## 基于FFmpeg的Web音视频处理系统的设计与实现

## 绪论

**1.1 研究背景与意义**

**1.2 研究现状**

Web音视频处理的研究现状

相关框架技术的研究现状

**1.3 论文的研究目标和内容**

本论文的主要工作为：

1. 研究音视频处理技术以及相关理论，确定Web浏览器处理音视频的技术方案。分析Web音视频处理系统的需求，我们设计了Web音视频处理系统的主要框架和满足需求的技术实现方案与逻辑，而且对有关的技术进行了深入的研究和学习。
2. 研究FFmpeg音视频处理库，掌握FFmpeg处理音视频数据的相关方法和原理。设计实现基于FFmpeg库的Web端音视频的解封装、软解码、软编码、封装的流程。核心是通过在Web浏览器中调用FFmpeg编译后的WebAssembly二进制字节码模块的架构来处理音视频，实现高兼容性音视频格式解析以及高性能处理。
3. 针对弱实时性音视频处理中音视频拼接、音视频转换、音视频裁剪进行实现。
4. 针对强实时性音视频处理，播放处理时环境之间的数据的传输的实时性要求更高，还有浏览器渲染yuv图像数据的策略设计
5. 对设计实现的Web音视频处理系统进行测试。

**1.4 论文组织结构**

## 二、相关技术

**2.1音视频技术**

**2.2 FFmpeg**

**2.3 WebAssembly**

**2.4 本章小结**

## 系统需求分析与总体设计

**3.1 基于FFmpeg的Web音视频处理系统需求分析**

基于FFmpeg的Web音视频处理系统最终目标是实现本地音视频多媒体数据可以在Web浏览器中进行剪辑和播放两种处理操作，因此处理系统应该具备在Web浏览器运行时环境中调用FFmpeg编解码音视频的功能，利用WebAssembly二进制字节码技术和音视频编解码技术实现对本地原始的音视频数据进行解封装、解码、编码、封装的基础功能。同时在Web编解码音视频功能基础上，根据音视频处理在JavaScript运行侧和C++运行侧的数据转移实时性划分为弱实时性的剪辑处理功能需求和强实时性的播放处理功能需求。

此外，针对现有的Web音视频处理系统中所存在的不足，设计出解决的思路和方法。针对JavaScript直接编解码音视频的性能非常差，本系统引入WebAssembly二进制字节码中间层来提高性能。针对弱实时性剪辑处理的灵活性，本系统设计DAG任务构建方案实现多剪辑任务的拓扑执行。针对强实时性播放处理的音频帧和视频帧同步问题，本系统设计实现多级缓存的帧同步策略。

最后是系统的非功能性需求主要包含安全性、易用性、性能和可维护性需求。

**3.1.1 系统功能性需求**

根据系统的设计目标，本小结分为三个部分描述系统的功能性需求：Web浏览器中本地音视频的编解码功能、弱实时性音视频剪辑处理功能以及强实时性音视频播放处理功能。

**1、Web音视频编解码模块**

Web音视频编解码功能是系统的最核心的部分，后续的弱实时性处理模块和强实时性处理模块都是基于此功能做的进一步扩展。

针对Web浏览器环境的特殊性，系统要实现对本地音视频文件的读取和下载功能。同时对读取的原始音视频文件，需要实现对音视频数据的解封装、解码、编码、封装的四个主要功能。在读取文件的过程，需要利用web worker线程来读取音视频，以防止音视频文件过大阻塞主线程的交互执行，同时我们读取文件是通过Stream方式读取，不会过度的消耗内存占用。

整个编解码处理同样是放在子线程的WASM模块中去处理，我们需要设计实现主线程和子线程的交互Message传递。

封装与解封装

avformat\_open\_input()

解码与编码

根据Web音视频编解码功能的需求，得到其需求用例图，如图3.1所示：

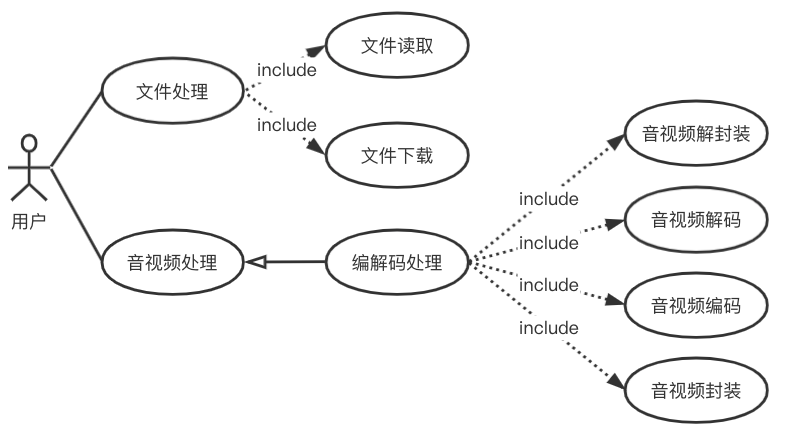


图3.1 Web音视频编解码功能需求用例图

用例名称:web音视频编解码处理。

说明:编解码处理模块接收到流式的文件数据，可以有效的兼容解析多种音视频文件，进一步进行处理包括音视频解封装、音视频解码、音视频编码、音视频封装四个重要功能。

参与者:音视频编解码模块。

前置条件:本地音视频文件读取成功。

后置条件:本地音视频文件可以被正确解析。

**2、弱实时性剪辑处理模块**

在基于上一小节实现的Web浏览器环境中对音视频文件的读取、解封装、编解码功能的基础上，实现弱实时性的音视频剪辑处理功能模块。

弱实时性处理模块基于Web编解码处理功能的基础上，利用DAG有向无环图的方式构建多剪辑任务的功能。同时将剪辑功能分为五个基本的原子处理单元，分别是剪切单元、压缩单元、抽取单元、合并单元和转换单元。

弱实时性音视频处理这类功能操作对于JavaScript运行侧和WASM模块运行侧的数据交互实时性不强，主要是单向的数据流动，由JavaScript运行侧读取音视频数据，将数据发送给WASM模块进行解析处理，最后再将处理后的结果数据返回给JavaScript侧进行下载。其中在WASM模块解析处理音视频数据的同时，并不会有实时性的解码后的音视频数据在两个运行环境传递。

功能流程上主要是通过JavaScript调用封装好的功能函数组合来实现，最后通过DAG调度器来处理完剪辑任务后，再由WASM运行侧将数据通过web worker线程所有权交由浏览器主线程进行下载。

根据弱实时性音视频剪辑处理功能模块的需求，得到其需求用例图，如图3.2所示：

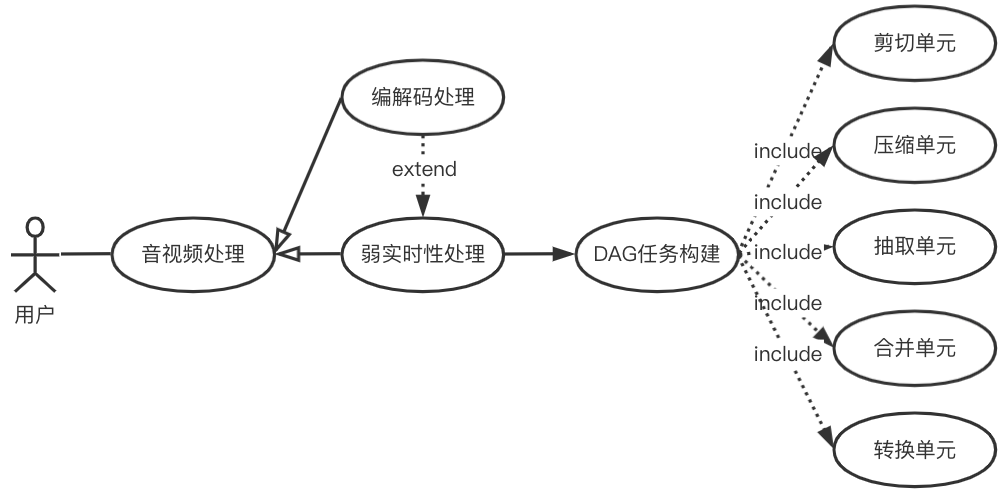


图3.2 弱实时性处理模块需求用例图

用例名称:弱实时性剪辑处理。

说明:基于Web音视频编解码模块的，通过基本的处理单元来构建DAG剪辑任务图，实现多任务处理功能。弱实时性处理模块包含剪切单元、压缩单元、抽取单元、合并单元以及转换单元。

参与者:web音视频编解码模块、弱实时性处理模块。

前置条件:本地音视频文件成功读取、解析并成功解码。

后置条件:剪辑处理后的音视频文件可以下载查看，并符合预期处理效果。

1. **强实时性播放处理模块**

强实时性播放处理模块同样是基于Web音视频编解码功能进行扩展。强实时性播放处理主要体现在FFmpeg侧实时的将每一帧音视频数据解码后，都需要通过我们定义的web worker消息机制将帧数据所有权转移给JavaScript侧运行时，然后在浏览器中可以通过WebGL和ContextAudio来分别对音频和视频进行消费。

强实时性处理模块主要分为顺序播放、倍速播放、全屏播放以及Seek播放。音视频的播放功能主要通过WebGL将视频帧的yuv图像数据渲染出来，以及通过Audio来播放音频采样数据。

根据强实时性音视频播放处理功能模块的需求，得到其需求用例图，如图3-3所示：

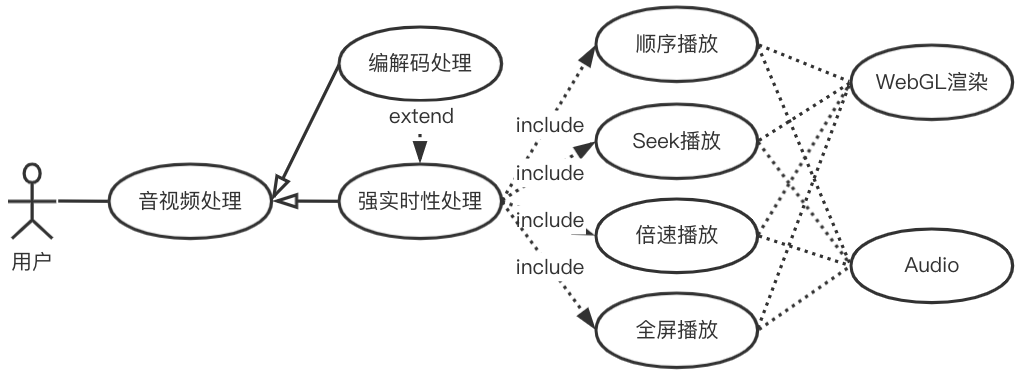


图3-3 强实时性处理模块需求用例图

用例名称:强实时性播放处理。

说明:基于Web音视频编解码模块的，通过将web编解码模块处理后的解码音视频数据从WASM运行侧转移给JavaScript运行侧，然后通过WebGL来进行yuv图像的绘制、Audio播放音频数据。强实时性播放处理模块包含顺序播放、倍速播放、全屏播放以及Seek播放。

参与者:web音视频编解码模块、强实时性处理模块。

前置条件:本地音视频文件成功读取、解析并成功解码。

后置条件:解码后的音视频数据可以成功传递，并成功同步绘制在页面进行播放。

**3.1.2 系统非功能性需求**

Web音视频处理系统的非功能需求主要体现除了功能性需求以外的其他特性，包括系统的安全性、易用性、性能、可维护性等需求，具体说明如下：

1. 安全性

系统的安全性主要体现在两个方面，分别是编码安全和用户内容安全两部分。编码安全指的是系统自身的代码需要经过严格的测试，尽量对编码过程中容易出现的内存泄漏、数据为空进行校验与捕获。用户内容安全是指系统在处理用户本地音视频数据时，保证可见性原则和不侵入原则，安全有效的处理用户指定的音视频文件。通过以上两个方面保障系统自身的安全性。

1. 易用性

Web音视频处理系统借助了B/S架构的易用优势，无需下载其他软件实现在线处理。同时系统在针对不同用户群体使用时，要尤其针对不具备音视频相关专业知识基础的用户在使用过程中，提供对应的默认参数以及相关参数的友好提示来帮助用户更好的使用平台。要做到系统页面逻辑清晰、功能使用简单方便。

1. 性能

性能需求主要指在Web浏览器中提高音视频处理性能。因为Web浏览器的音视频处理过程是非常消耗系统性能，因为Web浏览器并不适合处理CPU密集型计算任务，并且Web浏览器作为应用软件对操作系统做了封装，导致不能直接利用操作系统的资源。如果可以通过提高音视频处理在Web中的性能，可以缓解服务端压力，更好的利用客户端的计算力。

1. 可维护性

系统可维护性需求主要指系统在后续的更新维护中，能够根据业务需求的变更及时进行扩展。同时对于系统中关键日志能够及时记录，方便在出现错误的时候能够及时根据日志记录进行解决。如要进行新需求的开发时，能够使用模块化的思想，不影响其他模块功能的前提下，新增新的功能模块，实现业务的快速部署。

**3.2 系统总体设计**

本小节根据上一节的各个模块的功能性需求和非功能性需求为基础，以需求分析为设计目标，对本文所提出来的性能高、兼容性强的Web音视频处理系统进行总体方案设计，具体包含Web系统架构设计和系统功能设计。

**3.2.1 系统架构设计**

本节基于FFmpeg和WebAssembly的Web音视频处理系统整体框架进行设计，音视频处理系统总结构如图3-4所示，音视频处理系统的结构主要分为四个部分：音视频渲染交互层、音视频数据控制层、Web Worker线程层以及音视频处理层。其中音视频处理层是系统核心，是Web音视频处理的性能保证。Web Worker线程层主要通过多线程的方式加载本地音视频文件和WASM处理模块文件以及消息数据的交互，可以让音视频处理在子线程中而不影响主线程的运行，解耦了模块和模块之间的关系。提高了系统的可维护性和使用性。音视频控制层主要承接渲染交互层和下层的关系，确保处理的可靠性。渲染交互层主要是针对强实时性处理和弱实时性处理的不同抽象呈现。

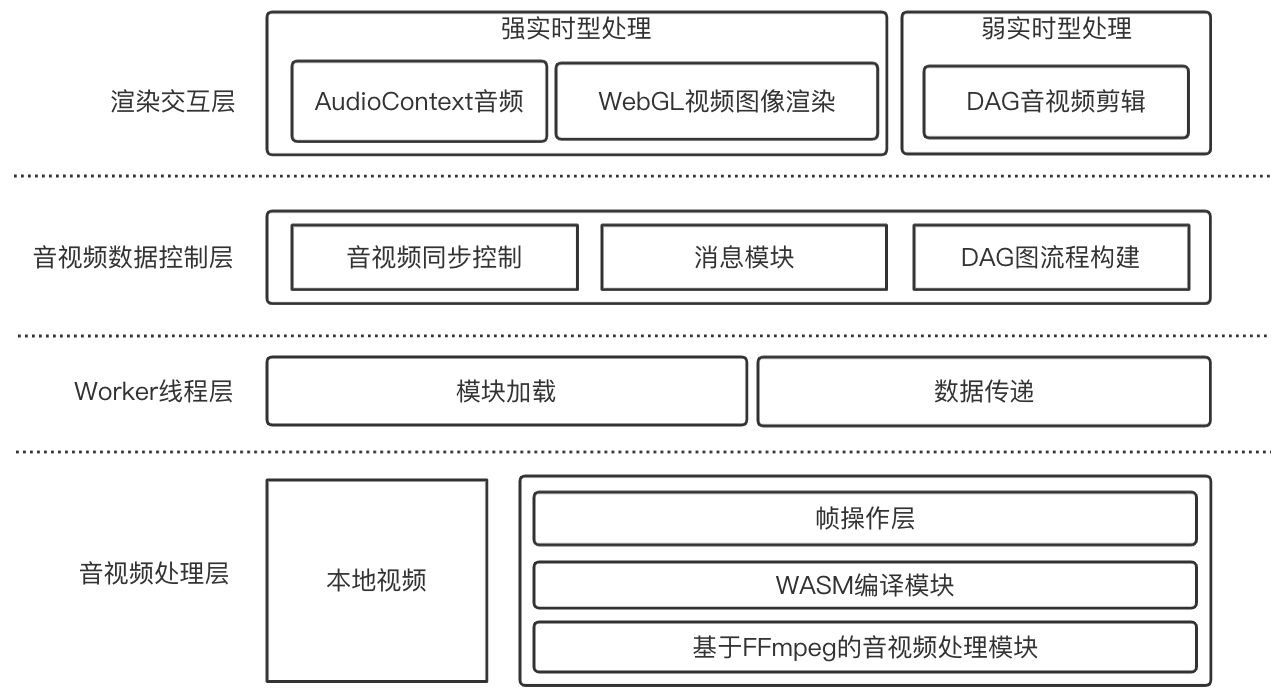
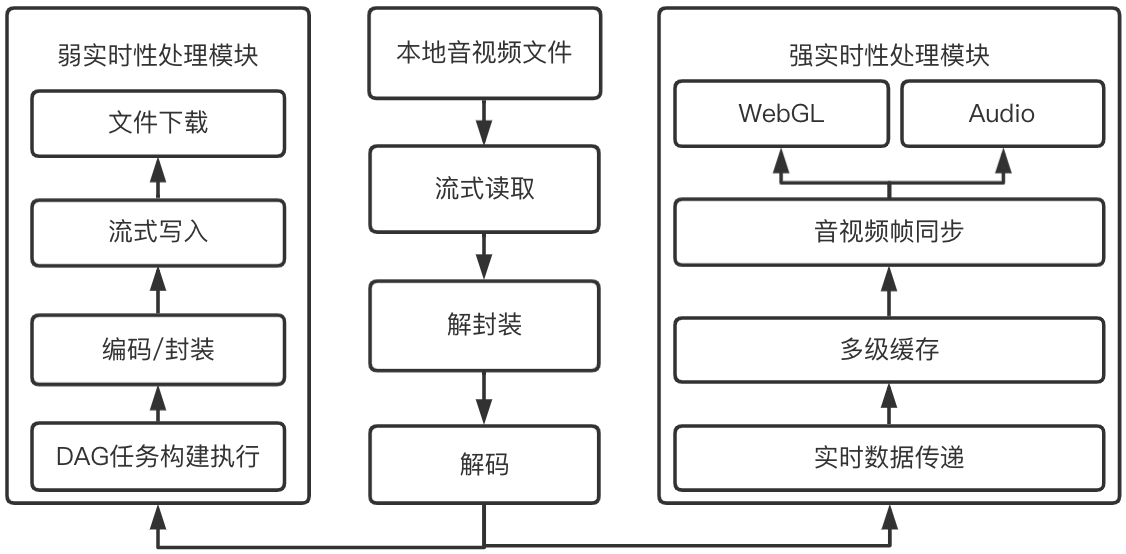


图3-4 音视频处理系统总体结构图

在Web音视频处理系统中，通过worker线程层加载原始音视频文件和WASM音视频处理模块，由音视频数据控制层来同步两个worker线程的数据传递和控制指令的逻辑执行。然后将处理后的音视频数据分为强实时性和弱实时性两种方式由主线程控制。如图3-5展示了Web音视频处理系统的基本工作流程。



如图所示，弱实时性处理和强实时性处理模块的前端都依赖Web音视频编解码模块。Web音视频编解码模块确保正确的解析解码不同格式的音视频文件。而弱实时性处理模块通过DAG构建剪辑任务来进一步处理解码后的数据，最终确保数据可以正确被下载。强实时性处理模块则是不断地实时的获取解码后的音频帧和视频帧，然后进一步将其渲染到浏览器上进行播放。

**3.2.2 系统功能设计**

Web音视频处理系统的整体功能结构如图3-5所示，本节将对于强实时性和弱实时性音视频处理这两部分功能进行设计说明。

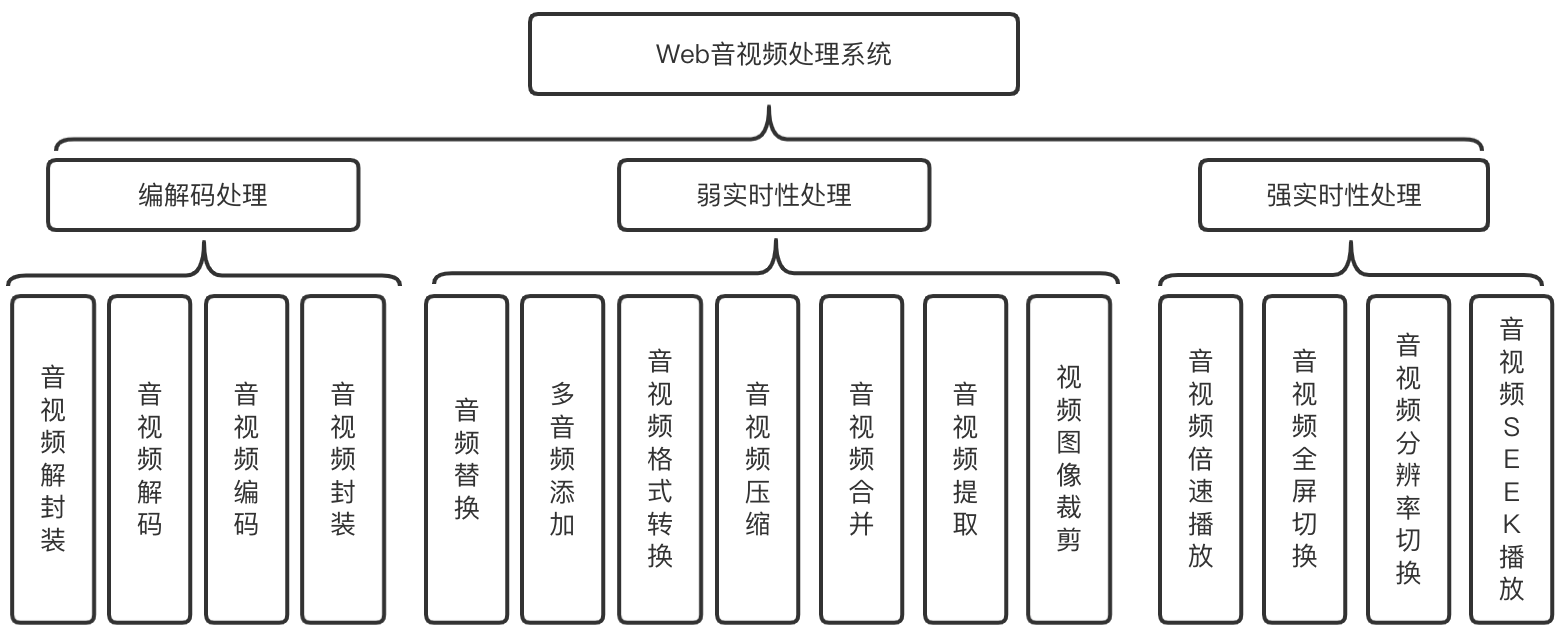


图3-5 音视频处理系统整体功能图

本系统中主要对音视频的处理分为弱实时性处理和强实时性处理。弱实时性处理主要是对输入的音视频进行音频或者视频本身的剪辑，例如音视频合并、裁剪、压缩等。而强实时性处理主要对音视频实时播放的处理，又分为顺序播放、倍速播放、全屏切换、Seek播放等。

**3.4 本章小结**

本章首先介绍了音视频处理系统在Web浏览器架构的位置，确定其主要功能是基于FFmpeg的音视频处理流程架构需求。然后在此基础上，针对实时性的高低进一步的对剪辑需求和播放需求进行分析。最后对系统的非功能性需求进行分析，主要是性能、安全性、兼容性与可维护性需求。

## 系统详细设计与实现

## 4.1 Web编解码处理模块的设计与实现

4.1.1 FFmpeg编译移植

FFmpeg是一个开源免费的跨平台音视频分离、转换、解码于一体的基础库，FFmpeg是Linux系统下开源音视频处理库，在Web浏览器环境不能直接调用使用FFmpeg相关功能。因此本文构建的系统通过引入WebAssembly这项技术，将FFmpeg编译为WASM模块再通过Web Worker加载来提高Web浏览器对CPU密集型计算的执行性能和兼容性。

FFmpeg自身包含了很多衍生的工具包和一些特定功能的基础包，在本文系统中并不是必须的，所以本文采用定制化编译的思路来裁剪最终的编译产物，也能在一定程度上减小编译产物以及提高加载速度，并增添扩展功能。

FFmpeg的编译依赖于Pkg-config，而它又依赖于GLib,而GLib又依赖于gettext,所以必须先安装并编译相关依赖。具体的编译步骤如下。

1. 下载FFmpeg源码以及fdk-aac、x264、x265等音视频编码函数库
2. 下载emconfigure、emmake构建工具链
3. configure
4. Make

4.1.2 Web编解码架构设计和流程设计

Web编解码架构依赖于Web Worker多线程，通过多线程可以很好的避免主线程和用户的直接交互体验下降。通过两个Worker子线程分别加载本地音视频文件和处理音视频的WASM模块。通过主线程为桥梁，不断地将流式的音视频文件数据通过从文件子线程传递到处理子线程中进行编解码处理，之后根据实时性划分来区别编解码处理子线程中处理后的数据是否实时的要转移到主线程进行进一步的渲染和消费。具体的Web编解码架构设计如图所示：

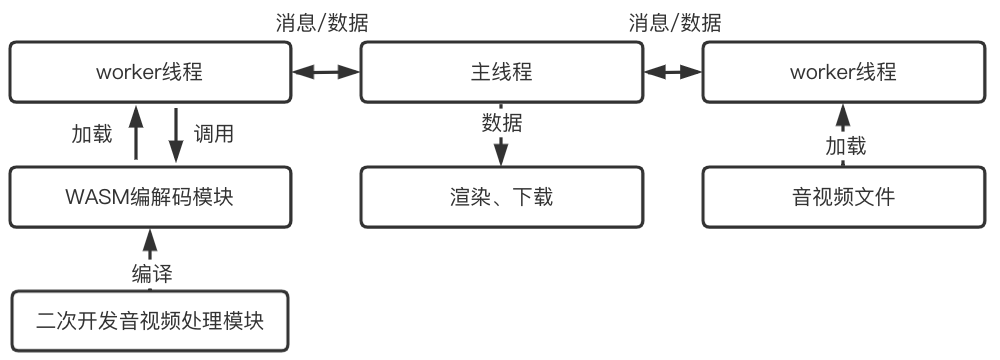


图 Web编解码具体设计

基于WASM音视频处理模块的Web音视频编解码流程如图所示。首先通过worker线程加载本地音视频文件，如果加载成功则读取音视频文件的封装信息，一般音视频文件的封装信息包含文件内的音频流和视频流编解码信息和相关格式信息，这部分信息可以帮助我们加载对应格式的音频解码和视频解码函数。解封装成功后，可以拿到音频流和视频流，当然有些视频没有音频那意味着音频流为空。通过拿到音频流河视频流可以进一步对其进行解码操作获取原始数据也可以对原始数据进行重新编码，最后将编码音视频流封装起来。整个Web音视频编解码流程的设计方案就是这样。

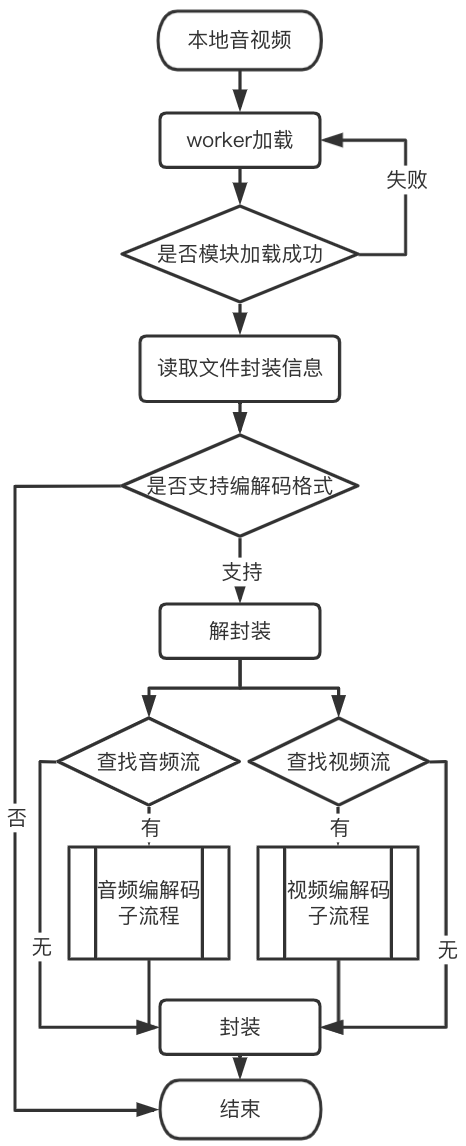


图 Web编解码流程

4.1.3 Web编解码具体实现

## 4.2弱实时性处理模块的设计与实现

本章主要介绍弱实时性音视频处理模块的详细设计与实现。

**4.2.1 抽象剪辑处理单元**

4.1.1 抽象处理单元

在弱实时性处理中，将音视频处理功能抽象设计为多个剪辑元件，共包含Compress处理单元、Transform处理单元、Input处理单元、Output处理单元，最后通过构建DAG图来组合多个元件来实现多任务处理的构建。

首先在系统内部注册相关元件，完成元件注册之后，就可以针对音视频实例或者中间实例进行创建和编辑相关参数。

1. Input处理单元

Input处理单元主要是有选择的初始音视频决定，每一个初始化的音视频都可以作为一个Input处理单元。其中包含每一个音视频的详细信息，包括音视频大小、格式、类型等等。

1. Compress处理单元

Compress处理单元主要用来对音视频进行压缩。

1. Extract处理单元
2. Merge处理单元

Merge处理单元用于两个及两个以上音频、视频的合并。

1. Cut处理单元

Cut处理单元对音频或视频进行裁剪操作，以时间为维度进行分割操作。

1. Transform处理单元

Transform处理单元主要用于对音频或视频格式、编解码协议等进行转换。

1. Output处理单元

Output处理单元主要作用是指定我们最终输出的音视频数量，因为每个阶段的操作都可以作为音视频输出。

**4.2.2 基于DAG的多剪辑任务处理实现**

基于DAG构建音视频的处理任务，可以帮助我们流程化的构建有向无环图。

4.2.1 构建DAG任务图

基于React-flow-render库构建Web拖拽层，用户可以通过预设的处理单元进行拖拽添加。

4.2.2 Task调度器

## 4.3强实时性处理模块的设计与实现

本章主要介绍关于在Web音视频处理系统中，强实时性播放处理模块的设计与实现。

**4.3.1 音视频速率同步控制模块的设计与实现**

多级缓存

交替数组

**4.3.2 简单播放的设计与实现**

**4.3.1 高级播放的设计与实现**

倍速播放

Seek播放

## 系统测试与分析

## 总结与展望

## 致谢

## 参考文献