刚刚在线

分享iOS开发技术经验的自媒体网站

- RSS

Search

Navigate...

- 首页
- 全部文章
- iOS开发
- 实战
- 程序员
- 源代码
- sdk服务
- 推荐
- 赞助作者
- 头条

iOS视频边下边播–缓存播放数据流

Mar 8th, 2016 11:13 pm

google搜索“iOS视频变下边播”，有好几篇博客写到了实现方法，其实只有一篇，其他都是copy的，不过他们都是使用的本地代理服务器的方式，原理很简单，但是缺点也很明显，需要自己写一个本地代理服务器或者使用第三方库httpSever。如果使用httpSever作为本地代理服务器，如果只缓存一个视频是没有问题的，如果缓存多个视频互相切换，本地代理服务器提供的数据很不稳定，crash概率非常大。

这里我采用ios7以后系统自带的方法实现视频边下边播，这里的边下边播不是单独开一个子线程去下载，而是把视频播放的数据给保存到本地。简而言之，就是使用一遍的流量，既播放了视频，也保存了视频。

用到的框架：`<AVFoundation/AVFoundation.h>`
用到的播放器：`AVplayer`

先说一下avplayer自身的播放原理，当我们给播放器设置好url等一些参数后，播放器就会向url所在的服务器发送请求(请求参数有两个值，一个是offset偏移量，另一个是length长度，其实就相当于NSRange一样)，服务器就根据range参数给播放器返回数据。这就是大致的原理，当然实际的过程还是略微比较复杂。

下面进入主题

产品需求：

1. 支持正常播放器的一切功能，包括暂停、播放和拖拽
2. 如果视频加载完成且完整，将视频文件保存到本地cache，下一次播放本地cache中的视频，不再请求网络数据
3. 如果视频没有加载完（半路关闭或者拖拽）就不用保存到本地cache

实现方案:

1. 需要在视频播放器和服务器之间添加一层类似代理的机制，视频播放器不再直接访问服务器，而是访问代理对象，代理对象去访问服务器获得数据，之后返回给视频播放器，同时代理对象根据一定的策略缓存数据。
2. AVURLAsset中的resourceLoader可以实现这个机制，resourceLoader的delegate就是上述的代理对象。
3. 视频播放器在开始播放之前首先检测是本地cache中是否有此视频，如果没有才通过代理获得数据，如果有，则直接播放本地cache中的视频即可。

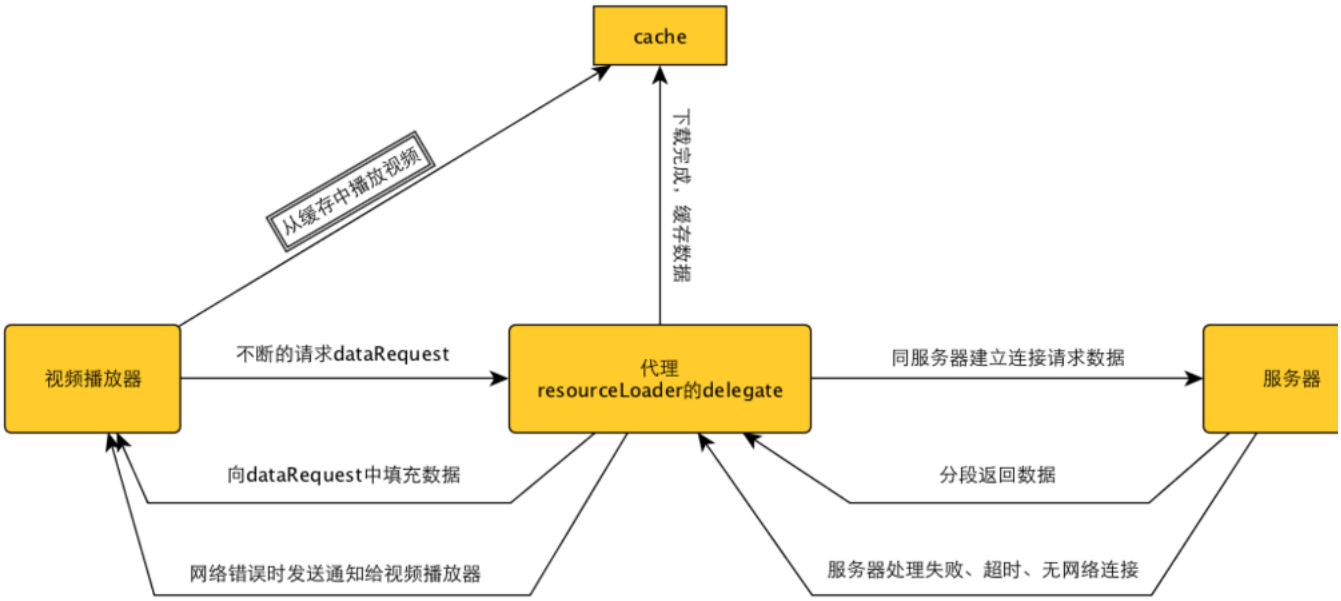
视频播放器需要实现的功能

1. 有开始暂停按钮
2. 显示播放进度及总时长
3. 可以通过拖拽从任意位置开始播放视频
4. 视频加载中的过程和加载失败需要有相应的提示

代理对象需要实现的功能

1. 接收视频播放器的请求，并根据请求的range向服务器请求本地没有获得的数据
2. 缓存向服务器请求回的数据到本地
3. 如果向服务器的请求出现错误，需要通知给视频播放器，以便视频播放器对用户进行提示

具体流程图



视频播放器处理流程

1. 当开始播放视频时，通过视频url判断本地cache中是否已经缓存当前视频，如果有，则直接播放本地cache中视频
2. 如果本地cache中没有视频，则视频播放器向代理请求数据
3. 加载视频时展示正在加载的提示（菊花转）
4. 如果可以正常播放视频，则去掉加载提示，播放视频，如果加载失败，去掉加载提示并显示失败提示
5. 在播放过程中如果由于网络过慢或拖拽原因导致没有播放数据时，要展示加载提示，跳转到第4步

代理对象处理流程

1. 当视频播放器向代理请求dataRequest时，判断代理是否已经向服务器发起了请求，如果没有，则发起下载整个视频文件的请求
2. 如果代理已经和服务器建立链接，则判断当前的dataRequest请求的offset是否大于当前已经缓存的文件的offset，如果大于则取消当前与服务器的请求，并从offset开始到文件尾向服务器发起请求（此时应该是由于播放器向后拖拽，并且超过了已缓存的数据时才会出现）
3. 如果当前的dataRequest请求的offset小于已经缓存的文件的offset，同时大于代理向服务器请求的range的offset，说明有一部分已经缓存的数据可以传给播放器，则将这部分数据返回给播放器（此时应该是由于播放器向前拖拽，请求的数据已经缓存过才会出现）
4. 如果当前的dataRequest请求的offset小于代理向服务器请求的range的offset，则取消当前与服务器的请求，并从offset开始到文件尾向服务器发起请求（此时应该是由于播放器向前拖拽，并且超过了已缓存的数据时才会出现）
5. 只要代理重新向服务器发起请求，就会导致缓存的数据不连续，则加载结束后不用将缓存的数据放入本地cache
6. 如果代理和服务器的链接超时，重试一次，如果还是错误则通知播放器网络错误
7. 如果服务器返回其他错误，则代理通知播放器网络错误

resourceLoader的难点处理

```
- (BOOL)resourceLoader:(AVAssetResourceLoader *)resourceLoader shouldWaitForLoadingOfRequestedResource:(AVAssetResourceLoadingRequest *)lo
{
    [self.pendingRequests addObject:loadingRequest];
    [self dealWithLoadingRequest:loadingRequest];

    return YES;
}
```

播放器发出的数据请求从这里开始，我们保存从这里发出的所有请求存放到数组，自己来处理这些请求，当一个请求完成后，对请求发出finishLoading消息，并从数组中移除。正常状态下，当播放器发出下一个请求的时候，会把上一个请求给finish。

下面这个方法发出的请求说明播放器自己关闭了这个请求，我们不需要再对这个请求进行处理，系统每次结束一个旧的请求，便必然会发出一个或多个新的请求，除了播放器已经获得整个视频完整的数据，这时候就不会再发起请求。

```
- (void)resourceLoader:(AVAssetResourceLoader *)resourceLoader didCancelLoadingRequest:(AVAssetResourceLoadingRequest *)loadingRequest
{
    [self.pendingRequests removeObject:loadingRequest];

}
```

下面这个方法是对播放器发出的请求进行填充数据

```
- (BOOL)respondWithDataForRequest:(AVAssetResourceLoadingDataRequest *)dataRequest
{
    long long startOffset = dataRequest.requestedOffset;

    if (dataRequest.currentOffset != 0) {
        startOffset = dataRequest.currentOffset;
    }

    if ((self.task.offset +self.task.downLoadingOffset) < startOffset)
    {
        //NSLog(@"NO DATA FOR REQUEST");
        return NO;
    }

    if (startOffset < self.task.offset) {
        return NO;
    }

    NSData *filedata = [NSData dataWithContentsOfURL:[NSURL fileURLWithPath:_videoPath] options:NSDataReadingMappedIfSafe error:nil];

    // This is the total data we have from startOffset to whatever has been downloaded so far
    NSUInteger unreadBytes = self.task.downLoadingOffset - ((NSInteger)startOffset - self.task.offset);

    // Respond with whatever is available if we can't satisfy the request fully yet
    NSUInteger numberOfBytesToRespondWith = MIN((NSUInteger)dataRequest.requestedLength, unreadBytes);


    [dataRequest respondWithData:[filedata subdataWithRange:NSMakeRange((NSUInteger)startOffset- self.task.offset, (NSUInteger)numberOfBy



    long long endOffset = startOffset + dataRequest.requestedLength;
    BOOL didRespondFully = (self.task.offset + self.task.downLoadingOffset) >= endOffset;

    return didRespondFully;


}
```

这是对存放所有的请求的数组进行处理

```
- (void)processPendingRequests
{
    NSMutableArray *requestsCompleted = [NSMutableArray array];  //请求完成的数组
    //每次下载一块数据都是一次请求，把这些请求放到数组，遍历数组
    for (AVAssetResourceLoadingRequest *loadingRequest in self.pendingRequests)
    {
        [self fillInContentInformation:loadingRequest.contentInformationRequest]; //对每次请求加上长度，文件类型等信息

        BOOL didRespondCompletely = [self respondWithDataForRequest:loadingRequest.dataRequest]; //判断此次请求的数据是否处理完全

        if (didRespondCompletely) {

            [requestsCompleted addObject:loadingRequest];  //如果完整，把此次请求放进 请求完成的数组
            [loadingRequest finishLoading];

        }
    }

    [self.pendingRequests removeObjectsInArray:requestsCompleted];   //在所有请求的数组中移除已经完成的

}
```

resourceLoader的难点基本上就是上面这点了，说到播放器，下面便顺便讲下AVPlayer的难点。

难点：对播放器状态的捕获

举个简单的例子，视频总长度60分，现在缓冲的数据才10分钟，然后拖动到20分钟的位置进行播放，在网速较慢的时候，视频从当前位置开始播放，必然会出现一段时间的卡顿，为了有一

在拖动到未缓冲区域内，是否需要加菊花转，如果加，要显示多久再消失，而且如果在网速很慢的时候，播放器如果等了太久，哪怕最后有数据了，播放器也已经“死”了，它自己无法恢复

有两个状态需要捕获，一个是正在缓冲，一个是正在播放，监听播放的“playbackBufferEmpty”属性就可以捕获正在缓冲状态，播放器的时间监听器则可以捕获正在播放状态，我的demo中一共有4个状态：

```
typedef NS_ENUM(NSInteger, TBPlayerState) {
    TBPlayerStateBuffering = 1,
    TBPlayerStatePlaying   = 2,
    TBPlayerStateStopped   = 3,
    TBPlayerStatePause     = 4
};
```

这样可以对播放器更好的把握和处理了。然后说一说在缓冲时候的处理，以及缓冲后多久去播放，处理方法：进入缓冲状态后，缓冲2秒后去手动播放，如果播放不成功（缓冲的数据太少，还不足以播放），那就再缓冲2秒再次播放，如此循环，看详细代码：

```
- (void)bufferingSomeSecond
{
    // playbackBufferEmpty会反复进入，因此在bufferingOneSecond延时播放执行完之前再调用bufferingSomeSecond都忽略
    static BOOL isBuffering = NO;
    if (isBuffering) {
        return;
    }
    isBuffering = YES;

    // 需要先暂停一小会之后再播放，否则网络状况不好的时候时间在走，声音播放不出来
    [self.player pause];
    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(2 * NSEC_PER_SEC)), dispatch_get_main_queue(), ^{

        // 如果此时用户已经暂停了，则不再需要开启播放了
        if (self.isPauseByUser) {
            isBuffering = NO;
            return;
        }

        [self.player play];
        // 如果执行了play还是没有播放则说明还没有缓存好，则再次缓存一段时间
        isBuffering = NO;
        if (!self.currentPlayerItem.isPlaybackLikelyToKeepUp) {
            [self bufferingSomeSecond];
        }
    });
}
```

这个demo花了我很长的时间，实现这个demo我也遇到了很多坑最后才完成的，现在我奉献出来，也许对你会有所帮助。如果你觉得不错，还请为我Star一个，也算是对我的支持和鼓励。

demo下载地址

```
https://github.com/suifengqjn/TBPlayer
```

文 / 夜千寻墨（简书作者）
来自：http://www.jianshu.com/p/990ee3db0563

Posted by 李刚 Mar 8th, 2016 11:13 pm 推荐

本文出处刚刚在线：http://www.superqq.com/blog/2016/03/08/ios-video-play-stream/

自由转载-请在开头注明本文出处。

微信公众号iOS开发：iOSDevTip

 « Objective-C和Swift混编的一些经验 iOS 9 App Thinning &ra/categories.htmlquo;

公众号Python开发：PythonPush

最新文章

- 关于NSRunLoop和NSTimer的深入理解
- iOS 9 App Thinning
- iOS视频边下边播–缓存播放数据流
- Objective-C和Swift混编的一些经验
- 刚刚在线–让学习iOS开发更简单

About Me

李刚：百度百家专栏作者，刚刚在线站长，iOS工程师非著名自媒体人，微信公众号iOS开发：iOSDevTip运营者

出师未捷名已落

新浪微博: 李刚移动

个人微信: **chinaligang** 欢迎调戏

iOS群: **218822587**

友情链接

- 刚刚在线教育
- 刚刚在线
- 程序员头条
- swift开发
- APP开发者
- 程序员的那些事
- Android开发精选
- Python开发中文网
- Linux开发中文网
- PHP开发中文网
- Java开发中文网
- WEB开发中文网
- HTML开发中文网
- 程序员聚合平台
- 技术小黑屋
- 庞海礁的个人空间
- PHP教程
- 干货集中营
- 阳和移动开发
- jquery教程
- WEB开发者
- 爱程序网
- IT技术文章

- 千一网络
- Bmob移动后端云
- 雷纯锋的技术博客
- 静觅
- 张飞的技术博客
- 一起Swift
- SwiftV课堂
- iOS开发
- AptuSource
- CSDN
- QQ空间
-

交换友链：欢迎各大程序员站点交换友情链接，如需交换，请添加好本站链接后发送邮件至下方邮箱。

格式：（友链文字：“刚刚在线”，链接：“http://www.superqq.com/”）

邮箱：worldligang@163.com