**SSET平台协议栈虚拟化方案**

星 级：

版本/修订：V1.0

编 制

审 核

|  |
| --- |
| 2022-XX-XX 文件发布日期 2022-XX-XX文件生效日期 |
|  |

**文档修订历史**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **版本** | **作者** | **版本变化对象** | **变化内容描述** | **审核人** | **批准人** | **修订日期** |
| V1.0 | 孙加伟 |  | 初始版本 |  |  | 2022/2/11 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

目 录

[1 引言 1](#_Toc96591341)

[1.1 编写目的 1](#_Toc96591342)

[1.2 背景 1](#_Toc96591343)

[2 设计分析 2](#_Toc96591344)

[2.1 用户切片 2](#_Toc96591345)

[2.2 异构系统仿真 3](#_Toc96591346)

[2.3 双重海量用户 3](#_Toc96591347)

[2.4 异地资源协同（去中心化） 4](#_Toc96591348)

[2.5 稳定性和可扩展性 4](#_Toc96591349)

[3 总体设计 5](#_Toc96591350)

[3.1 架构设计 5](#_Toc96591351)

[3.2 容器化 6](#_Toc96591352)

[3.2.1 部署方式 6](#_Toc96591353)

[3.2.2 host模式 6](#_Toc96591354)

[3.2.3 bridge模式 7](#_Toc96591355)

[3.2.4 端到端测试 8](#_Toc96591356)

[3.3 容器云化（进行中） 9](#_Toc96591357)

[3.3.1 功能划分 9](#_Toc96591358)

[3.3.2 资源线 10](#_Toc96591359)

[3.3.3 业务线 12](#_Toc96591360)

[3.3.4 host模式 15](#_Toc96591361)

[3.3.5 bridge模式 15](#_Toc96591362)

[3.3.6 问题 15](#_Toc96591363)

[3.4 私有云化 15](#_Toc96591364)

# 引言

## 编写目的

本文档的目的是从SSET 平台虚拟化需求出发，设计V1.0版本的的整体方案，可分解出进行概要设计和详细设计的模块和子模块，定义各模块和子模块的功能，模块和子模块的关系，满足协议系统虚拟化 V1.0版本的系统需求和软件设计需求。

本文档用于指导协议系统虚拟化V1.0版本的概要设计和详细设计文档，以及后续的集成测试。

## 背景

软件系统名称： SSET协议栈虚拟化

软件开发者：SSET协议栈组

开发语言：C/C++、shell、python语言

硬件平台：X86服务器

软件平台：Linux

# 设计分析

（关键词：云化、用户切片、IP模型镜像库、异构、双重海量、去中心化）

## **用户切片**



SSET平台设计采用B/S架构，针对多用户的场景利用切片的思想在物理资源池上按需个性化的编排虚拟化仿真资源。用户切片具有以下特点：

1. 专人专网：根据仿真用户权限、级别、动态分配仿真资源、动态弹缩。
2. 用户隔离：用户只会在各自虚拟资源中进行仿真，相互之间不会干涉。

## 异构系统仿真



随着业务场景的丰富，需要支持各种不同的系统，满足仿真模拟测试的需求。

## 双重海量用户



1. 前端支持海量用户。
2. 每个用户支持在各自虚拟环境内进行海量仿真。

## 异地资源协同（去中心化）



1、SSET平台在多地均有部署，形成去中心化的物理资源池，跨地区的物理资源共享。

2、用户需要使用自己的服务器，向SSET平台云管系统注册。

## 稳定性和可扩展性

由于虚拟资源相互隔离，系统更为健壮。若用户仿真资源发生故障，系统动态弹缩，快速恢复该用户仿真环境（记录销毁前用户信息和场景需求，pod守护），并且不会影响其他用户。

随着使用用户的增加和场景丰富，物理资源告警时，增加物理服务器不会影响现网服务的运行，用户无感知。

# 总体设计

## 架构设计



整个SSET平台云化可以分为以下三步开发：

1、SaaS：docker 容器化

1. PaaS：K8s+docker 容器云化

3、IaaS：OpenStak+K8s+docker 私有云化

## 容器化

### 部署方式

单机容器部署分为docker和docker-compose两种方式，docker用于单个容器部署，docker-compose用于多个容器一起部署。推荐使用docker-compose。

### host模式

#### 网络



#### 用户面路由

与非容器部署一致。

### bridge模式

#### 网络



#### 用户面路由

iptables -nvL

iptables -L -t nat

* 主动上行和应答下行

在核心网容器内部配置NAT：

*### Enable IPv4/IPv6 Forwarding*

*$ sudo sysctl -w net.ipv4.ip\_forward=1*

*$ sudo sysctl -w net.ipv6.conf.all.forwarding=1*

*### Add NAT Rule*

*$ sudo iptables -t nat -A POSTROUTING -s 10.49.0.0/16 ! -o ogstun -j MASQUERADE*

*$ sudo ip6tables -t nat -A POSTROUTING -s 2001:230:cafe::/48 ! -o ogstun -j MASQUERADE*

* 主动下行

1. 配置静态路由第一跳：

*route add –net 10.49.0.0/16 gw 10.37.6.44*

1. 开启forward转发:

*iptables –A FORWARD –s 10.37.6.0/21 –j ACCEPT*

1. 配置静态路由第二跳

*route add –net 10.49.0.0/16 gw 172.18.0.1*

### 端到端测试

#### Ping包

Ping 10.49.0.3（对端IP） –I 10.49.0.2（本机分配IP）

#### 灌包

客户端：./iperf –u –c 10.49.0.3(对端IP) –p 8888（重新指定端口） -i 1 –t 100 –b 50M

–B 10.49.0.2（本机分配IP）

服务端：./iperf –s –i 1 –u –p 8888 –B –B 10.49.0.3（本机分配IP）

## 容器云化（进行中）



容器云化是PaaS类型的云化，依托于Ks8和docker实现，架构相对IaaS轻量，适用于中小型体量的云平台搭建。现阶段SSET云平台推荐使用这种架构。

### 功能划分

按照功能划分，整个SSET云平台可以分为资源线和业务线。

资源线：负责用户切片管理、容器资源编排、平台监控、显示管理

### 资源线



#### 用户切片管理



|  |  |
| --- | --- |
| 用户切片标识 | |
| 静态标签 | 动态标签 |

切片静态标签：前台用户的注册信息、权限、等级、用户组别等

切片动态标签：仿真场景ID、平台分配资源ID（pod/docker ID等）等

由用户切片维护仿真用户和虚拟资源之间的关系，资源线和业务线均需要关联切片信息。所以切片信息至关重要。

#### 平台监控

资源监控：物理服务器状态监控、虚拟容器状态监控（资源运维监控）

用户监控：用户信息和行为监控（用户切片维护信息监控）

#### 编排管理



编排系统调用K8s API？可以实现后端业务的动态启停和守护。

#### 显示管理



### 业务线



#### OMC



OMC只负责处理业务的上下行，如配置下发，指令下发，指标上传等。OMC需要通过用户切片知晓该用户虚拟资源ID，才能将用户指令传递到正确的POD内部。

#### 协议业务



上面以海量用户仿真为例构建协议业务场景。每个前端用户在后端均申请有一个pod，pod内部为完整的海量用户仿真资源，所以单个pod就能完成10万用户接入流程模拟。



##### OAM



OAM是协议栈的管控模块和中间件，每个协议栈进程均需要和OAM维护心跳。同时OMC消息的下发和协议栈指标的上报均通过OAM交互。协议栈运行指标由OAM转交rokcetMQ。此外OAM还负责业务的负载均衡。

### host模式（不推荐使用）

#### yaml

#### 网络和路由

### 非host模式

#### 容器yaml

#### 网络和路由（calio插件）

iptables -nvL

iptables -L -t nat

* 主动上行和应答下行

在核心网容器内部配置NAT：

*### Enable IPv4/IPv6 Forwarding*

*$ sudo sysctl -w net.ipv4.ip\_forward=1*

*### Add NAT Rule*

*$ sudo iptables -t nat -A POSTROUTING -s 10.49.0.0/16 ! -o ogstun -j MASQUERADE*

* 主动下行（仅介绍原理）
* 从集群中容器下行：

Master节点：route add -net 100.45.0.0/16 gw 10.37.6.41（核心网node）

核心网node节点：route add -net 100.45.0.0/16 gw 10.244.166.146（核心网虚拟网卡）

* 从外部服务器下行：

服务器节点：route delete -net 100.45.0.0/24 gw 10.37.6.40（master节点）

*iptables –A FORWARD –s 10.37.6.0/21 –j ACCEPT*

Master节点：route add -net 100.45.0.0/16 gw 10.37.6.41（核心网node）

核心网node节点：route add -net 100.45.0.0/16 gw 10.244.166.146（核心网虚拟网卡）

### 问题

## 私有云化

尚未开始