第 2 章　开源云平台 OpenStack

2.1　OpenStack 发展史

OpenStack（http://www.openstack.org/）是由 Rackspace 和 NASA（美国国家航空航天局）共同研发的云计算平台，是一个旨在为公有云及私有云的建设与管理提供软件的开源项目。OpenStack 项目最初包括两个模块，一是 NASA 开发的计算服务模块 Nova，另一个是 Rackspace 开发的云存储（对象存储）模块 Swift。在 2010 年 10 月，用于镜像管理的部件 Glance 加入其中，形成了 OpenStack 的核心架构。

Rackspace 是全球三大数据中心之一，公司总部位于美国，在全球拥有 10 个以上数据中心，管理超过 10 万台服务器，是全球领先的托管服务器及云计算提供商。而美国国家航空航天局（NASA）的星云计划 Nebula 是 NASA 埃姆斯研究中心的一项云计算重点开发项目，它整合了一系列的开源组件，形成无缝自助平台，利用虚拟化、可扩展技术提供了高性能计算、存储和网络。

Rackspace 和 NASA 合作时决定 OpenStack 由 Rackspace 管理，为了使 OpenStack 更好地发展，Rackspace 联合部分成员于 2011 年成立了 OpenStack 基金会，其下有三个分支：技术委员会、用户委员会、董事会。OpenStack 基金会作为一个独立的组织，确保 OpenStack 在长期内获得更好的发展，保护、培育和提升 OpenStack 软件和社区，包括用户、开发者和整个生态系统。会员既包括 IT 厂商、公司成员，也包括以个人名义或代表公司加入的个人成员。OpenStack 发展迅速，除了得到 Rackspace 和 NASA 的支持，还吸引了 IBM、惠普、戴尔以及英特尔等著名 IT 巨头们的加入和支持，而且拥有 5600 位个人会员，其社区活跃度也已经超越了 Eucalyptus 和 CloudStack，成为仅次于 Linux 的世界第二大开源基金会。OpenStack 的技术更新和版本发行速度很快，从机构成立至第一个版本发布仅用了很短的时间，之后基本上每六个月发布一个新版本。OpenStack 的版本发展如表 2-1 所示。

表 2-1　OpenStack 版本发展

时　间	成　果
2010 年 6 月	Rackspace 和 NASA 成立 OpenStack
2010 年 7 月	超过 25 名合作伙伴
2010 年 10 月	首个版本发布，代号“Austin”，35 名合作伙伴
2011 年 2 月	代号“Bexar”版本发布
2011 年 4 月	代号“Cactus”版本发布
2011 年 9 月	代号“Diablo”版本发布，该版本相对比较稳定，可以开始大规模部署应用
2012 年 4 月	代号“Essex”版本发布
2012 年 9 月	发布 Folsom 版本，融合 Quantum 网络服务
2013 年 4 月	发布 Grizzly 版本，将 Melang 和 Quantum 融合起来支撑网络服务

续表

时　间	成　果
2013 年 10 月	发布 Havana 版本，正式发布 Ceilometer 项目和 Heat 项目，网络服务 Quantum 变更为 Neutron
2014 年 4 月	发布 Icehouse 版本，新项目 Trove 成为版本的组成部分
2014 年 10 月	发布 Juno 版本，实现对 Hadoop 和 Spark 集群管理和监控的自动化服务
2015 年 4 月	发布 Kilo 版本，实现首个完整版的 ironic 裸机服务

2.2 OpenStack 概述

OpenStack 的愿景是为所有公有云和私有云提供商提供可满足其任意需求、容易实施且可以大规模扩展的开源云计算平台。整个 OpenStack 项目被设计为可大规模灵活扩展的云计算操作系统，任何组织均可以通过 OpenStack 基于标准化的硬件设施创建和提供云计算服务。

由于亚马逊经过多年的发展并取得了巨大的成功，已经成为事实上的 IaaS 的标准，OpenStack 也希望通过标准化服务的方式，在云计算的标准化和规范化方面有所推动。OpenStack 定位是亚马逊 AWS（Amazon Web Services）的开源实现，无论在功能上还是 API 接口上，都尽可能与 AWS 保持兼容，所以 OpenStack 很多功能和亚马逊基本上是对应的，如表 2-2 所示。

表 2-2　OpenStack 与 AWS 映射

亚马逊 AWS	OpenStack
EC2 弹性虚拟机	Nova，虚拟化管理程序
S3 云存储	Swift，对象存储组件
EBS 弹性云硬盘	Nova-volume/Cinder，虚拟机的存储管理组件
ELB 负载均衡	Atlas-LB，OpenStack 外围项目，实现负载均衡
Console 控制台	Dashboard Horizon，界面访问控制台
VPC 虚拟私有云	Neutron，网络管理的组件
IAM 认证鉴权	Keystone，提供身份认证和授权的组件
Elastic MapReduce	Sahara，大数据方案

此外，OpenStack 提供了框架标准和 API，用户可以以此为基础构建云计算解决方案，这些标准与亚马逊云计算服务 AWS 是兼容的。OpenStack 所有模块子系统之间均是通过标准化的 API 实现服务调用的，标准化的接口服务框架意味着可以有差异化的实现，所以 OpenStack 也以包容的方式融合了诸多厂商的云计算服务，如 KVM、VMware、Xen、Hyper-V 等。

OpenStack 是一个开源软件，以 Apache 许可证授权。OpenStack 的版本由 OpenStack 基金会整理与发布，同时应用厂商对核心的要求也不断返回 OpenStack 基金会的技术委员会，并进一步促使拥有更强大功能的新版本推出。OpenStack 的开放性使其能够在推动技术创新的同时，达到与应用厂商共赢的局面。此外，由于开源软件的源代码是公开的，若源代码有质量方面的问题，则更易于被发现并被修正，从而源代码的安全漏洞也易于被发现并被修正。OpenStack 主要用 Python 编写，其代码质量相当高，带有一个完全文档化的 API，开发者可以很方便地从 OpenStack 的官方网站获取代码和文档。由此可见，OpenStack 的社会化研发、OpenStack 基金会的有效管理、Apache 许可证授权等原则和机制有力地保证了 OpenStack 的发展和创新。

由于 OpenStack 可帮助服务商实现类似亚马逊 EC2 和 S3 这种基础设施服务，因此被越来

越多的厂家和云计算服务提供商采纳并应用到生产环境中。Rackspace 已经采用 OpenStack 提供虚拟机和云存储服务，其中云存储 Swift 已经达到 100 PB。HP 推出的公有云服务也是基于 OpenStack 的。IBM 作为 OpenStack 基金会的白金会员，于 2013 年推出首个基于 OpenStack 的产品 SmartCloud Orchestrator。SmartCloud 是基于数据中心运行云部署的平台名称，Orchestrator 则为用户提供云应用所需的计算、存储和网络资源交付服务。通过 SmartCloud Orchestrator，用户可以在基础设施和平台层面进行端到端的服务部署，可以自定义工作流用于过程自动化和 IT 管理、资源监控、成本管理等。eBay 在使用 OpenStack 搭建其私有云之前，有自己研发的云，现在他们的一大工作是将之前开发的云的一些代码迁移到现在 OpenStack 环境中。从下面几个数字可以看出 eBay OpenStack 云的规模：8 个地理位置分散的完全隔离的可用域、7 000 多个 Hypervisor、65 000 个虚拟机、1.3 PB 块存储、90 TB 对象存储。

OpenStack 提供了公有云及私有云部署的解决方案，同时也逐渐成为混合云部署的标准。在实践应用中，很少企业有能力将其整个基础架构移至公有云，对于大多数企业而言，混合部署将成为常态。2015 年 Google 加入 OpenStack 基金会，将加速 OpenStack 的深入推广以及与 GCE（Google Compute Engine）等公有云的互联互通，使 OpenStack 成为企业级市场和互联网巨头共同认可的开放云、混合云平台标准。

2.3　OpenStack 架构剖析

关键需求决定架构，在分析 OpenStack 架构前，需要再次深入理解 OpenStack 的业务背景。OpenStack 项目被设计为可大规模灵活扩展的云计算操作系统，任何组织均可通过 OpenStack 创建和提供云计算服务。为了达到该目标，OpenStack 需要具有如下功能：项目所有的构成子系统和服务均被集成起来，一起提供 IaaS 服务；通过标准化公用服务接口 API 实现集成；子系统和服务之间可以通过 API 互相调用。

OpenStack 采用了职责拆分的设计理念，根据职责不同拆分成 7 个核心子系统，每个子系统都可以独立部署和使用。在每个子系统中，又根据分层（layer）设计理念，拆分为 API、逻辑处理（包含数据库存储）、底层驱动适配 3 个层次。OpenStack 的核心系统概念架构如图 2-1 所示。

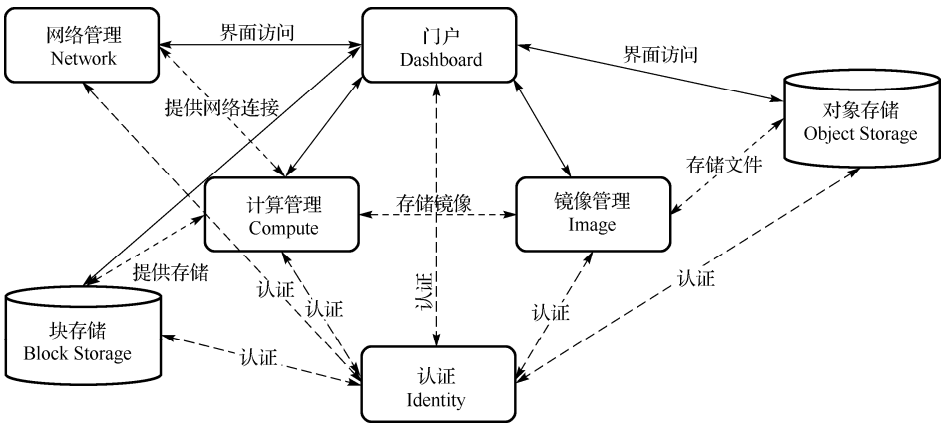

图 2-1　OpenStack 核心系统概念架构图

OpenStack 的核心项目包括

1）对象存储（Object Storage）：系统名称是 Swift，通过简单的 key/value 的方式实现对象

文件的存储和读取，适用于“一次写入，多次读取，无须修改”的情况，例如图片、视频、邮件附件等海量数据的存储。对象存储俗称**云存储**，国外的 dropbox、box，以及国内的云盘等均是云存储的应用，而 dropbox 则是基于亚马逊 S3 API 接口开发的典型案例。

2）镜像管理（Image）：系统名称为 Glance，提供虚拟磁盘镜像的目录分类管理以及镜像库存储管理，用于 OpenStack 虚拟机。

3）计算管理（Compute）：系统名称为 Nova，提供虚拟主机，包括虚拟机、弹性云硬盘等服务。通过虚拟化技术，例如 KVM、Xen、VMware ESXI 等实现计算、网络、存储等资源池的构建及应用，将计算能力通过虚拟机的方式交付用户。虚拟机的诞生很大程度上改变了 IT 支撑运维的管理模式，带来了诸如采购、管理、运维等的变革。

4）网络管理（Network）：系统名称为 nova-network 和 Neutron，其实现了虚拟机的网络资源管理，包括网络连接、子网 IP 管理、L3 的公网映射、后续的负载均衡等。

5）块存储（Block Storage）：系统名称为 nova-volume 和 Cinder，其实现了对块存储的管理，为虚拟机提供云硬盘（块设备）服务。块存储将物理存储根据需要划分成不同的存储空间提供给虚拟机，虚拟机将其识别为新的硬盘。该系统的规划支持主流的 IP-SAN、FC-SAN 等存储网络，以及 NAS 存储设备。Essex 版本中 nova-volume 实现了块存储（云硬盘）的管理，在 Folsom 版本后，独立新增加的 Cinder 项目增强了该方面的管理能力。

6）认证管理（Identity）：系统名称为 Keystone，其为 OpenStack 所有的系统提供统一的授权和身份验证服务。

7）界面展示：系统名称为 Horizon，是基于 OpenStack API 接口开发的 Web 呈现。

整个 OpenStack 对终端用户提供两大类访问入口：界面 Horizon 和 API。所有子系统提供标准化 API，终端用户（包括开发人员）通过 API 访问和调用不同子系统的服务。子系统内部划分 API、逻辑处理（Manager）和底层驱动适配（Driver）。

不同子系统通过 API 实现交互，这里主要是指 REST 风格的 API。RESTfull 风格的接口设计理念基于 HTTP 协议，类似于 WebService，但更简单。REST 提出了一些设计概念和准则，包括：网络上的所有事物都被抽象为资源（Resource）；每个资源对应一个唯一的资源标识（Resource Identifier）；通过通用的连接器接口（Generic Connector Interface）对资源进行操作；对资源的各种操作不会改变资源标识；所有的操作都是无状态的（Stateless）。

由于 OpenStack 不同子系统之间采用标准的接口（REST）实现交互，那么在架构上就天然具备了一些优势，如：具有高可用性和高可扩展性，具备分布式部署能力，具备基于 HTTP 的负载均衡能力，从而可实现大规模灵活扩展的设计目标；面向接口服务开发的模式，不同子系统间实现低耦合；具备灵活的扩展能力，可以灵活调整具体接口实现；具备优秀的集成能力，开发人员可以通过 API 实现应用定制开发，特别是定制符合客户需求的界面、原生 API 等。国内已经有不少公司基于 OpenStack 研发出适合于自己及相关市场的门户界面。

2.4 OpenStack 核心组件

2.4.1 Identity 组件 Keystone

OpenStack Identity 提供了对其他所有 OpenStack 项目进行身份验证的一种常见方法。每项多用户服务都需要一些机制来管理哪些人可以访问应用程序，以及每个人可以执行哪些操

作，私有云也不例外。OpenStack 已经将这些功能简化为一个单独的称为 Keystone 的项目。

Keystone 是 OpenStack Identity 的项目名称，该服务通过 OpenStack 应用程序编程接口 API 提供令牌、策略和目录功能。与其他 OpenStack 项目一样，Keystone 表示一个抽象层。它并不实际实现任何用户管理功能，而是会提供插件接口，以便使用者可以利用其当前的身份验证服务，或者从市场上的各种身份管理系统中进行选择。

Keystone 集成了用于身份验证、策略管理和目录服务的 OpenStack 功能，这些服务包括注册所有租户和用户，对用户进行身份验证并授予身份验证令牌，创建横跨所有用户和服务的策略，以及管理服务端点目录。身份管理系统的核心对象是用户，也就是使用 OpenStack 服务的个人、系统或服务的数字表示。用户通常被分配给称为租户的容器，该容器会将各种资源和身份项目隔离开来。租户可以表示一个客户、账户或者任何组织单位。

身份验证是确定用户是谁的过程。Keystone 确认所有传入的功能调用都源于声明发出请求的用户，通过测试凭证形式的声明来执行这一验证。凭证数据的显著特性就是它应该只供拥有数据的用户访问，该数据中可以只包含用户知道的数据（用户名称和密码或密钥）、用户通过物理方式处理的一些信息（硬件令牌），或者是用户的一些“实际信息”（视网膜或指纹等生物特征信息）。

在 OpenStack Identity 确认用户的身份之后，它会给用户提供一个证实该身份并且可以用于后续资源请求的令牌。每个令牌都包含一个作用范围，列出了对其适用的资源。令牌只在有限的时间内有效，如果需要删除特定用户的访问权限，也可以删除该令牌。

安全策略是借助一个基于规则的授权引擎来实施的。用户经过身份验证后，下一步就是确定身份验证的级别。Keystone 利用角色的概念封装了一组权利和特权。身份服务发出的令牌包含一组身份验证的用户可以假设的角色，然后，由资源服务将用户角色组与所请求的资源操作组相匹配，并做出允许或拒绝访问的决定。

Keystone 的一个附加服务是用于端点发现的服务目录。该目录提供一个可用服务清单及其 API 端点。一个端点就是一个可供网络访问的地址（例如 URL），用户可在其中使用一项服务。所有 OpenStack 服务，包括 OpenStack Compute（Nova）和 OpenStack Object Storage（Swift），都提供了 Keystone 的端点，用户可通过这些端点请求资源和执行操作。

Keystone 作为 OpenStack 的核心组件，为云主机管理 Nova、镜像管理 Glance、对象云存储 Swift 和界面仪表盘 Horizon 提供认证服务。以上这些组件的交互方式如图 2-2 所示。

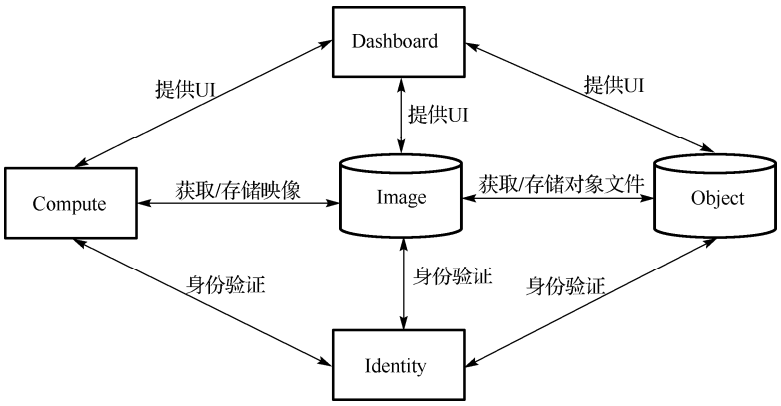

图 2-2　OpenStack 核心组件交互图

Keystone 作为 OpenStack 最早期的核心项目独立发展起来，由于 OpenStack 的核心设计理念是一切皆 API 化，所以涉及服务 API 的调用脱离不了 Keystone。Keystone 为整个 OpenStack 项目的其他子系统提供以下服务：用户信息管理，包括用户/租户基本信息、项目管理；认证服务，登录认证、鉴权管理，Keystone 通过角色及权限控制实现鉴权。

Keystone 涉及以下几个基本业务概念。

1）Service（服务）：Service 是基于 OpenStack 标准 REST API 对外提供的服务。例如云主机服务 Nova、弹性云硬盘服务 Cinder、认证服务 Keystone、镜像服务 Glance、网络服务 Neutron 等。

2）Endpoint（服务端点）：Endpoint 是指提供服务的 Server 端，一个可以通过网络访问的地址，以 URL 的形式表示。通常 OpenStack 的每一个服务都需要绑定 API 端点，例如，下面的命令就是为认证服务指定 API 端点，即创建一个端点并指定 public API，internal API 和 admin API 的 URL。

```
keystone endpoint-create \
  --service-id=$(keystone service-list | awk '/ identity / {print $2}') \
  --publicurl=http://control:5000/v2.0 \
  --internalurl=http://control:5000/v2.0 \
  --adminurl=http://control:35357/v2.0
```

3）Tenant（租户/项目）：使用 OpenStack 相关服务的一个用户组。租户可以是一个消费者（Consumer）、账户（Account）、组织（Organization）或者项目（Project）。一个用户（包括 admin user）必须至少属于一个项目，也可以属于多个项目。在 Nova 云主机中，Tenant 对应项目 Project，在其云存储中对应于 Account。

4）Role（角色）：对应于一个租户中的使用权限的集合，包括超级管理员 admin、普通成员 member 等。admin 具有管理员权限，member 使用租户内部的相关功能。

5）Token（令牌）：Keystone 成功验证用户身份后提供的凭证，用来实现单点登录，是各模块之间调用的认证令牌。

6）Credentials（凭证证书）：证明用户身份的数据，通常是用户名和密码、指纹识别或者是加密令牌等。

2.4.2 Storage 组件 Swift、Glance 及 Cinder

OpenStack 有三个与存储相关的组件，分别是对象存储 Swift、镜像存储 Glance 以及块存储 Cinder。其中 Glance 组件相对简单，主要是提供虚拟机镜像的管理功能；对象存储 Swift 出现时间较早，目前发展已经很成熟；块存储 Cinder 是比较新的组件，为虚拟机提供云硬盘（块设备）服务，并且和商业存储有结合的机会，所以有较好的发展和应用前景。

1）Swift：对象存储（Object Storage），提供弹性可伸缩、高可用的分布式对象存储服务，适合存储大规模非结构化数据。对象存储支持多种应用，如复制和存档数据、图像或视频服务，存储次级静态数据，存储容量难以估计的数据，为 Web 应用创建基于云的弹性存储等。Swift 具有很强的扩展性、冗余和持久性，与 Amazon 的简单存储解决方案 S3 API 兼容。

2）Glance：提供虚拟机镜像（Image）的存储和管理功能。镜像存储本身不存储大量的数据，需要挂载后台存储来存放实际的镜像数据。虚拟机镜像有三种配置方式：利用 OpenStack 对象存储机制来存储镜像；利用 Amazon S3 直接存储信息；或者将 S3 存储与对象存储结合起

来，作为 S3 访问的连接器。OpenStack 镜像服务支持多种虚拟机镜像格式，包括 VMware（VMDK）、Amazon 镜像（AKI、ARI、AMI）以及 VirtualBox 所支持的各种磁盘格式。

3）Cinder：块存储（Block Storage）服务，提供持久化块设备存储的接口。OpenStack 中的实例是不能持久化的，需要挂载卷（Volume），在卷中实现持久化。Cinder 就是提供对卷实际需要的存储块单元的管理功能，用户通过把块存储卷附加到虚拟机上实现虚拟机数据的持久化存储。

1．OpenStack 对象存储——Swift

Swift 是 OpenStack 开源云计算项目的子项目之一，它并不是文件系统或者实时的数据存储系统，它称为**对象存储**，主要用于永久类型的静态数据的长期存储，这些数据可以检索、调整或更新。Swift 前身是 Rackspace Cloud Files 项目，随着 Rackspace 加入到 OpenStack 社区，其于 2010 年 7 月贡献给 OpenStack，作为该开源项目的一部分。

Swift 具有极高的数据持久性（Durability）和很强的可扩展性，这里的可扩展性表现在两方面，一是数据存储容量的扩展，二是 Swift 性能的线性提升（如每秒查询率 QPS、吞吐量等）。由于通信方式采用非阻塞式 I/O 模式，所以极大地提高了系统吞吐和响应能力。

Swift 采用完全对称、面向资源的分布式系统架构设计，所有组件都可扩展，并且整个 Swift 集群中没有一个角色是单点的，能够有效地避免因单点失效而扩散并影响整个系统运转。对称架构意味着 Swift 中各节点可以完全对等，能极大地降低系统维护成本，并且易于扩容，只需简单地新增机器，系统便会自动完成数据迁移等工作，使各存储节点重新达到平衡状态。Swift 的元数据存储是完全均匀随机分布的，并且与对象文件存储一样，元数据也会存储多份，在架构和设计上保证了元数据信息的可靠存储。

Swift 的物理架构如图 2-3 所示，主要有三个组成部分：Proxy Server、Storage Server 和 Consistency Server。其中 Storage 和 Consistency 服务均允许部署在 Storage Node 上。为了统一 OpenStack 各个项目间的认证管理，认证服务目前使用 OpenStack 的认证服务 Keystone。

1）Proxy Server（代理服务）：用于对外提供对象服务 API，负责 Swift 其余组件间的相互通信，会根据环（Ring）的信息来查找服务地址，并转发用户请求至相应的账户、容器或者对象服务。由于采用无状态的 REST 请求协议，所以可以进行横向扩展来均衡负载。Proxy Server 也负责处理大量的失败，比如一个服务器不可用，它就会要求环为它寻找下一个接替的服务器，并把请求转发到那里。

2）Storage Server（存储服务）：提供了磁盘设备上的存储服务。在 Swift 中有三类存储服务：对象服务（Object Server）、容器服务（Container Server）和账户服务（Account Server）。

其中对象服务（Object Server）提供对象元数据和内容服务，每个对象的内容会以文件的形式存储在文件系统中，元数据会作为文件属性来存储，一般采用支持扩展属性的 XFS 文件系统。

容器服务（Container Server）提供容器元数据和统计信息，并维护所含对象列表的服务，每个容器的信息存储在一个 SQLite 数据库中。

账户服务（Account Server）提供账户元数据和统计信息，并维护所含容器列表的服务，每个账户的信息也被存储在一个 SQLite 数据库中。

3）Consistency Server：用于查找并解决由数据损坏和硬件故障引起的错误。主要有三项服务：审计服务（Auditor）、更新服务（Updater）和复制服务（Replicator）。

审计服务（Auditor）主要检查对象、容器和账户的完整性，如果发现比特级的错误，那么文件将被隔离，并复制其他副本以覆盖本地损坏的副本，其他类型的错误会被记录到日志中。

复制服务（Replicator）用以检测本地分区副本和远程副本是否一致，具体是通过对比散列文件和高级水印来完成，发现不一致时会采用推式（Push）更新远程副本，另外一个任务是确保被标记删除的对象从文件系统中移除。

更新服务（Updater）主要负责更新处理，当对象由于高负载的原因而无法立即更新时，任务将会被序列化到本地文件系统中进行排队，以便服务恢复后进行异步更新。例如成功创建对象后容器服务器没有及时更新对象列表，这个时候容器的更新操作就会进行排队，更新服务会在系统恢复正常后扫描队列并进行相应的更新处理。

审计服务在每个 Swift 服务器的后台持续地扫描磁盘来检测对象、容器和账号的完整性。如果发现数据损坏，审计服务就会将该文件移动到隔离区域，然后由复制服务负责用一个完好的复制来替代该数据。

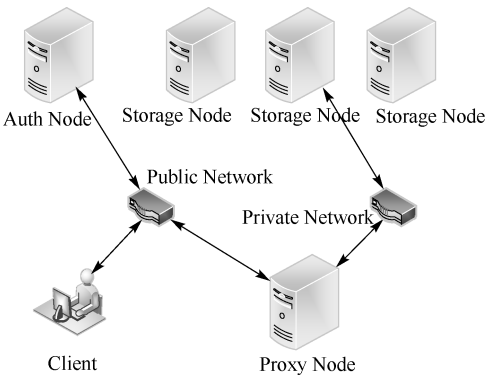

图 2-3　Swift 物理架构

2．OpenStack 镜像存储——Glance

Glance 实现虚拟机镜像模板的管理，是 Nova 系统架构中非常重要的模块。镜像管理的目标就是提供镜像的存储访问管理，OpenStack 将 Glance 独立出来的一个原因是尽可能将镜像存储至多种存储设备上。Glance 提供一个完整的适配框架，支持亚马逊对象存储 S3、OpenStack 自有的 Swift 对象存储，以及常用的文件系统存储。当然也可以自行开发拓展到别的存储上，例如 HDFS。下面对 Glance 的基本功能进行剖析。

1）镜像管理：包括镜像创建，基本信息更新，镜像文件上传、下载等。镜像可以看成虚拟机的模板，用于生成虚拟机实例。

2）快照管理：快照也是一种镜像，对虚拟机创建快照后，可以基于快照部署虚拟机实例。例如，部署一台 Linux 虚拟机，其上运行 Ubuntu，在该虚拟机上安装 Web 应用，安装调试完毕后，为该虚拟机当前状态做一个快照。以后如果需要部署该 Web 应用，可以直接通过该快照部署，几分钟内 Web 应用即可部署完毕，并且增加至负载均衡里面可实现应用处理能力的水平扩展。

3）镜像的存储管理：镜像特别是虚拟机快照类镜像，随着使用规模的扩大，镜像的数量和容量需求在不断扩大，对存储要求越来越高。用传统的文件系统难以解决大规模海量存储

要求，所以镜像的存储需要进行扩展，可以通过分布式文件系统或对象云存储的方式实现镜像文件存储。

镜像管理包括三个组成部分，分别是 API 接口服务、注册服务和存储适配器，其功能架构如图 2-4 所示。

1）API 接口服务 Glance-API：对外提供镜像接口服务，包括镜像的上传和下载、更改信息，以及虚拟机、云硬盘快照管理等接口服务；

2）注册服务 Glance-Registry：存储镜像元数据信息，与数据库交互实现镜像基础信息存储；

3）存储适配器 Store Adapter：存储镜像文件，提供多种存储适配，支持 S3 存储、Swift 存储以及文件系统等。

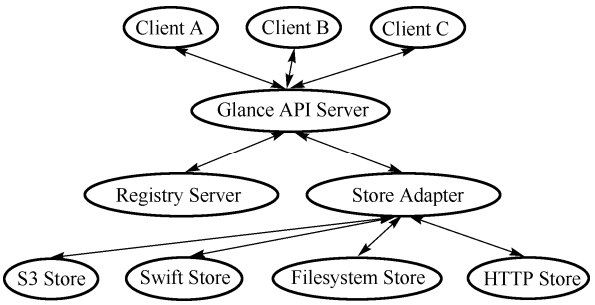

图 2-4　Glance 功能架构图

从 Glance 的功能架构图可以看出，API 接口服务调用注册服务和存储适配。在实际部署中，Glance-API 和 Glance-Registry 可以分离，所以 Glance-API 接口服务访问 Glance-Registry 注册服务需要通过远程 HTTP 方式访问，而接口服务与存储适配是通过本地接口调用实现的。

3．OpenStack 块存储——Cinder

在 OpenStack 的 Folsom 版本中，将之前 Nova 中的部分持久性块存储功能（Nova-Volume）分离了出来，独立为新的组件 Cinder。Cinder 的功能是实现块存储服务，根据实际需要快速为虚拟机提供块设备的创建、挂载、回收以及快照备份控制等。它并没有实现对块设备的管理和实际服务，而是为后端不同的存储结构提供了统一的接口，不同的块设备服务厂商在 Cinder 中实现其驱动支持，以与 OpenStack 进行整合。

Cinder 包括 API、调度 Scheduler 和存储适配 Cinder-Volume 三项服务，其中 Cinder-Scheduler 根据服务寻找合适的服务器 Cinder-Volume，发送消息到 Cinder-Volume 节点，由 Cinder-Volume 提供弹性云存储服务。Cinder-Volume 可以部署到多个节点上。其架构如图 2-5 所示。

1）Cinder-API：解析所有传入的请求并将它们转发给消息队列。

2）Cinder-Scheduler：调度程序，根据预定策略选择合适的块存储服务节点来执行任务。在创建新的卷时，该调度器选择卷数量最少的一个活跃节点来创建卷。

3）Cinder-Volume：该服务运行在存储节点上，负责管理存储空间，通过消息队列直接在块存储设备或软件上与其他进程交互。每个存储节点都有一个块存储服务，若干个这样的存储节点联合起来可以构成一个存储资源池。

Cinder 通过添加不同厂商的指定驱动来支持不同类型和型号的存储，目前能支持的商业存储设备有 EMC 和 IBM 的几款产品，也能通过 LVM 支持本地存储。对于本地存储，

Cinder-Volume 使用 LVM 驱动，需要在主机上事先用 LVM 命令创建一个 Cinder-Volumes 的卷组，当该主机接收到创建卷请求的时候，Cinder-Volume 在该卷组上创建一个逻辑卷，并且用 OpeniSCSI 将这个卷当成一个 iSCSI TGT 输出。虽然从管理的角度来看可以解决存储共享的问题，但是这样的设计对于本地存储的管理会产生较大的性能损耗，因为和直接访问相比，通常 iSCSI 导出会增加 30%以上的 IO 延迟。从目前的实现来看，Cinder 对本地存储和 NAS 的支持比较好，可以提供完整的 Cinder API V2 支持，而对于其他类型的存储设备，Cinder 的支持会受到一些限制。

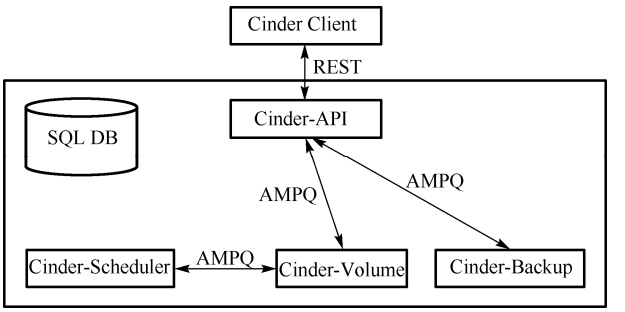

图 2-5　Cinder 的架构图

2.4.3　Compute 组件 Nova

OpenStack Compute 是云计算系统的结构控制器，它的功能是根据用户需求提供计算服务，配置虚拟机规格，负责对虚拟机进行创建并管理虚拟机实例的整个生命周期。OpenStack Compute 这个名称指的是一个特定的项目，该项目也被称为 Nova，但与计算和运行计算相关的组件有两个：镜像管理 Glance 和计算管理 Nova。其中 Glance 包含可执行代码和操作环境的静态磁盘镜像，Nova 用以管理正在运行的虚拟机实例。

Nova 是基础架构服务核心，也是 OpenStack 家族中最复杂的组件，具有高度分散的性质和多个流程。Nova 与其他几个 OpenStack 服务都有一些接口：它使用 Keystone 来执行其身份验证，使用 Horizon 作为其管理接口，并用 Glance 提供其镜像。Nova 与 Glance 的交互最为密切，它需要下载镜像，以便通过镜像来创建虚拟机。虽然 Nova 本身不包括任何虚拟化软件，但它可以通过与虚拟化技术有关联的驱动程序来集成许多常见的虚拟机管理程序。

启动虚拟机实例涉及识别并指定虚拟硬件模板（在 OpenStack 中称为 Flavor）。模板描述被分配给虚拟机实例的计算（虚拟 CPU）、内存（RAM）和存储配置（硬盘）。OpenStack 的默认安装提供了五种模板，它们由管理员配置。

Nova 通过将执行分配给某个特定的计算节点（在 OpenStack 中称为**主机**）来调度被请求的实例。每个 OpenStack 子系统都必须定期报告其状态和能力，调度程序使用数据来优化其分配。整个分配过程由两个阶段组成：Filtering（过滤）阶段和 Weighting（加权）阶段。Filtering（过滤）阶段应用一组过滤器生成最适合的主机的列表。调度程序将会缩小主机的选择范围，以找到符合请求参数的主机，每个 OpenStack 服务的能力是影响选择的重要因素之一。在 Weighting（加权）阶段使用一个特殊的函数来计算每个主机的成本，并对结果进行排序。这个阶段的输出是一个主机列表，这些主机可用最少的成本满足用户对给定数量的实例的请求。

Nova 还执行了其他一些函数，这些函数与涵盖网络、安全性和管理的其他 OpenStack 项目有密切的交互。但 Nova 一般处理这些项目中与实例相关的方面，如存储的连接和取消，分配 IP 地址，或创建运行实例的快照。

Nova 采用的是无共享架构，如图 2-6 所示，这样所有的主要部件都可以在不同的服务器上运行，分布式设计依赖于一个消息队列来处理组件对组件的异步通信。

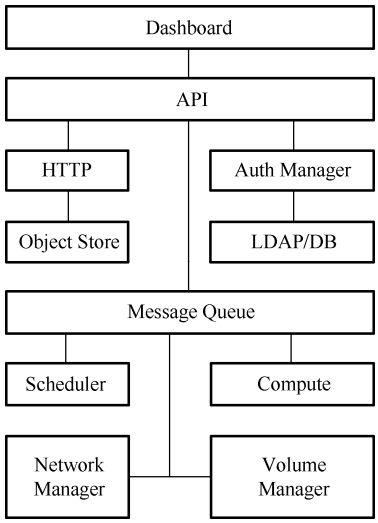

图 2-6　Nova 的无共享架构

Nova 将虚拟机的状态存储在一个基于结构化查询语言 SQL 的中央数据库中，所有的 OpenStack 组件都使用该数据库。该数据库保存了可用实例类型、网络和项目的详细信息。

OpenStack Compute 的用户界面是 Web 仪表板（Dashboard）。这也是所有 OpenStack 模块的中央门户，为所有项目提供图形界面，并执行应用程序编程接口（API）来调用被请求的服务。

Nova API 负责接收 HTTP 请求，处理命令，然后将任务通过消息队列或 HTTP（在使用对象存储的情况下）委派给其他组件。Nova API 支持 OpenStack Compute API、Amazon Elastic Compute Cloud（Amazon EC2）API，以及面向特权用户的 Admin API。

Authorization Manager 负责身份验证，每个 HTTP 请求都需要一个特定的身份验证凭据。它暴露了用户、项目和角色的 API 授权使用情况，并与 OpenStack Keystone 进行通信，以便获得详细信息。任何 OpenStack 组件都可以使用它进行身份验证。

Object Store 是一个基于 HTTP 或基于对象的简单存储（如 Amazon Simple Storage Service），专门针对镜像，通常可以用 OpenStack Glance 取代它。

消息队列 Message Queue 是为 Nova 中的所有组件提供相互通信、相互协调的一种手段，是所有 Nova 组件都共享和更新的一个任务列表。Nova 的组件都在一个非阻塞的基于消息的架构上运行，只要它们使用了相同的消息队列服务，就可以在相同或不同的主机上运行。这些组件使用 AMQP（Advanced Message Queuing Protocol）协议，以面向回调的方式进行交互。

Nova 有两个主要的守护进程：调度程序 Scheduler 和计算进程 Compute。调度程序 Scheduler 确定为虚拟机请求分配哪个计算主机。它采用了过滤和调度算法，并考虑多种参数，包括亲和性（与共置相关的工作负载）、反亲和性（分发工作负载）、可用区、核心 CPU 使用率、系统内存等。计算进程 Compute 是一个执行守护进程，用于管理虚拟机管理程序和虚拟机的通信。它从消息队列中检索其订单，并使用虚拟机管理程序的 API 执行虚拟机的创建和删除任务。计算进程还在中央数据库中更新其任务状态。

Network Manager 负责管理 IP 转发、网桥和虚拟局域网。它是一个守护进程，从消息队列读取与网络有关的任务。

Volume Manager 负责处理将持久存储的块存储卷附加到虚拟机，以及从虚拟机分离块存储卷（类似于 Amazon Elastic Block Store）。此功能现在已被分离到 OpenStack Cinder。

2.4.4　Network 组件 Neutron

OpenStack Networking 管理其他 OpenStack 项目之间的连接性。要在计算节点之间建立连

接并访问外部网络，可以利用现有的网络基础架构来分配 IP 地址并在节点之间传输数据。但在多租户环境中，已有的网络管理系统无法高效安全地在用户之间隔离流量，这是构建公有云和私有云时面临的一个巨大问题。

OpenStack 解决此问题的一种方式是，构建一个详尽的网络管理堆栈，用以处理所有网络相关请求。此方法面临的挑战是，每个实现都可能拥有一组独特的需求，包括与其他各种工具和软件的集成。OpenStack 为此采取了创建抽象层的方法，这个抽象层被称为 **OpenStack Networking**，可容纳大量与其他网络服务集成的插件。它为云租户提供了一个应用编程接口（API），租户可使用它配置灵活的策略和构建复杂的网络拓扑结构，例如用它来支持多级 Web 应用程序。

OpenStack Networking 支持使用第三方插件来引入高级网络功能，例如 L2-in-L3 隧道和端到端的服务质量支持。它们还可以创建网络服务，例如负载平衡、虚拟专用网或插入 OpenStack 租户网络中的防火墙。

在 OpenStack 的早期版本中，网络组件位于 OpenStack Nova（Compute）项目中。之后大部分组件被拆分为一个单独项目，最初称为 Quantum，后来重命名为 Neutron，以避免与公司 Quantum Corporation 的任何商标混淆。所以，在 OpenStack Networking 参考资料中经常会同时出现名称 Nova、Quantum 和 Neutron，这三个术语都与描述 OpenStack 的网络服务有关。

1．模型

OpenStack Networking API 基于一个简单的模型（包含虚拟网络、子网和端口抽象）来描述网络资源。网络是一个隔离的 2 层网段，类似于物理网络中的虚拟 LAN（VLAN）。具体来讲，它是为租户而保留的一个广播域，或者被显式地配置为共享网段。端口和子网始终被分配给某个特定的网络。

子网是一组 IPv4 或 IPv6 地址以及与其有关联的配置。它是一个地址池，OpenStack 可以从地址池中为虚拟机（VM）分配 IP 地址。每个子网被指定为一个 CIDR（Classless Inter-Domain Routing 无类别域间路由）范围，必须与一个网络相关联。除了子网之外，租户还可以指定一个网关、一个域名系统（DNS）服务器列表以及一组主机路由。这个子网上的 VM 实例随后会自动继承该配置。

端口是一个虚拟交换机连接点，一个 VM 实例可通过此端口将它的网络适配器附加到一个虚拟网络。在创建之后，一个端口可以从指定的子网收到一个固定 IP 地址，API 用户可以从地址池请求一个特定的地址，或者由 Neutron 分配一个可用的 IP 地址。在取消分配端口后，所有已分配的 IP 地址都会被释放并返回到地址池。OpenStack 还可以定义接口使用的媒体访问控制地址。

2．插件

最初的 OpenStack Compute 网络实现了采用一种基本模型，通过 Linux VLAN 和 IP 表执行所有隔离操作。OpenStack Networking 引入了插件的概念，插件是 OpenStack Networking API 的一种后端实现，可使用各种不同的技术来实现逻辑 API 请求。插件架构使云管理员可以非常灵活地自定义网络的功能。第三方可通过 API 扩展提供额外的 API 功能，这些功能最终会成为核心 OpenStack Networking API 的一部分。

Neutron API 向用户和其他服务公开虚拟网络服务接口，但这些网络服务的实现位于一个

插件中，插件向租户和地址管理等其他服务提供隔离的虚拟网络。任何人都能够通过 Internet 访问 API 网络，而且该网络实际上可能是外部网络的一个子网。Neutron API 公开了一个网络连接模型，其中包含网络、子网和端口，但它并不实际执行工作，Neutron 插件负责与底层基础架构交互，以便依据逻辑模型来传送流量。

一些 OpenStack Networking 插件可能使用基本的 Linux VLAN 和 IP 表，这些插件对于小型和简单的网络通常已经足够，但更大型的客户可能拥有更复杂的需求，涉及多级 Web 应用程序和多个私有网络之间的内部隔离。例如需要自己的 IP 地址模式（这可能与其他租户使用的地址重复），用来允许应用程序在无须更改 IP 地址的情况下迁移到云中。在这些情况下，就需要采用更高级的技术，比如 L2-in-L3 隧道或 OpenFlow。

现在 Network 组件已有大量包含不同功能和性能参数的插件，而且插件数量仍在增长。目前包含的插件有：Open vSwitch、Cisco UCS/Nexus、Linux Bridge、Nicira Network Virtualization Platform、Ryu OpenFlow Controller、NEC OpenFlow。云管理员可自行选择插件，他们可评估各个选项并根据具体的安装需求进行调整。

3．架构

Neutron 的系统架构及内部组成如图 2-7 所示。

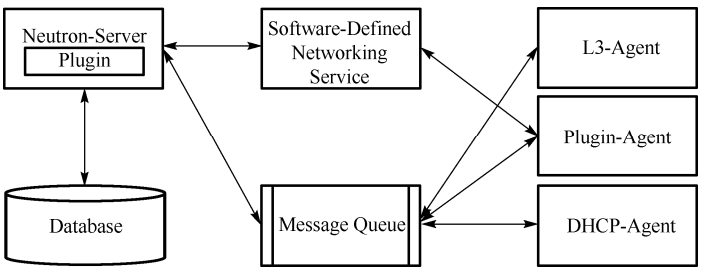

图 2-7　Neutron 系统架构图

Neutron Server 包含守护进程 neutron-server 和各种插件 neutron-*-plugin，它们既可以安装在控制节点也可以安装在网络节点。neutron-server 提供 API 接口，并把对 API 的调用请求传给已经配置好的插件进行后续处理。插件需要访问数据库来维护各种配置数据和对应关系，例如路由器、网络、子网、端口、浮动 IP 和安全组等。

OpenStack Networking 包含三个代理，它们通过消息队列或标准 OpenStack Networking API 与主要 Neutron 进程进行交互，这三个代理分别是 DHCP 代理 neutron-dhcp-agent、三层代理 neutron-l3-agent 和插件代理 neutron-*-agent。DHCP 代理向所有租户网络提供动态主机配置协议（Dynamic Host Configuration Protocol，DHCP）服务；三层代理执行 L3/网络地址转换（Network Address Translation）转发，以支持访问租户网络上的 VM；插件代理是一个特定于插件的可选代理（neutron-*-agent），在每个虚拟机管理程序上执行本地虚拟交换机配置。一般来说每一个插件都有对应的代理，选择了什么样的插件，就需要选择对应的代理。

OpenStack Networking 与其他 OpenStack 组件之间的交互方式是：OpenStack Dashboard（Horizon）提供图形用户界面，以便管理员和用户能够访问创建和管理网络服务的功能。这些服务也依照 OpenStack Identity（Keystone）对所有 API 请求执行身份验证和授权。当 Nova 启动了一个虚拟机实例时，Nova 服务会与 OpenStack Networking 通信，将每个虚拟网络接口接入到一个特定的端口中。

本 章 小 结

本章介绍了开源 IaaS 云平台 OpenStack，它是一个旨在为公有云及私有云的建设与管理提供软件的开源项目。OpenStack 框架由核心和扩展的项目构成，通过标准化公用服务接口 API 实现集成，子系统和服务之间通过 API 互相调用，这些项目相互协作为用户提供特定的服务。

OpenStack 的核心子项目包括：计算管理 Nova、镜像管理 Glance、网络管理 Neutron、认证管理 Keystone、块存储管理 Cinder、对象存储管理 Swift 以及 Web 界面管理 Horizon。其中 Nova 是云计算系统的结构控制器，根据用户需求将计算能力以虚拟机的方式提供给用户，负责管理虚拟机实例的整个生命周期。Glance 提供虚拟机镜像存储和管理，如查询、存储和检索等。Neutron 实现虚拟机的网络资源管理，提供比较完善的多租户环境下的虚拟网络模型。Keystone 为 OpenStack 所有的系统提供统一的授权和身份验证服务。Cinder 实现块存储服务，根据实际需要快速为虚拟机提供块设备的创建、挂载、回收以及快照备份控制等。Swift 主要用于永久类型的静态数据的长期存储，这些数据可以检索、调整或更新。Horizon 是基于 OpenStack API 接口开发的 Web 呈现，是用户使用云平台的界面。

思 考 题

1．目前主流的 IaaS 云计算解决方案主要有哪些？

2．OpenStack 包括哪些核心项目？这些项目分别提供什么服务？

3．在 OpenStack 中，项目、用户、角色之间是什么关系？你对 admin 角色是如何理解的？

4．OpenStack 的虚拟机是各组件进行交互的结果，OpenStack 在创建虚拟机时 Nova、Glance、Neutron、Keystone 之间是怎样的交互关系？

5．OpenStack 有三个与存储相关的组件，它们分别适用于怎样的场景？