|  |
| --- |
| 产品总体设计说明书 |
| VDI |
| 苏永刚 |

# 引言

## 编写目的

本文档从总体上描述了整个系统的体系架构、设计思路，个别关键技术进行了详细的设计。

本文档作为项目概要设计的入口条件，为项目进行概要设计确定基本思路，并提供可行性的验证。

本文档必须以满足用户需求、技术可行为前提条件。

## 背景

VDI依赖底层Hypervisor平台，实现将虚拟机封装成为虚拟桌面，向用户分配的功能。但是通用的Hypervisor在如下两个方面缺少面向虚拟桌面应用场景的优化，导致性能低下：

共享存储

虚拟桌面对共享存储的访问，很大程度上存在同质特性，并发访问的过程中，浪费了极大的存储网络带宽资源，极端情况下可达95%以上。

远程访问协议

远程访问协议是用户访问虚拟桌面过程中最重要的协议，可供使用的开源或者开放协议的远程访问协议，通用Hypervisor仅提供VNC和SPICE两个协议，虚拟机仅提供RDP一个协议。这些协议的性能，都无法满足大尺寸屏幕、高清视频的现实需求。

鉴于此，决定开发下一代虚拟桌面系统，提供具有竞争力的产品与服务。

# 概述

## 目标

本文档为了满足《VDI产品规格需求说明书》所规定的需求而进行的系统总体设计。为后续概要设计、详细设计奠定基础，并确定基本思路。

## 运行环境

本系统服务器的设计目标是独立的软件平台基于Linux系统，客户端支持Windows和Linux两类软件平台。

## 需求

请参见《VDI产品规格需求说明书》。

## 局限

公司缺乏足够的人力，因此需要大规模参考开源设计的方法；

由于底层虚拟化平台的限制，产品仅限于私有云环境。

# 总体设计

## 基本设计概念和处理流程

### 系统角色

VDI系统共有八个角色，这八个角色互相配合，为用户提供虚拟桌面服务。

#### 数据中心侧-Core

Core角色承担VDI的业务处理，是VDI系统最核心的角色。

Core角色是控制器类角色，需要实时在线运行，支持集群扩展，以获得高可用性与性能可扩展性。

每个VDI系统有且仅有一个Core集群（或单机）。

#### 数据中心侧-Gateway

Gateway角色承担客户端与虚拟桌面连接的流量转发与审计。

Gateway角色是选配的，满足如下两个条件时，可以不部署Gateway角色

* 系统不需要审计VDI流量数据
* Client角色与VM在网络上可达，Client角色中的Protocol Client可以直接连接到VM或Host提供的远程桌面服务

如果部署Gateway角色，每个VDI系统至少有一个Gateway角色。多个Gateway角色之间相互独立，由Core控制。

#### 数据中心侧-Database

Database角色承担VDI系统数据持久化的任务，

Database角色采用开源项目承载，支持高可用性方案。

每个VDI系统有且仅有一个Database角色（主备或单机）。

#### 数据中心侧-Center

Center角色承担硬件资源管理的任务。

Center角色管理计算资源、存储资源、网络资源。有别于基础设施的虚拟化控制器，Center提供了面向VDI的性能优化手段。

每个VDI系统有且仅有一个Center角色（主备或单机）。

#### 数据中心侧-Node

Node角色承担虚拟机的运行，负责提供计算、网络、存储的服务。

每个VDI系统至少有一个Node角色。多个Node角色之间的组织关系由Center控制。

#### 用户侧-Web Console

Web Console角色是管理VDI系统的入口。

Web Console是一组Web页面，可以在任何位置发起访问。

#### 用户侧-Client

Client角色是用户访问VDI系统的入口。

Client角色以三种形式出现：

**嵌入式系统**

Client以云终端设备的形式整体交付

**应用软件**

Client以应用软件的形式安装在用户操作系统中，需要支持主流的Windows操作系统

**移动设备APP**

Client以APP形式安装在移动设备中，需要支持主流的Android/iOS系统

#### 虚机侧-Tools

Tools角色是在虚拟机内部，配合Core进行管理工作，配合Client角色为用户提供桌面服务。

### 系统部署

根据用户规模的不同，系统的部署规模也有所不同。划分系统部署规模的主要依据在于高可用性的级别。

由于用户侧与虚机侧的部署位置比较明确，不再系统部署中加以描述。这里的系统部署，主要还是数据中心的服务器。

#### 单机部署

单机部署没有任何高可用性，设计这样一种模式，目标市场有两个：

**快速演示**

**小型企业或中型企业试用**

单机部署使用Host本地存储



#### 多机部署

多机部署没有任何高可用性。多机模式的Core/Database/Center三个角色必须部署在一个节点上，Host和Gateway根据实际情况作扩展。设计这样一种模式，目标市场有两个：

**快速部署临时性使用环境**

**小型企业**

多机部署使用Host本地存储

第一种部署方式：仅Host角色分离，适用于并发量小于500的场景。



第二种部署方式：Host与Gateway角色分离，适用于并发量大于500的场景。



#### 集群部署

集群部署提供高可用性。为了实现可靠的高可用性，角色通常都部署在不同的节点上。Core构成集群，Center构成主备，Database构成主备。Host和Gateway不需要做高可用性，分别由Center和Core做健康监测与调度。设计这样一种模式，目标市场有两个：

**对可靠性要求较高的企业**

**跨地区的企业**

多机部署使用Host本地存储或共享存储。



#### 多数据中心部署

多数据中心部署本身与系统的高可用性无关。由于多数据中心的客户处在不同的地理位置，有必要将Client、Gateway、Host集中在相同的地理位置。<TODO>

### 系统设计原则

#### 根据可靠性分级

可靠性的获取，是靠成本拉动的。VDI的软件设计，坚持对可靠性进行分级，全系统的可靠性，取决于最不可靠的组件，而其他组件的可靠性提升，无助于整体的可靠性提升。

根据可靠性的级别，组织软件的设计与部署的架构，不盲目，不浪费。

#### 高内聚，低耦合，可复用

系统设计过程中，尽可能减少角色间、模块间的耦合程度，以切割明确的接口，通用的交互协议实现系统的各个组件。有利于系统的平行可扩展性设计。

组件中相似功能使用通用中间组件实现，降低代码的重复程度，有利于控制设计与实现缺陷。

#### 机制与策略分离

系统功能的设计，贯彻机制与策略的分离，尽可能减少硬编码实现的策略行为，有利于未来灵活支持更多的协议。

## 系统体系结构

### 软件平台总体框架



每个角色的总体框架都是类似的，分为对外的服务、业务逻辑、对其他服务的依赖以及数据持久化几个方面。

对外服务根据服务类别，有两类：

**Web Service**

Web Service服务通过标准HTTP协议，采用json作为传递命令和返回结果的数据格式，对应每个组件，都会有对应的文档描述。

**Socket Server**

Socket Service服务通过约定的端口号简历TCP连接，自定义通信协议，采用json作为传递命令和返回结果的数据格式，对应每个组件，都会有对应的文档描述。

### 操作系统选择

考虑到系统的稳定性与安全性保障，数据中心侧的角色均部署在Linux操作系统中。为了获取更加完备的硬件兼容性，采取RHEL 7 Server标准发行版作为基础系统。数据中心侧的产品，以发行版方式提供，不单独发布服务器软件安装包。

用户侧与桌面侧的操作系统，要求支持如下版本：

Windows XP SP3

Windows 7

Windows 7 SP1

Windows 8

Windows 8.1

Windows 8.1 with update

Windows Server 2008 R2

Windows Server 2012

Windows Server 2012 R2

Windows Server 2012 R2 with update

Ubuntu 12.04（基本功能）

Android 4.0 or above（仅用户侧）

iOS 7 or above（仅用户侧）

### 软件平台分解描述

#### 零级描述

外部关系：

OPV-VDI可以使用OPV-Suite提供的虚拟化环境服务，形成虚拟桌面。



总体架构（略）：



角色描述：



模块间关系：

|  |  |  |
| --- | --- | --- |
| 模块名称 | 提供的服务 | 依赖的模块 |
| Web Console | 人机界面 | Core |
| Client | 人机界面 | Core  Gateway  Tools |
| Core | 向Client提供Web Service服务  向Web Console提供Web Service服务 | Database  Center  Tools |
| Database | 向Core提供数据持久化服务 |  |
| Center | 向Core提供Socket Service服务 | Host |
| Host | 向Center提供虚拟机部署服务  向Gateway提供连接服务 |  |
| Gateway | 向Client提供连接服务 | Host |
| Tools | 向Client提供单点登录、设备映射服务  向Core提供桌面部署服务 |  |

#### 一级分解

全貌总览：



由于参与角色不同，一级分解图，分为两种场景描述。管理员行为，是指系统安装后，管理员筹备虚拟桌面的过程，该过程仅有管理员参与。用户行为，是指用户登录到系统获取虚拟桌面服务的过程，该过程的行为都是由用户发起的，管理员在这个过程中只做观察，没有干预行为。

##### 管理员行为



**Web Console**

Web Console是一组Web页面，通过JavaScript编写的Web Service客户端调用Core提供的Web Service。管理员登录后，将获得全部管理功能。

**Core**

提供Web Service，业务请求经过处理，分别产生三类动作：

1. 数据持久化
2. 操作虚拟化平台
3. 操作虚拟机

**Database**

提供数据持久化服务。

由于Core支持集群方式部署，为了提供集群重启后，异步任务的追踪，Database也会存储部分异步任务数据。

**Center**

Center是虚拟化平台的控制器，提供计算资源、存储资源、网络资源的管理与控制。Center主要的管理对象是Host，附带管理共享存储。

**Host**

Host提供计算资源、存储资源、网络资源，在Center的控制下，运行虚拟机。

**VM**

VM在虚拟桌面的形成过程中扮演重要的角色。在不同的阶段，VM以模板筹备、虚拟桌面等形式出现，成为VDI系统的核心管理对象。

**Tools**

Tools在管理员部署桌面的过程中，接收来自Core的指令，配置虚拟机。

##### 用户行为



**Client**

Client调用Core提供的Web Service，获取桌面的信息。然后使用Protocol Client访问桌面服务。

**Core**

提供Web Service，向用户提供桌面的列表。当用户发起连接的时候，为其准备桌面，提供SSO服务。

**Database**

提供数据持久化服务。

**Center**

Center提供虚拟机的状态，供Core决策，同时接受Core的指令，新建或销毁虚拟机。

**Host**

Host提供计算资源、存储资源、网络资源，在Center的控制下，运行虚拟机。

**VM**

VM在此就是虚拟桌面。

**Tools**

Tools接受Center的指令，配置虚拟桌面的用户及登录会话。同时也会建立虚拟通道，和Client中的Protocol Client配合实现资源重定向等功能。

**Gateway**

Gateway转发Client到VM的数据流量，在其中负责网络地址转换，隔离外部网络对虚拟机的直接访问。此外在这个隔离点上可以实现流量复制，作为桌面行为审计的基础数据。

#### 二级分解

##### 数据中心侧-Core



##### 数据中心侧-Gateway



##### 数据中心侧-Center

<TODO>

##### 数据中心侧-Node

<TODO>

##### 用户侧-Web Console



##### 用户侧-Client



##### 虚机侧-Tools



## 系统代码结构

系统代码采用GIT作为版本管理工具，以代码目录划分功能，以代码分支划分开发阶段。代码、编译脚本、打包脚本和文档都在代码库中追踪。

### 版本号

版本号有别于产品名，是版本升级过程中最核心的依据，不应发生颠覆性的变化。

切分版本的依据是Client角色与Core角色之间API兼容性，目前采取方案一。

**方案一：目前的方式**

版本号： 3.x.y.z@build

当无法达到向后兼容时，x增加；

向后兼容能够得到满足，但是新版本增加了一些新特性，y增加；

特性不变，只是修改缺陷，z增加；

Build号采用git代码库的commit号，格式为g+commit号前7位，如g1234abc

**方案二：行业通用方式**

版本号： x.y.z.build

当无法达到向后兼容时，x增加；

向后兼容能够得到满足，但是新版本增加了一些新特性，y增加；

特性不变，只是修改缺陷，z增加；

build号是强制增加的，不应该出现重复的现象

### 目录结构

目录结构的原则如下，实际操作过程中，根据情况做调整。

#### 顶级目录结构

顶级目录按照内容类别划分

doc：文档

build：编译脚本

package：打包脚本

src：代码

Changelog：大版本的修改说明

README：项目说明文件

COPYING：版权说明文件

#### 一级目录结构

**src、build、package目录**

一级目录中按照角色划分

core

gateway

center

node

webconsole

client

tools

其他非核心组件，如console

**doc目录**

<TODO>

#### 二级目录结构

src目录中的二级目录按照模块划分，内部根据开发平台选择不同的组织形式

build目录中的二级目录根据开发平台选择不同的组织形式

package目录中的二级目录根据打包工具选择不同的组织形式

### 分支结构

#### 主分支

**名称：master**

主分支代表项目的最新状况，没有版本计划的代码在主分支上存在。

#### 开发分支

**名称：develop\_x.y**

开发分支代表项目某个版本的开发状况，x与y与版本号一致。

开发分支从master上分离出来。

#### 发布分支

**名称：release\_x.y**

发布分支代表项目某个版本的发布状况，x与y与版本号一致。

发布分支从开发分支上分离出来。

#### 特性分支

**名称：feature\_name**

特性分支代表开发一个特性过程中的代码分支，name表示开发特性的简要名称。

特性分支从开发分支上分离出来。

#### 缺陷分支

**名称：bug\_id**

缺陷分支代表修正一个缺陷过程中的代码分支，id表示缺陷追踪库中的缺陷编号。

缺陷分支从发布分支上分离出来。

开发阶段，直接在版本分支上修改；一旦进入到发布阶段，必须创建缺陷分支。

### 特性裁减控制

特性裁剪采用特性分支处理。

### 编译

src目录中存放一个名为VERSION的文件，内容是当前的版本号。各项目需要引用这个文件生成各类资源。

VERSION文件的内容只有一个版本号。

## 尚未解决的问题

<TODO>

# 接口设计

## 用户接口

Client与Web Console向用户提供人机界面。由界面原型文档描述。

## 外部接口

外部接口描述各个角色对外暴露的服务，分别由各个角色的接口文档描述。

|  |  |
| --- | --- |
| 模块名称 | 接口类型 |
| Core | Web Service |
| Database | SQL |
| Center | Socket Service |
| Host | Socket Service |
| Gateway | Web Service |
| Tools | Web Service |

## 内部接口

详见各角色设计文档。

# 用户界面设计

Client与Web Console向用户提供人机界面。由界面原型文档描述。

总体原则：

* 统计数据图形可视化
* 异步任务执行过程图形可视化
* 减少表格式呈现
* 浏览器兼容性要求HTML5

# 功能总体设计

## 用户

### 功能描述

#### 用户管理的层次

用户可以根据不同的目的聚合成为不同的群体，按照组织架构原则聚合，成为组织单元，组织单元的定义为AD的兼容性提供了支持。按照资源分配原则聚合，成为交付组，使用交付组可以灵活的分配桌面与应用资源，方便管理。

用户和组织单元构成一个森林，森林的非叶节点表示一个包含用户的组织单元，森林的叶节点表示一个用户或者一个不包含用户的组织单元。

一个交付组组是一个集合，集合的元素可以是用户，也可以是组织单元，但必须属于同一个域。

交付组与交付组之间没有关联关系，所有交付组构成一个集合。



交付组和组织单元有很大的不同。交付组主要用于权限设置，而组织单元是一个容器，主要用于组织架构描述。本地用户

用户管理模块维护一个本地域。为了避免与后续导入的域发生名字冲突，在数据库中该域没有名称，最终由用户界面赋予合适的名字，如“本地域”。该域与其它导入的域处于同一级别。本地用户全都属于该本地域，并且不可变更到其他域中。

用户管理模块为本地域维护组织单元，顶层组织单元的名称为“Users”，本地用户默认属于该顶层组织单元，后续建立新的组织单元后，可以将用户放置于新的组织单元之中。

#### Microsoft AD用户

用户管理模块可以导入Microsoft AD域用户。在导入的时候，需要指定导入域的名称、域控制器地址、相关权限管理员的登录凭据等。

考虑到个性化的使用，允许用户指定仅导入域中特定的组织单元，而不是把整个域的全部用户都导入系统。

添加AD域时，首先访问LDAP://RootDSE，获取其defultNamingContext属性和configurationNamingContext属性。

以defultNamingContext作为起始路径，查找distinguishedName属性，拼接域的DNS名称。

以defultNamingContext作为起始路径，查找objectGUID属性，作为域UUID。

以cn=Partitions,<configurationNamingContext>作为起始路径。使用(&(objectcategory=Crossref)(dnsRoot=dnsDomainName)(netBIOSName=\*))过滤查找，第一条记录的netbiosname属性，作为域NETBIOS名称。

以defultNamingContext作为起始路径，使用(objectCategory=organizationalUnit)过滤查找所有组织单元。查找objectGUID属性，作为组织单元UUID。

以defultNamingContext作为起始路径，使用(&(objectCategory=person)(objectClass=user))过滤查找所有用户。查找objectGUID属性，作为用户UUID。

每一次导入都将形成一条新的同步记录，Core定时刷新需要同步的记录，并将做出适当的调整：

* 如果对应域中增加了用户，则将其导入。该用户得益于所属的组织单元，将立即获取对某些资源的访问权限；
* 如果对应域中删除了用户，则将其删除。如果该用户已经持久化拥有了对某些资源的访问权限，则清理这些权限，如果有必要，进一步清理对应资源的状态；
* 如果对应的域或组织单元被删除，则参考用户被删除的操作，加以清理；
* 如果对应的域或组织单元无法访问，则暂时不做处理，避免因为网络故障导致大规模不必要的清理。

删除同步记录的时候，需要首先将用户拥有的资源加以清理，然后再将用户记录删除。

域用户被导入系统后，与本地用户一样，可以被加入交付组，可以分配资源。但是在使用虚拟桌面和虚拟应用的时候，域用户使用域身份认证，本地用户使用本地身份认证。Tools会配合Core做出相应的预处理。

### 行为定义

|  |  |  |
| --- | --- | --- |
| 功能 | 本地 | AD |
| 导入用户 | ▲上传Microsoft Excel 2003/2007格式文档导入用户  ▲导入用户的用户名若与现有用户相同，则跳过，并给出警示  ▲用户名和口令是必选项  ▲组织单元是可选项  若不提供组织单元，用户将属于默认的顶层组织单元  若提供了组织单元，用户将属于该组织单元。若该组织单元尚未被建立，则将被建立在顶层组织单元之下 | ↓\_↓见添加目录服务 |
| 创建组织单元 | ▲可以创建组织单元  ▲组织单元名和所属组织单元是必选项 | —不适用 |
| 删除组织单元 | ▲可以删除组织单元  ▲下属组织单元及多数用户将全被删除  ▲删除组织单元之前需要清理用户已经分配到的资源 | —不适用 |
| 查看组织单元 | ▲可以查看组织单元信息及其层次关系 | ▲可以查看组织单元信息及其层次关系 |
| 编辑组织单元 | ▲修改组织单元名  ▲修改所属组织单元 | ▲可以修改组织单元中的用户访问VDI的权限  1．本组织单元中的用户，但不包括子组织单元中的用户  2．本组织单元中的用户，包括子组织单元中的用户 |
| 创建用户 | ▲可以创建用户  ▲用户名、口令、所属组织单元是必选项 | ▲可以创建用户  ▲用户名、口令、所属组织单元是必选项  ▲同时在AD中创建用户 |
| 删除用户 | ▲可以删除用户  ▲删除用户之前需要清理用户已经分配到的资源 | ▲可以删除用户  ▲删除用户之前需要清理用户已经分配到的资源  ▲同时在AD中删除用户 |
| 查看用户 | ▲可以查看用户信息  ▲支持按照部分用户名筛选  ▲支持分页 | ▲可以查看用户信息  ▲支持按照部分用户名筛选  ▲支持分页 |
| 编辑用户 | ▲可以修改用户名  ▲可以修改口令 | ▲可以修改口令  ▲同时在AD中修改口令 |
| 批量操作 | ▲可以批量实施删除操作 | ▲可以批量实施删除操作 |
| 添加目录服务 | —不适用 | ▲可以设定访问LDAP的路径、用户名、口令，添加LDAP目录服务， |
| 编辑目录服务 | —不适用 | ▲可以修改访问LDAP的路径、用户名、口令 |
| 删除目录服务 | —不适用 | ▲可以删除对目录服务的访问  ▲删除目录服务之前需要清理用户已经分配到的资源 |

## 模板

### 模板生成方式

1. 选择模板T0，选择桌面池所属的域DOM
2. 使用模板T0生成新的虚拟机VM0
3. 将VM0开机，配置DNS为DOM的DNS服务器
4. 将VM0关机
5. 将VM0转换成模板T1
6. 将T1作为桌面池DP的当前模板（可选）

### 模板升级方式

1. 使用桌面池的当前模板T1生成新的虚拟机VM1
2. 将VM1开机，配置模板
3. 将VM1关机
4. 将VM1转换成模板T2
5. 将T2为桌面池DP的当前模板（可选）

### 模板删除条件

模板T必须同时满足如下两个条件才可以被删除：

* 模板T不是任何一个桌面池的当前模板
* 没有任何桌面是T生成出来的

## 桌面与桌面池

### 桌面生成方式

1. 使用桌面池的当前模板T生成新的虚拟机VM1
2. 将VM1开机
3. 重置VM1的SID，重新启动
4. 配置VM1的主机名，重新启动
5. 配置VM1的域，重新启动，如果是本地域，跳过此步骤
6. 将VM1关机
7. 为VM1设置快照，作为将来重置的回滚点
8. 重复1~7，生成足够数量的桌面

### 行为定义

| 功能 | 自动浮动池 | 自动固定池 | 手动浮动池 | 手动固定池 |
| --- | --- | --- | --- | --- |
| 创建桌面池 | ▲创建桌面池，然后从模板逐个创建指定数目的桌面  ▲池创建后，立即可以授予访问权限  ▲桌面创建后，立即可以分配给用户 | 🡨\_🡨同自动浮动池 | ▲创建桌面池，然后管理员手工选定一批主机加入池  ▲池创建后，立即可以授予访问权限  ▲桌面加入后，立即可以分配给用户 | 🡨\_🡨同手动浮动池 |
| 删除桌面池 | ▲标记池为正在删除，此时拒绝新的连接  ▲如果指定了强制删除选项，则立即断开已连接状态的桌面  ▲删除所有不在已连接状态的桌面及其对应的虚拟机，包括已经进入断开保护期的桌面，同时解除对应的用户对桌面池的访问权限和桌面分配关系  ▲如果还有桌面由于处于已连接状态而未被删除，则等待用户主动断开或被管理员终止连接，然后立即删除桌面，同时解除对应的用户对桌面池的访问权限和桌面分配关系  ▲所有桌面都被删除后，删除桌面池 | 🡨\_🡨同自动浮动池 | ▲标记池为正在删除，此时拒绝新的连接  ▲如果指定了强制删除选项，则立即断开已连接状态的桌面  ▲删除所有不在已连接状态的桌面，包括已经进入断开保护期的桌面，同时解除对应的用户对桌面池的访问权限和桌面分配关系，但不删除对应的虚拟机  ▲如果还有桌面由于处于已连接状态而未被删除，则等待用户主动断开或被管理员终止连接，然后立即删除桌面，同时解除对应的用户对桌面池的访问权限和桌面分配关系，但不删除对应的虚拟机  ▲所有桌面都被删除后，删除桌面池 | 🡨\_🡨同手动浮动池 |
| 扩大桌面池 | ▲从桌面池当前模板逐个创建指定扩大数目的桌面  ▲桌面创建后，立即可以分配给用户  ▲扩大过程中，如果有新的缩小或扩大指令，则排队新的指令，等待扩大操作结束后依次执行新指令 | 🡨\_🡨同自动浮动池 | ▲管理员手工选定一批主机加入池  ▲桌面加入后，立即可以分配给用户 | 🡨\_🡨同手动浮动池 |
| 缩小桌面池 | ▲管理员指定缩减的数目  ▲逐个删除未被分配的桌面及其对应的虚拟机（不包括或由于处在断开保护期而暂时保留分配关系的桌面），直到达到要缩减的数量  ▲如果还没有达到要缩减的数量，则等待用户主动断开或被管理员终止连接，并且超出断开保护期后，立即删除被释放出来的桌面及其对应的虚拟机，直到达到要缩减的数量  ▲每个桌面被删除时，同步调整桌面池大小  ▲缩减过程中，如果有新的缩小或扩大指令，则在完成当前桌面的处理后，终止缩减过程，执行新指令 | ▲管理员指定缩减的数目  ▲逐个删除未被分配的桌面及其对应的虚拟机，直到达到要缩减的数量  ▲如果还没有达到要缩减的数量，则等待有桌面被取消分配关系（可能是取消分配关系，也有可能是取消访问权限连带取消分配关系），然后立即删除桌面及其对应的虚拟机，直到达到要缩减的数量  ▲每个桌面被删除时，同步调整桌面池大小  ▲缩减过程中，如果有新的缩小或扩大指令，则在完成当前桌面的处理后，终止缩减过程，执行新指令 | ▲管理员手工选定一批桌面从池中删除  ▲如果指定了强制缩小选项，则立即断开这一批桌面中已连接状态的桌面  ▲删除这一批桌面中不在已连接状态的桌面，包括已经进入断开保护期的桌面，同时解除对应的用户对桌面池的访问权限和桌面分配关系，但不删除对应的虚拟机  ▲如果还有桌面由于处于已连接状态而未被删除，则等待用户主动断开或被管理员终止连接，然后立即删除桌面（不考虑断开保护期），同时解除对应的用户对桌面池的访问权限和桌面分配关系，但不删除对应的虚拟机  ▲每个桌面被删除时，同步调整桌面池大小 | 🡨\_🡨同手动浮动池 |
| 添加桌面 | ╳不允许手动添加桌面 | ╳不允许手动添加桌面 | 🡩\_🡩同扩大桌面池 | 🡨\_🡨同手动浮动池 |
| 删除桌面 | ▲管理员手工选定一批桌面从桌面池中删除  ▲如果指定了强制删除选项，则立即断开这一批桌面中已连接状态的桌面  ▲删除这一批桌面中不在已连接状态的桌面（包括已经进入断开保护期的桌面）及其对应的虚拟机，同时解除对应的用户和桌面分配关系，但是不取消用户对桌面池的访问权限  ▲如果还有桌面由于处于已连接状态而未被删除，则等待用户主动断开或被管理员终止连接，然后立即删除桌面及其对应的虚拟机，同时解除对应的用户和桌面分配关系，但是不取消用户对桌面池的访问权限  ▲每个桌面被删除时，同步调整桌面池大小 | 🡨\_🡨同自动浮动池 | 🡩\_🡩同缩小桌面池 | 🡨\_🡨同手动浮动池 |
| 重置桌面 | ▲将桌面对应的虚拟机回滚到配置好主机名的快照 | 🡨\_🡨同自动浮动池 | ╳不允许重置桌面 | ╳不允许重置桌面 |
| 授予用户  对桌面池的  访问权限 | ▲管理员授予交付组对桌面池的访问权限 | 🡨\_🡨同自动浮动池 | 🡨\_🡨同自动浮动池 | 🡨\_🡨同自动固定池 |
| 解除用户  对桌面池的  访问权限 | ▲触发条件1：管理员解除交付组对桌面池的访问权限  ▲触发条件2：管理员将用户从所有有权访问桌面池的交付组中删除  ▲如果指定了强制解除选项，则立即断开对应用户和桌面的连接  ▲如果用户正在连接，则等待用户主动断开或被管理员终止连接，然后立即解除对应的用户和桌面分配关系（不考虑断开保护期），同时取消用户对桌面池的访问权限 | ▲触发条件1：管理员解除交付组对桌面池的访问权限  ▲触发条件2：管理员将用户从所有有权访问桌面池的交付组中删除  ▲如果指定了强制解除选项，则立即断开对应用户和桌面的连接  ▲如果用户正在连接，则等待用户主动断开或被管理员终止连接，然后立即解除对应的用户和桌面分配关系，同时取消用户对桌面池的访问权限 | 🡨\_🡨同自动浮动池 | 🡨\_🡨同自动固定池 |
| 分配用户桌面 | ▲随机分配方式：用户在连接桌面池的时候，被随机分配一个桌面，建立分配关系  ▲人工指定方式：管理员授予用户对桌面池的访问权限，同时选取一个未分配的桌面，建立用户和桌面的分配关系（当且仅当用户没有和桌面池中的任何桌面建立分配关系）  ▲用户断开桌面，并且持续到超出断开保护期后，分配关系将被取消 | ▲随机分配方式：用户在连接桌面池的时候，被随机分配一个桌面，建立分配关系  ▲人工指定方式：管理员授予用户对桌面池的访问权限，同时选取一个未分配的桌面，建立用户和桌面的分配关系（当且仅当用户没有和桌面池中的任何桌面建立分配关系）  ▲用户断开桌面，分配关系仍将保留 | 🡨\_🡨同自动浮动池 | 🡨\_🡨同自动固定池 |
| 回收用户桌面 | ▲用户断开桌面，并且持续到超出断开保护期后，分配关系将被取消  ▲回收的桌面，做重置的操作，确保隐私数据的销毁 | ▲管理员强制取消用户和桌面的分配关系，用户下一次连接桌面池的时候，重新分配  ▲回收的桌面，做重置的操作，确保隐私数据的销毁 | 🡨\_🡨同自动浮动池 | ▲管理员强制取消用户和桌面的分配关系，用户下一次连接桌面池的时候，重新分配  ▲回收的桌面，做注销用户、删除用户配置文件的操作，确保隐私数据的销毁 |

### 权限指派

用户和组织单元可以被加入到交付组中。桌面池与交付组建立分配关系。

权限分配关系构成一个有向图，图中的节点有以下四种：

* 用户，此类节点的入度为0，且与用户组织单元关系森林中的用户节点一一对应
* 组织单元，此类节点与用户组织单元关系森林中的组织单元节点一一对应
* 交付组，此类节点与交付组集合中的元素一一对应，与每个组内部的元素没有关系
* 桌面池，此类节点的出度为0

图中的边有以下四种：

* 在用户组织单元关系森林中，一个用户(U)是一个组织单元(O)的直接子节点，则在图中对应节点之间有一条边，边的起点为(U)，边的终点为(O)
* 在用户组织单元关系森林中，一个组织单元(O1)是另一个组织单元(O2)的直接子节点，则在图中对应节点之间有一条边，边的起点为(U)，边的终点为(O)
* 一个组织单元(O)属于一个交付组(G)，则在两个节点之间有一条边，边的起点为(O)，边的终点为(G)
* 一个用户(U)属于一个交付组(G)，则在两个节点之间有一条边，边的起点为(U)，边的终点为(G)
* 一个交付组(P)被分配一个池(P)，则在两个节点之间有一条边，边的起点为(G)，边的终点为(P)

当且仅当一个用户(U)所对应的节点，与其要访问的某个桌面池(P)所对应的节点之间，至少有1条路径的时候，该用户(U)可以访问这个桌面池(P)。

## 数据中心

<TODO>

# 系统容错处理设计

## 错误提示信息

## 补救措施

## 系统维护设计

# 参考资料

