Files

ChangbaDevs / KTVHTTPCache

A media cache framework from Changba iOS Team.

172 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files

libobjc Create LICENSE

KTVHTTPCache.xcodeproj	update config
KTVHTTPCache	update KTVHCHTTPConnection
Vendors/CocoaHTTPServer	update server
demo/KTVHTTPCacheDemo	update config
.gitignore	add .gitignore
LICENSE	Create LICENSE
Readme.md	Update Readme.md

Readme.md

唱吧 iOS 音视频缓存处理框架

项目介绍

唱吧 iOS 团队为了解决音视频在线播放的缓存问题，开发了 KTVHTTPCache 这个框架。设计之初是为
的缓存问题，但其本质是对 HTTP 请求进行缓存，对传输内容并没有限制，因此应用场景不限于音视频
文件下载、图片加载、普通网络请求等场景。

技术背景

对于有重度音视频在线播放需求的应用，缓存无疑是必不可少的功能。目前常用的方案有 Local HTTP
AVAssetResourceLoader 两种。二者实现及原理虽有不同，但本质都是要 Hook 到播放器资源加载的
载逻辑。根据缓存状态，自行决定是否需要通过网络加载资源。从应用场景的角度看，二者有一个比较
配任意前端播放器，而后者只能配合 AVPlayer 使用。

个人认为，由于 AVAssetResourceLoader 是黑盒且会干预 AVPlayer 本身的播放逻辑，导致坑多且难
之间会有行为差异（例如近期发现在最新的 iOS 11 系统中，原本工作正常的代码，因为一个细小的行为
Bug），去适配它的逻辑会有不小的工作量。相反 Local HTTP Server 是完全 Open Source，我们能
辑，可以尽可能的规避缓存策略的引入带来的风险。

功能特点

- 支持相同 URL 并发操作且线程安全。
- 全路径 Log，支持控制台打印和输出到文件，可准确定位问题。
- 细粒度的缓存管理，可精确查看指定 URL 的完整缓存信息。
- 模块相互独立，提供使用不同 Level 的接口。
- 下载层高度可配置。
- 低耦合，集成简单。

结构设计 & 工作流程

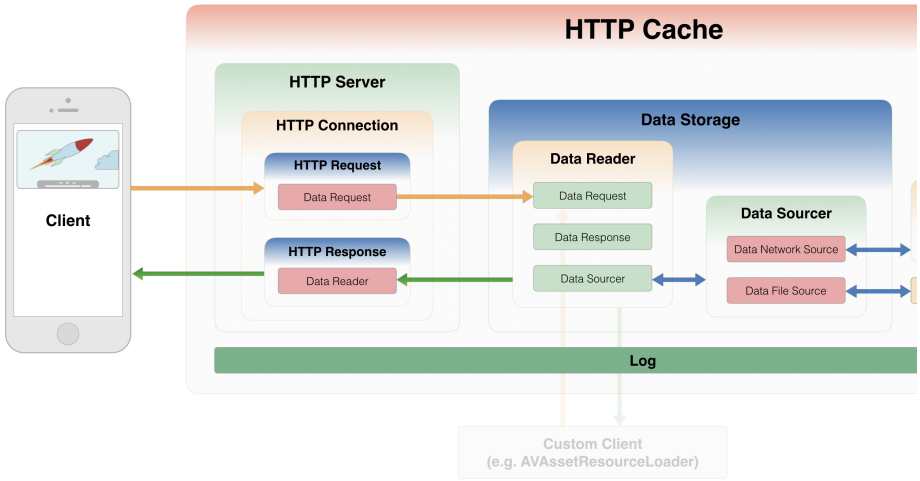
KTVHTTPCache 由 HTTP Server 和 Data Storage 两大模块组成。前者负责与 Client 交互，后者负责
为方便拓展，Data Storage 为独立模块，也可直接与 Client 交互（例如可与 AVAssetResourceLoad

结构及工作流程图如下：

Files
Search
Info
Settings

- KTVHTTPCache
- KTVHTTPCache.xcodeproj
- Vendors/CocoaHTTPServer
- demo/KTVHTTPCacheDemo
- .gitignore
- LICENSE
- Readme.md

Files



下面简述一下工作流程：

1. Client 发出的请求被 HTTP Srever 接收到，HTTP Server 通过分析 HTTP Request 创建用于访问 Request 对象。
2. HTTP Server 使用 Data Request 创建 Data Reader，并以此作为从 Data Storage 获取数据的通
3. Data Reader 分析 Data Request 中的 Range 创建对应的网络数据源 Data Network Source 和文 Source，并通过 Data Sourcer 进行管理。
4. Data Sourcer 开始加载数据。
5. Data Reader 从 Data Sourcer 读取数据并通过 HTTP Server 回传给 Client。

缓存策略

以网络使用最小化为原则，设计了分片加载数据的功能。有 Network Source 和 File Source 两种用于分别用于下载网络数据和读取本地数据。通过分析 Data Request 的 Range 和本地缓存状态来对应创

例如一次请求的 Range 为 0-999，本地缓存中已有 200-499 和 700-799 两段数据。那么会对应生成是：

1. Data Network Source: 0-199
2. Data File Source: 200-499
3. Data Network Source: 500-699
4. Data File Source: 700-799
5. Data Network Source: 800-999

它们由 Data Sourcer 进行管理，对外仅暴露一个 Read Data 的接口，根据当前的 Read Offset 自行选 Source。

使用示例

```
// 使用简单，基本可以忽略集成成本

// 启动（全局启动一次即可）
NSError * error;
[KTVHTTPCache proxyStart:&error];

// 使用
NSString * URLString = [KTVHTTPCache proxyURLStringWithOriginalURLString:@"原始 URL"];
AVPlayer * player = [AVPlayer playerWithURL:[NSURL URLWithString:URLString]];
```

唱吧的实践过程

方案演进

在音视频缓存上，我们一共采用过如下 4 个方案：

1. AVPlayer 纯在线播放。
2. AVPlayer + AVAssetResourceLoader + 下载模块。
3. AVPlayer + 一个开源的缓存项目（同样基于 AVAssetResourceLoader + 下载模块）。

Tutorial
Help
Sign In

Files

4. AVPlayer + KTVHTTPCache。

- 方案 1 简单直接，缺点也不必多说。
- 方案 2 的下载模块设计的比较简单，只能顺序下载，不支持分片。导致只能 Seek 到已下载完的地
较大的缺陷。
- 方案 3 在功能上已经可以满足需求，但在使用中问题较多，我们在源码基础上做了很多修改来填坑
理想，上线不长时间就将该功能下掉了。
- 方案 4 是唱吧现在的线上方案，目前在我们的使用场景中还没有发现问题。除稳定性的提升外，比
路径的 Log 模块。若用户或测试同学遇到问题，只需简单描述并回传 Log，就可以快速定位到原因
率。

踩过的坑

1. Content-Type 和 Path Extension

AVPlayer 在播放时会优先根据 Response Header 中的 Content-Type 判断当前资源是否可以播放。
给出有效信息时再去判断 URL 中的 Path Extension。

对应关系如下：

URL	Content-Type	是否可播
http://changba.com/video.mp4	video/mp4	YES
http://changba.com/video.mp4	application/octet-stream	YES
http://changba.com/video	video/mp4	YES
http://changba.com/video	application/octet-stream	NO

因此要想让 AVPlayer 正常播放，Content-Type 和 Path Extension 中至少能提供一个有效信息，否则

- 发现这一问题是因为在做 Original URL -> Proxy URL 映射时，将 Original URL 中的 Path Extens
Encode 时丢失了，再碰上某些情况 CDN 返回的 Content-Type 是 application/octet-stream 而
的确切类型时，AVPlayer 会直接报 Error。

2. 锁屏后 Server Socket 失效

在本地 Server 中有一个 Socket 用于接收 AVPlayer 发出的请求。如果在 AVPlayer 为非播放状态时锁
App，FD 虽然还在，但 Listen 的端口会被回收，导致 FD 接收不到事件，AVPlayer 发出的请求也就无
到。我们的解决办法是在做 URL 映射时 Ping 一下本地 Server，如果 Ping 不通，会重启本地 Server

最后

项目已经开源，GitHub 地址： https://github.com/ChangbaDevs/KTVHTTPCache

对重度影音类应用而言，音视频缓存属于比较重要的一环，对稳定性也有比较高的要求，我们在这上走
坑。希望 KTVHTTPCache 的开源能给大家带来一些帮助。也非常欢迎大家在项目中使用，如果遇到问
Issue 给我。