## 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. 研究学习音视频处理技术以及相关理论，分析Web音视频处理系统的需求，设计Web音视频处理系统的架构和满足需求的技术实现方案。
2. 研究学习FFmpeg音视频处理库以及WebAssembly二进制字节码编译技术。掌握FFmpeg处理音视频的核心流程和原理，并利用WebAssembly技术编译FFmpeg以实现Web浏览器中强兼容、高性能的音视频处理能力。
3. 设计实现基于FFmpeg的Web多线程音视频编解码模块，进一步设计实现根据数据是否实时性消费划分的弱实时性音视频剪辑处理模块和强实时性音视频播放处理模块。针对弱实时性处理模块实现基于DAG拓扑图的多剪辑任务处理。针对强实时性处理模块实现基于时间戳同步、多级缓存、交替存储数组结构的音视频同步播放处理。
4. 针对本文设计实现的Web音视频处理系统进行功能性测试和非功能性测试。功能性测试主要包括测试音视频编解码功能、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。非功能性测试主要包括系统的性能、代码质量等。

**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

本章内容通过分析系统的功能性需求和非功能性需求进而对系统整体架构和功能进行设计。借助用例图和模块数据流图对系统功能性需求进行分析，同时也对非功能性需求进行说明。最后通过系统模块架构图、整体数据流图、功能分解图清晰地描述系统中各个模块之间的关系。

**3.1 基于FFmpeg的Web音视频处理系统需求分析**

基于FFmpeg的 Web音视频处理系统的最终目标是实现在Web浏览器上下文环境中对本地音视频文件的高性能、强兼容的处理功能，包括基本的音视频编解码处理，以及在此基础上根据解码后数据实时性消费划分的弱实时性剪辑处理和强实时性播放处理。所以本文提出的Web音视频处理系统需要支持在Web浏览器上下文环境中嵌入并调用扩展的FFmpeg音视频处理模块，并通过结合多线程、WebAssembly编译技术的架构来增强原生浏览器对多种音视频格式的兼容识别和解析处理。

因此Web音视频处理系统的功能性需求包括三部分：Web音视频编解码处理功能性需求、弱实时性音视频剪辑处理功能性需求以及强实时性音视频播放处理功能性需求。非功能性需求主要包括安全性、易用性、性能、可靠性以及可维护性五部分。

**3.1.1 系统功能性需求**

1. **Web音视频编解码需求**

Web音视频编解码功能是本系统中最基础、也是架构设计最核心的部分，因为弱实时性剪辑处理模块和强实时性播放处理模块都是基于此功能进行扩展实现的。而Web音视频编解码处理模块的核心思想是在Web浏览器中实现以FFmpeg自身优秀的音视频处理流程和规范为基础的Web编解码功能。但因为Web浏览器执行引擎不能直接运行由C语言组成的FFmpeg音视频处理程序，所以通过引入WebAssembly二进制字节码编译技术来定制化裁剪编译扩展的FFmpeg音视频处理程序为WASM音视频处理模块，再进一步利用Web Worker子线程加载编译模块调用，从而实现在Web浏览器中对本地音视频文件的解封装、解码、编码、封装的基本处理功能。

WebAssembly二进制字节码程序相比较C编译型语言程序可以被Web浏览器执行引擎所执行，同时相比较高级抽象的JavaScript解释型语言程序的执行速度要更快，所以利用WebAssembly编译技术不但解决了FFmpeg扩展音视频处理程序的嵌入执行难题，而且还可以提高音视频处理在Web浏览器中的执行时间。

Web音视频编解码模块通过两个Web Worker子线程来分别读取本地音视频文件和加载初始化编译后的WASM音视频处理模块。在文件Worker子线程中可以利用FileReader对象异步读取存储在用户计算机上的多媒体文件，并以Web浏览器JavaScript主线程作为中间缓冲区将读取到的文件数据转发给处理Worker子线程。处理Worker子线程在加载WASM音视频处理模块时通过初始化已经保存了WASM模块的实例引用，通过此引用可以调用WASM模块暴露的音视频处理接口来进一步实现对音视频数据的解封装、解码、编码、封装四个重要功能。因此Web编解码模块的具体数据流图如图3-1所示：

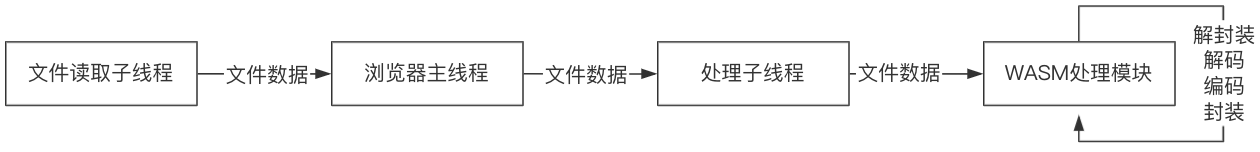


图3-1 Web编解码模块功能的数据流图

因为模块是通过多个Worker子线程来处理异步耗时的文件读取、音视频处理任务，所以还需要设计实现主线程与多个子线程之间的指令数据通讯机制协调实现UI交互操作得到正确响应。此外在Web音视频编解码功能基础之上，根据音视频处理在Web浏览器JavaScript主线程运行侧和WASM音视频处理模块运行侧的数据转移实时性可以划分为弱实时性音视频剪辑处理功能和强实时性音视频播放处理功能。

根据Web音视频编解码处理功能的需求，得到其需求用例图，如图3-2所示：

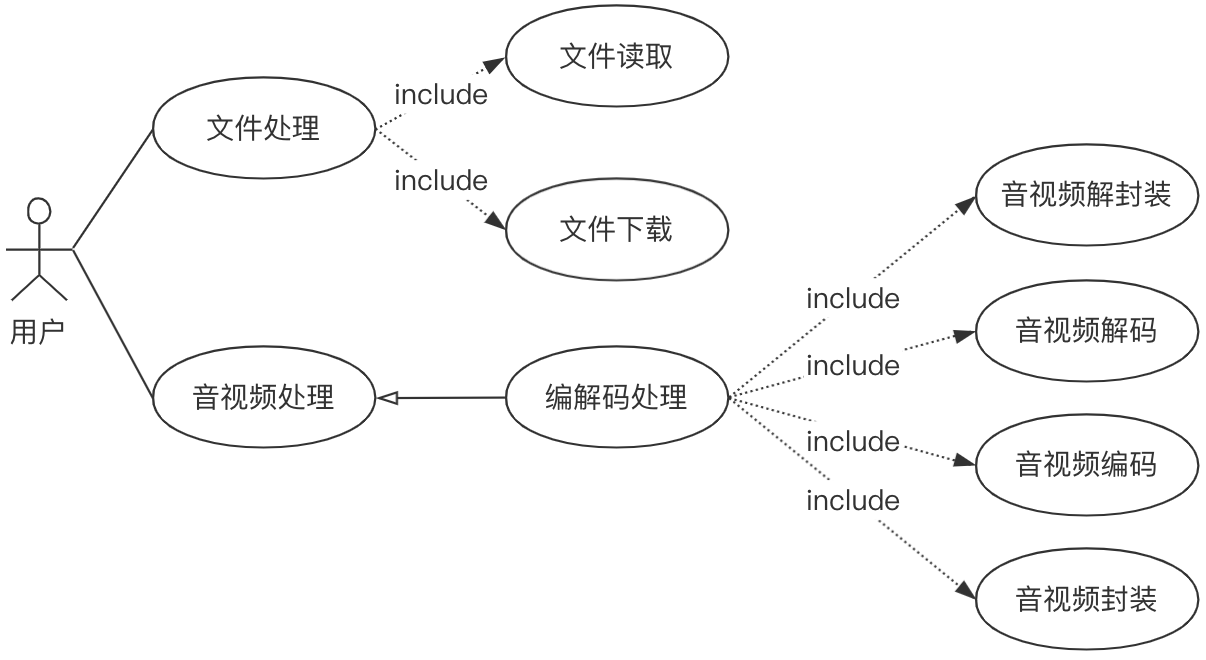


图3-2 Web音视频编解码处理功能需求用例图

用例名称:Web音视频编解码处理。

说明:在Web浏览器上下文环境中，实现文件读取子线程将音视频文件数据流式地传递到音视频处理子线程中FFmpeg定制化编译后的WASM处理模块中，实现高效兼容地解析多种音视频格式，以及包括音视频解封装、音视频解码、音视频编码、音视频封装四个重要基础处理功能。

参与者:Web音视频编解码模块，文件系统模块

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识别解析、解封装、解码、编码、封装。

**2、弱实时性剪辑处理的需求**

基于上一节Web浏览器上下文环境中对本地音视频文件的读取、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。其中弱实时性主要体现在Web浏览器JavaScript运行侧与WASM音视频处理模块运行侧的数据实时性交互不强。整体上音视频数据流向主要是单向流动，由文件子线程读取音视频文件数据通过主线程发送到处理子线程中，再由处理子线程中JavaScript运行侧利用WASM音视频处理模块的实例化引用分配内存空间进行处理。直到WASM音视频处理模块处理完用户定制的音视频剪辑任务后，才会把剪辑后的音视频数据重新编码、封装后再一步步从WASM音视频处理模块的内存空间转移到处理子线程中的JavaScript运行侧，再传递到主线程中进行下载查看。即就是WASM音视频处理模块在处理音视频数据的同时，解码后的音视频帧数据并不会实时地反向传递到Web浏览器主线程中，简化了相应的音视频处理复杂度。因此弱实时性剪辑处理功能的数据流图如图3-3所示：

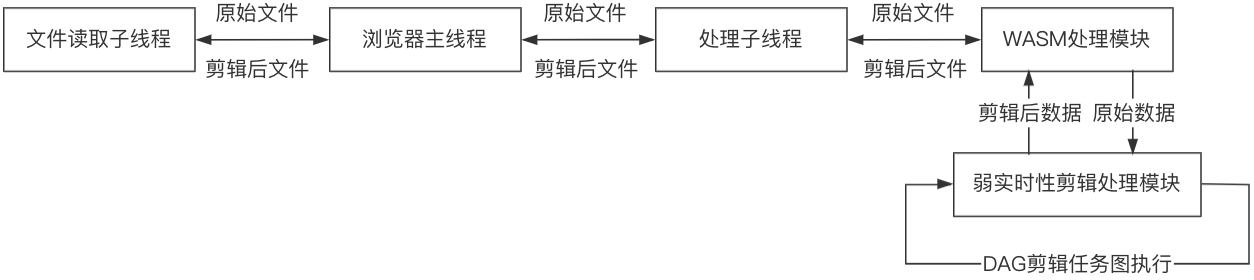


图3-3 弱实时性剪辑处理功能数据流图

音视频剪辑处理通过将剪辑功能操作抽象为五个基本的原子处理单元，分别是剪切处理单元、抽取处理单元、合并处理单元、压缩处理单元和转换处理单元。以及两个基础的辅助处理单元分别是输入处理单元和输出处理单元。输入处理单元是根据文件子线程读取本地文件数量一一对应生成的确保输入的完全可控性；输出处理单元用于标识一个剪辑处理任务的结束标志。之后在画布上成功注册上述的七种处理单元组件后，通过拖拽的组件的方式在画布上灵活组合多个处理单元构建DAG剪辑拓扑图达到最终的剪辑处理需求。最后根据构建好的剪辑拓扑图，执行处理模块再经过以下三个步骤来执行多个剪辑任务：

1、以输入处理单元为无入度根节点，检测多剪辑任务图是否为多个正确的有向无环图。

2、初始化系统任务执行器，并通过拓扑排序算法并行执行多个剪辑任务的各个处理单元。

3、以输出处理单元为无出度结束节点，标志当前音视频剪辑任务的结束，并提供下载查看剪辑后音视频的功能。

根据弱实时性音视频剪辑处理功能的需求，得到其需求用例图，如图3-4所示：

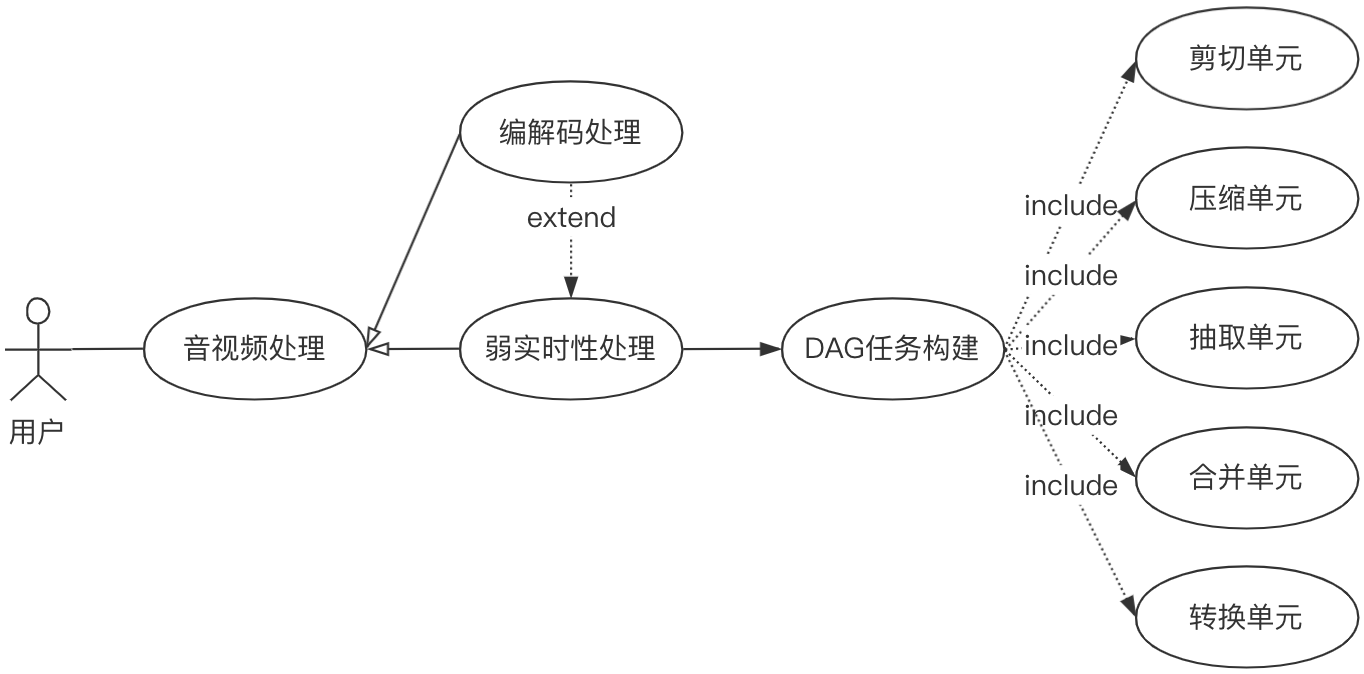


图3-4 弱实时性剪辑处理模块需求用例图

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，通过组合抽象的剪辑处理单元和基本输入输出处理单元来构建多DAG剪辑任务图，实现多音视频剪辑处理功能。剪辑处理单元抽象为剪切处理单元、压缩处理单元、抽取处理单元、合并处理单元以及转换处理单元。

参与者:Web音视频编解码模块、弱实时性剪辑处理模块。

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，并符合预期剪辑效果。

1. **强实时性播放处理的需求**

相比较弱实时性剪辑处理，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，不同之处在于播放处理模块要求WASM音视频处理模块运行侧将解码后的每一帧音视频数据实时地通过线程之间的数据通讯机制传递到Web浏览器主线程运行侧，在浏览器主线程中将解码后的视频YUV图像数据和音频采样数据分别通过WebGL和AudioContext进行渲染和播放。同时播放处理模块设计以时间戳同步机制为主的多级缓存策略和回退补偿策略来协调解决音频帧和视频帧的实时同步问题。最终强实时性播放处理模块的数据流图如图3-5所示：

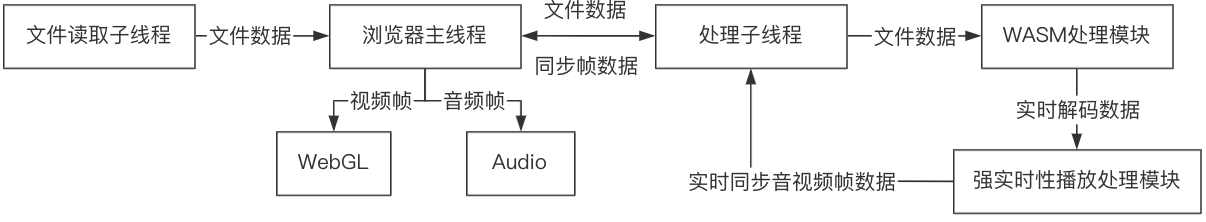


图3-5 强实时性播放处理功能数据流图

强实时性播放处理模块的功能主要分为简单顺序播放、高级播放，高级播放包括倍速播放、全屏播放以及Seek播放。在播放处理过程中，需要考虑协调音频帧和视频帧的实时同步问题，确保在任何播放状态下音频帧和视频帧都会同步被消费，也就是通常所说的音画同步。在播放处理模块中设计音频帧和视频帧的交替存储数据结构，以及设计实现多级缓存机制来确保音频帧和视频帧的同步。同时针对出现音画不同步的问题，强实时性播放处理模块需要一定的回退补偿和丢弃补偿策略来修正音画不同步的情况。

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3-6所示：

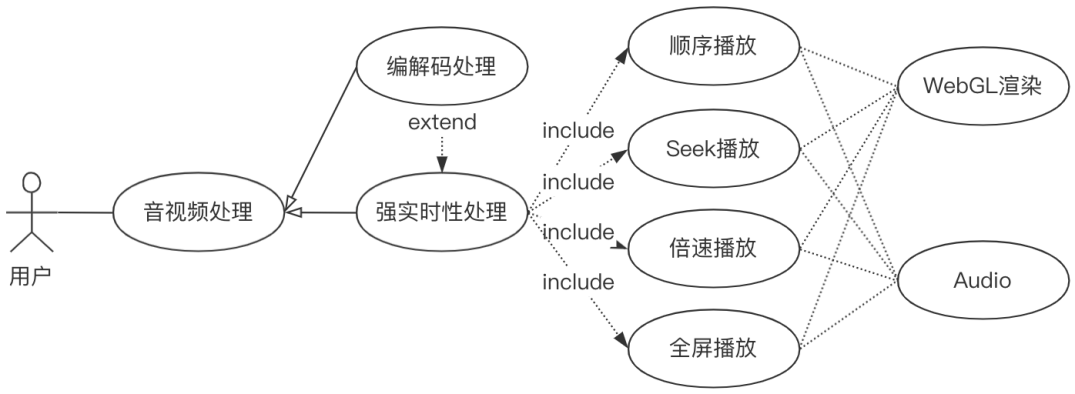


图3-6 强实时性处理模块需求用例图

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、强实时性播放处理模块。

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实时地从WASM处理模块子线程转移至Web浏览器JavaScript主线程中，并成功实时同步对视频帧YUV图像数据渲染和音频帧数据播放。

**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、可靠性以及可维护性需求，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音视频在线处理功能。系统面对不同用户群体，尤其是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功能。同时要做到系统页面逻辑清晰、功能操作简单方便等。

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资源。

1. 可靠性

系统可靠性需求指的是使用该Web音视频处理系统处理音视频时，需要严格按照使用者定义的规则进行处理。同时系统自身的稳定性需要控制在极小的波动范围内，不会对使用者的浏览器应用产生过大的内存负担而导致页面崩溃等问题。

5. 可维护性

系统可维护性需求主要指系统在后期的更新维护过程中，能够根据新的业务场景对原有系统进行扩展。同时系统本身运行过程中，通过埋点收集关键的日志信息，方便根据日志信息排查系统出现的问题并解决。同时在系统功能的设计实现上，尽量运用六大软件设计原则，做到系统中各个模块高内聚、低耦合的关系，提高系统的可维护性。

**3.2 系统总体设计**

根据上一节系统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，本节主要针对本文提出的高性能、强兼容性的Web音视频处理系统进行总体设计，具体包括系统架构设计和系统功能设计。

**3.2.1 系统架构设计**

因为Web浏览器中JavaScript执行引擎天然的不适合执行音视频编解码处理这类CPU密集型计算任务，以及无法直接在Web浏览器中运行C语言程序的局限性，所以本系统架构设计上需要考虑怎么样结合WebAssembly二进制字节码编译技术，将基于FFmpeg扩展的音视频处理程序编译后的WASM音视频处理模块通过Web Worker多线程加载到Web浏览器中进行调用来解决低性能、局限性的问题。

基于FFmpeg和WebAssembly的Web音视频处理系统的整体架构主要分为四层：音视频渲染交互层、音视频数据控制层、Web Worker线程加载层以及音视频处理层。接下来分别介绍这四层架构之间的关系以及各层的主要作用：

1、音视频处理层是系统核心功能层，也是Web音视频处理的兼容性和性能保证所在。在音视频处理层需要交叉链接编译FFmpeg裁剪模块和扩展程序为WASM音视频处理模块，并设计WASM处理程序与JavaScipt调用程序之间的交互接口，同时直接用WASM模块实例去分配音视频数据的存储内存来防止不同运行引擎之间非必要的数据拷贝以提高处理性能。

2、Web Worker线程加载层主要通过子线程加载本地音视频文件和WASM音视频处理模块，让本地音视频文件的读取和CPU密集型的音视频计算处理两种耗时操作在子线程中执行而不会阻塞主线程的运行来提高系统的整体性能。最后设计实现主线程和子线程之间的消息通讯机制来实现各个线程之间的交互调用逻辑来协作完成音视频处理任务，通过这种方式可以解耦模块和模块之间的关系，提高了系统的可维护性和稳定性；

3、音视频数据控制层主要作用是通过连接渲染交互层和Worker线程加载层，将渲染交互层的操作逻辑指令解释为Worker线程加载层提供的消息指令组合，再调用下层的核心处理功能。同时也将Worker线程层传递的解码后的音视频数据进行同步控制后再由渲染交互层进行消费来确保音视频帧的实时同步。

4、渲染交互层是使用者直接操控和查看的交互页面的统称，主要作用是抽象音视频处理任务的定制化构建过程以及实时音视频帧数据的渲染播放。

因此Web音视频处理系统的整体架构设计如图3-7所示：

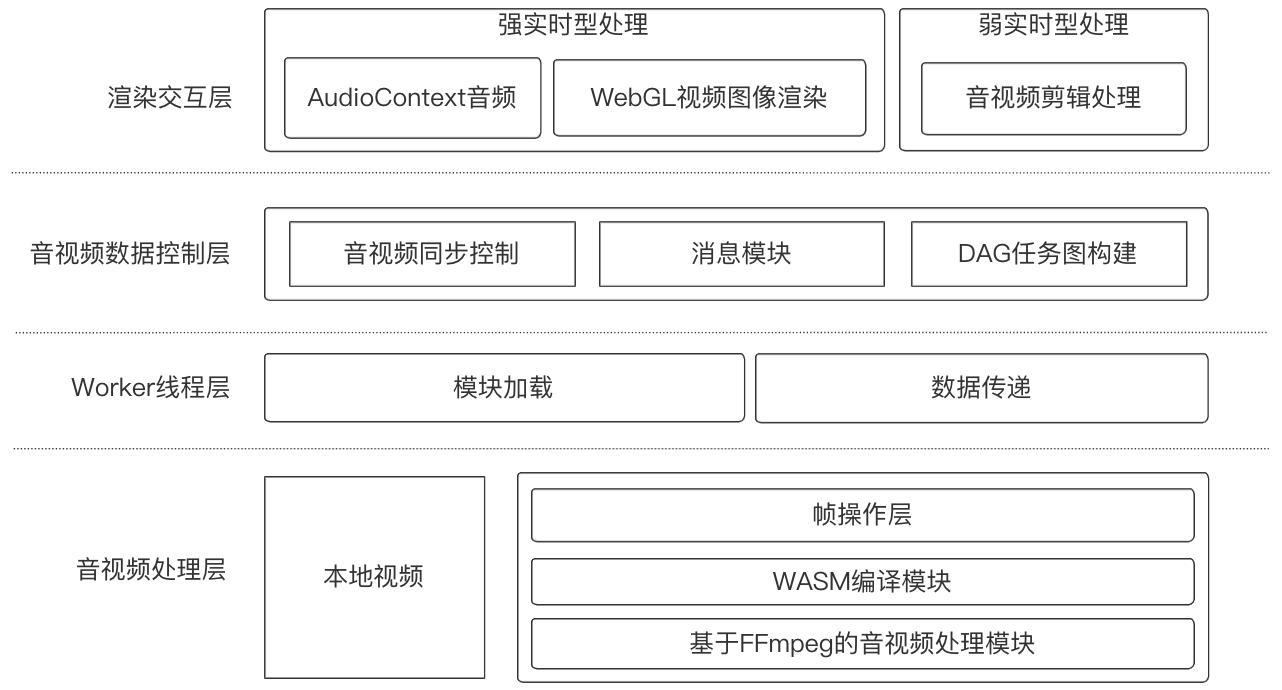


图3-7 音视频处理系统总体架构图

在Web音视频处理系统中，弱实时性音视频剪辑处理模块和强实时性音视频播放处理模块的前端部分都基于Web音视频编解码模块。根据不同的音视频处理需求，Web音视频编解码模块将解码后数据发送给具体的模块来进一步处理。如图3-8展示了Web音视频处理系统针对不同功能需求的具体工作流程。

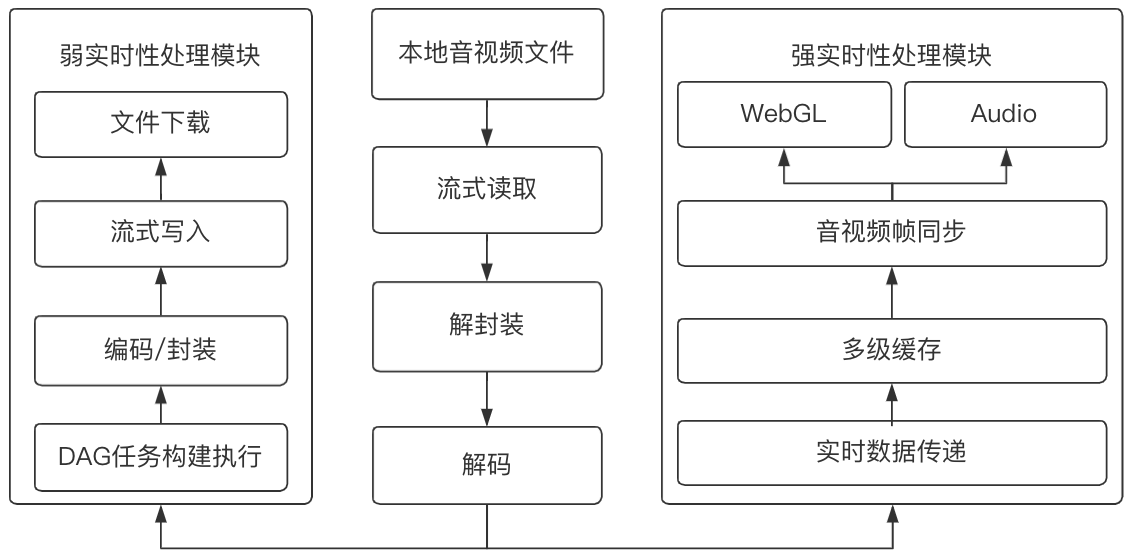


图3-8 系统整体数据流图

首先Web音视频编解码模块第一步是读取本地音视频文件数据，再根据音视频文件的头信息加载对应的解封装器、解码器来保证正确地解码不同格式的音视频文件，最后将解码后的数据发送给弱实时性处理模块或者强实时性处理模块来完成剪辑功能和播放功能。弱实时性处理模块根据DAG剪辑任务图自定义的各个节点处理单元来拓扑执行解码后的音视频数据，将处理完成的音视频数据通过参数指定的编码器和封装器来生成新的音视频文件，最后流式写入本地文件实现文件下载。强实时性处理模块则需要实时地获取到解码后的音频帧和视频帧数据，通过缓存模块和同步控制模块的处理，进一步把帧数据发送给Web浏览器主线程的WebGL模块和Audio模块来消费。

**3.2.2 系统功能设计**

根据Web音视频处理系统各相关功能之间的关系，将系统整体功能划分为文件模块和音视频处理模块，音视频处理模块又分为Web编解码模块、弱实时性剪辑处理模块和强实时性播放处理模块三个模块。最终系统整体功能分解如图3-6所示：

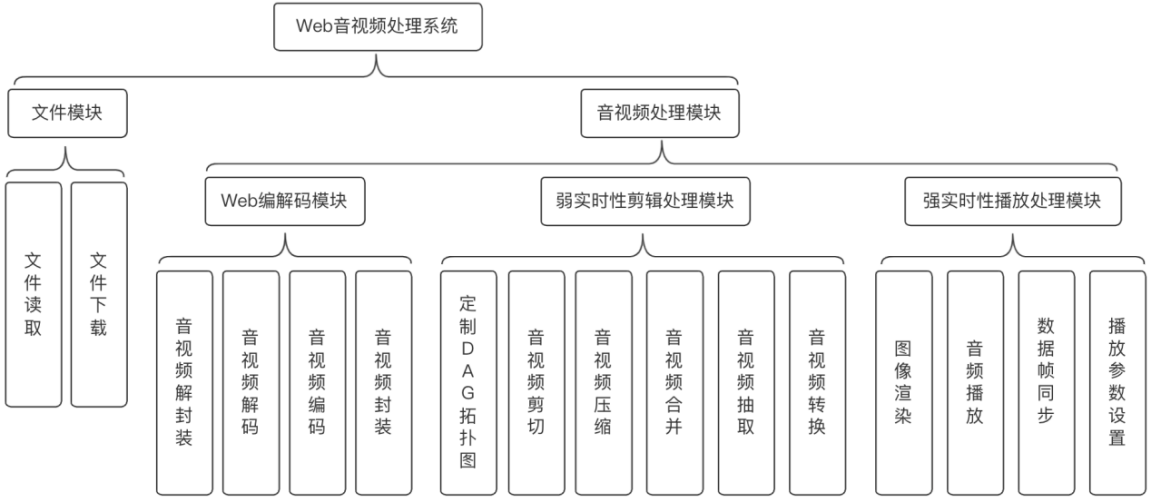


图3-6 音视频处理系统功能分解图

文件模块的主要作用是根据音视频处理模块的指令来控制流式读取文件的开始与停止，并提供处理完成后生成的新音视频文件的下载功能。音视频处理模块中Web编解码模块是核心基础模块，主要负责音视频文件的解封装、解码、编码、封装功能。弱实时性剪辑处理模块的主要功能是通过组合七种音视频剪辑处理单元，包含输入、输出、剪切、压缩、合并、抽取与转换，构建并拓扑执行DAG剪辑处理拓扑图来实现定制化地音视频剪辑功能。最后强实时性播放处理模块是基于WebGL渲染引擎、Audio播放引擎以及数据帧同步功能的基础上实现音视频的播放功能，再通过播放参数设置可以把播放细分为简单顺序播放、倍速播放以及Seek播放，同时也提供播放暂停、开始等基础的播放设置。

**3.4 本章小结**

本章第一部分内容根据Web音视频处理系统的最终目标逐次对三个功能性需求Web音视频编解码、弱实时性剪辑处理、强实时性播放处理进行需求分析，并对系统非功能性提出了五点要求，包括系统的安全性、易用性、性能、可靠性以及可维护性。其次本章第二部分内容是以需求分析为设计目标进行系统总体设计，包括系统架构设计和系统功能设计，清晰地描述了系统各个模块之间的协作关系以及各个功能的划分情况。

## 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**

**4.1.2 Web编解码流程设计**

**4.1.3 Web编解码实现**

## 4.2弱实时性处理模块的设计与实现

## **4.2.1 抽象剪辑处理单元**

**4.2.2 基于DAG的多剪辑任务处理实现**

## 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**

**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**

## 系统测试与分析

## 总结与展望

## 致谢

## 参考文献