文章编号:1006-3080(2015)02-0272-05

一种基于 OpenStack 的云应用开发框架

孙寒玉, 顾春华, 万 锋, 杨巍巍, 周冀平, 陈 煌
(华东理工大学信息科学与工程学院, 上海 200237)

摘要:云计算的迅速发展对云应用与云服务的开发提出了更高要求。然而,依靠提供商提供的云平台基本功能组件不能直接满足需求,对云平台进行二次开发会增加开发周期,降低开发速率。OpenStack 是一个开源的云管理平台,为客户提供云基础架构服务,包括计算服务、存储服务和网络服务。以 OpenStack 为基础,针对虚拟机管理应用、存储应用和定制应用,设计并实现了一种云应用开发框架,融合了服务组合和元数据的设计思想。基于该框架实现了一个虚拟机定制系统,并验证了该设计框架的可用性和有效性。

关键词:开发框架; 开源云平台; OpenStack; 元数据; 服务组合
中图分类号:TP39 **文献标志码**:A
DOI:10.14135/j.cnki.1006-3080.2015.02.021

A Development Framework for Cloud Applications Based on OpenStack

SUN Han-yu, GU Chun-hua, WAN Feng, YANG Wei-wei, ZHOU Ji-ping, CHEN Huang
(*School of Information Science and Engineering, East China University of Science and Technology, Shanghai* 200237, *China*)

Abstract: Cloud computing market is developing rapidly, promoting the requirement of developing cloud applications and cloud service. However, the cloud platform provided by the platform vendor cannot be able to meet the requirements completely, and developers need secondary development, which increases development cycle and reduces the efficiency. OpenStack is an open source cloud computing platform, providing service of computing, storage and networking. Integrating ideas of service-oriented architecture and metadata, a development framework of cloud platform based on OpenStack is proposed for development of virtual machine management applications, storage application and custom application. Based on the framework, a virtual machine customization system is developed, which verifies the usability and efficiency of the development framework.

Key words: development framework; open source cloud platform; OpenStack; metadata; service-oriented architecture

自 2006 年亚马逊推出弹性计算云服务,云计算的概念逐渐引起人们的关注。云计算是指基础设施的交付和使用模式,通过网络以按需、易扩展的方式获得所需的资源(硬件、平台、软件)[1]。众多企业向开发者提供云开发平台,但由于其经济原因和部署问题,开发者不能灵活应用这些云平台组件,企业、高校、研究机构等逐渐将目光汇集到开源云平台,在开源云平台上开发云应用与服务[2-4]。开源云平台以其灵活性高、成本低、可移植性强等优点被人们认可,并得到迅速的发展[5]。

收稿日期:2014-05-28
作者简介:孙寒玉(1991-),女,江苏宜兴人,硕士生,主要从事云计算方面研究。E-mail: lifeofshy@163.com
通信联系人:顾春华,E-mail: chgu@ecust.edu.cn

以开源云平台为基础的云应用开发，需要一个合理、健壮的开发框架，能与云服务运行环境结合，提供灵活且规范化的服务和接口，快速部署，高效地完成用户需求，提高资源利用率。

1　开源云平台 OpenStack

目前，比较主流的云平台有 Google 推出的 GAE(Google App Engine)、新浪推出的 SAE(Sina App Engine)[6]、开源云平台 OpenStack、Eucalyptus、CloudStack、OpenNebula 等[7]，各平台有其各自特点。其中，OpenStack 作为完全开源的云平台，以其成本低、部署方便快速、所用开发语言 python 编写灵活等特性赢得开发者青睐。因此，本文使用 OpenStack 作为研究基础。

OpenStack 由美国国家航空航天局和 Rackspace 合作研发，是以 Apache 许可授权的一个完全开源的云计算平台[8]。该项目旨在对所有类型的云计算提供简单的、大规模的、可伸缩的、功能丰富的解决方案[9]。OpenStack 的主要目标是管理计算资源、存储资源和网络资源[10]。管理大量虚拟机资源，可以按照需求为企业或个人提供计算资源；按照存储模式为云服务和云应用提供可配置的对象存储和块存储；按照网络管理方式提供高扩展和高自动化的插件式网络和 IP 管理方式。

OpenStack 至今已发布 8 个版本[11]，最新版本 Havana[12] 的基本组件包括：

(1) 认证组件 Keystone：提供认证和管理服务，管理用户、账号、角色的信息，分为验证和服务目录。

(2) 镜像组件 Glance：注册、查找和检索虚拟机的镜像文件，其中 Glance-API 负责处理镜像的接收、发现、检索和存储命令请求；Glance-registry 负责保存处理和检索镜像文件的元数据；A database 负责存储元数据；Store adapter 负责匹配不同的存储类型。

(3) 计算组件 Nova[13]：nova-api 接收外部访问；nova-compute 管理指令并对虚拟机实施具体管理工作；nova-scheduler 通过一定的算法调度计算资源。

(4) Dashboard 组件 Horizon：用户操作界面。

(5) 块存储组件 Cinder：执行 volume 的相关功能。

(6) 对象存储组件 Swift：具有高速直接访问和数据共享等优势的数据共享存储体系结构。proxy server 提供 API，负责与其他组件相互通信；storage server 提供磁盘设备上的存储服务；consistency server 负责故障处理。

(7) 网络组件 Neutron：提供灵活的物理网络管理。

(8) 自动化部署组件 Heat：在 Havana 版本中首次提出，旨在为用户提供可预先定义的云服务。

(9) 监测组件 Ceilometer：为计费和监控以及其他服务提供数据支撑。

2　框架设计

本文设计的云应用开发框架以 OpenStack 资源池作为底层的基础设施服务，将资源抽象为 User、Domain、Container、Data 4 种元数据形式，并提供相应的元数据服务，构建面向服务的体系结构模型，将服务按照需求以定义好的接口和协议连接起来。因此，框架中的 OpenStack 资源池作为资源底层，提供云资源服务；元数据层抽象整合资源，使得资源存储规范，高效利用；云服务层定制服务接口和连接协议；业务管控层整合业务规则；云应用界面层开发设计交互页面。框架总体设计图如图 1 所示。

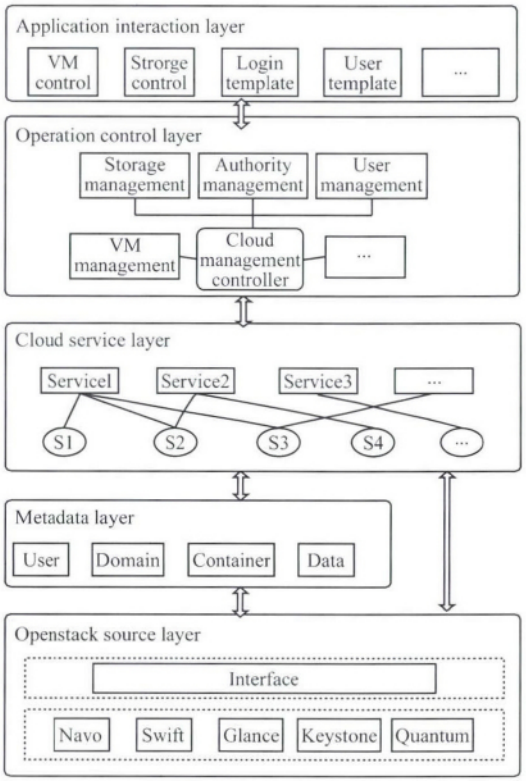

图 1　框架总体设计图

Fig. 1　Design of framework

2.1　OpenStack 资源池

OpenStack 资源池分为两层，上层作为接口层，

主要用于和虚拟资源池进行数据通讯和交换。使用API映射技术,开发者不需要了解底层OpenStack各模块的代码细节,所有与虚拟资源池交互的操作都在此层完成。

下层是开源云平台OpenStack的各个组件,它们之间相互协作,构建出一个庞大的虚拟资源池为上层提供资源和服务。

2.2 元数据层

元数据层是对资源对象进行描述,以数据的形式展现出来。本文借鉴CDMI(Cloud Data Management Interface)[14]中关于元数据的定义,结合框架自身的特点,将存储资源及计算资源抽象为4种数据对象,具体如下:

(1) User={ID,Name,domain,Info,islife},其中domain是指用户所在域。

(2) Domain={ID,Name,DomainURI,Users,Parents,Size,Createtime,Updatetime,Endtime},其中DomainURI为域的资源标识符,域通过此属性定位。

(3) Container={ID,Name,ContainerURI,Domain,Parent_ID,size,Createtime,Updatetime,Endtime},其中ContainerURI为Container的资源标识符,域通过此属性定位。属性Domain是域ID的集合{Domain_ID1,Domain_ID2,...}。

(4) Data={ID,Name,DataURI,Domain,Container_ID,size,Createtime,Updatetime,Endtime}。

数据对象是资源的表示形式,也是对资源的操作形式,元数据通过Restful方式与云服务层交互,提供元服务。

2.3 云服务层

云服务层分为两个部分,下层是由一系列不可分割的原子服务(以下统称元服务)组成,每一个元服务执行单一操作。元服务定义良好的接口,使云服务上层可以通过一定的方式对元服务组合,构建组合服务,满足需求。上层是一系列组合服务,它们是按一定的标准接口和规则将元服务组合在一起的。服务间的松耦合关系确保了服务的易维护性和可用性。各服务间相互独立,可单独进行调整以满足新的服务需求,具有良好的可扩展性。

云服务层中,Sj代表元服务,作为最小的服务粒度,由元数据层和OpenStack资源池实现。元服务层提供的组合服务由不同数量和序列的元服务构成。例如:Service1为创建虚拟机服务,创建虚拟机服务首先要通过keystone认证,由nova-compute组件通过Glance-api获取镜像,创建虚拟机,最后由Neutron组件为虚拟机配置网络。创建虚拟机服务过程如图2所示。

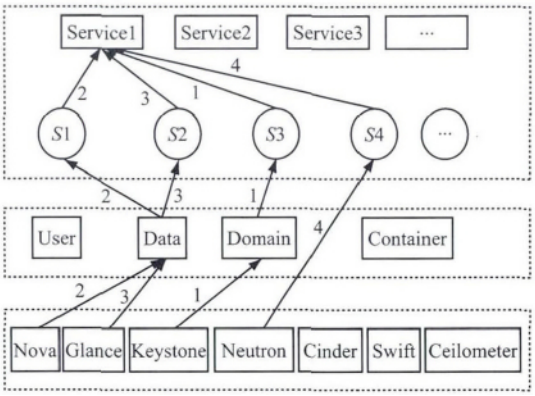

图2 服务构建

Fig. 2 Build the service

元服务集合如下:

元服务集={S1,S2,S3,S4,…}

S1:nova-compute组件执行虚拟机创建服务;

S2:通过Glance-api获取镜像服务;

S3:keystone认证服务;

S4:Neutron组件为虚拟机配置网络服务。

则创建虚拟机服务的集合为{S3,S1,S3,S2,S3,S4}。

2.4 业务管控层

业务管控层在应用开发中整合业务规则,实现业务流程等与业务需求相关的工作,另外完成一些验证性工作。由存储管理(Storage management)、虚拟机管理(VM management)、用户管理(User management)、权限管理(Authority management)等模块组成功能模块库。各功能模块通过云管理控制器管理调度,实现云应用界面层中页面控件及模块的功能。对于相同的功能模块不需要重新开发设计,可直接复用。模块之间相互独立,增加了框架的灵活性,修改单一模块不会影响到其他功能,增加了健壮性。

业务管控层中运行管理控制器Controller作为核心控制器,控制器作为Filter运行在Web应用,负责拦截页面请求。当请求到达时,该Filter会过滤请求,通过预先设定好的XML文件找到功能模块库中的对应功能模块,功能代码在模块的Action中实现。用户通过配置文件直接调用功能模块,也可创建定制功能模块,完成业务需求。

2.5 云应用界面层

云应用界面层负责界面的开发,为云应用及服务提供虚拟机管理控件(VM control)、存储管理控件(Storage control)等应用控件,登陆模板(Login

template)、用户管理模板(User template)等开发模板。开发者可选择控件模块，也可根据需求开发。主要控件及模板包括：

(1) 虚拟机管理模板，创建、删除、启动、挂起虚拟机，远程连接虚拟机等控件。

(2) 用户信息管理模板，用户的增删改查操作控件。

(3) 文件管理模板：文件的上传、下载以及删除控件。

3　应用案例及分析

3.1　案例需求分析

该系统根据课程需要创建虚拟机镜像，从而生成满足实验环境的虚拟机。学生拥有个人虚拟实验机，可随时进行实验操作并保存当前实验状态，解决了高校实验教学中物理机配置繁琐、更新困难、无法实现个性化定制的问题。以本文提出的框架为基础，实现了以开源云平台为基础的实验虚拟机定制系统，验证了框架的可用性和有效性。

3.2　案例实现及分析

3.2.1　**用户权限管理**　根据需求分析，系统共有 3 个角色：管理员、老师、学生，使用元数据服务创建对应的 domain。分别为：

管理员 domain：Domain_Admin={01，管理员，Domain_01，Users，00，500，20140201}Users 为用户 ID 集合。

老师 domain：Domain_teacher={02，老师，Domain_02，Users，01，500，20140201}。

学生 domain：Domain_student={03，学生，Domain_03，Users，Parents，500，20140201}。

Parents={01，02}。

一个用户可以在一个域中也可以在多个域中，若用户在某个域中，便具有此域中所能达到的所有权限。

3.2.2　**服务序列组合**　根据框架结构，在业务管控层中对业务流程处理，由云服务层提供服务，云服务层提供众多元服务，对其进行一定的逻辑排列生成组合服务以满足需求。以创建虚拟机为例，时序图如图 3 所示。

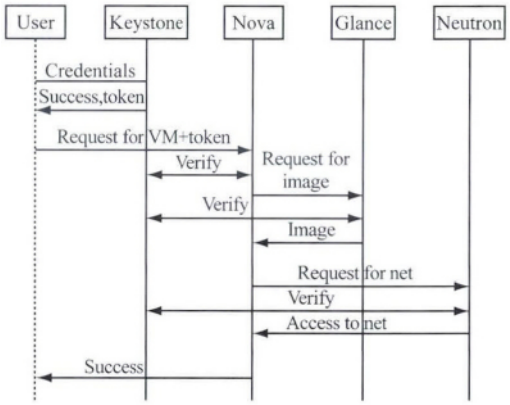

图 3　创建虚拟机时序图

Fig. 3　Sequence Diagram of Create a VM

业务管控层中创建虚拟机服务如下：

```
VMForCreate vmforcreate = new VMForCreate();
vmforcreate.createVM(flavor_id,image_ref,vmID);
```

则在云服务层中，createVM 服务操作如下：

```
if(Keystone_Userforvm_Create(token)){
//keystone 验证服务
If(Keystone_Userforimage(token)){
Metadata_VM_Create(flavor_id,image_ref,vmname,userID); //元数据驱动操作：创建虚拟机
If(Keystone_UserforSetIP(token)){
Metadata_VM_SetIP(vmname,IP);
//元数据驱动操作：为虚拟机设置 IP
}}}
else
…
```

3.2.3　**案例实现**　图 4 为实验虚拟机定制系统中虚拟机管理的页面，其中实验机申请、查看虚拟机各项信息、编辑、保存快照、重启虚拟机、删除虚拟机等功能都是由各项原子服务以一定的逻辑序列组合而成，这些服务再由应用管控层按照需求分析的业务流程和业务规则进行服务，最后由云应用界面层的界面模板及控件表现，实现系统与用户交互。

4　结束语

本文提出了一种基于开源云平台 OpenStack，以服务组合机制，元数据理念提供服务的云应用开发框架。基于框架，快速实现了实验室虚拟机定制系统，使得开发不再是繁杂的代码编写，而是简单的元服务逻辑组合，有效地缩短了软件开发的周期，提高了编程效率和规范性，提高软件的重用性、灵活性和扩展性。原子服务的组合理念使得系统后期维护变得简单，避免了牵一发而动全身的处境，有效提高了系统的稳定性。

图4 虚拟机管理界面

Fig. 4 Web of VM management

参考文献:

[1] **刘宇芳. 云计算及其实质的探究**[J]. **惠州学院学报**, 2010, **30**(6): 48-52.

[2] Chris, Preimesberger. **开源云平台5大优劣势分析**[J]. **通讯世界**, 2012(8):48.

[3] **黄志成. 开源云计算** OpenStack **在高校计算机机房中的应用研究**[J]. **计算机与现代化**, 2013(3):204-206,211.

[4] **肖飞, 杨晶, 刘黎明. 基于** OpenStack **的计算机实验室自助服务平台的设计与实现**[J]. **计算机与现代化**, 2013(7):202-203.

[5] **王海燚, 张云超, 邱晓瑞. 基于** OpenStack **的对象存储服务管理平台设计与实现**[J]. **广东通信技术**, 2013(4):23-27.

[6] **徐鹏, 陈思, 苏森. 互联网应用** PaaS **平台体系结构**[J]. **北京邮电大学学报**, 2012, **35**(1):120-124.

[7] **邓红, 王东兴. 主流开源云平台的商业选择**[J]. **电脑知识与技术**, 2012, **8**(32):7830-7832.

[8] **李小宁, 李磊, 金连文, 等. 基于** OpenStack **构建私有云计算平台**[J]. **云计算专栏**, 2012(9): 1-8.

[9] OpenStack. he 5-minute overview[EB/OL]. [2014-4-30]. http://www.openstack.org/.

[10] OpenStack. About OpenStack [EB/OL]. [2014-4-15]. http://www.openstack.org/software/.

[11] OpenStack. Releases[EB/OL]. [2014-4-20]. https://wiki.openstack.org/wiki/Releases.

[12] OpenStack. Installation guide for Ubuntu 12.04 [EB/OL]. [2014-4-10]. http://docs.openstack.org/havana/install-guide/ install/apt/content/.

[13] **陈伯龙, 程志鹏, 张杰. 云计算与** OpenStack:**虚拟机** Nova **篇**[M]. **北京:电子工业出版社**, 2013.

[14] SNIA. Cloud data management interface (CDMI) v1.0.2 [EB/OL]. [2014-4-13]. http://www.snia.org/cdmi.

(上接第259页)

[5] Lee Hansung. Face recognition based on sparse representation classifier with gabor-edge components histogram[C]// 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems (SITIS). Naples: IEEE Press, 2012:105-109.

[6] Yang Jian, Zhang David, Frangi A F. Two-dimensional PCA: A new approach to appearance-based face representation and recognition[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, **26**(1):131-137.

[7] Moon H, Phillips P J. Computational and performance aspects of PCA-based face-recognition algorithms [J]. Perception-London, 2001, **30**(3): 303-322.

[8] Bartlett M S, Movellan J R, Sejnowski T J. Face recognition by independent component analysis[J]. IEEE Transactions on Neural Networks, 2002, **13**(6): 1450-1464.

[9] Yu Hua, Yang Jie. A direct LDA algorithm for high-dimensional data——with application to face recognition[J]. Pattern Recognition, 2001, **34**(10): 2067-2070.

[10] Cui Zhen, Shan Shiguang. Structured sparse linear discriminant analysis[C]//2012 19th IEEE International Conference on Image Processing (ICIP). Orlando: IEEE Press, 2012: 1161-1164.

[11] Yang Meng, Zhang Lei, Yang Jian, *et al*. Metaface learning for sparse representation based face recognition[C]// 2010 17th IEEE International Conference on Image Processing (ICIP). Hong Kong:IEEE Press, 2010: 1601-1604.

[12] **朱杰, 杨万和, 唐振民. 基于字典学习的核稀疏表示人脸识别方法** [J]. **模式识别与人工智能**,2012,**25**(5):859-864.

[13] Yin Jun, Liu Zhonghua, Jin Zhong, *et al*. Kernel sparse representation based classification[J]. Neurocomputing, 2012, **77**(1): 120-128.

[14] Gao S, Tsang I W H, Chia L T. Sparse representation with kernels [J]. IEEE Transactions on Image Processing, 2013, **22**(2): 423-434.