华南理工大学

South China University of Technology

硕士学位论文

基于 QEMU-KVM 的办公桌面云系统的设计与实现

作 者 姓 名	杨亚伟
学 科 专 业	软件工程
指 导 教 师	王振宇 教授
所 在 学 院	软件学院
论文提交日期	2019 年 11 月 1 日

Design and Implementation of the Office Desktop Cloud System based on QEMU-KVM

A Dissertation Submitted for the Degree of Master

Candidate：Yang Yawei

Supervisor：Prof. Wang Zhenyu

South China University of Technology

Guangzhou, China

分类号：TP3 **学校代号：10561**

学　号：201620134711

华南理工大学硕士学位论文

基于 QEMU-KVM 的办公桌面云系统的设计与实现

作者姓名：杨亚伟　　指导教师姓名、职称：王振宇 教授

申请学位级别：硕士　　学科专业名称：软件工程

研究方向：云计算与软件服务

论文提交日期：2019 年 11 月 1 日　　论文答辩日期：2019 年 12 月 2 日

学位授予单位：华南理工大学　　学位授予日期：　　年　　月　　日

答辩委员会成员：

主席：　张平健

委员：　王振宇、谭明奎、吴庆耀、朱映波

华南理工大学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：杨亚伟　　日期：2019 年 12 月 3 日

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，即：研究生在校攻读学位期间论文工作的知识产权单位属华南理工大学。学校有权保存并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅（除在保密期内的保密论文外）；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。本人电子文档的内容和纸质论文的内容相一致。

本学位论文属于：

□保密（校保密委员会审定为涉密学位论文时间：___年__月__日），于___年__月__日解密后适用本授权书。

☑不保密，同意在校园网上发布，供校内师生和与学校有共享协议的单位浏览；同意将本人学位论文编入有关数据库进行检索，传播学位论文的全部或部分内容。

（请在以上相应方框内打“√”）

作者签名：杨亚伟　　日期：2019年12月3日

指导教师签名：　　日期：2019年12月3日

作者联系电话：　　电子邮箱：

联系地址（含邮编）：

摘 要

信息技术飞速发展的当下，越来越多的企业为了解决信息化发展需要，购置了大量的电脑终端设备。随着电脑的大量增加，同时也暴露了诸多问题，维护管理人员需要做大量的重复工作，管理成本提高，用户需求得不到及时响应，数据过于分散安全性得不到保障等。如何对桌面统一化管理、数据集中存储已经成为企业迫切解决任务之一，基于 VDI 架构的桌面云技术出现很好的解决这些问题。桌面云，通过服务器虚拟化技术将用户桌面云端化，所有的用户虚拟桌面环境以及数据集中存储在服务器上，对服务器上的虚拟桌面环境集中运维管理，利用服务器资源进行计算，终端与桌面环境解耦和，用户通过网络使用自己的虚拟桌面。管理人员只需对服务器上的虚拟桌面环境管理维护极大的缩减了运维时间，提高了办公效率，节省企业成本，数据集中存储在服务器上，数据安全性也得到了更好的保障。本文主要完成的工作有：

首先对桌面云关键技术的研究。研究了 QEMU-KVM 虚拟化技术，重点分析了 QEMU-KVM 对 CPU 虚拟化、内存虚拟化以及 I/O 设备虚拟化实现的工作原理。研究了开源 SPICE 桌面传输协议及其实现原理，分析了 SPICE 协议不足之处，总结了影响桌面传输协议性能的关键因素，之后研究了 RDP 桌面传输协议数据传输工作原理。

在完成以上理论技术研究的基础上，设计实现了针对企业办公的桌面云系统，该系统底层虚拟化技术采用的是 QEMU-KVM 虚拟化技术、桌面传输协议采用 SPICE 传输协议与 RDP 桌面传输协议相结合方式实现。

通过调研详细分析了系统功能需求与非功能需求，之后进行了系统的整体架构设计、系统整体的功能模块设计、系统的类图设计以及系统数据库设计，实现了企业信息管理模块、监控模块、镜像管理模块、网络管理模块、个人功能模块以及 RDP、SPICE 登录虚拟机功能，最后进行系统功能与性能测试，并对测试结果分析。测试结果表明该系统能够提高企业办公效率。

关键字：桌面虚拟化；RDP；SPICE；QEMU-KVM

Abstract

With the rapid development of information technology, more and more enterprises have purchased a large number of computer terminal equipment in order to solve the needs of information development. With the increase of computers and the exposure of many problems, maintenance managers need to do a lot of repetitive work, management costs are increased, user needs are not responded in time, data is too dispersed, and security is not guaranteed. How to unified management of desktops and centralized storage of data has become one of the urgent tasks for enterprises**.** Desktop cloud technology based on VDI architecture has solved these problems well. The desktop cloud uses the server virtualization technology to cloud the user desktops. All the user virtual desktop environments and data are stored centrally on the server. The virtual desktop environment on the server is centrally operated and managed, and the server resources are used for computing, and decoupling the terminal and the desktop environment. Users use their own virtual desktops over the network. Management personnel only need to manage and maintain the virtual desktop environment on the server, greatly reduced operation and maintenance time, improve the office efficiency, save the enterprise cost, and store the data centrally on the server, and the data security is better protected. The major work includes:

First of all, research on key technologies of desktop cloud. Research the QEMU-KVM virtualization technology, the focus of QEMU-KVM on CPU virtualization, memory virtualization and I / O device virtualization implementation works. Research on open source SPICE desktop transfer protocol and its implementation principle, analysis of the inadequacies of the SPICE protocol, summarizes the key factors affecting the performance of desktop transport protocols, after studying the working principle of RDP desktop transmission protocol data transmission.

On the basis of completing the above theoretical and technical research, design a desktop cloud system for enterprise office. The underlying virtualization technology of the system adopts QEMU-KVM virtualization technology, and the desktop transmission protocol is implemented by combining SPICE transmission protocol and RDP desktop transmission protocol.

Detail analysis of system functional requirements and non-functional requirements through research, after that design the overall architecture of the system, the overall functional modules, the system class diagram, and system database. To achieve the enterprise information management module, control module, image management module, network

management module, individual functional modules and features such as RDP, SPICE login virtual machine. Finally, the system function and performance test are carried out, and analyze the test results. Test results show that the system can improve business efficiency.

Key words：Desktop Virtualization；RDP；SPICE；QEMU-KVM

目 录

第一章 绪论

1.1 研究背景与意义

在大数据以及云计算技术日益成熟的当下，各种相关的产品层出不穷。各行各业为了信息化发展需要，也都购置了大量的台式机。人们的生活、工作和学习与电脑紧密相关。随着企业电脑增多，在原有的 IT 架构下，给企业设备运维管理人员带来了诸多不便。比如，为了满足员工个性化桌面需求，为员工配置的电脑版本多样化，员工也会在自己的电脑装各种系统以及不同软件，这就为管理员无形中增加了大量维护工作。管理员也无法对所用的用户上机操作做监控，一些员工的不安全上网，导致电脑中毒，企业内部都是局域网环境，有可能造成大量的内部电脑感染病毒以及电脑数据被损坏等安全事故。对于批量购买的新设备安装、部署、配置网络、安装不同部门所需的软件，到最后用户能够正常使用也会消耗大量时间，同时也浪费了员工大量的工作时间。用户使用过程中电脑死机，存储在本地电脑上的数据有可能无法恢复，数据的安全性得不到保障[1]。企业为了节约能耗，下班后除了服务器运行外，也会要求员工关闭电脑，有时员工想远程办公也不可能，必须等到第二天上班必须回到自己的办公区开启自己的电脑工作，换台电脑就不能工作，其他人也不能使用自己的电脑，这就导致不能更灵活的使用终端设备，进行移动办公。员工可以随意的拷贝终端设备上的企业数据，对企业信息数据安全也造成的威胁。服务器资源利用率并不高并且服务器能耗大，造成资源的浪费[2]。基于以上问题，企业迫切的需要一种更好的 IT 架构来解决这些问题。而桌面虚拟化技术的出现很好地解决了当前企业面临的问题。

桌面虚拟化技术是虚拟化中的一个比较成熟方案，能充分利用共享资源满足实际的应用需求，同时也是整合、简化硬件以及软件架构管理工具[3]。桌面虚拟化技术是将桌面环境以及相关的应用程序与底层的硬件架构解耦合的技术。用户可以使用多种设备通过网络就能实现随时随地的访问他们的虚拟桌面[4]。VDI(Virtual Desktop Infrastructure)虚拟桌面架构是目前桌面虚拟化系统主流的、应用最广泛架构和解决方案。绝大多数云计算企业也都开发了基于 VDI 架构桌面虚拟化产品，如 VMware、Citrix、微软、阿里、腾讯、华为、深信服等。VDI 架构[5]是将用户的桌面环境运行在远程服务器，用户的操作系统、使用的程序以及用户的数据全部存储在服务器上进行管理，依托服务器虚拟化

技术创建资源池，为上层的虚拟桌面提供资源服务，用户只需通过远程桌面协议就能连接使用自己的桌面环境，实现了让桌面环境与硬件解耦和、集中化管理、保护数据及降低成本，让用户的交互体验达到本地电脑桌面[6]。桌面虚拟化让管理人员能够整合企业所有的资源统一管理以及更好的响应企业中不断变化的 IT 需求。

桌面虚拟化方案实现了原有桌面环境与底层硬件的隔离，用户也不需要在固定的办公地点办公，本地终端只是为了显示桌面图像，只要能访问网络的终端设备用户就能使用自己的桌面环境办公，这样就让用户使用自己的桌面环境更加便捷，同时也达到了移动办公的目的。用户的桌面环境、应用及数据都是存储在远程服务器上，这样管理员就能对用户桌面统一管理。用户的桌面环境是运行在远程服务器端，本地终端设备出现故障，不会影响到用户的桌面环境，只需换一台终端设备就能使用自己远程虚拟桌面，数据也不会丢失。用户的虚拟桌面环境出现中毒等问题，管理员直接对用户使用虚拟机删除，重新启动新的虚拟机给用户使用，对本地终端也不会造成影响，也不会出现企业内大量的电脑中毒。运行在服务器上的虚拟机彼此隔离，一台虚拟机故障不会影响到其他虚拟机的使用。管理员通过为不同部门创建模板镜像，用户所使用的为模板镜像[7]的派生镜像，部门所使用的虚拟机如果需要系统升级、安装新的应用软件，管理员只需对模板镜像做更新操作，就能实现对一个部门所有的虚拟机镜像升级维护，不仅提高了管理员工作效率也节省了管理成本。通过对服务器虚拟化，在服务器上同时运行多台虚拟机，这些虚拟机共享服务器资源，从而也提高了企业资源的利用率，也整合了服务器资源[8]。

综上所示，对于企业来说，很有必要采用桌面虚拟化方案来解决企业目前 IT 架构中存在的问题，这样不仅能够降低管理员的工作量，更加灵活、高效的管理资源，方便、快捷的管理用户桌面，还能对用户桌面使用权限进行限制，提高资源利用率，节省企业成本，还能够让用户随时随地的使用自己的办公桌面，提高员工办公效率，延长终端设备的使用年限，弥补老旧设备计算能力差、存储小的不足，数据资源集中统一存储管理，提高了企业数据的安全性，降低企业功耗[9]。本文详细的研究了桌面虚拟化技术架构以及虚拟化具体实现过程原理以及桌面传输协议，在研究的基础之上，采用开源的 QEMU-KVM 虚拟化技术[10]、SPICE 协议、RDP 桌面传输协议[11]，设计开发一套针对企业办公的桌面云系统，具有研究意义以及一定的应用实践意义。

1.2 国内外研究现状

1.2.1 国外研究现状

虚拟化技术的快速发展，而作为虚拟化技术的一个重要应用桌面虚拟化也得到了很好的推广与应用，很多企业都参与其中，在国外主要是 Citrix、VMware 以及微软三大企业也推出了相当成熟的桌面虚拟化解决方案。

VMware 公司成立于 1998 年，主要是提供云计算和硬件虚拟化服务，在桌面虚拟化市场占据较高的份额，推出的最新桌面和应用交付平台为 JMP(Just-in-Time-Management Platform)[12]。JMP 技术实质上是把 VMware 原有的三种技术整合在了一起，三种技术分别是即时克隆(Instant Clone)技术、应用快速交付(App Volumes)技术及用户环境管理(User Environment Management) 技术。通过容器分发虚拟机中的桌面应用以及快配置用户的个性化桌面，只需几秒就能生成用户虚拟机，消耗更少的内存，节省更多的磁盘资源，实现了应用与桌面环境解耦和。同时也推出了全新的远程桌面协议 Blast Extreme 协议[13]，并且与原先与 Tcradici 公司合作开发的 PCoIP 协议[14]相比，Blast Extreme 协议对网络的支持更加友好，占用更少的带宽，更短的响应延迟，降低 CPU 负载。而且 Blast Extreme 协议也对移动设备支持，还能降低功耗延长了电池使用时长，还能更加充分的使用 GPU 资源加速处理视频流的压缩与编码工作，能够提高数据的传输效率[15] 。

Citrix 是一家相当著名虚拟化方案提供商，实力雄厚，虚拟化产品相当完善。在桌面虚拟化领域也推出了新一代的产品 Citrix Virtual Desktops 采用 FlexCast Management Architecture 体系架构，该架构也可以实现与 Citrix Virtual Apps 共享。Citrix Virtual Desktops 使用的自研高性能的 ICA(Independent Computing Architecture)独立计算架构协议，该协议不仅传输速度快，而且支持扩展，使用广域网也有很好的效果，也能够动态的调整适应网络的变化，为了支持更高清晰度的用户体验，Citrix 推出了 HDX 技术用来增强远程桌面性能[16]，HDX 技术一方面通过对传输数据的分析采用最优的数据压缩技术及硬件资源 GPU、CPU 使用率最优搭配，另一方面对传输的重复数据进行删除，让用户体验达到本地效果。

微软公司推出的桌面虚拟化方案为 Hyper-V，采用自研的 RDP(Remote Desktop

Protocol)传输协议对用户的虚拟桌面传输使用，同时也支持剪贴板、U 盘等可插拔外设的支持，也能使用本地磁盘、打印机[17]。为了满足用户对 GPU 资源的需求开发了 RemoteFX 技术提供 GPU 虚拟化支持，有了更好的图形、图像处理效果，让用户在使用游戏软件或 3D 制图应用也有了更好的体验。

除了上述三家知名的企业外，还有其他的一些公司也推出了自己的桌面虚拟化产品，如惠普开发的工程桌面云，底层采用开源 KVM 虚拟化技术架构，硬件方面依托自己公司的工作站以及存储等产品。戴尔公司推出的易安信桌面虚拟化，采用 VMware Horizon 和超融合技术，利用高可用、压缩去重等技术保障用户业务连续性，提供多种端到端解决方案。IBM 公司推出的 VMware Horizon on IBM Cloud 产品，桌面传输协议采用 PCOIP 和 Blast extreme，用户的虚拟桌面环境由 IBM 进行管理。NEC、WYSE、neoware、nComputing、Atlantis 等公司也都有推出了桌面虚拟化解决方案。

1.2.2 国内研究现状

国内在桌面虚拟化起步虽然滞后于国外，但是近几年来国内的发展越来越热，有更多的企业与高校科研机构也参与其中共同推进国内桌面虚拟化的发展，如上海交通大学、中国人民大学、电子科技大学、华南理工大学等。绝大多数企业发布的产品在开源的虚拟化技术基础上进行二次开发。

深信服公司推出的桌面虚拟化产品 aDesk[18]，通过对服务器、桌面、存储及 GPU 虚拟化的整合，构建的一种新型融合架构，通过 SRAP 协议传输桌面数据，实现了一站式端到端的解决方案。很好的外设兼容，用户体验效果好，安全性高，部署快捷，通过 web 管理平台，多种终端设备的支持。

和信创天公司推出的桌面虚拟化产品 VENGD[19]，基于 NGD(Next Generation Desktop)架构，该架构融合了 VDI、IDV(Intelligent Desktop Virtualization)的架构优势，很好的利用了终端设备与服务器资源。对网络的依赖低，用户体验效果好。

华为的桌面虚拟化解决方案 FusionAccess[20]，采用自研的 HDP(Huawei Desktop Protocol)协议传输桌面数据，在文字、图像及视频处理上有更好的效果，消耗网络带宽低。更高的安全性，应用的业务场景更广泛，运维部署更简便。

阿里的桌面虚拟化产品，云桌面是构建在阿里云飞天平台基础上，存储采用自研盘

古分布式系统，使用 ACDP 阿里云桌面显示协议进行桌面数据传输，连接网关采用负载均衡策略，保障用户使用体验，更全面的管控功能，便于运维管理。对用户提供 GPU 支持[21]。

联想公司桌面虚拟化产品为 LVD，是基于超融合的桌面虚拟化解决方案。支持多种外设，集中管控，良好的扩展性，稳定可靠，快速部署，桌面体验好，便捷的运维操作。提供 VDI 以及 IDV 架构的支持。

锐捷公司推出了云桌面产品，融合了 VDI 与 IDV 架构，提供端到端的解决方案。采用 OS Layering 系统分层技术配置虚拟桌面，采用 Power on Speed up 技术更快捷的启动 IDV，采用 Ruijie Peripheral Management 更简单的管理外设。

升腾咨询公司桌面云解决方案，采用自研 Xred 虚拟桌面传输协议保证更好的用户体验以及自研桌面链式克隆技术更快速的桌面管理，首个全国产桌面虚拟化方案，在国内桌面云市场占有很高的份额。

1.3 本文研究内容

通过调研工作了解企业 IT 架构存在的问题，并对用户需求具体分析，分析市场上主流的桌面虚拟化产品功能与优势，依托实验室原有桌面虚拟化技术成果，深入研究开源 QEMU-KVM 虚拟化技术架构及对硬件设备虚拟化的具体实现原理，开源的 SPICE 传输协议原理及 RDP 传输协议研究，实现了一种基于 QEMU-KVM 的混合桌面传输协议办公桌面云系统。本文主要研究内容如下：

1. 项目启动前期的调研工作，项目需求分析。
2. 开源虚拟化技术 QEMU-KVM 架构及对硬件设备虚拟化实现原理研究。
3. SPICE 传输协议架构原理研究并对协议不足之处原因分析。
4. RDP 传输协议协议栈架构原理、连接实现过程研究。
5. 办公桌面云系统整体架构设计，系统功能模块的设计与实现，数据库设计，RDP、SPICE 协议连接虚拟机功能实现。
6. 对办公桌面云系统进行测试并对测试结果进行分析。

1.4 论文组织结构

本文总共分为六个章节，各章主要内容如下：

第一章：绪论。论述本文的研究背景与研究意义，介绍了国内外桌面虚拟化研究的现状，最后对本文主要研究内容进行说明。

第二章：虚拟化技术和概念。引入了虚拟化技术分类，桌面虚拟化技术架构，桌面传输协议概念。

第三章：桌面云系统核心技术研究。主要是对实现办公桌面云系统所用到的核心技术详细的分析，包括 QEMU-KVM 虚拟化相关的技术、RDP 桌面传输协议以及 SPICE 传输协议，并分析了为什么选用两种桌面传输协议相结合实现方式。

第四章：办公桌面云系统的需求分析、设计与实现。主要介绍了项目功能需求及非功能需求分析，系统整体架构设计，数据库设计，混合桌面传输协议使用的设计与实现，办公桌面云系统各功能模块的设计与实现。

第五章：办公桌面云系统的测试。从功能测试与性能测试两个方面对系统检验，最后对性能测试结果进行分析。

第六章：总结和展望。对本文所研究内容的总结，以及今后虚拟化研究的规划。

第二章 虚拟化技术和概念

云计算已成为当下最热门的领域之一，越来越多的企业以及研究机构加入其中。云计算模式打破了以往的信息孤岛，从而可以用更好的服务形式和更低的成本提供给用户，而支撑云计算服务最核心、最底层的技术就是虚拟化技术[22]。

2.1 虚拟化

早在上世纪 60 年代已经有了虚拟化[23]用于 IBM 的大型机实现了分时系统，虚拟化技术是从 2008 年开始越来越成熟，经过学术界与企业界共同推进，得到了突飞猛进的发展。对于云计算的云端系统而言，其本质就是分布式系统。通过虚拟化技术在一个真实的物理平台上生成更多的虚拟平台，对于每个虚拟平台可以作为独立的终端运行在云端的分布式系统上。构建云基础架构关键技术之一就是虚拟化。虚拟化主要是通过一些技术将一个物理平台虚拟出多个虚拟平台，而且虚拟平台之间相互隔离[24]，共享一台物理平台上的资源，系统能够依据用户具体的使用情况动态的调整资源，从而达到更高效的利用物理资源，更好的避免资源的浪费现象。对于企业来说，利用虚拟化可以直接在原先的基础架构上安装部署云基础架构，更大程度上节省企业成本。

虚拟化技术对物理平台上底层资源(如：CPU、内存、I/O 设备等)进行抽象，屏蔽硬件具体实现细节，对底层物理资源进行隔离，为上层提供统一的管理调度等服务工作。具体实现大致为：在原先的直接运行在物理硬件资源上的操作系统进行隔离，在操作系统与底层硬件之间加入了虚拟化层。在虚拟化技术中，新加入的虚拟化层被称作虚拟机监视器(Virtual Machine Monitor，简称 VMM，有时也被叫作 Hypervisor)，物理平台被称之为宿主机(Host Machine)，而运行在宿主机内部的系统被称之为宿主机的操作系统，虚拟出来的平台被称之为客户机(Guest Machine) 或者虚拟机(Virtual Machine)，运行在虚拟机内部的系统被称之为虚拟机的操作系统[25]。虚拟化架构如图 2-1 所示：

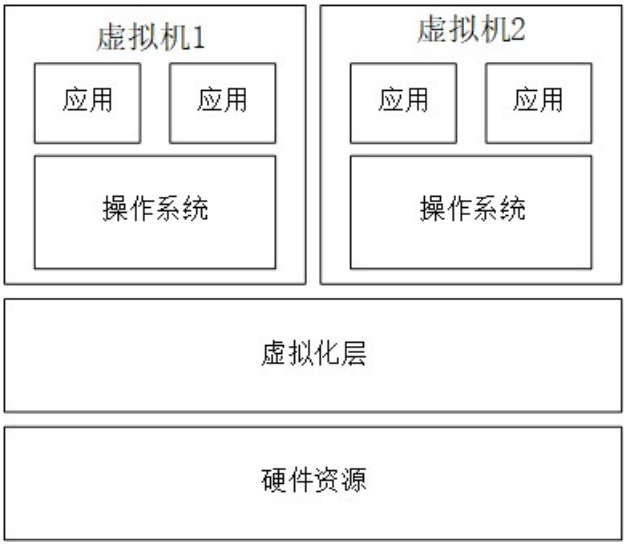

图 2-1 虚拟化基本架构

虚拟化技术允许多个应用程序或操作系统使用宿主机的硬件、软件资源。虚拟化层处在底层硬件与操作系统之间不仅提供透明访问低层硬件资源还起到管理应用及操作系统。虚拟化技术，从本质上，利用软件方法对 IT 计算资源重新定义划分，从而实现了硬件资源共享、动态分配调度，提高硬件资源的利用率，灵活多样的满足用户不同的个性化需求。虚拟化的用途也有很多种：

整合服务器资源，通过虚拟化技术对可以将不同架构的物理服务器整合成一个服务器之后在虚拟成不同的服务器对外提供服务，这样能提高服务器的效率、可用性及响应速度等，服务器的资源利用率也有了很大的提升。虚拟化可以多个环境用于应用或程序运行，也能够通过分配更多的资源提升提高服务质量。对硬件资源的虚拟化可以让一个物理硬件虚拟多个虚拟硬件资源让多个虚拟机使用，让虚拟机感觉像是独占物理资源。虚拟化可以在一台宿主机上同时运行多个操作系统，每个操作系统上根据用户的需要可以运行不同的应用程序。

2.1.1 虚拟化技术分类

虚拟化层的功能为获取虚拟机对底层物理资源的直接访问，并将需求重新分配给虚拟资源。对于虚拟化层的实现有两种不同实现方式，一种为纯软件方式，另外一种为物理资源提供相应机制来实现，前一种被称为软件虚拟化，后一种被称为硬件虚拟化。

QEMU(硬件设备模拟器)就是一款众所周知的纯软件实现的开源虚拟化方案，QEMU 采用纯软件形式模拟仿真宿主机上处理器功能(如：取指令、解码以及执行指令)，

客户机的指令并不是直接在宿主机上执行。由于所有的指令都是软件来模拟实现的，也就不可避免的导致系统效率低下。

硬件虚拟化，也就是宿主机对一些特殊指令的获取以及重定向提供硬件上的支持。这就需要宿主机的硬件(如：CPU)具有虚拟化功能，具有虚拟化功能的硬件可以提供额外的硬件资源，帮助软件完成重要硬件的虚拟化工作，进而提高系统性能。工作的原理大致为：当具有虚拟化功能的 CPU，检测到虚拟机直接访问物理资源时，CPU 会暂停虚拟机运行，并将控制权交给 VMM 去处理该工作。或者，当虚拟机访问特定资源时，并不需要暂停虚拟机运行，而是硬件资源直接将虚拟机的请求重定向到虚拟资源(由 VMM 指定)。

还有一种分类是依据是否对虚拟机操作系统内核进行修改，将虚拟化技术分为准虚拟化(Para-virtualization，也被称之为半虚拟化)[26]，全虚拟化(full virtualization)。

准虚拟化技术，通过对虚拟机操作系统内核修改，能够使虚拟机与 VMM 协同工作，将 VMM 被动获取虚拟机访问物理资源变成虚拟机直接通知 VMM 需要访问物理资源，也就相应的提高了系统的性能。代表性方案有 XEN。

全虚拟化技术，并不需要对运行在宿主机上的虚拟机操作系统内核进行修改，VMM 为虚拟机创造完整的虚拟化环境，让虚拟机感觉不到自己运行虚拟化环境。

2.2 桌面虚拟化

桌面虚拟化，又被称之为桌面云，是虚拟化技术的应用之一。是指通过服务器虚拟化技术，让更多的虚拟桌面运行在服务器上，通过桌面传输协议，使用户可以在本地客户端上使用。桌面虚拟化将桌面环境、用户所使用的软件与客户端物理设备隔离开来，实现桌面环境与终端设备解耦和。让用户更加便捷使用自己桌面。桌面环境以虚拟机的形式运行在服务器上，服务器负责任务的执行处理以及存储数据信息，客户端的作用就是一个显示器，客户端如果出现故障并不会影响到用户桌面环境和数据，这样就避免了数据丢失、桌面环境破坏等重大问题。决定桌面虚拟化性能两个最重要的技术为服务器虚拟化与桌面传输协议。

服务器虚拟化，简而言之，就是对服务器硬件资源的虚拟池化，对服务器资源更好的分割管理，在一个服务器上可以运行多个虚拟机，而且虚拟机之间相互隔离，互不影

响。对于某个虚拟机出现故障，运行在同一台服务器上的虚拟机还能正常使用。

桌面传输协议的职责就是连接终端用户与服务器，传输图像、音频、鼠标等封装的数据信息，数据在服务器上处理，客户端仅仅作为显示界面，让用户感觉像操作真实的物理机一样。

2.2.1 桌面虚拟化架构

桌面虚拟化架构主要有两种，VDI 架构和 IDV 架构。

1. VDI 架构

VDI 架构是最早提出的桌面虚拟化技术架构，得益于当时服务器的硬件配置提升带来的计算能力增强以及服务器虚拟化技术支持，使得 VDI 架构的桌面云更好的普及运用。VDI 架构的特点为“集中存储，集中管理，集中计算”。通过服务器虚拟化技术对云端服务器资源池化，为上层运行的用户桌面环境提供虚拟化资源进行运算与数据存储等工作，对于终端用户使用桌面传输协议访问自己桌面环境[27]。也即是说，用户使用的并不是本地桌面环境，而是通过瘦客户端或移动终端利用桌面传输协议连接服务器上的虚拟桌面，利用服务器强大的计算能力更高效的完成任务，而且服务器利用虚拟化技术能支撑很多的虚拟桌面环境，让更多的用户同时使用自己的虚拟桌面环境。

与传统的计算机相比，这种 VDI 架构优点：首先，便捷的使用，由于桌面环境与终端设备的解耦和，也就摆脱了固定办公的限制，无论何时、何地都能通过网络连接使用自己的桌面环境；其次，便于维护管理，用户的桌面环境都是集中运行在服务器上，对于管理人员来说摆脱了物理环境的距离，在服务器上对所有的用户桌面环境进行维护；最后，安全性提升，因为桌面环境运行在服务器上，产生的数据也保存在服务器上，增强了数据信息的安全。由以上优点特性，VDI 架构也被国内外各大企业广泛运用。虽然该架构有众多优点，还是有一些弊端存在。对网络依赖度高，如出现网络中断问题，就会造成用户无法使用自己的桌面环境；由于所有的数据都是在服务器进行处理然后通过网络进行图片、视频数据的传输，网络带宽不足的话，响应时延就会增大，必将影响用户体验；对服务器要求高，用户桌面环境集中在服务器上，所需的资源也都有服务器提供，势必需要高配置的服务器提供支持。

2. IDV 架构

IDV 架构是 Intel 公司最早提出的。该架构的特点：集中存储，分布计算。分布计算设计的思想就是想利用终端客户机的资源进行数据的处理工作，把繁重的计算工作从服务器上迁移到本地客户端，服务器的重点工作为用户桌面环境的管理与存储工作[28]。该种设计降低了对服务器的依赖，更好的利用客户端的资源，对网络的依赖也没有 VDI 高。对于桌面环境需要在本地运行也就导致了本地客户端需要虚拟化支持。工作原理为：客户端运行后，需要从服务器上将用户所需桌面环境传输到本地电脑，客户端接收完数据后，在本地操作并由本地客户端提供所需资源进行数据处理，为了保证本地与远程服务器上用户桌面环境一致，需要将差异的数据同步到服务器上。IDV 架构虽然对网络依赖没有 VDI 高，但是 IDV 架构对于终端设备配置要求较高，必须要胖客户端的支持，同样的也不支持移动办公。

2.2.2 桌面传输协议

桌面传输协议设计的目的是为了解决云端服务器与客户端通信，让用户更加便捷的使用桌面环境。桌面传输协议将用户在客户端鼠标操作、键盘操作等通过协议封装传输到服务器端解析处理后，再通过协议将服务器端处理的结果进行封装传输到客户端，客户端进行相应的解析最后在显示器上显示处理后的画面信息。基本原理如图 2-2 所示：

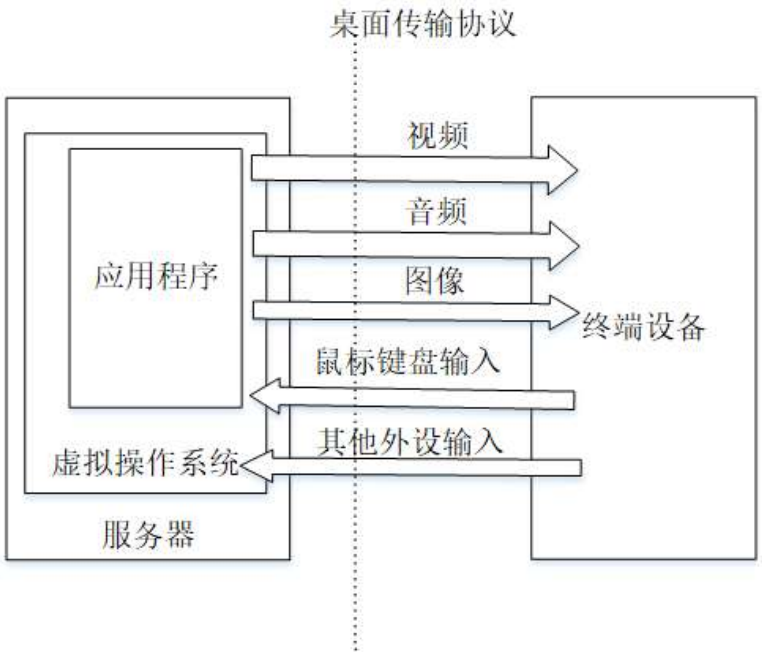

图 2-2 桌面传输协议基本原理

由此可知，桌面传输协议的职责为实现客户端与远程服务器端之间数据交互的传输工作，具体是怎么实现的过程，以及用什么格式传输，传输的效率，传输数据的安全等，这些都是传输协议要考虑事情。传输协议设计的目的就是更快、更好的传输数据，尽最大程度降低响应延迟，提高用户满意度。

目前桌面虚拟化有两类分别是开源的和商用的，商用的主要有微软的 RDP 协议、VMware 公司虚拟化产品 VMware View 采用的 PCoIP 协议、Citrix 公司虚拟化产品 XenDesktop 使用的 ICA 协议，开源的桌面传输协议主要有 AT&T 开发的 VNC 传输协议和 Redhat 公司开发的 SPICE 传输协议。影响桌面传输协议性能的因素，主要有以下几方面：

图形数据处理。在桌面云系统中，客户端与服务器端之间需要进行大量的图形数据传输，以怎样的格式传输、采用什么样的压缩技术，对桌面传输协议都是要考虑的。尤其是用户使用客户端看视频的时候，对屏幕的更新频率高，图形的传输需要很大的带宽。目前各种桌面传输协议对图形数据的处理采用的方式有两种，即基于位图的处理方式和基于矢量数据的处理方式。基于位图的处理方式是图形数据的渲染工作在服务器端完成，压缩服务器端产生的位图数据，对压缩之后的数据再进行编码处理，最后传输到客户端直接显示，采用位图处理方式的桌面传输协议有 RDP 传输协议、PCoIP 传输协议。基于矢量数据处理方式是服务端对图形数据转换成多种数据形式(如：位图数据、图形命令、文本等)进行传输，对图形的渲染处理工作是在客户端完成的，采用矢量数据处理方式的桌面传输协议有 ICA 传输协议。而 SPICE 协议不仅采用了基于位图处理方式，也采用了基于矢量数据处理方式，在数据传输时根据不同的情况选用合适的处理方式。在带宽方面，矢量数据处理方式所需带宽小于位图处理方式。

压缩与缓存处理。压缩与缓存在桌面传输协议主要是为了解决网络带宽，减少数据的传输量。图形的压缩算法对用户使用的虚拟桌面体验效果有相当大的影响，而且客户端与云端服务器之间有大量的图像进行传输，在网络带宽受限的情况下，如何保证图像更好的传输，让用户有更好的使用体验，这就对桌面传输协议提出了较高的要求怎样才能更好的解决网络带宽受限与大量图像传输的问题。对于各种桌面传输协议采用的压缩方式也不尽相同，也都有各自独特的处理方式，其中，ICA 传输协议采用层级压缩的思想[29]，因为传输协议是分层的，对不同的层采用不同的数据压缩；PCoIP 传输协议依据

的是对不同特征的传输图像进行分类压缩处理的方法。对图形数据缓存最基本的实现方式为保存当前的图像数据，对于整个桌面图像并不是所有的区域都有变化，更多的时候都是桌面的某一个区域或者某几个像素点发生了改变，因此，图形数据传输的时候，只需要把变化的区域部分进行传输，这样做就减少了数据量的传输，从而节省了网络带宽。SPICE 传输协议对于视频数据传输使用了压缩算法为静态图像压缩算法对只是仅仅对帧内数据进行压缩，忽略了帧间数据的相关性，没有对帧间数据采用压缩算法，也就导致了用户使用客户端播放视频时出现丢帧、延时等现象，严重影响了用户体验。

多通道支持。对于终端用户来说，操作虚拟桌面需要传输多种外设(如：键盘、鼠标等)数据信息，为了对不同设备更好的处理，桌面传输协议为不同的 I/O 设备数据传输设置不同的虚拟通道，不同的设备产生的数据通过不同的虚拟通道进行传输处理更快，让用户体验效果更好，对可以对通道设置不同的优先级。RDP 传输协议具有 64000 个独立的虚拟通道。ICA 传输协议定义了 32 个虚拟通道传递不同的设备数据信息。SPICE 传输协议定义了 6 个虚拟信道用于不同的设备数据传输。

传输层协议。数据的传输过程是需要用到 TCP 或 UDP 协议的，二者都是网络运输层协议。TCP 协议的好处是保证传输数据的完整，避免数据丢失，这就像双刃剑，有利就有弊，为了保证数据的完整性，TCP 协议都会对每次接受的到的数据做 ACK 确认应答，发送方如果没有收到应答报文，就自认为数据丢失，接收方没有接收到数据，会再次发送原先的数据，这样就增大了数据传输的开销，TCP 协议主要用于敏感性、安全要求高的数据传输，例如鼠标的操作、键盘输入、打印机等的数据传输。UDP 协议没有确认机制，这样比 TCP 协议传输数据效率高，面对复杂的网络结构，无法保证数据的完整性。UDP 协议主要应用在对数据完整性要求不高的业务，如视频数据传输，对于桌面传输协议的运输层协议选择一般都是根据不同业务的需要组合使用 TCP 与 UDP 协议，这样会带来更加的用户体验[30]。

2.3 本章小结

本章主要是简单介绍了桌面虚拟化相关的技术与概念，包括虚拟化技术、虚拟化技术分类、桌面虚拟化技术架构、桌面传输协议等。重点分析了虚拟化技术分类以及影响桌面传输协议性能关键因素。下一章将着重介绍实现办公桌面虚拟化关键技术做深入分析。

第三章 桌面云系统核心技术研究

对于桌面云是运用虚拟化技术将桌面环境、相关的软件与物理设备相分离，从而实现桌面环境与用户使用的物理资源解耦合。用户可以使用笔记本、手机以及瘦客户端在任何时间、任何地点登陆自己的桌面环境，从而实现访问桌面的灵活性、高效性。

3.1 QEMU-KVM 虚拟化技术

QEMU-KVM 虚拟化技术实际上是由两部分构成：KVM(Kernel-based Virtual Machine)虚拟化技术和 QEMU(quick emulator)设备模拟器。KVM 主要任务是操作系统的虚拟化和模拟 CPU、内存硬件资源，QEMU 主要任务是模拟虚拟机中其他 I/O(磁盘、网卡、显卡等)设备[31]。

3.1.1 KVM 虚拟化技术

KVM(Kernel-based Virtual Machine)，中文名称作内核虚拟机，最初是由以色列的 Qumrant 公司开发的。后来该公司被 Redhat 公司收购。KVM 初期开发并没有从底层开始，而是基于 Linux Kernel,在其上加载新的模块来实现虚拟化功能，对于 I/O 管理、进程调度、内存管理等功能的实现也是直接重用内核的功能模块。这样做目的是简化开发周期，从而也迅速发展，占领市场。在 2007 年 2 月被正式合并到 Linux 2.6.20 核心，使之成为内核源代码的一部分[32]。运行在 KVM 上虚拟机其本质就是 Linux 的一个进程。有自己的 PID 号。

KVM 是基于虚拟化扩展(Intel VT or AMD-V)的 X86 硬件[33]，Linux 全虚拟化解决方案。从 KVM 的 Makeflie 文件中了解到，KVM 的内核模块主要是由 kvm.ko、kvm-intel.ko 及 kvm-amd.ko 三部分模块构成[34]。kvm.ko 为 KVM 的核心模块，主要功能是提供与硬件平台无关实现虚拟化核心功能。另外，kvm-intel.ko、kvm-amd.ko 两者与硬件平台相关的模块，根据不同厂家提供的 CPU 使用不同的模块，提供 CPU 公司主要就两家 Intel 与 AMD，为了对两家企业的 CPU 虚拟化技术支持开发了不同模块。如果是 Intel 公司的 CPU 就加载 kvm-intel.ko，使用 AMD 公司的 CPU 就加载 kvm-amd.ko 模块。

1) KVM 的架构

依据虚拟机的基本架构，我们可以把虚拟机分成两类，即类型一与类型二。

类型一的虚拟机是当系统通电的时候，最先执行的是虚拟机的监控程序，在其创建的虚拟机内运行操作系统。虚拟机的监控程序运行在硬件设备之上，主要的功能是给上层的虚拟机提供系统初始化、管理物理资源的调度等工作，对于类型一的虚拟机的监控程序，在某种程度上它存在的意义，也被定义为一个专门虚拟机量身打造的操作系统内核。另外，该类型的虚拟机的监控程序还会提供一个特殊的虚拟机，该虚拟机的主要功能是对用户平常操作以及使用管理操作系统提供支持。Xen(知名的开源虚拟化软件)、VMware ESX/ESXi(著名的商业软件)以及 Hyper-V(微软的虚拟化产品)都是采用该类型设计。

类型二的虚拟机是当系统通电的时候，执行宿主机操作系统，而对于虚拟机的监控程序仅被当作应用程序。由于这样的设计，使得该类型的虚拟机能够更加充分的利用宿主机的操作系统。因此，虚拟机的监控程序并不需要设计物理资源的管理以及调度功能。也正是这样的设计，也就无法避免对宿主机操作系统的依赖，限制了虚拟机性能更好的发挥。正如，也就无法为了虚拟机的优化，去改动宿主机操作系统。不得不承认该类型的虚拟机设计简洁、实现方便，也很好的体现了 Linux 的设计哲学——实用至上。KVM、VMware Workstation 以及 VirtualBox 都属于该类型的虚拟机。

对于一般的 Linux 系统内核只提供两种执行模式：Kernel Mode(内核模式)和 User Mode(用户模式)。当 Linux 内核集成了 KVM 模块，为了适应 CPU 的虚拟化功能，出现了第三种模式，即 Guest Mode(客户模式)。

KVM 的架构如图 3-1 所示，可以看出 KVM 的架构并不复杂。KVM 虚拟机的核心部分就是 KVM 模块，KVM 模块主要功能是在 User mode 与 Guest mode 之间起到桥梁作用。工作流程是在用户模式下 QEMU-KVM 执行 ICOTL 命令来运行虚拟机，KVM 模块在接收到该请求后，首先执行一些前期的准备工作，例如把 vcpu 的上下文加载到 VMCS(virtual machine control structure)等操作，之后将 CPU 切换到客户机模式，从而达到运行客户机代码的目的。其实，从本质上来说，创建的虚拟机就是宿主机的一个标准的进程，也就意味着可以用 Linux 的进程管理命令对虚拟机进行管理。

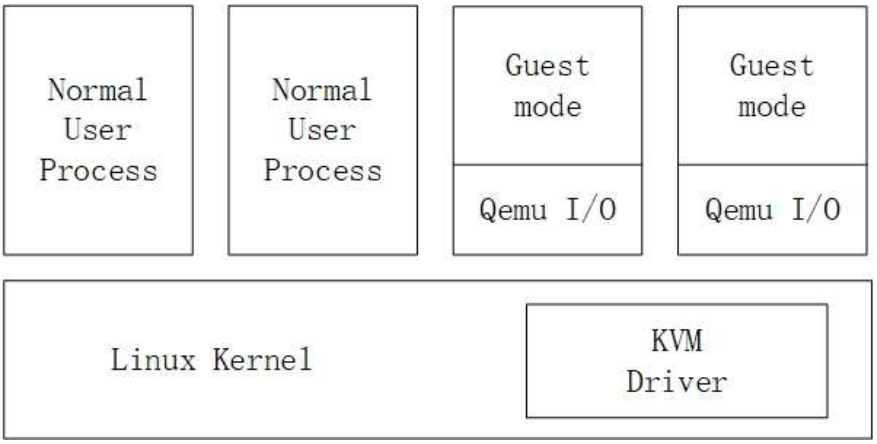

图 3-1 KVM 的基本架构

2) KVM 工作原理

在用户模式下 QEMU 通过 KVM 提供的 Libkvm 应用程序接口，执行 IOCTL 系统调用，进入 Linux 内核模式，内核中的 KVM 模块为虚拟机创建所需的虚拟资源(如：虚拟内存、虚拟 CPU)后，再通过调用执行 VMLAUNCH 指令从内核模式切换到客户机模式，进入客户模式加载客户操作系统。当客户操作系统发生外部中断等情况时，客户操作系统会暂停运行，退出客户模式切换到内核模式来处理异常，当中断处理完之后，再次切换到客户模式，执行后续操作。当发生的是 I/O 事件时，会切换到用户模式来处理相关的 I/O 操作。KVM 工作原理流程如图 3-2 所示

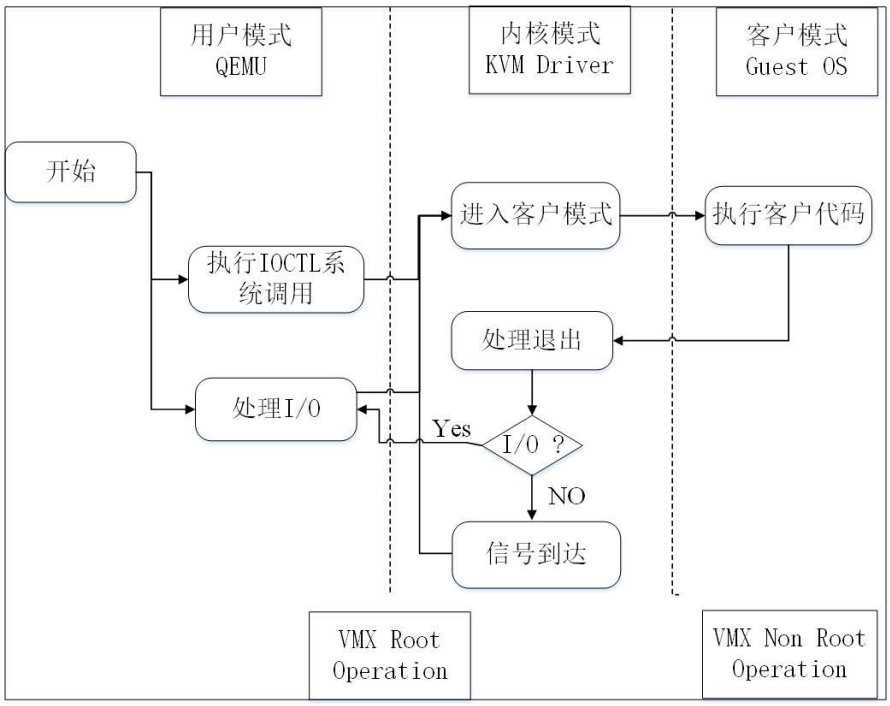

图 3-2 KVM 工作原理流程

3) KVM 实现 CPU 虚拟化原理

从上面的介绍中，我们知道 KVM 是硬件辅助虚拟化，也就是说，KVM 对 CPU 的虚拟化需要物理 CPU 提供虚拟化支持[35]。X86 架构把指令集划分为 4 个级别,从 Ring0 到 Ring3。其中操作系统运行在 Ring0，应用程序运行在 Ring3，Ring1 与 Ring2 被分配给去驱动程序。

由于宿主机操作系统已经运行在了 Ring0 级别，客户机操作系统就不能直接运行在 Ring0，为了解决这个问题，以 Intel 公司为例，Intel 在其 CPU 中增加了 Intel-VT 技术，提供两种模式：VMX(Virtual--Machine Extensions) root operation 模式和 VMX non-root operation 模式。两种模式都支持 Ring0 到 Ring3 特权模式。客户操作系统运行在 VMX non-root operation 模式，宿主机运行在 VMX root operation 模式。CPU 可以在两种模式间进行切换，来执行宿主机操作系统代码或客户操作系统代码。当从根模式切换到非根模式，被称作 VM entry；从非根模式切换到根模式，称之为 VM exit。对于 KVM 虚拟机来说，当客户机操作系统需要执行指令时，处于根模式下的 VMM 通过执行 VMLAUNCH 指令将 CPU 切换到非根模式来执行客户操作系统指令，这个过程也就是上面说的 VM entry；当运行在非根模式的 CPU 需要执行特殊的操作，如访问硬件 I/O 或者执行访问特殊设备寄存器等，此时 CPU 需要从非根模式切换到根模式，将 CPU 的控制权交还 VMM，进入称之为的“陷入模拟”，同时将用户的上下文保存在虚拟机控制结构中。等到 VMM 执行完特殊指令操作后，将结果和 CPU 的控制权返还给客户，同时虚拟机控制结构将客户上下文恢复，执行后续操作，直到指令操作结束。VCPU 创建工作过程如下图 3-3 所示。

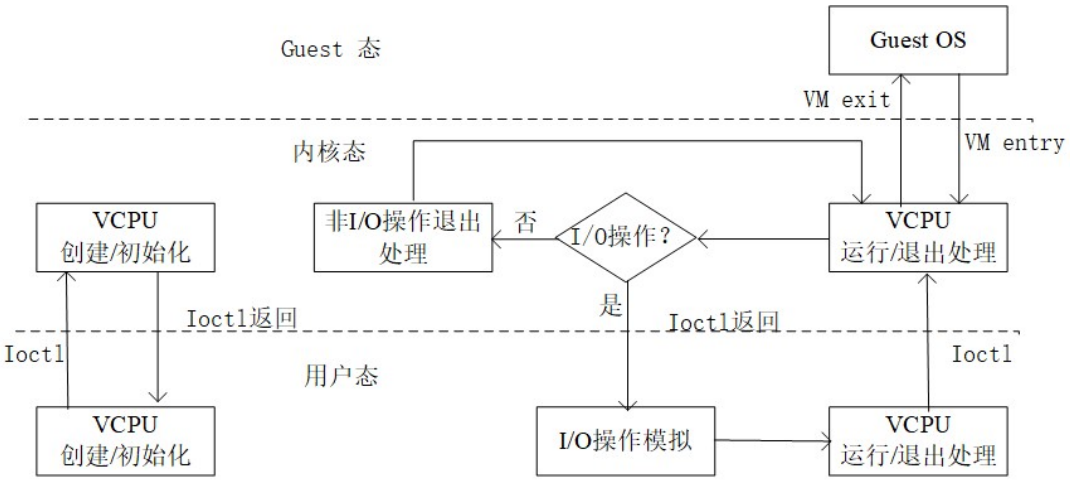

图 3-3 VCPU 创建工作模型

创建 VCPU 是发生在创建虚拟机过程中，VCPU 实质上是 VMCS(Virtual Machine Control Structure)结构体，该结构体是 Intel-VT 技术设置的，特意用来储存 VCPU 状态以及该 VM 所在 Host Machine 的相关状态信息，VMCS 与物理 CPU 资源是一一对应的关系。创建过程分为两步：第一步为在内存中分配空间给 VMCS；第二步对 VMCS 执行初始化操作，操作完成之后即可以用于虚拟机给虚拟机提供虚拟处理器支持。在 KVM 中，一个虚拟机其实就是一个进程，虚拟机中的分配的虚拟处理器其实就是线程。QEMU 通过 ioctl 系统调用与内核 KVM 模块交互，QEMU 向 KVM 发送 VCPU 创建指令，KVM 执行创建过程。创建 VCPU 之后，KVM 执行 VM entry 操作将 CPU 切换到虚拟机操作系统，虚拟机开始工作。

4) KVM 实现内存虚拟化原理

正如虚拟 CPU 一样，每个虚拟机也同样拥有属于自己的虚拟内存，KVM 虚拟化为了实现让每个虚拟机得到一个相互隔离以及连续的内存空间，设置了虚拟机物理地址空间(Guest Physical Address, GPA)，实际上该地址空间并不是真正意义上的物理地址空间，它仅仅是宿主机的虚拟地址空间(Host Virtual Address, HVA)的映射。GPA 与 HVA 映射关系如下图 3-4 所示。

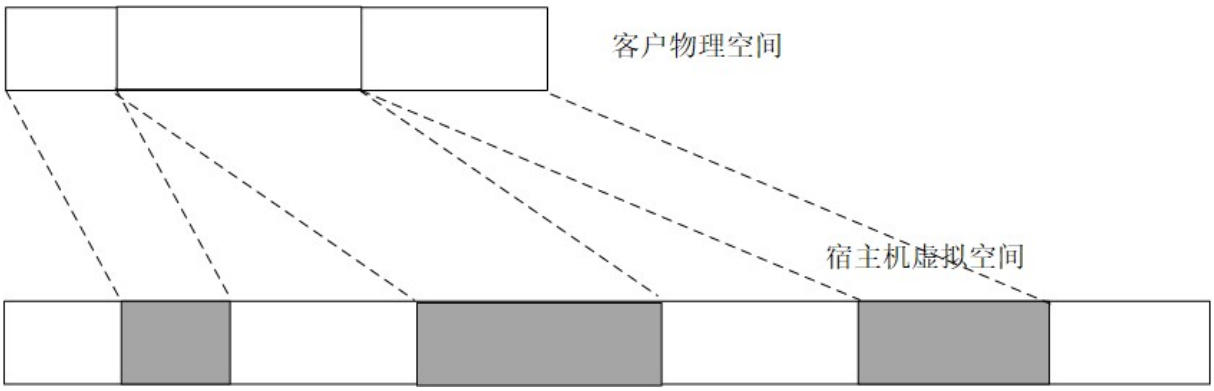

图 3-4 GPA 与 HVA 映射关系

基本的客户虚拟地址(Guest Virtual Address, GVA)到宿主机物理地址(Host Physical Address, HPA)转换过程是先把客户的虚拟地址转换成客户的物理地址空间，之后 KVM 通过 kvm_memory_slot 数据结构将客户的物理地址空间转换为宿主机的虚拟地址空间，最后将宿主机的虚拟地址空间转换成宿主机的物理地址空间。GVA 到 HPA 转换过程如下图 3-5 所示。

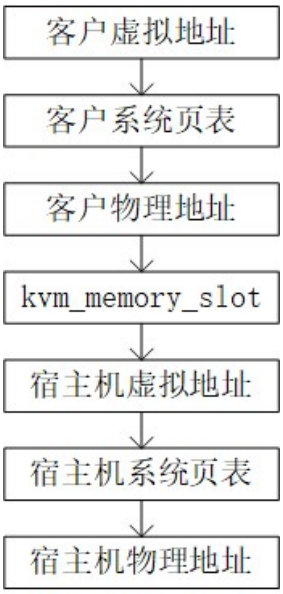

图 3-5 GVA 到 HPA 转换过程

对于此种转换过程每次一的地址转换都需要 KVM 操作，我们也能直观的感受到中间多次进行软件操作，不难发现整个转换过程效率相当低的，为了解决这种效率低下的问题，KVM 内存虚拟化采用了两种优化的实现方案。

方案一是采用影子页表(Shadow Page Table) 实现客户虚拟地址直接转换为宿主机物理地址空间，该方案是纯软件实现的，缺点实现复杂而且内存开销大转换过程，如下图 3-6 所示。

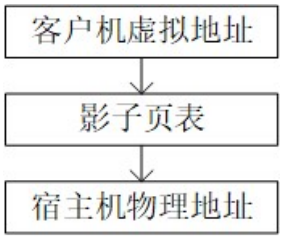

图 3-6 利用影子页表从 GVA 到 HPA 转换过程

方案二是采用硬件 CPU 对虚拟化的支持来实现，以 Intel 为例，该公司提供了 EPT(Extent Page Table)技术对内存虚拟化进行支持，该技术支持从客户虚拟地址空间转换客户物理地址空间以及客户物理地址空间转换宿主机物理地址空间的两次地址转换全部由硬件完成。优点为提升虚拟化性能并且降低虚拟化实现复杂度如下图 3-7 所示。

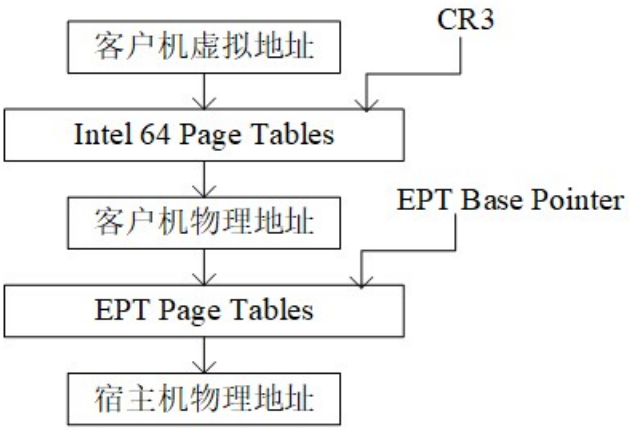

图 3-7 借助 EPT 技术从 GVA 到 HPA 转换过程

3.1.2 QEMU 硬件设备模拟器

QEMU 原本就是一个开源的虚拟化软件，能够用纯软件的方式模拟虚拟机各种所需硬件设备。由于是纯软件设计开发，在其创建的虚拟机性能比较低下，对于商业化方案，一般都采用与 KVM 虚拟化结合的方法来实现虚拟化工作。在 QEMU-KVM 虚拟化中，对于 QEMU 模块的功能主要是实现 I/O 设备虚拟化工作。

1. 硬件 I/O 设备虚拟化

QEMU 对于硬件 I/O 设备的模拟包括有：硬盘设备、网卡设备、显卡设备、声卡设备等的模拟实现。对于该部分主要是着重介绍对存储、网卡、显卡三种设备的模拟工作。

1) 对存储设备的模拟

QEMU 对于存储设备模拟的支持有许多种，不仅有软盘、U 盘、普遍的 IDE 磁盘，而且提供对具有 virtio 功能的磁盘支持，对启动这些设备的顺序提供相当灵活的配置。QEMU 创建的虚拟机是以镜像文件形式存储的，QEMU 支持的镜像文件格式很多种，常用的文件格式为 raw，qcow2 等。使用文件的形式做镜像好处也很多，例如：在一个存储设备上可以存放很多镜像文件，对镜像文件管理起来也比较方便，在移动以及复制镜像文件也相当便捷。镜像文件的构建方式也有很多种，比如：物理磁盘、NFS(Network File System) 文件系统、GFS2。

2) 对网络设备的模拟

网络功能对于各种云计算产品都相当重要，网络故障对于互联网企业来说是相当可怕的，网络的配置在各种虚拟化方案中也是相当重要的。QEMU 也提供了多种网卡设备

模拟的支持，最常用的模拟网卡为“rtl8139”，它已经成为 QEMU-KVM 中默认的网卡。对网络模式的配置 QEMU 也提供了多种支持，包括：

a) 基于网桥(bridge)的模式

网桥模式使客户机有独立的 IP 地址并且共享宿主机的网络设备，让客户机具有了直接连接网络的功能，对外部网络而言，客户机就像一个真实的物理主机一样，外部网络与客户机能够直接相互访问。

b) 基于 NAT(Network Address Translation)的模式

NAT 中文名为网络地址转换，它是一种广域网接入技术。在这种模式下客户机访问外部网络需要进行网络地址转换，将分配给客户机的内部 IP 地址转换成可以访问外部网络的 IP 地址。一方面 NAT 可以节省 IP 地址资源，另一方面内部的 IP 地址并未暴露给外部网络也就避免了虚拟机被外部网络攻击。

c) QEMU 内部的用户模式网络(user mode networking)

用户模式网络是由 QEMU 自身实现的，使用起来最为方便。虚拟机可以访问宿主机也可以访问外部网络，但是，反过来却不能访问，也即是说宿主机与外部网络都不能直接对虚拟机进行访问。对于部分网络功能也缺少支持，由于都是 QEMU 来实现，造成了该模式性能不太好。

3) 显卡设备的模拟

QEMU 提供了多种模拟的显卡设备，比如 VGA(Video Graphics Array)显卡的模拟类型提供了多种，对于其中的 cirrus 类型提供了“Cirrus Logic GD5446”显卡的模拟，vmware 类型提供了“VMWare SVGA-II”显卡的模拟。

2. QEMU 模拟硬件设备流程

QEMU 对于设备的模拟采用的纯软件形式，该方式的优点很明显就是可以模拟各种不同的设备，缺点每次设备的模拟都需要多次上下文切换，多次复制，不仅耗时而且模拟的设备性能比较差。

硬件设备模拟的大致流程为：客户机的设备驱动程序(Device Driver) 发起 I/O 操作请求，内核中的 KVM 模块捕获该次 I/O 请求，并将该次处理后的 I/O 请求信息存放到 I/O 共享页(I/O Sharing page)中，并报告给用户空间的 QEMU 模块。用户空间的 QEMU 模块从 I/O 共享页获取具体的 I/O 操作信息，并将该信息交由 QEMU 的 I/O 模拟代码进

行处理。处理完成之后，QEMU 模块将结果返回给用户空间的 I/O 共享页，然后 QEMU 模块通知内核中的 KVM 模块。KVM 模块中的 I/O 捕获代码从 I/O 共享页中获取处理结果，最后把结果反馈给客户机。模拟流程图如图 3-8 所示。

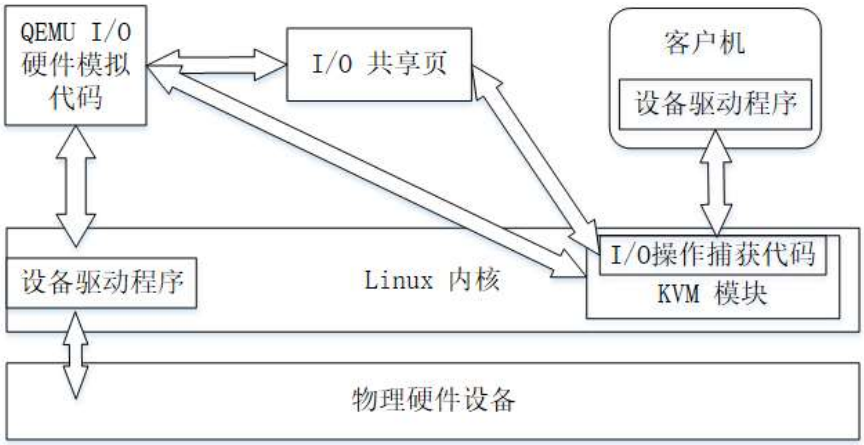

图 3-8 QEMU 模拟 I/O 流程

3.1.3 VIRTIO 半虚拟化驱动

上文对 QEMU 的分析，我们知道 QEMU 对设备的模拟采用的是纯软件形式，在 QEMU 模拟硬件设备过程，需要进行多次上下文切换，还要进行多次的复制工作，这就不可不避免的造成QEMU模拟硬件设备效率性能较差。为了弥补I/O设备性能不足问题，KVM 采用 VIRTIO 半虚拟化驱动方式来提高 I/O 性能。

VIRTIO 半虚拟化驱动是半虚拟化管理程序中通用 I/O 模拟设备抽象[36]。管理程序通过 API(应用程序接口)向外部提供通用的模拟 I/O 设备。VIRTIO 是运行在虚拟机监视器之上的抽象 API 接口，使得客户机能够感知到是在虚拟环境下运行。在 QEMU/KVM 中，VIRTIO 基本结构组成如下：

前端驱动：该部分是安装在客户机中的驱动程序模块，由六部分组成 virtio-blk、virtio-net、virtio-pci、virtio-balloon、virtio-scsi、virtio-console。 对于不同的系统也都提供了特定的 virtio 驱动支持，对于现在新发行的 Linux 版本已经将 virtio 驱动直接进行编译为模块客户机可以直接使用，对于 Windows 系统由于并不开源，需要用户单独安装才能使用。

后端处理程序：在 QEMU 中实现，调用宿主机上的硬件 I/O 设备。

virtio 层：虚拟队列接口层，根据不同的需求提供不同数量的队列。例如网络驱动

程序(virtio-net)分配了接收与发送两个虚拟队列，块驱动程序(virtio-blk)分配了一个虚拟队列。

transport 层：又称为 virtio-ring，主要是实现环形缓冲区(ring buffer), 对前端驱动以及后端处理程序的信息保存，并且 ring buffer 一次性能够存储多次前端驱动发来的 I/O 请求，之后交由后端驱动进行批量 I/O 处理工作，最后实际上是由宿主机的设备驱动来完成物理上的 I/O 请求。

VIRTIO 的基本结构框架如图 3-9 所示：

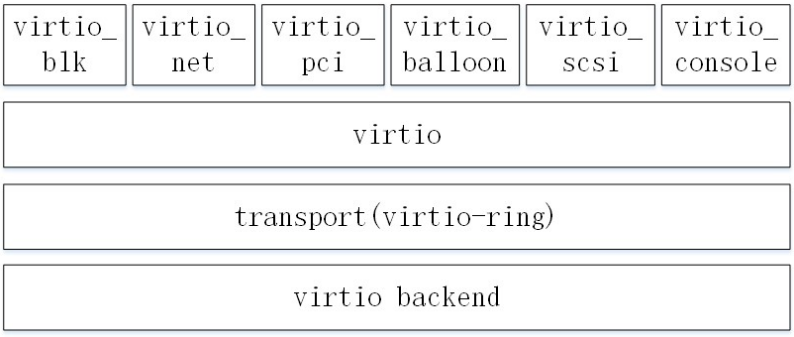

图 3-9 VIRTIO 基本框架

VIRTIO 的优点可提供接近本地的 I/O 性能，缺点就是需要在客户机中装载特定的 VIRTIO 驱动，数据的传输也有格式上的要求需按照 VIRTIO 的规定。

3.1.4 libvirt 库

Libvirt 库是一个开源并且支持多种主流开发语言的函数库[37]，设计的目的是为了方便虚拟化技术的平台管理。常用的虚拟机管理工具以及 Openstack 等云计算框架平台底层都有用到。libvirt 也对常用的虚拟化方案(如 KVM、QEMU、Xen 等)提供支持。libvirt 是采用基于驱动程序的架构来实现对不同虚拟化方案的支持。作为中间适配层的 libvirt 对于底层虚拟化方案的实现细节有很好的屏蔽作用，为上层的管理工具提供统一的应用程序接口。避免了研究不同虚拟化方案的麻烦。Libvirt 基本结构如图 3-10 所示。

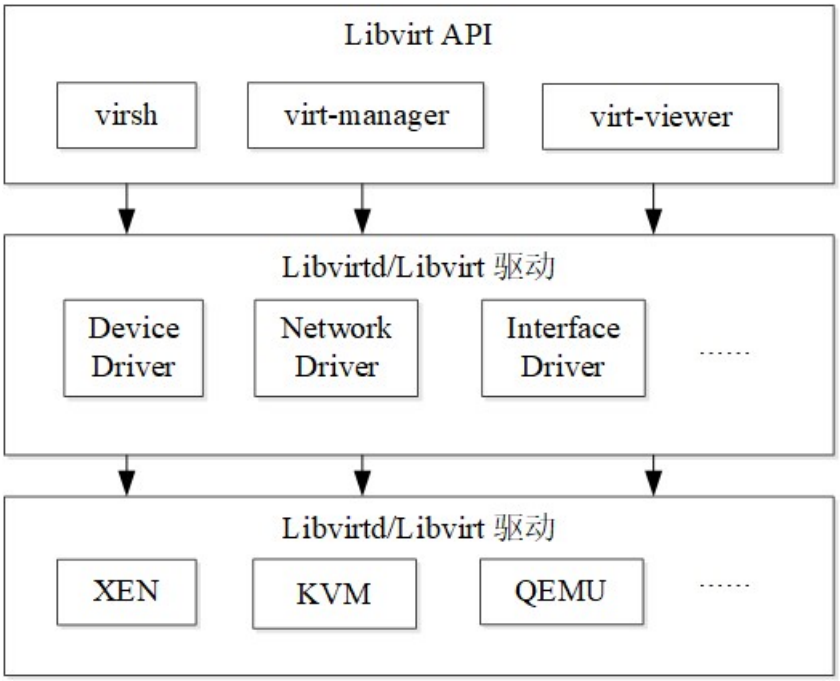

图 3-10 Libvirt 基本架构

libvirt 主要是由 libvirt 守护进程、virsh 命令行管理工具以及应用程序接口，三部分组成。libvirtd 守护进程负责执行对节点上的域的管理工作，管理工具对虚拟机进行管理时，libvirtd 守护进程必须是处于运行工作状态。virsh 是对虚拟化功能实现的管理命令。应用程序接口，又称为 libvirt API，主要有 5 类构成。它们之间的关系如图 3-11 所示。

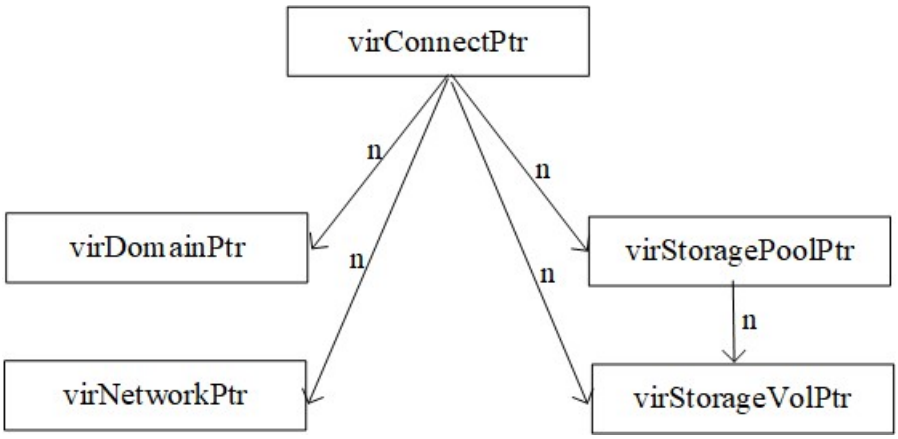

图 3-11 libvirt API 主要对象之间关系

libvirt 的管理功能主要有：

1) 对域的管理

域是指运行在虚拟机监视器(Hypervisor)之上的虚拟机操作系统实例。libvirt 对域的管理也就是对虚拟机的整个生命周期管理，包括虚拟机的启动、虚拟机的停止、虚拟机的保存、虚拟机的恢复以及虚拟机的动态迁移，以及对多种硬件设备(如磁盘、网卡、

内存)的热插拔操作。

2) 对远程节点的管理

对远程节点的管理是通过在远程节点上运行 libvirtd 守护进程来实现的。

3) 对存储的管理

存储的管理包括存储池和存储卷两种。存储池可以划分成不同的存储卷，存储卷可以分配给用户使用。对存储的管理主要的功能有：对不同文件格式虚拟机镜像的创建、在给多台服务器挂在共享存储、对磁盘设备分区以及远程管理存储。

4) 对网络的管理

对网络的管理包括虚拟网络的接口的创建，网络接口的配置，为虚拟机分配虚拟网络，设置 NAT 网络，管理 VLAN 等。

决定桌面云性能的三个关键因素之一就是桌面传输协议。对于提供桌面云服务的企业来说，桌面传输协议同样也是他们最核心的技术。桌面传输协议的效率直接影响到用户体验，而用户体验的好坏对桌面云产品的市场份额也会产生影响。桌面传输协议其实质就是定制的数据传输规则，让数据高效的在客户端主机与终端服务器之间传输，满足用户体验需求。下文将详细介绍为实现办公桌面云系统采用的桌面传输协议。

3.2 RDP 桌面传输协议

RDP(Remote Desktop Protocol)远程桌面协议是微软公司基于国际电信联盟的 T.share 协议基础上改进得来的[38]。设计开发该协议的目的，是为了实现无论在何时、何处，用户都能通过 RDP 远程桌面协议来操作远程服务器或虚拟机，并把处理之后的结果传输到本地桌面，就像操作本地电脑一样，即客户端通过加密的形式将鼠标或键盘的信息，传输到后端服务器再重现还原用户的操作并执行，之后服务端再以加密的方式将处理的结果回传到客户端，最后通过客户端的图形界面把结果显示出来。在本文的办公桌面云设计实现中，我们采用该协议实现本地客户端与服务器上自己专属虚拟机交互以及信息数据传递。

3.2.1 RDP 协议栈

我们已经知道 RDP 协议是从 T.share 协议的基础上衍生出来的，而 T.share 协议有是

T.120 系列协议组成部分，T.120 协议集的目的是为了给应用程序以及服务程序，提供多点、实时的数据通信支持。由于它支持应用程序之间的共享、实时文本会议的功能，它又被称作远程应用程序的共享协议，让客户端的计算机共享远程计算机的应用程序以及实时图像信息，我们也可以把它称作远程共享显示协议。因此，RDP 协议也具有该功能。为了让多用户同时登录共享服务器资源，微软又从 Citrix 购买 MultiWin 技术支持，最后才有了微软的 RDP 桌面传输协议。由此我们可知，微软的远程桌面技术是由两部分构成的，远程图形显示协议以及远程桌面服务。

RDP 协议位于 TCP/IP(Transmission Control Protocol/Internet Protocol) 系列协议的应用层。其实，RDP 协议本身就是众多协议的集合体，从最初的 RDP4.0 版本发布，到后来的发布的各种版本，到现在已经发布到 10.0 版本，功能不断地增强，不断地提高用户体验，更加优化的数据传输效率以及美观的界面，同时越来越多的协议被融入了 RDP 协议里，不同协议之间相互配合集成，让本地计算机与远程虚拟机之间的交互更加高效便捷。

与其他通信协议一样，RDP 协议也有自己协议栈，也是有多层次组成，每一层的设计也是为了实现不同的功能提供不同的服务。对于 RDP 协议有五层组成，自下而上，最底层是 TCP 层，而后依次是 ISO 层、MCS 层、SEC 层以及最上层的 RDP 层[39]。下面部分将分层介绍不同层的功能，RDP 协议的协议栈模型如下图 3-12 所示。

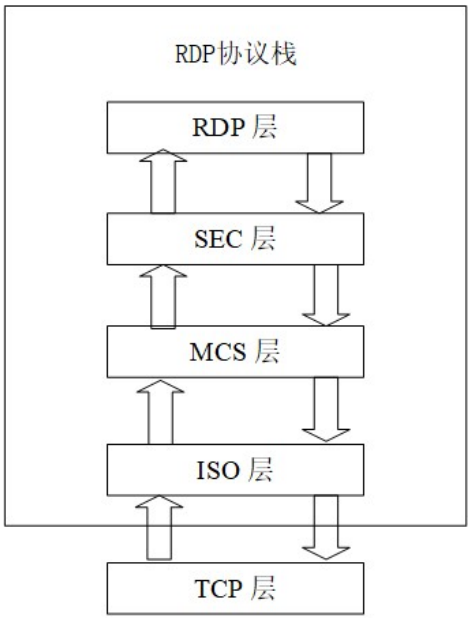

图 3-12 RDP 协议栈

RDP 层，属于应用程序层，位于协议栈的顶层，主要职责是协定客户端与服务端相互之间数据传输的格式，也即是说将客户端用户操作的鼠标以及键盘等数据信息以及服务器的图像等信息以不同的类型传输。最后生成该层 RDP 包头并将带有包头的数据包传输给下一层。该层的协议基础是 T128，即多点应用程序共享协议，它也是 RDP 协议最核心的协议。

SEC(Secure Connection)层，属于安全层，紧邻 RDP 层提供数据的加密以及签名服务。为了保证数据传输安全，采用对称的加密算法对传输的数据进行加密处理，服务端接受数据后用协商好的密钥对数据进行解密，达到预防数据流被非法劫持或修改。

MCS(Multipoint Communication Service)层，位于 SEC 层下方，该层主要功能是提供组播服务，支持一对一、一对多的连接方式，除了组播服务外，还提供通道管理以及数据传输等服务。RDP 协议定义了数量众多的虚拟通道，而对通道进行管理的任务是 MCS 层完成的，这些通道都是为了数据传输。对于鼠标数据、键盘数据、剪贴板、声音、图片等数据都是通过不同的虚拟通道进行传输的，不同的数据传输到客户端或者服务端通过协商好的信道也能判断传输的是什么数据。

ISO(Open System Interconnection)层，属于传输服务层，主要职责是将连续的数据流进行不同大小的分片处理，生成传输数据包，之后调用底层服务(TCP 服务)对数据包进行传输。ISO 层也提供了四种不同的服务类型，分别是处理数据传输服务、断开连接请求服务、连接请求服务以及连接确认服务。处理流程大致如下：首先客户端发出连接请求服务，服务端收到请求信号后，发出连接确认服务，客户端收到连接确认信号得知自己的请求得到相应并开始数据传输，当数据传输完成之后，客户端发出断开连接请求服务，服务端收到该信号知道客户端数据传输完毕，并响应客户端请求断开连接，一个服务流程完成。从这个处理流程可知，服务端处于被动应答位置。

TCP 层，处在 RDP 协议栈最底层，主要是提供 TCP 服务。通过上面四层传输到 TCP 层的数据已经经过的加密、分段、划分不同的虚拟信道等处理，该层依照 TCP/IP 格式对数据进行相应的封装，通过路由寻址找到目的服务器 IP 并将处理的数据送达，服务端收到数据进行相反的分拆数据包，最后应用程序处理数据。RDP 数据包结构图如下图 3-13 所示。

图 3-13 RDP 协议数据包

我们都知道 RDP 协议是微软公司开发的商业化远程通信协议，该协议并没有开源，而 rdesktop[40]是一个开源的连接 Windows 远程桌面运行在 Linux 平台的客户端，同时它参照 RDP 协议方式，实现了该协议的功能，也给我提供了详细深入研究 RDP 协议指明了方向，上面部分只是大致简单的介绍了 RDP 协议栈的架构以及不同层提供的服务和功能，下文将更深入的分析 RDP 协议客户端，RDP 协议实现连接过程，RDP 协议数据的传输。

3.2.2 RDP 客户端

对 rdesktop 的源代码研究之后，了解到 RDP 协议客户端业务逻辑流程，用户点击客户端输入计算机名、用户名等相关信息，客户端程序对相应输入的信息做检查，如果正确，客户端建立与服务器的连接；如果信息输入有误，则返回，与之对应的源程序实现是由 rdp_connect() 等完成的,按照程序实现层次调用依次为 sec_connect、mcs_connect、iso_connect、tcp_connect, 之后执行登陆请求信息是通过 rdp_send_logon_info 实现，该过程包括数据包初始化、数据包发送以及加密处理逻辑、客户端会与服务端进行一些重要的参数协商，如虚拟信道的商定等，之后客户端的远程桌面管理模块来处理远程桌面会话工作[41]。后续的操作主要由客户端的远程控制管理模块完成，对应的程序函数实现 rdp_main_loop()，流程为首先监控是否收到从服务端传输过来的数据，如果收到数据，对数据分析，判断是什么类型数据，进行相应的命令数据操作或者显示数据操作，然后继续接受处理后续数据，如果没有后续数据处理完毕，执行关闭窗口操作，断开与远程主机连接，最后退出客户端程序，RDP 客户端业务逻辑流程如下图 3-14 所示。

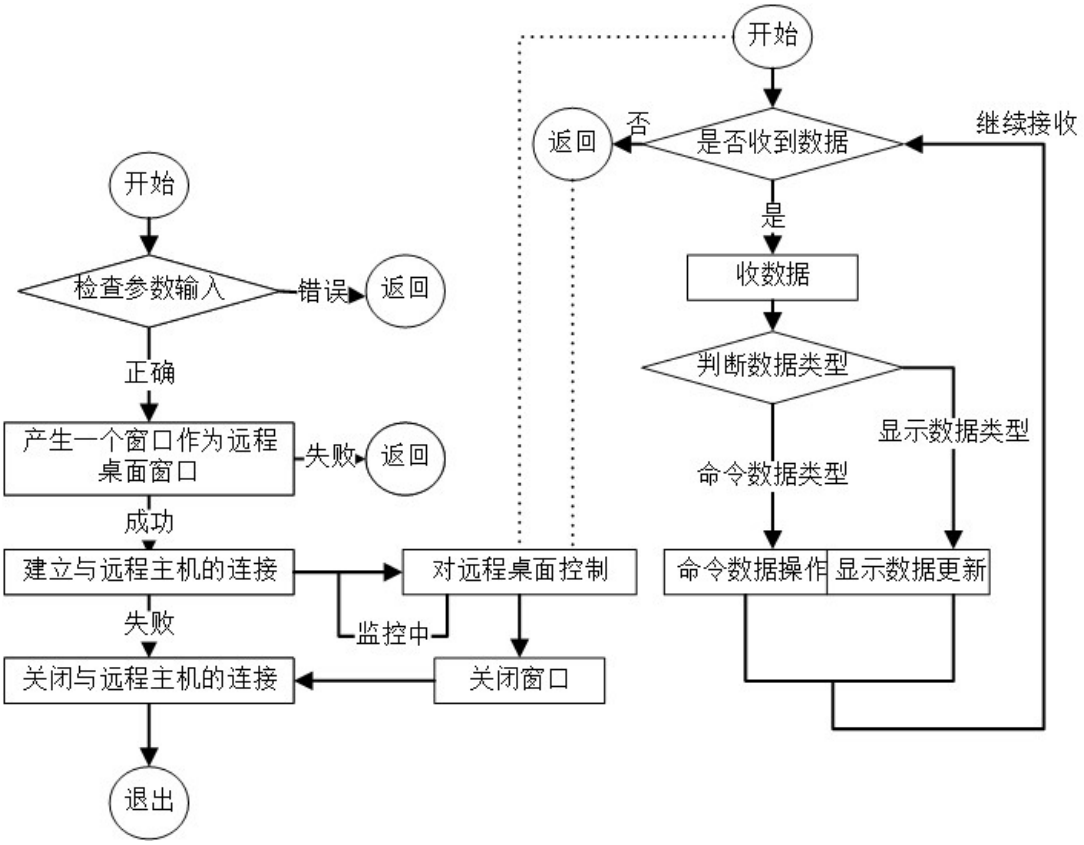

图 3-14 RDP 客户端业务逻辑流程

3.2.3 RDP 协议连接过程

在微软提供的官方文档[42]中，我们得知 RDP 协议连接过程被分为八个不同的阶段，连接通信的主要任务是为了让客户端与服务器端能够更加高效便捷的交换信息，同时包括一部分常用的配置信息的定义，目的是为了之后连接通信过程的稳定以及持久，让客户端与服务端之间能够高效的交换桌面图像以及其他相关的各种通信数据。RDP 协议连接过程被分为八个不同的阶段[43]，分别是连接初始化阶段、基本设置信息交换阶段、通道连接过程阶段、RDP 安全初始化阶段、安全设置交换阶段、客户端身份认证阶段、性能交换阶段以及连接完成。下文将对 8 个阶段详细的讲解。

1. 初始化连接阶段：该阶段分为两步，首先为客户端发送连接请求数据包到服务端，之后服务端接收请求后，响应客户端的请求并向客户端发送连接确认的数据包。

2. 交换基本设置阶段：该阶段分为两步，客户端发送 MCS 连接初始化数据包到服务端；服务端做出响应并向客户端发送 MCS 连接响应数据包，完成基本设置信息的交换工作。对于其中的 MCS 连接初始化数据包中包含有一个 GCC(Generic Conference

Control) 会议的创建请求数据包，服务端也会相应的在 MCS 连接应答数据包中添加 GCC 应答数据包对客户端做出响应。对于 GCC 协议的功能主要是用于传输大量连续的数据时，对数据进行分块传输处理工作。对于 RDP 而言 MCS 一般都是与 GCC 紧密合作，目的是为了保证在 RDP 连接阶段大量数据传输的稳定性。

3. 虚拟通道连接建立过程阶段：该阶段被分为 5 步，该过程为客户端发送一个 MCS 建立域请求的数据包，随后客户端再发送一个 MCS 附加请求用户信息数据包，服务端接收请求后，发送一个附加用户确认数据包对客户端发送的 MCS 请求用户做出回应，该回应数据包中包含有用户信道 ID。 之后客户端与服务器端之间相互发送请求及应答相应的虚拟信道数据包，最终成功加入所有虚拟通道。需要注意的是虚拟通道的建立过程是依次建立的，对于客户端的请求建立信道数据包必须等到服务端发送响应的虚拟信道应答数据包完成之后，客户端才能发起其他的请求信道数据包。

4. RDP 安全初始化阶段：对于 RDP 安全加密启动与否取决于之前的 GCC 请求数据包相关参数的设置，如果设置了相应的参数，该 RDP 安全机制启动，客户端发送一个包含了 32 字节公钥加密的随机数安全交换数据包。对于 32 字节的随机数来自于服务端的 GCC 创建应答数据包。之后客户端与服务端利用该随机数生成用于客户端与服务器端 RDP 数据通信的会话密钥。如果该加密行为一旦产生，那么后续的通信都会被添加一个安全头。当安全加密机制生效时，对于客户端发送到服务端的通信数据全部加密，而对于服务端到客户端的通信数据并没有这样的强制要求。

5. 安全设置交换阶段：客户端将安全信息数据包发送到服务端。安全信息包含用户名、密码等信息。

6. 证书发放阶段：该过程为服务端发送一个证书到客户端。对于不同安全级别的客户端，该过程还是有差异的，对于安全级别比较低的客户端并不需要此阶段。因为安全级别低的客户端可能并不需要证书下发以及存储。

7. 功能需求信息交换阶段：该过程为服务端向客户端发送一个服务端所支持的权限需求数据包，客户端收到该数据包后，响应服务端并做出应答，客户端发送一个响应权限设置数据包。

8. RDP 协议连接完成阶段：这是最后一个阶段，该阶段服务端与客户端会进行多次数据包的交互通信，对于客户端与服务器端之间请求与响应并没有严格的次序要求，也

即是说对于客户端发送的一次请求并不需要等到服务端响应该请求并接收到响应再发送其他的请求，而是客户端可以连续发送请求数据包，服务端接收到服务端发送的数据包，分别对数据包做出相应的回复，向客户端发送相应的应答数据包。

以上就是 RDP 协议连接过程，可以看出该通信过程也是以 TCP 为基础的，下文介绍 RDP 协议数据的传输，主要介绍桌面图像数据的传输。

3.2.4 RDP 协议数据传输

对于桌面云来说，RDP 主要传输数据为客户端用户操作虚拟机鼠标以及键盘的输入数据，以及服务器端将处理结果以图像的形式传输给用户更新用户虚拟桌面。我们知道 RDP 协议客户端显示的桌面实际上是虚拟桌面，仅仅是远程服务器端为用户虚拟机桌面在本地的投影。因此，RDP 协议数据传输主要讲解对服务端的图形传输更新。

图形数据传输都包含相应的图像处理命令，目的是说明该图形数据包执行怎样的操作。对于 RDP，它有大量的图形处理命令，对于这些众多的图形命令大致上被归为两类，即首要图形命令与次要图形命令。对于两类图形处理命令区分归类主要是该命令能否直接被图形设备接口命令调用执行在屏幕上显示，前者能被直接调用处理，后者需要缓存数据并依靠前者命令处理才能在屏幕显示。

首要图形命令包括有画矩阵命令、画线命令、画位图命令、画文字命令等。次要图形命令有位图数据命令、文字缓冲数据命令等。对于位图数据命令需要画位图命令配合完成，同样的文字缓存数据命令也需要画文字命令来配合执行完成。

画矩阵命令对于 RDP 图形命令被执行的频率是相当高的。画矩阵命令主要有 5 个参数构成，分别是矩阵的横坐标、矩阵的纵坐标、矩阵的宽、矩阵的高、填充颜色。为了减少字节，尽可能的节省网络带宽，图形命令的参数往往是以前一个同样的命令参数为基准，只发送相对于上一个相同命令参数的偏移量。

位图命令是处理位图图像，对于桌面图像实质上是由像素点按照某种规律的排列组合，像素点颜色的变化就会在桌面上生成不同的图片。位图命令的目的也就是在屏幕区域做填充，渲染出桌面图像。对于该命令由于数据传输量大，为了减少网络带宽的消耗，RDP 协议采用了压缩算法，即为 LZ77 算法。该算法创新性的将字典的方式运用到数据压缩，该算法的本质就是高效的利用数据中重复的信息来实现压缩从而减少编码的目的。

LZ77 压缩算法可以将 RDP 协议传输的图像数据压缩到原来的 40%，这样在较低的网络带宽情况下，LZ77 算法能够极大的降低数据的传输量，让用户有更好的使用体验。

3.3 SPICE 传输协议

SPICE(Simple Protocol for Independent Computing Environment)协议是开源的远程桌面传输协议，与其他桌面传输协议不同在于 SPICE 可以直接使用服务器上的资源，如对图形处理工作可以利用 GPU，因为它并没有运行在虚拟机内，而是运行在服务器上。通过 VDI(Virtual Device Interface)虚拟设备接口与虚拟机交互。

3.3.1 SPICE 架构

SPICE 主要是由四个部分构成：SPICE Protocol、SPICE Client、SPICE Server 和 Guest。SPICE 基本架构如下图 3-15 所示。

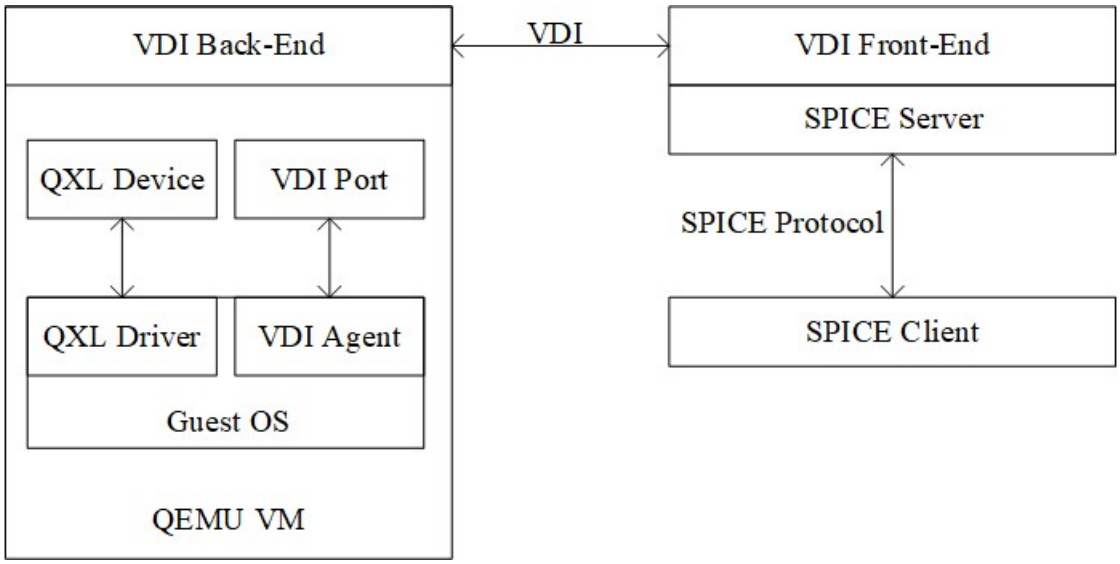

图 3-15 SPICE 基本架构

SPICE Client 是客户端，功能为显示桌面画面，也即是提供桌面环境给用户使用操作；SPICE 协议负责 SPICE 服务端与 SPICE 客户端交互，将用户的 I/O 操作压缩封装传输到服务端，并将处理结果反馈给客户端；虚拟机与 SPICE 服务端的交互是通过 VDI 进行的。

3.3.2 SPICE 图像处理流程

SPICE 处理图像流程开始于虚拟机中用户应用程序发出请求处理请求，GDI/X 图像

引擎获取请求并把请求发送给 QXL 驱动[44]。QXL 驱动得到请求并把请求转换成 QXL 命令，之后把命令发送到 QEMU 中 QXL 设备加入命令环中，命令环所使用的是内存空间，所需空间大小命令环可以自己调节。libspice(虚拟设备接口可插拔库)从 Commands Ring 中得到命令，把命令存放进图像命令树，在图像命令树里存放着即将显示的命令集合，也即是用于需要进行局部更新的图像相关信息，图像命令树是经过 libspice 优化的命令集合，把与图像处理不相关的命令去除。libspice 将命令树中的命令传送到发送队列，之后再将发送队列的命令转译为 SPICE Client 能够理解的 SPICE 协议信息，传送给 SPICE Client 并在图形界面显示出来更新部分画面。命令发送到 SPICE client 后，命令树与发送队列将会把该命令去除。当某条命令完成自己的任务，libspice 会把不需要的命令发送到 Release Ring，最后驱动根据 Release Ring 中的命令，去处理命令资源释放工作。SPICE 图像处理流程如下图 3-16 所示。

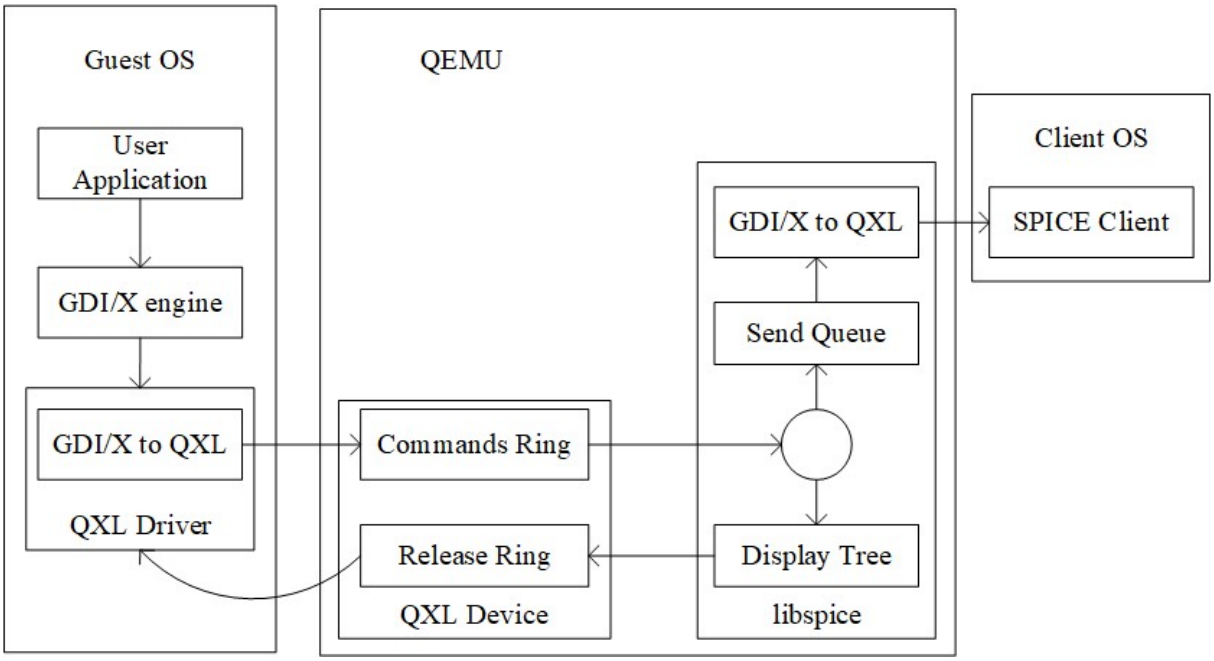

图 3-16 SPICE 图像处理流程

3.3.3 SPICE 协议不足之处分析

使用 SPICE 协议连接虚拟机，在虚拟机中播放高清视频，滑动网页，编辑 word，移动鼠标，经常能感受到画面卡顿、操作有时延等明显影响用户体验的现象。为了查明原因，需要对 SPICE 协议进行深入分析。

SPICE 协议对视频的压缩是采用的 MJPEG，它是一种静态图像压缩技术[45]。该压

缩技术的工作方式为只对帧内图像数据进行压缩，并没有考虑帧间传输图像关联性，我们都知道视频传输的相连帧之间有很多重复的数据，因此就无法避免造成过多的重复数据被传输，同时也是造成 MJPEG 压缩算法压缩率不高的原因。SPCIE 协议对播放视频的帧率设置是固定的，即每秒传输帧数为 30。

SPICE 协议对图像的压缩有多种图像压缩算法，包括 QUIC、LZ、GLZ。其中 LZ/GLZ 在合成图像中有较好的压缩效果，QUIC 对真实的图像压缩效果更好[46]。SPICE 协议可以动态的调整不同的压缩算法来适应不同的场景。采用渐进度算法对图像分析，来区分是文字类还是图像类图片，进而选用不同的图像压缩算法进行处理。渐进度算法不足之处就是会造成图像分类不合理选，用了不适合的图像压缩算法，导致压缩效果并不好，而且如果检测到是视频就会切换到视频压缩算法进行处理，这样经常的切换也会造成桌面抖动不流畅的现象出现。

经过对比分析，RDP 远程桌面协议有一个问题就是 qemu-kvm 并没有对 RDP 图形显示启动虚拟机的支持，因此采用间接方式，图形显示启动参数选择 SPICE 方式，开机后用 RDP 方式连接远程虚拟桌面，制作模板镜像过程也得选用 SPICE，因为 RDP 远程桌面服务是操作系统内服务，而制作模板镜像系统服务并未开启，即使开启了服务，也得需要完成连接前对操作系统内 RDP 的配置工作，才能使用远程桌面服务连接远程虚拟桌面。最终采用 RDP 协议与 SPICE 协议相结合的方式来实现办公桌面云系统。

3.4 本章小结

本章主要是对桌面云的构建底层所用到的技术进行深入的分析研究，主要有 QEMU-KVM 虚拟化技术，RDP 协议，SPICE 协议。深入分析了 KVM 架构以及对 CPU 虚拟化、内存虚拟化实现过程原理，QEMU 对 I/O 设备虚拟化原理，为了弥补 QEMU 的短板，KVM 引入了 VIRTIO 半虚拟化，并对 VIRTIO 架构实现原理深入的分析，最后是对办公桌面云系统所用的 RDP 协议与 SPICE 协议相结合的方式进行深入分析，并解答了为什么选择这两种协议来实现办公桌面云系统。为系统功能的实现奠定的理论基础，在下一章，将具体的介绍办公桌面云功能实现的具体细节。

第四章 办公桌面云系统的需求分析、设计与实现

前面几章节主要是对实现系统所需的技术做详细介绍，为系统的构建提供理论支撑。本章详细的讲解办公桌面云系统的设计与实现，项目的总体设计目标就是为了提高办公效率，降低企业成本。首先进行需求分析。

4.1 系统需求分析

4.1.1 背景介绍

在一个企业中，不同部门之间所从事的办公任务也不相同，不同的办公任务也就意味着所使用的办公软件也不尽相同。为了节省成本，对于企业采购来说，需要购买不同配置电脑满足不同部门办公需要。员工离职对所用电脑回收，也只能等本部门新人入职分配使用，设备不能在不同部门间流通，也就必然会导致设备的回收利用率不高，不能很好的缩减企业成本。每个部门也都有各自的部门主管。员工在规定的上班时间前打卡签到，之后打开自己电脑办公。对于运维人员，如果是新旧设备更换，根据部门提交的软件清单，还要进行大量的重复安装工作，对于一些比较专业的软件有时进行多次尝试才能够安装成功。对于企业来说，迫切的需要一种新的 IT 架构来满足发展的需求。

因此，对于办公桌面云系统需要实现的目标可以归纳为一下几点：

1) 高效的运维管理。

对于不同的部门制作不同的镜像，在镜像中安装部门需要软件作为初始镜像，用户使用的为初始镜像的派生镜像。这样对于每个部门来说，所需的软件只需安装一次，并不需要对所有用户安装，极大的缩减了安装时间。如果需要增加新的软件，也只需对初始镜像做安装操作。对于个别用户桌面故障，也只需简单的重新派发新的镜像。管理员在系统上就可以监控所有虚拟机的工作情况，对于一些突发问题也能做到及时响应。

2) 资源高效利用。

桌面云系统所消耗的资源全部来至于服务器，对客户终端来说只是提供桌面显示，只需消耗很少的资源，对本地电脑配置要救不高，这样就很好的利用到旧设备，提高终端设备的利用率，延长电脑使用年限。对于服务器端，动态分配虚拟机，让虚拟机均衡的分配在服务器集群，避免出现集群里服务器负荷高低失衡现象，对于不用的虚拟机进

行资源的释放，归还资源池，为资源高需求的虚拟机分配更多的资源，从而提高了服务器资源利用率。

3) 提高办公效率。

对于员工来说，虚拟机实现了桌面环境与硬件的解耦和。员工使用自己的桌面也并不需要在一个固定的地方，只需有一台连接网络的终端设备就能使用自己的办公桌面，提高了员工的办公效率。

4) 安全性。

用户的桌面环境以及数据全部集中存储在服务器端，解决的原先数据分散存储在各个终端设备上的不便。数据并不在终端设备存储，很好的避免了终端设备系统故障、中毒等造成数据丢失问题。

4.1.2 功能需求

办公桌面云系统功能需求绝大多数也都是为了管理员，对于一般用户只需要通过客户端登陆自己的虚拟桌面环境办公。员工信息管理，镜像管理，服务器集群管理，网络管理，部门管理等，这些都是管理员所需的功能，因此系统设置员工、管理员两个角色。

1. 管理员功能需求

1) 员工信息管理。

包括：创建员工信息、删除员工信息、查看员工信息、批量导入员工信息。

2) 镜像管理。

包括：为不同的部门制作模板镜像安装所需的软件，对于不使用的模板镜像进行删除，储存模板镜像，登陆模板镜像，已创建好的模板镜像如需添加新的软件可以进行修改操作。

3) 部门管理。

包括：创建部门、删除部门、查看部门信息、为部门添加员工、为部门设置部门负责人。

4) 网络管理。

包括：创建虚拟网络、删除虚拟网络、为部门虚拟机分配虚拟网络、对已分配的虚拟网络回收。

5) 集群管理。

包括：查看服务器运行状况以及硬件资源负荷、每台服务器上已经运行多少台虚拟机。

6) 虚拟机管理。

包括：虚拟机创建、虚拟机的分发、删除虚拟机、关闭虚拟机、启动虚拟机、查看虚拟机工作状态以及硬件使用情况。

7) 账号信息管理。

从安全的角度出发，对于账号使用者可以修改初始登陆密码。

管理员用例如图 4-1 所示

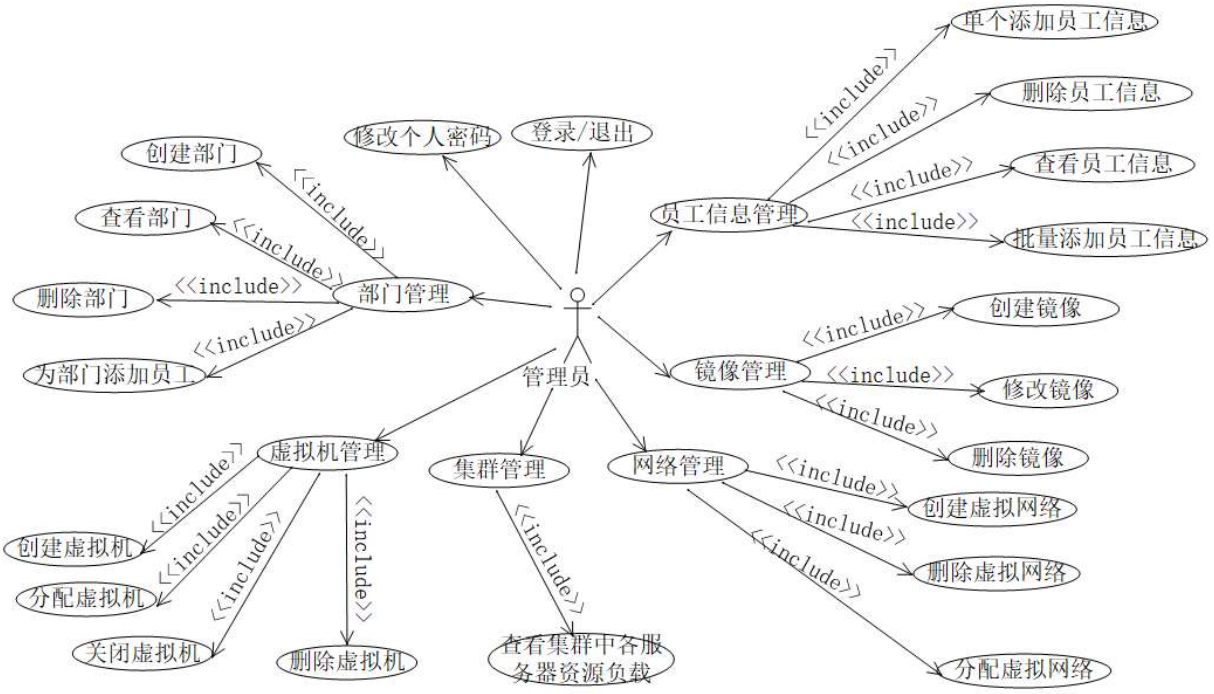

图 4-1 管理员用例图

2. 员工功能需求

员工功能需求与管理员功能相比简单。主要是为了登陆远程服务器上虚拟桌面环境进行办公，具体需求如下。

1) 员工使用自己的账号信息能够登录系统。

2) 员工可以查看系统为自己创建的所有的虚拟机。

3) 员工可以选择所需虚拟机登录进行办公，并且在虚拟机中可以使用 U 盘，虚拟桌面与本地终端数据等资源相互传输以及使用本地打印机设备。

4) 员工可以修改自己的密码，使用完虚拟机能够退出系统。用例图如 4-2 所示

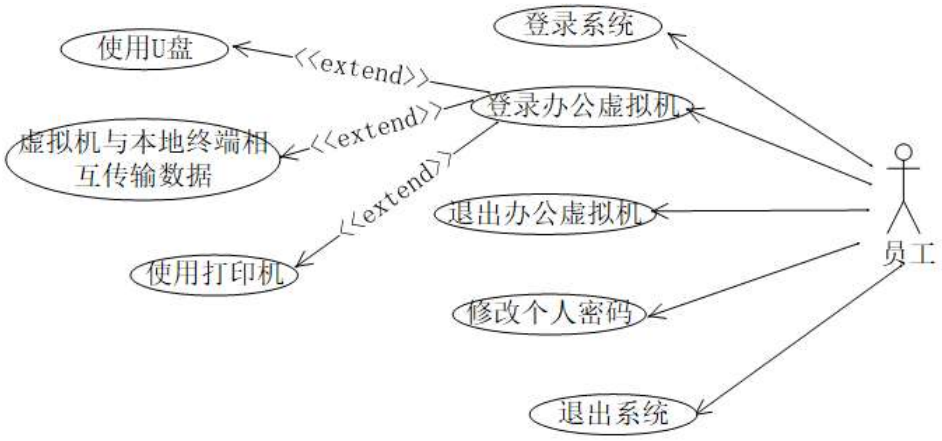

图 4-2 员工用例图

4.1.3 非功能需求

非功能需求主要是对系统的稳定性、易用性、安全、性能等方面的要求。

1) 稳定性。

对于任何软件来说稳定性都至关重要，用户使用虚拟机时保证虚拟机稳定运行，不会出现异常关机，保证用户使用流畅，不卡顿，也不会出现异常退出客户端，无法登陆虚拟机。对于一些突发情况如个别服务器宕机、断电等，也有应对之策，如虚拟机迁移、冗余备份等。

2) 易用性。

对于用户上手简单，经过简单的培训就能上手使用，不用对本地终端过多的配置，安装便捷，简洁的登陆操作，对于管理员各项管理功能流程简单，合理的划分功能模块便于用户使用。

3) 安全。

对于软件基本的安全是对使用平台的用户身份进行验证，防止非系统用户使用。对于不同员工开放不同权限，让用户在权限范围内正常使用。对于服务器端的虚拟机等数据资源应有冗余备份功能，防止用户虚拟机死机、中毒等情况造成数据丢失。

4) 性能。

保证用户使用虚拟机更好的用户体验，操作的流畅度，更快速的响应，让用户感觉像在操作本地终端。硬件方面，这就需要高性能的服务器为虚拟机运行提供资源支持，另外还需高效的实时传输，这就需要更好的网络带宽支持。软件方面，高效的调度资源，

对内存、CPU 虚拟化的优化，选用更好的桌面传输协议以及对传输协议优化等。

4.2 办公桌面云系统整体设计

4.2.1 办公桌面云系统整体物理架构设计

为了实现办公桌面云系统的各个模块的功能，针对办公的桌面云系统平台采用 C/S 整体设计架构，平台的整体物理架构如下图 4-3 所示。

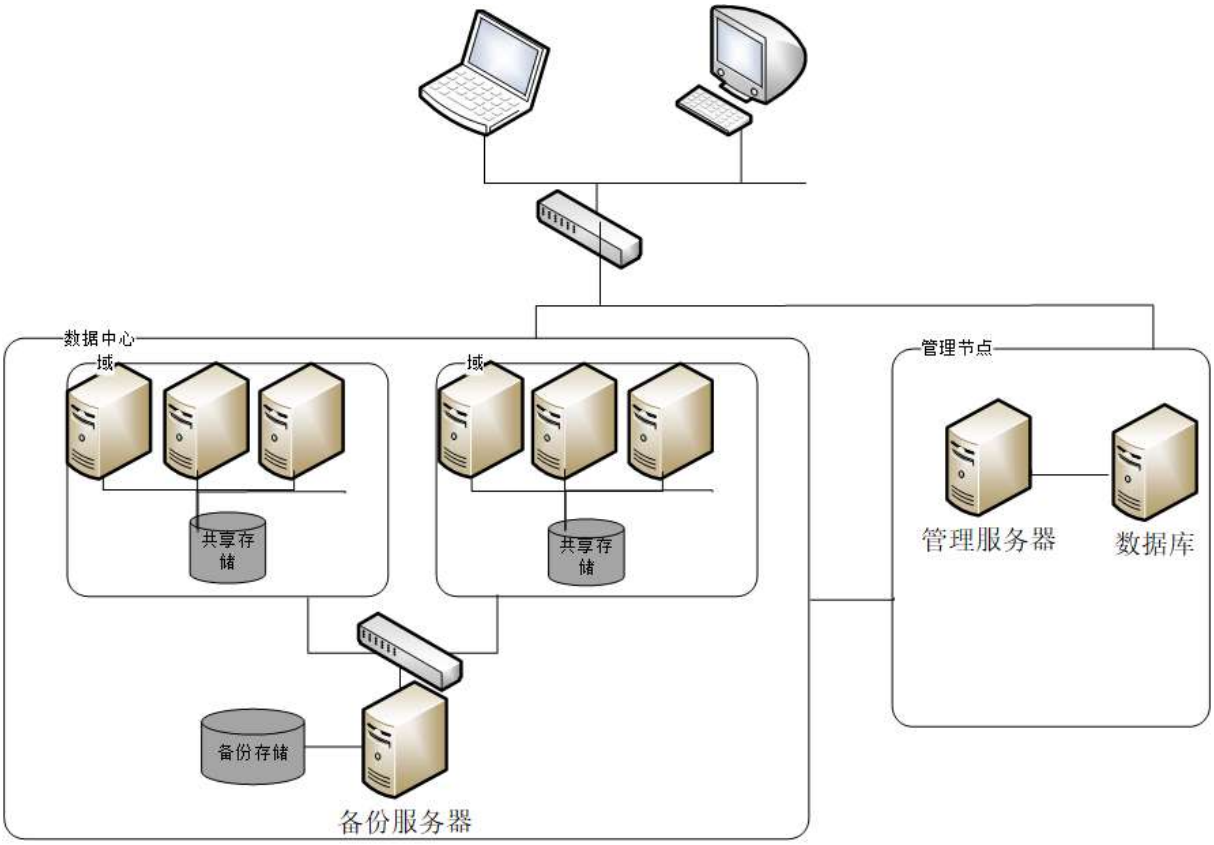

图 4-3 办公桌面云系统物理架构

办公桌面云系统由三部分构成：终端设备、管理节点、数据中心。

终端设备上装有办公云系统客户端，终端设备通过网络与管理节点、数据中心相连通。首先管理服务器对用户身份进行验证，之后进行各种业务交互。用户使用 RDP 客户端，通过 RDP 协议连接到自己的远程虚拟桌面进行办公操作。

管理节点不仅对客户端请求响应，还要对数据中心资源进行管理。为用户创建专属的虚拟机，并对虚拟机的整个生命周期进行管理，管理用户信息。将数据信息保存在数据库中。

数据中心为用户的虚拟机运行提供运行环境及所需的资源，并将虚拟机的镜像保存在共享存储里。虚拟化环境的实现依赖的是 QEMU-KVM 虚拟化技术，通过虚拟化技术对服务器资源抽象屏蔽硬件具体实现细节，让多台虚拟机同时运行在一台服务器上。为了数据的安全性，会在备份存储中保存副本数据。

4.2.2 办公桌面云系统逻辑架构设计

办公桌面云系统的逻辑架构设计如下图 4-4 所示，逻辑架构为 5 层设计，最上层为应用层，其次为服务接口层，再次为管理层，之后为虚拟化层，最底层为物理层。

图 4-4 办公桌面云系统逻辑架构

应用层直接与用户交互，通过调用下层服务接口实现用户办公桌面云所需的功能，满足用户的功能需求。比如：创建虚拟网络、添加用户信息、登录虚拟机镜像办公等。

服务接口层是为了实现应用层具体的系统功能需求提供标准的 API 接口供应用层调用，具体的实现是交由桌面云管理层相应的模块来完成的。

管理层是系统最重要的部分，业务功能最为繁重，不仅需要对整个系统的资源进行监控管理工作，还有完成对用户虚拟机的整个生命周期管理，同时还要及时响应上层服务接口层发起的调用请求任务。

桌面云虚拟化层采用虚拟化技术实现对底层服务器集群物理资源虚拟化工作，为用户虚拟机提供运行环境，满足上层管理层创建虚拟机所需的虚拟资源要求，如：虚拟 CPU、虚拟内存、虚拟硬盘等。该办公桌面云系统虚拟化技术采用的是 QEMU-KVM 虚拟化技术。

物理层为上层虚拟化层提供所需的硬件设备资源。包括：服务器资源、共享存储资源、网络设备资源。虚拟机所需的虚拟 CPU、虚拟内虚资源是服务器提供的。对于虚拟

机镜像文件的创建所需的资源使用的共享存储。

4.2.3 办公桌面云系统功能结构设计

办公桌面云系统总体功能设计分为五个模块，分别是企业信息管理模块、监控模块、镜像管理模块、网络管理模块、个人功能模块。办公桌面云系统总体功能结构如图 4-5 所示。

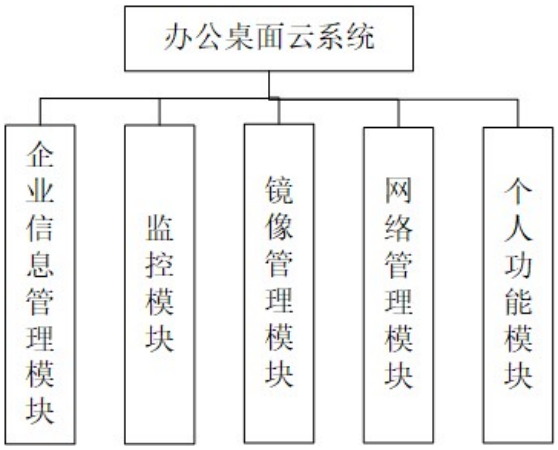

图 4-5 系统功能结构

企业信息管理模块包括部门信息以及员工信息管理。部门信息管理主要负责部门信息维护管理工作，包括添加部门、查看部门信息、删除部门以及为部门导入员工等操作。员工信息管理模块主要负责员工信息维护管理工作。包括员工信息添加、查看、删除等操作。

监控模块主要是负责对集群中服务器节点的运行状态信息实时监控以及服务器上运行的员工的虚拟桌面镜像状态监控工作，通过监测服务器指标及时发现服务器是否有异常情况，系统是否需要扩容。通过监测虚拟机的运行，如有异常情况也能及时处理。

镜像管理模块主要工作有根据部门的需要创建合适的模板镜像，在模板镜像的基础上为部门员工创建派生镜像，部门员工登陆自己专属的镜像进行办公，因部门需要对模板镜像进行修改操作等。

网络管理模块主要职责是为员工虚拟机提供网络环境让员工能够访问网络。主要功能有虚拟网络的创建、分配、删除等操作。

个人功能模块主要职责为确保员工以及管理员登陆桌面云系统、退出桌面云系统以及修改自己的密码，满足员工使用 U 盘、打印机等需求。

4.2.4 系统类图设计

办公桌面云系统服务端设计的主要类以及该类的说明如表 4-1 所示。

表 4-1 系统服务端主要类

类	说明
DomainShell	用于处理虚拟机的相关业务逻辑，包括创建虚拟机、创建镜像、安装操作系统等。
PacketWrapper	包含所有与客户端进行 socket 通信的细节，用于后台向客户端发送响应。
Schedule	用于完成虚拟机和虚拟资源的调度和分配
StaffAction	用于处理员工相关的业务逻辑，包括员工登录、修改密码、登录办公镜像等。
AdminAction	用于处理管理员相关的业务逻辑。包括用户信息管理、部门信息管理、虚拟网络管理、镜像管理等
HostAction	主要用于维护服务器节点信息，包括新节点添加、节点退出更新操作等
Session	由 StaffAction、AdminAction 调用，负责维护成功登陆的用户的会话信息。
DBO	包含了所有数据库相关的业务逻辑
Service	主要处理用户与服务器节点的请求，并将请求分发到相应的 Action 类处理
NetworkXml	用于动态生成虚拟网络 xml 配置信息，然后用 libvirt 创建虚拟网络时根据动态生成的 xml 信息进行创建
DomainXml	用于动态生成虚拟机的 xml 配置信息，然后用 libvirt 创建虚拟机时根据动态生成的 xml 信息进行创建

类之间的关系如图 4-6 所示

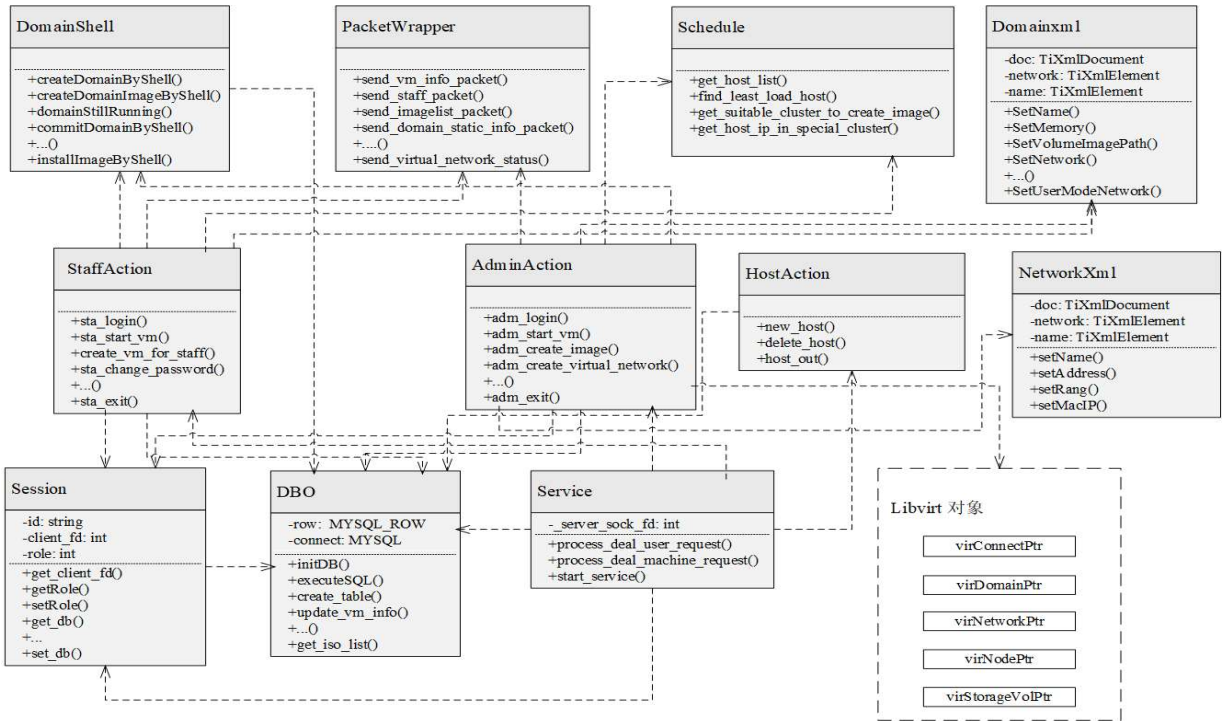

图 4-6 系统类图

Libvirt 库是高效的管理虚拟机的应用程序接口，在上一章 3.1.4 小节中也对 Libvirt 做相应的介绍。要使用 libvirt 提供的丰富的虚拟化管理 API，需要建立与 VMM 连接。本系统底层虚拟化技术采用的 QEMU-KVM 虚拟化技术。Libvirt 也提供了多种方式连接 QEMU，本系统采用的远程 URI 的实现方式。而虚拟机创建、虚拟网络创建/删除、对服务器节点监控以及对虚拟机监控等功能底层实现是通过调用 libvirt 提供的相应的 API 实现的。这些功能的具体实现下文中有介绍。

4.3 功能模块设计与实现

本节是对办公桌面云系统功能各个模块进行详细的设计与实现。

4.3.1 企业信息管理模块设计与实现

企业信息管理模块主要涉及系统最基本的信息的创建工作。对于员工信息的添加、删除操作与部门信息的添加、删除操作，两者设计实现过程大致上相同，在这里只对员工信息的添加介绍，添加员工信息的流程图如图 4-7 所示。

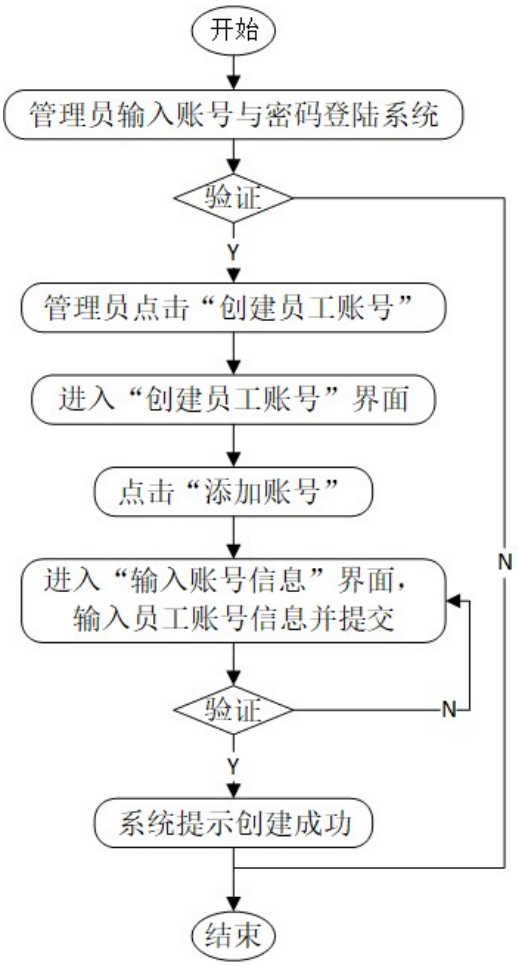

图 4-7 添加员工流程

办公桌面云系统架构采用的 C/S 架构，客户端采用 C++程序编写，使用 Qt 完成客户端 UI 用户交互界面设计，前端与后台程序通过 socket 进行通信，后端采用 C++程序编写。对于员工信息的添加业务，前端只是获取用户输入的数据，将数据传输到服务端调用 AdminAction 类的 adm_create_staff()方法创建员工函数执行添加员工信息到数据库。实现添加员工信息的服务端核心代码如下表 4-2 所示：

表 4-2 添加员工信息功能服务端核心代码实现

```
int AdminAction::adm_create_ staff (const string &staff_id,const string &name, const
string &password,DBO * db){
    int ret = OPERATION_FAILURE;
    Staff staff(staff_id,name,password,"0");
    if(db->insert_staff(staff)){ret = OPERATION_SUCCESS;}
     return ret;}
```

对于企业信息管理模块还用到了 AdminAction 类的方法有见下表 4-3 所示。

表 4-3 企业信息管理模块用到 AdminAction 类的方法

方法	作用
adm_check_staff_list()	查询员工列表
adm _delete_ staff ()	删除指定员工
adm _create_dapartment()	创建部门
adm_query_dapartment()	查询部门列表
adm_import_staff_for_dapartment()	为部门添加员工
adm_delete_dapartment()	删除部门信息
adm_check_stafflist_by_dapartmentid()	查询部门员工列表

表 4-3 中所提的方法的具体代码实现就不再具体展示。

4.3.2 监控模块设计与实现

监控模块主要的职责是对服务器的运行状态监控，还有对员工的虚拟机运行状态的监控。对于服务器来说，主要监控的指标分为两类：服务器基本的硬件配置信息，运行的服务器硬件资源负荷。管理员通过监测服务器运行指标，能够较早的发现异常服务器，并对异常服务器进行处理。对员工的虚拟机运行状态监控，主要的的监控指标与服务器基本相同。监控集群服务器节点和员工虚拟机功能实现过程相似，都是服务端节点通过调用 libvirt 提供的接口获取服务器节点以及虚拟机实时状态信息。因此在这里以监控虚拟机为例，虚拟机监控的流程图如图 4-8 所示。

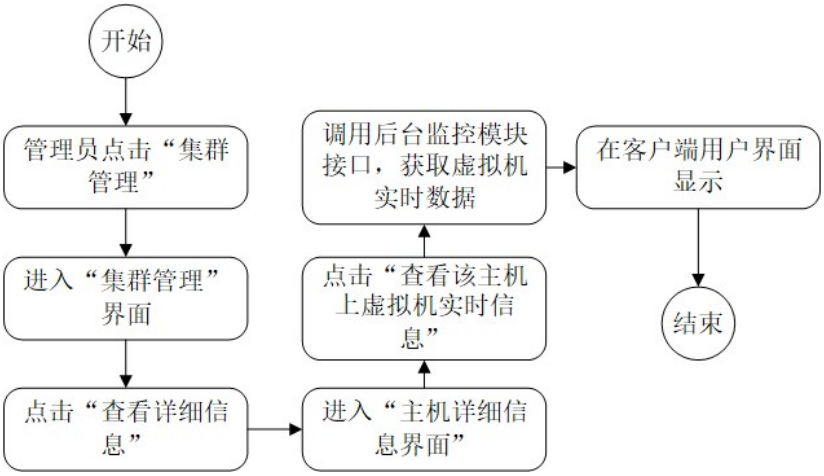

图 4-8 虚拟机监控流程图

对虚拟机监控的实现，libvirt 是开源的虚拟化管理工具，libvirt 对于底层虚拟化方

案的实现细节有很好的屏蔽作用，为上层的管理工具提供统一的应用程序接口，其中virDomainPtr 接口实现的功能就是实现对虚拟机的管理。因此通过调用该接口提供的方法来获取员工虚拟机资源配置以及虚拟机实时的硬件资源负荷数据。对于虚拟机的静态资源配置信息，libvirt 是通过 XML 文件形式存储的，因此系统获取员工虚拟机配置的虚拟机的名称、虚拟机的 CPU 数、虚拟机的内存、虚拟机的操作系统等信息是通过调用 libvirt 提供的 virDomainGetXMLDesc 方法得到员工虚拟机的 XML 配置信息，之后对 XML 文件进行解析得到员工虚拟机的静态配置参数。

对于虚拟机内存负荷、CPU 负荷等动态，分别是通过调用 libvirt 的 virDomainGetInfo、virDomainGetCPUStats 方法获得。获取虚拟机内存负荷的核心代码实现如下表 4-4 所示：

表 4-4 获取虚拟机内存负荷的核心代码

```
int AdminAction:: GetDomainMem(){
    …
    int ret=virDomainGetlnfo(dom, &domainlnfo);
    mem.totalMem=domainlnfo.maxMem/1024;
    mem.curMem=domainlnfo.memory/1024;
    mem.freeMem = mem.totalMem - mem.curMem;
    ....
}
```

对于监控模块还用到的 libvirt 的方法见表 4-5 所示。

表 4-5 监控模块用到的 libvirt 其他的方法

方法	作用
virConnectGetHostname()	获取主机名
virConnectGetMaxVcpus()	获取最大虚拟 CPU 个数
virNodeGetFreeMemory()	获取节点空闲内存
virNodeGetInfo()	获取节点硬件信息

4.3.3 镜像管理模块设计与实现

镜像管理模块主要功能有：员工登录自己的镜像进行办公操作，管理员根据部门需求制作该部门的模板镜像，管理员对模板镜像进行修改满足企业新的需求。

1. 员工登录自己的镜像进行办公操作，流程图如图 4-9 所示。

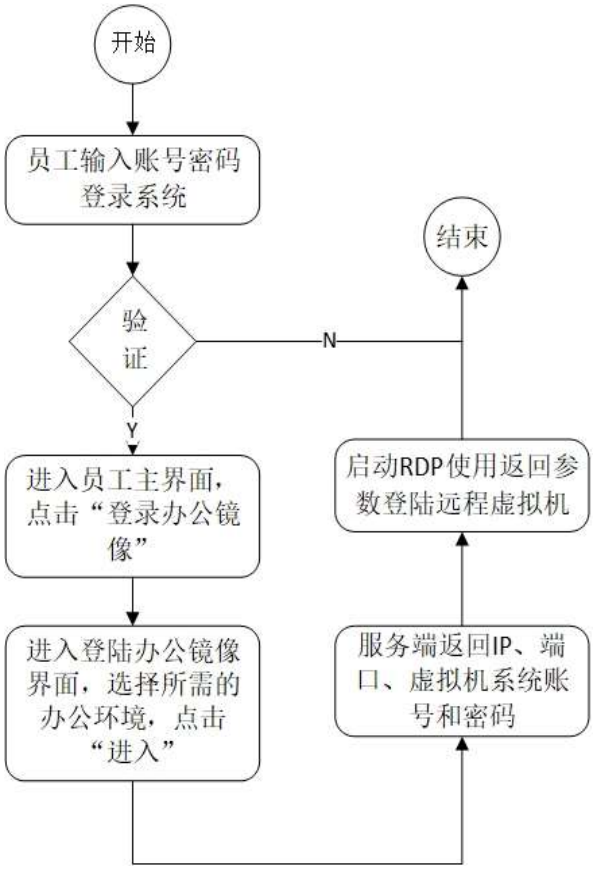

图 4-9 员工登陆办公镜像流程

员工登陆办公镜像功能实现过程如下：

1) 员工发起登陆虚拟机镜像请求。
2) 系统服务端首先检查数据库 virtualmachine 表，查询是否为该员工创建所需虚拟机记录。
3) 如果记录不存在，服务端调用 DomainShell 类提供的 createDomainImageByShell 方法为该员工派生所需的镜像，并更新数据库 virtualmachine 表添加该条记录。调用 DomainShell 类提供的 createDomainByShell 方法利用创建好的派生镜像启动虚拟机，主服务端将该 IP、端口、虚拟机系统用户名以及密码发送给客户端，客户端使用 RDP 登陆远程虚拟机。

4) 如果记录存在，调用 DomainShell 类提供的 domainStillRunning 方法判断虚拟机是否在运行。如果员工虚拟机在运行，主服务端将该 IP、端口、虚拟机系统用户名以及密码发送给客户端，客户端使用 RDP 登陆远程虚拟机。如果不在运行，服务端调用 DomainShell 类提供的 createDomainByShell 方法利用创建好的派生镜像启动虚拟机，主服务端将该 IP、端口、虚拟机系统用户名以及密码发送给客户端，客户端使用 RDP 登陆远程虚拟机。主要函数实现如表 4-6 所示：

表 4-6 员工登录镜像功能的主要函数实现

```
int DomainShell::createDomainImageByShell(const string &template_image,
        const string &sub_image, const string &max_image_size,
        const string &clusterid, DBO * db) {
    string dest_ip = Schedule::get_host_to_do_create_and_commit_job(clusterid,
            db);
    string cmd = "ssh " + dest_ip + " -l root " + "\"qemu-img create -f qcow2 "
            + SUB_IMAGE_PATH + sub_image + " -o backing_file="
            + TEMPLATE_IMAGE_PATH + template_image + " " +
max_image_size + " &\" &";
    system(cmd.c_str());
while (!derivedImageHasCreated(dest_ip, sub_image)) {
        …….
    }
    db->update_host_vmnumber_by_image_add(dest_ip);
    db->update_cluster_vmnumber_by_image_add(clusterid);
    cout << "Derived image created!" << endl;
    return 1;}
```

对于创建员工虚拟机服务器节点的选择采用了动态负载均衡算法。主要是为了解决集群中服务器节点负载不均，更好的利用服务器资源。该算法实现过程为遍历集群服务器，获取服务器的内存利用率，选择其中内存利用率最小的服务器节点为创建员工虚拟机所需的服务器。

2.管理员为部门制作模板镜像，镜像制作过程如下图 4-10 所示

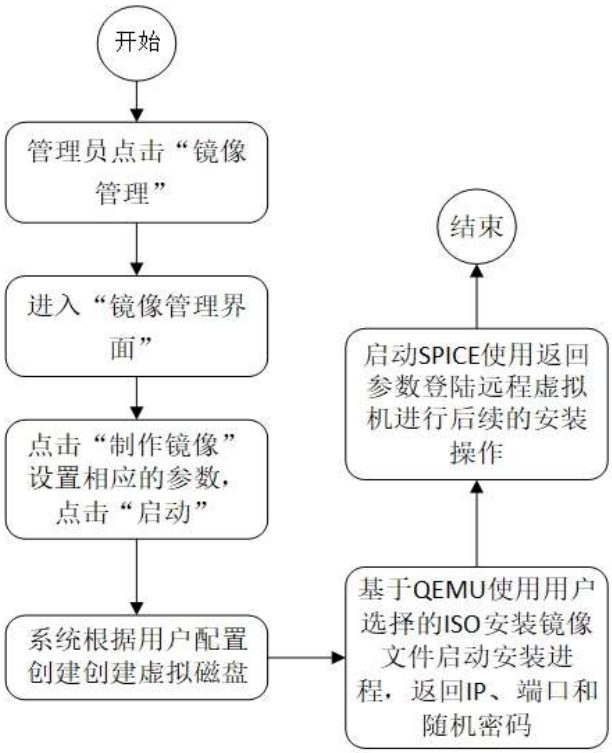

图 4-10 管理员制作镜像流程

管理员为部门制作模板镜像功能实现过程如下：

1) 管理员发起镜像制作请求。
2) 服务端检查数据库 templateimage 表，查看模板镜像名字是否重名，如果重名向客户端返回错误信息。如果不重名执行后续操作。
3) 将创建的模板镜像信息记录插入 templateimage 表，获取一个可用端口和一个随机密码。
4) 将准备启动的虚拟机记录插入数据库 adminvirtualmachine 表，并设置镜像状态为正在安装。
5) 调用 DomainShell 类提供的 createImageByShell 方法创建模板镜像的虚拟磁盘。
6) 调用 DomainShell 类提供的 installImageByShell 方法，启动 iso 系统映像安装进程。
7) 主服务端将虚拟机的 IP、port 和随机密码传送给客户端，客户端启动 SPICE 进行具体的安装过程，用户关闭 SPICE，确认安装完毕。服务端更新镜像状态为安装完成。主要函数实现如表 4-7 所示：

表 4-7 管理员制作镜像主要函数实现

```
int DomainShell::createImageByShell(const string &imagename,
        const string &disk, const string &clusterid, DBO * db) {
    string host_ip = Schedule::get_host_to_do_create_and_commit_job(clusterid,
            db);
    string cmd = "ssh " + host_ip + " -l root " + "\"qemu-img create -f raw "
            + TEMPLATE_IMAGE_PATH + imagename + " " + disk + " &\" &";
    system(cmd.c_str());
    while (!templateImageHasCreated(host_ip, imagename)) {
    ….
    }
    cout << "Template image created!" << endl;
    return 1;
}

int DomainShell::installImageByShell(const string &host_ip,
        const string &imagename, const string &memory, const string &iso,
        const string &port, const string &password) {
        string cmd = "ssh " + host_ip + " -l root " + "\"qemu-kvm -hda "
            + TEMPLATE_IMAGE_PATH + imagename +" -m " + memory+"G"
            + " -boot d -cdrom " + ISO_IMAGE_PATH + iso + "-spice port="
            + port + ",password=" + password + ",disable-ticketing &\" &";
    while (!domainStillRunning(host_ip, port, imagename)) {
    ….
    }
    system(cmd.c_str());
    cout << "Installation started!" << endl;
    return 1;
}
```

3.管理员修改部门模板镜像，主要是为了满足部门提出的新需求，比如系统软件更新。对部门模板镜像修改过程如图 4-11 所示。

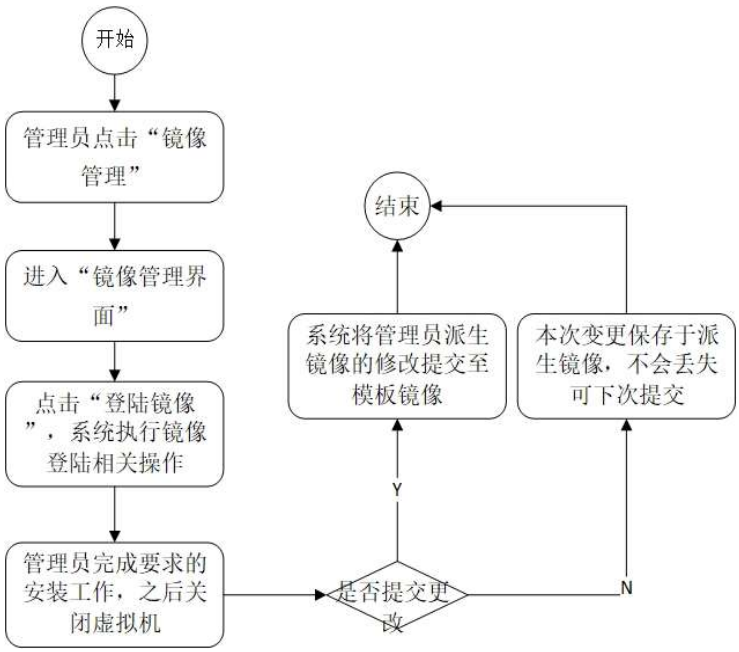

图 4-11 模板镜像流程图

部门模板镜像修改功能实现过程如下：

1) 主服务端进行管理源登录镜像做更新操作，关闭虚拟机。
2) 是否保存修改，如果选择不保存，则更改操作保存在派生镜像中，可以下次在提交
3) 如果选择保存，服务端调用 DomainShell 类提供的 commitDomainByShell 方法，将修改提交到模板镜像中。主要的函数实现如表 4-8 所示：

表 4-8 修改镜像主要函数实现

```
int DomainShell::commitDomainByShell(const string &sub_image,
        const string &clusterid, DBO * db) {
    string dest_ip = Schedule::get_host_to_do_create_and_commit_job(clusterid,db);
    string cmd = "ssh " + dest_ip + " -l root " + "\"qemu-img commit -f qcow2 "
            + SUB_IMAGE_PATH + sub_image + " &\" &";
    system(cmd.c_str());
    cout << "Change committed!" << endl;
    return 1;}
```

镜像管理模块用到的其他类的方法如表 4-9 所示。

表 4-9 镜像管理模块用到的其他类的方法

方法	作用
Schedule::find_least_load_host()	选择集群中负载最小的服务器作为调度目标
DomainShell::killProcessByShell()	根据虚拟机的端口号强行关闭虚拟机
virConnectOpen()	建立连接
DomainXml()	动态生成虚拟机的 xml 配置信息
virDomainCreateXML()	创建虚拟机
virDomainGetName()	获取虚拟机名
virConnectClose()	断开连接
AdminAction::adm_create_image()	管理员创建镜像
AdminAction::adm_commit_image()	管理员提交镜像
AdminAction::adm_finish_image()	管理员完成镜像制作

4.3.4 网络管理模块设计与实现

网络管理模块主要是实现员工的虚拟机能够正常的使用网络服务，功能有虚拟网络创建、虚拟网络分配、虚拟网络删除。

对于虚拟机 MAC 地址的生成设计一种算法，我们知道每个虚拟机都分配有独有的虚拟 IP 地址。MAC 地址为 48 位，通常是用 16 进制表示，刚好是 12 个，6 段。而 IP 地址为 32 位，转换为 16 进制，需要 8 个，4 段。因此，该算法让 IP 地址映射到 MAC 地址的后四段，前面两段用固定的值拼接，生成如 IP 地址一样，独有的 MAC 地址。该算法实现大致过程为：每次创建虚拟网络时，进行 MAC 地址的创建，对用户提交的虚拟 IP 地址判断属于哪一类地址，之后进行相应的转换操作，完成 MAC 地址创建。

1.虚拟网络创建过程如图 4-12 所示。

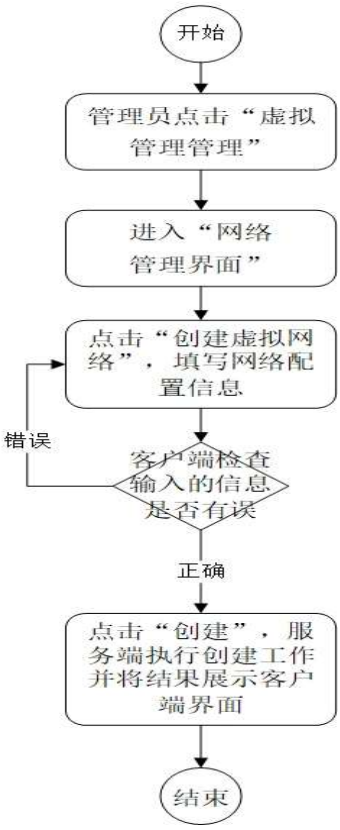

图 4-12 虚拟网络创建流程图

虚拟网络创建功能是通过调用 libvirt 库来实现为系统中虚拟机创建虚拟网络，实现过程如下：

1) 服务端从数据库 vnetwork 表获取本次虚拟网络创建所需的参数。如 netname, subnet,netmask 等。
2) 调用 NetworkXml 类并配置参数。
3) 从数据库 host 表获取集群中所有服务器 IP 地址。
4) 调用 libvirt 库提供的 virNetworkCreateXML 对象完成虚拟网络创建。
5) 在服务器上完成配置工作。第一步为添加 vlan 接口，第二步启动 vlan 接口，第三步将 vlan 接口加入网桥中。服务端实现功能函数的核心代码如表 4-10 所示：

表 4-10 创建网络核心代码

```
int AdminAction::adm_create_virtual_network(const string &netname, DBO
*db,string &failedresason)
{
```

表 4-10 创建网络核心代码(续)

```
if(db->get_vnet_ip_and_rang(netname, subnet,netmask, startip, endip, vlan))
        {
        netXml.setAddress(subnet.c_str(),netmask.c_str());
        netXml.setRang(startip.c_str(), endip.c_str());
        }
        ….
       if(db->get_hosts_ip(hostiplist))
       {
       for (unsigned int i = 0; i < hostiplist.size(); i++)
       {
       string hostip = hostiplist[i];
       string uri = "qemu+tcp://root@" + hostip + "/system";
       virConnectPtr connection = virConnectOpen(uri.c_str());

       virNetworkCreateXML(connection, printer.CStr())
       char * bridgeName = virNetworkGetBridgeName(net);
       string vconfig="ssh "+hostip+" -l root "+"\"vconfig add "+nic+" "+vlan+"&\"&";
       string ifconfig="ssh "+hostip+" -l root "+"\"ifconfig "+nic+"."+vlan+" up&\"&";
       string brctl="ssh "+hostip+" -l root "+"\"brctl addif "+bridgeName+" "+nic+"."+
        vlan+" &\"&";
       system(vconfig.c_str());
       system(ifconfig.c_str());
      system(brctl.c_str());
       …
      }
```

2.虚拟网络分配过程比较简单，将创建好的虚拟网络分配给部门已创建的虚拟机，在这里不做过多的介绍。

3.虚拟网络删除过程如图 4-13 所示。

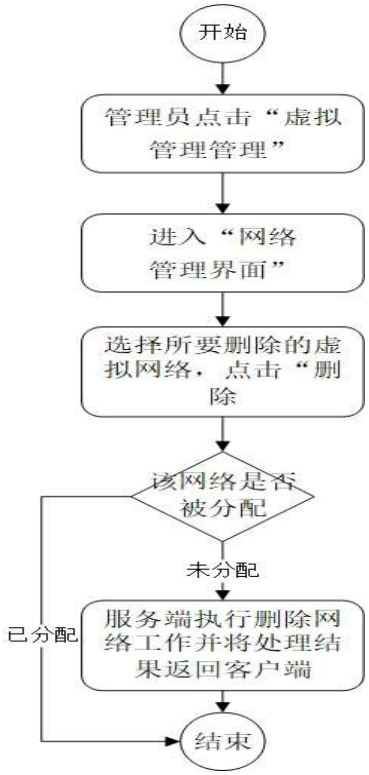

图 4-13 虚拟网络删除过程

虚拟网络的删除与虚拟网络创建一样都需要 libvirt 库来实现，实现过程如下：

1) 服务端从数据库 host 表中获取服务器 IP 地址。
2) 解除服务器上虚拟网络配置。第一步删除加入网桥中的 vlan 接口，第二步关闭 vlan 接口，第三步删除 vlan 接口。
3) 调用 libvirt 提供的 virNetworkDestroy 方法完成虚拟网络删除。服务端实现功能函数的核心代码如表 4-11 所示：

表 4-11 删除虚拟网络的核心代码

```
int AdminAction::adm_destroy_virtual_network(const string &netname, DBO
*db){
vector<string> hostiplist;
if(db->get_hosts_ip(hostiplist)){
for (unsigned int i = 0; i < hostiplist.size(); i++){
string hostip = hostiplist[i];
….
virNetworkPtr net = virNetworkLookupByName(connection, netname.c_str());
```

表 4-11 删除虚拟网络的核心代码(续)

```
    char * bridgeName = virNetworkGetBridgeName(net);
    cout<<"bridgeName "<<bridgeName<<endl;
    ….
    string vconfig="ssh "+hostip+" -l root "+"\"vconfig rem "+nic+"."+vlan+"&\"&";
    string ifconfig="ssh "+hostip+" -l root "+"\"ifconfig "+nic+"."+vlan+" down&\"&";
   string brctl="ssh "+hostip+"          -l root "+"      \"brctl delif "+bridgeName+"
   "+nic+"."+vlan+" &\"&";
    system(brctl.c_str());
    system(ifconfig.c_str());
    system(vconfig.c_str());
    virNetworkDestroy(net);
    …..
}
```

对于网络管理模块用到的其他类的方法见表 4-12 所示

表 4-12 网络管理模块用到的其他类的方法

方法	作用
AdminAction ::adm_get_virtual_network_status()	管理员查看虚拟网络状态
virConnectOpen()	建立连接
virNetworkLookupByName()	获取虚拟网络名
virNetworkIsActive()	查询网络是否正在使用

4.3.5 个人功能模块设计与实现

个人功能模块主要功能有登录系统、退出系统以及 U 盘、打印机的使用等。

登录过程为：以员工为例，员工输入用户名以及密码，服务端查询数据库 staff 表，对用户信息验证，验证通过进入系统，并更改数据库 staff 表中用户的状态为在线状态。

退出过程为：员工点击退出，服务端更改数据库 staff 表中用户的状态为下线状态。

对于 U 盘、打印机设备的使用是通过 RDP 启动配置文件实现的。与 U 盘相关的参数为 drivestoredirect，打印机设备相关的参数为 redirectprinters。

个人功能模块所用到的类的方法见表 4-13 所示。

表 4-13 个人功能模块所用到的类的方法

方法	作用
AdminAction::adm_login()	管理员登陆
AdminAction::adm_exit()	管理员退出
AdminAction::adm_change_password()	管理员修改密码
StaffAction::sta_login()	员工登陆
StaffAction::sta_exit()	员工退出
StaffAction::sta_change_password()	员工修改密码

4.4 系统中桌面传输协议登录虚拟机功能的实现

办公桌面云系统中，需要使用桌面传输协议登陆虚拟机操作的有两个部分：第一部分为员工使用虚拟机做相关的办公操作；第二部分为管理员管理镜像，包括制作镜像、修改镜像，都需要连接远程虚拟机进行操作。对用户体验影响最大的地方为第一部分，这部分使用 RDP 连接远程虚拟桌面；第二部分管理员制作镜像、修改镜像使用 SPICE 连接。首先介绍 RDP 登录虚拟机功能的实现。

4.4.1 RDP 登录虚拟机功能的实现

1. 远程桌面服务配置

RDP 客户端是 Windows 操作系统系统软件，这也是 RDP 的一个优势，不用像 SPICE 协议需要单独安装。Windows 操作系统远程桌面服务是系统默认开机启动项，但并不能直接使用，需要自己对远程桌面服务做连接前的参数配置。RDP 的使用只需对远程桌面服务的参数做简单的配置，用户就能连接远程服务器上运行的虚拟桌面。

下面以 win10 系统为例，介绍 RDP 实现远程连接之前的设置工作。首先鼠标右键

我的电脑，选择属性，高级系统设置，在远程栏，远程协助勾选“允许远程协助连接这台计算机”以及远程桌面设置中“允许远程连接到此计算机”如图 4-14 的设置

图 4-14 系统属性设置

之后在本地组策略中，找到“网络访问：本地账户的共享和安全模型”，鼠标右键选择“属性”，在本地安全设置栏中做如图 4-15 的选择。完成配置工作。

图 4-15 本地账户的共享和安全模型设置

之后就是启动 RDP 客户端连接远程桌面。RDP 客户端如下图 4-16 所示

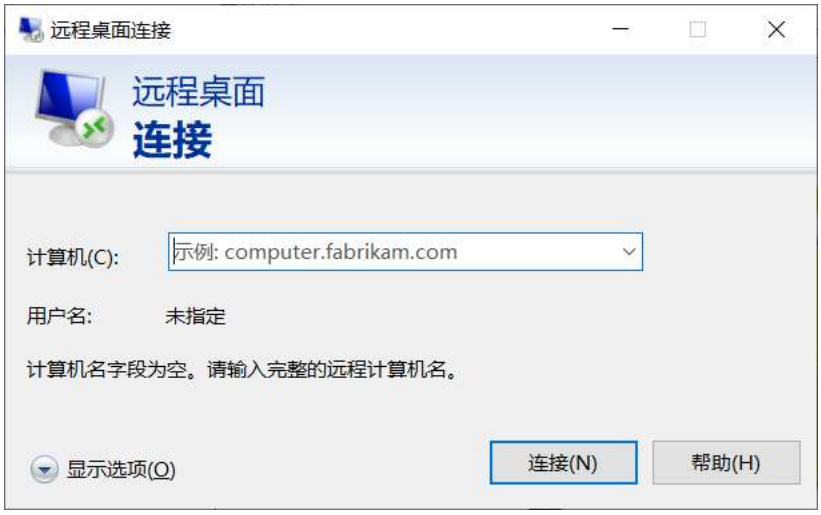

图 4-16 RDP 客户端

2.前端代码实现

客户端是使用的 C++编程，借助 Qt 实现 UI 用户交互界面。对于客户端使用 RDP，就是对 RDP 协议客户端(Microsoft terminal services client, MSTSC)软件调用。服务端创建的虚拟机都分配有虚拟 IP，但是该虚拟 IP 是无法从外界直接访问，为了实现 RDP 连接虚拟机，需要将服务器的一个端口映射到虚拟机的 3389 端口(该端口为 Windows 默认的远程服务端口)，通过访问服务器特定端口实现访问内部虚拟机远程连接。后台从数据库表获取远程虚拟机的 IP 地址、用户名及密码，将这些数据传输 RDP 客户端实现远程连接功能。

前端实现调用本地 RDP 连接远程虚拟机部分核心代码如表 4-14 所示。

表 4-14 RDP 连接虚拟机前端核心代码

```
void StartVMTable::sendStartVMRequest(const QString &categoryid){
SocketWrapper::fromInstreamToBuffer(in,buffer,packet_length);
QString ip = strtok(buffer,PACKET_DELIMITER_STR);
QString port = strtok(NULL,PACKET_DELIMITER_STR);
QString vmname = strtok(NULL,PACKET_DELIMITER_STR);
QString vmpassword = strtok(NULL,PACKET_DELIMITER_STR);
QString program = "rdp.cmd";
QStringList arguments;
```

表 4-14 RDP 连接虚拟机前端核心代码(续)

```
QString id=ip+":"+QString::number(port);
arguments << id<< vmname<< vmpassword;
QProcess *myProcess = new QProcess(0);
myProcess->start(program,arguments);
myProcess->waitForFinished();
// 使用 rdp 文件连接虚拟机
rdpProcess->start("mstsc",QStringList()<<"computerA.rdp");}
```

其中，ip 为虚拟机所在服务器的 ip。port 为服务器的端口。" rdp.cmd " 为 windows 批处理文件，目的是为了生成 RDP 客户端连接设置文件"computerA.rdp"。" rdp.cmd "主要执行的命令如表 4-15 所示。

表 4-15 rdp.cmd 配置文件

```
:: Variables:
:: "hashtool"      - location of the hash tool
:: "outputfile" - destination and name for the .rdp file
:: "comp"           - computer name (can be FQDN or NetBIOS)
:: "domain"         - name of authenticating domain
:: "usr"            - Username
@echo Off
If "%1"=="" Goto EOF
If "%2"=="" Goto EOF
If "%3"=="" Goto EOF
set pwd=%3
set hashtool="cryptRDP5.exe"
set outputfile="computerA.rdp"
set comp=%1
set usr=%2
```

表 4-15 rdp.cmd 配置文件(续)

```
for /f "tokens=*" %%a in ('%hashtool% %pwd%') do set pwdhash=%%a
:CreateRDP
If EXIST %outputfile% del %outputfile%
Echo full address:s:%comp%>> %outputfile%
Echo redirectprinters:i:1>> %outputfile%
Echo redirectcomports:i:1>> %outputfile%
Echo redirectsmartcards:i:1>> %outputfile%
Echo redirectclipboard:i:1>> %outputfile%
Echo redirectposdevices:i:0>> %outputfile%
Echo redirectdirectx:i:1>> %outputfile%
Echo devicestoredirect:s:*>> %outputfile%
Echo drivestoredirect:s:*>> %outputfile%
Echo username:s:%usr%>> %outputfile%
Echo password 51:b:%pwdhash%>> %outputfile%
:EOF
```

为了对用户虚拟机密码保护，从后端传递的虚拟机密码，通过 cryptRDP5.exe 进行加密处理。RDP 参数设置介绍如表 4-16 所示。

表 4-16 RDP 参数设置说明

参数	说明
redirectprinters:i:1	本地计算机的打印机可以在远程桌面使用
redirectcomports:i:1	本地计算机的 COM 端口可以在远程桌面使用。该端口通常用于鼠标、键盘的连接
redirectsmartcards:i:1	本地计算机的智能卡设备可以在远程桌面使用
redirectclipboard:i:1	本地计算机的剪贴板可以在远程桌面使用
redirectdirectx:i:0	本地计算机的 DirectX 可以在远程桌面使用。增强图形和声音效果。

表 4-16 RDP 参数设置说明（续）

devicestoredirect:s:*	本地计算机的可插拔设备可以在远程桌面使用
drivestoredirect:s:*	本地计算机的硬盘设备可以在远程桌面使用

3.后端代码实现

办公桌面云系统对于创建的虚拟机的网络配置选用的基于 NAT 的虚拟网络模式，分配给虚拟机的 IP 为内部 IP 地址，外部不可见，为了实现对外部可见，可将虚拟机的某些服务端口映射到服务器的某个端口。后端设计主要是实现端口转发，将服务器的一个端口映射到内部虚拟机的远程服务 3389 端口，从而实现用户访问远程虚拟桌面。这里先介绍一个实现端口转发的工具 Rinetd。

Rinetd 安装过程如表 4-17 所示：

表 4-17 Rinetd 安装过程

```
wget http://www.boutell.com/rinetd/http/rinetd.tar.gz
tar zxvf rinetd.tar.gz
cd rinetd
mkdir -p /usr/man/man8
yum install gcc
make && make install
```

安装完成之后，直接在 rinetd.conf 文件中添加端口转发规则，按照给定的格式设置，该格式为：[bindaddress] [bindport] [connectaddress] [connectport]，bindaddress 为服务器 IP 地址，bindport 服务器的端口，connectaddress 内部虚拟机的 IP，connectport 虚拟机的端口。最后执行/usr/sbin/rinetd -c /etc/rinetd.conf 就实现了端口转发。

服务器端口的分配采用全局分配策略，为了避免与其他端口产生冲突，设置端口以 50000 作为起始值，获取端口主要代码如表 4-18 所示：

表 4-18 获取用于转发的服务器端口

```
sql_str = "select    rdpport from virtualmachine order by rdpport asc;";
int t = mysql_query(connection, sql_str.c_str());
if (t) {
    printf("Error making query: %s\n", mysql_error(connection));
```

表 4-18 获取用于转发的服务器端口(续)

```
        return false;
    } else {//初始化逐行的结果集检索
        res = mysql_use_result(connection);
        //检索一个结果集合的下一行
        row = mysql_fetch_row(res);
        if (row <= 0) {
            printf("No virtual machine yet!\n");
            port = MIN_PORT_STR;
            return true;
        }
        int val_s = MIN_PORT;
        while (row) {//check
            if (val_s == atoi(row[0])) {
                val_s++;
                row = mysql_fetch_row(res);
            } else {
                 cout << "Available port: " << val_s << endl;
                stringstream ss;
                ss << val_s;
                ss >> rdpport;
                break;}
        }
        if (row == NULL) {
            cout << "Available port: " << val_s << endl;
            stringstream ss;
            ss << val_s;
            ss >> rdpport;}
```

4.4.2 SPICE 登录虚拟机功能的实现

该部分只是针对管理员介绍设计，通过 SPICE 完成制作镜像等过程。

1.前端代码实现

前端代码主要实现的功能是调用 SPICE 客户端，并将从后台返回的参数传递给 SPICE 客户端登录远程虚拟机。主要实现代码如表 4-19 所示

表 4-19 SPICE 连接虚拟机前端代码实现

```
SocketWrapper::fromInstreamToBuffer(in,buffer,packet_length);
QString ip = strtok(buffer,PACKET_DELIMITER_STR);
QString port = strtok(NULL,PACKET_DELIMITER_STR);
QString vmpassword = strtok(NULL,PACKET_DELIMITER_STR);
delete [] buffer;
QStringList parametersList;
QString parameters = "spice://" + ip + ":" + port + "?" + "password" + "=" +
vmpassword;
parametersList << parameters;
spicecProcess->start(SPICEC_PATH, parametersList);
```

2.后端代码实现

主要将 SPICE 登录虚拟机所需的参数传递给前端，实现代码如表 4-20 所示

表 4-20 传递用于 SPICE 连接的参数

```
void PacketWrapper::send_vm_info_packet(VirtualMachine * vm,int client_fd){
    string packet(vm->ip);
    packet = packet + PACKET_DELIMITER + vm->port + PACKET_DELIMITER
+ vm->lastpsw + PACKET_DELIMITER;
    unsigned int packet_length = packet.size();
    char * buffer = new char[sizeof(int) + packet_length];
    memcpy(buffer,&packet_length,sizeof(int));
```

表 4-20 传递用于 SPICE 连接的参数（续）

```
memcpy(buffer+sizeof(int),packet.c_str(),packet_length);
send(client_fd,buffer,sizeof(int)+packet_length,0);
delete buffer;}
```

4.5 数据库设计

办公桌面云系统采用的是 Mysql 数据库。依据功能需求设计创建以下 16 个数据库表。系统的 E-R 图如图 4-17 所示。

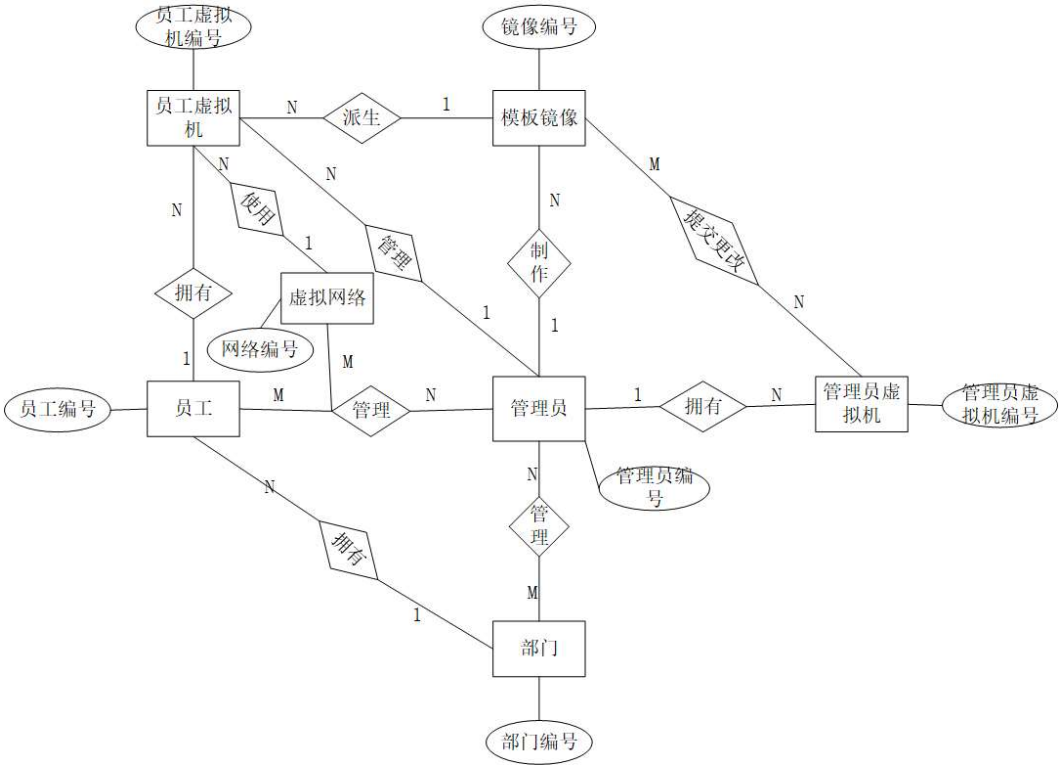

图 4-17 系统 E-R 图

1. 管理员表(admin)：存放管理员数据信息，详细信息见表 4-21 所示。

表 4-21 管理员表

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	管理员编号	非空	主键
name	varchar	50	管理员姓名	空	—
pwd	varchar	20	密码	空	—
state	varchar	1	管理员的状态	空	—

2. 员工表(staff)：存放部门员工数据信息，详细信息见表 4-22 所示。

表 4-22 员工表

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	员工编号	非空	主键
name	varchar	50	员工姓名	空	—
pwd	varchar	20	密码	空	—
state	varchar	1	员工的状态	空	—

3. 管理员虚拟机表(adminvirtualmachine)：存放管理员虚拟机数据信息，详细信息见表 4-23 所示。

表 4-23 管理员虚拟机表

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	虚拟机编号	非空	主键
ip	varchar	20	服务器 IP	空	—
port	int	11	端口	空	—
image	varchar	50	镜像名称	空	—
adminid	varchar	20	管理员编号	空	外键
imageid	int	11	模板镜像编号	空	外键
newpsw	varchar	20	最新密码	空	—
isinstalled	varchar	1	是否正在安装	空	—
isderived	varchar	1	是否已派生镜像	空	—

4. 部门表(department)：存放部门数据信息，详细信息见表 4-24 所示。

表 4-24 部门表

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	部门编号	非空	主键
name	varchar	30	部门名称	空	—
unit	varchar	30	所属单位名称	空	—
year	int	11	设立年份	空	—

5. 部门员工关联表(departmentstaff)：存放部门与员工关联数据信息，详细信息见表 4-25 所示。

表 4-25 部门员工关联表

字段	字段类型	字段长度	说明	Null	主外键
departmentid	varchar	20	部门编号	非空	主键、外键
staffid	varchar	20	员工编号	非空	主键、外键

6. 集群表(cluster)：存放服务器集群数据信息，详细信息见表 4-26 所示。

表 4-26 集群表

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	集群编号	非空	主键
name	varchar	30	集群名称	空	—
imagenum	int	11	集群已创建虚拟机数	空	—
capacity	int	11	集群虚拟机总数	空	—

7. 主机表(host)：存放服务器数据信息，详细信息见表 4-27 所示。

表 4-27 主机表

字段	字段类型	字段长度	说明	Null	主外键
id	int	11	服务器编号	非空	主键
ip	varchar	20	服务器 IP	空	—
port	varchar	10	心跳连接端口	空	—
localport	varchar	10	代理端端口	空	—
state	varchar	1	服务器状态	空	—
vmnumber	int	11	虚拟机数目	空	—
clusterid	varchar	20	所在集群编号	空	外键
isnfs	varchar	1	是否 NFS 服务端	空	—
isScp	varchar	1	是否负责同步模板镜像	空	—

8. 系统安装镜像表(iso)：存放创建镜像所依赖的 iso 文件数据信息，详细信息见表 4-28 所示。

表 4-28 系统安装镜像表

字段	字段类型	字段长度	说明	Null	主外键
id	int	11	镜像编号	非空	主键
name	varchar	30	镜像名称	空	—
format	varchar	20	格式	空	—
createtime	datetime	0	创建时间	空	—
disk	double	16	大小	空	—

9. 用户类别表(category)：存放为部门员工设置的镜像类别，详细信息见表 4-29 所示。

表 4-29 用户类别表

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	类别编号	非空	主键
name	varchar	30	类别名称	空	—
createtime	datetime	0	创建类别时间	空	—

10. 类别部门关联表(categorydepartment)：存放用户类别与部门关联数据信息，详细信息见表 4-30 所示。

表 4-30 类别部门关联表

字段	字段类型	字段长度	说明	Null	主外键
categoryid	varchar	20	类别编号	非空	主键、外键
departmentid	varchar	20	部门编号	非空	主键、外键

11. 类别部门负责人关联表(categorymanager)：存放用户类别与部门负责人关联数据信息，详细信息见表 4-31 所示。

表 4-31 类别部门负责人关联表

字段	字段类型	字段长度	说明	Null	主外键
categoryid	varchar	20	类别编号	非空	主键、外键
managerid	varchar	20	负责人编号	非空	主键、外键

12. 部门负责人表(manager)：存放部门负责人数据信息，详细信息见表 4-32 所示。

表 4-32 部门负责人

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	负责人编号	非空	主键
name	varchar	30	负责人名字	空	—
password	varchar	20	密码	空	—
mail	varchar	20	邮箱	空	—

13. 模板镜像表(templateimage)：存放创建模板镜像数据信息，详细信息见表 4-33 所示。

表 4-33 模板镜像表

字段	字段类型	字段长度	说明	Null	主外键
id	int	11	镜像编号	非空	主键
name	varchar	30	镜像名称	空	—
format	varchar	20	镜像格式	空	—
systemtype	varchar	20	镜像系统类型	空	—
createtime	datetime	0	镜像创建时间	空	—
lastrevisetime	datetime	0	镜像最后修改时间	空	—
disksize	double	16	镜像分配磁盘大小	空	—
depiction	varchar	100	对创建的镜像描述	空	—

14. 员工虚拟机表(staffvirtualmachine)：存放为员工分配的虚拟机数据信息，详细信息见表 4-34 所示。

表 4-34 员工虚拟机表

字段	字段类型	字段长度	说明	Null	主外键
id	varchar	20	虚拟机编号	非空	主键
ip	varchar	20	宿主机 IP	空	—
port	int	11	端口	空	—
image	varchar	100	镜像名称	空	—
staffid	varchar	20	员工编号	空	外键
categoryid	varchar	20	类别编号	空	外键
newpsw	varchar	30	最新密码	空	—
rdpport	int	11	用于 RDP 连接端口	空	—
vmname	varchar	30	虚拟机系统设置的用户名	空	—
vmpwd	varchar	30	虚拟机系统设置的密码	空	—

15. 虚拟网络表(vnetwork):存放系统创建的虚拟网络，详细信息见表 4-35 所示。

表 4-35 虚拟网络表

字段	字段类型	字段长度	说明	Null	主外键
id	int	11	虚拟网络编号	非空	主键
vnetname	varchar	30	虚拟网络名	非空	—
subnet	varchar	60	网段	非空	—
netmask	varchar	60	网络掩码	非空	—
startip	varchar	60	网络开始地址	非空	—
endip	varchar	60	网络结束地址	非空	—
vlan	varchar	20	vlan 号	非空	—

16. MAC、IP 分配表(vmacip)：存放 MAC、IP 分配信息，详细信息见表 4-36 所示。

表 4-36 MAC、IP 分配表

字段	字段类型	字段长度	说明	Null	主外键
id	int	11	编号	非空	主键
vnetname	varchar	30	虚拟网络名	非空	—
mac	varchar	60	MAC 地址	非空	—
ip	varchar	60	IP 地址	非空	—
categoryid	varchar	20	类别号	空	外键
dapartmentname	varchar	30	部门名	空	—
staffid	varchar	20	员工号	空	外键

4.6 本章小结

本章首先对系统的需求分析入手，从功能需求、非功能需求两个方面对系统做详细的论述。依据不同的用户的功能需求设计系统的整体功能结构以及系统服务端类图，并对系统各个功能模块做详细的设计实现。对混合型桌面传输协议登录虚拟机的实现做详细的说明，最后设计了系统的数据库。

第五章 办公桌面云系统运行与测试

5.1 测试环境

办公桌面云系统运行环境是由三部分构成：终端设备、数据中心与管理中心。本节是对这三部分软件配置与硬件配置进行介绍。

5.1.1 终端设备

终端设备使用的是实验室机房的台式机，终端配置如下表 5-1 所示。

表 5-1 终端配置

主机型号	CPU	内存	硬盘	操作系统
惠普 Z240 台式机	Xeon E3-1225 @3.30GHz	32G	1T	Windows10 企业版

网络环境：百兆局域网

5.1.2 数据中心

数据中心的任务是负责给虚拟机的运行提供环境并存储虚拟机的镜像等数据资源，因此数据中心包括服务器与共享存储。数据中心硬件配置如表 5-2 所示。

表 5-2 数据中心硬件配置

CPU	内存	硬盘	网卡
型号：Xeon E5-2620 主频：2.00GHz 核数：12 硬件虚拟化支持： Intel VT-d	126G	本地存储：800G 共享存储：10T	4 卡口 BCM5720 千兆网卡

软件配置如表 5-3 所示。

表 5-3 数据中心软件配置

操作系统	CentOS release 6.6
主要依赖软件	Libvirt(0.10.2)、QEMU-KVM(0.12.1.2)

网络环境：千兆局域网

5.1.3 管理中心

管理中心在办公桌面云系统中主要职责为管理资源和管理虚拟机。管理中心程序单独部署在服务器上，通过局域网与数据中心服务器连接。管理中心硬件配置如表 5-4 所示。

表 5-4 管理中心硬件配置

CPU	内存	硬盘	网卡
型号：Xeon E5-2620 主频：2.00GHz 核数：12 硬件虚拟化支持：Intel VT-d	64G	本地存储：600G	4 卡口 BCM5720 千兆网卡

软件配置如表 5-5 所示。

表 5-5 管理中心软件配置

操作系统	CentOS release 6.6
主要依赖软件	Libvirt(0.10.2)、QEMU-KVM(0.12.1.2)、MySQL(5.1.73)

网络环境：千兆局域网

5.2 系统运行效果

办公桌面云系统运行,包括管理员创建员工账号、创建部门、为部门添加员工、创建虚拟网络、为部门分配虚拟机等管理员的功能模块测试还有员工使用虚拟机功能测试。因此，分为管理员使用系统、员工使用系统两部分介绍，首先是管理员使用系统。测试所用的虚拟机的配置：操作系统为 win7 旗舰版，CPU 2 核，内存 3G, 硬盘 30G。

5.2.1 管理员使用系统

1) 管理员创建员工账号。首先管理员登录系统。办公桌面云系统登录客户端界面如下图 5-1 所示

图 5-1 登录办公桌面云系统界面

进入创建用户账号功能界面，点击“单个添加”输入两个新的员工信息，点击“提交”，完成新员工信息创建，如图 5-2 所示。

图 5-2 管理员添加员工信息

2) 管理员创建部门信息。与创建员工信息实现过程类似。进入创建部门功能界面，点击“单个添加”输入新部门信息，点击“提交”，完成新部门，如图 5-3 所示。

图 5-3 管理员添加部门信息

3) 管理员创建部门负责人、管理员创建新用户类别功能，过程与创建部门信息类似，在这里就不用图片展示。下面介绍虚拟网络创建，进入虚拟网络管理界面，创建一个名为 fm8 的虚拟网络，如图 5-4 所示。

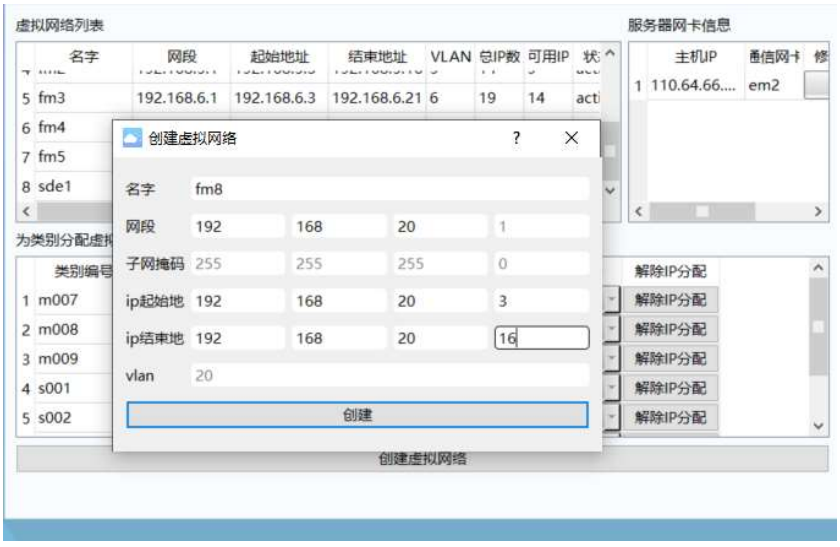

图 5-4 管理员创建虚拟网络

4) 管理员创建镜像。进入镜像管理界面，点击“制作镜像”，输入相应的参数，点击“启动制作”，如图 5-5 所示。之后就启动 SPICE 连接虚拟机制作 win7 系统镜像。

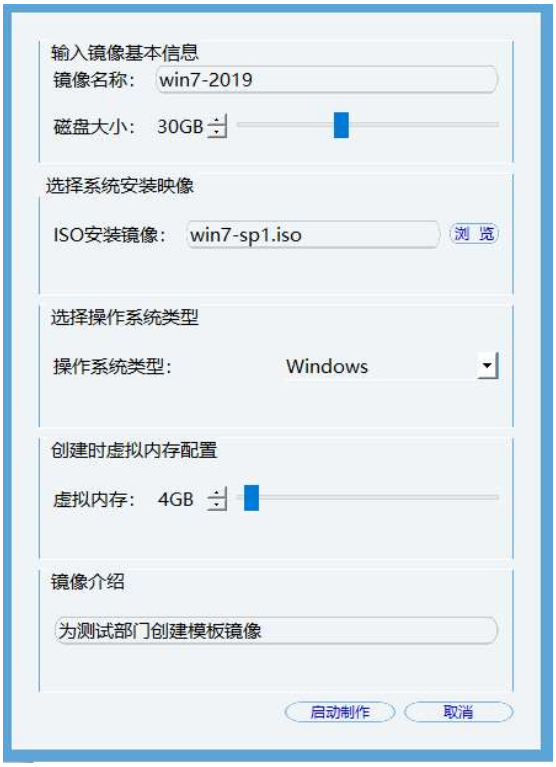

图 5-5 管理员制作镜像

5.2.2 员工使用系统

员工输入账号、密码登录后的主界面如下图 5-6 所示。

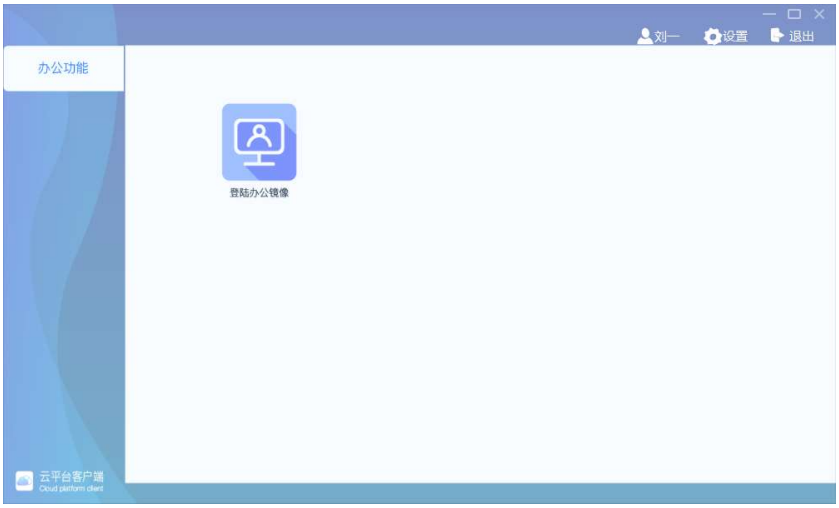

图 5-6 员工登录后主界面

点击“登陆办公镜像”，进入登陆办公镜像界面，选择相应的镜像登陆办公虚拟机。登陆虚拟机后，就进入用户办公环境如下图 5-7 所示。

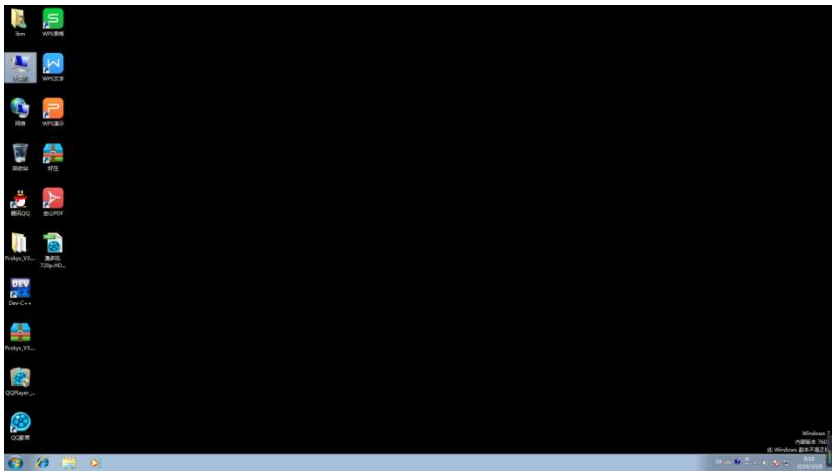

图 5-7 用户登陆虚拟机

之后就可以使用自己的虚拟机进行办公操作，图 5-8 为在虚拟机中播放本地高清视频。

图 5-8 播放本地视频

图 5-9 为使用 WPS 办公软件进行文档编辑

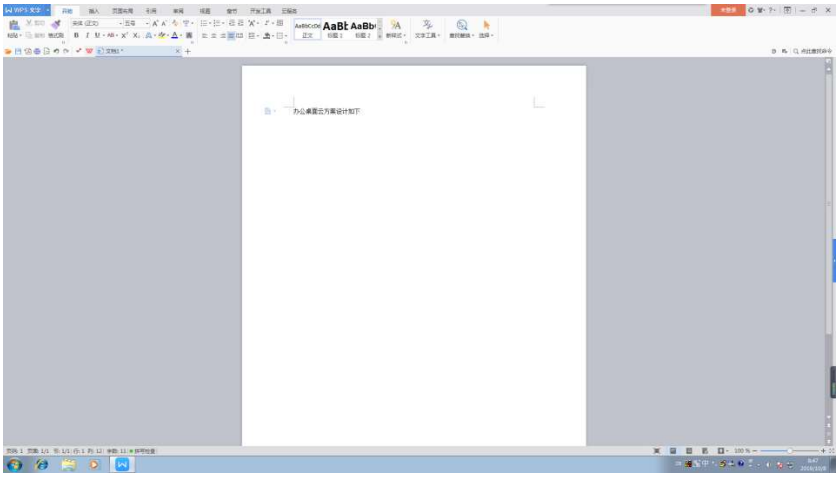

图 5-9 使用 WPS 办公软件

在虚拟机中浏览网页信息如图 5-10 所示

图 5-10 浏览网页

在虚拟机中使用编程软件进行程序设计，如图 5-11 所示。

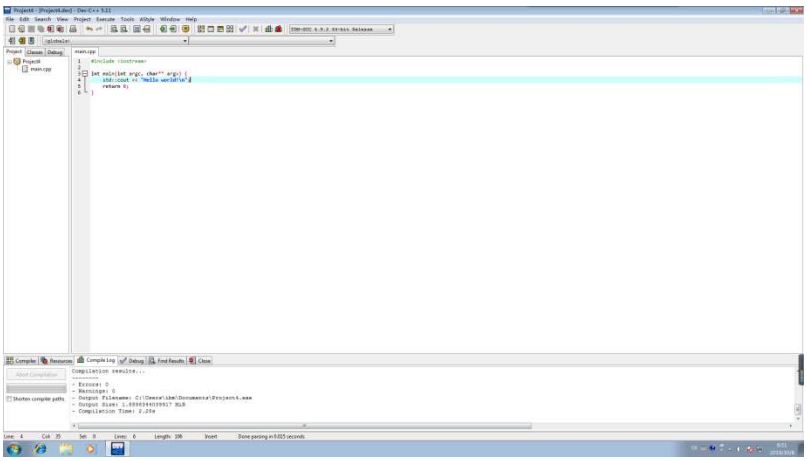

图 5-11 使用编程软件

5.3 系统性能测试

对于办公桌面云系统性能测试的主要方式为单台服务器在保证用户良好的体验前提下，最大能支撑多少虚拟机同时运行。模拟用户在不同场景使用不同的应用，在这里测试的场景有三个，分别为使用 WPS 办公软件、使用 Dev-C++编程工具以及使用 QQ 影音播放本地 720P 高清视频。

测试方式为每种场景单独测试，测试用户数起始设置为 5 并以 5 的倍数增加测试用户数，通过监测服务器硬件资源、网络负荷以及用户使用体验效果，得出不同场景下服务器能支撑的最佳的用户数。

1.播放本地高清视频测试结果如表 5-6 所示。

表 5-6 视频测试结果

同时使用人数	CPU 负荷	内存负荷	网络	体验效果
5	45%	14.6%	11MB/s	播放非常流畅
10	62%	26.8%	12MB/s	播放流畅
15	70%	40.2%	12MB/s	播放流畅
20	80%	52.9%	12MB/s	视频播放出现丢帧卡顿现象
25	91%	65.1%	12MB/s	丢帧卡顿现象明显
30	98%	77.2%	12MB/s	丢帧卡顿现象严重

表中我们可以看出，随着使用人数的增多，服务器的 CPU、内存的负荷呈明显增长趋势，当同时使用虚拟机播放高清视频的用户数为 30 时，CPU 的负荷为 98%，服务器处理能力达到了最大限度，网络速率基本保持不变。用户体验方面，在使用人数为 20 时，视频播放出现丢帧，播放不流畅，当用户数达到 30 时，视频播放非常卡顿。因此，为了保证用户体验度，同等配置的服务器能够支撑不超过 20 个虚拟机播放本地高清视频。

2.编程测试结果如表 5-7 所示。

表 5-7 编程测试结果

同时使用人数	CPU 负荷	内存负荷	网络	体验效果
5	16%	14.6%	4.5MB/s	体验非常流畅
10	28%	26.8%	5.2MB/s	体验流畅
15	43%	40.2%	5.4MB/s	体验流畅
20	56%	52.9%	5.6MB/s	体验流畅
25	69%	65.1%	5.8MB/s	操作出现延迟
30	83%	77.2%	6.1MB/s	延迟现象较严重

从编程测试结果来看，服务器的 CPU 负荷、内存负载及网络速率都是随着用户数增加呈上升趋势，CPU、内存负荷上升速度更显著。随着同时使用的用户数增加用户的体验性是逐渐下降，用户数在 25 时，出现操作响应延迟用户感觉不流畅现象，当达到

测试最大值 30 时，用户体验性大幅下降。因此，为了保证用户体验度，同等配置的服务器能够支撑不超过 25 个虚拟机较好的进行编程设计。

3.使用 WPS 办公软件测试结果如下表 5-8 所示。

表 5-8 使用 WPS 办公软件测试结果

同时使用人数	CPU 负荷	内存负荷	网络	体验效果
5	7%	14.6%	3.2MB/s	体验非常流畅
10	12%	26.8%	3.9MB/s	体验非常流畅
15	18%	40.2%	4.3MB/s	体验流畅
20	25%	52.9%	4.5MB/s	体验流畅
25	33%	65.1%	4.7MB/s	体验流畅
30	43%	77.2%	5.1MB/s	体验较流畅

从用户使用 WPS 测试结果来看，服务器的 CPU 负荷、内存负荷及网络速率随着同时使用的用户数增加而呈上升趋势，用户的体验性随着用户数的增加而有所下降，当用户数达到最大值 30 时，也有较好的用户体验效果。因此，为了保证用户体验度，同等配置的服务器能够支撑 30 个虚拟机较好的使用 WPS 办公应用。

5.4 测试结果分析

上节进行了三种场景的测试，从测试结果来看是有差异的，差异主要体现在服务器的 CPU 负荷及网络速率，而内存负荷并没有区别。

在 CPU 负荷方面，在 WPS 办公应用的 CPU 负荷是最低的，远低于播放本地高清视频与编程测试。

在网络速率方面，播放高清视频网络速率是最高的，而编程测试与 WPS 办公应用相差不大。

测试结果分析，对于视频播放场景，需要对大量的视频图像数据处理，特别是高清视频，需要消耗大量的 CPU 资源，这就导致了 CPU 负荷过高。虽然对图像数据进行了压缩、缓存等处理，但是在高清视频的播放，图像更新频率高且本地桌面变化范围大，需要大量的图像数据传输，因此需要更大的网络速率传输图像。对于编程测试场景，测

试的程序为高计算量的 C++程序，因此会导致服务器 CPU 负荷过高，相对于高清视频来说，编程测试桌面变化区域不大，传输的图像数据量小，也就不需要过高的网络速率进行图像数据的传输。对于 WPS 办公应用场景，大部分时间等待输入操作，对 CPU 消耗相对较小，因此服务器 CPU 负荷较低，对于网路速率与编程场景类似，虚拟桌面变化区域不大，需要的网络速率也较低。对于内存负荷，三种场景是相同的，因为对于虚拟机的内存分配，是在启动虚拟机时预分配的，而每个虚拟机的启动参数是相同的，对于服务器来说，每个虚拟机进程所占用的内存是一样的，这就造成了虽然场景不同，但是从服务器的内存负荷指标来看是一样的。

5.5 本章小结

本章介绍了办公桌面云系统测试环境的硬件及软件配置，之后通过系统的运行效果测试系统的功能，最后在三种基本的场景下，进行系统的性能测试并对测试结果进行了分析。

第六章 结论和展望

虚拟化技术是当今计算机领域比较热门的技术，桌面云是虚拟化技术一个比较成熟的应用，也催生出了大量从事桌面云的企业，越来越多的企业开始使用桌面云产品进行高效的管理办公。桌面云实现了桌面环境与终端设备的解耦和，让个人的桌面可以集中的运行在服务器上，让用户能够随时随地的灵活使用自己的桌面环境进行办公操作提高了员工的工作效率，另一方面，桌面的集中化，给运维管理人员了带来更加便捷高效的管理，不仅节省了运维管理人员大量的时间，同时为企业节省了成本。对于员工不同的需求，管理员也可以更加及时高效满足用户的需求，更加高效的利用服务器资源。数据集中存储在服务器端，数据的安全性也有更好的保障。桌面环境与终端设备解耦和，这就避免了终端设备损坏造成用户的桌面环境破坏以及用户数据的丢失风险。而一台虚拟机的出现异常也不会对运行在同一台服务器上的其他虚拟机造成影响，还能更快速的恢复虚拟机的运行。本文在对 QEMU-KVM 虚拟技术、SPICE 桌面传输协议及 RDP 传输协议研究的基础上，实现了一种基于 QEMU-KVM 的办公桌面云方案，主要工作有以下几点：

1.通过对虚拟化技术的研究和学习，并深入了解虚拟化技术对 CPU 虚拟化、内存虚拟化、硬件 I/O 设备虚拟化等资源虚拟化实现原理，确立了采用开源 QEMU-KVM 虚拟化技术。

2.通过对桌面传输协议实现原理的研究，对比不同传输协议优缺点，深入了解影响桌面传输协议效率的因素，确立了采用采用 SPICE 桌面传输协议与 RDP 传输协议相结合的实现方式。

3.通过调研分析，了解用户需求，对需求进行建模，桌面云架构采用 VDI，办公桌面云系统整体架构采用 C/S 客户端服务端方式，确立系统登陆身份为员工、管理员两个角色，员工的主要目的就是使用虚拟机进行办公操作相对来说功能简单，而管理员的功能相对来说就比较复杂了，对服务器集群管理、用户信息管理、虚拟机管理、部门信息管理、镜像管理等，设计系统整体功能模块，对各个功能模块做详细的设计与实现，并详细的设计的办公桌面云系统的数据库。

4.对办公桌面云系统进行详细的测试，包括功能测试与性能测试，性能测试采用了多种场景，每一种场景进行了不同数目的并发用户测试，通过对服务器的资源负荷监控，

对不同测试场景结果进行详细的分析，经过分析最后找出了造成不同场景结果差异的原因所在。

由于个人能力以及时间有限，该办公桌面云系统还存在一些不足之处，这也是下一步对系统进行改进、完善的工作。

1.RDP 协议安全性改进。防止攻击者对 RDP 协议通信双方进行内容的截获、破坏等操作，可以从 RDP 配置和 RDP 协议改进等方面进行优化。

2.数据的安全性。对于企业来说，数据的安全性相当重要，对于数据库数据的安全性，后续可以考虑实现数据库高可用，避免出现数据库故障，影响系统的使用。对于系统的高可用也是后续工作的重点。

参考文献

[1] Pan H, Yuan Y, Song W, et al. The Design and Implementation of Secure Cloud Desktop System[C]. International Conference On Signal And Information Processing, Networking And Computers. Springer, Singapore, 2018: 212-218.

[2] Padala, Praneel Reddy. Virtualization of Data Centers: study on Server Energy Consumption Performance[D]. Sweden: Blekinge Institute of Technology, 2018.

[3] Klement M. Models of integration of virtualization in education: Virtualization technology and possibilities of its use in education[J]. Computers & Education, 2017, 105: 31-43.

[4] Song T, Wang J, Wu J, et al. FastDesk: A remote desktop virtualization system for multi-tenant[J]. Future Generation Computer Systems, 2018, 81: 478-491.

[5] Agrawal S, Biswas R, Nath A. Virtual desktop infrastructure in higher education institution: Energy efficiency as an application of green computing[C].2014 Fourth International Conference on Communication Systems and Network Technologies. IEEE, 2014: 601-605.

[6] Zheng H, Liu D, Wang J, et al. A QoE-perceived screen updates transmission scheme in desktop virtualization environment[J]. Multimedia Tools and Applications, 2019, 78(12): 16755-16781.

[7] 许艳军, 姜进磊, 王博, 等. 几种虚拟机镜像格式及其性能测评[J]. 计算机应用, 2013, 33(A01): 22-25.

[8] Soriga S.G, Barbulescu M. A comparison of the performance and scalability of Xen and KVM hypervisors[C]. 2013 RoEduNet International Conference 12th Edition: Networking in Education and Research. IEEE, 2013: 1-6.

[9] 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.

[10] Morabito R. Power consumption of virtualization technologies: an empirical investigation[C]. 2015 IEEE/ACM 8th International Conference on Utility and Cloud

Computing (UCC). IEEE, 2015: 522-527.

[11] 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.

[12] 傅纯一.Horizon 新一代桌面和应用交付平台 JMP[EB/OL]. https://blogs.vmware.com/china/2017/11/09/horizon-新一代桌面和应用交付平台-jmp/, 2017-11.

[13] 傅纯一. 新一代 VDI 网络传输协议 Blast Extreme[EB/OL]. https://blogs.vmware.com/china/2017/11/09/新一代-vdi-网络传输协议-blast-extreme/, 2017-11.

[14] Casanova L, Kristianto E. Comparing RDP and PcoIP protocols for desktop virtualization in VMware environment [C]. 2017 5th International Conference on Cyber and IT Service Management (CITSM). IEEE, 2017: 1-4.

[15] VMware. VMWARE HORIZON 7 新功能特性-重新构思桌面与应用虚拟化[EB/OL]. https://www.vmware.com/content/dam/digitalmarketing/vmware/zh-cn/pdf/products/horizon/vmware-horizon7-whats-new.pdf, 2018.

[16] Citrix. Citrix Virtual Apps and Desktops [EB/OL]. https://docs.citrix.com/zh-cn/citrix-virtual-apps-desktops/technical-overview/hdx.html, 2019-8-29.

[17] 时钰森. 面向桌面云的 Android 虚拟机设备虚拟化的研究与实现[D]. 广州：华南理工大学, 2016.

[18] 百度文库. 深信服一站式 aDesk 桌面云方案技术白皮书[EB/OL]. https://wenku.baidu.com/view/e2a80a3aad51f01dc381f1b7.html, 2016-3-24.

[19] 和信创天. 和信下一代云桌面 VENGD 产品[EB/OL]. https://www.vesystem.com/products/3, 2018-4.

[20] 华为. 华为 FusionCloud 桌面云解决方案 5.2 详版彩页[EB/OL]. http://e.huawei.com/cn/marketingmaterial/cn/products/cloud_computing/cloud_computing/desktop_ cloud/hw_143874, 2014.

[21]阿里. 阿里云桌面产品介绍[EB/OL]. https://help.aliyun.com/product/53782.html?spm=a2c4g.11186623.6.540.77ff27acKh5OpO, 2017-6-7.

[22]IBM 虚拟化与云计算小组. 虚拟化与云计算[M]. 北京：电子工业出社, 2010: 26-54.

[23]Obasuyi G C, Sari A. Security challenges of virtualization hypervisors in virtualized hardware environment[J]. International Journal of Communications, Network and System Sciences, 2015, 8(07): 260-273.

[24]Sahoo J, Mohapatra S, Lath R. Virtualization: A survey on concepts, taxonomy and associated security issues[C].2010 Second International Conference on Computer and Network Technology. IEEE, 2010: 222-226.

[25]任永杰, 单海涛. KVM 虚拟化技术:实战与原理解析[M]. 北京：机械工业出版社, 2013: 17-20.

[26]曹欣. 半虚拟化技术分析与研究[D]. 杭州：浙江大学, 2008.

[27]范平. 虚拟化大不同:VDI 与 IDV 上演生死对决[J]. 网络与信息, 2012, 26(4): 38-39.

[28]吕俊宏. 关于新一代 IDV 架构的桌面虚拟化探讨[J]. 中国信息化, 2014(243): 61-63.

[29]Yang S J, Nieh J, Selsky M, et al. The Performance of Remote Display Mechanisms for Thin-Client Computing[C]. USENIX Annual Technical Conference, General Track. 2002: 131-146.

[30]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.

[31]Liao X, Jin H, Hu L, et al. Towards virtualized desktop environment[J]. concurrency and computation: Practice and experience, 2010, 22(4): 419-440.

[32]肖力. 深度实践 KVM:核心技术, 管理运维, 性能优化与项目实施[M]. 北京：机械工业出版社, 2015: 5-7.

[33]徐燕雯. 基于 KVM 的桌面虚拟化架构设计与实现[D]. 上海：上海交通大学, 2012.

[34]时卫东. 基于内核的虚拟机的研究[D]. 长春：吉林大学, 2011.

[35]Goto Y. Kernel-based virtual machine technology[J]. Fujitsu Scientific and Technical Journal, 2011, 47(3): 362-368.

[36] Russell R. virtio: towards a de-facto standard for virtual I/O devices[J]. ACM SIGOPS Operating Systems Review, 2008, 42(5): 95-103.

[37] Libvirt. The Virtualization API[EB/OL]. https://libvirt.org/, 2019-8.

[38] Microsoft. Understanding the Remote Desktop Protocol (RDP)[EB/OL]. https://support.microsoft.com/en-us/help/186607/understanding-the-remote-desktop-protocol-rdp, 2018-4.

[39] 王悦. RDP 协议的安全性分析与中间人攻击[D]. 北京：北京邮电大学, 2008.

[40] Rdesktop: A Remote Desktop Protocol Client[EB/OL]. www.rdesktop.org, 2019-7.

[41] 鲍捷, 宋靖雁, 姚丹亚, 等. NC 环境中的 RDP 协议解析[J]. 计算机应用与软件, 2004, 21(10)：67-69.

[42] Microsoft. Microsoft Remote Desktop Protocol(RDP) Features and Performance White paper[EB/OL]. https://www.microsoft.com, 2016-4.

[43] 杨子超. 基于 Android 的 RDP 客户端的设计与实现[D]. 成都：电子科技大学, 2013.

[44] 杨飞. 基于 SPICE 协议的虚拟桌面设计与实现[D]. 西安：西安邮电大学, 2016.

[45] 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(6): 885-892.

[46] 乔咏. SPICE 协议的视频传输分析与改进[D]. 济南：山东大学, 2013.

攻读硕士学位期间取得的研究成果

一、已发表（包括已接受待发表）的论文，以及已投稿、或已成文打算投稿、或拟成文投稿的论文情况**（只填写与学位论文内容相关的部分）：**

序号	作者（全体作者，按顺序排列）	题　　目	发表或投稿刊物名称、级别	发表的卷期、年月、页码	相当于学位论文的哪一部分（章、节）	被索引收录情况

注：在“发表的卷期、年月、页码”栏：

1 如果论文已发表，请填写发表的卷期、年月、页码；

2 如果论文已被接受，填写将要发表的卷期、年月；

3 以上都不是，请据实填写“已投稿”，“拟投稿”。

不够请另加页。

二、与学位内容相关的其它成果（包括专利、著作、获奖项目等）

1、软件著作权名称：《基于 openshift 的管理平台 V1.0》，登记号：2019SR0619493，

颁布日期：2019 年 06 月 17 日

致 谢

转瞬间，三年多的研究生时光就要结束了。在这短暂的研究生学习阶段，使我受益良多，不仅仅学业上进步，还学了更多的为人处事之道，让我能够更加坦然的面对生活，面对工作。在此，我衷心的感谢这段时间对我有帮助的老师、家人、同学以及朋友，谢谢你们的支持与鼓励。

我首先要感谢的就是我敬爱的导师。如果没有导师在学业上的谆谆教导，我是不可能完成我的论文的。在我论文书写的整个过程中，导师给予的指导和帮助引领我不断向前。导师不仅学识渊博，工作严谨负责，科研积累丰富，而且对新知识、新技术的敏感度极强的人。短短几年接触，我深深的被导师所影响。让我更加深刻的理解了学无止尽，离开学校是新的学习开始。

其次，我要感谢实验室的同门，谢谢你们的包融与帮助，让我有幸结识一群优秀的你们，遇见你们真好，怀念那些一起聚餐、打球、讨论学术问题的时光。祝愿各位同门学业进步、生活快乐、工作顺利。

然后，感谢我的家人，谢谢你们对我学业以及生活上的支持，让我更加安心从事学术研究，你们的爱与支持，让我走到今天。

最后，感谢百忙之中来参加论文评审和答辩的老师和专家，谢谢你们的指导。

Ⅳ - 2 答辩委员会对论文的评定意见

杨亚伟同学的硕士论文“基于 QEMU-KVM 的办公桌面云系统的设计与实现”对实际问题展开研究，具有一定的新颖性和较好的实践价值。

论文对桌面虚拟化技术架构、虚拟化具体实现过程原理和虚拟桌面传输协议等进行了研究，分析了 QEMU-KVM 架构和硬件设备虚拟化实现过程，对 SPICE 虚拟桌面传输协议和 RDP 传输协议进行了分析比较，在此基础上，实现了办公桌面云系统的架构设计和功能实现，并通过实验验证了系统的可用性。

论文工作量相对饱满，收集的数据和文献材料比较丰富，论文结构清晰，语言比较通顺，观点表达准确，采用的方法科学，取得的结论正确。反映出作者对所研究领域的理论知识掌握地较好，具有一定的研究素养。

学位申请人答辩过程讲述清楚，对评阅意见中提出的问题或质疑已作出明确的回复，答辩委员判定学位申请人的回复已达到评阅专家的要求。经答辩委员会无记名投票，同意该同学通过硕士论文答辩，同意授予硕士学位。

论文答辩日期：2019 年 12 月 2 日　　答辩委员会委员 5 人

表决票数：同意毕业及授予学位（5）票；

同意毕业，但不同意授予学位（0）票；

不同意毕业（0）票

表决结果（打“√”）：通过（√）；不通过（　）

决议：同意授予硕士学位（√）　不同意授予硕士学位（　）

答辩委员会成员签名	（主席）		