

一种混合式多路高清实时监控系统设计及优化

程浩, 徐涛, 郑博方

(浙江大华技术股份有限公司, 浙江 杭州 310051)

摘要: 随着智能交通、平安城市等安防工程的推进,多路高清视频实时监控的需求日益增长,但现有B/S架构的视频监控系统在处理多路高清视频时存在性能不足的问题。因此,提出一种混合式多路高清实时播放方法,结合JavaScript和C++的优势,通过JavaScript实现多路高分辨率实时播放的UI交互界面,提高C/S架构的UI界面开发效率。同时,使用C++完成多路高分辨率视频的同时解码、渲染等复杂业务,实现低延时、低资源消耗、稳定的多路高分辨率视频播放系统。实验结果表明,相比传统的HTML5方案,该方法解码延迟减少了32.9 ms,稳定在约49 ms,解码帧数稳定在24帧;相比WebAssembly方案,CPU使用率降低了69.1%,内存占用减少了417 MB,解码延迟显著缩短。该系统充分发挥了两种语言的优势,提高了视频监控系统的性能和用户体验,为视频监控系统的发展提供了新思路。

关键词: C/S框架;UI界面;多路高清视频;实时监控系统;资源消耗;解码延迟

DOI:10.11907/rjtk.251219

开放科学(资源服务)标识码(OSID):

中图分类号:TP319

文献标识码:A

文章编号:1672-7800(2025)007-0104-07



Design and Optimization of a Hybrid Multi-Channel HD Real-Time Monitoring System

CHENG Hao, XU Tao, ZHENG Bofang

(Zhejiang Dahua Technology Co., Ltd., Hangzhou 310051, China)

Abstract: With the advancement of intelligent transportation and safe city projects, there is a growing need for real-time monitoring of multiple high-definition video channels. However, existing B/S-based video surveillance systems are inadequate in handling multiple high-definition video streams. This paper presents a dual-language hybrid method for real-time playback of multiple high-definition video channels, combining the strengths of JavaScript and C++. JavaScript is used for the UI interaction interface of real-time playback, enhancing UI development efficiency in the C/S architecture. C++ handles simultaneous decoding and rendering of multiple high-resolution videos, creating a stable, low-latency, and low-resource multi-channel high-resolution video playback system. Testing over 100 consecutive trials showed that, compared with the traditional HTML5 solution, the proposed method reduces decoding delay by 32.9 ms, stabilizing it at around 49 ms with a stable 24 frames per second. In contrast to WebAssembly, it cuts CPU usage by 69.1%, reduces memory consumption by 417 MB, and shortens decoding delay significantly. This system leverages both languages' advantages, boosting video surveillance system performance and user experience, and offers new ideas for video surveillance system development.

Key Words: C/S architecture; UI interface; multiple high-definition videos; real-time monitoring system; resource consumption; decoding latency

0 引言

随着智能交通、平安城市等安防工程的不断发展,视

频监控系统在社会安全防范中的重要性日益凸显。近年来,视频监控市场呈现快速增长的趋势^[1]。然而,随着网络带宽的增加和流媒体技术的不断进步,用户对视频监控系统的需求也不断增长,特别是在多路高清视频实时播放

收稿日期:2025-03-27

扫描二维码阅读全文:



基金项目:浙江省“尖兵领雁+X”科技计划项目(2025C01065)

作者简介:程浩(1985-),男,硕士,浙江大华技术股份有限公司工程师,研究方向为声纹识别、智能安防;徐涛(1990-),男,硕士,浙江大华技术股份有限公司高级工程师,研究方向为分布式存储;郑博方(1990-),男,浙江大华技术股份有限公司职员,研究方向为视频大数据分析挖掘、知识图谱。本文通讯作者:程浩。

方面^[2]。

浏览器/服务器架构(Browser/Server, B/S)因其便捷的访问方式,无需额外安装应用程序,通过浏览器即可实现视频监控功能,体现了极高的便捷性^[3]。此外,该架构在界面美观性与用户体验方面也表现出色,能够很好地满足用户对界面精美、操作简单的需求。因此,其已成为视频监控领域的主流架构选择^[4]。B/S架构的网页开发主要采用JavaScript语言。JavaScript是一种高级、动态、解释型的编程语言,其语法简单、直观,容易快速上手,并且具有丰富的UI(User Interface)框架库,所以在UI界面开发方面有着高效、更易维护的优势^[5]。

在浏览器中,JavaScript采用单线程模式运行,若某一任务过度占用CPU时长,例如执行复杂的计算或长时间的I/O操作,很可能会使网页出现卡顿现象,甚至完全失去响应能力^[6]。因为JavaScript是解释型语言,在运行过程中需要实时解释和执行代码,而C++等编译型语言在编译后会生成机器码,能够直接在硬件上高效运行^[7]。所以,当涉及到大规模数据处理,例如视频解码以及复杂计算任务时,JavaScript的效率往往不及C++^[8]。尤其是在多路高分辨率视频监控场景下,迫切需要一个稳定、低延时、低资源消耗的视频监控系统,而B/S架构难以满足性能需求。

近年来,随着HTML5、WebAssembly、WebRTC等前端新技术的不断涌现,B/S架构在视频监控系统的发展迎来了新契机^[9-12]。陈璐^[13]创新性地提出一种在浏览器网页中播放视频的方法,该方案巧妙地借助WebSocket协议实现客户端与服务器的通信,并充分利用HTML5技术进行视频渲染。虽然该方法在开发效率和兼容性方面展现出明显优势,但在面对多路高清监控视频实时播放场景时,仍不可避免地暴露出CPU占用高的问题。谈鸿韬等^[14]提出一种基于HTML5的Web无插件播放安防实时视频的方法,通过转码技术将视频流转码为H.264格式,并借助HTML5的MSE接口实现视频播放。刘伟杰等^[15]通过视频编码技术、音频编码RTP/RT-CP流媒体网络传输协议等多种技术,解决了高清网络监控传输带宽的压力和网络延迟问题。目前B/S架构在安防高清视频监控中能较好地解决单路4K视频的实时播放卡顿问题,但在多路视频实时播放的情况下,B/S架构的局限性依然明显,难以同时满足低延时、低资源占用以及稳定性等多重要求。

在多路高清视频监控系统中,曲洪权等^[16]采用客户端/服务器(C/S)架构,合理地将计算任务分配到C/S两端,分担了服务器的压力,让服务端可以处理更多视频码流数据。同时,客户端的运行不会占用大量资源,提升了用户体验。郝朝等^[17]同样采用C/S架构,利用C++高性能、能直接调用硬件资源的特点,多线程协作完成解码显示,解决多路视频同步回放的稳定性问题。然而,C++语言的复杂性以及缺乏丰富的UI工具库和框架,使其在开发UI界面时面临诸多挑战。相较于JavaScript等语言,C++开发

UI界面的效率较低,维护成本也较高。具体而言,C++的语法相对复杂,学习成本较高,开发者需要深入理解内存管理、指针操作等概念,从而在一定程度上增加了开发的难度和复杂性。而且C++缺乏像JavaScript那样丰富成熟的UI框架和库,如React、Angular、Vue等,这些框架为JavaScript开发者提供了大量现有组件和工具,大大简化了开发流程。另外相比JavaScript,C++开发同样的UI界面需要编写更多代码,导致开发周期变长。

因此,针对B/S架构在多路高分辨率视频监控场景下的性能瓶颈,以及C/S架构中C++语言在UI界面开发方面的局限性,本文提出一种双语言混合式开发的多路高清视频实时播放系统。该系统充分发挥了JavaScript和C++的各自优势,通过使用JavaScript的React框架实现多路高分辨率实时播放的UI交互界面,提高了C/S架构的UI界面开发效率。同时,利用C++完成多路高分辨率视频的解码、渲染等复杂业务逻辑,从而实现低延时、低资源消耗、稳定的多路高分辨率视频播放系统。这种混合式开发方法不仅解决了B/S架构在性能方面的不足,而且克服了C/S架构中UI界面开发效率低、维护成本高的问题,为视频监控开发提供了一种高效、稳定、灵活的解决方案。

1 客户端设计

该系统采用双语言混合式开发模式,结合了JavaScript和C++的优势。其中,JavaScript用于实现多路高分辨率实时播放的UI交互界面,C++用于完成多路高分辨率视频的同时解码、渲染等复杂的业务逻辑。该架构既能发挥JavaScript在UI开发方面的高效性,又能利用C++在性能方面的优势,实现低延时、低资源消耗、稳定的多路高分辨率视频播放系统。系统整体架构如图1所示,主要包括客户端、服务器。其中,客户端主要包括界面模块、解码渲染引擎、通信模块和基础库。

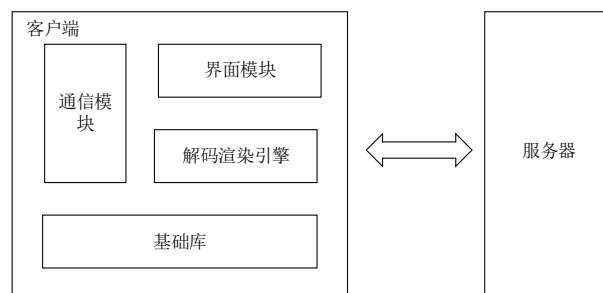


Fig. 1 System architecture

图1 系统架构

客户端负责与用户交互、多路视频的解码显示,服务器负责向客户端发送流数据。系统整体流程如下:客户端启动后,用户输入服务器URL地址;服务器向客户端返回网页信息,客户端的解码渲染引擎利用Qt的QWebEngineView对象完成网页的渲染和显示;解码渲染引擎中的窗口模块创建WebSocket服务,用于与网页通信;用户点击页

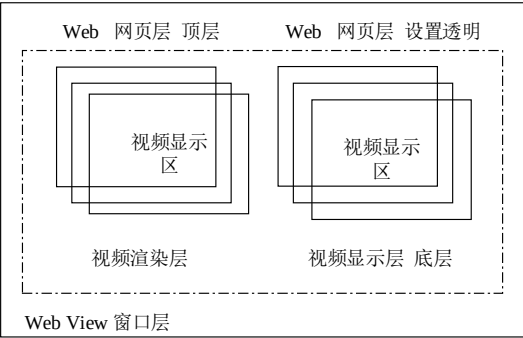


Fig. 3 Window overlay structure
图3 窗口叠加结构

问题。目前 JavaScript 调用 C++ 的方法包括:①N-API 将 C++ 代码编译为原生模块,然后 JavaScript 直接调用编译后的代码;②使用 Foreign Function Interface 技术的三方库直接调用 C++ 动态库;③使用 WebAssembly 技术调用。但是,以上方法都无法满足多视频、高分辨率场景下的需求。

本文采用一种类似进程间通信的方式解决该问题。UI 界面模块与通信模块的连接采用 WebSocket 协议。UI 界面模块通过 JavaScript 的 WebSocket 对象创建一个 Web-Client,并向 WebClient 传入服务端的 IP 和端口建立 Web-Socket 连接。在 UI 界面模块与通信模块完成连接后,双方即可进行双向通信。通信内容采用 JSON 格式,协议中包括操作命令、数据参数、视频资源的 ID 等。通信模块采用 C++ 实现 WebSocket 服务。在收到 Web 客户端的消息后,通信模块对 JSON 格式的数据进行序列化与反序列化处理,完成 JSON 格式化消息与 C++ 功能接口的映射。也就是说,UI 界面模块通过 WebSocket 向通信模块发送一串 JSON 消息即可直接调用通信模块的 C++ 实现功能。该方法使 JavaScript 能动态调用 C++ 模块,从而实现 UI 界面与业务逻辑的解耦。

在多路高清视频实时播放系统中,视频解码是性能瓶颈的关键环节,尤其是在处理多路高清视频流的过程中。为了提升解码效率并降低 CPU 负载,本文系统的渲染模块基于 FFmpeg 实现硬件加速解码技术^[18]。该技术利用图形处理单元(GPU)的并行计算能力,将复杂的视频解码任务从 CPU 转移到 GPU 上,从而实现高效、低延迟的视频处理。

硬件加速解码的实现步骤如下:首先确保系统已安装 NVIDIA 驱动和 CUDA 工具链,并在编译 FFmpeg 时启用 `-enable-cuda` 和 `-enable -cuid` 选项以支持 CUDA 加速,同时采用其异步解码流水线设计^[19]。在初始化阶段,利用 `avformat_open_input()` 打开输入流,通过 `avformat_find_stream_info()` 获取流信息并确定视频流索引。然后,调用 `avcodec_find_decoder()` 函数匹配 H.264 的解码器,并通过 `avcodec_alloc_context3()` 函数创建解码器上下文(dec_ctx)。接下来利用 `av_hwdevice_ctx_create()` 函数创建 CUDA 硬件设备上下文(hw_device_ctx),并将其赋值给 `dec_ctx->hw_device_ctx` 对象,同时设置像素格式为

AV_PIX_FMT_CUDA,以此启用 GPU 解码。完成上述操作后,使用 `avcodec_open2()` 打开解码器。在解码过程中,循环调用 `av_read_frame()` 读取数据包,并依次通过 `avcodec_send_packet()` 和 `avcodec_receive_frame()` 进行解码,解码后的帧(格式为 AV_PIX_FMT_CUDA)将直接存储在 GPU 显存中。若需 CPU 进行处理,则用 `av_hwframe_transfer_data()` 将显存数据回传至内存。整个过程结束后,释放解码上下文及资源。该方法通过卸载 CPU 解码任务至 GPU,可适用于高分辨率、高帧率视频的实时处理及流媒体分析场景。

本文系统的拉流流程具体如下:客户端启动 WebSocket 服务,UI 界面模块通过 JavaScript 创建的 WebClient 与该服务建立连接。连接成功后,UI 界面模块发送视频播放请求,WebSocket 服务接收请求并将其分发给通信模块。通信模块收到消息即向服务器发起拉流请求,服务器收到请求后将视频流数据返回给通信模块。通信模块收到数据,将其传递给解码渲染引擎,解码渲染引擎完成解码和渲染显示。视频播放流程如图 4 所示。

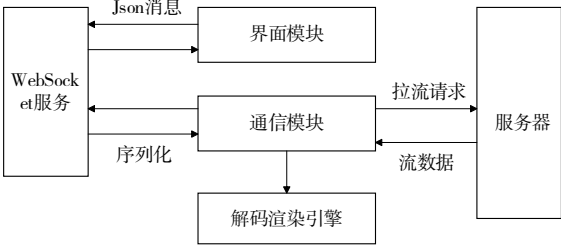


Fig. 4 Video playback flow
图4 视频播放流程

2 实验与结果分析

实验网络环境模拟真实场景构建,采用双核工业交换机,搭配 2 芯单模千兆光纤搭建出自愈环网网络。该单环具备强大的数据传输能力,可支持 96 路 4 K 视频的同时传输,全方位确保网络的高效性、稳定性与可靠性,以精准适配并还原真实复杂的网络工况。实验系统由监控视频源、服务器和客户端 3 部分组成。监控视频源采用浙江大华 4 款 IPC-HFW5541E 系列的 4 K 网络摄像机,测试中采用同样的配置:视频编码方式为 H.264/H.265,分辨率为 3 840×2 160(4 K),25 fps 的帧率,8 192 kbps 的码率。4 款相机的型号如表 1 所示。

Table 1 Network camera model list
表1 网络摄像机型号列表

项目	相机 1	相机 2	相机 3	相机 4
型号	IPC-HFW5541E-ZE	IPC-HFW5541E-Z	IPC-HFW5541E-4ZS	IPC-HFW5541E-41E

客户端采用 Windows10 企业版系统,测试终端参数如表 2 所示。

为全面检验系统性能的稳定性和可靠性,采用 4 款同

Table 2 Client-side test terminal parameters

表2 客户端测试终端参数

终端	处理器	GPU	内存	浏览器
联想	Intel(R) Core(TM)i5 -10500	Intel UHD Graphics 630	16 G DDR4	Chrome

系列的4K相机,分别进行1、2、4、8路多组实验,每组路数下均进行100次连续测试。将本文提出的双语言混合式多路高清视频实时播放方法(以下简称本文系统)与HTML5方案和WebAssembly方案进行比较,综合评估不同方法在多路高清视频播放中的性能表现。在评估过程中,本文关注解码帧率、CPU使用率、GPU使用率、内存占用和解码延迟5个关键指标,因为这些指标能够直接反映系统的性能和资源消耗情况。最后利用Kruskal-Wallis方差统计方法阐述本文系统的稳定性^[20]。具体结果如表3—表5所示,表中的所有数据均为100次实验所得的平均值。表6展示了在不同解码路数下,CPU与GPU使用率的均值、标准差、标准误差、误差范围及其对应的95%置信区间。随着视频解码路数的增加,CPU使用率从7.4%上升至30.9%,GPU使用率从24.2%上升至100%。置信区间显示,各路数下CPU和GPU使用率均有较小的波动,表明系统在不同负载下的资源消耗具有稳定性和可预测性。表7—表8分别表示4路和8路场景下的方差对比与方差降幅。

Table 3 Performance of Web HTML5 scheme

表3 Web HTML5方案性能

视频路数	帧率/FPS	CPU使用率/%	GPU使用率/%	内存占用/MB	解码延迟/ms
1	25	5.5	25.2	132.2	5.3
2	25	7.4	55.8	311.4	8.4
4	21	13.1	96.1	704.6	26.8
8	11	24.4	100	1 353.9	82.1

Table 4 Performance of WebAssembly scheme

表4 WebAssembly方案性能

视频路数	帧率/FPS	CPU使用率/%	GPU使用率/%	内存占用/MB	解码延迟/ms
1	25	32.4	28.2	152.2	6.5
2	25	68.2	70.3	405.3	8.7
4	22	81.6	96.8	728.5	25.7
8	10	100	100	1 626.3	78.5

Table 5 Performance of the proposed system

表5 本文系统性能

视频路数	帧率/FPS	CPU使用率/%	GPU使用率/%	内存占用/MB	解码延迟/ms
1	25	7.4	24.2	112.3	3.9
2	25	11.2	51.1	211.6	5.2
4	24	18.3	93.2	601.7	13.9
8	22	30.9	100	1 289.3	49.2

实验结果表明,本文提出的基于硬件加速的视频解码系统在8路4K视频处理场景下展现出较好的性能优势与稳定性。在解码效率维度,系统随视频路数增加呈渐进式

Table 6 95% confidence intervals of CPU and GPU usage across different decoding stream numbers

表6 不同解码路数下CPU与GPU使用率的95%置信区间

路数	指标	均值/%	标准差/ σ	标准误差/误差	误差范围	置信区间(95%)
1	CPU使用率	7.4	0.787 4	0.078 74	0.154 4	(7.245 6, 7.554 4)
	GPU使用率	24.2	0.781 0	0.078 10	0.153 1	(24.046 9, 24.353 1)
2	CPU使用率	11.2	0.793 7	0.079 37	0.155 7	(11.044 3, 11.355 7)
	GPU使用率	51.1	0.781 0	0.078 10	0.153 1	(50.946 9, 51.253 1)
4	CPU使用率	18.3	0.781 0	0.078 10	0.153 1	(18.146 9, 18.453 1)
	GPU使用率	93.2	0.787 4	0.078 74	0.154 4	(93.045 6, 93.354 4)
8	CPU使用率	30.9	0.787 4	0.078 74	0.154 4	(30.745 6, 31.054 4)
	GPU使用率	100	0.793 7	0.079 37	0.155 7	(99.844 3, 100.155 7)

Table 7 Comparison of variances between 4 and 8 channels

表7 4路与8路方差对比

	本文系统	HTML5	WebAssembly
4路解码帧数方差	0.63	1.42	1.22
8路解码帧数方差	0.88	3.82	3.51

Table 8 Comparison of the reduction in variance for 4 and 8 channels

表8 4路与8路方差对比降幅

对比项	降幅(HTML5)/%	降幅(WebAssembly)/%
4路场景	55.6	48.4
8路场景	76.9	74.9

性能衰减特性,与HTML5及WebAssembly方案形成鲜明对比:单/双路解码时,三者均维持25 fps满帧率,但在4路解码时,本文系统保持平均帧数在25 fps,而HTML5与WebAssembly的帧率分别下降了12%至22 fps($\Delta = 3\text{fps}$);在8路解码时差异进一步扩大,本文系统帧数仍保持22 fps(衰减率12%),而对比方案HTML5和WebAssembly分别降低了10 fps、11 fps,衰减率分别达到60%和56%。通过Kruskal-Wallis方差分析发现,本文系统帧率的稳定性显著优于对照组,具体表现为:4路实际解码的方差值为0.63,较HTML5与WebAssembly的1.42、1.22分别降低了55.6%和48.4%;8路实际解码的方差为0.88,较对照组3.82、3.51的降幅达到76.9%~74.9%(见图5、图6)。该特性验证了系统在动态负载下的鲁棒性,可满足高密度视频分析场景下的实时性需求。

Kruskal-Wallis和方差降幅计算公式如下:

$$H = \frac{12}{N(N+1)} \sum_{i=1}^k \frac{R_i^2}{n_i} - 3(N+1) \quad (1)$$

$$\text{降幅} = \frac{\text{对照组方差} - \text{本文系统方差}}{\text{对照组方差}} \times 100\% \quad (2)$$

因为本文系统采用CUDA异步解码流水线设计,通过显存复用策略将4路解码内存占用优化至601.7 MB,较HTML5与WebAssembly的704.6 MB、728.5 MB分别减少了14.6%和17.4%。尽管受限于Intel UHD Graphics 630的硬件算力(GPU占用率100%),但系统在8路解码情况下仍将CPU占用率控制在30.9%(标准差值在1.48),较WebAssembly方案的100%降低了69.1%(见表6)。

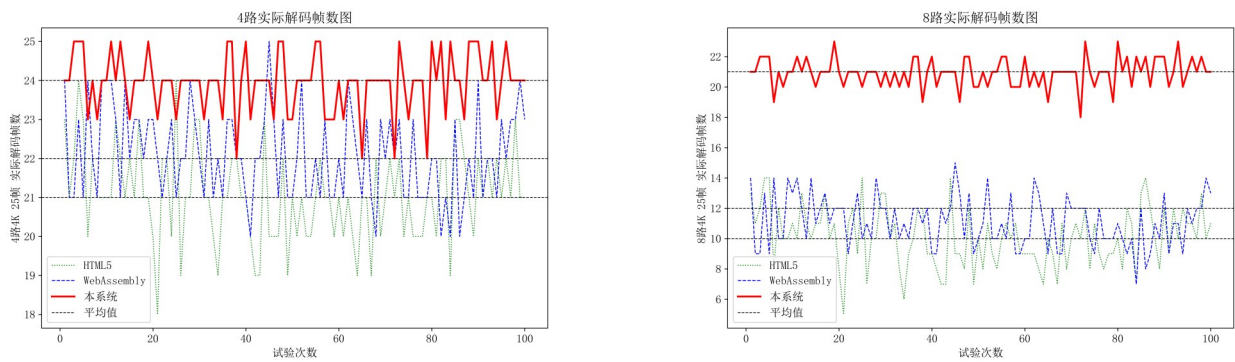


Fig. 5 Actual decoding frame rate diagram of 3 schemes for 4 and 8 channels

图5 4、8路3种方案实际解码帧数图

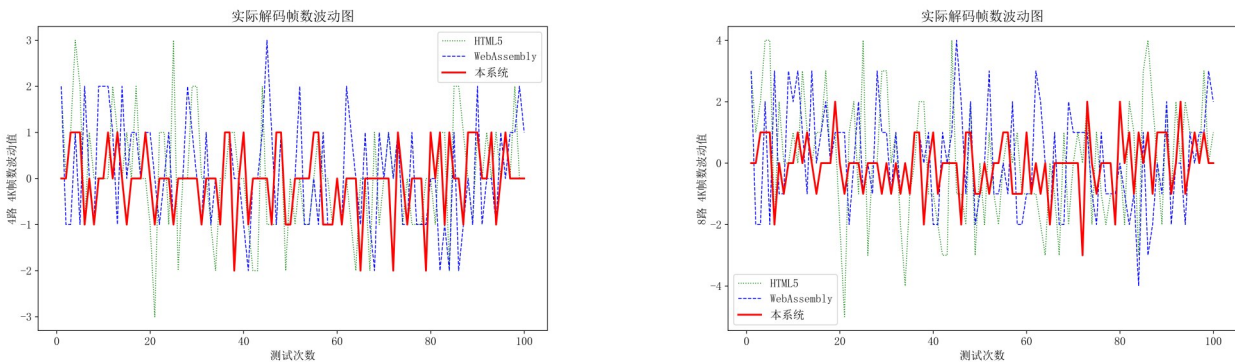


Fig.6 Fluctuation diagram of decoding frame rate of 3 schemes for 4 and 8 channels

图6 4、8路3种方案解码帧数波动图

3 结语

本文创新性地提出一种混合式多路高清实时播放方法,结合了JavaScript和C++各自的优势,有效破解了现有B/S架构视频监控系统在多路高清视频实时播放过程中遭遇的性能瓶颈问题。实验结果表明,与HTML5和WebAssembly方案相比,本文系统不仅能够有效降低CPU占用率,而且能显著减少内存占用,并大幅缩短解码延迟。特别是在高路数视频播放的场景下,本文方法的优势更为突出,不仅降低了系统资源消耗,而且极大地提升了用户的操作体验。

尽管本文方法在性能优化上效果显著,但仍存在进一步提升的空间。例如,当前方案对硬件解码的依赖性较高,对于一些低性能的GPU或无GPU的设备,其适用性可能受到限制。未来的工作将集中在以下几个方面:一是进一步优化硬件解码算法,降低硬件对GPU性能的依赖,使其能够在更多低性能设备上高效运行;二是探索5G网络和边缘计算技术在视频监控中的应用,提升系统的稳定性和响应速度;三是研究如何在更复杂的网络环境下保持视频播放的流畅性,以满足不同应用场景的需求。

综上,本文提出的混合式多路高清视频实时播放方法为安防监控领域提供了一种高效、低资源消耗的解决方案,具有重要的实际应用价值和广阔的发展前景。未来将继续探索相关技术的优化和创新方法,以满足日益增长的视频监控需求,推动视频监控技术的进一步发展。

参考文献:

[1] ZHENG T L. Research on the development strategy of DH Company[D]. Hangzhou: Zhejiang University of Technology, 2017.
郑天伦. DH公司竞争战略研究[D]. 杭州: 浙江工业大学, 2017.

[2] CHEN J. Research on the development strategy of KD Co. Ltd. [D]. Wuhan: Huazhong University of Science&Technology, 2017.
陈军. KD公司发展战略研究[D]. 武汉: 华中科技大学, 2017.

[3] CHEN Y. Design and implementation of intelligent chemical park management system based on B/S architecture[D]. Nanjing: Nanjing University of Information Science&Technology, 2024.
陈洋. 基于B/S架构的智慧化工园区管理系统设计与实现[D]. 南京: 南京信息工程大学, 2024.

[4] GOU T W. Design and implementation of Web audio and video process system based on FFmpeg[D]. Nanjing: Southeast University, 2023.
缙通旺. 基于FFmpeg的Web音视频处理系统的设计与实现[D]. 南京: 东南大学, 2023.

[5] WU Y J, KOCESKI K, PEDRINI M, et al. Pattern template manifest for live video streaming[C]// West Lafayette: 2024 IEEE 26th International

- Workshop on Multimedia Signal Processing, 2024.
- [6] JIAO Y M H. Design of miniaturized video conference system based on B/S architecture[D]. Harbin: Harbin Engineering University, 2021.
焦阳明皓. 基于B/S架构的小型化视频会议系统的设计[D]. 哈尔滨: 哈尔滨工程大学, 2021.
- [7] ZEBAR I M I, ZEEBAREE S R M, YASIN H M. Real time video streaming from multi-source using client-server for video distribution[C]//2019 4th Scientific International Conference Najaf, 2019: 29-30.
- [8] DIAZ D, STOLARZ D, AGUERRE J. Toward real-time video streaming over WebRTC data channels to support supplementary video codecs and formats in the browser[C]//Chicago: 2024 IEEE International Conference and Expo on Real Time Communications at IIT, 2024.
- [9] TUSHAR, MOHAN B R. Comparative analysis of JavaScript and WebAssembly in the browser environment[C]// 2022 IEEE 10th Region 10 Humanitarian Technology Conference, 2022: 232-237.
- [10] QIAN F, XIA J, YANG D, et al. Design and application of comprehensive management and control platform for non-road mobile machinery based on B/S architecture[J]. Computer Application and Software, 2024, 41(12): 384-389.
钱枫, 夏俊, 杨栋, 等. 基于B/S架构的非道路移动机械综合管控平台设计与应用[J]. 计算机应用与软件, 2024, 41(12): 384-389.
- [11] ZHANG Z W. Research on secure and high-performance server-side Webassembly runtime[D]. Nanjing: Southeast University, 2024.
张之伟. 面向安全, 快速的服务器端Webassembly运行时研究[D]. 南京: 东南大学, 2024.
- [12] GE W B. Design and implementation of Web-based plugin scalable video live broadcasting system[D]. Beijing: Beijing University of Posts and Telecommunication, 2023.
葛文博. 基于Web的插件式可扩展视频直播系统的设计与实现[D]. 北京: 北京邮电大学, 2023.
- [13] CHEN L. A method and system for playing a video in a browser Web page[P]. China, CN112422508A, 2020-10-19.
陈璐. 一种在浏览器网页中播放视频的方法及系统[P]. 中国, CN112422508A, 2020-10-19.
- [14] TAN H T, LIU S H, LIAO Q Z. A method for playing security real-time video without plugin on HTML5-based Web[P]. China, CN113556579 B, 2021-06-07.
谈鸿韬, 刘树惠, 廖乔治. 一种基于HTML5的web无插件播放安防实时视频的方法[P]. 中国, CN113556579B, 2021-06-07.
- [15] LIU W J, LI B. Real-time high-definition audio and video monitoring system based on B/S architecture[J]. Modern Electronics Technique, 2019, 42(12): 45-47.
刘伟杰, 李博. 基于B/S架构实时高清音视频监控系统[J]. 现代电子技术, 2019, 42(12): 45-47.
- [16] QU H Q, TAN L. Design and implementation of passenger flow detection system based on C/S architecture[J]. Journal of Jishou University, 2019, 40(4): 19-23.
曲洪权, 谭磊. 基于C/S架构的客流检测系统设计与实现[J]. 吉首大学学报, 2019, 40(4): 19-23.
- [17] HAO Z, NIE Y J. Synch playback technology of multi-channel airborne videos based on C/S mode[J]. Computer Technology and Development, 2019, 29(4): 144-148.
郝朝, 聂瑶佳. 基于C/S模式的机载多路视频同步回放技术[J]. 计算机技术与发展, 2019, 29(4): 144-148.
- [18] MEI H, XU Z X, ZHOU M Z, et al. Design and implementation of an FFmpeg decoding hardware acceleration monitoring platform[J]. Industrial Control Computers, 2022, 35(8): 39-40.
梅寒, 徐子潇, 周明政, 等. 一种FFmpeg解码硬件加速监控平台设计与实现[J]. 工业控制计算机, 2022, 35(8): 39-40.
- [19] LIU Y, DONG S J. Acceleration algorithm of multi-channel video image stitching based on CUDA kernel function[J]. Computer Science, 2022, 49(6): 441-446.
刘云, 董守杰. 基于CUDA核函数的多路视频图像拼接加速算法[J]. 计算机科学, 2022, 49(6): 441-446.
- [20] ZUO Y. Target detection system of agricultural economic output efficiency based on Kruskal algorithm[C]// 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications, 2022: 1-5.

(责任编辑: 黄 健)