第3章

CHAPTER 3

网络服务

云计算是一种按需交付IT资源的服务模式，这种模式正在成为企业进行IT变革的主流。网络作为云计算的骨架，是IaaS服务的基础。因此，云计算的服务能力对网络的依赖程度极高，云计算厂商的实力很大程度上也体现在网络上。

云计算的资源交付方式是动态的、按需的，这与传统网络的建设方式有很大区别。传统网络都是根据计算、存储、数据库等的部署需求，提前做好规划后再实施，一旦网络交付，就不会再做大的架构性调整。云计算的出现打破了这种交付方式，云计算模式下，计算资源被虚拟化，业务部署在虚拟机或容器上；分布式存储开始大规模应用，并按需给用户分配存储资源。这些资源的交付，都可以在分钟级完成，也就意味着网络要在相同的时间内做好部署，因此网络要具备快速交付能力，并按业务需要进行灵活调整。同时，大规模的云计算运营，需要在不同地域建设多个可用区以提供资源扩展能力和高可用的容灾能力。

根据服务方式和在云计算中的角色，将云计算中的网络分为基础网络和虚拟网络两部分。基础网络由数据中心网络、数据中心互联网络和互联网交换中心网络组成，其中数据中心网络主要满足服务器等资源的高速接入，数据中心互联网络将云计算中的所有数据中心连接起来，而互联网交换中心则将不同数据中心的互联网出口统一调度起来与运营商网络对接，实现互联网出口资源的整合，提升互联网访问的体验。

3.1 数据中心网络

数据中心网络(Data Center Network)是数据中心内部的网络，提供物理服务器和数据中心内所有设备的接入。可以将数据中心比作信息技术领域的心脏，因为重要的数据资源和IT系统都部署在数据中心里，所以数据中心网络一直是IP网络领域的制高点，是网络设备厂商的必争之地，也是互联网公司和大型云计算公司重点研究的领域。数据中心网络架构的规模性、可扩展性、健壮性和可靠性一直都是被关注的重点，随着云计算的发展，数据中

心网络架构也在不断演进。

数据中心网络经历了 MSTP、堆叠(VSS、IRF)、跨设备链路聚合(思科的 vPC、华为的 M-Lag)、Trill、VXLAN 等技术的演进。在组网模式上,传统的三层架构主要将网关部署在核心或汇聚层,接入交换机运行在二层(数据链路层),仅作服务器接入,如图 3.1 所示。但随着互联网业务的发展,为了支撑更大规模的业务量,互联网公司在做数据中心架构时,开始将网关下沉到接入交换机,全网运行三层路由协议(BGP 或 OSPF),将数据中心划分为多个 POD。但这种方式,要求每个接入交换机上的网关都不一样,服务器还是无法形成统一的资源池。

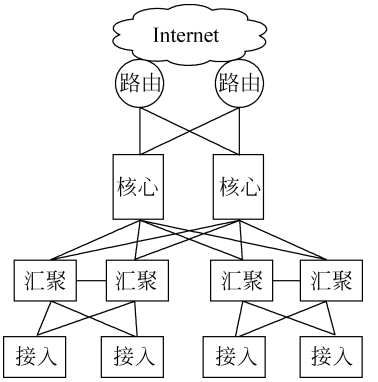

图 3.1　传统数据中心网络架构

虚拟化技术在云计算中被广泛应用,大量的虚拟机和容器部署于云数据中心,这会带来两个最直接的结果:一是网络流量的爆发式增长;二是数据中心网络流量模型从南北向为主转变为了以东西向为主。虚拟机之间的互通、虚拟机的迁移、存储数据的同步等,更多的带宽将被东西向流量所占用,数据中心网络的架构也从传统的烟囱式架构向分布式架构转变,从纵向扩展向横向扩展演进。

可以看到数据中心网络架构的演进有两个趋势:一是 CLOS 网络架构(见图 3.2)被规模应用,让网络具备更好的弹性扩展能力,数据中心网络开始由传统的"核心+汇聚+接入"的三层架构向"骨干节点(Spine)+叶子节点(Leaf)"的二层架构演进;二是 25G/100G 端口接入正在逐渐成为主流,为云计算业务提供更大的带宽支持。同时,随着人工智能、大数据和高性能分布式存储等业务的发展,应用对时延提出了更高的要求,RDMA(Remote Direct Memory Access,远程直接数据存取)网络也开始纳入数据中心网络的架构设计中。

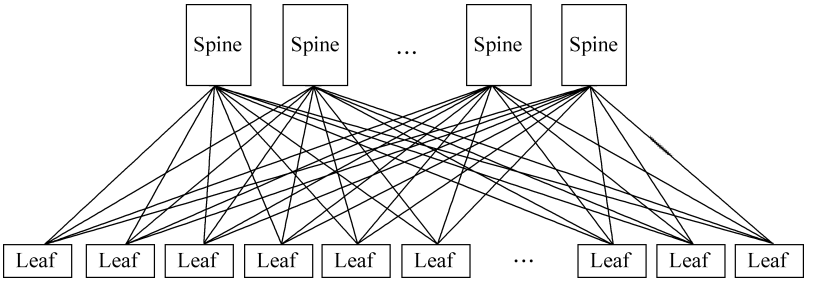

图 3.2　数据中心 CLOS 网络架构

随着 VXLAN、SDN 等技术的出现,云计算场景下,业界开始将数据中心网络抽象为 Underlay 网络(物理网络)和 Overlay 网络(叠加网络),将负责 VXLAN 封装的 VTEP(VXLAN Tunnel End Point,VXLAN 隧道端点)下沉到接入交换机或者物理服务器,数据中心的物理网络只需要提供三层可达的通道即可,而 CLOS 网络架构因其具备非常好的可

扩展性而被普遍采用，已成为当前数据中心网络架构的主流。

3.2　数据中心互联网络

数据中心互联网络(Data Center Interconnect Network，DCI)对云计算也具有非常重要的作用。无论从资源的扩展性，还是从业务的高可用角度，多数据中心都是云计算建设的必然选择。因为业务的发展，数据中心的建设也不仅仅停留在国内，多数云厂商在国外也会有站点布局。因此，需要一张高带宽、高可靠的骨干网络，将各数据中心连接起来。DCI 网络架构如图 3.3 所示。

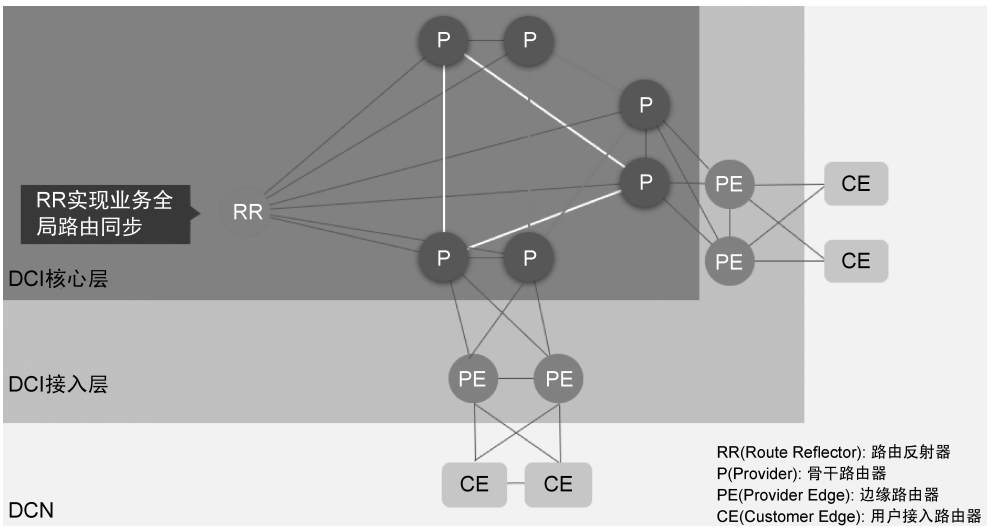

图 3.3　DCI 网络架构

所有跨数据中心的互通都会通过 DCI 网络实现。业务流量的不确定性和外部线路的稳定性会影响 DCI 网络的可靠性。同时，业务的多样性和访问的随机性都需要 DCI 网络能够隔离传输并区分对待不同的业务，不同云和业务存在多种接入类型和需求，这就要求 DCI 网络具有丰富的接入能力和灵活的接入方式。基于 DCI 网络的上述特点，以及架构简化和高效运维的原则，通常会把骨干和接入分离，在骨干网络构建专门的转发面，与边缘不同功能的接入面分开，骨干核心架构负责全网流量的智能调度和快速转发，边缘架构则实现不同业务的分类接入和差分服务。

DCI 网络根据地域可划分为同城骨干网和全球骨干网，从协议栈的角度分为底层的光传输网络和上层承载业务的 IP 网络，而云计算则在 IP 承载网络中划分出多个虚拟网络，为用户提供跨数据中心的通信服务。DCI 网络的建设，首先需要构建起一个基础的光传输网络。在国内，光传输网络的建设主要是与运营商合作，而在国外，有条件的云服务商还可以选择自建光缆。在光传输网络的基础上，使用路由器搭建起 IP 网络。当前的 DCI 网络在

接入方面，除使用 MPLS 技术提供二、三层 VPN 接入外，还通过 EVPN＋VXLAN 的形式提供二、三层 VPN 接入；而在流量智能调度方面，分布式 RSVP-TE 仍然是现网的主流方案，随着 SDN 的兴起，通过控制器进行全局集中式调度的方案也已经开始试点，SR-TE 已经出现少量的商用案例，SRv6 更是打破多协议网络的复杂局面，大大简化了网络的封装隔离和流量的按需调度。

3.3 互联网交换中心网络

互联网交换中心（Internet Exchange Point，IXP）原本是互联网中用于满足不同运营商连通而建立的交换平台，一般由第三方中立运营，国外知名的 IX（Internet Exchange，互联网交换）有 AMS-IX、HKIX、Equinix IX 等，国内知名的第三方 IX 有位于上海的 We IX（驰联网络）、位于北京的 CNISP 等。随着国内互联网业务的快速发展，以腾讯、阿里和百度为代表的大型互联网公司都在建设自己的互联网交换中心，但这个交换中心仅用于各运营商访问该互联网公司的内容。为了让用户获得更好的互联网访问体验，有实力的云计算厂商也会考虑建立自己的互联网交换中心，以满足同城和跨地域的数据中心之间互联网出口流量的调度需求。

这里提到的 IXP 主要指云计算服务商为提供互联网访问服务、提高用户使用体验而建设的网络平台。初始阶段的 IXP 受制于数据中心所在的基础运营商环境，云的互联网出口本地化使用大大限制了互联网的接入类型和覆盖质量，无法实现数据中心资源和互联网出口资源的解耦。随着 DCI 网络的发展和可靠性的提升，通过结合 DCI 网络把各地域的互联网出口资源整合起来，构建出口资源池，实现云的可用区和互联网出口架构松耦合，各自向优发展，在公有云的可用区之外形成出口调度平台。出口调度平台跟随 DCI 网络扩展到海内外，最终实现云服务商多个地域 IT 资源灵活的互联网接入环境，不仅丰富对接的资源、优化互联网覆盖的质量，而且大大降低了互联网出口的成本。

3.4 虚拟专有网络

3.4.1 网络虚拟化的演进

在网络领域，虚拟化并不是一个新概念。在网络发展之初，为了方便网络管理和隔离，就提出了 VLAN 技术，随后在广域网场景，又有 VPN 技术，根据业务场景的不同，分别有基于 MPLS 的三层 VPN 和基于 VPLS 的二层 VPN 等。在数据中心场景，网络虚拟化也经历了将多台交换机虚拟成一台设备的多虚一技术，以及将一台交换机虚拟成多台交换机的一虚多技术。多虚一技术有 IRF（Intelligent Resilient Framework，智能弹性架构）、VSS（Virtual Switch System，虚拟交换机系统）、CSS（Cluster Switch System，集群交换机系统）；一虚多技术有 MDC（Multitenant Device Context，多租户设备环境）。从路由转发角

度看，也有虚拟路由转发（Virtual Routing Forwarding，VRF）技术，通过软件的方式，在一台硬件设备上实现逻辑上的转发面隔离，每个 VRF 有独立的路由表。

传统的网络虚拟化技术并不能解决云计算时代网络所面临的主要矛盾。以实现逻辑隔离的 VLAN 技术为例，在大规模云网络环境部署中有两个局限性：一是 VLAN 基于第二层网络进行标签隔离，在存在大量虚拟机的云计算环境中，虚拟机可能会被调度到数据中心网络中的任何一台物理服务器，而一个 VLAN 的范围不能无限制放大，配置到所有接入交换机；二是 IETF 定义的标准协议中，VLAN ID 长度为 12 位，取值范围为 0～4095，可用数量为 4096，而在云计算环境中，对网络隔离的需求将远不止 4096 个虚拟网络。

在云计算场景下，对网络虚拟化的理解主要是两个维度：一是在一个物理网络可以虚拟出多个相互隔离的虚拟网络，从而不同用户可以使用独立的网络资源切片，在满足用户网络隔离需求的同时，提高网络资源利用率；二是通过 NFV（Network Function Virtualization，网络功能虚拟化）的方式，将部分网络功能虚拟化，用标准化的服务器替代专有硬件设备，以更灵活的方式提供网络服务。

1. Overlay 网络虚拟化技术

鉴于传统虚拟化技术的局限性，业界开始采用新的虚拟化技术来满足云计算对虚拟网络隔离的要求，当前的主要思路是采用 Overlay（逻辑隔离）技术。Overlay 在网络领域指的是一种物理网络架构上叠加虚拟网络的模式，其大体框架是对基础网络不进行大规模修改的条件下，实现应用在网络上的承载，并能与其他网络业务分离。Overlay 网络是建立在物理网络上的虚拟网，逻辑节点和逻辑链路构成了 Overlay 网络。广义上讲，VPN 其实也是一种 Overlay，但在数据中心场景下，Overlay 主要指通过 VXLAN、GRE 等技术实现的数据中心网络虚拟化。针对上文提到的 VLAN 技术的两个局限性，Overlay 在很大程度上提供了全新的解决方案。

（1）虚拟机创建的灵活性：Overlay 是一种封装在 IP 报文之上的新的数据格式，可以在三层路由的网络中建立起逻辑的二层网络，因而具备大规模扩展能力。同时，IP 网络本身具备很强的故障自愈和负载均衡能力，采用 Overlay 技术后，利用现有网络进行技术迭代，便可用于支撑新的云计算业务。

（2）针对网络隔离规模的限制：在 Overlay 技术中引入了类似 VLAN ID 的用户标识，但对标识的范围做了很大的扩展，如 VXLAN 技术引入的 VXLAN ID，支持上千万级别的用户标识。

Overlay 网络由边缘设备、控制平面和转发平面组成。边缘设备是指与虚拟机直接相连的设备；控制平面主要负责虚拟隧道的建立维护以及主机位置性信息的通告；转发平面是承载 Overlay 报文的物理网络，如图 3.4 所示。

Overlay 的类型可以分为网络 Overlay、主机 Overlay 和混合式 Overlay 三大类，如图 3.5 所示。

（1）网络 Overlay 指的是隧道封装和解封装在接入交换机完成，通过隧道协议构建起

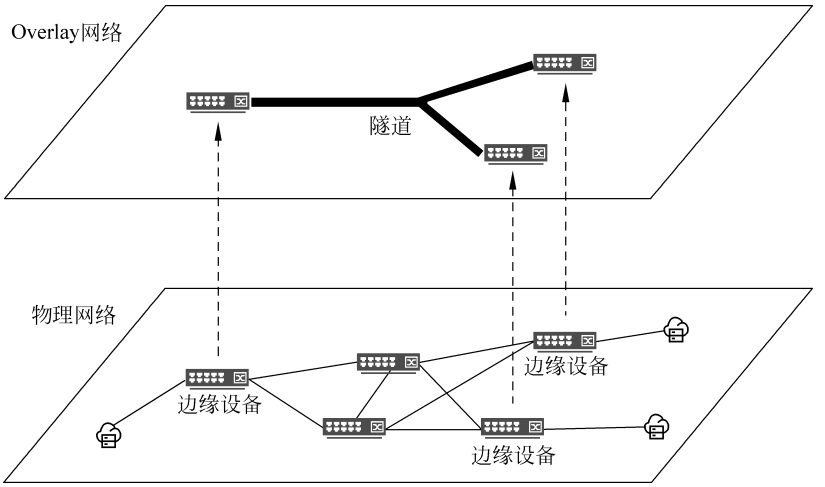

图 3.4　Overlay 体系架构

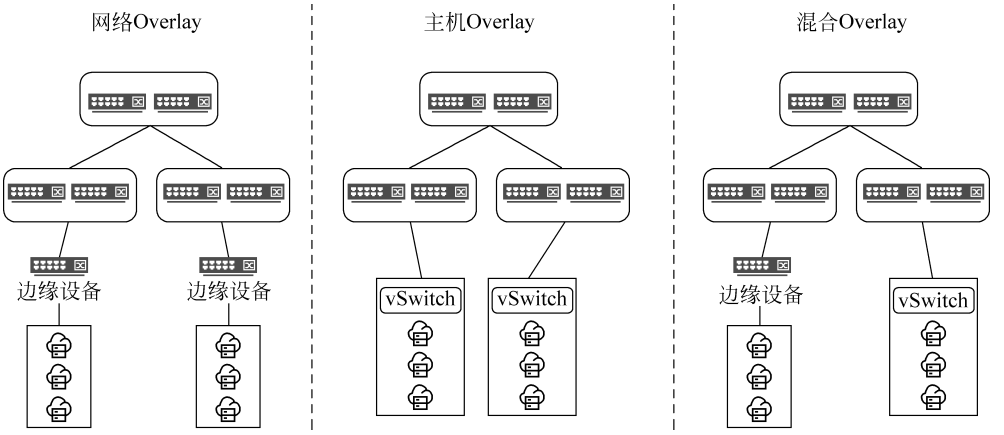

图 3.5　Overlay 的类型

逻辑二层网络。

（2）主机 Overlay 指的是隧道封装在服务器的 vSwitch 上完成，不用增加新的网络设备即可完成 Overlay 部署，可以支持虚拟化的服务器之间组网互通。

（3）混合 Overlay 指的是采用网络 Overlay 和主机 Overlay 的混合组网，可以支持物理服务器和虚拟服务器之间的组网互通。混合 Overlay 采用软硬件结合的方式，使得软硬件都能发挥自己的优势，也保障了 Overlay 网络的整体性能。

Overlay 在现阶段有如下三大主流技术路线：VXLAN、NVGRE 和 STT。这三种 Overlay 技术，大体思路均是将以太网报文承载到某种隧道层面，差异在于选择和构造隧道的实现方式不同，而底层均是 IP 转发。

表 3.1 展示了这三种技术的主要特性对比。VXLAN 和 STT 对网络设备的流量平衡

要求较低，即对链路的负载分担适应性较好，一般网络设备可以对 L2～L4 数据进行链路聚合或等价路由流量负载分担；NVGRE 要求网络设备检测 GRE 扩展头并对 Flow ID 执行哈希算法，对硬件要求比较高；STT 对 TCP 进行了修改，让隧道模式接近 UDP 性质，技术相对复杂；VXLAN 利用通用的 UDP 传输技术，适用性很强。总体来看，VLXAN 技术优势明显。

表 3.1　Overlay 三大主流技术的主要特性对比

技术名称	支　持　者	支持方式	网络虚拟化方式	数据新增报头长度	链路哈希算法能力
VXLAN	Cisco/VMWare/Citrix/RedHat/Broadcom	L2 over UDP	VXLAN 报头 24 位 VNI	50 字节（＋原数据）	现有网络可进行 L2～L4 哈希算法
NVGRE	HP/微软/Broadcom/Dell/Intel/Emulex	L2 over GRE	NVGRE 报头 24 位 VSI	42 字节（＋原数据）	GRE 头的哈希算法需要网络升级
STT	VMWare(Nicira)	L2 over TCP（无状态 TCP，即 L2 在类似 TCP 的传输层）	STT 报头 64 位 Context ID	58～76 字节（＋原数据）	现有网络可进行 L2～L4 哈希算法

由于 VXLAN 是基于通用 UDP 运行，所以只要是在以 IP 构建的物理网络上，所有 IP 可达的边缘设备（VTEP，可以是交换机或服务器）都有条件构建一个大范围二层网络。这种实现，屏蔽了物理网络的模型与拓扑差异，将物理网络的技术实现与计算虚拟化的关键要求分离开来，几乎可以支持以太网在任意网络上的透传，使得云计算资源调度范围急速扩大。

VXLAN 在 IP 网络的每个端点上都有一个 VTEP 负责 VXLAN 协议报文的封包和解包，也就是在原始数据报文上封装 VTEP 通信的报文头部。物理网络上可以创建多个 VXLAN 网络，这些 VXLAN 网络可以认为是多个隔离的隧道，不同节点的虚拟机能够通过隧道直连。每个 VXLAN 网络由唯一的 VNI 标识，不同的 VXLAN 相互隔离。

2. 网络虚拟化功能

除了通过 Overlay 技术解决用户的网络隔离需求外，在云计算场景下，用户会按需购买、使用和释放 L4～L7 层网络服务，如与互联网通信时用到的负载均衡、公网 IP 地址映射等功能，这些网络服务具有生命周期动态变化和服务性能弹性伸缩等特点。在此背景下，一是要解决不同租户网络资源的隔离；二是要做到快速创建、配置、扩容和回收这些网络资源。

负载均衡、VPN 和 NAT 等这些 L4～L7 层网络服务在传统数据中心都依赖于专有硬件设备去提供，但基于硬件的网络功能设备存在灵活性差、扩展能力差、统一管理困难等问题。NFV 是一种利用虚拟化技术实现网络功能的方式，在标准服务器上提供以往在专用硬

件中实现的网络功能，通过将网络功能和物理设备解耦，使网络功能不受硬件设备限制，部署方式更灵活。对于 NFV 产品的研发，在业界也是热点，传统厂商、运营商和云计算厂商都在投入资源研发 NFV，NFV 的概念很广，在这里不做展开介绍，对云计算场景而言，最为重要的是将租户最常用的 NAT、ELB、VPN 等功能通过软件化的方式实现，并按需分配给租户使用。

有了虚拟网络的 Overlay 和 NFV 之后，还需要对网络进行统一的编排和控制。通过 API 自动化编排，允许网络根据需要进行扩展，使网络服务能够在多个设备上进行配置，并且可以根据需要部署资源，从而使网络的部署更加便捷和快速，让用户获得更好的使用体验。网络的自动化编排主要涉及两方面内容：一是网络资源的生命周期管理，包括创建、删除、修改、扩容等一系列动作；二是对转发的指导，确保数据报文可以按网络编排的需要进行转发。

3.4.2 软件定义网络与网络虚拟化

传统的网络建设方式，需要工程师手动部署所有网络设备的配置，这样不但效率低下，而且对人力的消耗也很大。同时，每个设备都独立完成网络控制，从源地址到目的地址的转发行为由每个网络节点独自决定。在云数据中心场景中，为了满足服务的快速部署和动态调整，传统的网络建设方式已完全不能适应业务的发展，必须实现对网络资源的自动化交付。对数据报文的转发也应该按云计算业务的需要进行表项和策略的下发。

2006 年，SDN(Software Defined Network，软件定义网络)诞生于美国的斯坦福大学 Clean Slate 课题，其核心理念是对传统网络的控制和转发层面进行改造，让其控制与转发分离，将软件应用到网络控制当中。

2011 年，一个以网络用户为主导的非营利性组织 ONF(Open Networking Foundation)诞生了。ONF 是致力于推进 SDN 标准化的一个用户驱动的组织，工作重点是制定南向接口标准协议 OpenFlow，制定硬件行为转发标准。

2013 年以设备商和软件商为主导的 SDN 组织 ODL(Open Day Light)腾空出世。ODL 是由 Linux 基金会推出的一个开源项目，集聚了行业中领先的供应商和 Linux 基金会的一些成员。ODL 的主要目的在于推出一个通用的 SDN 控制平台，在这个平台上进行 SDN 普及与创新，供开发者基于其构建商业产品，或者贡献代码。至此，SDN 进入一个崭新的时期，逐渐应用到商业产品当中。

SDN 将网络划分为数据转发层和控制层，控制层与数据转发层分离：控制层更灵活，数据转发层更标准化。SDN 的出现为网络虚拟化服务部署提供了新的解决方案。通过集中控制，网络管理员可以通过控制器的 API 编写程序，实现自动化服务部署，大大缩短业务部署周期，并实现按需动态调整。SDN 的体系架构如图 3.6 所示。

SDN 的出现使网络虚拟化更加灵活、高效，网络虚拟化已成为 SDN 的重量级应用。当前的 SDN 网络架构通常采用控制器集群的形式，收集整个网络的拓扑和流量信息，计算流量转发路径，并将转发表项发送给交换机(包括硬件交换机和 vSwitch)。交换机根据下发

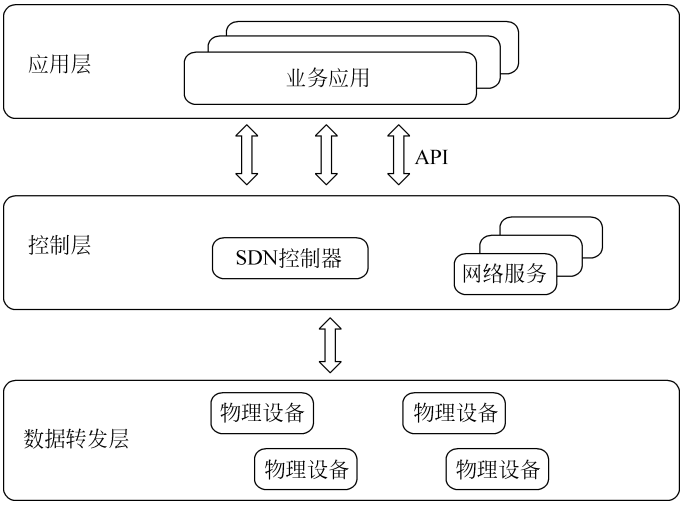

图 3.6　SDN 的体系架构

的转发表里的条目执行转发操作。控制层从传统网络的单个设备剥离并集中在控制器上。转发层由交换机组成。

与传统网络相比，SDN 网络具有许多优点，例如控制和转发的分离。这一设定打破了传统设备的约束，提高了新服务的部署速度，并优化了整个网络层的流量。在 SDN 网络中，开发人员和用户都可以更多地发挥想象力，而不受各种传统网络协议的限制。

狭义的 SDN 主要是转控分离，通过 SDN 控制器集中进行转发表项的下发；广义的 SDN 更强调网络的可编排性，通过软件的方式实现网络业务的自动化开通，并根据业务需要进行网络调整。在云计算场景下，SDN 对网络的编排，主要是以下两方面。

（1）**租户网络的虚拟化**：通过 Overlay 技术将物理网络抽象出来，划分出多个隔离的虚拟网络提供给租户，让不同的租户独立管理自己的网络环境。而这里的网络虚拟化，都通过 SDN 控制器对网络设备进行统一控制来实现。租户可以根据自己的业务需求，灵活编排自己的网络。

（2）**网络功能的虚拟化**：通过标准服务器将网络功能虚拟化以后，NFV 网元具备了提供网络服务的能力。但是，要满足租户使用云计算网络服务的要求，还需要对 NFV 网元进行统一编排。按照云计算的业务需求，对 NFV 网元实现生命周期管理，可以按需动态创建、删除和扩容 NFV 资源，并且根据业务的流量模型，下发对应的转发表项或转发规则，确保业务报文被正确转发。

3.4.3　虚拟网络的安全隔离技术

云环境下的虚拟网络是构建在物理网络上的逻辑网络，如何实现虚拟网络的安全隔离，可以从以下三方面展开来看。

(1) 网络资源的安全隔离：网络资源虚拟化只是完成从物理资源到虚拟资源的抽象过程，资源本身还是混在一起的。为了实现真正意义上的网络虚拟化，还需要有租户的概念，根据租户来分配隔离的网络资源，以便不同租户的计算和存储等资源不受其他租户的影响。

(2) 虚拟网络内部的安全隔离：虚拟化带来的最大威胁是虚拟机之间如何隔离。云计算环境通过部署分布式防火墙(例如，iptables)，实现在同一虚拟网络内部(同一台物理服务器或者不同的物理服务器)不同虚拟机之间的安全隔离，避免虚拟机被非法访问或被恶意攻击，虚拟机之间的访问规则可以由租户自己定义。

(3) 网络地址映射：出于安全的考虑，并不会将云主机的地址直接暴露在互联网，而是通过地址映射的方式，实现对云主机的保护。租户可以购买提供互联网服务的 NAT 网关或负载均衡产品，当租户的虚拟机需要与互联网通信时，可以将云主机的 IP 转化成公网 IP。不同租户的虚拟网络是相互隔离的，而与互联网通信时，公网 IP 地址映射确保租户在创建云主机时可以使用自定义的 IP 地址，即便不同租户的 IP 地址发生冲突，也不会对业务产生影响。

3.4.4 平安云的虚拟专有网络

1. VPC

平安云的虚拟专有网络(Virtual Private Cloud，VPC)是基于平安云构建的一个隔离的网络环境，是用户在云上的私有网络，可以由用户自定义，不同 VPC 在逻辑上相互隔离。用户能够在自己定义的虚拟网络中使用平安云资源，例如，在 VPC 中创建云主机、存储和数据库等资源，并通过安全组实现 VPC 内的访问控制，通过 NAT 网关、ELB 等产品连接公网。平安云 VPC 的架构如图 3.7 所示。

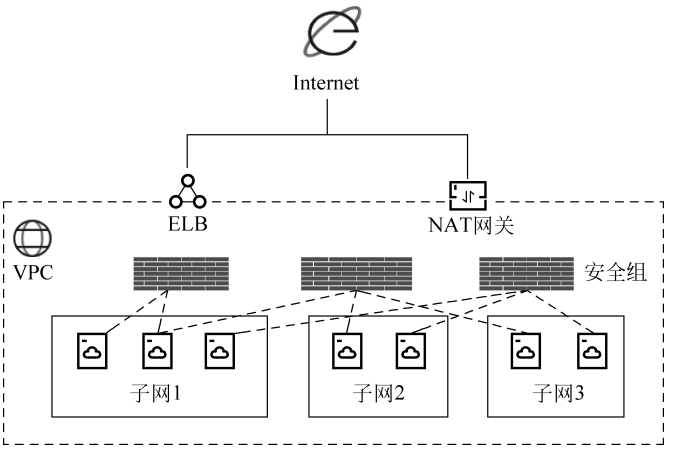

图 3.7 平安云 VPC 的架构

对于企业用户而言，VPC 主要起以下作用。

(1) 资源隔离：VPC 提供隔离的计算和网络环境，满足不同部门/业务的网络隔离需求；

(2) 增值业务接入：每个 VPC 可以提供独立的 NAT 网关、弹性 IP、安全组、VPN、ELB 等增值业务。

VPC 是地域级的产品，而一个地域内可以有多个可用区(AZ)，所以 VPC 是可以覆盖到不同 AZ 的。VPC 中的子网是 AZ 级的产品，只在一个 AZ 内有效。同一个 VPC 的子网可以分布在不同的 AZ 来实现业务在网络层面的高可用。

平安云的用户可以根据业务需求创建多个 VPC，多个 VPC 在网络层通过隧道进行逻辑隔离。而不同用户的 VPC 之间如果要进行通信，除了需要打通底层网络的隧道隔离，还需要在 API 层进行用户账号之间的授权，为用户的云上资源提供双层保护。

一个 VPC 内的 ECS 云主机之间的传输数据包都会加上隧道封装，带有唯一的隧道标签进行标识，然后送到传统物理网络上进行传输。不同 VPC 内的 ECS 云主机因为所在的隧道标签不同，本身处于两个不同的逻辑网络，所以不同 VPC 内的 ECS 实例无法进行通信，天然地进行了隔离。

平安云最早沿用金融数据中心的标准，将 VPC 划分为 DMZ、SF、PTR 三个默认隔离的网络域，以方便进行业务部署。用户可以在网络域下创建子网，并将虚拟机部署在子网里。在逐步发展演进的过程中，也为了更好地服务平安云的用户，对于 VPC 内的隔离已经交由用户自主规划，可通过安全组来自定义 VPC 内的隔离区域。

另外，平安云通过 NFV 软件化了 L4～L7 层网络功能，提供了负载均衡 ELB、VPN 网关、NAT 网关等服务。通过 NFV，实现了网元纯软件化和硬件设备解耦，一方面可以采用通用硬件降低成本；另一方面可以快速部署业务，并且实现网络资源的智能调度，满足自动扩容和节能减排的需求。

2. 安全组

针对 VPC 内的云主机访问控制的需求，平安云提供了安全组进行云主机之间的安全隔离。安全组从底层技术实现来看是一种分布式的防火墙，可以对云主机的出入流量进行访问控制，帮助用户在云上的网络环境中划分隔离区域。

安全组中可以定义各种访问规则，当云主机加入该安全组后，即受到这些访问规则的保护。可以说安全组其实是一组云主机的集合，当集合有与其他隔离区域互通的需求时，就需要设置对应的安全组规则。安全组会记录有关进出云主机实例的流量的信息。将基于流量的连接状态以确定允许还是拒绝流量。这使安全组可以是有状态的。有状态是一种记忆功能，即一个包允许入站就允许出站，允许出站就允许入站。

用户可以根据虚拟机的属性灵活定义安全组，包括授权策略、出入方向、协议、授权地址等元数据。安全组是具有同等安全要求的一组虚拟机。安全组的策略可以细化到每台云主机。每个安全组对应一个安全隔离域，可以基于安全组定义自己的访问控制规则，实现域内

和域外的安全隔离。用户可以将具有相同属性的虚拟机绑定到同一个安全组，实现 VPC 内的业务划分。如果一台虚拟机具有多重属性，可以通过绑定到不同的安全组来满足业务需求。

3.5 NAT 网关

3.5.1 NAT 网关技术原理

对于 NAT(Network Address Translation，网络地址转换)这个术语大家应该都不陌生，这一神奇技术的诞生几乎已经让我们忘记了 IPv4 的地址空间已经耗尽的这个事实。2011 年 2 月 3 日，互联网数字分配机构(the Internet Assigned Numbers Authority，IANA)将 IPv4 地址的最后 5 个地址块分配给了其下的 5 个地区委员会，至此 IPv4 地址已经耗尽。虽然 IPv4 地址最多包含 43 亿个地址，但是为了路由和管理方便，IPv4 地址按照不同的前缀长度被划分成了 A、B、C、D 四类地址，以便分给不同规模的组织。这样的规划导致 IPv4 地址的消耗非常迅速，全球的网络专家都在寻找解决方案来减缓或者延迟 IPv4 地址的消耗。

在这样的大环境下，NAT 技术应运而生，并给后来的网络发展，甚至于当下的网络发展带来了不可替代的、深远的影响。NAT 技术针对 IP 报文头部中的地址信息进行转换。NAT 网关通常部署在一个企业或组织的公网出口处，通过将内部 IP 地址转换为公网 IP 地址来使该企业或组织具备访问公网的能力。当报文离开企业或组织内部网络进入公网时，NAT 网关将报文的源 IP 地址转换为公网 IP 地址(通常是 NAT 网关的接口地址)，该请求包到达目的主机后，表现为该企业或组织的 NAT 网关对目的主机的访问，所以目的主机将响应包发送回 NAT 网关，NAT 网关再将目的地址转换为内部 IP 地址，并发回内部网络，这样一次私网主机和公网主机之间的通信就完成了，如图 3.8 所示。

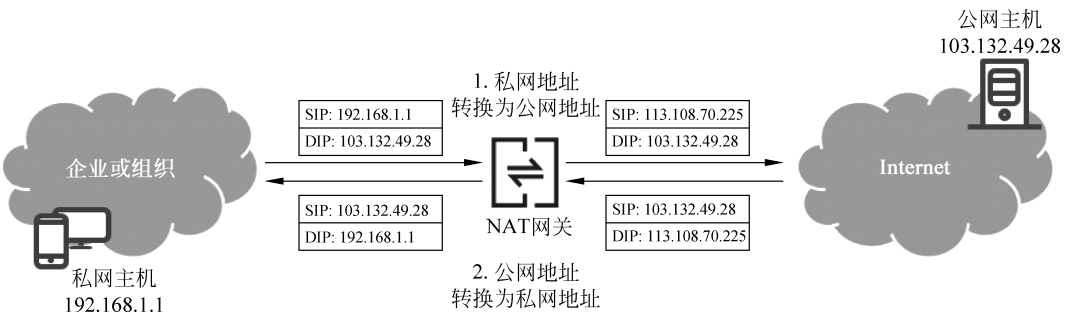

图 3.8 NAT 转换过程示意图

从上面的通信过程描述中可以总结出 NAT 网关的几个关键特点。

(1) 网络被划分为私网和公网，企业或组织内部采用私网地址，NAT 网关位于私网到公网的出口位置，进出流量都需要通过 NAT 网关。

(2) NAT 网关在进出方向都需要做地址转换：出方向做源地址转换，进方向做目的地址转换。

(3) NAT 网关对于通信双方来说是透明的，通信双方感知不到 NAT 网关的存在。

(4) NAT 网关需要维护一张会话表，用于保存会话信息，完成双向转换的功能。

NAT 有三种类型：静态 NAT、动态 NAT 和 NAPT(Network Address Port Translation，网络地址端口转换)。

静态 NAT 是最简单、最容易实现的一种 NAT，私网中的每个主机都被映射成公网中的一个 IP 地址。当私网内的每一个主机都必须有一个固定的公网 IP 地址时，可以通过静态 NAT 的方式实现，但是静态 NAT 的方式对于节省 IPv4 没有任何作用。

动态 NAT 是在公网中申请了一组 IP 地址(地址池)，采用动态分配的方式映射到私网中的主机。当私网中有一台主机需要访问公网时，NAT 网关从公网 IP 地址池中选择一个可用的地址分配给该主机使用，当通信完成或者通信超时后，该公网 IP 地址被释放回地址池。

不管是静态 NAT 还是动态 NAT，都是实现私网 IP 地址和公网 IP 地址之间的转换，也就意味着私网中同时和公网进行通信的主机数量受到 NAT 网关中公网 IP 地址数量的限制。为了克服这种限制，就需要使用 NAPT。NAPT 将 IP 地址转换扩展到端口的转换，把私网地址和端口映射到公网地址和端口，多个私网地址可以映射到同一个公网地址的不同端口，也就是进行“私网地址＋私网地址端口”与“公网地址＋公网地址端口”之间的转换，使得多台私网主机可以利用同一个公网 IP 地址和公网进行通信。

也就是说，静态 NAT 和动态 NAT 都是一对一的 NAT 转换，而 NAPT 则是一对多的 NAT 转换。在实际应用场景中，静态 NAT 和 NAPT 被广泛应用，静态 NAT 用于满足一些特殊的网络架构需求，NAPT 用于满足节约上网费用、简易防火墙等需求。

3.5.2　NAT 网关应用场景

通过 3.5.1 节的简单介绍，NAT 网关的功能和概念已经比较清晰了。鉴于云上的客户一样有 IPv4 地址消耗的问题和确保私网地址安全性的需求，所以云上也提供了 NAT 网关产品。目前，云上 NAT 网关支持以下几种公网访问模式。

(1) SNAT(源地址转换)：仅提供私网云主机主动访问公网的能力，即只能由云主机实例主动发起公网访问的请求后，公网的服务器才能回复该请求，而不具有面向公网提供服务的能力。

(2) DNAT(目的地址转换)的 IP 地址映射：可以同时提供主动访问公网的能力和被动访问公网的能力，既可以由云主机实例主动发起公网访问的请求，也可以由公网客户端主动发起访问请求，但是该方式无法满足节约公网 IP 地址的需求。

(3) DNAT 的端口映射：仅提供被动访问公网的能力，即只能由公网的客户端主动发起访问请求后，云主机实例才能访问公网并回复该请求，不具备主动发起公网访问的能力。

下面介绍 NAT 网关的主要应用场景。

(1) 企业或组织内的多台主机共享公网 IP 地址访问互联网。

内部网络往往存在一些服务器需要访问公网的情况,包括软件升级、网页浏览等,出于节约成本的考虑,需要多台服务器共享一个公网 IP 地址,可以使用 NAT 网关的 SNAT 功能来实现。

(2) 企业或组织对私网提供简单的安全防护功能。

通过 NAT 网关的 SNAT 功能可以实现私网的安全防护功能。建立 NAT 网关可以针对公网服务器的访问请求进行控制,相当于把私网服务器隐藏到 NAT 网关后面,不会被黑客扫描到,也不会轻易地被黑客攻击。公网服务器不能主动对私网服务器发起访问请求,只有接收到私网服务器发送的访问请求的公网服务器才能和私网服务器进行通信,而且需要匹配通信的五元组信息。

(3) 搭建简单的网站或 App 服务。

NAT 网关的 DNAT 功能可以实现 1～2 台服务器对公网提供网页服务或 App 服务。当服务器只需要占用少量的固定端口用于提供服务时,可以使用 NAT 网关的 DNAT 端口映射功能来实现,最大限度地利用单个公网 IP 地址;当服务需要的端口不固定或者需要大量的端口时,可以使用 NAT 网关的 DNAT IP 地址映射功能来实现,该功能其实就是静态 NAT 功能,实现私网地址和公网地址的一对一映射。

3.5.3 平安云的 NAT 网关

平安云的 NAT 网关产品是基于自主研发的 PAGW 网关系统来实现的。PAGW 网关系统具有高可靠性、可扩展性、高性能等特点,是平安云自主研发的集群式网关系统,也是平安云网络产品的核心组件,基于该系统已逐步支持了多种网络产品的功能。

平安云 NAT 网关的架构图如图 3.9 所示。平安云 NAT 网关由以下几部分组成。

(1) NAT 网关实例:一个运行的 NAT 网关服务,为所属的 VPC 内的云主机提供 NAT 服务,需要配合 NAT 规则和带宽包才能实现访问互联网的业务。

(2) NAT 规则:用户定义的 SNAT 规则和 DNAT IP 映射及端口映射规则。客户根据业务场景的不同,可以进行各种 NAT 规则的单独使用和组合使用。

(3) 带宽包和公网 IP:带宽包是用户购买的公网带宽资源,可以指明共享该资源的公网 IP 数目。

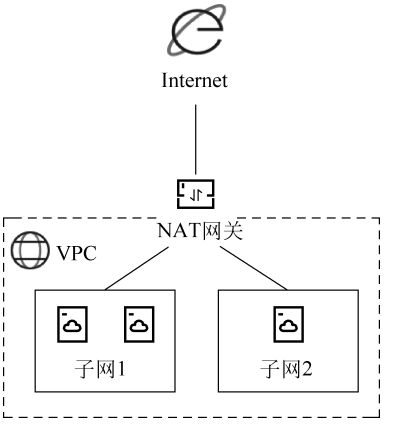

图 3.9 NAT 网关的架构图

平安云 NAT 网关提供 SNAT、DNAT 和带宽包功能。

(1) SNAT 功能为 VPC 内的云主机实例提供访问互联网的能力,避免私网的后端服务

器暴露于公网，只有主动和外部服务器建立连接的后端服务器才可以被外部服务器访问。SNAT 的场景图如图 3.10 所示。

（2）DNAT 功能将 NAT 网关中申请的公网 IP 地址映射给云主机实例使用，使云主机实例能够提供互联网服务或者能够访问互联网。DNAT 支持 IP 映射和端口映射两种模式。图 3.11 为 DNAT IP 映射的场景图。

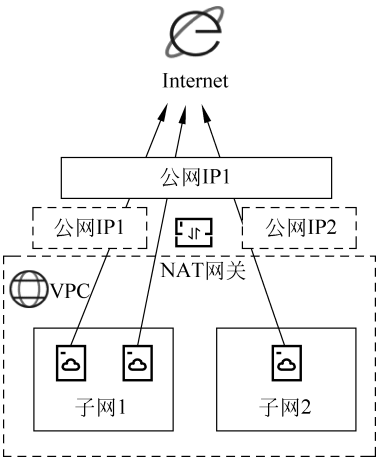

图 3.10　SNAT 的场景图

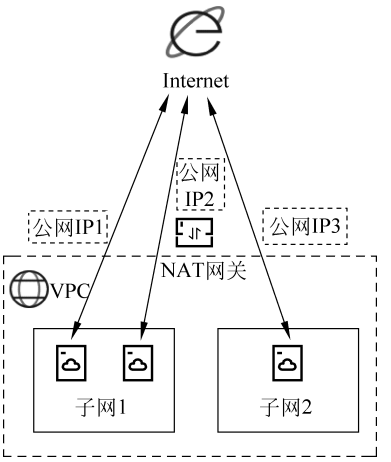

图 3.11　DNAT IP 映射的场景图

（3）带宽包功能能够实现带宽的共享和复用。当前每个带宽包最多支持 5 个公网 IP。

3.6　负载均衡技术

负载均衡(Load Balance)技术是一项重要的网络均衡转发技术，它构建于网络基础之上，提供透明、无感知、低成本、高性能的均衡分发服务。负载均衡技术从整体上提升了网络设备、服务器设备的响应能力，增强了网络对数据的处理能力，同时提升具体应用的对外服务能力、应用的吞吐量和应用的稳定性。除此以外，在云环境中，负载均衡产品还能够加强网络安全，为应用带来更多的安全控制能力。

但也不能简单地通过以上定义固化负载均衡技术的含义，因为随着传统网络向云平台的演进，发现了更多负载均衡技术所带来的好处。下面通过两种情况来说明负载均衡技术的典型优势。

（1）应用服务要响应海量的并发请求，接纳大量的数据流量，经过负载均衡技术的处理后，不仅可以实现应用高可用的需求，同时减少了请求等待响应的时间。

（2）负载均衡改善了单台服务器重负载的现象，在运算请求负荷很大的情况下，负载均衡技术实现了计算的并行处理，系统处理能力大大提升。

总体来讲，负载均衡技术的优点在于它可以根据应用流量进行判断，自动完成请求的分发，同时通过弹性伸缩资源策略，来应对流量变化对计算能力的需求。

3.6.1 负载均衡技术原理

负载均衡技术的原理是一个值得探讨和讲述的话题，因为客户端发起的每一次请求传递到服务器的过程中，有很多环节可以实现负载均衡，各个云厂商在架构自己的负载均衡时，实现方式不同，所以负载均衡技术的原理是灵活多变的。

1. 负载均衡分类

根据架设负载均衡的环节的不同，负载均衡大致分为以下几类。

1）数据链路层负载均衡

数据链路层负载均衡使用虚拟 MAC 地址的方式提供服务，请求数据包中的二层目的地址封装的是对外服务的虚拟 MAC 地址，负载均衡接收请求后，会将请求分配到具有实际 MAC 地址的设备上进行响应。

在这个过程中，负载均衡利用修改目的 MAC 地址的方式进行请求转发，而不修改 IP，因为组建这样的负载均衡模型时，真实物理服务器集群的 IP 和负载均衡服务 IP 是一致的。这种方式的好处是不需要修改数据包的源 IP 和目的 IP 就可以实现数据分发。

如图 3.12 所示，服务器的服务 IP 地址和数据请求的目的 IP 一致时，负载均衡不需要转换 IP 地址，可将应答的数据包直接返回给客户端。如果公网 IP 地址充足，甚至可以为每一台服务器都配置自己的公网 IP，服务器便可以使用自己的公网 IP 进行请求的应答，在一定程度上，巧妙地规避负载均衡服务器本身的网卡带宽成为系统瓶颈的问题。不过这种情况下，要为每一台服务器都绑定负载均衡的服务 IP，才能确保服务器收到请求报文。这种三角传输的负载均衡方式也称为直接路由（DR）方式。这种方式被非常广泛地用于应用业务中以获取最好的工作性能，只是配置起来略显复杂。

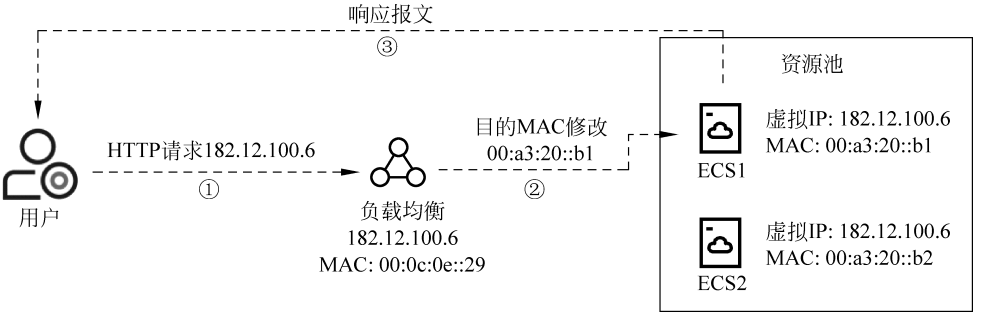

图 3.12 数据链路层负载均衡模型

2）网络层负载均衡

网络层负载均衡（三层负载均衡）采用虚拟 IP 地址的方式提供服务。请求数据包中的 IP 地址为负载均衡服务 IP，并且请求将通过实际互联 IP 地址分配到后端实际的服务器上进行响应。在这一原理中，请求数据包将被负载均衡服务器的内核进行处理，通过负载均衡

算法得到一个目标真实服务器的地址，并修改请求数据包中的目的 IP 为真实服务器 IP，来完成请求的分发。而真实服务器处理请求后，会将响应返回给负载均衡服务器，在负载均衡服务器上再次完成源 IP 的修改，用负载均衡自身的服务 IP 对客户端进行回应。

如图 3.13 所示，负载均衡在修改目的 IP 地址的同时修改了响应的源 IP 地址，所以又被称为 SNAT 方式的负载均衡。所有的请求报文和响应报文都需要经过负载均衡服务器的处理，集群的吞吐能力将受到服务器网卡带宽的限制。

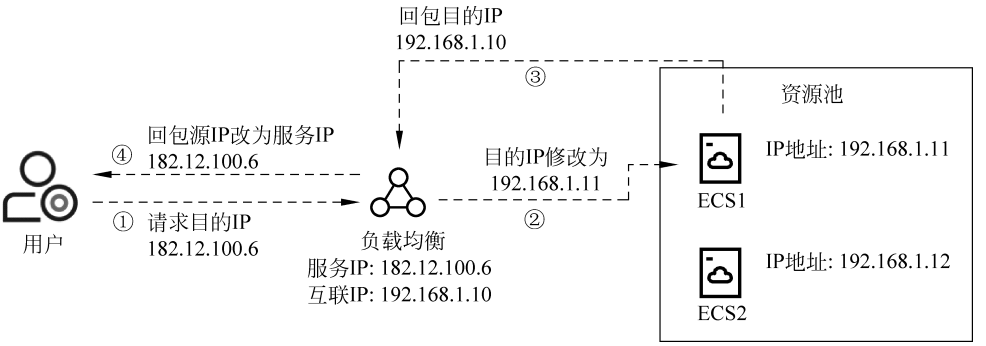

图 3.13　网络层负载均衡模型

有三层的负载均衡，就有四层的负载均衡，四层负载均衡在图 3.13 所示的技术特点的基础之上，增加了四层协议 TCP/UDP 的端口，根据 IP＋端口来调度请求，分发到后端的服务器上进行响应。

3）反向代理负载均衡

利用反向代理实现负载均衡已经成为很多云平台厂商常用的方法。反向代理位于服务提供侧，部署在服务器的前端，替服务接收 HTTP 请求。服务器也不直接暴露于公网提供服务，因此并不需要部署外网 IP，所以反向代理还有保护网站安全的作用。

如图 3.14 所示，反向代理服务接收到 HTTP 的访问请求后，按照负载均衡算法将请求分配到服务器端获取响应，当接收到响应报文后对外提供代理应答，所以也同时充当了负载均衡器的作用。反向代理服务器需要拥有两块网卡，分别用于配置内部互联 IP 和外部服务 IP。

反向代理得到服务器的响应内容后，同时可以缓存到本地，当内容再次被请求时，代理可以通过缓存内容对用户进行响应，很大程度上降低了服务器的负载。

由于反向代理负载均衡大多用于应用层请求，可以针对域名、URL 甚至是主机名接收的请求做负载均衡，所以也被称为应用层负载均衡或 HTTP 负载均衡。

反向代理负载均衡集成了反向代理和负载均衡两种功能，所以部署相对简单，但反向代理是所有应用层请求的中转站，其性能会有一定的瓶颈，在架构和部署时要充分考虑扩容和性能提升的问题。

4）DNS 负载均衡

DNS 负载均衡是利用处理 DNS 域名解析请求的同时进行负载均衡。可以在 DNS 服

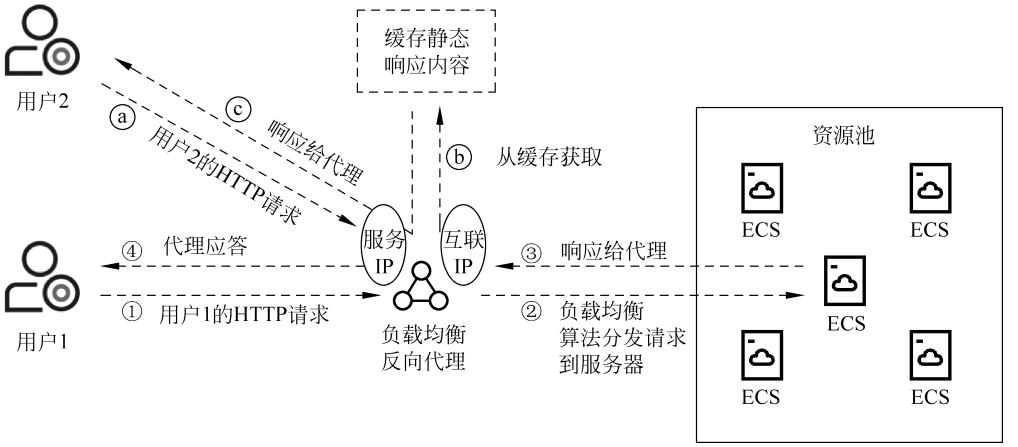

图 3.14　反向代理负载均衡模型

务中为同一个域名配置多条 A 记录，使得每一次的域名解析请求根据负载均衡算法分配到不同的服务 IP。

如图 3.15 所示，当 yun.pingan.com 存在 3 条不同的 A 记录时，用户的每一次解析请求可以获得不同的结果，从而使不同用户的请求分配到不同的服务器上，这就是 DNS 实现负载均衡的方式。

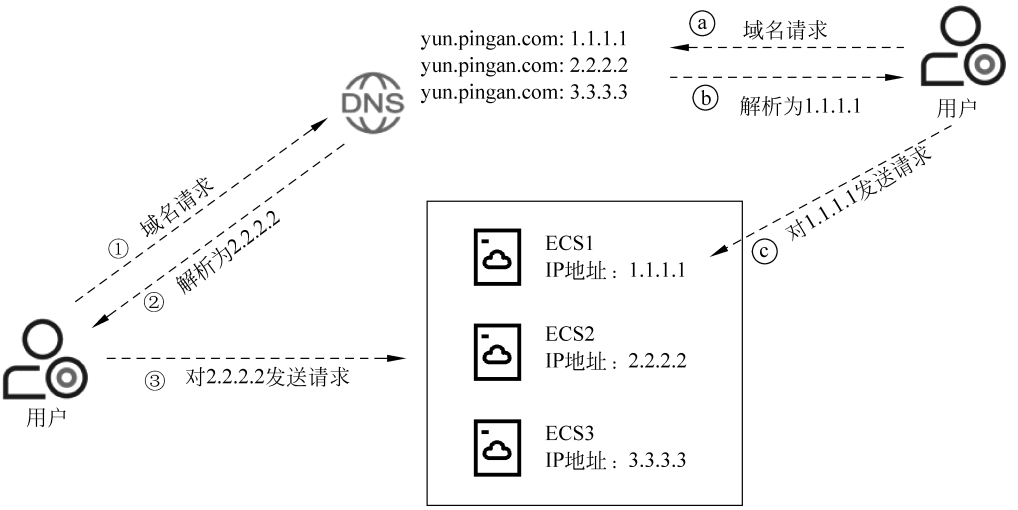

图 3.15　DNS 负载均衡模型

在 DNS 环节部署负载均衡的好处是可以将负载均衡的工作交给 DNS 来完成，规避了服务侧网络功能部署的复杂性，但由于 DNS 对 A 记录具有缓存性，受网站服务侧变更控制有较长的延迟，不容易在第一时间发生连动。

事实上，很多大型线上服务的应用都将 DNS 作为负载均衡的第一级手段，然后再在其

他环节部署负载均衡。

DNS负载均衡使用起来非常简单，并且DNS的域名解析性能很高，反应速度快，如果可以解析成距离用户最近的服务位置，还可以提升用户的访问速度。但DNS负载均衡也有一些不足，如：由运营商控制的DNS多级服务新增和修改起来有延迟，而且不方便对DNS做更多的改善和扩展，只能寄希望于域名提供商的维护；同时DNS离真实的服务端较远，不能区分后端真实设备性能的好坏。了解了这些优点和不足之处便可以更灵活地将DNS负载均衡运用于网络服务中了。

5）HTTP重定向负载均衡

HTTP重定向负载均衡中需要有一个重定向服务器来完成HTTP请求的重定向任务，通过重定向建立连接的目的地址来完成负载均衡。但是这一过程会使客户端的浏览器发起两次链接请求才能正常访问，并且即使是重定向的过程，对于重定向服务器而言也是一次标准的HTTP请求响应过程，重定向服务器会回复302响应状态码给浏览器。

如图3.16所示，假设重定向服务器后端有4台服务器，那么重定向服务器的流量将是后端集群请求的4倍。如果是100台服务器的集群，结果就是100倍。

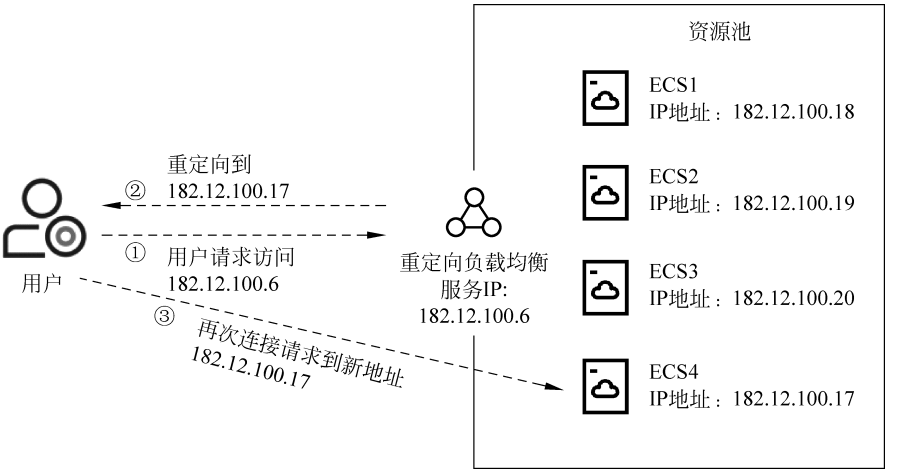

图3.16　HTTP重定向负载均衡模型

除了以上介绍的5种类型的负载均衡外，还有很多其他方式的负载均衡。在云平台的构建中，一般不仅仅使用上述的一种负载均衡方式，而是根据特定的场景和需求来判断使用，并且会组合运用多种负载均衡方式来提升服务的能力。

2. 负载均衡的主流技术

下面介绍负载均衡的几种主流技术。

1）四层负载均衡技术

四层负载均衡是基于IP＋端口的负载均衡。常见的四层负载均衡组件包括LVS、

Nginx 和 HAProxy 等。LVS 四层负载均衡是非常常见的负载均衡技术，具有三种工作模式：NAT 模式、TUN 模式和 DR 模式，并且支持 10 种负载均衡调度算法，下文中将详细介绍。Nginx 是一种反向代理服务器，同时具有缓存等高级功能，也支持轮询、权重、IP 哈希算法以及其他第三方转发策略的方式来实现负载均衡。HAProxy 是一款支持四层高性能负载均衡的开源代码，自带强大的监控服务器状态页面。

2）七层负载均衡技术

七层负载均衡是基于 URL 等应用层信息的负载均衡。常见的支持七层负载均衡的组件包括 HAProxy、Nginx、Apache 和 MySQL Proxy 等。HAProxy 支持七层负载均衡，并提供全面七层代理和会话保持功能，HAProxy 作为 MySQL、邮件等应用的负载均衡也具有非常突出的性能。Nginx 实现的七层负载均衡在 HTTP 和 HTTPS 上可以发挥出很好的能力。Apache 和 MySQL Proxy 也可以实现七层负载均衡，但功能一般，这里不多作介绍。

3）负载均衡算法

每一个负载均衡服务器上都有一个调度器（LB），依靠它将客户请求按照一定的规则分发到后端服务器，这些规则是预先定义好的调度算法。常见的调度算法有：

（1）轮询（Round Robin）。

这是最简单的调度算法，LB 按照顺序将请求一次性转发给后端的真实服务器（RS），并不参考后端 RS 的处理能力和响应速度。但是在大多数的真实环境中，RS 的状态各不相同，这种算法无法满足合理利用资源的要求。

（2）加权轮询（Weighted Round Robin）。

带权重的轮询是在轮询算法的基础上为后端 RS 赋予权重值，权重值越高的 RS 被分配的请求越多，这种方式适合按照服务器性能的高低赋予不同的权重值，从而达到合理的资源利用。

（3）最少连接（Least Connections）。

该算法会将请求分发到连接数最少的 RS 上，连接数越少，说明 RS 相对空闲，从而实现均衡的负载。

（4）加权最少连接（Weighted Least Connections）。

该算法在采用最少连接数的同时为 RS 分配权重，目的是在资源实际使用的基础上，达到人为控制连接请求的分配。

（5）基于局部性的最少连接（Locality-Based Least Connections，LBLC）。

这是一种基于报文目的 IP 的负载均衡调度方式，常用于 Cache 集群系统。Cache 集群中客户请求报文的目的 IP 地址是变化的，目的是将相同目的 IP 的请求调度到同一台 RS 上响应，以便提高每台 RS 的访问局部性和主存 Cache 的命中率，从而提升整个集群的处理能力。该算法根据请求的目的 IP 地址首先找出该目的 IP 地址最近使用的服务器，若该服务器是可用的且没有超载，会将请求发送到该服务器；若服务器不存在，或者该服务器超载且有其他服务器处于较低的工作负载状态，则用“最少连接”原则选出一个可用的服务器，将请求发送到该服务器。

(6) 带复制的基于局部性最少连接(Locality-Based Least Connections with Replication, LBLCR)。

与 LBLC 场景相同,常用于 Cache 集群系统。但是 LBLC 是维护一个目的 IP 到一台服务器的映射,而 LBLCR 算法是用于维护一个目的 IP 到一组服务器的映射。即,根据请求的目标 IP 找出对应的服务器组,按"最少连接"原则从服务器组中选出一台服务器来响应,若服务器没有超载,将请求发送到该服务器;若服务器超载,则按"最小连接"原则从这个 Cache 集群中选出一台服务器,将该服务器加入到服务器组中,再将请求发送到该服务器。

(7) 目标地址散列(Destination Hashing,DH)。

该算法是一种静态映射算法,针对请求报文的目的 IP,通过一个哈希函数将一个目的 IP 映射到一台服务器。DH 算法将请求的目的 IP 作为 Hash Key 并从静态分配的散列表中找出对应的服务器。如果该服务器是可用的并且没有超载,则将请求发送给该服务器,否则返回空。

(8) 源地址散列(Source Hashing,SH)。

SH 算法与 DH 算法相反,它根据请求中的源 IP 地址作为 Hash Key 并从静态分配的散列表中找出对应的服务器。如果该服务器是可用的并且没有超载,则将请求发送给该服务器,否则返回空。

(9) Fair 调度算法。

该算法按照后端服务器的响应时间来分配请求,响应时间越短越优先分配。

(10) URL 哈希算法。

该算法按访问 URL 的 Hash 结果来分配请求,使每个 URL 定向到一个后端服务器,后端服务器有缓存功能时,可以有效地提升处理能力。

4) 健康检查技术

负载均衡的健康检查是用来判断后端服务器的业务可用性的,也是负载均衡提升业务整体可用性的一种机制。负载均衡开启健康检查功能后,通过健康检查规则判断后端服务器的异常状态。当服务器异常时,负载均衡会自动将新的请求发送给其他健康的服务器进行处理;而当异常服务器恢复正常时,负载均衡又会让该服务器正常参与请求的处理。这种机制主要是为了保证业务层面的可用。健康检查的种类有如下几种:

(1) TCP 健康检查机制。

TCP 健康检查针对四层 TCP 监听,是一种较为高效的检查机制。它通过使用 TCP 建立连接的方式探测后端服务器的健康状况,即向后端 RS 的互联 IP+端口发起 TCP SYN 报文,并得到后端 RS 的响应报文 SYN+ACK。如果后端 RS 在预定义的响应超时时间内完成回应,则认为服务器正常;如果没在响应超时时间内收到应答,则判定为异常状态。

(2) UDP 健康检查机制。

该方式是通过 UDP 报文向后端服务器 IP+端口发起探测,如果后端 RS 的端口未正常监听,则系统会返回 port xx unreachable 的 ICMP 报错信息,反之正常情况不会做任何响应处理。如果在预先定义的响应超时时间内收到了后端 RS 的上述回应,则认为服务器异常,如果没在响应超时时间内收到应答,则判定为正常状态。

（3）HTTP 健康检查机制。

HTTP 健康检查机制针对七层 HTTP/HTTPS 协议监听生效。HTTP 健康检查通过向后端“服务器 IP＋健康检查端口＋检查路径”发送 HTTP HEAD 请求进行探测，后端服务器在收到请求后，会根据相应的业务运行情况返回 HTTP 状态码，例如 2xx、3xx、4xx、5xx 等。如果在预先定义的响应超时时间内收到了后端服务返回的状态码，则与预定义的健康状态码比对，判定业务的健康状态；如果在响应超时时间内，没有收到任何回应，则判断业务不可达，处于异常状态。

3.6.2 负载均衡应用场景

为什么要使用负载均衡呢？在原始的场景中，当应用面临大量的用户访问时，这个应用系统的负载是很高的，并且负载分配不均匀。最初会通过增加服务器数量的方式横向扩展，以便通过提升集群整体能力的方式来提升处理能力。后来又将相同的应用部署在不同的高性能集群上，为了解决统一出口和访问调度的问题，在这种横向扩展的基础上，采用负载均衡的调度方式，就有了图 3.17 所示的经典的负载均衡模型。

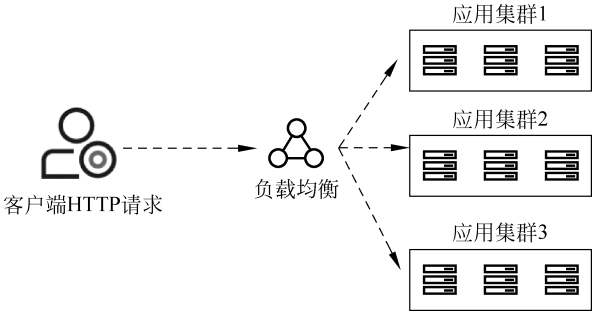

图 3.17　经典的负载均衡模型

- 应用集群上部署的是相同的应用，组成高可用处理集群，用于接收负载均衡设备分发的请求；
- 负载均衡服务器主要提供负载均衡算法，将用户请求分配到可用的真实服务器上进行处理。

但是不能简单地理解为负载均衡的应用场景就是将请求分发到不同的服务器上去响应，目的是让服务器的负载达到均衡的状态。现阶段负载均衡之所以能成为每一个在线应用必不可少的环节，成为网络架构中不可或缺的组件，是因为负载均衡在服务的过程中发挥以下作用：

（1）解决并发压力，提升应用处理性能；

（2）避免单点故障，实现应用高可用；

（3）弹性伸缩扩展，节省服务器资源消耗；

（4）对资源进行压缩传递，SSL 证书卸载，减轻服务器负担；

（5）提供访问控制的黑名单/白名单功能，实现安全防护。

基于上述作用，负载均衡被用于很多商业应用场景来解决问题，尤其是在云平台的网络配置中，负载均衡是用户使用最多的一款产品。

下面介绍负载均衡在云环境中常见的五个应用场景。

（1）Web 服务器负载均衡。

通过负载均衡对 Web 服务器建立有效的健康检查和负载均衡机制，而不改动原有的网络架构，从而提供一种高性价比、透明的容错方法对网络设备进行扩容，并提升服务器的带宽利用率、吞吐量和数据处理能力，提高 Web 业务的灵活性和可用性。

（2）云网络内部系统间业务调用。

如果 VPC 内部不同的系统间的访问量很高，可以通过负载均衡在 VPC 内部的系统间建立有效的负载均衡机制，将流量分发到不同的后端服务器上，从而分担系统之间的业务请求负荷，提升系统的数据处理能力，提高访问的效率。

（3）电商促销抢购场景。

电商业务有明显的网络潮汐效应，通过负载均衡和弹性伸缩组的无缝对接，可以实现自动创建后端云服务器，并有效校验服务器的健康状况，将流量自动分发到新扩容的服务器上，缓解电商促销高峰时的业务系统压力。

（4）跨可用区同城容灾。

负载均衡可以将访问请求和流量分发到多个可用区的云服务器上进行响应，建立实时的跨可用区同城容灾机制，满足不同企业对业务系统高可用性的要求。

（5）HTTPS 业务 SSL 卸载。

SSL 卸载技术是通过将 HTTPS 应用访问过程中的 SSL 加解密过程转到负载均衡设备上，从而减少服务器端对 SSL 证书进行处理所产生的性能压力，提升客户端的访问响应速度。

3.6.3　平安云的负载均衡

平安云的负载均衡指的是在平安云网络环境中提供给 VPC 使用的负载均衡产品。平安云负载均衡不是一个网络环境中的功能组件，而是一款能够适用于绝大多数客户的负载均衡产品。所以平安云负载均衡产品不会将部署过程、安全防御、限速控制、智能监控等功能寄希望于其他网络组件，而是要权衡线上的多种业务场景来提供一款多功能的负载均衡器。

1. ELB 服务逻辑架构

ELB 服务的逻辑架构如图 3.18 所示，由 ELB 实例、监听器、服务器资源池三部分构成。

1）负载均衡实例

负载均衡实例提供负载均衡服务。负载均衡实例提供服务 IP 地址作为客户端访问的统一接口。每个负载均衡实例能够配置多个监听器，对同一个服务 IP 地址提供多端口的监听服务。

2）监听器

监听器的作用是接收客户端的请求并将请求转发给服务器资源池中的主机，并对后端

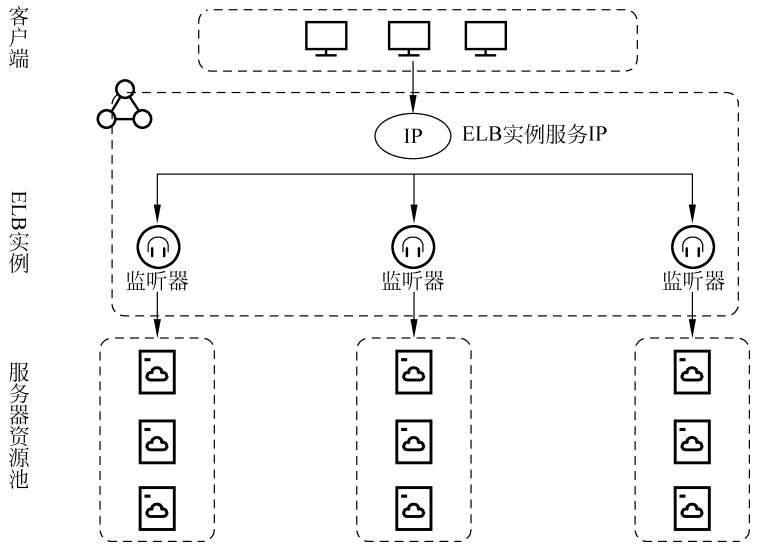

图 3.18　ELB 服务的逻辑架构

主机进行健康检查。每个监听器都能够配置一个特定的服务器资源池。

3）服务器资源池

服务器资源池是一组用于接收前端请求的 ECS 或 BMS 实例。可以添加一个或多个 ECS 实例或 BMS 实例到服务器资源池并配置权重和后端服务端口。

2. ELB 服务核心功能及类型

ELB 服务包含监听与调度两大核心功能，可在监听器中配置前端端口、协议、调度算法、URL 转发、健康检查、证书、会话保持、访问控制以及服务器资源池关联信息等。具体功能如表 3.2 所示。

表 3.2　ELB 服务核心功能

名　　词	说　　明
负载均衡服务	平安云为客户提供的一种网络负载均衡服务。该服务为后端服务器形成的计算节点集群提供统一的访问接口。平安云主要提供 TCP、UDP、HTTP 和 HTTPS 四种协议的负载均衡服务
负载均衡实例	负载均衡服务的实际载体。用户通过创建负载均衡实例及对实例进行配置，达到部署负载均衡服务的目的
监听器	负载均衡实例中的重要功能项，提供监听端口、协议、负载均衡算法、健康检查以及服务器资源池的配置。负载均衡服务的运行情况主要依赖于用户对监听器的创建和配置
服务器资源池	提供某种互联网服务的计算资源集群形成了一个服务器资源池。负载均衡服务为互联网客户端提供统一的访问接口，并通过负载均衡算法将客户端的访问流量分发给服务器资源池中的成员服务器，以此达到负载均衡的目的

续表

名　　词	说　　明
访问控制	负载均衡提供监听级别的访问控制，用户可以通过白名单或黑名单的方式，对不同的监听配置不同的访问控制策略，从而达到控制访问权限的目的
证书管理	创建 HTTPS 时，用户需要自助上传 SSL 证书，并使用证书进行安全验证

ELB 服务分为公网和私网两种类型。

（1）**公网型**：通过公网 IP 地址对外提供服务，提供四层和七层负载均衡能力，适用于在互联网上发布网站或应用程序上线，可以支持多可用区部署，当主可用区服务发生故障时，负载均衡可以自动切换到备份可用区，实现服务的高可用，其部署架构如图 3.19 所示。

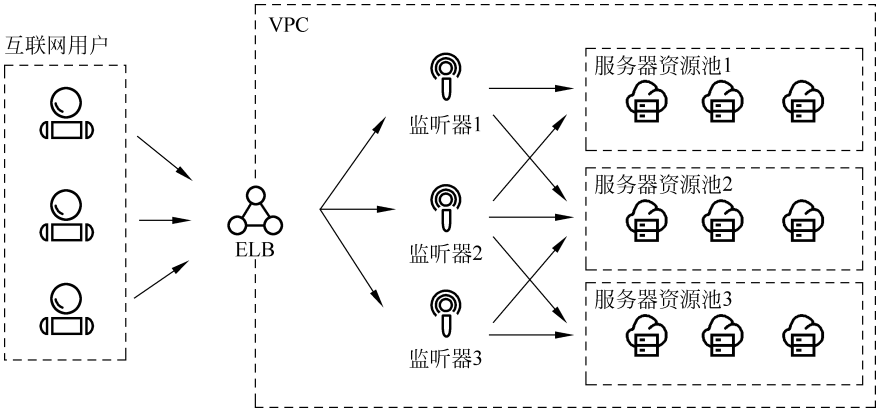

图 3.19　公网型 ELB 部部署架构

（2）**私网型**：通过私网 IP 地址提供服务，具有四层负载均衡能力，适用于承载 VPC 内部系统间的调用请求，其部署架构如图 3.20 所示。

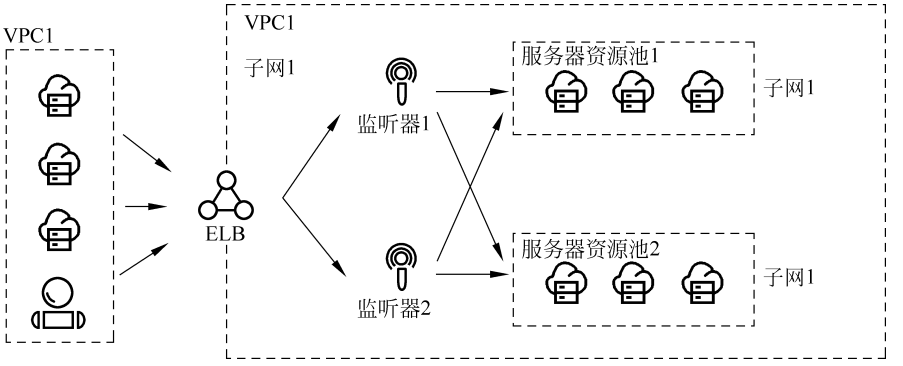

图 3.20　私网型 ELB 部署架构

综上所述，平安云采用当今最主流的负载均衡技术 LVS 和 Nginx，实现了负载均衡在云平台上的自动化部署，将负载均衡器打包成一款实用性很强的产品，通过图形化界面来满足用户上手可用的需求。

第4章

CHAPTER 4

计算服务

计算服务，作为应用运行的载体，是给各种类型应用提供运行环境的服务。在性能的追求上，向下力求更少的 CPU 内存消耗、更细致的资源控制粒度，向上充分利用日益提升的硬件性能，不断提升计算处理能力；在服务的形态上，随着硬件和软件交替革新，从 PC 服务器到小型机，从 CPU 到 GPU、FPGA，从虚拟机到裸金属、容器、Serverless，计算服务不断推出新的计算形态以满足不断发展的业务需求。

作为云计算的核心服务，计算服务的设计目标主要是如下两个方向。

（1）追求资源的投入产出比。尽可能地将底层 CPU、内存、硬盘等硬件资源软件定义化，实现按需调度，将单一硬件资源虚拟抽象化以实现最小的资源分割；扩大单位资源的并发能力及资源交付速度，降低人工运维成本；海量规模场景下通过优化资源调度算法来提升资源利用率，同时保证服务安全可靠。

（2）优先满足大批量、标准化的应用场景，再逐步扩展到更多个性化的应用场景。早期互联网时代的虚拟主机建站服务，只要实现操作系统层面的资源隔离即可，无须考虑企业内应用的多样性计算要求。借助主机虚拟化技术解决了用户对通用操作系统快速使用的需求，催生出最早期的计算服务应用场景。伴随日益丰富的用户需求（例如多样化的应用部署环境）及快速迭代的云计算技术，如今计算服务已经呈现出前面提到的多种形态。仅从云服务器这一类计算服务来看，就推出了通用型、内存型、I/O 型、高性能计算型等多种产品形态的云服务器套餐，每种形态下又划分多个细分规格族。另外，基于安全隔离及性能上的更多需求，云服务器还推出了专属服务器、异构服务器、裸金属服务器等计算产品。面向更轻量级、更高资源利用率的计算需求，业界还推出了容器服务、Serverless 等多种新型计算产品。本章将针对云服务器这一大类计算服务展开介绍。

4.1　软件定义计算设计要素

如何构建计算资源共享最大化和管理自动化的软件定义计算是计算服务的核心需求。业界流行的解决方案是计算虚拟化技术结合分布式计算资源调度来构建计算服务。

软件定义计算的设计要素主要包含计算虚拟化和分布式计算调度这两方面。除此之外，计算服务还要考虑高可用设计，使平台提供的产品达到对客户承诺的服务等级协议(Service Level Agreement，SLA)，也就是保障服务的可持续性。

4.1.1　虚拟化引擎

目前主流的几种虚拟化引擎有 ESXi、Hyper-V、KVM、Xen。前两个都是商业化的闭源产品，分别属于 VMware 和微软；后两个是开源的虚拟化技术。随着开源虚拟化技术的不断发展，会发现一个趋势：以前商业虚拟化产品 VMware 市场占比大，慢慢地开源的虚拟化产品市场占比更高。特别在互联网行业，公有云厂商基本都采用开源的 KVM 作为虚拟化引擎，2017 年 AWS 宣布它已经创建了新的基于 KVM 的虚拟化引擎，而不是它多年来依赖的 Xen 引擎。国内的公有云厂商，如阿里、腾讯、平安等也都是基于 KVM 作为虚拟化引擎。

虚拟化是构建云基础架构不可或缺的关键技术之一。云计算的云端系统实质上就是一个大型的分布式系统。通过虚拟化可以在一个物理平台上虚拟出更多的虚拟平台，而其中的每一个虚拟平台都可以作为独立的终端加入云端的分布式系统。比起直接使用物理平台，虚拟化在资源有效利用、动态调配和高可靠性方面有巨大的优势。利用虚拟化技术，企业不必抛弃现有的基础架构即可构建全新的基础架构，从而更加充分地利用原有的 IT 资源。

计算虚拟化技术的实现形式是在系统中加入一个虚拟化层，将下层的资源抽象成另一种形式的资源，供上层使用，如图 4.1 所示。计算虚拟化技术的通用实现方案是将软件和硬件相互分离，在操作系统与硬件之间加入一个虚拟化软件层，将服务器物理资源抽象成逻辑资源，使得上层操作系统可以直接运行在虚拟环境上，并允许具有不同操作系统的多个虚拟机(Virtual Machine，VM)相互隔离，并行运行在同一台物理机上，从而提供更高的 IT 资源利用率和灵活性。每台虚拟机都是一个完整的系统，它具有处理器、内存、网络设备、存储设备和 BIOS。在虚拟机中运行的操作系统软件称为虚拟机操作系统 Guest OS。计算虚拟化的这个软件层，也就是虚拟机监控器(Virtual Machine Monitor，VMM)，通常被称为 Hypervisor。

按照虚拟化的程度计算虚拟化可以分为准虚拟化和全虚拟化。在介绍 Xen、KVM 等著名虚拟化引擎之前，先来了解下准虚拟化与全虚拟化的区别。

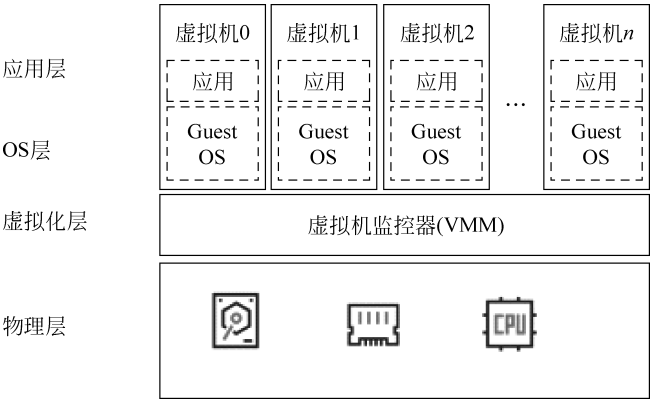

图 4.1　计算虚拟化

1. 准虚拟化与全虚拟化

1）准虚拟化

软件虚拟化可以在缺乏硬件虚拟化支持的平台上完全通过 Hypervisor 软件来实现对各个虚拟机的监控，以保证它们之间彼此独立和隔离。但是付出的代价是软件复杂度的增加，以及性能上的损失。减轻这种负担的一种方法就是改动客户操作系统，使它以为自己运行在虚拟环境下，能够与 Hypervisor 协同工作。这种方法就叫准虚拟化（Para-Virtualization），也叫半虚拟化。本质上，准虚拟化弱化了对虚拟机特殊指令的被动截获要求，将其转化成客户机操作系统的主动通知。但是，准虚拟化需要修改客户机操作系统的源代码来实现主动通知。

2）全虚拟化

全虚拟化（Full Virtualization）与准虚拟化不同，全虚拟化为客户机提供了完整的虚拟 X86 平台，包括处理器、内存和外设，支持运行任何理论上可在真实物理平台上运行的操作系统，为虚拟机的配置提供了最大程度的灵活性。不需要对客户机操作系统做任何修改即可正常运行任何非虚拟化环境中已存在的基于 X86 平台的操作系统和软件，是全虚拟化无可比拟的优势。可以预见，基于硬件的全虚拟化产品将是未来虚拟化技术的核心。

下面逐一介绍几个比较著名的虚拟化引擎。

2. Xen 虚拟化

Xen 是由剑桥大学计算机实验室开发的一个开源项目，支持全虚拟化和准虚拟化。Xen 能够支持多种处理器，如 X86、X86-64、Power PC 和 ARM 等。目前 Xen 支持的客户操作系统有 Linux、NetBSD、FreeBSD、Solaris、Windows 和其他一些常用的操作系统。

Xen 包含三种基本组件，分别是 Xen Hypervisor、Domain 0（XEN DM&C）和 Domain U（PV Guest & HVM Guest）。Xen 体系架构如图 4.2 所示。

1）Xen Hypervisor

XEN Hypervisor是介于操作系统和硬件之间的一个软件描述层。它负责在各个虚拟机之间进行CPU调度和内存分配。XEN Hypervisor不仅抽象出虚拟机的硬件，同时还控制着各个虚拟机的执行。XEN Hypervisor不会处理网络、存储设备、视频以及其他I/O。

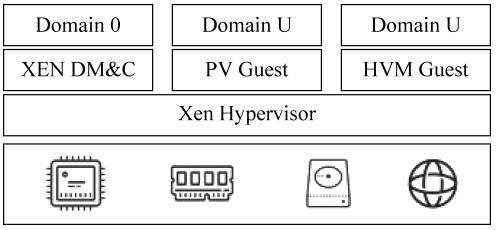

图4.2　Xen体系架构

2）Domain 0

Domain 0是一个修改过的Linux Kernel，是唯一运行在Xen Hypervisor之上的系统虚拟机，它拥有访问物理I/O资源的权限，同时和系统上运行的其他虚拟机进行交互。Domain 0需要在其他Domain启动之前启动。

3）Domain U

运行在Xen Hypervisor上的所有准虚拟化（Para-Virtualization）虚拟机被称为“Domain U PV Guest”，其上运行着被修改过内核的操作系统，如Linux、Solaris、FreeBSD等其他UNIX操作系统。所有的全虚拟化虚拟机被称为“Domain U HVM Guest”，其上运行着不用修改内核的操作系统，如Windows等。

3. KVM虚拟化

KVM(Kernel-based Virtual Machine)是基于Linux内核的开源的全虚拟化解决方案，支持硬件辅助虚拟化技术(Intel VT或AMD-V)。KVM从2.6.20版本开始被合入Kernel主分支维护，成为Linux的重要模块之一。KVM可在具备Intel VT或AMD-V功能的X86平台上运行，它也被移植到S/390、Power PC与IA-64平台上。在Linux内核3.9版中，加入了ARM架构的支持。

KVM是一个独特的管理程序。通过将KVM作为一个内核模块实现，在虚拟环境下Linux内核集成管理程序将其作为一个可加载的模块可以简化管理和提升性能。在这种模式下，每个虚拟机都是一个常规的Linux进程，运行在QEMU-KVM进程的地址空间，通过Linux调度程序进行调度。KVM本身只能够提供CPU、内存虚拟化等部分功能，而I/O等其他设备的虚拟化则需要依靠QEMU来完成，如图4.3所示。

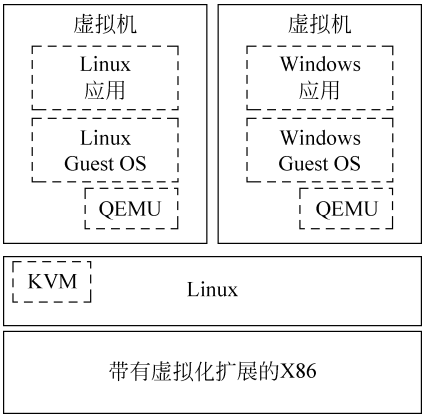

图4.3　KVM体系架构图

不过，采用QEMU纯软件的方式来模拟I/O设备虚拟化的效率并不非常高。目前，也可以考虑基于Virtio这个Linux上的设备驱动标准框架在客户机中使用半虚拟化驱动（PV Driver）来提高客户机的I/O性能。

4. ESXi

ESXi 是 VMware 的一个裸机虚拟化管理程序，是创建自动化的动态数据中心的基础。它提供一个虚拟层对物理主机的处理器、内存、存储和网络连接资源进行抽象处理，使其转化成多个虚拟机。在 ESXi 体系结构中，虚拟机中运行的应用无须直接访问底层硬件设备即可访问 CPU、内存、磁盘和网络接口。ESXi 虚拟机管理程序称为 VMkernel，它可以从虚拟机监控器（VMM）接收虚拟机的资源请求，然后将这些请求传送给物理硬件。每台虚拟机上的虚拟机监控器（VMM）都负责向虚拟机提供虚拟硬件并接收请求，如图 4.4 所示。

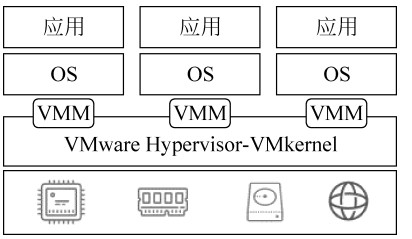

图 4.4　ESXi 体系架构

5. Hyper-V

Hyper-V 是微软的硬件虚拟化产品，是基于虚拟机监控程序（VMM）的虚拟化技术。Hyper-V 使用 Windows 虚拟机监控程序，需要具有二级地址转换（SLAT）的 64 位处理器。在大多数情况下，虚拟机监控程序管理硬件和虚拟机之间的交互，控制访问硬件为虚拟机提供了隔离的运行环境。在某些配置中，虚拟机或虚拟机中运行的操作系统具有直接访问图形、网络或存储硬件的能力。

Hyper-V 主要包括 Windows 虚拟机监控程序、Hyper-V 虚拟机管理服务、虚拟化 WMI 提供程序、虚拟机总线（VMbus）、虚拟化服务提供商（VSP）和虚拟基础结构驱动程序（VID）。

Hyper-V 可以采用准虚拟化和全虚拟化两种模拟方式创建虚拟机。准虚拟化方式要求 Linux 虚拟机的操作系统安装特定的设备驱动集成服务（Integration Service），Windows Server 2012 和 Windows 8.1 以上版本已经内置该服务，集成服务是允许虚拟机与 Hyper-V 主机通信的服务，可以使虚拟机具有更多的特性和更好的性能；全虚拟化方式要求 CPU 支持全虚拟化功能（如 Intel VT 或 AMD-V），以便能够创建使用更多操作系统（如 Mac OS、没有集成服务驱动的其他 Linux 发行版）的虚拟机。

4.1.2　分布式计算资源调度

计算虚拟化技术解决了单机上计算单元的模拟和模拟出来的计算单元的隔离问题，那么对于成百上千乃至上万台物理机构成的计算资源池，如何进行集群化管理、分布式调度虚拟机资源使得既能够快速部署虚拟机又能够满足资源利用最大化呢？这就是分布式计算资源调度需要解决的问题。

当作为计算资源的物理服务器数量到达一定规模时，需要考虑如何更好地组织和调度资源。经常把云计算比作水和电，而 IT 资源的供给就需要许多类似于大大小小的水厂、电

厂的 IT 资源工厂，也就是所说的 IT 数据中心。IT 资源不是无限制供给的，数据中心一般都会规划一定的容量，然后人们在多个地域建立多个数据中心，而数据中心内部又会划分高可用域、机架、集群，按照从大到小的粒度来将计算资源组成计算资源池。而资源调度算法就是为了解决指定虚拟机实例到整个云数据中心计算资源池内最合适的物理机的问题。

下面来看看开源计算平台 CloudStack 和 OpenStack 是如何做计算资源调度的。

1. CloudStack 调度

CloudStack 的计算资源调度策略通过部署模式(Deployment Model)和部署计划(Deployment Planner)协同完成。

1) 部署模式

部署模式包括 Strict(强制)模式和 Preferred(优先)模式。Strict 模式要求物理机必须严格满足部署计划的要求；Preferred 模式相比 Strict 模式限制要宽一些，优先选择满足 Strict 要求的物理机，如果没有满足要求的物理机，则会放宽条件。

2) 部署计划

(1) 优先适配计划(First Fit Planner)。它是指按照每个满足条件的集群的总容量(默认是 CPU)的使用率从低到高排序，然后在使用率最低的集群中随机选取一个符合条件的物理机。

(2) 隐式独占计划(Implicit Dedication Planner)。在使用隐式独占计划及 Strict 模式的情况下，会选取该租户独占的物理机；若使用 Preferred 模式，会优先选取该租户独占的物理机，如果没有满足条件的物理机则会选取那些其上运行的 VM 没有使用隐式独占计划的物理机。

(3) 跳过启发式计划(Skip Heuristics Planner)。它是指调度时会忽略集群容量阈值的限制继续尝试部署，一般用于 VM 的高可用场景，也就是用于当 VM 发生故障迁移时的调度。

(4) 用户集中机架部署计划(User Concentrated Pod Planner)。它是指如果租户已经有 VM 运行在某些机架上了，那么就优先选取这些机架上的物理机，机架物理机按其上运行的该租户的 VM 数量从高到低排序，先选取 VM 数量多的物理机。

(5) 用户分散计划(User Dispersing Planner)。这个部署计划综合考虑了优先适配计划和用户集中机架部署计划，通过传入 0～1 的参数来决定两种计划的权重，如果为 0，则按优先适配计划的方式部署，如果为 1，则按用户集中机架部署计划的方式部署，如果在 0～1，则计算权重来决定租户 VM 的集中度。

2. OpenStack 调度

OpenStack 的计算资源调度策略主要是由过滤调度器 FilterScheduler 和随机调度器 ChanceScheduler 实现的，其中 FilterScheduler 作为默认的调度引擎实现了基于物理机过滤(Filtering)和权值计算(Weighting)的调度算法，而 ChanceScheduler 则是基于随机算法来

选择可用物理机的简单调度引擎。

图 4.5 是 FilterScheduler 的虚拟机调度过程。它支持多种内置的过滤器(Filter)和计量器(Weigher)来满足一些常见的业务场景。在设计上,OpenStack 基于 Filter 和 Weigher 支持第三方扩展,因此用户可以通过自定义 Filter 和 Weigher,或者使用 JSON 资源选择表达式来扩展虚拟机的调度策略,从而满足不同的业务需求。

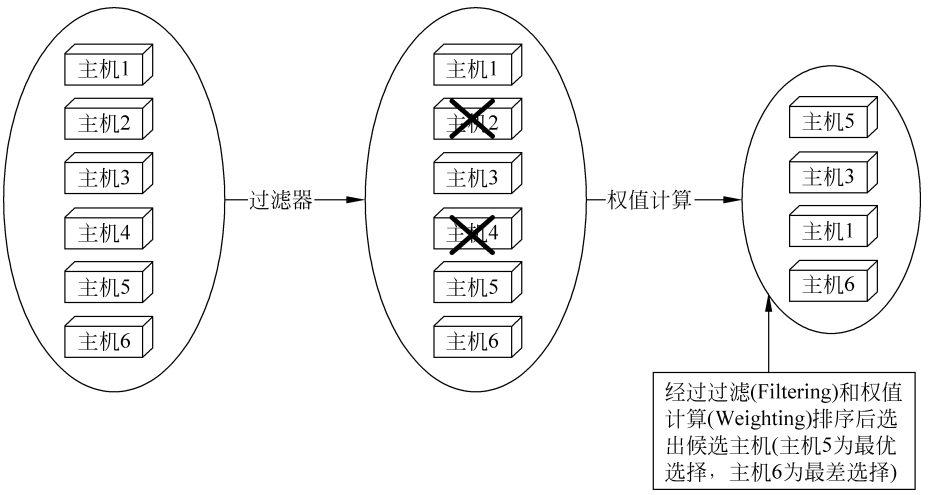

图 4.5　OpenStack FilterScheduler 调度流程

1) 调度 Filter

OpenStack 内置的 Filter 组件主要包括:

(1) 计算节点过滤器(ComputeFilter),过滤计算节点宕机的物理机;

(2) CPU 核心过滤器(CoreFilter),过滤 vCPU 不满足虚拟机请求的物理机;

(3) 磁盘过滤器(DiskFilter),过滤 Disk 不满足虚拟机请求的物理机;

(4) 内存过滤器(RamFilter),过滤 RAM 不满足虚拟机请求的物理机;

(5) 镜像属性过滤器(ImagePropertiesFilter),过滤镜像属性(OS 属性、Hypervisor 类型等)不满足虚拟机请求的物理机;

(6) 相同物理机过滤器(SameHostFilter),过滤和指定虚拟机在同一个物理机上的物理机;

(7) 不同物理机过滤器(DifferentHostFilter),过滤和指定虚拟机在不同物理机上的物理机;

(8) JSON 表达式过滤器(JsonFilter),过滤不满足 OpenStack 自定义的 JSON 资源选择表达式的物理机。例如,JSON 资源选择表达式形如 query＝'["＞","＄cpus",4]'表示过滤掉 CPU 小于或等于 4 核的物理机。

2) 调度 Weigher

FilterScheduler 通过一系列 Filter 过滤选出了符合条件的物理机,接下来会通过

Weigher 对每个物理机计算权重值 Weight，大概计算公式如下：

$$\text{Weight} = w1_multiplier * norm(w1) + w2_multiplier * norm(w2) + \cdots$$

其中：multiplier 是每个计量器事先定义好的乘数；w 是每一个计量器 Weigher；norm()是每个计量器特定的计算函数。

常用 Weigher 包括：

（1）内存计量器(RAMWeigher)，根据物理机的可用内存计算权重；

（2）磁盘计量器(DiskWeigher)，根据物理机的可用磁盘容量计算权重；

（3）I/O 操作(Input/Output Operation)计量器(IoOpsWeigher)，根据物理机的 I/O 负载计算权重。

最后 FilterScheduler 根据每个物理机的权重值排序，优先选取权重值大的物理机，如图 4.6 所示。

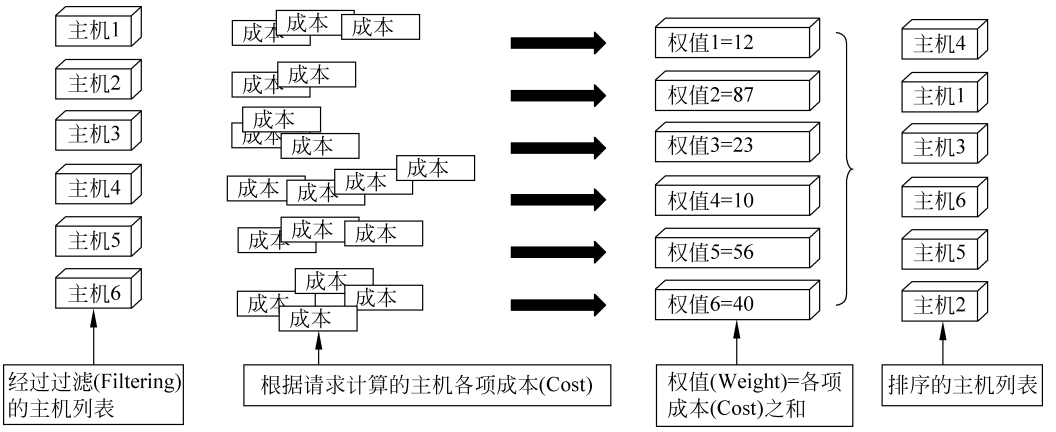

图 4.6　FilterScheduler 计算权重流程

3）组合调度策略

基于 Filter 和 Weigher 可以组合形成如下常用的调度策略。

（1）打包(Packing)：虚拟机尽量放在包含虚拟机数量最多的物理机上。

（2）剥离(Stripping)：虚拟机尽量放在包含虚拟机数量最少的物理机上。

（3）CPU 负载均衡(CPU Load Balance)：虚拟机尽量放在可用 CPU 核最多的物理机上。

（4）内存负载均衡(Memory Load Balance)：虚拟机尽量放在可用内存最多的物理机上。

（5）亲和性(Affinity)：多个虚拟机需要放在相同的物理机上。

（6）反亲和性(Anti-Affinity)：多个虚拟机需要放在不同的物理机上。

（7）CPU 利用率负载均衡(CPU Utilization Load Balance)：虚拟机尽量放在 CPU 利用率最低的物理机上。

4.1.3 高可用设计

计算服务的高可用大致需要从以下三方面考虑：平台高可用、虚拟机高可用及物理机高可用。

1. 平台高可用

平台高可用是指计算平台本身的高可用，能够不间断地为用户提供计算产品的自助操作，如门户网站访问、云主机购买、开机、关机、云硬盘挂载等。

平台高可用设计要点如下。

(1) 避免单点故障：可部署跨可用区跨地域的多节点结合负载均衡技术实现同城灾备、异地灾备，做到自动故障切换。

(2) 应用的高可用性：从平台角度，提供服务治理(服务降级、限流)，容错自愈的能力，提高服务可用性；提供系统监控(CPU、内存、磁盘)，链路监控，日志监控等能力，便于故障追踪、自动预警；提供数据分片、读写分离等手段有效应对大规模数据量，实现数据库的无缝扩容。

(3) 分布式架构下的可伸缩设计：既支持基于服务器硬件能力升配/降配的垂直伸缩，也支持服务器数量增减的水平伸缩。

2. 虚拟机高可用

虚拟机高可用是指当虚拟机操作系统异常宕机、虚拟化 QEMU-KVM 进程被物理机操作系统异常终止时，计算平台会自动发现该现象并在当前物理机或另外可用的物理机上重新启动该虚拟机。

虚拟机上的应用通常会配置为随系统开机自启动，来适应这种虚拟机异常会被计算平台重新调度的机制。由于虚拟机高可用机制的存在，应用间接具备了快速恢复的能力。

虚拟机高可用主要由计算平台结合监控来实现。监控对象包括虚拟机的网络、虚拟机的虚拟化进程、虚拟机的磁盘使用活动等。

3. 物理机高可用

物理机高可用是指当物理机由于一些突发的、不可预知的原因发生异常宕机时，计算平台会自动发现该现象并将该物理机上面的所有虚拟机在另外可用的物理机上面重新启动。这些原因包括硬件故障、驱动故障、系统故障等。

物理机高可用相比虚拟机高可用，本质区别在于控制粒度。物理机高可用机制触发时，直接重新调度其上的所有虚拟机，不需要每个虚拟机单独去发现调度，提高了调度效率。

另外，物理机高可用设计还可以引入基于带外控制的物理机状态检测、隔离和恢复机制，以保障其高可用触发机制的准确性。

4.2　计算服务总体架构

4.2.1　计算能力分层管理

计算服务能力涉及的范围很广，并且相关产品功能特性仍在不断更新迭代，因此在计算产品能力构建上，一般会分为三层模块：资源控制层、功能服务层以及操作系统辅助功能层。

1. 资源控制层

该层体现计算产品的核心能力，提供基础的资源生命周期管理、资源变化管理。例如，云服务器的创建、删除、升降配、重启等；BMS 裸机服务器的创建、销毁、重启等。

此层面主要关注点在底层资源集群管理和分配调度，目标是配合更新的计算、存储、网络技术能力，提供更新型、更高性能的计算资源，同时在资源控制上能适应更多维度、达到云运营优化的效果。

2. 功能服务层

该层提供该类计算产品的特有功能。例如，云服务器的控制台访问、宿主物理机硬件故障自动恢复、镜像复制等；BMS 裸机服务器的控制台访问、硬件查看、镜像复制等。

此层面是用户在使用过程中最为关注的计算能力使用层。资源在交付完毕后，如何运维和利用计算能力是整个资源生命周期管理中的主要工作。随着功能层的不断更新，云原生应用可以借助云计算产品功能拓展实现应用的非功能性需求，例如使用故障自愈功能实现应用高可用方案、利用镜像复制实现应用快速扩容等。

3. 操作系统辅助功能层

该层面提供计算资源之上的操作系统层面的功能支持。例如，账号密码设置、开机脚本执行、系统补丁、安全组件、默认监控组件等。

用户在购买计算产品后，要求能够得到可以管理控制的操作系统，所以计算产品交付主要包含两个步骤：资源交付及操作系统交付。操作系统交付模块需要能满足以下几类需求。

（1）操作系统登录访问控制：实现安全可靠的密码或证书支持，实现多租户管理，以及接入加密机实现平台级的密码保护，仅用户自己可见。

（2）实现操作系统交付时的自定义应用编排能力：例如，开机可运行自定义系统初始化脚本，以及应用部署脚本，从而用户无须二次登录云服务器进行应用部署。

（3）在交付操作系统时能提供基础的辅助关联服务：例如，系统安全检视、基础监控检视，保证操作系统在未正式投产定制前，依靠云平台即可保障基础可靠性。