论文题目:面向云桌面的图像压缩优化研究

学位类别：工学硕士学位

学科专业：计算机科学与技术

年　　级：2021

研 究 生：谭皓星

指导教师：戴元顺

二零二四年三月

国内图书分类号：TP393.027　　密级：公开
国际图书分类号：004

西 南 交 通 大 学
研 究 生 学 位 论 文

面向云桌面的图像压缩优化研究

年　　级　2021
姓　　名　谭皓星
申请学位级别　硕士
专　　业　计算机科学与技术
指 导 老 师　戴元顺

二零二四年三月二十一日

Classified Index: TP393.027
U.D.C: 004

Southwest Jiaotong University

Master Degree Thesis

RESEARCH ON IMAGE COMPRESSION OPTIMIZATION FOR CLOUD DESKTOP

Grade: 2021

Candidate: Haoxing Tan

Academic Degree Applied for : Master

Speciality: Computer Science

Supervisor: Yuanshun Dai

Mar.21,2024

西南交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复印手段保存和汇编本学位论文。

本学位论文属于

1．保密□，在　　年解密后适用本授权书；

2．不保密☑，使用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：　　指导老师签名：

日期：　　日期：

西南交通大学硕士学位论文主要工作（贡献）声明

本人在学位论文中所做的主要工作或贡献如下：

（1）云桌面协议是云桌面系统中的关键技术之一，云桌面协议很大程度上决定了用户的体验。目前大部分云桌面协议对带宽有着极高的要求，其原因是常见的云桌面协议，如：RDP 协议、VNC 协议、ICA 协议、SPICE 协议等在进行图像编码时大多使用 JPEG 编码，导致在一些高运动场景性能较差，表现为带宽占用急剧升高甚至出现卡顿、失帧、画面撕裂等严重影响用户体验的情况。针对上述情况，采用了网格量化 TCQ 方法来优化 JPEG 编码过程，提高了 JPEG 的编码效率，从而降低云桌面系统中图像传输的带宽需求。在多个数据集上进行了对比实验，实验的结果显示改进后的方法能够优化 JPEG 编码的压缩率。

（2）针对当前云桌面协议在高运动场景下表现不佳的问题，深入研究和分析了现有的渐进式压缩方法，在此基础上，采用一种改进的扫描分割策略来优化 JPEG 的渐进式压缩技术。该策略通过对 DCT 系数按特征进行分组，并根据不同的优先级分别对其进行编码，首先传输携带图像信息较多的序列，从而有效提升了压缩效率。通过一系列精心设计的对比实验，在相关数据集上验证了该优化方案，将其性能与原始方案的实验结果进行了对比。实验结果清晰地展示了改进方案在压缩率方面有所提升。

（3）基于 X2Go Kdrive 平台对上述两种技术进行了实现，对实现前后的云桌面传输性能进行了对比测试，对实验数据进行了详细的展示和分析。实验结果中优化后的云桌面协议在客户端 CPU 上较优化前降低了 6.25% - 42.58%，网络带宽占用降低了 13.73% - 28.41%，表明本文提出的优化策略能够有效提升云桌面传输协议的性能，从而更好地满足云计算环境对资源利用率和网络适应性的需求。

本人郑重声明：所呈交的学位论文，是在导师指导下独立进行研究工作所得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中作了明确说明。本人完全了解违反上述声明所引起的一切法律责任将由本人承担。

学位论文作者签名：

日期：

摘 要

5G 技术普及和网络技术突飞猛进所带来的网络速度的显著提升，不仅为公众提供了更加迅捷的连接速度，使得计算和存储资源的访问变得更为便捷，而且也催生了云计算市场的迅猛发展。随着硬件投入成本的降低，云计算正在推动多个行业向平台化和智能化的方向转型。通过集中式的资源管理和虚拟化技术，云计算实现了资源的高效部署和利用。虚拟化技术，特别是在服务器、桌面和应用三个层面的应用，为创建多个虚拟环境和专用资源提供了强大的支持。其中，应用虚拟化尤为突出，用户仅需通过云桌面协议便能够访问服务器上的图像和音视频资源。云桌面技术利用远端服务器的处理能力，实现了桌面环境、应用程序和数据的集中存储和管理，相较于传统桌面模式具有显著的优势。云桌面系统中客户端和服务端的数据均通过网络传输，云桌面协议是其中的关键技术之一。目前主流的云桌面协议有 RDP 协议、ICA 协议、PCoIP 协议、SPICE 协议等，它们在功能上大多都较为完整，但在低带宽、瘦客户端等特定云桌面环境下仍存在许多能够优化的空间。

本文通过对当前主流云桌面协议的图像压缩方法进行分析，发现尽管云桌面技术已广泛应用，但其图像压缩方面仍存在一定的优化空间，例如过高的传输带宽需求和客户端计算负担等问题。基于此，本文深入分析了主流云桌面协议中采用的 JPEG 图像压缩算法，揭示了其在低带宽和瘦客户机等特定环境下的不足，并针对其不足采用了两种优化方案：一是在 JPEG 编码过程中引入了网格量化 TCQ 方法。通过创建一个量化系数候选集，评估得到失真与比特率之间平衡最佳的量化系数，改进 JPEG 的量化过程，以此提高 JPEG 编码的效率，降低云桌面系统中图像传输的带宽需求；二是基于量化后 AC 系数的分布特征，采用一种新的扫描分割方法，将量化后的 AC 系数分成多个序列，利用连续零序列的特性，在保证图像质量的前提下提升云桌面协议图像传输过程的压缩性能。

本文针对当前云桌面协议在低带宽、瘦客户端等特定云桌面环境下表现不佳的问题，在其原有图像压缩方法基础上，通过优化量化过程和扫描分割过程对其进行优化，并对优化前后的云桌面协议在多项性能指标上进行了对比测试。最终测试结果表明，优化后的云桌面协议在客户端 CPU 占用率和带宽占用方面均实现了显著改进，优化了云桌面环境下用户的整体体验。

关键字：云计算；虚拟桌面架构；X2Go；JPEG；TCQ

Abstract

The widespread adoption of 5G technology and the rapid advancements in network technology have not only provided the public with much faster connection speeds, making access to computing and storage resources more convenient but have also spurred the rapid development of the cloud computing market. With the reduction in hardware investment costs, cloud computing is driving the transformation of multiple industries towards platformization and intelligence. Through centralized resource management and virtualization technology, cloud computing achieves efficient deployment and utilization of resources. Virtualization technology, especially in the application at the server, desktop, and application levels, provides strong support for creating multiple virtual environments and dedicated resources. Among them, application virtualization is particularly prominent, where users can access graphics and audio-video resources on servers just through a cloud desktop protocol. Cloud desktop technology leverages the processing power of remote servers to achieve centralized storage and management of desktop environments, applications, and data, which has significant advantages over traditional desktop models. In cloud desktop systems, data from both clients and servers are transmitted over the network, and the cloud desktop protocol is one of the key technologies. Currently, mainstream cloud desktop protocols include RDP, ICA, PCoIP, SPICE, etc., which are mostly complete in functionality, but there is still much room for optimization in specific cloud desktop environments with low bandwidth and thin clients.

This article analyzes the image compression methods of current mainstream cloud desktop protocols and finds that although cloud desktop technology has been widely used, there is still room for optimization in image compression, such as the high demand for transmission bandwidth and the computational burden on clients. Based on this, this article delves into the JPEG image compression algorithm used in mainstream cloud desktop protocols, revealing its shortcomings in specific environments such as low bandwidth and thin clients. To address these shortcomings, two optimization schemes are proposed: one is the introduction of the Trellis Coded Quantization (TCQ) method in the JPEG encoding process. By creating a set of candidate quantization coefficients and evaluating to find the optimal balance between distortion and bit rate, the JPEG quantization process is improved to enhance the efficiency of JPEG encoding and reduce the bandwidth requirements for image transmission in cloud desktop systems. The second is a new scanning segmentation method based on the distribution characteristics of quantized AC coefficients, which divides the quantized AC coefficients into multiple sequences. By leveraging the properties of continuous zero sequences, this method improves the compression performance of the cloud desktop protocol's image transmission

process while ensuring image quality.

This article addresses the poor performance of current cloud desktop protocols in specific cloud desktop environments such as low bandwidth and thin clients. By optimizing the quantization and scanning segmentation processes on the basis of the original image compression method, the optimized cloud desktop protocols were tested against various performance metrics. The final test results show that the optimized cloud desktop protocols significantly improved in terms of client CPU usage and bandwidth occupancy, enhancing the overall user experience in cloud desktop environments.

key words: Cloud Computing; Virtual Desktop Infrastructur; X2Go; JPEG; TCQ

目 录

第 1 章 绪 论

1.1 研究背景与意义

随着 5G 技术的普及和网络技术的飞跃进步，我们见证了网络速度的显著提升，这为大众提供了更快的连接速度，让他们随时随地便捷地访问所需的计算和存储资源。与此同时，由于硬件投入成本的下降，云计算市场正不断膨胀，促进了多个行业朝着更加平台化和智能化的方向演变。我国的云计算市场增长势头迅猛，根据中国信息通信研究院的数据，2022 年市场规模已经攀升至 4550 亿元，同比增长幅度高达 40.91%，这一增长速度大大超过了全球平均的 19%。尽管面对全球宏观经济的多重挑战，我国的云计算[1]市场依然展现出了强大的逆周期性和抗风险的能力。预测表明，到 2025 年，这一市场的规模有望冲破万亿大关。这一切不仅标志着我国的云计算服务业务模式正在经历爆炸性的增长，而且指向了一个事实：随着云计算市场的持续扩展，相应的技术支撑也需同步创新与进步来满足不断上升的市场需求。

云计算通过统一的资源管理和虚拟化技术，实现了资源的集中部署。虚拟化技术[2]以单一物理环境为基础，构建了多个虚拟环境和专用资源。这项技术主要存在三种形式：包括服务器虚拟化[3]、桌面虚拟化[4]和应用虚拟化[5]。特别地，在应用虚拟化中，用户仅需经由云桌面传输协议[6]登录即可使用安装在服务器上的虚拟机中的应用，而相关的图像和音视频信息则通过云桌面协议被传递至用户的设备。云桌面通过将处理核心放在远端服务器[7]，使得桌面环境、应用和数据得以集中存储，同时配合虚拟化技术，相比于传统模式拥有数个显著优点：

1）集中化管理与维护：在云桌面环境中，所有操作系统和桌面应用都在虚拟化的数据中心内运行，便于管理人员统一进行维护。无论是系统升级还是应用更新，都由管理员在数据中心内统一执行，显著减轻了人力工作量。

2）增强的信息安全：由于用户终端不再存储操作系统和相关应用，减少了病毒侵害的风险，并在本地设备故障时保证数据的安全存放于云端，即使终端设备更换，数据同样安全无损。

3）便捷的灵活性：云桌面为用户提供了随时随地工作的能力，只要网络覆盖，用户即可接入自己的云桌面，从而满足移动办公和异地工作的需求。这一优势为现代工作方式提供了极大的方便，增强了工作的灵活性。

个人电脑作为一种集计算能力、存储空间、网络访问及操作系统于一体的独立硬件设备[8]，能够独立执行各类程序。与此相对的是云桌面技术[9]，它通过虚拟化手段，将个人电脑上的桌面环境和应用程序集中托管于服务器端，再通过网络将这些资源交付给各种不同类型的客户端设备。这些客户端设备可以是轻薄的瘦客户机、传统的台式机、便携的智能设备或平板电脑。由于计算和数据处理都被转移到服务端，因此客户端

设备的性能要求大为下降，它们主要负责提供用户界面的输入输出功能。在云桌面系统的构建中[10]，两个核心的技术要素尤为重要：一是服务器侧的虚拟化技术；二是云桌面传输协议。服务器虚拟化技术已被广泛研究并文献所记载，云桌面的虚拟化技术与之并无根本差异，故本文不作赘述。重点在于云桌面的传输协议[11]，它承担着将图形、图像和音频数据从服务器传输到用户终端的重任，同时还负责将用户的输入信息，如鼠标点击、键盘敲击及打印指令等回传到服务端。这种传输协议确保了服务端与终端之间顺畅有效的交互和数据流通。在数据传输过程中，涉及 TCP/IP 网络传输层的各种协议，通常选用 TCP 或 UDP 协议以实现不同的传输需求。

云服务需求的转变正从基础架构即服务（IaaS）向软件即服务（SaaS）迅猛发展[12]，吸引了越来越多企业和个人用户的目光。云平台[13]的一个关键优势是提供全面的端对端应用解决方案。相比之下，传统应用程序通常需要在用户本地机器上下载、安装，并经历繁琐的部署和更新过程，这不仅降低了效率，还可能导致计算机速度减慢以及隐私泄露和病毒感染的风险。此外，在传统的 C/S 架构云桌面连接中，用户需通过特定的瘦客户端[14]软件来访问虚拟桌面，这样的软件需要为不同的操作系统（如 Windows、Linux、Mac OS 等）单独开发，既增加了成本也增加了用户的负担，且它们通常缺乏跨平台兼容性。云桌面和基于 Web 的云桌面协议[15]的问题正在通过新一代 Web 技术得到解决，特别是 HTML5[16]标准的推出显著增强了浏览器功能，现在用户可以直接通过 Web 浏览器访问云应用程序，这无疑使采用 B/S 架构成为了最优选。这种基于浏览器的解决方案具有更广泛的平台兼容性和更低的维护成本，为用户提供了无需额外安装插件且独立于平台选择之外的便利性。因此，Web 技术提供了一个有效的策略来克服传统 C/S 架构的许多局限性，为用户创造了更为丰富和方便的云访问体验。

在主流云桌面协议的使用过程中，特别是当网络带宽受限时，传统图像压缩算法被用来确保云桌面的功能性，但这常常牺牲了用户的视觉体验。云桌面数据传输的核心在于桌面图像的高效传输，而图像压缩技术是实现这一点的关键。许多云桌面解决方案通常采用 JPEG 压缩，以减小图像数据量，节省传输所需的带宽。不过，JPEG 压缩在减小文件大小的同时，往往会丢失大量的颜色和结构细节，只保留亮度信息，这在带宽有限的场合下会导致重建的图像出现块状伪影、模糊和色彩失真。这不仅影响了图像质量，同时也降低了用户远程操作的精确性。传统的压缩方法在试图提高数据传输效率的同时，不幸地在图像质量和用户体验方面做出了妥协。因此，对于云桌面传输协议中的图像传输算法，寻求一个能够在保证图像质量及用户体验的前提下，有效减小使用带宽的优化方案，已成为一项迫切的技术挑战。

1.2 国内外研究现状

云桌面[17]技术随着时间的推移已经呈现出了各种部署形态，并不断地发展和演进。在云桌面领域，最早也是最广泛采用的部署模式是虚拟桌面基础设施（Virtual Desktop

Infrastructure，VDI)[18]。VDI 利用虚拟化软件在物理服务器上生成虚拟机环境，这些环境模拟独立的计算硬件，为用户提供个性化的桌面实例。简而言之，在 VDI 框架内，每个用户都被分配到一个专属的虚拟机实例，其中运行着他们的私有桌面计算环境。VDI 的核心在于虚拟桌面协议，这是保障服务器与客户端通信的网络通道的根本。由于整个系统的性能及用户体验依赖于数据的顺畅传输，优化网络数据传输过程是提升用户体验的关键，也是众多虚拟桌面协议[19]研究和改进的焦点所在。

随着云桌面技术的不断演进，出现了新的问题：一些用户的资源需求很低，但在传统的云桌面部署方案中，却可能被分配到了过量的资源。为了更有效地分配和利用资源，应用虚拟化[5]的概念被引入其中。应用虚拟化关注于以应用程序作为最基本的分配单位，不同的应用被载入到虚拟机中，并且这些应用可以被多个用户共享，也就是说，不同的用户可以通过服务器建立独立的应用程序连接。这种方式让每名用户在服务器上变身为一个或多个独立的进程，从而有效地提高了资源的利用效率。然而，不可忽视的是，尽管应用虚拟化在模式上有所创新[20]，但其底层依然是依赖于传统虚拟桌面技术的模式[21]。这种方法在提升资源管理效率的同时，也继承了一些虚拟桌面存在的挑战，尤其是在应用程序的交付和性能优化方面。

尽管云桌面的部署模式[22]千差万别，但都依赖于同一种决定性技术因素，即云桌面传输桌面协议。现如今大大小小的云桌面传输协议已有几十种[6]，市场占有率最高的云桌面协议是由几家主要的虚拟化解决方案提供商开发的。著名的协议包括 PCoIP 协议[23]、RDP 协议[24]、开源协议 SPICE[25]、ICA/HDX 协议[26,27]和开源的 RFB 协议[28]。

PCoIP（PC over IP，即通过 IP 的 PC）协议是一种用于云桌面服务的显示协议。该协议由 Teradici 公司开发，允许所有计算和图形处理在云或数据中心的主机计算机上进行，而只将显示像素传输到客户端设备。这意味着客户端设备不需要高性能的处理能力，因为它只负责显示接收到的像素数据。PCoIP 协议广泛应用于虚拟桌面基础设施（VDI）和远程工作站[29]解决方案中，为用户提供高质量的图形和视频体验，即使在低带宽和高延迟的网络环境下也能保持良好的性能。PCoIP 协议支持多种部署场景，包括云环境、数据中心以及混合环境[30]，能够满足不同规模和需求的企业的云桌面和虚拟化需求。通过优化带宽占用和提供高性能的远程访问体验，PCoIP 在远程办公、教育、医疗和媒体等多个领域得到了广泛应用。PCoIP 协议的数据是使用 UDP 协议来传输的，这也是其高效传输的原因。在传输过程中可能会遇到防火墙拦截的问题，此时就需要对防火墙进行特殊配置过滤特定端口的数据[31]。

由于其优秀的性能，PCoIP 协议在 2008 年被 VMware 公司在其产品中采用，同样亚马逊也在 2013 年在旗下的产品中使用了 PCoIP 协议作为传输数据的方式。

远程桌面协议（Remote Desktop Protocol，RDP）是一种由微软公司开发的多通道协议，旨在提供用户与远程计算机系统之间的虚拟桌面会话服务。自从 1990 年代中期

首次引入以来，RDP 已成为云桌面服务领域的核心技术之一，广泛应用于企业远程办公、虚拟化基础设施以及云计算服务中[29]。本文旨在从技术和应用的角度，探讨 RDP 协议的架构、功能特性及其在当代 IT 环境中的应用价值。

RDP 基于客户端/服务器模型，客户端应用程序通过网络与远程服务器建立连接，服务器端接收来自客户端的输入并返回图形界面输出[32]。RDP 的核心优势在于其高效的数据压缩、加密和传输机制，确保了远程会话的流畅性和安全性。RDP 协议通过高效的编码和压缩算法优化了数据传输，使其能够在带宽受限的环境下仍然保持良好的云桌面体验。此外，RDP 的适应性编码策略能够根据网络条件动态调整数据传输质量，保证最佳的用户体验[33]。然而，RDP 面临的挑战主要包括网络延迟和数据安全问题[34]。网络延迟对于图形密集型的应用尤为敏感，可能导致用户体验下降。尽管 RDP 采取了多种安全措施，但在高度敏感的环境中，额外的安全层或专用网络依然是推荐的做法。总体来说 RDP 作为一种成熟的云桌面协议，不仅提供了高效、安全的远程访问能力，而且通过不断的技术迭代，适应了现代网络环境[35]。

SPICE (Simple Protocol for Independent Computing Environments) 是一个开源的云桌面协议，最初由 Qumranet 公司开发，后来被 Red Hat 公司收购。SPICE 专为虚拟化环境设计，主要用于虚拟桌面基础设施 (VDI) 中，以提供高性能和高品质的图形和交互体验。与其他云桌面协议如 RDP 和 PCoIP 相比，SPICE 旨在为虚拟机提供更好的图形渲染和视频播放性能，同时也支持音频、视频、USB 设备重定向等功能[36]。SPICE 协议主要用于 VDI 和云计算环境中，特别是在需要高品质图形和视频渲染的场景下，如 CAD/CAM 设计、视频编辑和游戏等。在教育、健康保健和远程工作等领域，SPICE 也展现出了其优异的云桌面体验[37]。

虽然 SPICE 在虚拟化环境中提供了优秀的性能和体验，但它也面临一些挑战，如对客户端资源的高需求、在低带宽或高延迟网络中的性能下降等。此外，SPICE 的支持和兼容性主要集中在 Linux 和 KVM 虚拟化平台上，可能限制了其在非 Linux 环境中的应用[38]。尽管如此，SPICE 协议作为一个专为虚拟化设计的云桌面协议，通过其对高性能图形和多媒体的支持，提供了一个强大的解决方案，用于满足虚拟桌面和云计算环境中的需求。随着虚拟化技术和云服务的不断发展，SPICE 及其未来的改进和扩展将继续在提供高质量云桌面体验方面发挥重要作用[39]。

ICA（Independent Computing Architecture）是由 Citrix Systems 开发的一种专有的云桌面协议。它主要用于 Citrix 的虚拟化产品，如 Citrix Virtual Apps and Desktops（以前称为 XenApp 和 XenDesktop），允许用户通过网络从任何地点访问中央服务器上托管的应用程序和桌面。ICA 协议的设计重点是提供高效的远程访问能力，即使在低带宽条件下也能保持良好的性能和用户体验。ICA 协议广泛应用于企业级的虚拟化解决方案中，特别是在需要远程访问应用程序和桌面环境的场景。这包括远程办公、分布式工

作团队、外包和业务流程外包（BPO）、远程教育以及医疗保健等领域[40]。ICA 的高效和灵活性使其成为构建大规模虚拟桌面基础设施（VDI）和应用程序虚拟化环境的理想选择。即使拥有强大的性能和舒适的使用体验，但 ICA 协议毕竟是一款商业协议，其高昂的部署成本相较于其它开源协议使其被选择的优先级大大降低。不过还是有很多企业愿意使用 ICA 协议作为他们的产品方案，ICA 在虚拟化技术和云桌面协议领域占据着重要的席位。ICA 协议不断在进行技术的探索和创新，以优化用户的体验，帮助企业创建一个优秀的解决方案，满足日益增长的云计算需求。HDX 技术就是在这一前提下诞生，其对高清图像、视频做出优化的同时还考虑倒了网络带宽，为云桌面用户提供了良好的用户体验[41]。

RFB 协议是 VNC 软件采用的无状态虚拟桌面协议，它基于帧缓冲区并易于实现跨平台兼容，但因直接操作像素数据而非操作系统特有命令，所以比其他协议如 PCoIP 更耗费带宽和计算资源。尽管支持多种图像编码，诸如 RAW、RRE、CoRRE，RFB 协议的压缩效率较低，可能导致网络资源的较大消耗。加之不支持音频传输和外部设备功能，它可能不太适合对数据传输效率和多媒体支持要求较高的办公场景[42]。

针对不同云桌面协议，已经有许多人对优化方案进行了研究，并取得了一些成果。文献[43]使用一种自适应方法来优化瘦客户端协议，如 RDP 协议、RFB 协议等。这种方法在上行方向通过缓存用户事件来减少数据包开销，在下行方向则采用自适应的基于调度的传输模式。研究目的是为了适应网络变化的特性，并减少瘦客户端移动设备的资源消耗。研究结果显示，这种优化策略能够在保持用户体验质量的同时减少带宽的使用。文献[25]研究了 SPICE 协议在桌面虚拟化技术中的应用，并提出了改进方案。文章中开发了 SPICEx 协议，通过优化协议的各个方面，以提高桌面虚拟化环境中的性能。文献[44]分析了 SPICE 协议，并在此基础上提出了一个透明桌面（Transparent Desktop）服务机制，通过对桌面虚拟化服务进行优化，以提供更加高效的虚拟桌面服务给用户。文献[45]讨论了如何设计并实现一个混合云显示协议，目的是在瘦客户端设备上优化多媒体体验。由于传统的云桌面协议，如 RFB，可能在多媒体传输效率上存在限制，文章提出了一个结合现有技术的新协议或改进的协议方案。文献[46]分析了 SPICE 协议在视频传输方面存在的性能瓶颈，然后提出了一系列优化方案，包括对丢帧情况和网络传输的改进，对原协议性能进行了提升，并探讨了进一步改进的可能方向。针对云桌面协议的优化研究已经有很多，但针对其图像编码算法的具体优化目前还几乎没有。本文将基于特定的云计算场景，对云桌面协议的图像压缩方法进行优化以适应低带宽环境，同时重点考虑到瘦客户端的 CPU 计算负担。通过这种优化，旨在提升云桌面服务的整体性能和用户体验，确保图像的有效传输同时最小化客户端使用过程中的资源消耗和带宽占用。

1.3 主要研究内容

云桌面传输协议是云计算基础架构的核心组成部分，它对用户体验和计算资源的高效分配有着直接的影响。本文将以云计算低带宽场景为背景，结合瘦客户端的 CPU 计算压力考虑，计划通过结合理论研究与技术应用，对云桌面协议中图像压缩方法进行深入分析。基于此，本文将进一步提出对云桌面协议图像压缩的优化策略，并将这些优化应用实施，目的是为了更好地满足云计算的发展需求，并推动云计算虚拟桌面服务以及整个云计算的进步。

本文主要对云桌面系统中的关键技术——云桌面协议的图像压缩方法进行研究，分析其在特定云桌面场景下需要优化的地方。同时从量化过程和渐进式压缩两个角度，对适应瘦客户端和低带宽环境下的云桌面协议中的 JPEG 算法进行优化改进，并通过实验验证改进的有效性。

本文研究内容主要包括以下方面：

（1）研究当前云计算当中，云桌面系统的背景和意义，分析和研究当前主流云桌面协议的优劣之处，同时分析了当前针对云桌面协议优化的国内外研究现状。针对当前研究的不足之处，以当前云桌面环境为背景，优化现有云桌面协议的图像压缩算法；

（2）针对 JPEG 算法中的量化过程，提出使用网格量化对其进行优化，在不损失图像质量的情况下提升图像压缩效率；

（3）针对 JPEG 算法中量化后系数的特征，使用一种新的渐进式分割方法，充分利用量化后 AC 系数零序列的特性，提升熵编码压缩效率；

(4) 根据以上的改进，将其运用到云桌面协议当中，优化其原有的图像压缩方法，并以多个指标对改进后的协议进行对比实验，验证改进的可行性。

1.4 论文结构安排

本文主要有五个章节，每个章节的具体安排如下：

第一章绪论。首先介绍了云计算中云桌面系统的研究背景和意义，然后分析了当前主流的云桌面协议的优劣之处，包括开源的和商用的，接下来介绍了针对云桌面协议优化的国内外研究现状，然后是本文的主要研究内容与论文结构安排的详细说明。

第二章主要描述了本文研究的相关理论和技术，分为三个部分：首先是对云桌面系统基本架构的介绍，紧接着对第一章提到的主流云桌面协议进行了详细介绍，最后对本文研究的重点——JPEG 图像压缩算法进行了介绍，深入分析了其原理和当前的不足之处。

第三章主要针对 JPEG 编码过程的量化过程进行改进优化。首先 3.1 对目前的 JPEG 算法进行了分析，提出了其中的不足与优化的方向。接着 3.2 对网格量化进行了介绍，包括其由来与运用的领域，并将这种方法运用到 JPEG 算法中量化系数的选择过程当

中。然后 3.3 详细介绍了本文网格量化的设计，包括量化器设计、变换系数的重构和量化系数的选择过程。最后 3.4 对加入了网格量化的 JPEG 算法进行了对比实验，介绍了实验的环境、评价标准、数据集选择，并对实验结果进行了展示和分析。

第四章针对 JPEG 编码过程中量化过后的 AC 系数的编码进行改进和优化。首先 4.1 介绍了渐进式 JPEG 的概念，同时介绍了几种常用的渐进式 JPEG 方法，包括光谱选择方法，逐次逼近方法和两种方法相结合的组合方法。然后 4.2 分析了 JPEG 压缩中，量化过后的 AC 系数经过 zigzag 扫描后序列化的数据的特点，采用一种新的渐进式分割方法，优化编码过程。紧接着 4.3 对这种分割方法进行详细介绍，包括分割的过程和分割后各序列的编码方法。最后 4.4 对这种方法进行了实验，并对结果进行展示和分析。

第五章根据前两章的成果，将两种方法组合后加入到云桌面协议的图像压缩过程当中，在 X2Go Kdrive 平台上进行实现以测试优化后的云桌面协议性能。首先 5.1 介绍了系统测试的环境，5.2 分析了实验的评价标准，包括 CPU 占用率、内存占用率和带宽占用。最后 5.3 对三种云桌面使用场景进行了对比实验，展示了实验的结果，并对其进行了分析。

第 2 章 相关技术研究

本章首先对云桌面系统基本架构进行了简单的介绍，其中包括云计算的发展阶段历程以及云桌面传输协议的诞生。然后介绍了主流的云桌面协议，包括开源的与商用的，最后详细介绍了这些云桌面协议中主要使用的图像压缩算法——JPEG 算法的相关知识和工作原理。

2.1 云桌面系统架构

云计算的发展经历了三个阶段，第一个阶段是底层系统资源的分配与管理，虚拟机就是诞生于这个阶段；第二个阶段是完成了一系列功能上的实现，如协议、环境、应用等，这一阶段的云无需高速传输应用支持，对性能以及传输效率上未可以做优化，故实时性要求不高；第三个阶段，也就是当前云计算发展的阶段，所面临的挑战就是在第二个阶段的基础上，对协议、应用等进行优化，提高系统性能以及传输效率，完成所有应用的云化，以及对高速传输应用的支撑，提升用户的体验。而云计算的终极目标就是，所有的东西都在云上解决，完成终端的轻量化，终端即客户机只需要解决两件事情，一是画面的显示，二是网络传输，也就是指令的发送以及图像的接收，对硬件性能要求不是那么高，完成客户端的“瘦”化[47]。

云计算与云桌面协议（如 Remote Desktop Protocol (RDP), Independent Computing Architecture (ICA), PC over IP (PCoIP), SPICE 等）之间的关系在于，云桌面协议是实现云计算中的桌面虚拟化技术的关键工具。桌面虚拟化涉及将用户的桌面环境托管在云中的服务器上，用户通过客户端设备连接到这个远端桌面以访问应用程序、数据和服务。这些协议管理云桌面的图像传输、输入输出重定向、安全认证、带宽优化、会话管理等方面，从而确保终端用户获得高效、流畅且安全的远程工作体验。因此，云计算为企业和个人提供了灵活的计算资源，而云桌面协议确保这些资源能够有效、安全地传输和使用，特别是在远程访问桌面和应用程序方面。

云桌面系统，其核心架构依然是虚拟桌面架构(VDI)，VDI 利用虚拟机来管理和提供虚拟桌面。虚拟桌面不是局限于特定的物理设备，而是操作系统的预配置映像，可以从任何兼容设备访问其应用程序。借助 VDI，桌面环境可以托管在集中式服务器上，并按需部署到最终用户，其架构如图 2-1 所示。

云桌面系统的核心在于将计算密集型的服务端程序部署在远程的数据中心或云服务器上，客户端只需负责界面展示和简单的输入处理。客户端软件设计轻量化，能够在配置较低的 PC 或移动设备上顺畅运行，这也符合云计算的目标。通常情况下，客户端可能仅仅是一个基于浏览器的应用或提供必要图形界面的程序。云桌面服务遵循了相同的理念，其服务端软件运行于服务器，而客户端通过云桌面客户端软件与之建立会话，实现对服务器操作系统的接入，从而使用云端计算资源[48]。云桌面协议的基本逻

辑是序列化用户的操作指令，并按照约定的数据格式在网络中传送至服务端，服务端处理这些指令后，将屏幕更新的数据以相同的数据格式返回给客户端显示。像 RDP 和 VNC 等流行的云桌面解决方案都运用了这种架构，使得客户端不必处理或维护复杂的应用程序。这使得云桌面通常是“无状态”的，不会保存任何用户特定的信息[49]。用户仅需在身份验证后保持会话信息，就可以随时回到上次云桌面会话的状态。云桌面的这种随时随地工作的特性，现在已经成为远程办公不可或缺的一部分。

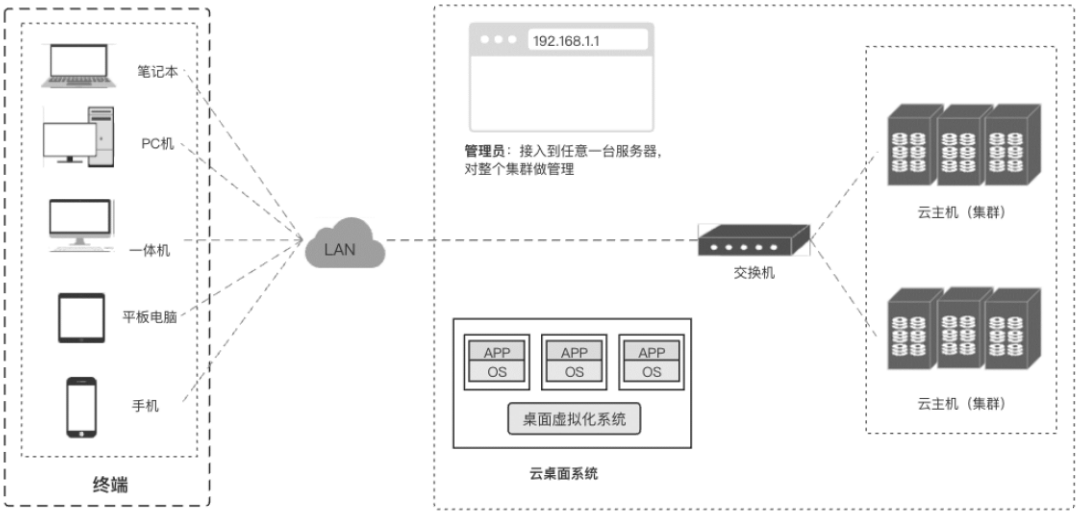


图 2-1 VDI 架构图

2.2 云桌面协议

2.2.1 RDP 协议

RDP（Remote Desktop Protocol）是由微软开发的一种网络通信协议，它是基于 T-120 标准系列协议的扩展。RDP 支持多通道技术，允许多个虚拟通道同时传输如演示数据、串行设备通讯、授权信息以及高度加密数据等，包括键盘和鼠标活动信息。RDP 扩展了 T.Share 协议的核心功能，保留了其他若干特性，如必要的多点支持架构功能，它允许应用在多会话中实时共享数据，如同虚拟白板共享，无需重复传输相同数据给每个参与者。RDP 设计时考虑了对不同网络拓扑的支持，如 ISDN 和 POTS，并且意在兼容多种 LAN 协议，包括 IPX、NetBIOS 和 TCP/IP。尽管如此，现行版本的 RDP 只采用 TCP/IP 协议。RDP 的数据传输活动符合现有 LAN 网络的 OSI 七层模型标准。在此模型中，应用程序或服务的数据会逐层通过协议栈下行传递，过程中进行分割和信道定向（通过 MCS），然后加密、打包、制帧，并最终封装入网络协议，随后寻址并通过网络发向客户端。数据在返回的途中经过相反的处理步骤：从数据包剥除地址，解包、解密，直至恢复原始的应用程序数据格式供应用使用。协议栈的主要修改发生在 OSI 模型的第四层（传输层）到第七层（应用层），其中数据经历了加密、打包、制帧、信道定向和优先排序等步骤的处理[50]。

RDP 栈实例中重要的组件包括多点通信服务多路复用器 (MCSMUX)、通用会议控制 (GCC)、Wdtshare.sys 和 Tdtcp.sys。

MCSmux 和 GCC 是国际电信联盟(ITU)T.120 系列的一部分，MCS 由以下两个标准组成：

1） T.122：定义多点服务

2） T.125：指定数据传输协议

在云桌面会话和虚拟化方案中，MCSMux 承担着关键的信息传输职能。它负责利用协议内定义的预设虚拟通道执行数据的多路复用、信道分配（确保数据流通过适当的通道）、优先级设置（赋予不同数据流不同的处理优先级），以及分割发送中的数据包。从协议的视角来看，MCSMux 的作用是将多个 RDP 栈从 GCC（群组通信与协作）的角度抽象成一个整体，使多个通信通道统一进行管理和调度。 GCC 是主要负责会话的管理，包括创建、删除会话连接以及控制多点控制服务（MCS）中的资源分配。相对应的每个终端服务器协议，如 RDP 或 Citrix 的相关 ICA，都包含有专门的协议栈实例，它们负责侦听并等待连接请求。 终端服务器设备驱动程序则协调和管理 RDP 活动中各部分的互动，这通常涉及对协议行为的整合运作。该设备驱动程序由一系列小型组件构成，其中包括负责图形界面(UI)传输、数据压缩、加密、组帧的 RDP 驱动程序（Wdtshare.sys），以及将 RDP 协议封装并传输至底层网络协议 TCP/IP 的传输驱动程序（Tdtcp.sys）。这些组件整合在一起，确保了云桌面通信过程的高效和安全运行。

2.2.2 ICA 协议

ICA（Independent Computing Architecture）协议是由 Citrix Systems 公司开发的专有协议，设计用于在客户端设备和服务器之间传送键盘、鼠标输入和显示输出信息。在云桌面服务和虚拟化领域，ICA 协议被广泛应用，允许用户通过一个轻量级的客户端程序连接到托管在数据中心或云环境中的虚拟桌面或应用程序。

ICA 协议是一种多通道协议，设计思想在于将不同类型的流量（如屏幕刷新、键盘输入、鼠标移动和打印作业）分隔在不同的通道中传输。这允许协议在保持高效的同时，为各种数据提供优化的传输。每个通道负责传送特定类型的信息，并且能够独立于其他通道进行数据加密和压缩。ICA 协议以它的灵活性和可扩展性而著称，它至关重要地允许跨越不同的网络连接和客户端硬件进行通信。协议数据单元（PDU）包括图形输出命令、客户端设备的输入、以及控制和管理信息。

ICA 客户端和服务器之间的通信遵循客户端-服务器模型。用户启动 ICA 客户端程序后，客户端程序会请求与服务器上的特定应用程序或桌面会话建立连接。服务器接收到请求后，在服务器上为用户会话创建一个虚拟化环境。用户的输入通过 ICA 协议发送到服务器，由服务器处理后，响应信息再通过 ICA 协议返回客户端。ICA 协议使用尽可能小的数据包来实现这种交互，以减少传输内容并优化性能。

ICA 协议具有以下特点：

1）轻量级传输：ICA 的设计目标是为了在有限的带宽下实现高效的数据传输。它通过智能压缩算法将图形界面的屏幕变更、鼠标动作和按键信息等压缩成极小的数据包。

2）会话可靠性：ICA 协议设计了会话保持和断线重连机制。当网络发生中断时，用户会话不会立刻关闭，一旦网络重新连接，用户就能恢复到中断前的状态，无需重新登录。

3）自适应性：ICA 协议能够根据网络条件自动优化性能。当带宽足够时，ICA 会提供更多视觉细节；带宽紧张时，则通过降低颜色深度和分辨率等手段来适应。

4）广泛的客户端支持：ICA 客户端兼容性极强，支持从传统的桌面操作系统到移动设备，乃至轻薄客户端等多种终端类型，使得用户无论在什么设备上都能访问到远程资源。

5）多会话支持：用户能够打开多个应用程序会话，每个会话像是在独立的窗口中运行，彼此之间不会相互干扰，这提高了工作效率和灵活性。

相较于微软的 RDP 协议，ICA 更加注重在不牺牲用户体验的前提下，通过使用数据重定向、优化技术来适应低带宽和高延迟的网络环境，这在云桌面虚拟化的体验上往往能提供更好的性能。ICA 协议通过优化数据流和压缩技术，能有效降低会话的延迟和带宽消耗。其高度灵活和适应性强的远程连接解决方案，在全球范围内赢得了业界的广泛认可，尤其在那些对于远程工作体验有高要求的行业中得到了重要应用。

2.2.3 RFB 协议

RFB（Remote FrameBuffer）协议是 Virtual Network Computing (VNC) 技术的核心组成部分，它定义了一种与平台无关的方式，用于远程显示和输入控制信息的交换。该协议的设计目标是简单性和性能。RFB 工作在应用层，并且它是基于 TCP/IP 协议栈的。它由 AT&T 的奥利弗斯通实验室在 20 世纪 90 年代后期发明。

RFB 协议是基于帧缓冲(framebuffer)更新和事件消息的协议。帧缓冲可以被看作是一个图像显示的数组，包括像素颜色值。RFB 的目标是实现远程终端和主机之间的实时图像更新，用户输入，如键盘和鼠标事件，然后在主机上进行相应的响应。RFB 协议以客户端-服务器模式工作。在这个模型中，VNC 服务器是控制实际帧缓冲的实体，用户通过 VNC 客户端连接到服务器。通信过程开始于客户端向服务器发起连接请求。随后，经过一系列的协商和认证过程，建立起一个可靠的会话，会话中的数据交换包括帧缓冲更新以及客户端的输入事件。典型的 RFB 会话分为几个阶段：

1）协议初始化（Protocol Initialization）

2）安全性协商（Security）

3）服务器初始化（Server Initialization）

4）客户端与服务器之间的正式消息交换（Normal Message Exchange）

RFB 协议具有如下几个特点：

1）跨平台：RFB 协议的一个主要优点是跨平台性。因为 RFB 协议依赖于对屏幕像素的操作，而不是特定的窗口系统或图形界面，它可以在几乎任何图形操作系统中实现。

2）无状态协议：RFB 本身是一个无状态协议，这意味着协议不会保留任何会话相关的信息。

3）桌面共享：RFB 不仅用于远程控制，也允许进行桌面共享。这是 VNC 系统中的一个突出特点，使得多个用户可以同时查看和操作同一台被控制计算机的桌面。

4）多种编码和压缩方式：RFB 协议支持多种编码格式来优化数据传输。例如，可以使用 Hextile 或 ZRLE 编码来减少传输的图形数据量，这在低带宽连接中特别有用。

与其他云桌面协议如 Citrix 的 ICA 或微软的 RDP 相比，RFB 可能在图形更新的效率和带宽消耗方面表现不尽人意，特别是在需要高分辨率和颜色深度的情况下。然而，其开放源代码的实现和对商业的非限制性使用促进了 RFB 广泛被接受、并在众多项目和产品中得到应用。RFB 协议作为 VNC 的核心，已经被运用在教育、企业 IT 支持、远程工作协作和虚拟化领域。它允许支持人员远程连接到问题计算机进行故障诊断和解决，也便于某些需要远程工作的用户获得公司服务器上的工作环境。此外，RFB 由于其免费和开源的特点，被广泛集成到各种第三方软件与硬件解决方案中。

2.2.4 SPICE 协议

SPICE（Simple Protocol for Independent Computing Environments）是一种开放的远程显示系统协议，旨在为虚拟化环境中的桌面提供高性能和高品质的图形和交互体验。SPICE 由 Qumranet 公司设计，后被 Red Hat 公司收购并开源。它让终端用户能够在任何地点通过客户端接入远程虚拟机，并提供近本地的体验。

SPICE 的设计之初就考虑到了性能和用户体验，使之成为企业级桌面虚拟化解决方案的理想选择。协议旨在处理视频流媒体、声音、互动式图形和密集的输入输出任务，支持高解析度视频、动画和各种交互式内容。它通过智能处理远程虚拟机的图形和声音输出，并优化这些数据的网络传输，确保用户在低延迟和高带宽效能下获得良好体验。SPICE 协议的工作过程涉及客户端和服务器之间的数据传输和处理。服务端负责运行虚拟机实例，通过 SPICE 服务器组件处理图形和音频数据，将其编码后传送给客户端。客户端使用相应的 SPICE 组件接收、解码并展示这些数据。

图 2-2 为 SPICE 协议基础架构，它展示了 SPICE 协议的几个重要组件及其之间的关系。主要有：

1）SPICE 协议：作为通信的规范，定义了数据传输与处理的标准和方法。

2）SPICE 客户端：是位于用户端的软件，负责接收服务端发送的数据流、处理数

据，并将用户的输入和操作返回给服务端。

3）SPICE 服务端：存在于管理虚拟机的宿主机中，通常以 libspice 库的形式被集成到虚拟机监控器中，例如 QEMU。服务端主要负责虚拟机的图像和声音输出，以及虚拟渲染的管理。

4）SPICE GUEST 组件：这是安装在虚拟机内部的一套软件，为了释放 SPICE 协议的全部功能。它通常包括了 QXL 驱动，用于改善和优化图像的渲染，以及 SPICE VDI Agent，用于支持剪贴板共享、文件拖拽等增强功能。

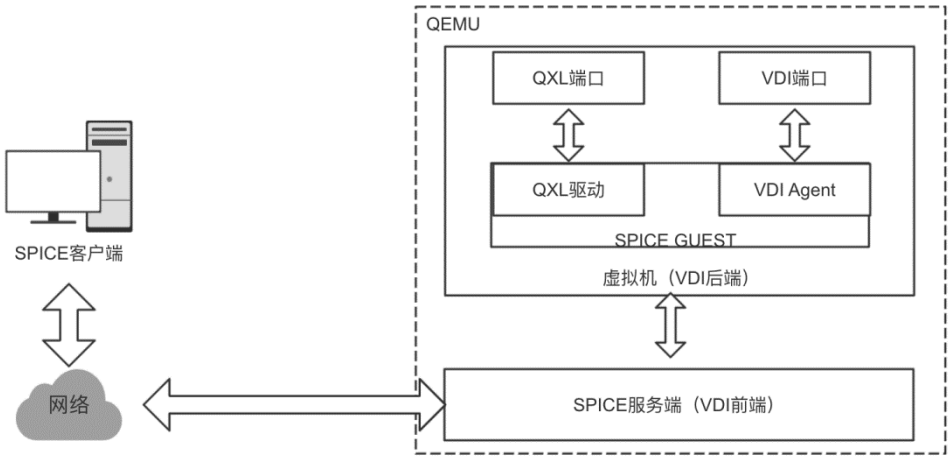


图 2-2 SPICE 基础架构图

SPICE 协议的工作过程可以分为以下几个主要阶段：

1）初始化阶段：客户端与 SPICE 服务器建立连接，并进行协议版本号对比、身份验证及会话配置。

2）图形处理：SPICE 服务端会捕获和压缩显示输出，利用 QXL（SPICE 协议专用的图形设备）或其他图形代理将图像传输至客户端。

3）音频处理：音频信号经过编码和压缩后发送至客户端，并保持与视频的同步。

4）输入设备处理：从客户端回传的输入事件（如鼠标移动、键盘敲击等）会被服务端捕获，并传递给虚拟机。

相较于其他远程桌面协议，例如 RDP 或 VNC，SPICE 在处理图形密集型应用时能提供更好的性能和用户体验。SPICE 是为虚拟化环境专门设计的，因而在虚拟桌面基础设施（VDI）场景中特别有优势。SPICE 的架构允许它更有效地处理虚拟机的图像和声音输出，并针对高延迟和有限带宽的网络优化传输。在虚拟化和云计算日益流行的今天，SPICE 协议展示了它在虚拟桌面交付领域的关键作用。作为一种协议，SPICE 为虚拟化技术提供了一个高效、高品质且可扩展的远程交互方式，无疑是构建未来虚拟化桌面基础设施的重要技术之一。

2.2.5 X2Go 协议

X2Go 是一个开源的云桌面协议，它提供了一个安全、快速且低带宽需求的远程访问解决方案[51]。X2Go 协议允许从客户端设备远程连接到运行 Linux 操作系统的主机。通过采用轻量级的图形界面传输手段和优秀的数据压缩技术，X2Go 能够在带宽受限的情况下为用户提供高质量的远程使用体验。X2Go 协议广泛应用于许多方面，如远程教育、虚拟工作站和服务器维护等。X2Go 协议致力于通过安全的 SSH 通道传输经过优化的 X11 会话数据。SSH（Secure Shell）是一个用于加密网络连接的协议，而 X11 是一个位图显示环境，常用于 UNIX 和 Linux 系统[52]。X2Go 客户端和服务器使用 SSH 加密所有通信数据，包括屏幕更新、鼠标和键盘输入、以及文件传输。

2.2.5.1 X2Go 协议的工作原理

X2Go 客户端是允许连接到远程服务器并在客户端计算机上显示图形桌面/应用程序的应用程序。X2Go 客户端需要本地 X11 服务器来显示远程会话。在微软的 Windows 上，这样的 X11 服务器随 X2Go 客户端一起提供，在 Linux 上，X2Go 的客户端部分使用本地 Xorg 服务器，在 Mac OS X 上，需要将 XQuartz X11 服务器[53]作为额外的组件安装。而运行 X2Go 服务器的计算机称为远程计算机。应用程序/会话在此远程计算机上启动，应用程序将其窗口/桌面传输到客户端。

X2Go 的运作基于客户端-服务器模型，并且密切整合了 SSH 和 NX 技术。SSH 提供了一个安全的通道来加密所有传输的数据，而 NX 是优化压缩 X11 协议的数据以提升性能的技术。在 X2Go 的体系中，服务器端负责运行 Linux 系统和虚拟会话，客户端则是用户与远程会话交互的入口。NX 是在 X11 的基础上优化而来的，原本的 X11 中，X Server 和 X Client 是通过 X11 协议直接通讯的，而 NX 技术中，X Server 和 X Client 不直接通讯了，而是通过 NxProxy 进行通讯。用户端的 X Server 误以为 NxProxy 就是 X Client，使用原生的 X11 于其通讯，而用户端的 NxProxy 接收到 X11 协议数据后，经过压缩，再通过 NX 协议与服务端的 NxProxy 通讯。在服务端，NxProxy 内嵌于 NxAgent 中，NxAgent 再将 NX 协议与 X11 进行转化，并且冒充 X Server 与 X Client 进行通讯，从而达到优化的目的。由此可以得到 X2Go 的基本架构图如下图 2-3 为 X2Go 的架构图。

X2Go 协议主要包括以下关键技术：

1）连接建立：客户端向服务器发起 SSH 连接请求，对话始于经过认证的安全通道的建立。

2）会话管理：服务器端会创建新的或恢复已存在的虚拟会话，并将 GUI 反馈至客户端。

3）数据传输优化：采用 NX 库对 X11 协议的数据进行压缩和优化，以减少网络传输时间和带宽消耗。

4）用户交互响应：用户通过客户端进行交互（如键盘输入、鼠标移动等），并实时地在远程会话中获得响应。

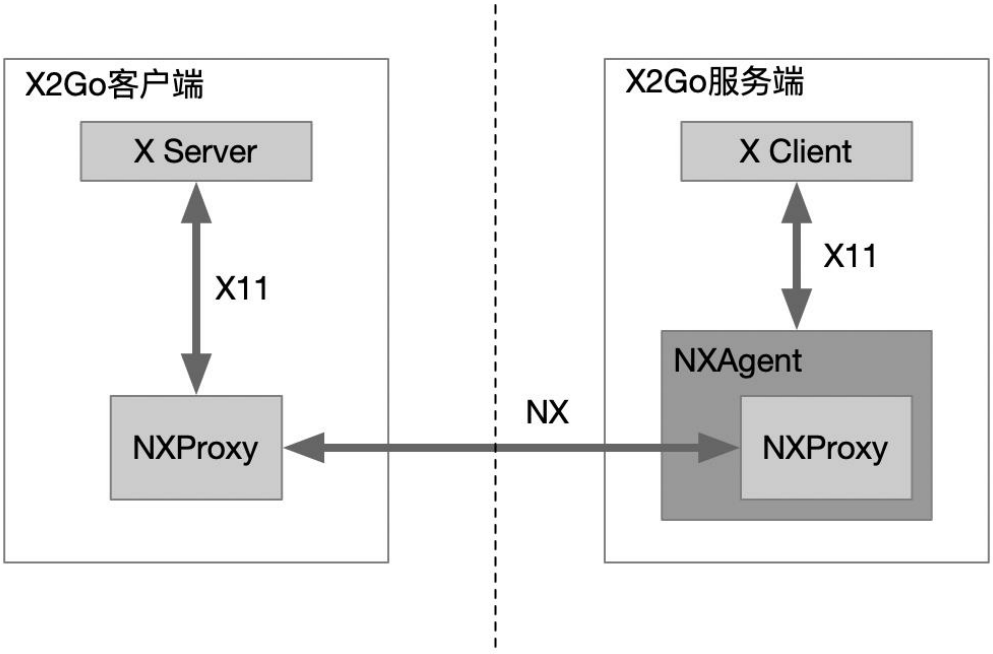


图 2-3 X2Go 基本架构图

2.2.5.2 X2Go KDrive 协议

X2Go KDrive 是用于 X2Go 的新 kdrive X 服务器。它可以代替 x2goagent。X2Go KDrive 的工作方式类似于 Xephyr，并且基于它的代码。X2Go KDrive 的目的是不受限制地运行现代桌面，例如 Unity、KDE-5、Gnome-3 以及使用最新工具包构建的应用程序。它与 x2goagent 的主要区别在于，x2goagent（或 nxagent）基于 NX 技术，它基本上是一种 X 代理（如 Xnest），它接受来自 X-Client 的 X 协议请求，并将其转发到远程 X-Server；相反，X2Go KDrive 是一个真正的 X-Server（就像 Xephyr 一样）。它真正处理来自 X-Client 的请求，并使用视频驱动程序在远程显示器上呈现图像，就像普通的 X-Server 将它们渲染到图形显示器中一样。可以说，x2goagent 和 X2Go KDrive 之间的区别与 Xnest 和 Xephyr 之间的区别相同。

X2Go KDrive 的最大优势在于在进行云桌面连接时，不需要在客户端配置 X 服务器，这显著简化了部署过程并提高了系统的适应性。因为若要在客户端上使用 NX 库，必须部署一个完全功能的 X 服务器，例如 Mac OS 需要 XQuartz，而 Windows 则需 VCXsrv。然而，这些 X 应用程序并非这些操作系统的本地组件，经常会引发各种兼容性和稳定性问题。而对于 Android 或 iOS 等操作系统，现阶段根本不存在合适的 X 服务器软件。使用 X2Go KDrive，就可以规避了上述问题，就能够为各种平台构建原生客户端，甚至可以开发 HTML5 客户端，这极大地扩展了云桌面技术的适用范围和方便性。X2Go KDrive 还有一个优点就是其架构更适合 Firefox 或 Chrome 等现代浏览器。

这些浏览器可以在不同的平台上运行，并且使用了跨平台图形库（如 Skia 等），现代浏览器和其他一些应用程序与 X2Go KDrive 的配合比使用 NX 要好得多。因此在本文中，使用 X2Go Kdrive 平台来进行最终优化后云桌面协议的性能实验。

2.3 JPEG 图像压缩算法

2.3.1 JPEG 算法介绍

在大部分主流云桌面协议中，主要使用到的图像压缩算法为无损压缩算法 PNG 和有损压缩算法 JPEG。PNG 是一种无损图像压缩算法，这意味着在压缩过程中它保留了图像的所有数据信息，在解压缩后可以完全复原到原始数据，由于 PNG 已经采用了一种高效的无损压缩算法（通常是 DEFLATE 算法），所以在不牺牲图像质量的前提下，无损压缩算法的优化空间并不大。本文的研究主要针对 JPEG 图像压缩算法进行。

JPEG 是一种广泛使用的图像压缩标准[54]，自 1992 年左右开始使用。它是 Web 上最流行的有损压缩图像格式，并且已经存在了很长时间，因为它可以在极大减小文件体积的同时，保持与原图相近的清晰度。JPEG 以其压缩算法的简单性和易于理解而著称，网络上几乎每张照片都以 JPEG 格式提供，不仅在互联网上，JPEG 格式也是智能手机和相机中用户存储和查看图片的主要格式。它是唯一一种实现了几乎通用兼容性的有损压缩图像格式，不仅与 Web 浏览器兼容，而且与所有可以显示图像的软件兼容。以视频压缩标准 H.264 为例的视频压缩算法，尽管应用于不同的媒体类型，它们在压缩方法上大多基于 JPEG 算法。JPEG 是一种高效的压缩算法，能大幅节省服务器存储空间，从而在互联网服务运营中减少经济成本。多年来，JPEG 一直在“进化”，不断有研究人员对 JPEG 进行优化，提升它的性能。那么 JPEG 算法在被提出 30 多年的今天是否以及释放了它全部的潜力了呢？答案是没有，JPEG 还有许多值得优化的地方，官方库 libjpeg 与 libjpeg-turbo 如今仍然在更新，同时也有许多针对特定环境的优化由于各种原因未被纳入 libjpeg 和 libjpeg-turbo，所以本文将基于特定的云计算场景，同时重点考虑到客户端的 CPU 计算负担，对云桌面协议的图像压缩方法进行优化以适应瘦客户端和低带宽环境。

在云桌面环境下，用户的体验主要与画面的延迟有关，虽然一些云桌面协议针对低带宽环境而设计，但由于 JPEG 算法的局限，它在公有云环境下的性能未能达到虚拟桌面架构中用户期望的灵活性和便捷性，特别是在处理一些复杂图形场景时表现欠佳。实验室测试表明，当前的主流云桌面协议在播放高清视频时仍需要极高的网络带宽，但当前公网环境往往难以提供如此高的带宽。鉴于此，本文提出了通过优化图像压缩算法来改善云桌面协议的策略，通过研究优化 JPEG 图像压缩算法来优化云桌面协议的使用体验，旨在减少所需的带宽消耗并为用户带来更流畅的使用体验。通过这一优化，可以显著提升在多变网络条件下的云桌面操作效能，更好地满足用户的远程接入

需求。

2.3.2 JPEG 压缩工作原理

人眼中的视杆细胞和视锥细胞两种光感受器对于我们感知世界是至关重要的。视杆细胞在低光环境下发挥作用，对色彩的感知较弱，而视锥细胞则对红绿蓝三原色非常敏感，负责我们对颜色的识别和细节的分辨。由于视杆细胞的数量远多于视锥细胞，人眼对亮度的感知远超对于色彩的敏感度。JPEG 压缩算法正是基于这一生物学原理设计的，它在压缩图像时会大量剔除人眼不容易察觉的细节和色彩变化（主要是图像中的高频信息），同时尽可能保留亮度信息，这样既减少了图片的文件大小，又能在视觉上保持较高的图像质量。如图 2-4 为 JPEG 压缩算法的大致步骤：

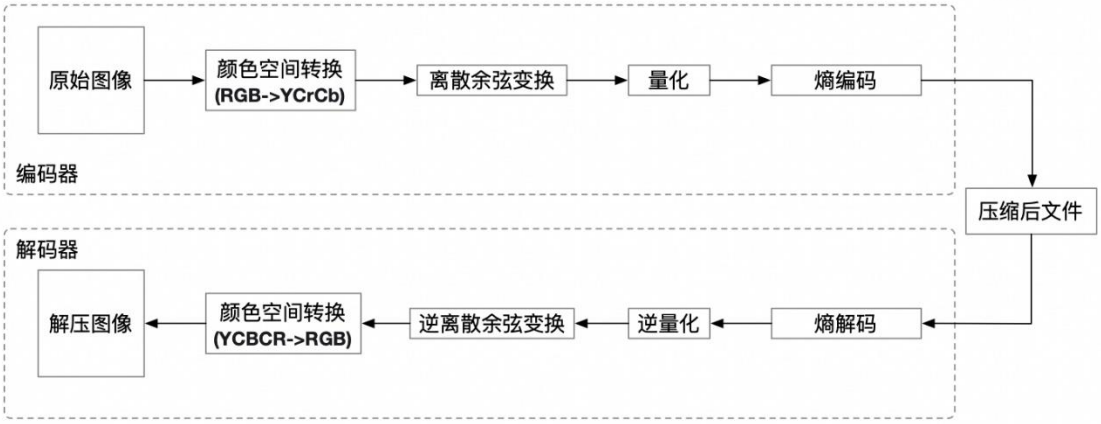


图 2-4 JPEG 压缩算法步骤

首先，图像应从 RGB 颜色空间转换到叫做 Y′CBCR（或 YCbCr）的不同颜色空间，它包括三个分量 Y'、CB 和 CR：Y'分量表示像素的亮度，CB 和 CR 分量表示色度（分为蓝色和红色成分），这种颜色空间表示与我们日常生活中使用的数字彩电和 DVD 所使用的颜色空间相同。Y′CBCR 颜色空间更有利于后面的步骤中在获得更好的压缩的同时不影响原本的图像质量（或者说在相同压缩水平的情况下获得更好的图像质量），简而言之就是可以进行更有效的压缩，因为可以将亮度信息单独提取出来，该信息对于人类视觉系统对图像的感知更为重要，颜色转换通过统计去相关也提高了压缩效率。转换为 Y′CBCR 颜色模型可以实现降低 Cb 和 Cr 分量的空间分辨率（称为“下采样”或“色度子采样”），通常对 JPEG 图像进行缩减采样的比率为 4：4：4（无缩减采样）、4：2：2（在水平方向上减少 2 倍）或（最常见的）4：2：0（在水平和垂直方向上减少 2 倍）。对于其他的压缩过程，Y'、Cb 和 Cr 以非常相似的方式单独处理。

子采样后，每个通道必须分成 8×8 个块。根据色度子采样，这将产生大小为 8×8（4：4：4 –无子采样）、16×8 （4：2：2） 或最常见的 16×16 （4：2：0） 的最小编码单元 （MCU） 块。在视频压缩中，MCU 被称为宏块。如果通道的数据不表示整数个块，则编码器必须用某种形式的虚拟数据填充不完整块的剩余区域。用固定颜色

（例如，黑色）填充边缘可能会沿着边框的可见部分产生振铃现象；重复边缘像素是一种常见的技术，可以减少（但不一定消除）此类伪影，也可以应用更复杂的边框填充技术。

做好色彩空间转换和采样后，将输入的图片分割成 8×8 像素的单元格，对每个单元格进行离散余弦变换（DCT）。计算 8×8 块的 DCT 之前，将其值从正值范围移动到以零为中心的值域。对于 8 位的图像，原始块中的每个条目都位于[0, 255]范围内。从每个条目中减去范围的中点，以生成以零为中心的数据范围，因此修改后的范围为[-128, 127]。此步骤降低了随后的 DCT 处理阶段的动态范围要求。公式(2-1)为 DCT 变换的公式，其中 k 的范围为 0 到 N-1。

$$X_k = \sum_{n=0}^{N-1} x_n \cos[\frac{\pi}{N}(n+\frac{1}{2})k] \tag{2-1}$$

人眼对于相对较大的区域内亮度的差异更为敏感，但不太擅长区分高频亮度变化的确切强度。这样的话减少高频分量中的信息量对于人眼感知不会有太大变化。这是通过简单地将频域中的每个分量除以该分量的常数，然后四舍五入到最接近的整数来完成的。在高精度下的 DCT 变换当中，该舍入操作是这一过程中除了色度子采样外唯一“有损”的步骤，许多高频分量被舍为 0，其余还有许多分量变成较小的数，它们通常可以用很少的位来表示。量化过程中，量化表是一个 8×8 的矩阵，它里面的数值从左上角逐渐增大到右下角。标准亮度量化表 Q_Y 和标准色度量化表 Q_C 如图 2-5 所示。

$$Q_Y = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \quad Q_C = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

图 2-5 原始 JPEG 标准中 50%质量的量化矩阵

其设计反映了人眼对图像中不同频率成分的敏感度——人眼对高频信息（细节部分）不如低频信息（大致轮廓）敏感。DCT（离散余弦变换）处理后的 8×8 基础图像块系数矩阵 G，当除以量化表 Q 的对应元素后，得到的结果矩阵 f[i]的右下角数值大致为 0，这代表高频信息被削减。将结果矩阵中的每个元素四舍五入到最近的整数后，就完成了量化，这相当于高频区域的数值被归零，实现了数据的压缩。

以上量化过程可以用公式(2-2)来描述，其中 i 和 j 的范围是 0 到 7，表示矩阵中的行和列。这个过程对亮度（亮度成分）进行，同样也应用于色度层（红色和蓝色色差层），但采用不同的量化表，这样进一步地去除了大量人眼难以察觉的色彩信息，加深了压缩效果。量化表的元素值大小直接影响压缩比率，数值越大，相应的压缩率也越

高，但也更可能导致压缩失真。

$$B_{j,k} = round(\frac{G_{j,k}}{Q_{j,k}}) \tag{2-2}$$

最后是熵编码，这是一种特殊形式的无损数据压缩，它涉及采用将相似频率组合在一起的运行长度编码（RLE）算法以“之字形”顺序排列图像分量，该算法将相似的频率组合在一起，插入长度编码零，然后在剩余部分使用霍夫曼编码。

JPEG 图像压缩算法虽广泛应用，却伴随一些不容忽视的弊端，特别是在高压缩比下，图像质量的明显下降。由于算法在压缩过程中大量丢弃了高频信息，留下的主要是低频细节，这使得复原的图像无法恢复细微的纹理和边缘的锐利度，导致可见的失真和模糊，尤其在强烈的颜色边界处形成了众所周知的块状伪影效应。此外，JPEG 并不适合压缩需要高度细节保留的矢量图形，因为它在图形放大时会出现边缘锯齿化，损害图形的整体视觉效果。这些局限性使得将 JPEG 压缩算法作为核心图像压缩方法的云桌面在图像方面有了较大改进的空间，存在更好或者更适合特定场景的方法来避免 JPEG 算法在云桌面场景下的压缩缺点。

2.4 本章小结

本章首先对云桌面系统架构的相关知识进行了介绍，并表明了云桌面协议在其中的作用以及重要性，引出了本文研究的重点。然后介绍了几种市场占有率较高的云桌面协议，其中有商用的 RDP 协议和 ICA 协议，也有开源的 RFB 协议、SPICE 协议和 X2Go 协议，包括它们的基础架构以及工作流程，讨论了它们在云桌面场景中的应用，包括各自的优势、局限以及实际应用中的考量因素。接下来介绍了主流云桌面协议中运用最多的有损图像压缩算法——JPEG 算法，同时也是当前网络中使用最多的图像压缩格式之一，对其原理和压缩过程进行了概述。本章提供了对云桌面系统各个方面的深入理解，并通过探讨架构设计、传输协议以及压缩技术的关系和作用，为实现高效、稳定的云桌面环境奠定了理论基础。这为后续的性能优化和实验设计，提供了坚实的理论支持和技术指导。

第 3 章 网格量化优化 JPEG 图像压缩

JPEG 压缩过程可大体分为颜色空间转换、DCT 变换、量化和熵编码过程。其中，量化过程是 JPEG 算法“有损”的重要步骤，量化步骤在 DCT 之后，每个系数会被量化表中相应的值直接除以。然而，这种直接的方法可能没有充分优化。本章将介绍一种网格量化方法，并用它来优化云桌面传输协议中的 JPEG 图像压缩过程，并对改进后的 JPEG 算法进行对比实验，对改进后的算法性能进行评估。实验结果表明，改进后的 JPEG 算法相较于改进前的方法取得了压缩率上的提升。

3.1 JPEG 优化分析

自 1992 年由联合图像专家组（JPEG）提出以来，JPEG 图像压缩标准已成为最广泛使用的图像标准之一。JPEG 的基本原理是基于离散余弦变换（DCT）的应用，它可以将图像数据转换到频域。通过利用人类视觉系统对低频信号更敏感的事实，JPEG 采用了有损压缩技术，有选择性地消除与高频分量相关的信息。量化表是 JPEG 中的一个 8×8 矩阵，在确定压缩和视觉保真度之间的折衷时发挥着关键作用。

尽管存在一些广为人知的限制，JPEG 仍被普遍认可并当作一个行业标准，采用新格式来替换如此根深蒂固的 JPEG 标准无疑是一个挑战。Taubman 和 Marcellin 在 2002 年提出了 JPEG 的显著改进版本——JPEG2000 [55]。随后的研究[56]显示，JPEG2000 在几乎所有方面都相较于 JPEG 有显著的提升。然而，尽管性能更优越，JPEG2000 并未获得广泛采用，行业内也尚未得到广泛认可。即使 JPEG2000 展现了作为 JPEG 的优秀替代品的潜力，其有限的应用也反映了替换一个广泛确立的行业标准所面临的挑战。由此看来，图像压缩标准的未来发展不仅需要技术上的革新，还需要考虑到兼容性、计算可行性及用户接受度。

量化步骤是 JPEG 中至关重要的一个环节，其中的量化表（8×8 的矩阵）在控制压缩比与图像质量之间的关键权衡中扮演重要角色。通常在 JPEG 编码的流程中，量化步骤在 DCT 之后立即发生，每个系数会被量化表中相应的值直接除以。然而，这种直接的方法可能没有充分优化，而在 DCT 和量化步骤之间精心调整这些系数可以进一步提升图像的清晰度和压缩效率。本章将在维持现有 JPEG 实现的兼容性的同时，通过优化量化系数，增强 JPEG 压缩和提高图像清晰度。

3.2 网格量化介绍

在图像编码的领域中，一个常见而有效的方法是采用基于块的混合编码策略。这种方法首先将图像分割为一系列的矩形块，每个块内的像素都是局部相连的，这意味着块内的像素通常具有高度的相关性。变换编码步骤包括应用一个正交（或近似正交）变换，比如离散余弦变换 (DCT)，这帮助将图像数据从空间域转换到频率域。变换的目

的是减小输入块样本间的统计关联，从而更有效地编码图像信息。这通常会导致转换后的大部分能量集中在较少的变换系数上，使得一些系数比其他系数包含更多信号能量，最终的效果是提高了后续标量量化的编码效率[57]。此类系数通常对应于图像中的低频信息，而那些近似为零的系数则对应于高频信息，后者往往不那么重要。量化是接下来的关键步骤，它采用标量量化方法将变换系数转换成量化系数。量化的主要任务是减少所需的比特数，同时尽可能保持图像质量，它基于人类视觉系统对图像的不同频率部分的不同敏感度，选择性地削减变换系数。熵编码是编码过程中的最后一步，目的是将量化系数以最高效的方式编码，这通常意味着使用尽可能少的比特。熵编码技术，如哈夫曼编码和算术编码，会利用变换块内部的量化系数之间的剩余统计依赖关系，即使是量化之后也存在一定的预测性和模式性，以进一步减少最终编码的比特流的长度。熵编码的目标是用尽可能少的比特来表示量化指标。

现代视频编码标准，如 H.264/AVC 和 H.265/HEVC[58]，采用的是均匀重构量化器（Uniform Reconstruction Quantizer，简称 URQ）进行标量量化。在 URQ 中，重构的值由单个参数量化步长 Δ 所确定。量化后的系数 q 与重构后的变换系数 t'的关系由一个简单的乘法关系定义，即 $t' = q \cdot \Delta$。当处理变换系数的典型分布时，这种均匀重构量化器能够提供与最优（考虑熵约束的）标量量化器相近的率失真性能[59, 60]，同样，图像压缩也可以借鉴这种方法，使用 URQ 以实现更优的率失真性能。

然而，变换编码设计的实际编码效率并不仅仅被其结构设计所影响，它在很大程度上还取决于在编码器里如何选取量化系数的算法。为了选择一组量化系数来编码一个块，现代视频编码器通常利用优化算法，通过最小化由失真 D 和编码的比特数 R 所构成的拉格朗日函数 $D + \lambda R$ 来选择量化系数以达到最优压缩效果[61-63]。这里的 λ 是一个用于调节失真与比特数之间权衡的拉格朗日乘数。这类优化方法，将熵编码过程中的量化系数依赖关系考虑在内，通常称为软判决量化或率失真优化量化（Rate-Distortion Optimized Quantization，简称 RDOQ）。在 H.265/HEVC[58] 和新的标准化多功能视频编码 Versatile Video Coding（VVC）[64]等先进视频编码标准中，RDOQ 算法是编码过程中的一个重要组成部分。通过 RDOQ 算法，可以确保每个块的量化系数都是在考虑编码效率和视觉质量损失的基础上选择出来的，优化了编码效果。这种算法的应用表明，在图像数据压缩中寻求编码率和失真的最佳平衡是当前图像编码技术发展的关键点之一。

变换编码结构的特定约束，即使用正交变换配合标量量化，不可避免地会导致编码效率损失。这是因为在变换编码中，一个具有 N 个样本的信号块集合在 N 维空间[65]内构成了一个正交网络。通过引入某种形式的矢量量化，可以减少这种效率的损失。无约束的矢量量化，虽然在图像编码方面理论效率极高，但由于其计算复杂性，难以在实践中被广泛应用。为了解决这一挑战，有人开发了一些简化版本的矢量量化方法，这些方

法在保持编码效率的同时，确保了编码器和解码器的复杂性在一个合理的范围内。例如，H.265/HEVC 标准中就包括了一种称为符号数据隐藏[66, 67]的方法，它实现了一种非常基础的形式的矢量量化，同时保持编码的简单性和效率。还有一些方法被用来解决这个问题，其中一个就是网格编码量化（Trellis Coded Quantization，TCQ）[68-70]，本文简称网格量化或 TCQ，TCQ 将 N 维重构向量表示为 N 维整数向量，其组成部分保留了传统标量量化中量化系数的类似统计性质。因此，大部分熵编码无需进行太大的修改即可使用。

网格量化早在 1990 年的文章“Trellis coded quantization of memoryless and gauss-markov sources”中就有研究，TCQ 是一种精妙的量化方法，结合了标量量化的简单性与完全矢量量化的性能优势。TCQ 在效率和解码器复杂性之间寻找了一种平衡，这使得它能够在实用性和编码效率方面取得折中。作为一种受限的多维矢量量化器，TCQ 对每个分量的量化进行了限制，这种限制在一定程度上弥补了纯标量量化所无法避免的性能损失。

在多功能视频编码（Versatile Video Coding，VVC）中，TCQ 的实现基于标准的标量量化，但引入了两个量化器（Q_0、Q_1）以及四个过渡状态。具体来说，它将块中的一系列量化候选值排列在一种称为“网格图”的结构中。在该网格图上，最优的量化结果对应于总成本最低的路径，此成本反映了失真和压缩率（R-D 成本）。TCQ 利用了变换系数之间的相关性，这样重构向量、扩增量化器，以及候选向量间的联系就更加紧凑，从而提高了率失真（R-D）性能。在 H.264/AVC 中，TCQ 技术也被用来改进量化过程，这有助于在相同的质量标准下，以更低的数据率输出视频，从而提升了视频压缩的效率。通过在编码过程中采用基于速率-失真优化的方法，TCQ 能够降低编码视频的失真，同时减少所需的比特率，相较于传统的标量量化，这提供了一种在保证视频质量前提下减少数据传输量的有效编码策略。

从解码器的角度看，TCQ 涉及两个标量量化器和它们之间的切换过程。在典型的 TCQ 设计中，两个标量量化器可用的重构值代表量化步长 Δ 的整数倍。一个量化器包含 Δ 的偶数倍，而另一个量化器包含 Δ 的奇数倍。在这两个标量量化器之间的切换可以由一个具有 2^K 状态的状态机（其中 $K\geq2$）来表示，每个状态对应于一个特定的量化器。当前的状态，从而用于给定样本的量化器，由前一个状态和前一个量化系数的值唯一确定。在编码过程中，两个标量量化器之间潜在的转换可以通过每个样本有 2^K 状态的网格图来表示。找到最佳的量化系数序列相当于识别具有最小率失真成本的网格图路径。当一个量化系数的熵编码仅依赖于相关的状态，而不依赖于前序量化系数的实际值时，可以通过 Viterbi[71]算法找到最优解。

在这种方法中，TCQ 代替了标准变换编码中通常使用的标量量化环节。TCQ 的核心目标是为了借鉴矢量量化的优点，尤其是在空间填充方面。矢量量化之所以效果显

著，是因为它可以更有效地覆盖信号空间，提供更高的信号再现精度，并且通常具有更低的比特率，这是由于它可以利用信号样本之间的相关性，但随之带来的是更为复杂的计算和编码难度。TCQ 在变换编码流程中的作用是通过实现更紧密的空间填充以及更精确的信号表示使量化步骤更为高效。它通过将变换后的系数（即预测误差样本的结果）视为一个整体并作为矢量来处理，不仅单独考虑每个系数，还考虑了它们之间的相互关系。这种方法有效减少了由于标量量化的量化噪声导致的失真，从而实现空间填充的优势。与传统变换编码相似，TCQ 在变换域中应用量化步骤，并紧接着使用适当的熵编码技术来对量化后的系数进行编码。这一步骤利用了预测误差样本之间的依赖关系——也就是说，尽管样本是量化的，但它们之间的统计相关性并未被完全消除。通过熵编码的策略，如算术编码或哈夫曼编码等，可以进一步压缩数据，以最有效的方式表示变换和量化后的信号，进一步减少编码所需的比特数。总的来说，TCQ 通过结合矢量量化的空间填充效率和变换编码的依赖性利用，提供了一种改进的量化方式。这种方式旨在保持或增强信号的再现精度，同时减少所需的编码比特率，它在追求优化编码效率和图像输出质量的场景下起到了关键的作用。

在本文工作中，通过将上述的 TCQ 方法应用到 JPEG 图像压缩当中，来改进 JPEG 压缩算法，提升压缩效率，从而减少云桌面传输协议在传送图像中所需的带宽，优化用户在低带宽环境下的体验。

3.3 网格量化设计

3.3.1 量化器设计

使用的两个标量量化器 Q_0 和 Q_1 如图 3-1 所示。它表示使用 Q_0 和 Q_1 两个量化器编码的量化系数所对应的重构值（横坐标），重构值是量化步长 Δ 的整数倍。Q_0 的重构值由是量化步长 Δ 的偶数倍，Q_1 的重构值是量化步长 Δ 的奇数倍，另外还有零值。文章[72]的研究结果表明，如果两个量化器中都包括零值，TCQ 的低比特率性能通常可以得到提升。

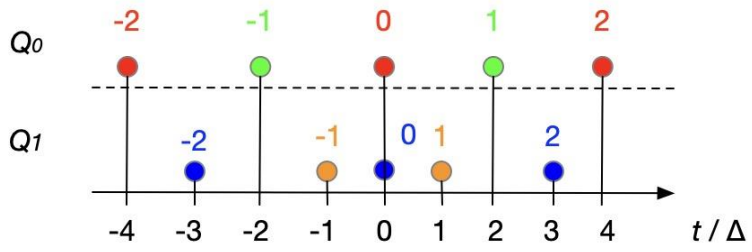
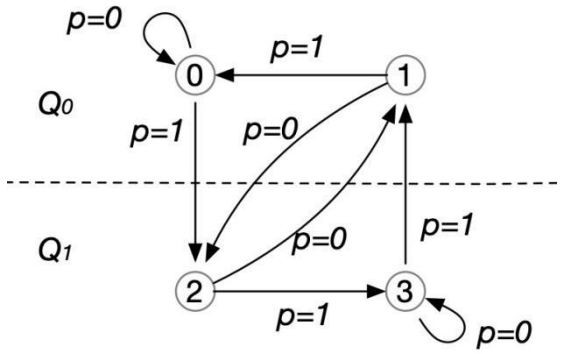


图 3-1 标量量化器 Q_0 和 Q_1

同时 Q_0 和 Q_1 都选用了对称的量化器，实验结果表明这样的设计提供了更高的编码效率。对于两个量化器，重构值与量化系数一一对应，量化系数由 q 表示，量化系数为 0 表明对应重构值也为 0，如图 3-1 所示。

在图像编码中，考虑到变换块中通常只有少数系数是非零的，因此选用了简单的 4 个状态 TCQ 设计。这种设计既顺应了非零系数的稀疏性，又有助于最大程度地减少编码器的复杂性。状态转换的量化器选择过程如图 3-2 所示，一个块的变换系数按照预先定义的顺序被逐一重构，这个顺序可以通过索引 k = 0, 1, ... 来表示。编码顺序的起始状态 s_0（对应于编码顺序中的第一个变换系数 t_0）被初始化为 0。随后，根据当前的状态 s_k 和当前的量化系数 q_k 来确定下一个状态 s_{k+1}，具体的状态转换取决于当前量化系数 q_k 的奇偶性 p_k。如果当前状态 s_k 为 0 或 1，那么变换系数 t_k 将使用 Q_0 量化器；如果当前状态 s_k 为 2 或 3，则使用 Q_1 量化器。

这样的状态转换过程将两个量化器的量化系数分成了两个子集，一个为奇数，一个为偶数，选择的子集，结合当前的状态，就能决定接下来的变换系数将会使用哪个量化器。在图 3-1 中，两个不同的子集被虚线隔开。



a) 状态机

state s_k	quantizer	next state s_{k+1}	
		$p_k = 0$	$p_k = 1$
0	Q_0	0	2
1	Q_0	2	0
2	Q_1	1	3
3	Q_1	3	1

b) 转换表格

图 3-2 用于量化器选择的状态转换过程

3.3.2 变换系数的重构

与标量量化不同的是，一个块变化系数的重构必须按照顺序进行。由于量化过后熵编码过程，重构顺序需要与量化索引的编码顺序相同。给定一个块的 N 个量化系数 q_k，k 表示编码顺序，相关的重构变换系数 t_k 可以通过以下简单算法获得：

算法 3-1 变换系数算法

输入：量化系数个数 N；量化系数 q_k

输出：重构变换系数 t'_k

1. $s_0 = 0$
2. **for** $k = 0$ *to* $N - 1$ **do**
3. $t'_k = (2 \cdot q_k - (s_k \gg 1)) \cdot \mathrm{sgn}(q_k) \cdot \Delta$
4. $s_{k+1} = \mathrm{stateTransTable}[\, s_k \,][\, q_k \,\&\, 1]$
5. **end for**

sgn()表示符号函数，Δ 表示量化步长，二维数组 stateTransTable 表示图 2 所示的状

态转换表，运算符“ >>”表示二进制向右移动一位，$s_k >> 1$ 即表示所选的量化器，运算符“ & ”应表示二进制“与”，$q_k \& 1$ 指定量化系数 q_k 的奇偶校验结果 p_k。

3.3.3 量化系数的选择

3.3.3.1 系数候选值

选择合适的量化系数是编码过程中一个关键的决策点，因为它直接影响着重构图像的质量与编码的压缩效率。不能随意选择量化系数，这个选择必须和整个编码过程的目标相一致，即在最大限度减小失真的情况下实现高效编码，也就是用最少的比特数表示最清晰的图像内容。

由于编码的目标是最大化压缩效率，量化系数的选择应当基于比特数。所以采用一种查看编码每个可能的量化系数所需的比特数量的方法。对于每个非零的量化系数 q，在编码位数范围内从 1 位开始进行遍历，直到达到 q 本身的二进制位数。换句话说，应该考虑所有可能比 q 表示比特数要少的情况。

对于那些编码位数小于 q 所需位数的候选量化系数，应当选择当前编码位数可能的最大量化系数，这是因为选择最大的二进制值可以使得当前编码位数的候选值与原本的量化系数最接近，即使得失真 D 最小化。按照这种方式，可选择的候选序列将包括诸如 1，3，7 等（都是 2 的 n 次幂减 1 的数值），因为这些数值是在给定位数下能够表示的最大值。而对于编码位数恰好等于 q 所需位数的情况，最佳选择就是 q 本身，因为这种情况下失真 D 最小，即当前编码位最小的失真。简而言之，合适的量化系数候选值集合应由最大化编码效率和最小化失真决定，考虑到的是在特定比特数约束下能够使用的最大值，以及原始量化系数 q 本身。通过这种方式，编码器确保了以最高效的数据率传输尽可能高质量的图像内容。

根据上述分析，对于一个给定的非零量化系数 q，从最小的编码位数开始，也就是 1 位，对该位数下可表示的最大值进行编码，这相当于二进制下的“1”。随后，增加一位，继续采用该位数下可表示的最大值，等于二进制的“11”（十进制下的 3），然后是“111”（十进制下的 7），以此类推，一直到达到能够与量化系数 q 具有相同编码位数的值。在这一过程中，实际上是在考虑每个可能的比特数，并与之关联的最可能的最大量化系数，这是为了在特定的比特预算下最小化失真。当达到与量化系数 q 具有相同位数的情况时，候选值就定为 q 本身，因为此时失真为最小，即候选值与原始量化系数完全一致。图 3-3 清晰地演示了这个量化系数候选值集合生成的过程。这个图展示了二进制数值如何随着比特位数的增加而逐步增长，并且在每一个级别上，取每个可能编码长度下的最大值，直至到达与量化系数 q 有相同编码位数的情况。通过在不同的比特层级上选择这样的值，得到了一组有效的候选量化系数集合，既能够保证信息的准确传递，又能够最小化编码的总体比特数，符合高效视频编码的总体目标。

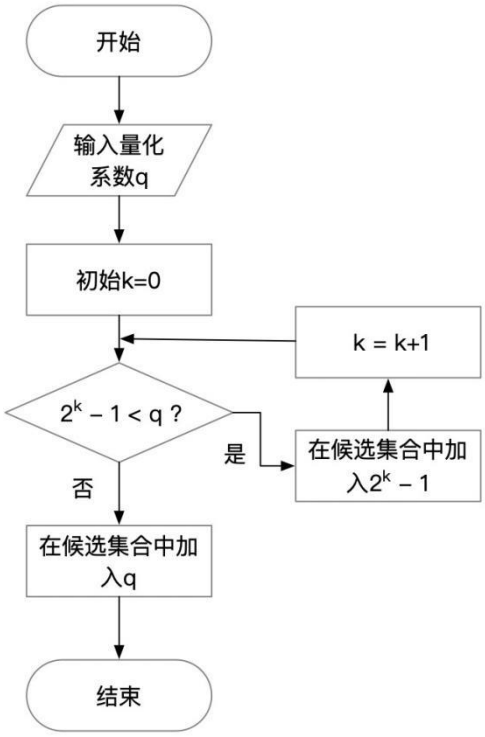


图 3-3 量化系数候选集确定

3.3.3.2 选择过程

为了在保证图像质量的情况下最大化编码效率，选用拉格朗日函数 $J = D + \lambda R$ 来评估编码的效果，编码器需要仔细挑选量化系数 q_k 来最小化 J，同时考虑到失真 D 和传输量化系数所需的比特数 R。拉格朗日乘数 λ 是由量化步长 Δ 决定的，且失真选用均方差失真（Mean Squared Error，MSE）。由于所使用的变换通常是正交的，给定块的总失真 D 可以表示为每个变换系数 t_k 的单独失真 D_k 的总和，由公式(3-1)给出。

$$D_k(q_k, s_k) = (t_k - \Delta \cdot (2 \cdot q_k - (s_k >> 1) \cdot sgn(q_k)))^2 \tag{3-1}$$

其中，t_k 表示原本的变换系数，q_k 表示量化系数，s_k 表示当前的 TCQ 状态，t_k 后面的部分表示重构后的变换系数 t'_k，所以这个式子代表的是原本的变换系数 t_k 与重构后的变换系数 t'_k 之间的差的平方。

正如前文所述，TCQ 中存在的依赖性可以通过网格图形象的表示。为了更好地考虑实际的熵编码，使用了一个 5 状态格子图而不是传统的 4 状态格子图，如图 3-3 所示。这个格子图除了包含状态 0-3 外，还增加了一个“未编码”的状态。“未编码”的状态代表在编码顺序中出现在第一个非零量化系数之前的等于 0 的量化系数。格子图的各个阶段按编码顺序处理，由索引 k 表示。一个量化系数的码率取决于前面量化系数的实际值，尽管由于熵编码中的复杂依赖性，Viterbi 算法 [51]并不能得出最优解。然而即使如此，由于 Viterbi 算法在编码效率和实现复杂性之间提供了良好的平衡，因此仍然采用了 Viterbi 算法。

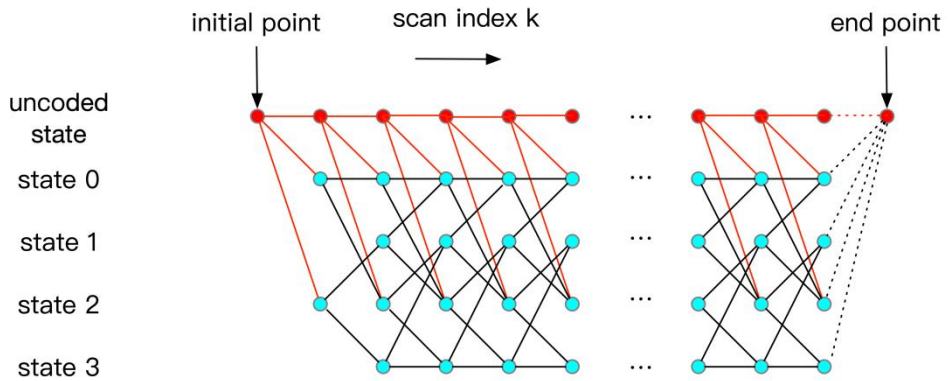


图 3-4 TCQ 中网格结构

起始状态的率失真成本设为 $J_{-1}=0$。然后，按编码顺序处理网格图，编码顺序由索引 k=0, ⋯ , N−1 表示，其中 N 是块中变换系数的数量。对于从扫描索引 k−1 到扫描索引 k 的每个转换，阶段 k−1 和 k 的网格节点之间的所有连接都与率失真成本 J_k 相关联。其转换关系由公式(3-2)给出。

$$\begin{aligned} &uncoded \to uncoded: \ J_k = J_{k-1} + D_k(0,0) \\ &uncoded \to (0,2): \quad J_k = J_{k-1} + D_k(q_k, s_k) + \lambda \cdot (R_{first}(x_k, y_k) + R_k(q_k)) \\ &(0,1,2,3) \to (0,1,2,3): J_k = J_{k-1} + D_k(q_k, s_k) + \lambda \cdot R_k(q_k) \end{aligned} \tag{3-2}$$

其中，J_k 代表节点的率失真成本。$R_k(q_k)$代表传输一个量化系数 q_k 所需的预估比特数。$R_{first}(x_k, y_k)$代表将当前扫描索引 k 的位置(x_k, y_k)作为第一个非零量化系数的位置传输所需要的预估比特数。所以从“uncoded”状态到状态 s_k = 0, 2 的转换只可能对于量化系数 $q_k \neq 0$。对于每个连接，只需要保留最小化相关失真 $D_k(q_k, s_k)$的量化系数 q_k。除了图 3-4 中显示的网格连接，还应检查对应编码子块标志等于 0 的子块的连接，这些连接仅存在于当前扫描 k 对应于子块内最后一个扫描索引时。

在评估了所有到达阶段 k 对应节点的连接后，对于当前阶段的每个节点，只保留具有最小率失真成本 J_k 的连接，并将关联的成本 J_k 赋给目标节点。这个过程持续进行，直到 k = N - 1，这时通过网格图可以得到 5 条最优的路径。最终，选择具有最小成本 J_{N-1} 的路径，组成该路径的量化系数序列 $q_0, q_1, ..., q_{N-1}$ 即是最终选择的具有最优率失真性能的路径。

3.4 实验与分析

3.4.1 实验测试环境

本文通过实验室服务器上的虚拟机进行测试，实验的测试环境如表 3-1 所示。

表 3-1 测试环境系统配置

配置项	参数
虚拟化服务器 CPU	Intel(R) Xeon(R) E5-2678 v3 2.50GHz
虚拟机操作系统	Debian12.0
虚拟机 CPU	vCPU
虚拟机硬盘	64G
虚拟机内存	8G

3.4.2 实验评价标准

（1）压缩率（Compression Ratio）

压缩率是计算机科学与信息论中用来评断资料压缩算法好坏的指标之一，通常可借由压缩率得知资料被压缩的程度，进而判断压缩算法的优劣，同等情况下压缩率越小，代表压缩过后的文件大小越小，即压缩算法的性能越好。

压缩率的定义如下：

$$Compression\ Ratio = \frac{B_1}{B_0} \times 100\% \tag{3-3}$$

B0 为资料压缩前的位元数（资料量大小），B1 为资料压缩后的位元数。

（2）峰值信噪比（Peak signal-to-noise ratio，PSNR）

峰值信噪比是一个表示讯号最大可能功率和影响它的表示精度的破坏性噪声功率的比值的工程术语。由于许多讯号都有非常宽的动态范围，峰值信噪比常用对数分贝单位来表示。

它常简单地通过均方误差（MSE）进行定义。两个 m×n 单色图像 I 和 K，I 为一无噪声的原始图像，K 为 I 的噪声近似（例：I 为未压缩的原始图像，K 为 I 经过压缩后的图像），那么它们的均方误差定义为：

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{3-4}$$

峰值信噪比定义为：

$$PSNR = 10 \cdot log_{10}(\frac{MAX_I^2}{MSE}) = 20 \cdot log_{10}(\frac{MAX_I}{\sqrt{MSE}}) \tag{3-5}$$

其中，MAX_I 是表示图像点颜色的最大数值，如果每个采样点用 8 位表示（例如图像处理），那么就是 255。更为通用的表示是，如果每个采样点用 B 位线性脉冲编码调制表示，那么 MAX_I 就是 2^B-1。

对于每点有 RGB 三个值的彩色图像来说峰值信噪比的定义类似，除了横轴、纵轴 m 和 n 以外，还要考虑它的颜色组成 RGB，需各别对每个颜色处理其 MSE，因为有 3 个颜色通道，所以 MSE 需再除以 3。

彩色图像的峰值信噪比定义为：

$$PSNR = 10 \cdot log_{10}\left(\frac{MAX_I^2}{\frac{1}{3mn}\sum_{R,G,B}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I_{color}(i,j) - K_{color}(i,j)]^2}\right) \tag{3-6}$$

在 8 位位深的有损图像和视频压缩中，典型的峰值信噪比（PSNR）值通常在 30 到 50 分贝（dB）之间，数值越高表示质量越好。对于 12 位图像，当 PSNR 值达到或超过 60 分贝时，被认为具有高处理质量。对于 16 位数据，PSNR 的典型值在 60 到 80 分贝之间。无线传输质量损失的可接受值被认为是大约 20 到 25 分贝。在没有噪声的情况下，两个图像 I 和 K 相同，因此 MSE 为零。在这种情况下，PSNR 是无限的。

（3）结构相似性指标（Structural similarity index，SSIM index）

结构相似性指标[73]是一种用于量化两张数字图像相似程度的评估工具。当其中一张图像为原始无失真图像，另一张为经过某种处理或失真后的图像时，该指标可以作为失真图像质量的一个度量。相较于传统的图像质量评估方法，如峰值信噪比，结构相似性在评估图像质量时更能贴近人眼对图像质量的实际感知。

结构相似性指标的核心思想在于，自然图像通常具有高度的结构化特性，即相邻像素之间存在紧密的相关性，这种相关性承载了图像中物体的结构信息。人类在观察图像时，会习惯性地提取这些结构性信息。因此，在评估图像失真程度时，对结构性失真的考量显得尤为重要。结构相似性指标正是基于这一思想，通过量化原始图像与失真图像之间的结构相似性，来评估图像的质量。

这里使用其一般形式，给定两个图像 x 和 y，两者的结构相似性定义为：

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3-7}$$

其中，μ_x 和 μ_y，σ_x 和 σ_y 分别为 x 和 y 的平均值和标准差，σ_{xy} 为 x 和 y 的协方差，C1、C2 为常量。结构相似性指标的值越大，代表两个图像的相似性越高，即压缩后的图像质量越好。

（4）压缩时间

云桌面系统是一个实时系统，所以图像压缩过程的时间消耗非常重要，过大的时间消耗将影响用户的使用体验，所以将压缩图像的时间作为评测的指标以评估算法的性能。

3.4.3 数据集选择

本文使用了两个数据集，一个是美国 TEXAS 大学图像视频工程实验室提供的 LIVE 图像质量评价数据集[74]。LIVE 图像质量评价数据集来自 LIVE（Laboratory for Image & Video Engineering）与德克萨斯（TEXAS）大学奥斯汀分校心理学系合作进行的一项实验[74]，目的是获得人类受试者对不同类型失真的图像的评分。这些图像是为了支持一个通用的图像研究项目。实验范围很广，其中就包括图像压缩方面的实验，所得结果将有助于研究人员进行图像相关研究或相关算法的性能评估。其中有五个子数据集，每个子数据集有超过 300 张图片，具体如下。

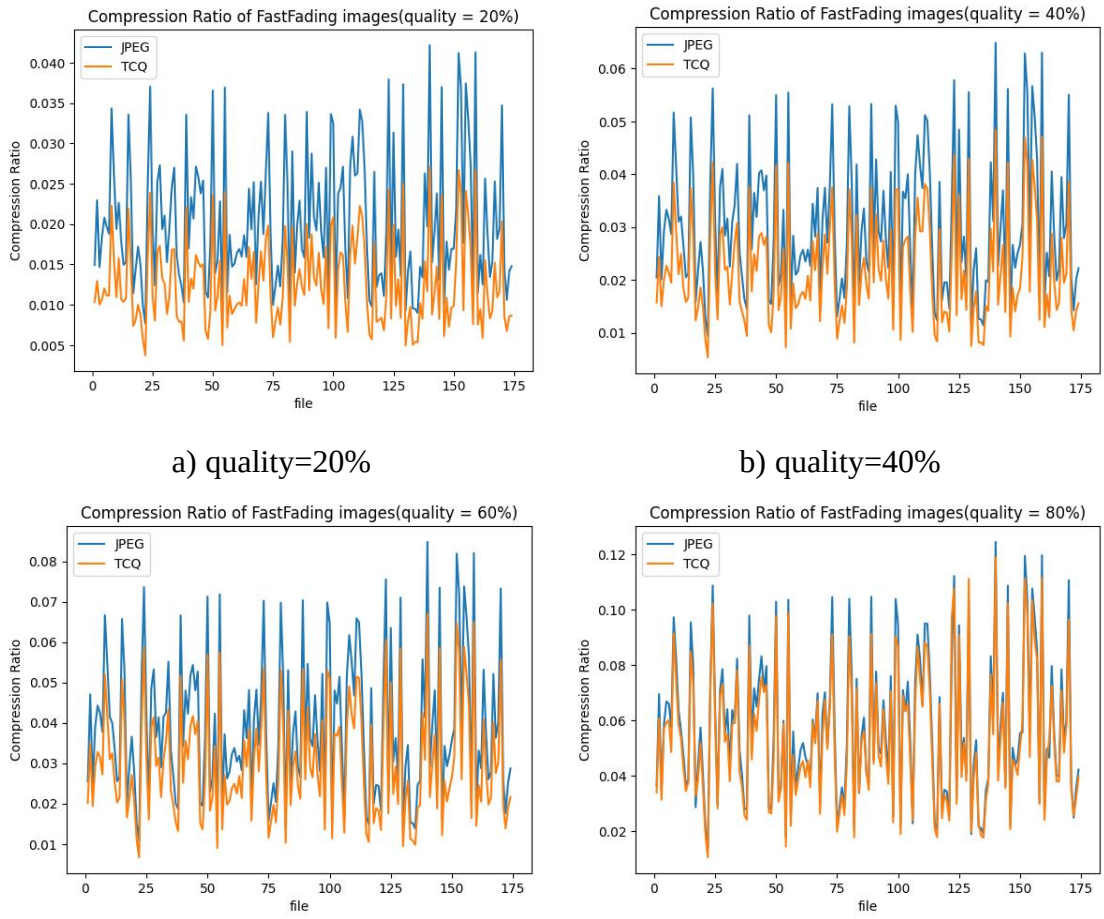
- JPEG——JPEG 压缩图像；
- JPEG2000——JPEG2000 压缩图像；
- Gaussian blur——R, G 和 B 组件使用标准偏差 sigma 的圆对称二维高斯核进行过滤；
- White noise——在图像的 RGB 分量中加入标准差为 sigma 的高斯白噪声；
- Fast Fading Rayleigh——接收机信噪比(模拟)用于生成不同误码比例的图像。

另一个是 Image Compression 官网的数据集[75]，Image Compression 数据集包含 14 张高分辨率高精度图像，这些是从各种来源选择的摄影图像，每个图像都是为了强调算法的不同方面而选择的。一般用于图像压缩研究和算法评估。

3.4.4 实验结果

使用传统的 JPEG 压缩方法和改进的基于 TCQ 的 JPEG 压缩方法，分别将压缩参数 quality 设置为 20%、40%、60%和 80%，在这四种情况下对几个数据集中的图片进行压缩，quality 的范围为 1%-100%，值的大小代表 JPEG 压缩的程度，值越大代表压缩后的图像质量较高，但文件大小相对较大，并比较相应的压缩率和压缩时间，得出优化后算法的性能。

传统 JPEG 与基于 TCQ 的 JPEG 在 quality 参数设置为 20%、40%、60%和 80%的情况下在 Fast Fading Rayleigh 数据集的压缩率对比如图 3-5 所示。横坐标为被压缩的图像编号，纵坐标是压缩率，蓝线表示 JPEG，橙线表示基于 TCQ 的 JPEG。可以得出，TCQ 方法在压缩率方面的表现优于传统方法，特别是在 quality 值较低的情况下。



a) quality=20% b) quality=40%

c) quality=60% d) quality=80%

图 3-5 Fast Fading Rayleigh 数据集下 TCQ 方法与传统 JPEG 方法在不同 quality 参数下压缩率对比

上图仅为 Fast Fading Rayleigh 数据集下的测试结果，所有数据集下测试结果即平均压缩率对比如表 3-2 所示。

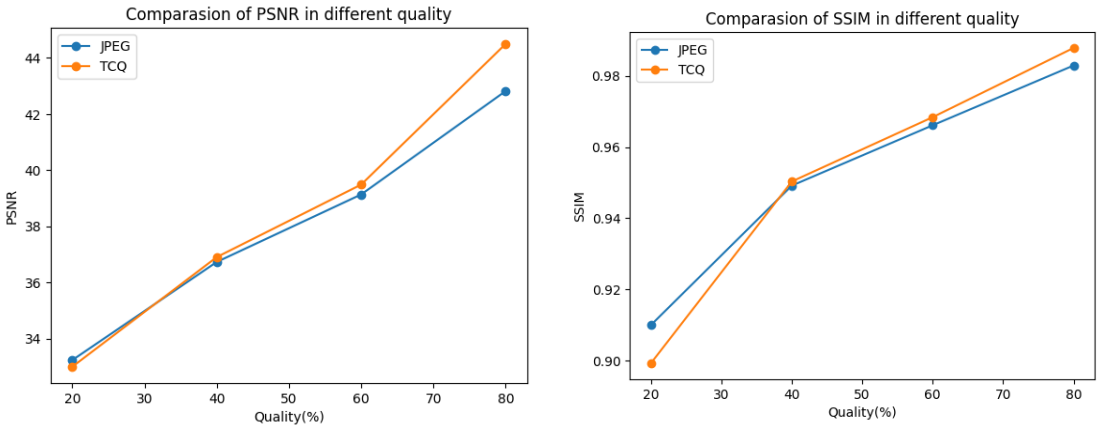
表 3-2 不同 quality 参数下算法性能对比

数据集	算法	平均压缩率（%）			
		20%	40%	60%	80%
JPEG	JPEG	2.12016	2.95915	3.67758	5.02871
	TCQ	**1.31844**	**2.07147**	**2.68007**	**4.22808**
JPEG2000	JPEG	2.22915	3.29554	4.23176	6.13018
	TCQ	**1.40377**	**2.40294**	**3.27115**	**5.51343**
Gaussian Blur	JPEG	1.85549	2.72851	3.51098	5.10602
	TCQ	**1.16955**	**1.98328**	**2.70646**	**4.76101**

表 3-2 （续）

数据集	算法	平均压缩率（%）			
		20%	40%	60%	80%
White Noise	JPEG	4.31707	7.26871	9.77449	14.56206
	TCQ	**2.30407**	**4.87054**	**7.24012**	**15.52051**
Fast Fading Rayleigh	JPEG	2.03596	3.02943	3.91501	5.74753
	TCQ	**1.26751**	**2.19502**	**3.0205**	**5.33834**
Image Compression	JPEG	1.26933	1.95261	2.64293	4.15582
	TCQ	**0.74613**	**1.29904**	**1.87774**	**4.0981**

由上表测试结果可得，基于 TCQ 的 JPEG 压缩在四种 quality 参数下，压缩率表现均优于传统 JPEG 方法，更具体的：JPEG 数据集的压缩率约为 1.32% - 4.22%，JPEG2000 的压缩率为 1.40% - 5.51%，Gaussian blur 的压缩率为 1.17% - 4.76%，White noise 的压缩率为 2.30% - 15.52%，Fast Fading Rayleigh 的压缩率为 1.26% - 5.34%，Image Compression 的压缩率为 0.74% - 4.10%，在几乎所有数据集下均明显优于 JPEG 压缩(除了 White noise)，具体的图像压缩率减少百分比为：quality=20%时 37.0% - 46.6%（平均 39.3%），quality=40%时 27.1% - 33.0%（平均 28.7%），quality=60%时 22.8% - 27.0%（平均 24.4%），quality=80%时 7.2% - 16.1%（平均 9.4%）。实验结果表明优化后的方法在压缩率方面明显优于优化前。



a）PSNR b）SSIM

图 3-6 两种方法压缩图像的 PSNR 与 SSIM 对比

获得了图像压缩率的提升，那么图像质量有没有随之降低呢？为了验证这一问题，选择在不同 quality 参数下测试压缩后的图像对比原图像的 PSNR 值和 SSIM 值，如图 3-6 所示。测试结果中，优化后的算法在 quality 为 40%、60%和 80%的情况下两个参数

值均大于优化前，说明优化过后的算法图像质量比原方法更好，说明该方法在不牺牲图像质量的情况下相较于原算法取得了更好的压缩率。

云桌面是一个实时系统，所以图像压缩过程的时间消耗非常重要，而加入了量化系数的选择过程很有可能会导致压缩时间的消耗，考虑到云桌面系统常用的压缩参数 quality 为 80%左右，所以对参数 quality 为 80%的情况下个数据集压缩时间进行了测试，如表 3-3 所示。

表 3-3 两种方法在参数 quality 为 80%时各数据集压缩时间对比

数据集	压缩时间（ms）			
	JPEG		TCQ	
	总时间	平均时间	总时间	平均时间
JPEG	6240	13.4	7816	16.8
JPEG2000	6297	13.9	8743	19.3
Gaussian Blur	4779	13.7	6048	17.4
White Noise	5227	15	15738	45.2
Fast Fading Rayleigh	4810	13.8	6582	18.9
Image Compression	2376	169.7	3491	249.4

与预期的一样，量化系数的选择导致了压缩时间的消耗。对于大多数云桌面应用，单帧压缩时间保持在 10 到 20 毫秒之内是可接受的，可以提供足够流畅的体验并维持 30 帧每秒的帧率。在上表中，除 Image Compression 数据集外优化后方法平均压缩时间基本都在 20 毫秒以内，这是由于该数据集图像较大，一张图片就有 10 多 MB 甚至 100 多 MB。实验中虚拟机分配的性能较差，实际在高性能服务器下这一时间将会更小。而且除了图像压缩，云桌面系统还有很多其他的部分会消耗时间，这部分时间消耗对于整个系统来说不会有太大的影响，具体体现在云桌面系统中的效果将在第五章进行测试。

3.5 本章小结

本章节首先回顾并分析了主流云桌面协议中使用的 JPEG 图像压缩协议所面对的挑战，并探索了可能的改进路径。随后，引入了网格量化 TCQ 方法，并论述了将 TCQ 应用于 JPEG 编码的潜力。通过改进 JPEG 的量化过程，创建一个量化系数候选集，评估得到失真与比特率之间平衡最佳的量化系数，以此提高 JPEG 编码的效率，降低云桌面

系统中图像传输的带宽需求。章节的最后部分，在美国 TEXAS 大学图像视频工程实验室提供的 LIVE 图像评价数据集和 Image Compression 官网提供的数据集上对提出的改进方案进行了一系列实验，将改进后的方法与原方法进行了细致的性能对比。并对实验结果进行分析和展示，表明了改进后的方法在压缩率上有显著提高。

第 4 章 渐进式 JPEG 扫描分割优化

渐进式传输方法最初被用于JPEG压缩过程当中仅是为了使解码时可以先呈现图像的大致轮廓，而不是等到所有信息都被解码之后再呈现在用户面前。但人们发现，通过特定的扫描分割方法，可以优化渐进式 JPEG 的压缩率，所以逐渐有研究人员对这方面进行探索。本章将介绍一种特殊的渐进式 JPEG 扫描分割方法，用这种方法对原版的 JPEG 进行改进优化。最后将与第三章方法结合的改进后的方法替换掉云桌面系统中原本的图像压缩方法，并在第五章进行综合测试。实验表明，改进后的方法对比原版 JPEG 方法在压缩率上有所提升。

4.1 渐进式 JPEG 介绍

在渐进式 JPEG（Progressive JPEG, PJPEG）模式中，8×8 的块以相同的顺序形成，然后进行离散余弦变换（DCT）变换并且量化到特定数量的位。量化后的 DCT 变换系数接着在多次扫描中被部分编码，每次扫描对应于一个或多个 DCT 变换系数的几个位，因此代表了正在编码/解码的图像的一部分。DC 系数是 8×8 块的平均值，当仅解码 DC 系数时，就会获得包含统一灰度块的图像。8×8 系数中前几行和列的低频 AC（交流）系数分别很好地代表了图像中的垂直和水平边缘。高频 AC 系数代表了图像中细微的、往往不那么重要的细节。每一个 DCT 系数都贡献着不同的图像质量，可以按照 PJPEG 模式的规定划分为许多组或分解为位，以使得随着接收和解码更多组或 DCT 系数的位，重构的图像质量得到提升。

在实现 PJPEG 的多种不同的方法中，通常有两种主要的方法，通过这些方法可以在一次扫描中部分编码量化系数：一种方法被称为光谱选择，它将每个 8×8 块中的锯齿序列 DCT 系数划分为不同长度的频带。然后，在一次扫描中对每个频带进行编码。另一种方法被称为逐步逼近，在每次扫描中，DCT 系数不需要编码到其全部精度。通过将 8×8 块的 DCT 系数除以 2 的幂（或将其二进制表示向右移位）来降低这些系数的精度，然后再对这些系数进行编码。接收到的 DCT 系数在 IDCT（逆离散余弦变换）操作前被解码并乘以相同的 2 的幂（或将其二进制表示向左移位）。在后续的每次扫描中，DCT 系数的精度增加一位。这两种方法可以独立使用或以灵活的方式结合使用，产生多种组合。以下小节将详细描述光谱选择方法和逐步逼近方法。

4.1.1 光谱选择方法

光谱选择是一种在渐进式 JPEG 编码中广泛使用的技术，其核心思想是将 DCT（离散余弦变换）系数根据频率划分成不同的频带，并在不同的扫描中对这些频带逐一进行编码。这种方法的优点在于能够让解码器在接收到所有数据之前，先重建图像的大致轮廓和主要特征，从而实现较快的图像预览效果，而且，由于不同频段的系数具有不

同的特征，分别扫描往往能达到更高的压缩效率。在实际场景中，

在具体实现时，光谱选择首先将每个 8x8 像素块的 DCT 系数按照它们的频率特性进行排序，通常是按照 zigzag 锯齿形序列。然后，这些系数被分成几个频段，每个频段包含一定范围内的低频到高频系数。每次扫描仅编码一个或多个频段中的系数，从而逐步提高图像的频率细节和质量。

图 4-1 展示了一种光谱分割方式。其中，8×8 块的 64 个系数被分为四个部分，分别是{J1[0]，J2[1:5]，J3[6:20]，J4[21:63]}，可以根据频率优先级顺序来安排每个任务，并且可以在任何给定时间终止任务，仍然能够产生结果。这种方法的灵活性允许编码过程在必要时提前结束，而不会完全丧失已经处理的数据的价值。

压缩图像中的高频带往往包含很多零值，这是因为图像的大部分能量都集中在低频区域，而高频区域往往代表了图像的细节部分。在许多情况下，这些细节可能不是特别显著或对于图像的总体感知质量不是非常关键。因此，通过将低频带分组，可以优先对这些包含大量图像信息的区域进行编码，从而实现最大的编码效益。

在这种分组方法中，J1 块的 DC 系数和 J2 块包含的低频系数将被优先处理，因为它们代表了图像的主要视觉内容。随后，J3 和 J4 块中的高频系数，这些系数往往包含较少的图像能量且可能只对图像的细节贡献略微，将在之后的阶段被编码。这种分层和优先级的方法使得在图像的早期阶段就能获得较好的视觉效果，同时保留了在后续阶段进一步提升图像质量的可能性，同时也能提升压缩效率。

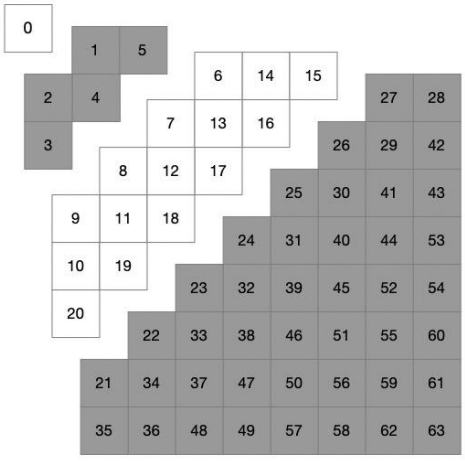


图 4-1 一个光谱分割方法的例子

4.1.2 逐次逼近方法

逐次逼近是另一种在渐进式 JPEG 编码中使用的技术，它允许在每次扫描中逐渐提高 DCT 系数的精度。这种方法的关键在于，初次扫描不会完全精确编码每个系数，而是将它们简化为更粗略的近似值。随着后续扫描的进行，这些系数的精度逐步提高，从而逐步增强图像的细节和质量。对于系数值而言，最高有效位（Most Significant Bits，

MSB）具有最高的权重，可以首先被编码并发送，而最低有效位（Least Significant Bits，LSB）可以稍后编码并发送。这种方法利用了图像 DCT 系数的一个重要特性：图像的大部分能量都集中在低频系数中，这意味着高频系数（尤其是在高位平面中）往往可以被量化为零而对视觉质量的影响最小。因此，通过优先传输更重要的信息（如 DC 系数和 AC 系数的高位平面），即使在较低的比特率下，也能保证图像的基本视觉质量。

在逐次逼近方法扫描的过程中，每个 8×8 像素块的 DCT 系数首先被简化，以降低其精度。这通常通过对系数进行右移操作（即除以 2 的幂）来实现，从而生成一系列较为粗略的近似值。初次扫描编码的是这些简化后的系数，它们占用的比特数较少，因此可以快速传输。随着编码过程的进行，每一轮的扫描都会逐步增加系数的精度。具体来说，之前被简化的系数在解码的时候通过后续扫描的系数逐步恢复，每次通过解码时的左移操作（即乘以 2 的幂）来实现，逐渐接近其原始值。这种方法允许每一步扫描都在图像质量上做出微小的提升，直到所有的扫描完成，系数恢复到其最大精度，图像质量达到最优。

采用逐次逼近方法传输图像时，即使在部分数据丢失或延迟到达的情况下，仍然可以保证接收端获得图像的一个基本版本，这大大提高了传输过程的鲁棒性。此外，由于最高有效位的信息（MSB）首先被发送，它为图像的整体结构和重要特征的快速重建提供了基础，从而实现了高效的渐进式图像传输。逐次逼近方法通过将 DCT 系数的每一位分别编码和传输，为图像传输提供了更细致的控制，从而在不同的解码阶段逐渐提升图像的质量，这种方法尤其适合于那些对实时性和传输效率有较高要求的应用场景。

4.1.3 光谱选择结合逐次逼近方法

光谱选择和逐次逼近方法除了可以单独使用之外，还能相结合起来。这样兼顾了图像质量与编码效率。通过这种综合方法，不仅能够优先处理和传输图像中最关键的视觉信息，还能在此基础上逐步细化图像的细节，实现渐进式的图像质量提升。

假设 8 位输入精度，对应每个 DCT 系数具有 11 位精度，10 位用于值的大小和 1 位用于符号。符号位在优先考虑位时不被单独考虑，因为它总是与第一个非零最高有效位一起编码。暂时不考虑符号位，一个 8×8 的 DCT 系数块可以被视为包含 8×8×10 位的长方体。图 4-2 展示了一个 8×8×10 的长方体。解码器假设所有零值系数的 8x8 块对应的长方体都是 0。随着位的解码，0 被实际位的值所替换。

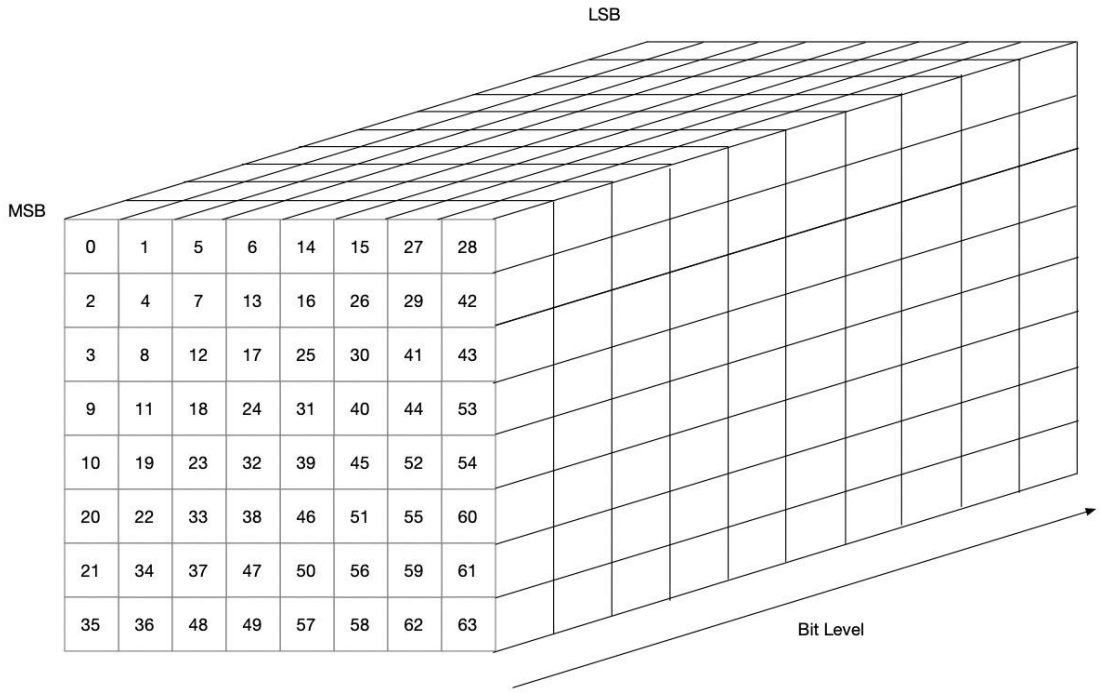


图 4-2 8×8×10 的长方体

例如，将 DCT 系数分为两部分，高 5 位和低 5 位，分别用 J_1、J_2 表示。它们对应的频段分割方式如下：

J_1 = {$J_{1,1}$ [0:4], $J_{1,2}$ [5:9], $J_{1,3}$ [10:63] };

J_2 = {$J_{2,1}$ [0:2], $J_{2,2}$ [3:7], $J_{2,3}$[8:15], $J_{2,4}$ [16:63] }

假设 J_1 优先级高于 J_2，则扫描的顺序由图 4-3 所示。

这种结合方法的优势在于其灵活性和效率。以上两种方法各有优劣，频带和不同的位分别存储了图像不同的细节。通过结合这两种方法，不仅能够在有限的带宽下快速传输图像，还能在用户端实现渐进式图像显示，即使在图像数据未完全接收的情况下，也能提供有用的视觉信息。此外，这种策略在处理网络波动或带宽限制时表现出了较高的鲁棒性。即便在数据传输受到干扰的情况下，接收端也能基于已接收的部分数据，重建出一个相对较好的图像版本，保证了用户体验的连续性和稳定性。

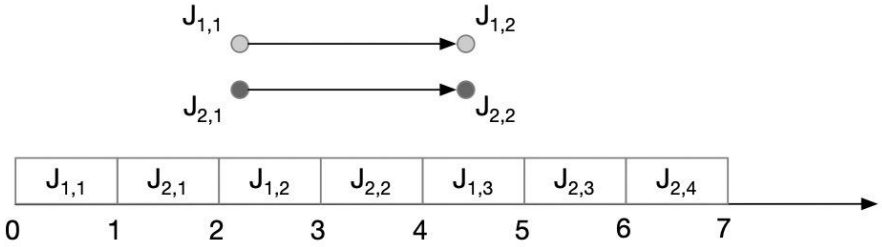


图 4-3 一个扫描顺序例子

4.2 扫描分割方法分析

在 JPEG 标准中，一块 8×8 像素的图像通过离散余弦变换（DCT）转换，产生的系数被量化并重新排序，即 zigzag 扫描，后续部分可以被视为减小编码值的幅度的过程，同时增加连续零的数量，以便随后的行程编码（Run-length Encoding，RLE）和哈夫曼编码过程更有效。因此，通过进一步减小系数的幅度值并增加零的连续长度，可以提高 JPEG 的压缩率。基于这一特性，采用一种新的渐进式 JPEG 分割方式，将系数中的小系数提取出来单独编码，大系数留下，原位置用零替代，这样就可以增加原序列中连续零的长度，提高压缩率，同时提取出来的小系数可以单独编码，先传输大系数序列，后传输小系数序列，再分别进行解码还原图像。

在 JPEG 压缩标准中，图像压缩过程始于将 8×8 像素块的图像数据通过离散余弦变换（DCT）转换为频率域表示。转换后得到的 DCT 系数随后经过量化处理，以降低图像信息的精细度，从而实现数据的压缩。接着，这些量化后的系数会按照一种称为“zigzag”扫描的方式进行重新排序，此步骤旨在将低频系数（通常包含更多图像能量）与高频系数（往往是细节部分，且更可能接近零）分离，从而使得后续系数中出现更长的零值序列。这样的排列方式为随后的行程编码（Run-length Encoding，RLE）和哈夫曼编码步骤提供了高效编码的可能性，因为这两种编码技术都对连续相同值（特别是零）的序列更为高效。

考虑到以上特性，本文采用了一种新的系数分割策略，该策略进一步细化了系数的处理方法。具体来说，本策略在传统的量化和 zigzag 排序之后，通过将较小的系数从原始 DCT 系数块中分离出来，并将其位置用零填充，从而在原序列中创造出更长的零值序列。这种方法显著增加了连续零的长度，为行程编码和哈夫曼编码提供了更为优化的条件，进而提高了整体的压缩效率。与此同时，被分离出的较小系数将被单独编码，这种编码可以更加紧凑，因为这部分系数的动态范围较小。在图像传输过程中，首先传输经过零填充处理的主要系数序列，随后传输被分离的小系数序列。在接收端，通过对这两个序列的解码和重组，可以高效还原出原始图像。

此外，这种系数分割方法不仅提高了压缩效率，同时也可能对图像质量产生积极影响。由于大系数往往对应图像的主要视觉内容，而小系数则与细节和纹理信息有关，因此，这种分离策略在保持图像主要视觉特征的同时，对于图像的细节处理提供了更大的灵活性。这样的处理可能使得在压缩比例相同的情况下，图像质量得到进一步的保障，特别是在高压缩比需求的应用场景中。

通过对 JPEG 压缩流程中 DCT 系数的精细处理，本文使用的系数分割方法为提高压缩率同时维持甚至提升图像质量提供了一种有效途径，展现了在现有压缩标准下进一步优化的潜力。

4.3 扫描分割设计

4.3.1 分割过程

在对 JPEG 图像块的 AC 系数进行处理时，首先提取系数值为 1 和-1 的项，并形成两个单独的序列：一个中间序列和一个尾部序列。这一操作的目的是为了利用 zigzag 扫描中，排序靠后的 DCT 系数再经过量化之后会出现更多更长的连续零序列这一特性，所以这里将尾部序列单独提取出来编码，其分割依据为块的 AC 系数中最后一个绝对值大于 1 的系数，其前面提取的值为中间序列，后面提取的值为尾部序列。中间序列包含了初步提取的 1 和-1 的系数，而尾部序列则包含的连续零序列，以便更有效地利用这一量化后的特性。为了在熵编码阶段准确地标识这些被提取系数在原始 AC 系数序列中的具体位置和长度，在原始序列中相应的位置上放置了零值作为占位符。在 JPEG 标准中，后续熵编码时尾随的零会被删除，用一种名为结束符（End of Block，EOB）的特殊符号替代，用于表示一维 DCT 系数序列的结束，尤其是在遇到一串连续零时。在霍夫曼编码的上下文中，EOB 是一个特定的码字，它告诉解码器在当前块中所有剩余的系数都是零，并且该块的处理在此结束。在提取过程中，不仅提取了 1 和-1，还提取了 1 和-1 之间的零，这样的目的是在解码时可以方便找到提取的 1 和-1 在原始序列中的正确位置。分割方法如图 4-4 所示。

采用上述系数分割策略后，获取到了经过重新组织的 AC 系数序列，该序列接下来将按照标准 JPEG 编码流程进行处理。此外，还得到了两个独立的序列——即中间序列和尾部序列，其中均包含了系数值为 1 和-1 的项及其间的零值。这两个序列将采用特定的编码方法进行处理。在解码过程中，首先使用标准的 JPEG 解码器对主要的 AC 系数序列进行解码，获取图像的主要信息。随后，再对中间序列和尾部序列进行解码，以细化和完善图像的细节部分。

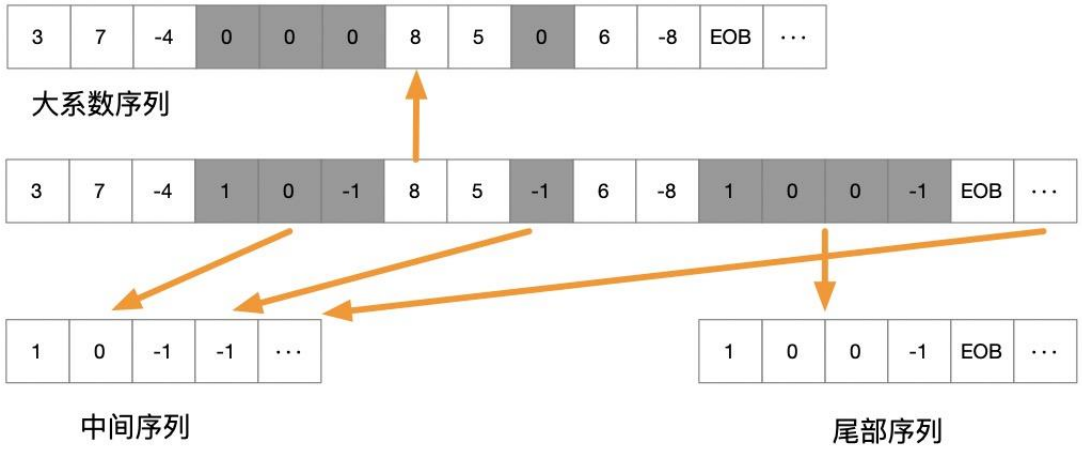


图 4-4 系数分割方法

4.3.2 编码方法

4.3.2.1 剩余大系数序列编码

在完成了 4.3.1 中所述的 AC 系数分割过程之后，剩余的大系数序列拥有两个显著的特征：一是序列中出现了更长的连续零段，二是序列中不再包含值为 1 或-1 的系数。这些变化对于基于行程编码机制的 JPEG 压缩算法来说具有重要意义。行程编码依赖于将连续的相同值来进行数据的编码，分割过后的序列可以将连续的零值和紧随其后的一个非零系数编码为一对数据，因此，更长的零段有助于显著提升数据的压缩效率。

在此基础上，通过优化用于 AC 系数编码的哈夫曼表，可以进一步提高压缩比。通常，在 JPEG 标准的编码过程中，由于 1 和-1 的系数出现频率最高，它们被分配了最短的编码值。然而，在当前的剩余序列中，由于不包含这些高频系数，可以针对剩余序列来优化哈夫曼表从而优化剩余系数的压缩过程，从而实现更高的压缩率。尽管为剩余系数序列定制优化的哈夫曼表可以进一步提高压缩率，这一过程伴随着不小的计算成本，特别是在需要针对每张图像动态生成优化哈夫曼表的情况下。因此，为了平衡压缩效率和计算复杂性，这里采取了另一种策略：将剩余序列中的系数统一减去 1，然后利用标准的哈夫曼表进行编码。编码过程如图 4-5 所示。

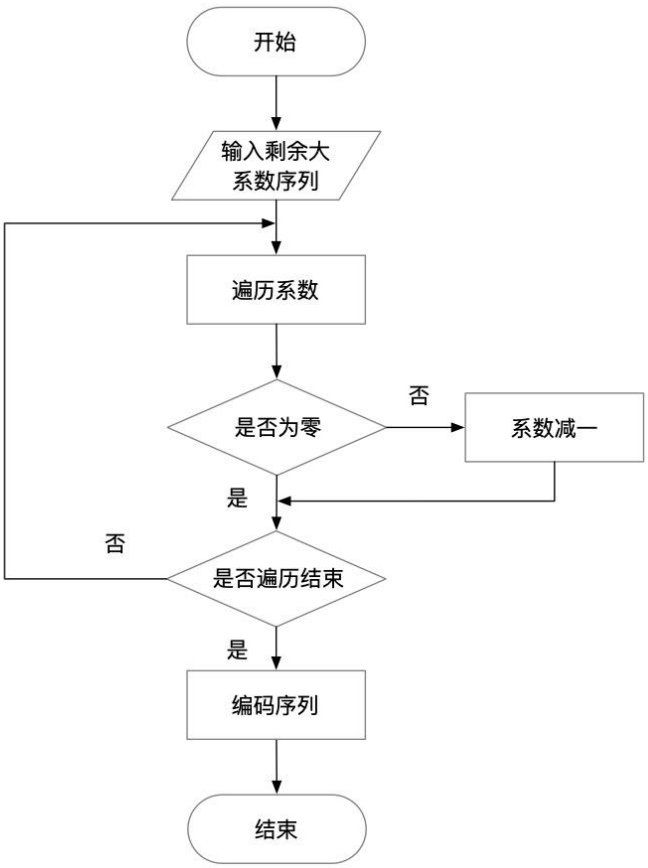


图 4-5 剩余序列编码过程

这种方法虽然可能不如为每个序列量身定制的哈夫曼表在编码效率上那么高，但它避免了对每张图像进行哈夫曼表优化的额外计算负担，同时在许多场景下仍能为 AC 系数分配更短的代码，实现有效的压缩。

如 4.3.1 节所述，经过处理提取的系数序列被明确地划分为两个不同的组别：中间序列和尾部序列，每组因其独有的特征而区分。中间序列来自图像块的中间部分，这些值在原序列中随机出现。尾部序列与中间序列相比具有相对较长的连续零序列。将图像中每个块的相应序列类型进行合并，即分别将所有中间序列和所有尾部序列连接起来，形成两个分别具有统一特性的长序列，再使用特定的方法对它们进行编码。这种方法不仅有助于维持序列内部的一致性，而且通过集中处理具有相似特征的数据，可以进一步优化压缩算法的效率。

4.3.2.2 中间序列编码

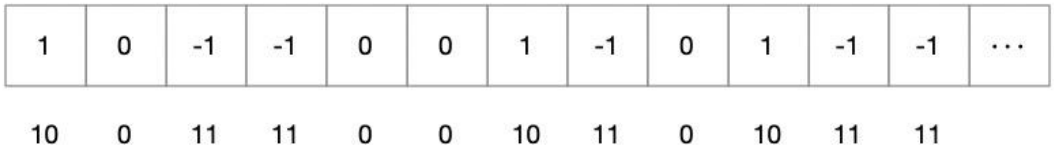


图 4-6 中间序列编码

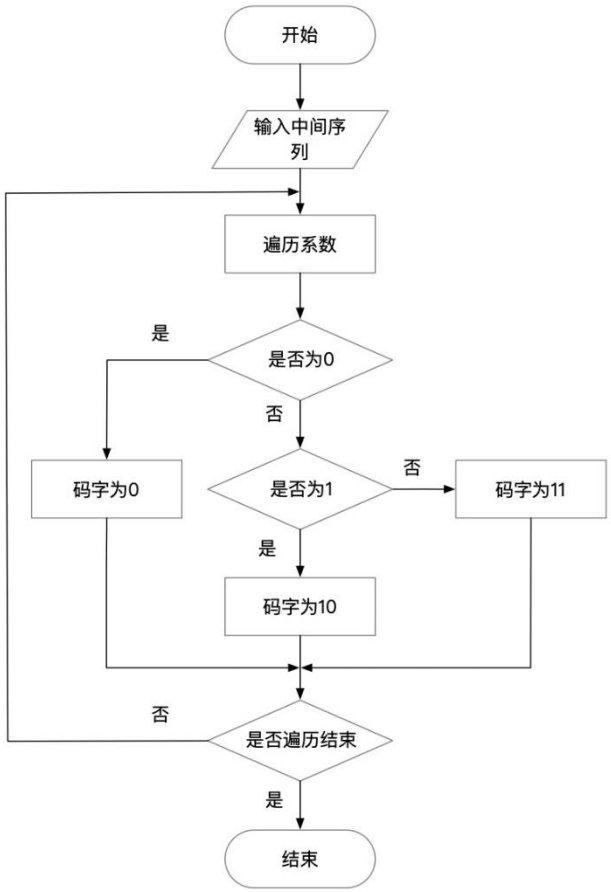


图 4-7 中间序列编码过程

中间序列的特征主要体现在其包含的值仅限于零、1 和-1 这三种情况，其中零值在大多数图像数据中出现的频率远高于 1 和-1。基于这一观察，可以明确的为这三种值分配编码值：将 0 编码为 0，1 编码为 2（二进制为 10），而-1 则编码为 3（二进制为 11），数据对应码字如图 4-6 所示，编码过程如图 4-7 所示。

值得注意的是，采用这种显式的编码分配方法可以在一定程度上加速整个数据的编码和解码过程。这是因为在实际的编码解码操作中，无需依赖于一个完整的霍夫曼编码表，从而简化了查找和匹配过程，降低了处理复杂度。这种方法的实施不仅保持了编码的效率和有效性，而且还提高了处理速度，特别适合于需要快速处理大量图像数据的应用场景，为云桌面系统高效图像压缩和传输提供了一种实用的策略。

4.3.2.3 尾部序列编码

对于尾部序列，其显著特征在于包含较长的连续零序列，这为进一步的压缩提供了有利条件。为有效利用这一特点，行程编码方法成为了处理此类数据流的理想选择，因为它专门设计用于减少连续重复数据的表示长度，从而有效降低所需的数据量。在行程编码的实施过程中，连续零序列被编码为单一的代码字，大大简化了数据的表示。

与 JPEG 标准中采用的编码策略类似，这里使用的行程编码不仅表示连续零序列的长度，还将其与序列后继非零系数的符号进行配对。具体来说，将连续零的长度与下一个非零值的符号（其中 1 表示正值，2 表示负值）组合为一对数据，形成了（长度，符号）这样的数据对，如图 4-8 所示。编码过程如图 4-9 所示。

这种编码方式不仅减少了数据的体积，还保留了足够的信息以确保数据在解码过程中能够被准确还原。

在处理每个连续零序列和其后继非零系数组成的运行符号对时，理论上可以通过构建专门针对当前数据特性的最优哈夫曼树来确定每个数据对的码字，以达到最高的编码效率。然而，考虑到在实际应用中需要权衡计算效率和编码效率之间的关系，与前面一样，这里沿用了 JPEG 标准所提供的哈夫曼表来进行编码工作。这一决策背后的考虑不仅基于计算资源的优化使用，还考虑到了与现有 JPEG 标准的兼容性。

这也是上面用 1 表示正值，2 表示负值的原因。因为 JPEG 标准 AC 系数哈夫曼表中，数据对用（run，length）表示，其中 length 为 0 的情况仅有两种特殊编码：一个是（0，0）用于表示块的结束（End of Block，EOB），另一个是（15，0）用于指示一个长达十六个连续零的序列。因此这里从 1 开始对正负值进行编码，旨在适配 JPEG 标准哈夫曼表的这一特性。

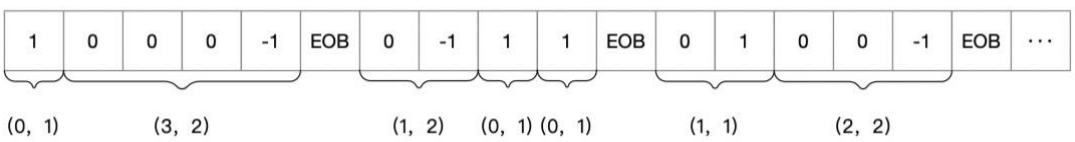


图 4-8 尾部序列编码

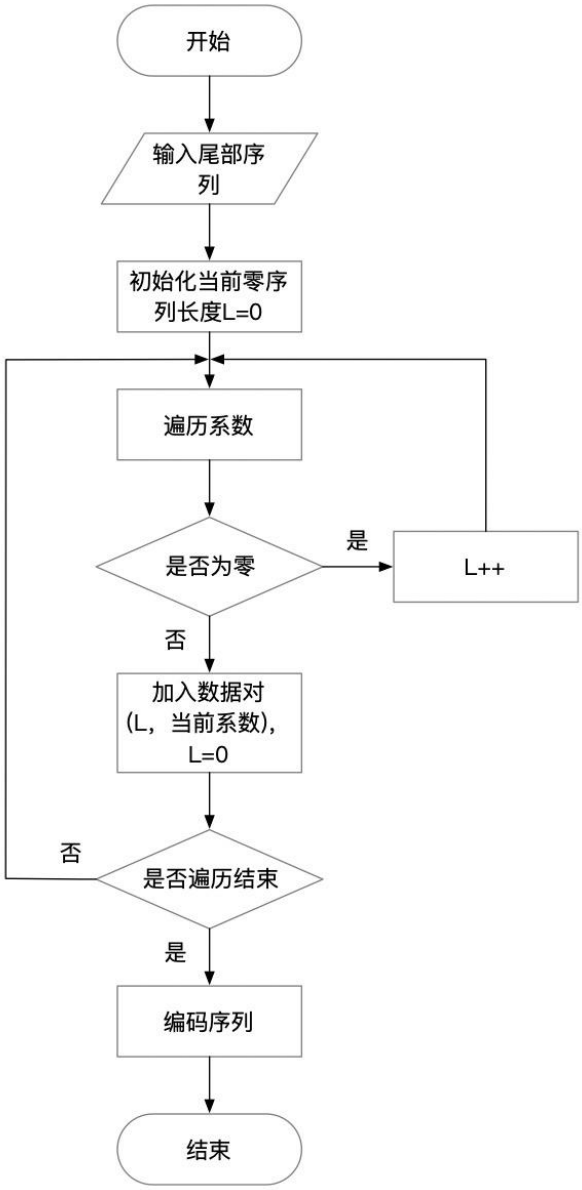


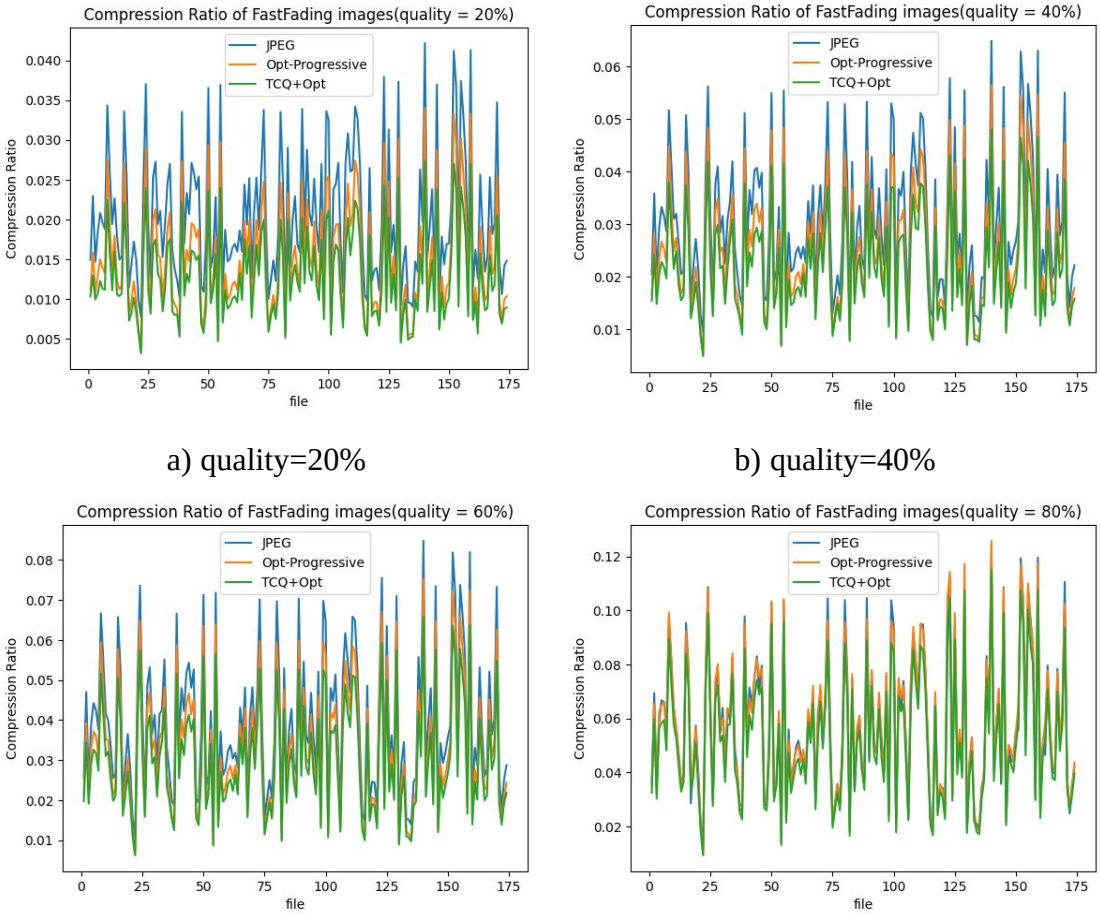
图 4-9 尾部序列编码过程

在对尾部序列的长度进行确定时，其处理方式与中间序列的处理不同。对于中间序列，其长度的判定依赖于在进行系数分割后剩余的大系数序列中连续零的出现频率及其分布模式。具体而言，这一序列部分的长度可通过观察剩余大系数中连续零序列的长度来加以确定，这种方法利用了大系数序列在经过分割处理后形成的连续零段的特性。相比之下，尾部序列的长度判定则需要使用不同的策略，这主要是由于分割尾部序列时的策略决定的。由于尾部序列最后会有较长的零序列，所以与原序列一样，使用 EOB 来标准尾部序列的结束。

4.4 实验与分析

在本节中，为了证明本章改进的方法能够在不影响图像质量的情况下，降低图像的压缩率，在 3.4.3 中提到的几个数据集下做了多组对比实验，首先将优化前的 JPEG 算

法与本章改进的方法（记为 Opt-Progressive）相对比，还加上了第三章中 TCQ 方法与本章方法的结合（记为 TCQ+Opt），为第五章云桌面系统中的实验做铺垫。对数据集中包含的图像进行压缩，对比不同压缩参数下压缩率。几种方法以不同的 quality 参数在 Fast Fading Rayleigh 数据集下的压缩率对比如图 4-10 所示。横坐标为被压缩的图像编号，纵坐标是压缩率，蓝色、橙色、绿色分别表示优化前的 JPEG 算法、本章改进的优化渐进式压缩算法以及本章方法与第三章中 TCQ 方法相结合的算法。可以得出，在压缩率上本章方法与结合后的方法表现优于优化前的方法。



a) quality=20%　　b) quality=40%

c) quality=60%　　d) quality=80%

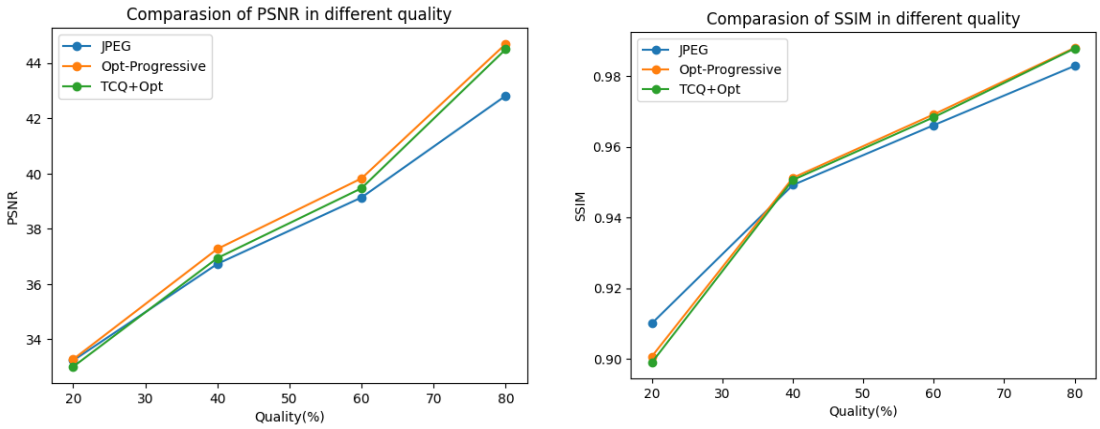
图 4-10 Fast Fading Rayleigh 数据集下几种方法在不同 quality 参数下压缩率对比

上图仅为 Fast Fading Rayleigh 数据集下的测试结果，所有数据集下测试结果即平均压缩率对比如表 4-1 所示。

表 4-1 不同 quality 参数下算法性能对比

数据集	算法	平均压缩率（%）			
		20%	40%	60%	80%
JPEG	JPEG	2.12016	2.95915	3.67758	5.02871
	Opt-Progressive	**1.52602**	**2.29720**	**2.90688**	**4.46444**
	TCQ+Opt	1.31597	2.05982	2.65485	4.15422
JPEG2000	JPEG	2.22915	3.29554	4.23176	6.13018
	Opt-Progressive	**1.36782**	**2.74569**	**3.63775**	**5.92682**
	TCQ+Opt	1.40876	2.39885	3.24799	5.42601
Gaussian Blur	JPEG	1.85549	2.72851	3.51098	5.10602
	Opt-Progressive	**1.67745**	**2.24469**	**2.99855**	**5.02929**
	TCQ+Opt	1.17794	1.98601	2.69215	4.68810
White Noise	JPEG	4.31707	7.26871	9.77449	14.56206
	Opt-Progressive	**3.35608**	**6.05595**	**8.33063**	**16.63716**
	TCQ+Opt	2.31277	4.79796	7.05912	14.93505
Fast Fading Rayleigh	JPEG	2.03596	3.02943	3.91501	5.74753
	Opt-Progressive	**1.50638**	**2.50301**	**3.35820**	**5.54733**
	TCQ+Opt	1.27227	2.18929	2.99573	5.24484
Image Compression	JPEG	1.26933	1.95261	2.64293	4.15582
	Opt-Progressive	**0.83601**	**1.48894**	**2.13939**	**4.01157**
	TCQ+Opt	0.71413	1.27515	1.84996	3.99972

由上表测试结果可得，使用优化渐进式压缩的 JPEG 算法在四种 quality 参数下，压缩率表现均优于传统 JPEG 方法，更具体的：JPEG 数据集的压缩率约为 1.52% - 4.46%，JPEG2000 的压缩率为 1.36% - 5.92%，Gaussian blur 的压缩率为 1.67% - 5.03%，White noise 的压缩率为 3.35% - 16.64%，Fast Fading Rayleigh 的压缩率为 1.51% - 5.75%，Image Compression 的压缩率为 0.84% - 4.01%，在几乎所有数据集下均优于 JPEG 压缩(除了 White noise)，具体的图像压缩率减少百分比为：quality=20%时 10.8% - 35.7%（平均 27.1%），quality=40%时 16.7% - 24.2%（平均 19.7%），quality=60%时 14.2% - 21.0%（平均 15.8%），quality=80%时 2.6% - 11.2%（平均 5.3%）。观察组合后的方法的性能表现，可以得出 TCQ+Opt 在大部分情况下优于单独使用 TCQ 方法和优化渐进式压缩方法。



a) PSNR b) SSIM

图 4-11 几种方法压缩图像的 PSNR 与 SSIM 对比

在不同 quality 参数下对以上实验压缩后的图像对比压缩前图像计算 PSNR 值与 SSIM 值，如图 4-11 所示。测试结果中，优化后的算法在 quality 为 40%、60%和 80%的情况下两个参数值均大于优化前，说明优化过后的算法图像质量比原方法更好，说明该方法在不牺牲图像质量的情况下相较于原算法取得了更好的压缩率。

与第三章一样，在参数 quality 等于 80%时对比几种方法压缩图像的时间消耗，对其性能做出评估。结果如表 4-2 所示。

表 4-2 两种方法在参数 quality 为 80%时各数据集压缩时间对比

数据集	压缩时间（ms）					
	JPEG		Opt-Progressive		TCQ+Opt	
	总时间	平均时间	总时间	平均时间	总时间	平均时间
JPEG	6240	13.4	6229	13.4	8007	17.2
JPEG2000	6297	13.9	6874	15.1	8930	19.7
Gaussian Blur	4779	13.7	5145	14.8	6420	18.4
White Noise	5227	15	7057	20.3	16271	46.8
Fast Fading Rayleigh	4810	13.8	5367	15.4	6943	20
Image Compression	2376	169.7	4506	321.9	7824	558.9

扫描分割的选择过程导致了压缩时间的消耗，在 3.4.4 中已经提到过，在大多数云桌面应用中，单帧压缩时间在 10 到 20 毫秒之内可以为帧率为 30 帧下运行的云桌面系统提供足够流畅的体验，上表中除 Image Compression 数据集外优化后方法平均压缩时间基本都在 20 毫秒以内。且该时间消耗的增加集中在编码端，即云桌面协议的服务端，

服务端的强大 CPU 性能能够缩小这一时间。Image Compression 数据集上时间消耗增加较大，是由于该数据集图像较大，一张图片就有 10 多 MB 甚至 100 多 MB，这种情况在云桌面系统运行过程中不会出现，所以可以排除。结合后的算法的评估将在第五章进行。

4.5 本章小结

本章首先阐述了渐进式 JPEG 技术的基本概念与工作原理，并紧接着深入分析和总结了一些最新的相关研究成果。基于这些研究，本章采用了一种改进的扫描分割策略，旨在优化传统 JPEG 中的渐进式压缩技术。该策略通过对 DCT 系数按特征进行分组，并根据不同的优先级分别对其进行编码，从而有效提升了压缩效率。在章节的后半部分，通过一系列精心设计的对比实验，在相关数据集上验证了本章的优化方案，将其性能与原始方案进行了详尽对比。此外，还将改进方案与第三章的 TCQ 方法相结合，并将此结合方案加入实验对比，为第五章的综合实验奠定了基础。通过对实验结果的系统分析和呈现，清晰地展示了改进方案在压缩率方面有所提升。

第 5 章 系统测试与分析

本章主要对经过优化的云桌面协议进行全面的测试与分析。首先将详细介绍系统测试的环境设置，包括硬件配置、操作系统版本，确保实验结果的可靠性和可复现性。并与改进前的协议性能进行对比实验，旨在从多角度比较优化前后的性能差异。随后，章节将转入系统性能的综合测试阶段。鉴于本次优化主要聚焦于提升系统效率和资源使用的优化，选择 CPU 占用率、内存占用率以及网络带宽占用这三个关键性能指标。通过对这些指标的精确测量，能够定量地评估优化措施对系统性能的具体影响。通过对实验数据的详尽分析，本章旨在揭示经过优化的云桌面协议在实际应用中的表现。

5.1 系统测试环境

本节将详细介绍本章系统测试所依赖的机器环境配置。将云桌面协议部署在 Debian 操作系统上，为了全面评估优化措施的效果，测试环境包括一台高性能虚拟化服务器，该服务器上配置了四台虚拟机，每台虚拟机都安装了 Debian 操作系统。这些虚拟机分别扮演优化前后云桌面协议的服务端和客户端，从而提供了一个理想的对照实验环境。为确保实验的准确性和可靠性，每台虚拟机都被赋予了统一且充足的资源分配，包括处理器核心数、内存大小等。

详细的机器配置如表 5-1 中所示。实验环境旨在为云桌面协议的优化效果提供一个全面、准确的评估基础，确保实验结果的有效性和实际应用的参考价值。

表 5-1 实验环境机器配置

机器名称	CPU	内存	硬盘	操作系统
虚拟化服务器	Intel(R) Xeon(R) E5-2678 v3 2.50GHz	128G	4T	CentOS7
优化前云桌面协议客户端	vCPU(4 核)	4G	40G	Debian12
优化前云桌面协议服务端	vCPU(4 核)	4G	40G	Debian12
优化后云桌面协议客户端	vCPU(4 核)	4G	40G	Debian12
优化后云桌面协议服务端	vCPU(4 核)	4G	40G	Debian12

5.2 实验评价标准

云桌面协议通常有许多使用场景，考虑到各种场景的需求，使用以下三个指标来评价优化的结果。

（1） CPU 占用率

云桌面协议是随着云计算的需求而诞生的，其目的是客户端的瘦化以及最大化利用硬件资源，因此服务器和客户端的 CPU 等硬件设备的占用就显得非常重要。云桌面协议工作过程中频繁使用图像压缩算法来编码每一帧图像，而这一过程需要大量的 CPU 计算量。因此对云桌面图像压缩的优化可能会带来过高的 CPU 占用率。因此对云桌面协议客户端和服务端 CPU 占用率进行测试来评估优化的效果。

（2）内存占用率

内存是一个系统最重要的基础硬件资源之一，它和 CPU 资源一样重要，任何计算过程都需要用到内存，程序的运行同样依赖内存。所以选用内存占用率来评估云桌面系统的性能，通过优化前后的客户端和服务端的内存占用率对比，可以衡量优化后的云桌面协议的性能和可用性，如果内存占用率过大，该系统将不具有实用性，所以内存占用率是评估云桌面系统性能的重要指标。

（3）网络带宽占用

云桌面协议客户端和服务端的交互是通过网络传输各种数据来进行的，带宽占用可以直接影响到用户的使用体验。云桌面协议传输需要的带宽能直接影响到用户的体验，如果需要的带宽很大，那么在低带宽环境或者网络环境不佳的时候会极大的影响用户使用云桌面系统的流畅性。同时，本文对云桌面协议优化的内容主要是压缩率方面，这可以通过网络带宽直接体现，所以带宽占用是评价云桌面协议优化的重要性能指标。

针对上述三个评价标准，根据画面的变化快慢设计了三种不同的用户使用场景来进行实验，包括低速、中速、高速运动场景。以下将展示实验的结果，并对结果做出分析。

5.3 实验结果及分析

本节将对优化前后的云桌面协议在不同的用户使用场景中进行详细的对比测试，然后根据测试的结果，对优化后的云桌面协议进行分析。

（1）低速运动场景

低速运动场景画面变化小，元素比较单一，如图 5-1 所示。测试过程为打开预先准备好的文本，使用脚本模拟键盘输入相同的内容，以保证测试操作的一致，实验时间为两分钟。服务端和客户端均使用 pidstat 软件进行 CPU 占用率和内存占用率的监测，使用 dstat 软件监测服务端的发送带宽。为保证实验数据的准确性，每个实验均进行多组测试，在结果中去掉极端情况，最终结果为所有测试结果的平均值。

在低速运动场景中，优化前后的客户端和服务端的 CPU 占用率对比如图 5-2 和图 5-3 所示，内存占用对比如表 5-2 所示，带宽占用对比如图 5-4 所示。

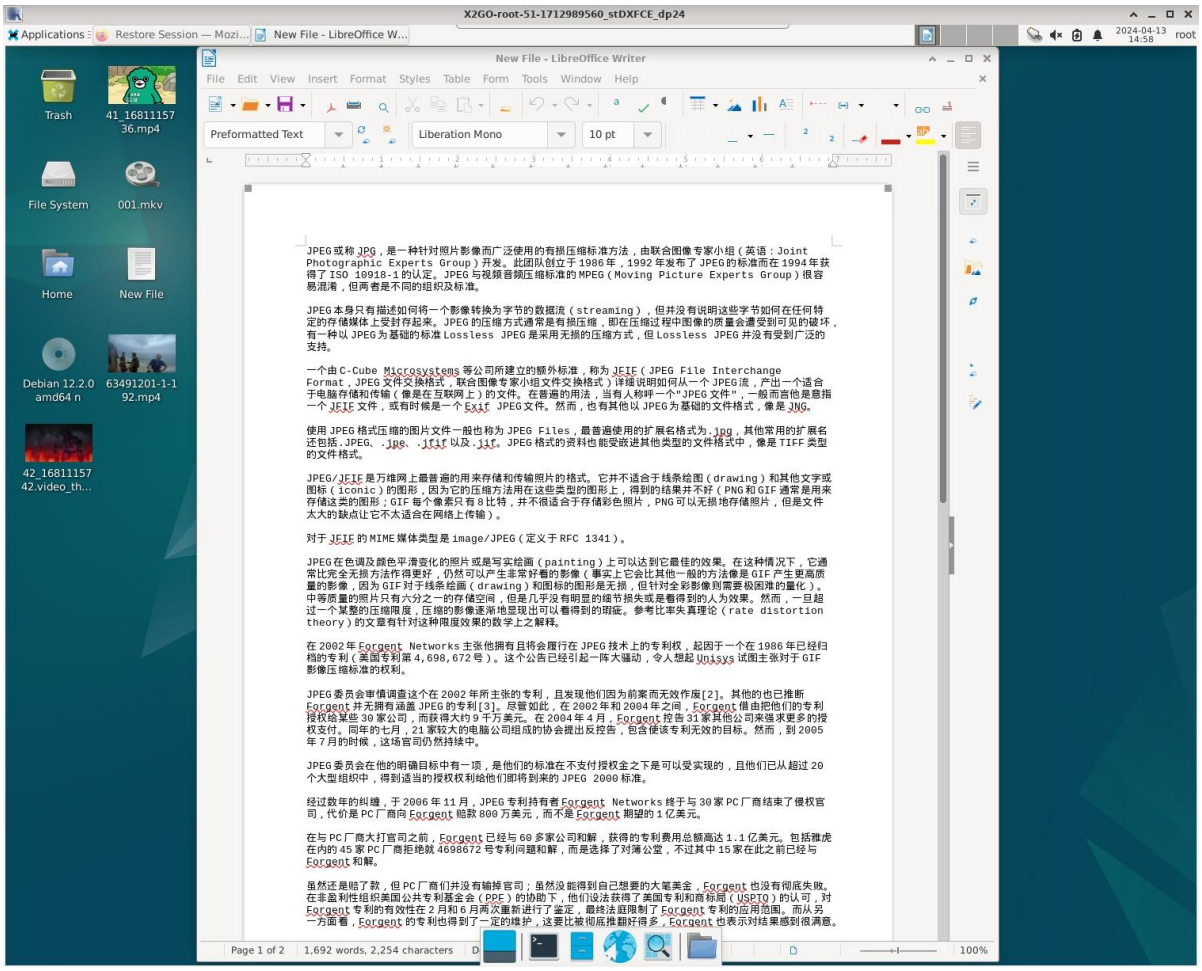


图 5-1 低速运动场景

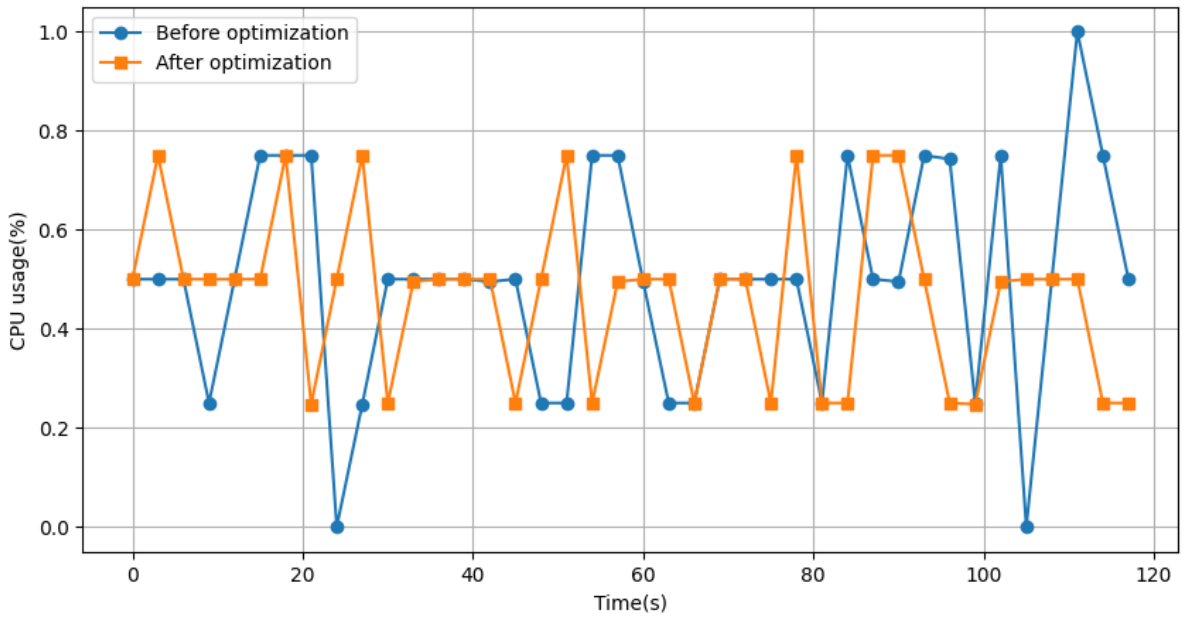


图 5-2 低速运动场景下客户端 CPU 占用率对比

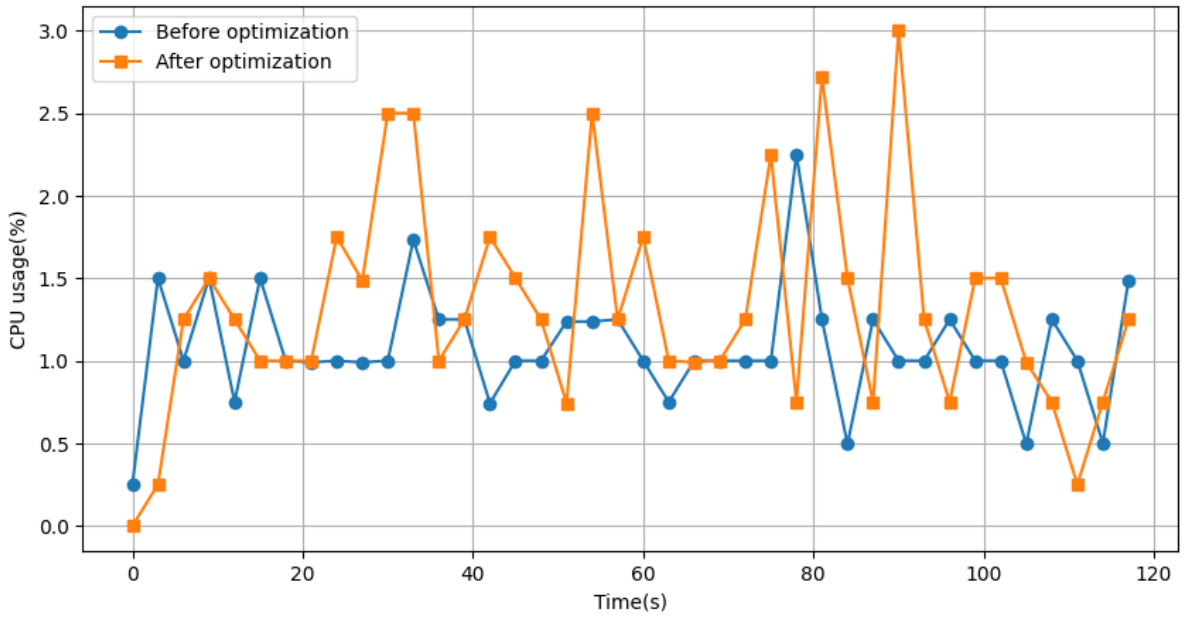


图 5-3 低速运动场景下服务端 CPU 占用率对比

为了防止 120 个数据点的对比图过于杂乱，所以作图过程中采取从第一个数据开始，每三个数据采样一次的方式作折线图，平均值的计算仍然包含所有数据。

图 5-2 和图 5-3 所示的 CPU 占用率对比中，由于低速运动场景图像简单，有大量白色背景，且图像变化很小，色彩单一，无需复杂的图像压缩处理，无论是客户端还是服务端整体 CPU 占比都很小，客户端均在 1%以下，服务端均在 3%以下。具体的平均值数据为：优化前云桌面协议客户端 0.48%，服务端 1.03%；优化后云桌面协议客户端 0.45%，服务端 1.28%。在客户端 CPU 占用率降低的原因是优化过后的云桌面协议编码后的图像数据更加紧凑，使得解码过程效率更高。在服务端 CPU 占用率提高的原因是加入了额外的编码过程，使得计算量提高了。

由于内存占用在实验过程中变化不大，所以优化前后云桌面协议的内存占用的对比仅用表格展示其平均值。优化后云桌面协议在内存占用上对比优化前无论在客户端还是服务端均有上升。这是因为优化过后的云桌面协议额外的计算和扫描过程使得中间数据增多，导致内存占用上升。

表 5-2 低速运动场景下内存占用对比

云桌面协议	内存占用（%）	
	客户端	服务端
优化前	3.23	6.91
优化后	4.57	7.39

带宽测试对比图采样方式与 CPU 占用率对比相同，在低速运动场景下优化前后带宽占用均较低，该场景下优化前后的带宽占用大体上差不多，无论优化前还是优化后，除个别峰值外都在 30kb/s 以下。但是还是可以看出优化后云桌面协议总体比优化前要略低一些，优化前后平均分别是 20.4kb/s 和 17.6kb/s，平均带宽降低了 13.73%。带宽降低的原因是优化后的云桌面协议较优化前在图像编码效率上有所提升，使得编码过后的图像数据更小，故传输这部分数据所需的带宽随之降低。

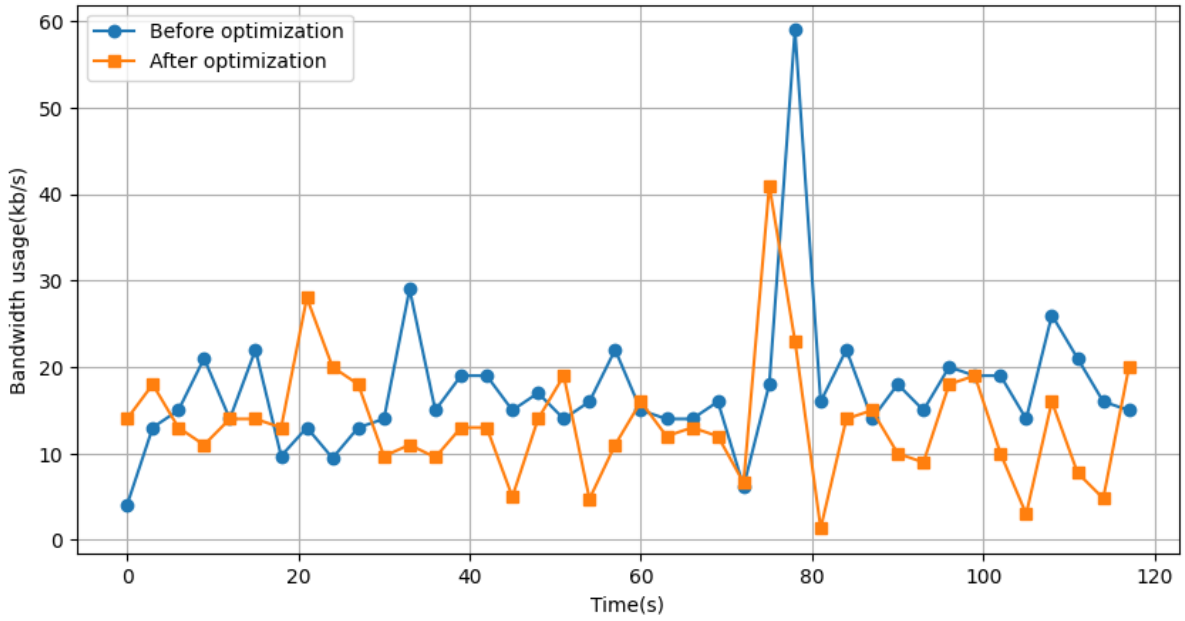


图 5-4 低速运动场景下带宽占用对比

（2）中速运动场景

中速运动场景画面相对低速运动场景要更加复杂，主要由文字和图片组成，且变化速度更快，如图 5-5 所示。测试选用百度新闻网页，过程中提前加载好网页，同样使用脚本模拟键盘输入每秒翻页一次，保证测试环境的一致。实验测试方法与低速运动环境相同。

在中速运动场景下，优化前后的客户端和服务端 CPU 占用率对比如图 5-6 和图 5-7 所示，内存占用对比如表 5-3 所示，带宽占用对比如图 5-8 所示。

在中速运动场景下，优化前后的差距开始显现出来，主要是因为中速运动场景中图像开始多了起来，色彩度也更加复杂，图像处理的过程变多了。在图 5-6 和图 5-7 所示的 CPU 占用率对比中，可以清晰的看出优化后云桌面协议相对于优化前客户端 CPU 占用率有所降低，而服务端 CPU 占用率明显升高了。具体的平均值数据为：优化前云桌面协议客户端 3.99%，服务端 6.57%；优化后云桌面协议客户端 2.96%，服务端 11.16%。

图 5-5 中速运动场景

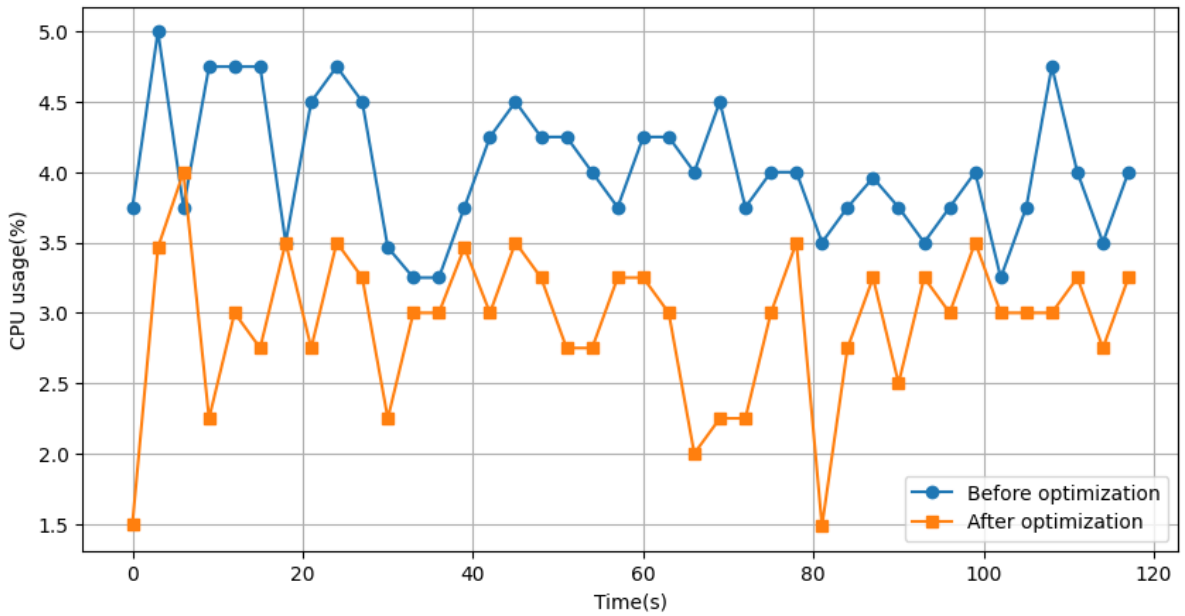


图 5-6 中速运动场景下客户端 CPU 占用率对比

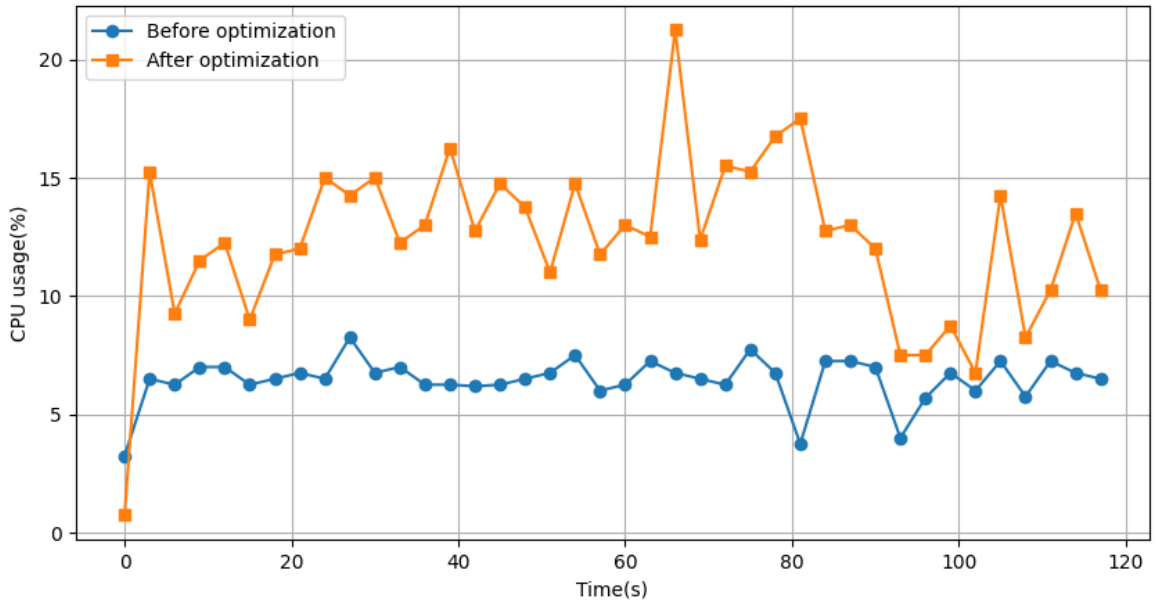


图 5-7 中速运动场景下服务端 CPU 占用率对比

与低速运动场景相同，表 5-3 表示的中速运动场景中优化后云桌面协议在内存占用上对比优化前无论在客户端还是服务端均有所上升。

表 5-3 中速运动场景下内存占用对比

云桌面协议	内存占用（%）	
	客户端	服务端
优化前	4.03	6.25
优化后	4.66	6.71

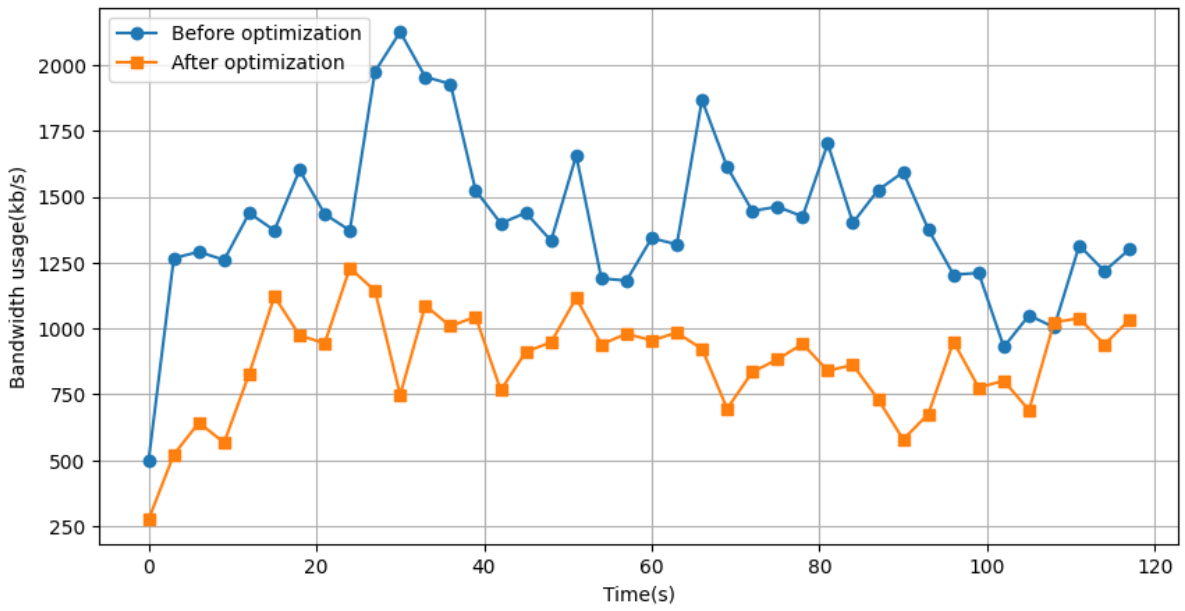


图 5-8 中速运动场景下带宽占用对比

由于优化过后云桌面协议在编码效率上的提升，图 5-8 的带宽占用均小与优化前，由平均 1353.5kb/s 降低到了平均 969kb/s，平均带宽降低了 28.4%。

（3）高速运动场景

在高速运动场景中，使用轻量化视频播放工具 MPV 全屏播放同一个 720P 的视频，如图 5-9 所示。高速运动场景中图像比中速运动场景更为复杂，画面变化更快，图像压缩处理也更多，优化前后各项实验测试指标的差距将进一步增大。

图 5-9 高速运动场景

高速运动场景下优化前后客户端和服务端的 CPU 占用率对比如图 5-10 和图 5-11 所示，内存占用对比如表 5-4 所示，带宽占用对比如图 5-12 所示。

在高速运动场景中，优化后的云桌面协议 CPU 占用率在服务端比优化前有明显升高，但是都在 25%以下，在客户端 CPU 占用率降低，与低速运动场景和中速运动场景一致。优化后的云桌面协议 CPU 占用率在服务端升高的原因是第三章和第四章中对于 JPEG 的优化方法需要额外的计算过程，所以需要更多的 CPU 资源来计算。而客户端 CPU 占用率更低的原因是优化过后的方法压缩编码后的图像数据更加紧凑，所以在解码过程中消耗的计算量更少。由于额外计算和扫描过程带来的中间数据增多，图 5-11 所示的内存占用中，优化后的云桌面协议在客户端和服务端均多于优化前，但无论客

户端还是服务端增加幅度都很小，在实际场景中，这点增加不会影响用户的使用体验。

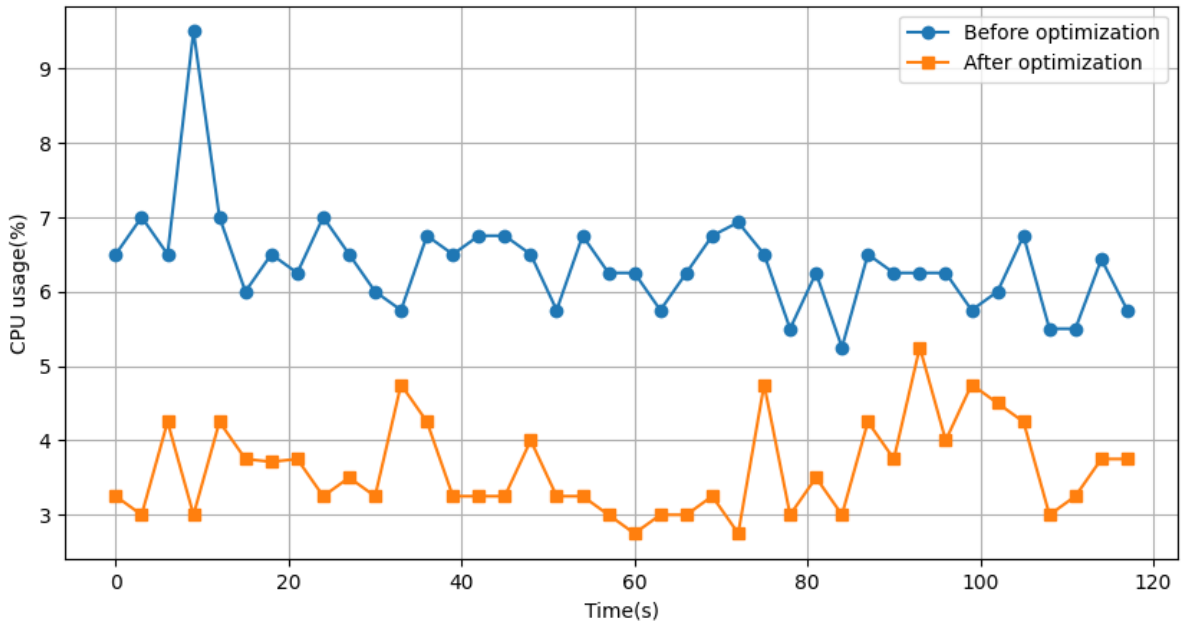


图 5-10 高速运动场景客户端 CPU 占用率对比

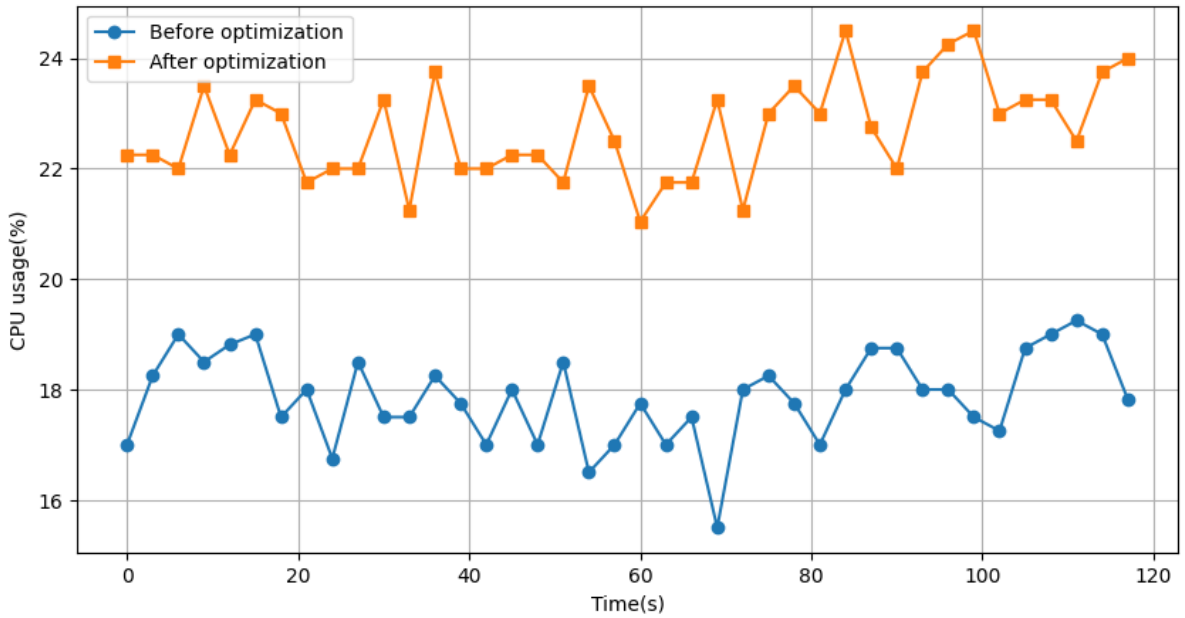


图 5-11 高速运动场景服务端 CPU 占用率对比

表 5-4 高速运动场景内存占用对比

云桌面协议	内存占用（%）	
	客户端	服务端
优化前	8.29	10.3
优化后	8.81	11.37

图 5-12 所示的带宽对比中，可以明显的看出优化后的云桌面协议的带宽占用低于优化前，平均带宽从 1713.9kb/s 降低到了 1290.7kb/s，降低了 24.69%。由此可以得出，优化过后的云桌面协议能够有效的降低用户使用过程中的网络传输带宽。

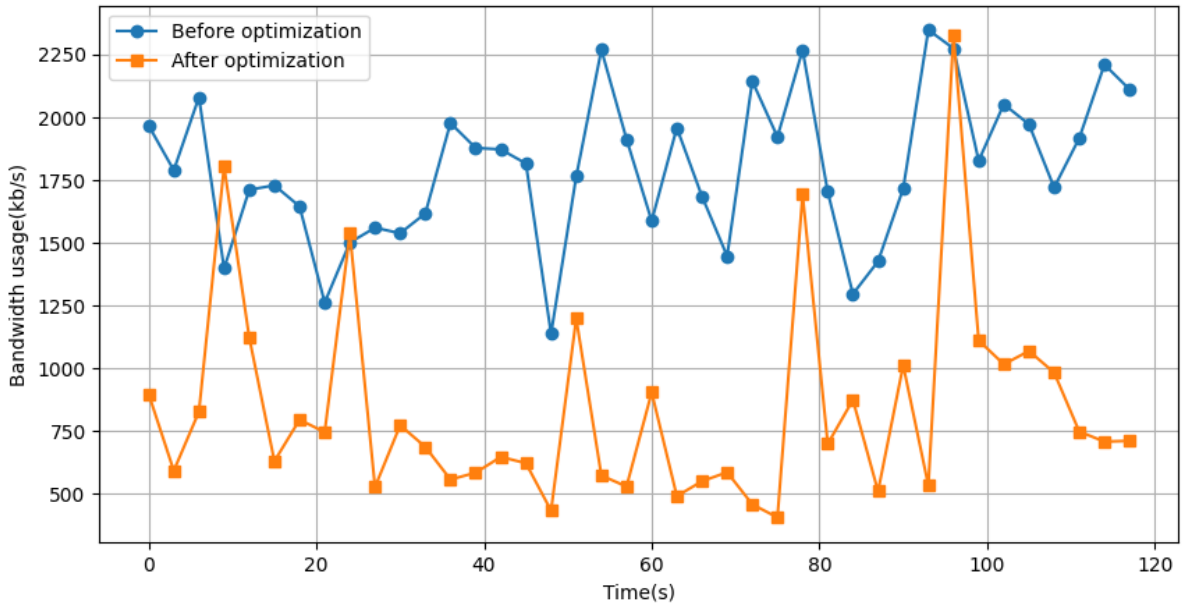


图 5-12 高速运动场景带宽占用对比

综上，经过优化的云桌面协议在服务端的 CPU 占用率有明显上升，但总体都维持在 25%以下，实际云桌面系统中强大的服务器性能还会缩小这一差距。在客户端和服务端的内存占用上略有上升，幅度很小，实际应用中不会影响用户使用体验。优化后的云桌面协议在其他方面的表现都优于优化之前的版本，在客户端的 CPU 占用率上降低了 6.25% - 42.58%（这里指的是相对于优化前的百分比，而不是 CPU 占用率的实际值），在网络带宽占用上降低了 13.73% - 28.41%，表明优化后的协议在这两方面性能上都有不小的提升。说明本文所提出的优化方法，提高了系统资源的利用率，这对于客户端设备的瘦化有着重要意义。此外，由于降低了图像压缩过程中的压缩率，减少了网络带宽的占用，使得云桌面系统更加适应低带宽环境的需求，这不仅提升了系统性能，也使得云桌面系统更加契合云计算愿景中对资源利用和网络适应性的高要求，为云桌面协议的未来发展和应用提供了有价值的参考和实践依据。

5.4 本章小结

本章系统地阐述了对云桌面传输协议优化效果的实验测试过程。首先详细介绍了实验所使用的机器环境，包括硬件配置和操作系统版本，以确保实验结果的准确性和可重现性。接下来，介绍了实验评价标准，旨在全面评估协议优化的效果，包括 CPU 占用率和内存占用和网络带宽占用。基于这些评价标准，对优化前后的云桌面协议在客户端和服务端的表现进行了深入测试。通过对比实验，详细展示了实验结果，并对数据

进行了细致的分析。实验结果证明了提出的优化措施能够有效改善云桌面协议的性能，更好地符合云计算环境对资源利用率和网络适应性的需求。此外，通过使用本文提出的优化方法，能够让云桌面协议进一步瘦化客户端和适应低带宽环境以满足特定场景中用户使用云桌面系统的需求。总体而言，本章的研究为云桌面协议的优化提供了有力的实验支持，展现了其在实际应用中的潜在价值和广阔前景。

总结与展望

本文工作总结

本文对目前云计算下云桌面系统的现状进行了分析，并针对其中图像压缩过程的不足，对当前主流云桌面协议中的主要图像压缩算法——JPEG 算法进行了两个方面的改进和优化，包括量化系数的选择和量化后 AC 系数的分割和编码。同时将这两个方法相结合，以优化当前主流云桌面协议当中原有的 JPEG 图像压缩算法。并对优化后的云桌面协议进行了对比实验，证明了优化后的协议在某些方面得到了提升，更能适应瘦客户端、低带宽环境，符合云计算的愿景。本文主要工作如下：

（1）云桌面协议是云桌面系统中的关键技术之一，云桌面协议很大程度上决定了用户的体验。目前大部分云桌面协议对带宽有着极高的要求，其原因是常见的云桌面协议，如：RDP 协议、VNC 协议、ICA 协议、SPICE 协议等在进行图像编码时大多使用 JPEG 编码，导致在一些高运动场景性能较差，表现为带宽占用急剧升高甚至出现卡顿、失帧、画面撕裂等严重影响用户体验的情况。针对上述情况，采取了网格量化 TCQ 方法来优化 JPEG 编码过程，提高了 JPEG 的编码效率，从而降低云桌面系统中图像传输的带宽需求。在多个数据集上进行了对比实验，实验的结果显示改进后的方法能够优化 JPEG 编码的压缩率。

（2）针对当前云桌面协议在高运动场景下表现不佳的问题，深入研究和分析了现有的渐进式压缩方法，在此基础上，采用一种新的扫描分割策略来优化 JPEG 的渐进式压缩技术。该策略通过对 DCT 系数按特征进行分组，并根据不同的优先级分别对其进行编码，首先传输携带图像信息较多的序列，从而有效提升了压缩效率。通过一系列精心设计的对比实验，在相关数据集上验证了该优化方案，将其性能与原始方案的实验结果进行了对比。实验结果清晰地展示了改进方案在压缩率方面有所提升。

（3）基于 X2Go Kdrive 平台对上述两种技术进行了实现，对实现前后的云桌面传输性能进行了对比测试，对实验数据进行了详细的展示和分析。实验结果表明本文提出的优化策略能够有效提升云桌面传输协议的性能，从而更好地满足云计算环境对资源利用率和网络适应性的需求。

未来工作展望

本文采用的针对瘦客户端、低带宽环境的云桌面系统中图像压缩过程的优化方法，虽然可以在带宽、内存和客户端 CPU 占用上提升云桌面系统的性能，但是服务端 CPU 占用却远远提高，虽然服务端性能远强于一般瘦客户端，但过多的 CPU 占用会影响服务器能够服务的用户数量。同时，JPEG 算法包括了采样、DCT 变换、量化和熵编码等多个过程，本文仅针对量化过程和编码前的分割过程进行了优化，同样，云桌面协议也

还有许多不足的地方，还需要进一步研究，这里提出几点未来研究的方向：

（1）由于云桌面系统是一个实时系统，在网络环境不稳定的情况下，若画面的反馈出现卡顿，会十分影响用户体验。可以结合渐进式传输的特性使用一种监测机制，服务器先传输图像的大部分信息，这部分信息仅需要少部分比特编码，再传输包含图像的纹理细节的剩余部分信息，若这个过程中网络出现卡顿，超出了指定的时间，就直接将图像的大致信息进行展现，这样用户会看到画面的大致图像，提升了用户的体验。

（2）针对 JPEG 算法，可以对其中各个部分做出优化，这里仅提出一种。熵编码的方法有很多，其中哈夫曼编码和算数编码被加入了标准 JPEG 库，而算数编码由于其版权原因和计算成本原因未被大部分应用支持，即使其有优秀的压缩效率，图像压缩中主流的熵编码仍然是哈夫曼算法。文章[76]提出了一种新的熵编码方法，它通过一种任意符号概率分布的不对称数值系统，获得了与算数编码相当的编码效率，同时保持了较低的计算复杂度。将该方法加入到 JPEG 算法中，使用它进行熵编码过程，可以优化 JPEG 图像压缩的效率，进一步优化云桌面系统的图像压缩过程。

致 谢

准备落笔写下致谢时，我才意识到自己的研究生涯即将画上一个句号。时光荏苒，三年的时间转眼已经快要迎来谢幕的时刻，很荣幸能够与西南交通大学计算机与人工智能学院计算机全体老师和同学一起经历，故在此，向陪伴我和给予我帮助关怀的计算机全体老师同学致以我最衷心的谢意。

首先，我想表达我对我的导师，戴元顺老师的深深感激。自从第一次遇见戴老师，他那真挚的人格魅力与对学术的深刻认真就深深触动了我。戴老师不仅传授了我丰富的学术知识和研究技能，还以他亲切和蔼的态度，对我的学术旅程提供了无价的建议和指导。他教我要有洞察力，勇于探索，但在学术探求中也要保持严谨。从开始决定研究主题直到完成，戴老师始终细致地指导我，关心每位学生的进展，确保对每个人都尽职尽责。我由衷地感谢戴老师对我的指导和支持。

同时，我也非常感激实验室的余盛季老师在我研究生学习期间的悉心指导。在余老师的指导下，我不仅获得了宝贵的项目实践经验，还学会了如何将理论知识应用于实践。面对任何问题，无论大小，余老师总是以无比的耐心为我解答。我从余老师那里学到了很多宝贵的知识和经验。在此，我要特别感谢余老师的耐心教导和帮助。

“天高地厚念尊恩，父母恩情似海深。”感谢我的家人们，家人的教育和支持使我能够完成研究生三年的学习并且在今后的道路上走得更高更远。家人永远是我最宝贵的财富，是我力量的源泉。

今将卒业，致感之情，不知所言。

参考文献

[1] Monika, Sangwan O P. A framework for evaluating cloud computing services using AHP and TOPSIS approaches with interval valued spherical fuzzy sets[J]. Cluster Computing, 2022, 25(6): 4383-4396.

[2] Steinder M, Whalley I, Chess D. Server virtualization in autonomic management of heterogeneous workloads[J]. ACM SIGOPS Operating Systems Review, 2008, 42(1): 94-95.

[3] 王春海. VMware 虚拟化与云计算应用案例详解[M]. 中国铁道出版社有限公司, 2021.

[4] Yan L. Development and application of desktop virtualization technology[C]//2011 IEEE 3rd International Conference on Communication Software and Networks. IEEE, 2011: 326-329.

[5] Spruijt R, van Leeuwen J, Monaghan R. Application virtualization smackdown[J]. no. December, 2013.

[6] 李承东. 云桌面远程传输协议综述[J]. 现代电信科技, 2014, 44(8): 23-26.

[7] Ruffini M, Slyne F. Moving the network to the cloud: The cloud central office revolution and its implications for the optical layer[J]. Journal of Lightwave Technology, 2019, 37(7): 1706-1716.

[8] 中国信通院发布云计算发展白皮书(2018 年) [J]. 自动化博览, 2018, (9): 3-3.

[9] Zhaldak M I, Franchuk V M, Franchuk N P. Some applications of cloud technologies in mathematical calculations[C]//Journal of Physics: Conference Series. IOP Publishing, 2021, 1840(1): 012001.

[10] Fuzi M F M, Hamid R S, Ahmad M A. Virtual desktop environment on Cloud computing platform[C]//2014 IEEE 5th Control and System Graduate Research Colloquium. IEEE, 2014: 80-84.

[11] Li F, Guo T, Li X, et al. Transportation of Service Enhancement Based on Virtualization Cloud Desktop[J]. Electronics, 2023, 12(7): 1572.

[12] Nooney L, Driscoll K, Allen K. From programming to products: Softalk magazine and the rise of the personal computer user[J]. Information & Culture, 2020, 55(2): 105-129.

[13] Ray P P. A survey of IoT cloud platforms[J]. Future Computing and Informatics Journal, 2016, 1(1-2): 35-46.

[14] Tan K J, Gong J W, Wu B T, et al. A remote thin client system for real time multimedia streaming over VNC[C]//2010 IEEE International Conference on Multimedia and Expo. IEEE, 2010: 992-997.

[15] Magaña E, Sesma I, Morato D, et al. Remote access protocols for Desktop-as-a-Service solutions[J]. PloS one, 2019, 14(1): e0207512.

[16] Hickson I. A vocabulary and associated APIs for HTML and XHTML[J]. W3C Editor's Drift, World Wide Web Consortium (W3C), 2014.

[17] Liu X. Research and applications of cloud desktop[C]//2014 International Conference on Computer, Communications and Information Technology (CCIT 2014). Atlantis Press, 2014: 14-17.

[18] Nakhai P H, Anuar N B. Performance evaluation of virtual desktop operating systems in virtual desktop infrastructure[C]//2017 IEEE Conference on Application, Information and

Network Security (AINS). IEEE, 2017: 105-110.
[19] Nehra S, Kumar C R S. Enterprise virtual desktop infrastructure architecture on OpenStack cloud with lightweight directory access protocol[C]//2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO). IEEE, 2020: 1050-1055.
[20] Casas P, Seufert M, Egger S, et al. Quality of experience in remote virtual desktop services[C]//2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013). IEEE, 2013: 1352-1357.
[21] Fylaktopoulos G, Goumas G, Skolarikis M, et al. An overview of platforms for cloud based development[J]. SpringerPlus, 2016, 5: 1-13.
[22] Zhang K. Application of desktop computing technology based on cloud computing[J]. International Journal of Information Technologies and Systems Approach (IJITSA), 2021, 14(2): 1-19.
[23] Wilieyanto E. Performance analysis of vdesktop using PCoIP accelerator vs vSGA-based on VMware environment—A case study at UKRIDA University[C]//2018 International Conference on Information and Communications Technology (ICOIACT). IEEE, 2018: 705-708.
[24] Montoro M. Remote desktop protocol, the good the bad and the ugly[J]. Oxid. it, 2005.
[25] Lan Y, Xu H. Research on technology of desktop virtualization based on SPICE protocol and its improvement solutions[J]. Frontiers of Computer Science, 2014, 8: 885-892.
[26] Richardson T, Stafford-Fraser Q, Wood K R, et al. Virtual network computing[J]. IEEE Internet Computing, 1998, 2(1): 33-38.
[27] Vuza D T, Frosch R. RFID readers for the HDX protocol-a designer's perspective[J]. Current Trends and Challenges in RFID, 2011: 229-254.
[28] Richardson T, Wood K R. The rfb protocol[J]. ORL, Cambridge, January, 1998.
[29] Emerson S, Emerson K, Fedorczyk J. Computer workstation ergonomics: Current evidence for evaluation, corrections, and recommendations for remote evaluation[J]. Journal of Hand Therapy, 2021, 34(2): 166-178.
[30] Casanova L, Kristianto E. Comparing RDP and PcoIP protocols for desktop virtualization inVMware enviroment[C]//2017 5th International Conference on Cyber and IT Service Management (CITSM). IEEE, 2017: 1-4.
[31] Langone J, Leibovici A. VMware View 5 Desktop Virtualization Solutions[M]. Packt Publishing, 2012.
[32] Chen H, Pu F, Yang R, et al. Rdp-net: Region detail preserving network for change detection[J]. IEEE Transactions on Geoscience and Remote Sensing, 2022, 60: 1-10.
[33] Zhan H. Comparison and Performance Analysis of EVENODD and RDP Fault-Tolerant Coding Methods[J]. Highlights in Science, Engineering and Technology, 2024, 87: 37-44.
[34] Muhammad Faseeh Qureshi N, Shin D R. RDP: A storage-tier-aware Robust Data Placement strategy for Hadoop in a Cloud-based Heterogeneous Environment[J]. KSII Transactions on Internet and Information Systems (TIIS), 2016, 10(9): 4063-4086.
[35] Kouril J, Lambertova P. Performance analysis and comparison of virtualization protocols, RDP and PCoIP[C]//Proceedings of the 14th WSEAS international conference on Computers. 2010: 782-787.
[36] Hagström M. Remote desktop protocols: A comparison of Spice, NX and VNC[J]. 2012.

[37] Yang L, Li C, You R, et al. A keystroke-based continuous user authentication in virtual desktop infrastructure[C]//2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS). IEEE, 2021: 753-758.
[38] Jung J K, Jung S M, Kim T K, et al. Implementation of enhanced android SPICE protocol for mobile cloud[C]//Computational Science and Its Applications–ICCSA 2013: 13th International Conference, Ho Chi Minh City, Vietnam, June 24-27, 2013, Proceedings, Part I 13. Springer Berlin Heidelberg, 2013: 372-381.
[39] 徐浩, 兰雨晴. 基于 SPICE 协议的桌面虚拟化技术研究与改进方案[J]. 计算机工程与科学, 2013, 35(12): 20-25.
[40] Ibrahim A A Z A, Kliazovich D, Bouvry P, et al. Virtual desktop infrastructures: architecture, survey and green aspects proof of concept[C]//2016 Seventh International Green and Sustainable Computing Conference (IGSC). IEEE, 2016: 1-8.
[41] Citrix HDX technology White Paper[EB/OL], https://docs.citrix.com/en-us/citrix-virtual-apps-desktops/1912-ltsr/technical-overview/hdx.html. 2013-05.
[42] Stolerman A. RFC 6143: The Remote Framebuffer (RFB) Protocol Analysis[J]. 2013.
[43] Fok A C F, Lécroart B, Chan É, et al. An adaptive approach to optimize thin client protocols[C]//2010 Future Network & Mobile Summit. IEEE, 2010: 1-9.
[44] Li W, Wang B, Yu J, et al. The optimization of Transparent-Desktop service mechanism based on SPICE[J]. Concurrency and Computation: Practice and Experience, 2016, 28(18): 4543-4556.
[45] Simoens P, Praet P, Vankeirsbilck B, et al. Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices[C]//2008 Australasian Telecommunication Networks and Applications Conference. IEEE, 2008: 391-396.
[46] 杨飞, 朱志祥, 梁小江. 基于 SPICE 协议的视频性能优化与改进[J]. 计算机技术与发展, 2016, 26(12): 73-76.
[47] Lai A M, Nieh J. On the performance of wide-area thin-client computing[J]. ACM Transactions on Computer Systems (TOCS), 2006, 24(2): 175-209.
[48] Lan Y, Xu H. Research on technology of desktop virtualization based on SPICE protocol and its improvement solutions[J]. Frontiers of Computer Science, 2014, 8: 885-892.
[49] Gallidabino A, Pautasso C. The Liquid WebWorker API for Horizontal Offloading of Stateless Computations[J]. Journal of Web Engineering, 2018, 17(6–7): 405-448.
[50] Ramirez R. Behavioral Characterization of Attacks on the Remote Desktop Protocol[J]. 2022.
[51] de la Torre L. An Implementation of a Web Laboratory Converting Off-Line Experiments into Remotely Accessible Experiments[C]//Sinteza 2022-International Scientific Conference on Information Technology and Data Related Research. Singidunum University, 2022: 169-175.
[52] Schmid A, Wimmer R. Measuring the Latency of Graphics Frameworks on X11-Based Systems[C]//Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems. 2023: 1-7.
[53] Baiocchi G. Modern R workflow and tools for microeconometric data analysis[M]//Handbook of Research Methods and Applications in Empirical Microeconomics. Edward Elgar Publishing, 2021: 518-563.

[54] Wallace G K. The JPEG still picture compression standard[J]. Communications of the ACM, 1991, 34(4): 30-44.
[55] Taubman D S, Marcellin M W. JPEG2000: Standard for interactive imaging[J]. Proceedings of the IEEE, 2002, 90(8): 1336-1357.
[56] Man H, Docef A, Kossentini F. Performance analysis of the JPEG 2000 image coding standard[J]. Multimedia Tools and Applications, 2005, 26: 27-57.
[57] Goyal V K. Theoretical foundations of transform coding[J]. IEEE Signal Processing Magazine, 2001, 18(5): 9-21.
[58] Sullivan G J, Ohm J-R, Han W-J, et al. Overview of the high efficiency video coding (HEVC) standard[J]. IEEE Transactions on circuits and systems for video technology, 2012, 22(12): 1649-1668.
[59] Sullivan G J, Sun S. On dead-zone plus uniform threshold scalar quantization[C]//Visual Communications and Image Processing 2005. SPIE, 2005, 5960: 1041-1052.
[60] Tsubota K, Aizawa K. Comprehensive comparisons of uniform quantizers for deep image compression[C]//2021 IEEE International Conference on Image Processing (ICIP). IEEE, 2021: 2089-2093.
[61] Sullivan G J, Wiegand T. Rate-distortion optimization for video compression[J]. IEEE signal processing magazine, 1998, 15(6): 74-90.
[62] Wang H, Kwong S. Rate-distortion optimization of rate control for H. 264 with adaptive initial quantization parameter determination[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(1): 140-144.
[63] Yuan H, Wang Q, Liu Q, et al. Hybrid distortion-based rate-distortion optimization and rate control for H. 265/HEVC[J]. IEEE Transactions on Consumer Electronics, 2021, 67(2): 97-106.
[64] Bross B, Wang Y-K, Ye Y, et al. Overview of the versatile video coding (VVC) standard and its applications[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2021, 31(10): 3736-3764.
[65] Gray R. Vector quantization[J]. IEEE Assp Magazine, 1984, 1(2): 4-29.
[66] Kuo W-C, Wang C-C, Hou H-C. Signed digit data hiding scheme[J]. Information Processing Letters, 2016, 116(2): 183-191.
[67] Xie X-Z, Chang C-C, Hu Y-C. An adaptive reversible data hiding scheme based on prediction error histogram shifting by exploiting signed-digit representation[J]. Multimedia Tools and Applications, 2020, 79: 24329-24346.
[68] Marcellin M W, Fischer T R. Trellis coded quantization of memoryless and Gauss-Markov sources[J]. IEEE transactions on communications, 1990, 38(1): 82-93.
[69] Fischer T R, Marcellin M W, Wang M. Trellis-coded vector quantization[J]. IEEE Transactions on Information Theory, 1991, 37(6): 1551-1566.
[70] Fischer T R, Wang M. Entropy-constrained trellis-coded quantization[J]. IEEE Transactions on Information Theory, 1992, 38(2): 415-426.
[71] Forney G D. The viterbi algorithm[J]. Proceedings of the IEEE, 1973, 61(3): 268-278.
[72] Joshi R L, Crump V J, Fischer T R. Image subband coding using arithmetic coded trellis coded quantization[J]. IEEE Transactions on Circuits and Systems for Video Technology, 1995, 5(6): 515-523.
[73] Wang Z, Bovik A C, Sheikh H R, et al. Image quality assessment: from error visibility to

structural similarity[J]. IEEE transactions on image processing, 2004, 13(4): 600-612.
[74] Sheikh H R, Sabir M F, Bovik A C. A statistical evaluation of recent full reference image quality assessment algorithms[J]. IEEE Transactions on image processing, 2006, 15(11): 3440-3451.
[75] THE NEW TEST IMAGES[EB/OL], http://www.imagecompression.info/test_images. 2008.
[76] Duda J, Tahboub K, Gadgil N J, et al. The use of asymmetric numeral systems as an accurate replacement for Huffman coding[C]//2015 Picture Coding Symposium (PCS). IEEE, 2015: 65-69.

攻读硕士学位期间发表论文及科研成果

[1] Tan H. Trellis Coded Quantization for JPEG Compression[C]//2023 9th International Symposium on System Security, Safety, and Reliability (ISSSR). IEEE, 2023: 251-255.

[2] 中国国家铁路集团有限公司科技研究开发计划重点课题：基于知识图谱的铁路信息系统故障分析和预测关键技术研究（N2022S006）.