系统时间戳，生成每5 秒钟的时间分片；即系统每5秒钟保存一次m3u8文件；

key=$2_$p2: 根据通道名称和时间戳生成redis的key键值；

p3=`redis-cli set $key "$value" ex 72000`: 把m3u8的内容保存进本地的redis中，key过期时间为72000秒(20h)，过期时间可根据实际情况进行调整；

该程序一般需要增加进/usr/bin中，并且执行权限为777，能够通过nginx-rtmp模块进行正常的调用；

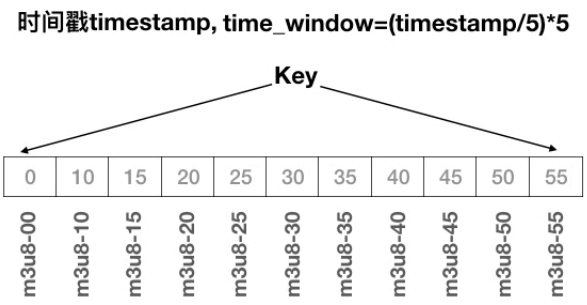
时移数据处理程序mvtm.lua

```
local function close_redis(red)
    if not red then
        return
    end
    local ok, err = red:close()
    if not ok then
        ngx.say("close redis error : ", err)
    end
end

local function getKey(p_ch, p_app, p_t1, p_t2)

    t1,t2 = math.modf((p_t1 - p_t2)/5);
    t1 = t1 * 5
    redkey = p_ch.."_"..p_app.."_"..t1
    return redkey
end

local args = ngx.req.get_uri_args()
local app= args["app"]
local ch= args["ch"]
local time2 = os.time()
local mvtm = args["mvtm"]
local redis = require(“resty.redis")
--创建实例
local red = redis:new()
--设置超时（毫秒）
red:set_timeout(1000)
--建立连接
local ip = "127.0.0.1"
local port = 6379
local ok, err = red:connect(ip, port)

if not ok then
    ngx.say("connect to redis error : ", err)
```

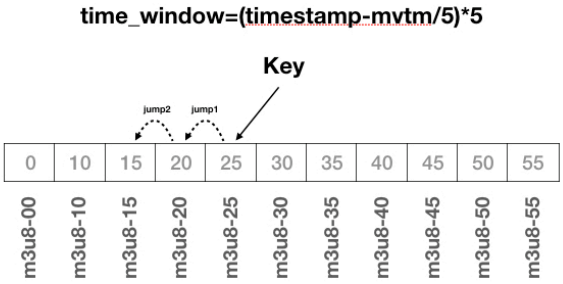
```
    ngx.exit(404)
    return close_redis(red)
end
ngx.header.content_type = "text/plain"
--调用API获取数据
local resp, err = red:get(getKey(ch, app, time2, mvtm))
if not resp then
    ngx.say("get msg error : ", err)
    ngx.exit(404)
    return close_redis(red)
end
--得到的数据为空处理,则再重试二次;
if resp == ngx.null then
    resp, err = red:get(getKey(ch, app, time2-5, mvtm))
    if resp == ngx.null then
        resp, err = red:get(getKey(ch, app, time2-10, mvtm))
    end
end
close_redis(red)
if resp == ngx.null then
    ngx.say("")
    ngx.exit(404)
else
    ngx.say(resp)
    ngx.exit(200)
end
```

该程序负责根据 mvtm传入的参数从redis中取出时移m3u8的内容，并返回给播放器；因为redis中的key值是按5秒钟时间片进行保存，所以需要根据当前的时间转换成5秒的片来进行提取；又因为有时会有处理时间的延误，所以当前的时间可能没有生成m3u8文件，为了保证系统的可使用性，如果在redis中没有保存请求时间片的内容，则自动向前5秒尝试取值，尝试两次(10秒)失败后才返回404错误；

m3u8按时间片保存策略：



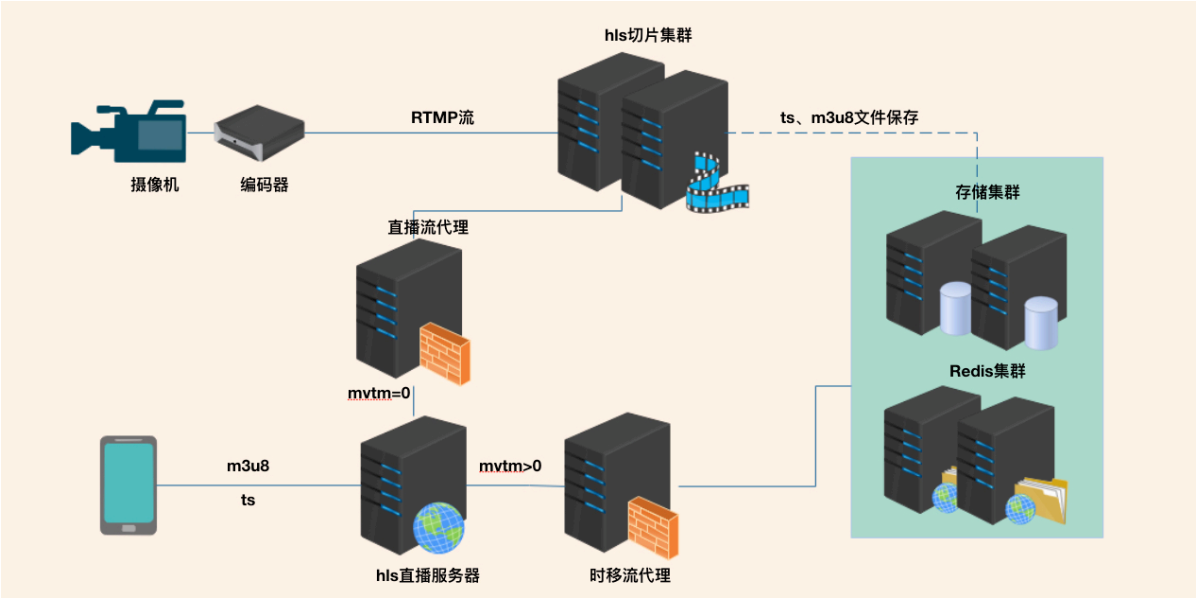
m3u8按时间片提取策略



通过时间片生成Key键，访问redis中保存的m3u8内容，如果该键值不存在，则最前一个5秒钟键值的内容，如果还不存在，则再访问前一个键值内容，设定只能提取前10秒钟内的键值(二次)，如果需要更改提取前键值的内容，则只须修改该部分的代码，重新加载即可完成；

五、安装与部署

1) 网络拓扑结构图：



2) 环境说明：

硬件环境：

服务器：时移系统是依赖于直播系统进行搭建，只是在直播系统的基础上进行了扩充，所以所有的服务器都是直播的服务器，只是在软件上做相应的调整即可，无需新增服务器；

操作系统：linux

软件环境：nginx1.9.4

nginx-rtmp-moudle

lua

redis5.0

hls切片服务器(集群)安装与配置

安装环境：

1、安装nginx+nginx_rtmp_moudle（如果以前支持，不需要重新安装）；

2、安装本地redis(使用远程redis服务，可以只安装redis-cli)

3、拷贝 tm.sh到/usr/bin，并修改权限(chmod 777 tm.sh);

配置nginx.conf：

启用rtmp中的收录功能，例：

```
application myapp {
        live on;

        hls on;
        hls_path /home/thumbnail/myapp;#视频流存放地址
        hls_fragment_naming system;
        hls_fragment 5s;
        hls_playlist_length 30s;
        hls_continuous on; #连续模式。
        hls_cleanup off;   #对多余的切片进行删除。
        hls_nested on;    #嵌套模式。
    #下面内容是为了实现时移功能进行的设置，利用每5秒5秒左右生成录收文件后，调用exec来定时保存m3u8文件；该方式侵入性小，比较灵活；
        record keyframes;
        record_path /home/thumbnail;
        record_max_frames 1;
        record_interval 5s;
        record_suffix _myapp.flv;
        exec_record_done tm.sh $dirname/myapp/$name/index.m3u8 $basename;
      }
}
```

时移流代理服务器(集群)安装与配置

安装环境：

1、安装nginx+nginx_rtmp_moudle+lua+lua_resty_redis（如果以前支持，不需要重新安装）；

2、拷备mvtm.lua 到/usr/local/nginx/lua 下(根据实际配置情况);

配置nginx.conf：

```
http {
   lua_package_path "/usr/local/nginx/lua/lua-resty-redis/lib/?.lua;;";
   server {
      listen   80;
#默认lua_code_cache是开的，为了调试现在关闭，运行时需要打开这个设置
lua_code_cache off;
     location /myapp {
          root    /home/thumbnail;
```

```
            #判断是否是直播或者时移业务，时移操作判断mvtm=字段是否存在，如果存在则认为是直播支持时移的，如果没有则认为不支持时移操作；
                if ( $arg_mvtm ~ \d*[1-9]\d* ) {
                    rewrite ^(.*)/(.*)/(.*)/(.*)$  http://192.168.2.81/mvtm$uri?app=$2&ch=$3 break;
                }
            }
            location ~ ^/mvtm/(.*).m3u8$  {
                #    root    /home/thumbnail;
                content_by_lua_file lua/mvtm.lua;
            }
            #在此对于ts文件做二次的重定向，因为保存在m3u8中的ts内容可能会因为迁移造成访问路径的变化，所以需要按实际的情况进行路径的重写，以保证ts文件能够被正确的访问到；
            location ~ ^/mvtm/(.*).ts$ {
                rewrite ^/mvtm/(.*)$ http://192.168.2.81/vod/$1 break;
            }
            #加入别名的处理是解决ts写入路径和访问路径的虚拟目录命名不一致的情况，可视实际情况进行调整；
            location /vod {
                alias /home/thumbnail/;
            }
        }
```

3) 验证与测试：

安装完毕后，启用编码器进行直播推流，并根据直播地址进行观看，如:

http://192.168.1.23/streamname/channel

把直播地址加上mvtm=300观看5分钟以前的时移内容

http://192.168.1.23/streamname/channel?mvtm=300

此文档访问地址：github.com/xiaobo0903/timemove