Android 平台中轻量级音视频引擎①

殷继平[1,2], 高 春[3], 梁崑涛[4], 刘 涌[4]

[1](中国科学院大学, 北京 100049)
[2](中国科学院沈阳计算技术研究所, 沈阳 110168)
[3](辽宁大学, 沈阳 110031)
[4](沈阳广播电视台, 沈阳 110004)

摘 要: 随着移动互联网络和移动终端设备硬件的快速发展, 移动音视频的应用越来越普遍, 这要求应用软件能高效的处理音视频文件以满足大众的需求. 为提高软件运行效率, 通过对多媒体处理流程的分析研究, 设计和实现了轻量级的音视频引擎, 通过模块化的设计方法, 实现了对媒体文件的接收、解析识别文件格式、分离容器中的音频和视频流并分别解码输出, 只使用 HTTP 传输协议和 H.265 视频解码、AAC 音频解码, 将软件的体积降到最小, 实现与 Android 平台交互 JNI 的接口, 在应用程序中使用 MediaPlayer 类实现音视频的播放.

关键词: Android; 音视频; 引擎; 轻量级; 解封装; MediaPlayer

Lightweight Audio and Video Engine for Android

YIN Ji-Ping[1,2], GAO Chun[3], LIANG Kun-Tao[4], LIU Yong[4]

[1](University of Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computing Technology of Chinese Academy of Sciences, Shenyang 110168, China)
[3](Liaoning University, Shenyang 110031, China)
[4](Shenyang Radio and TV Station, Shenyang 110004, China)

Abstract: With the rapid development of mobile Internet and mobile terminal equipment, mobile audio and video is applied more and more widely, the application software should process audio and video files efficiently to meet the needs of the public. For the purpose of improving the operational efficiency, this paper analysis and research multimedia processing, design and implement a lightweight audio and video engine, use the modular method, accomplished receive the media file, recognize the file format, separate containers into the audio and video streams and decode to output, use HTTP protocol and H.265 video decoder, AAC audio decoder, the software size is reduced to be smaller, then complete the Android platform interactive JNI, use the MediaPlayer of Android in the application to implement the audio and video play.

Key words: Android; audio and video; engine; lightweight; demux; MediaPlayer

随着 3G、4G 移动互联网络的快速发展和 WiFi 无线局域网的大量覆盖, 以 iPhone、Android 系列为代表的智能手机大量出现[1], 现代移动终端设备在 CPU 处理能力、内存大小、操作系统、存储容量以及屏幕尺寸方面都有了极大的发展, 智能终端的通信能力和处理能力越来越强大, 人们在把它们当做基本通讯工具的同时, 已逐步将其作为必备的信息获取及处理工具[2]. 与此同时, 人们对于广泛且易访问的音视频内容有着巨大且与日俱增的消费需求, 为满足这种需求并保持访问的易用性, 用于处理和传输这些多媒体内容的相关设备必须具备更高的承担能力, 尤其当音视频被传送到客户端设备上时, 处理能力的提升显得更为重要[3].

由于 Android、iOS 系统的多媒体框架对音视频文件支持的类型比较少, 所以可以移植开源的音视频编解码库以处理更多格式种类的媒体. 多媒体客户端的应用软件开发可将音视频功能模块抽取出来, 再进行独立的开发工作, 这种音视频模块被称为“音视频引

① 收稿时间:2015-04-07;收到修改稿时间:2015-05-15

擎”[4]. 开源项目中也有很多表现优异的音视频处理工具可以提供有力的支持, 但是这类工具普遍具有代码量庞大、处理过程复杂、使用时占用较大的运行内存、处理器占用率较高等问题, 不足以满足移动终端应用程序体积小、占用少、运行快等要求, 所以, 实现满足移动应用软件需求的轻量级的音视频引擎具有重要的理论研究意义和实在应用价值.

本文针对移动终端应用程序对音视频的应用需求, 设计并实现了轻量级的音视频引擎, 处理流程简单、内存和处理器占用更小, 编译并移植到 Android 平台开发项目中, 能够实现接收输入文件、解码特定格式的音频或视频文件并输出.

1 系统概述

1.1 整体结构

音视频引擎模块分为: 读文件模块, 主要实现读取移动应用传来的输入文件; 解封装模块, 识别文件类型, 分离出个媒体原始数据流; 解码模块, 解码音频或视频数据包; 颜色空间转换模块, 把视频解码器解码出来的数据转换成显示系统支持的颜色格式; 时间同步模块, 通过对比同步时钟信息分别排序音频数据帧和 PCM 数据; 输出模块, 将视频和音频数据传给上层应用程序. 音视频引擎的整体结构图如图 1 所示:

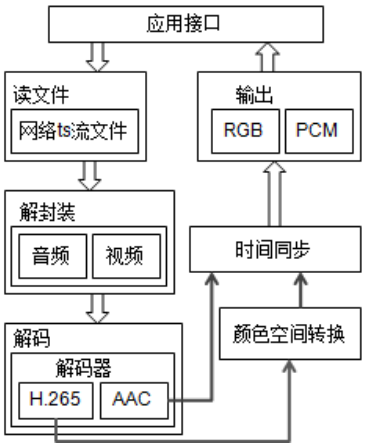

图 1 音视频引擎整体结构图

2 音视频引擎的设计

2.1 功能设计

音视频引擎在移动多媒体应用软件中主要完成对多媒体输入文件的分析和解码工作, 通过接口与上层应用进行交互, 在得到一个输入之后调用引擎中各个模块, 将处理后得到的数据输出给应用程序. 本文实现了此引擎在 Android 平台中的应用, 主要功能如下:

1)获取应用程序传来的数据流, 识别文件类型和媒体类型, 分离出原始流.

2)解码数据流, 传递同步时钟信息, 将解码器解码出的 YUV 数据转换成 RGB 格式[5], 并按时钟信息排序数据帧, 将相同时戳的图像信息和语言信息输出给应用程序.

3)使用先进的 H.265 视频解码和 AAC 音频解码技术[6], 将引擎编译后移植到 Android 平台应用中实现读取 HTTP 协议传输的网络流文件, 解码后显示输出.

音视频引擎的主要编程语言是 C 语言, 操作系统选择工作稳定性和可靠性均很高的 Linux 操作系统. 根据相应的功能需求设计并实现各个子模块.

2.2 音视频引擎实现机制

本引擎实现的几大功能是: 读文件、识别格式、音视频解码, 在设计上, 把每个大功能抽象成一个接口类的数据结构, 其中的功能函数指针在编译的时候能静态确定, 每一个功能都支持多种类型的广义数据, 本文把广义数据的共同部分抽象成对应的 Info 结构, 这些 info 的核心成员只能在程序运行时动态确定其值. 媒体引擎主要数据结构直接的关联图如图 2 所示:

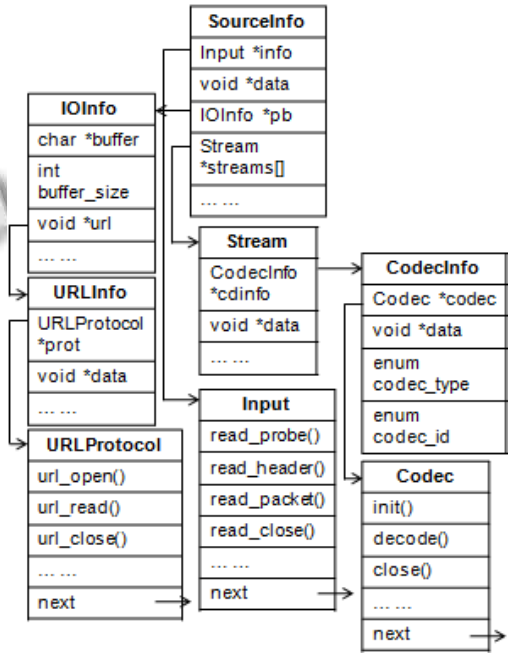

图 2 引擎数据结构关联图

为实现引擎的轻量级, 本文首先简化多媒体文件

处理流程，实现出代码框架，对外采用统一接口，再根据实际应用移植需要的解码器，重写解码单元的接口即可编译运行，实现了可扩展功能的同时将引擎的内存占用量降到最小.

2.3 主要模块中数据结构的实现

表 1 媒体引擎主要数据结构及说明

SourceInfo	描述媒体文件或流的构成和基本信息
Input	代表一个 URL 地址指向的媒体文件
Stream	存储每个视频/音频流信息
IOInfo	描述输入输出数据及存储位置
URLInfo	具体文件协议的相关数据描述
URLProtocol	广义的输入文件的结构
CodecInfo	保存流的详细编码信息，如视频宽高
Codec	存储编解码信息，包含调用的函数

2.3.1 读文件模块的实现:

读文件模块，可以从实现的顺序分为三层，最底层是 URLProtocol 这类具体的网络协议；中间抽象层用 URLInfo 结构来统一表示底层具体的网络协议，相关操作也只是简单的调用底层具体协议的支撑函数，最上层用 IOInfo 结构来扩展 URLProtocol 结构生成内部有缓冲机制的广泛意义上的文件，并且仅由 IOInfo 对模块外提供服务，结构图如图 3.

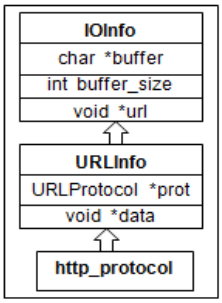

图 3 读文件模块结构图

URLProtocol 是接口类的数据结构，表示广义的输入文件，一种输入文件对应一个 URLProtocol 结构.

URLInfo 结构表示程序运行的当前广义输入文件使用的上下文，存储所有广义输入文件共有的属性和关联其他结构的字段. prot 字段关联相应的广义输入文件; data 字段关联各个具体广义输入文件的句柄.

IOInfo 结构扩展 URLProtocol 结构生成内部有缓冲机制的广泛意义上的文件，改善广义输入文件的 IO 性能. 由其数据结构定义的字段可知，主要是缓冲区相关字段，标记字段，和一个关联字段 url 来完成广义文件读写操作，url 关联字段用于关联 URLInfo 结构，间接关联并扩展 URLProtocol 结构.

2.3.2 解封装模块的实现:

解复用模块，可以简单的分为两层，底层是如 TSInfo 等具体媒体的解封装结构和相关的基础程序，上层是 Input 结构和相关的程序，上下层之间由 Input 相对应的 SourceInfo 结构的 data 字段关联 TSInfo 等具体的文件格式，结构如图 4.

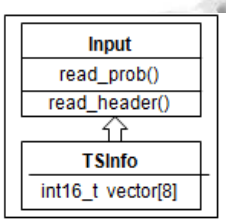

图 4 解封装模块结构图

Input 接口类的数据结构，表示输入文件容器格式，一种文件容器格式对应一个 Input 结构. data_size 标示具体的文件容器格式对应的 Info 的大小，这些具体的结够定义分布于各个.c 文件中.

SourceInfo 结构表示程序运行的当前文件容器格式使用的上下文，存储所有文件容器共有的属性(并且是在程序运行时才能确定其值)和关联其他结构的字段. input 字段关联相应的文件容器格式; pb 关联广义的输入文件; streams 关联音视频流; data 字段关联各个具体文件容器独有的属性上下文，和 data_size 配对使用.

2.3.3 解码模块的实现

解码模块，也可以简单的分为两层，由 Codec 统一表述. 上层是 Codec 对应的 CodecInfo 结构和相关的基础程序，底层是如 TSInfo 等具体的编解码器内部使用的结构和相关的基础程序. CodecInfo 结构的 data 字段关联 TSInfo 等具体解码器上下文，结构如图 5.

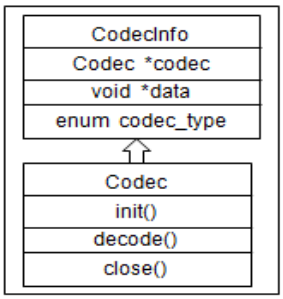

图 5 解码模块结构图

Codec 接口数据结构, 表示音视频编解码器, 一种媒体类型对应一个 Codec 结构, 在程序运行时有多个实例. next 变量用于把所有支持的编解码器连接成链表, 便于遍历查找; id 确定了唯一编解码器; data_size 表示具体的 Codec 对应的 Info 结构大小. CodecInfo 结构表示程序运行的当前 Codec 使用的上下文, 存储所有 Codec 共有的属性(并且是在程序运行时才能确定其值)和关联其他结构的字段.

2.4 媒体引擎在 Android 平台的实现

根据新媒体互动广播 Android 客户端项目应用的需要, 本引擎在实现中只支持网络 ts 流的接收和解析工作, ts 流中封装的是高效的 H.265 编码视频和 AAC 编码的音频流, 使用运行开销较低的 HTTP 协议传输, 由 Android 客户端实现网络码流的接收并将其传给引擎, 之间通过 JNI 技术连接, 引擎将文件处理后将视频图像和音频 PCM 数据传给 Android 应用, 由客户端实现图片的显示和声音的播放.

2.4.1 Android 的 JNI 技术

由于 Android 应用层使用 Java 作为开发语言, 要在应用层使用媒体引擎功能, 需使用 JNI 技术, JNI 接口的主要目的是使虚拟机 VM 可以通过 Java 函数找到按照 JNI 规定实现的与其对应的 C 函数实现, 在 Android 中使用了 Java JNI 的方法来实现本地函数, 以 Java 和 C 函数的映射表数组的形式来说明函数对应关系[7], 图 6 为媒体引擎在 Android 平台中的位置图.

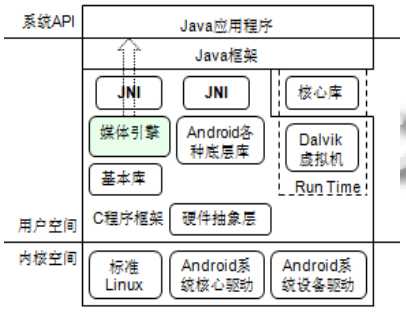

图 6 媒体引擎在 Android 平台中的位置图

2.4.2 音视频引擎的编译

在完成流媒体引擎的设计和程序是实现后, 对它进行交叉编译移植, 才能在 ARM 平台上运行[8], 同时由于在媒体引擎的解码层和解封装层中使用了开源库, 因此, 首先应交叉编译和移植这些开源库, 生成静态库, 给媒体引擎各层调用, 然后分别编译数据接收层、数据输出层、解码层和解封装层生成动态库, 调用这些动态库编译用户接口层, 最终实现将整个媒体引擎编译成 JNI 动态库, 通过 JNI 接口, 供 Java 层调用. 各个库之间的调用关系如图 7 所示.

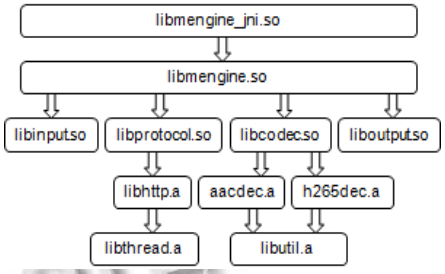

图 7 编译库调用关系图

2.4.3 音视频引擎的移植

编译后将媒体引擎库移植到 Android 平台应用中, 新媒体互动广播项目中音视频引擎的位置如图 8.

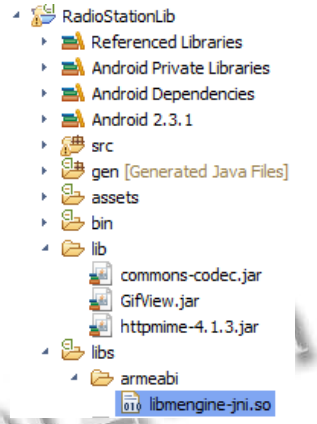

图 8 引擎移植到 Android 项目中结构图

2.4.4 Android 客户端的实现

在应用程序中实现 HTTP 协议连接请求服务器, 接收到 ts 流文件通过 JNI 接口传给引擎, 音视频引擎处理后再将结果返回给应用程序, 应用层界面采用 Android 自带的 MediaPlayer 类中的方法实现显示图像和播放声音[9].

3 实验测试

在本文的测试阶段, 实验环境为一台 HTTP 媒体流服务器、一部 Android 系统的华为手机, 手机中安装新媒体互动广播应用软件. 手机硬件环境为: 操作系统 Android4.2, 四核, RAM 容量 1G, ROM 容量 4G. 在 3G 和 WiFi 网络下均可以流畅的播放图像和声音. 视频播

放截图如图 9 所示.

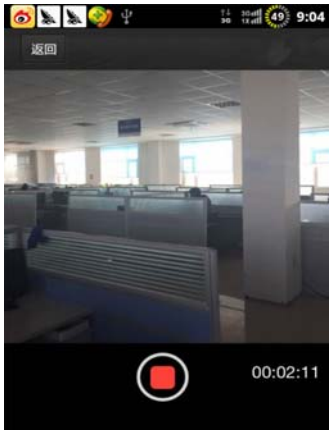

图 9 视频播放截图

资源利用率测试, 将移植了轻量级音视频引擎与移植 ffmpeg 编解码库的应用程序在播放视频运行时的资源利用率对比结果如下:

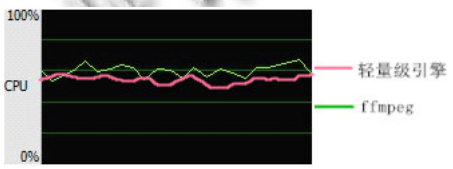

图 10 移植后应用的 CPU 占用

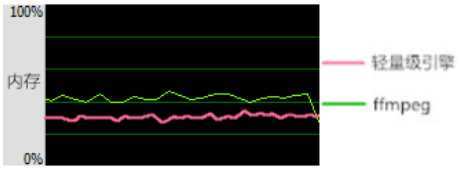

图 11 移植后应用的内存占用

由实验可看出, 两种引擎在实际运行时, CPU 占用基本都维持在 40%~60%之间, 相差不是很多, 轻量级音视频引擎比 ffmpeg 运行时 CPU 占用量稍少一些.

内存的使用量上, 轻量级引擎基本占 20%~40%之间, ffmpeg 在 40%~50%之间, 轻量级引擎在内存空间的占用量至少节省了 10%, 这对应用程序的开发很有意义.

4 结语

本文通过对一些开源代码的分析研究[10], 设计并实现了一个轻量级的音视频引擎. 首先分析了音视频引擎在移动终端应用程序中的作用, 接着详细描述了引擎各个模块和接口的设计和实现过程, 对各个模块进行了详尽的分析, 逐步完成音视频引擎的各个功能, 最后引入 H.265 解码器, 编译并移植到 Android 客户端并实现运行界面, 在引擎的运行过程中, 发现其应用的 CPU 占用率和内存使用量较开源的音视频编解码库有了明显改进, 可以稳定的运行在移动应用中.

参考文献

1 刘杰.基于智能手机平台无线视频传输的研究与设计.大连:大连理工大学,2013.
2 张璇.基于智能手机的流媒体播放及编解码研究.南京:南京邮电大学,2011.
3 何金.新一代智能终端多媒体播放平台的技术研究.宁波:宁波大学,2012.
4 盛小俊.一种低能耗移动流媒体播放器的设计与实现.武汉:华中科技大学,2009.
5 霍龙社,甘震.业务与运营,2010,4:6–13.
6 徐国强.基于 Android 的移动视频控制系统设计.太原:太原理工大学,2014.
7 王涔.基于 Android 的反向移动视频监控系统.广州:华南理工大学,2013.
8 Nightingale J, Wang Q. The Impact of Network Impairment on Qualityof Experience in H.265/HEVC Video Streaming. IEEE and Sergio Goma. 2014.
9 Westerink P, Schaffa F. A high level flexible framework for building multi-platform streaming applications. IBM T.J. Watson Research Center. 2010.
10 HTTP://ffmpeg.org/.