第 3 章

CHAPTER 3

边缘计算相关技术

3.1 边缘计算关键技术

3.1.1 虚拟机和容器

云计算技术的进步，使在诸如基站和网关等大量通用服务器上部署虚拟机变得更加容易。云计算能够提供强大的处理能力和海量的存储资源，云计算和物联网的整合也已被证明有利于提供新的服务，因此，将云计算功能集成到移动网络，能够为新兴移动服务的供应和管理提供高效的解决方案。

虚拟机(Virtual Machine，VM)技术是虚拟化技术的一种，所谓虚拟化技术就是将事物从一种形式转变成另一种形式，具体而言就是在一个宿主计算机体系结构上进行客户机各种操作系统模拟运行，对宿主计算机、客户机体系结构无明确要求，例如可以在一个 x86 计算机上运行基于 ARM 体系结构的不需要做任何修改的系统。从这个角度来为虚拟机下定义，可知虚拟机主要是指虚拟技术运行的媒介，即通过软件模拟的具有完整硬件系统功能的、在一个完全隔离环境中运行的一

个完整的计算机系统。

虚拟化技术可应用的领域十分广泛，但是在不同的领域中应用原理存在着明显差异。具体而言，虚拟化技术主要通过拆分、整合、迁移这三项内容得以实现。虚拟机技术的应用多采用拆分原理，当某台计算机性能较高但是工作负荷与其不相匹配时，容易造成资源浪费，使用拆分虚拟技术即可将该计算机拆分为逻辑上的多台计算机，实现多名用户共同使用，在此情况下该计算机硬件资源的利用程度将会明显提高。

虚拟机技术是将一个物理计算机划分为一个或多个完全孤立的虚拟机技术，这些虚拟机并非运行在物理硬件之上，而是运行在通过虚拟化软件来生成一个虚拟的物理硬件层之上。实际上对于操作系统来讲就是运行在其之上的应用程序，但是在虚拟机的使用中会共享计算机的物理硬件，并且具有明显的优势：资源共享和隔离。在虚拟机的状态下，各种资源可以根据需要分配，甚至可以不重启虚拟机即可分配硬件资源；虚拟机环境能够实现隔离，即能够根据自身使用的需求在物理计算机上运行几个不同的操作系统，它们之间独立运行，但各自互不干扰。可以是同一种操作系统，也可以是不同的操作系统。这也是虚拟机技术和操作系统虚拟化技术的最大区别。

虚拟机通常包括整个操作系统及应用程序。它们还需要与之一起运行的管理程序来控制 VM。由于包括操作系统，因此大小为数吉字节。使用 VM 的缺点之一是需要花费几分钟来启动 OS，并且初始化所托管的应用程序，因此，又引入了容器技术。容器是轻量级的 OS 级虚拟化，可在资源隔离的过程中运行应用程序及其依赖项。运行应用程序所需的所有必要组件都打包为一个映像，并且可以重复使用。执行映像时，它在隔离的环境中运行，并且不共享内存、CPU 或主机 OS 的磁盘。这保证了容器内部的进程无法监视容器外部的任何进程。同时，这些容器是轻量级的，并且大多在兆字节范围内。

与传统云计算中的虚拟机技术不同，新兴的容器技术是一种内核轻量级的操

作系统层虚拟化技术。它能够划分物理机的资源，创建多个与 VM 相比尺寸小得多的隔离用户空间实例。由于容器的轻量级特性，其能够在执行应用程序或服务时，提供简单的实例化。借助于容器技术的使用，能够实现边缘计算服务的便携式运行，为移动用户带来很大的便利。此外，由于容器技术提供了快速打包的机制，服务器端也能非常方便地将服务部署到大规模互联的边缘计算平台。

以 Docker 为主的容器技术是边缘设备上微服务的运行环境，并不需要特殊的虚拟化支持，然而，硬件虚拟化可以为 Docker 容器提供更加安全的隔离。2017 年年底，OpenStack 基金会正式发布基于 Apache 2.0 协议的容器技术 Kata Containers 项目，主要目标是使用户能同时拥有虚拟机的安全及容器技术的迅速和易管理性。云服务提供商使用虚拟化或者容器来构建平台及服务，如图 3.1 所示，应用程序和依赖的二进制库被打包运行在独立的容器中。在每个容器中，网络、内存和文件系统是隔离的。容器引擎管理所有的容器。所有的容器共享物理主机上的操作系统内核。

图 3.1　典型的容器结构

如上所述，以 Docker 为主的容器技术逐渐成为边缘计算的技术标准，各大云计算厂商选择容器技术构建边缘计算平台的底层技术栈。

边缘计算的应用场景非常复杂。从前面的分析可以清晰地看到边缘计算平台

并不是传统意义上的只负责数据收集转发的网关。更重要的是，边缘计算平台需要提供智能化的运算能力，而且能产生可操作的决策反馈，用来反向控制设备端。过去，这些运算只能在云端完成。现在，需要将云端的计算框架通过裁剪、合并等简化手段迁移至边缘计算平合，使能在边缘计算平台上运行云端训练后的智能分析算法，因此，边缘计算平台需要一种技术在单台计算机或者少数几台计算机组成的小规模集群环境中隔离主机资源，实现分布式计算框架的资源调度。

边缘计算所需的开发工具和编程语言具有多样性。目前，计算机编程技术呈百花齐放的趋势，开发人员运用不同的编程语言解决不同场景的问题已经成为常态，所以在边缘计算平台需要支持多种开发工具和多种编程语言的运行环境，因此，在边缘计算平台使用一种运行时环境的隔离技术便成为必然的需求。

容器技术和容器编排技术逐渐成熟。容器技术是在主机虚拟化技术后最具颠覆性的计算机资源隔离技术。通过容器技术进行资源的隔离，不仅对 CPU、内存和存储的额外开销非常小，而且容器的生命周期管理也非常快捷，可以在毫秒级的时间内开启和关闭容器。

3.1.2　软件定义网络

目前，云计算和边缘计算都不能把对方所取代，可以互为补充、协同发展，可以通过软件定义网络(Software Defined Network，SDN)管理云计算和边缘计算之间的交互。随着移动智能终端设备的增加，给云计算传输网络带来了巨大的挑战和压力，而这种现状随着物联网设备和移动终端的骤增将进一步加剧。为此，创建云计算和边缘计算资源统一、协同、管控平台，是应对高负荷和高延迟的有效方式。SDN 作为统一云计算和边缘计算的技术，可在动态的、实时的、不可预测的场景下为负载配备资源，利用开放编程接口高效完成工作任务。

SDN 是一种新型的网络体系架构，SDN 技术使边缘网络具有智能化、可编程

和易于管理的特点。SDN 的主要思想是分离网络的控制面和数据面，它的优势主要包括在通用硬件上创建网络控制面、通过 API 开放网络功能、远程控制网络设备及将网络智能从逻辑上解耦为不同的基于软件的控制器。借助 SDN 技术可以实现移动边缘计算平台所需的分层管理。

SDN 与传统网络最大的区别在于可以通过编写软件的方式灵活定义网络设备的转发功能。在传统网络中，控制平面功能分布式地运行在各个网络节点中，因此，新型网络功能的部署需要所有相应网络设备的升级，导致网络创新难以实现，而 SDN 将网络设备的控制平面与转发平面分离，并将控制平面集中实现，这样，新型网络功能的部署只需要在控制节点进行集中的软件升级，从而实现快速灵活地定制网络功能。另外，SDN 体系架构还具有很强的开放性，它通过对整个网络进行抽象，为用户提供完备的编程接口，使用户可以根据上层的业务和应用个性化地定制网络资源来满足其特有的需求。由于其开放可编程的特性，SDN 有可能打破某些厂商对设备、协议及软件等方面的垄断，从而使更多的人参与到网络技术的研发工作中。SDN 不仅是新技术，而且变革了网络建设和运营方式，从应用的角度构建网络，用 IT 的手段运营网络。

目前，各大厂商对 SDN 的系统架构都有自己的理解和认识，而且也都有自己独特的实现方式，其中最有影响力的开放网络基金会（Open Networking Foundation，ONF）在 SDN 的标准化进程中占有重要地位。图 3.2 给出了 ONF 定义的 SDN 系统架构，它认为 SDN 的最终目的是为软件应用提供一套完整的编程接口，上层的软件应用可以通过这套编程接口灵活地控制网络中的资源及经过这些网络资源的流量，并能按照应用需求灵活地调度这些流量。从图 3.2 中可以看到，ONF 定义的架构由 4 个平面组成，即数据平面（Data Plane，DP）、控制平面（Control Plane，CP）、应用平面（Application Plane，AP）及右侧的管理平面（Management Plane，MP），各平面之间使用不同的接口协议进行交互。

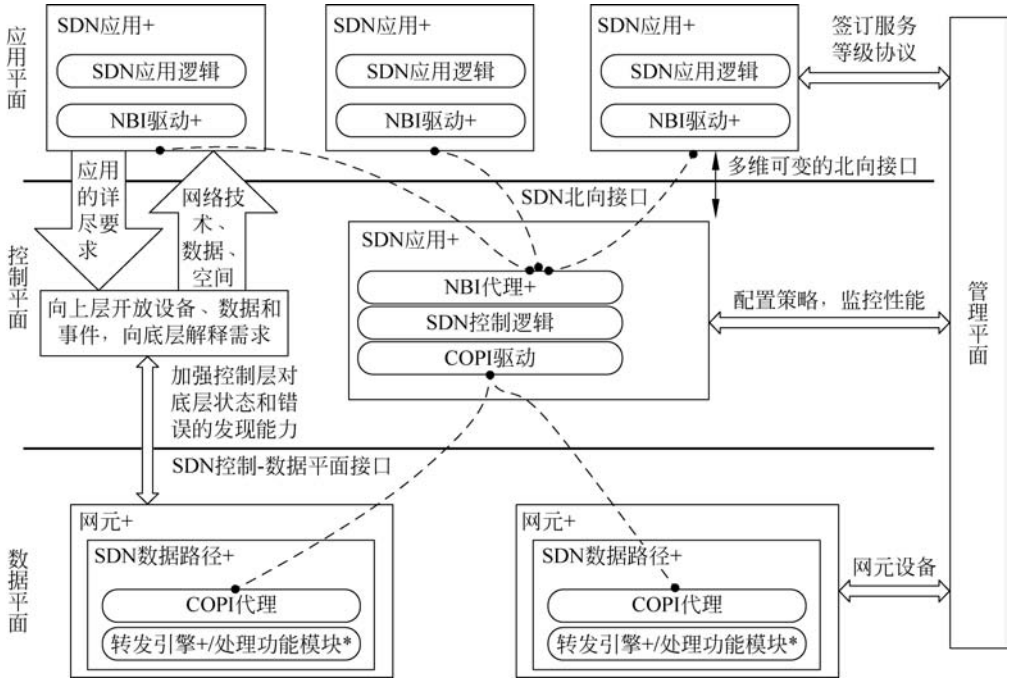

图 3.2　ONF 组织提出的 SDN 系统架构

1. 数据平面

数据平面由若干网元构成，每个网元可以包含一个或多个 SDN Datapath，是一个被管理资源在逻辑上的抽象集合。每个 SDN Datapath 是一个逻辑上的网络设备，它没有控制功能，只是单纯地用来转发和处理数据，它在逻辑上代表全部或部分物理资源，包括与转发相关的各类计算、存储、网络功能等虚拟化资源。

2. 控制平面

SDN 控制器是一个逻辑上集中的实体，它主要承担两个任务：一是将 SDN 应用层请求转换到 SDN Datapath；二是为 SDN 应用提供底层网络的抽象模型。一个 SDN 控制器包含北向接口代理、SDN 控制逻辑及控制数据平面接口驱动 3 部

分。SDN 控制器只要求逻辑上完整，因此它可以由多个控制器实例协同组成，也可以是层级式的控制器集群。从地理位置上讲，既可以所有控制器实例在同一位置，也可以多个实例分散在不同位置。

3. 应用平面

应用平面由若干 SDN 应用构成，SDN 应用是用户关注的应用程序，它可以通过北向接口与 SDN 控制器进行交互，即这些应用能够通过可编程方式把需要的网络行为提交给控制器。一个 SDN 应用可以包含多个北向接口驱动，同时 SDN 应用也可以对本身的功能进行抽象、封装，来对外提供北向代理接口，封装后的接口成为更高级的北向接口。

4. 管理平面

管理平面主要负责一系列静态的工作，这些工作比较适合在应用平面、控制平面、数据平面外实现。例如，进行网元初始配置、指定 SDN Datapath 控制器、定义 SDN 控制器及 SDN 应用的控制范围等。

SDN 系统架构采用集中式的控制平面（通常是控制器）和分布式的转发平面，两个平面相互分离，控制器位于上层应用与物理设备之间，控制器首先负责把网络中的各种功能进行抽象，建立具体的操作模型，并向上提供编程接口，上层应用着重根据业务需求通过控制器与物理设备进行交互，网络中的设备通过控制器向应用平面传递信息。控制器利用控制-转发通信接口对转发平面上的网络设备进行集中控制，并向上提供灵活的可编程能力，为网络提供了可编程性和多租户支持，从而实现新型服务的快速部署，极大地提高了网络的灵活性和可扩展性。

在 SDN 架构中，控制平面的控制器通过控制-转发通信接口对网络设备进行集中控制，这部分控制信令的流量发生在控制器与网络设备之间，独立于终端之间通信产生的数据流量，网络设备通过接收控制信令生成转发表，并决定数据流量的

处理，不再需要复杂的分布式网络协议来决策数据转发。

边缘计算的基础是连接性。所连接物理对象的多样性及应用场景的多样性，需要边缘计算具备丰富的连接功能，如各种网络接口、网络协议、网络拓扑、网络部署与配置、网络管理与维护。连接性需要充分借鉴吸收网络领域的先进研究成果，如实时网络（Time Sensitive Network，TSN）、SDN、NFV、Network as a Service、WLAN、NB-IoT、5G 等，同时还要考虑与现有各种工业总线的互联互通。

其中，将 SDN 应用于边缘计算的独特价值如下。

1）支持海量连接

支持百万级海量网络设备的接入与灵活扩展，能够集成和适配多厂商网络设备的管理。

2）模型驱动的策略自动化

提供灵活的网络自动化与管理框架，能够将基础设施和业务发放功能服务化，实现智能资产、智能网关、智能系统的即插即用，大大降低对网络管理人员的技能要求。

3）端到端的服务保障

对端到端的 GRE、L2TP、IPSec、Vxlan 等隧道服务进行业务发放，优化 QoS 调度，满足端到端带宽、时延等关键需求，实现边缘与云的业务协同。

4）架构开放

将集中的网络控制及网络状态信息开放给智能应用，应用可以灵活快速地驱动网络资源的调度。

3.1.3　内容交互/分发网络

内容分发网络（CDN）本质上是优化资源的存储位置，把一些内容进行分散、分拆、缓存，从而为用户提供高质量的、就近的、低延迟的服务，它依靠部署在各地的

边缘服务器，通过中心平台的负载均衡、分发、调度等功能模块，使用户就近获取所需内容，降低网络拥挤，提高用户访问响应速度和命中率。CDN 的关键技术主要有内容存储和分发技术。

CDN 与边缘计算技术在理念上有很大交集，主要作为内容资源方的分发平台存在。从技术发展的角度看，CDN 不能完全依托云计算，原因在于云计算是把大量 IT 基础设施集中到一起，而不是分布在各地，从目前看来，把过多的服务器集中部署到某一地方，形成数据中心、云计算中心，所有的通信网络过度集中在一起，从技术上就不能称为 CDN。CDN 本质上是一种靠近用户侧、需求侧的内容边缘分发网络，根据实时请求，选择最佳的路径及 IT 资源为用户提供内容分发服务。

随着用户规模的继续攀升，未来的 CDN 会需要更多的边缘设备，以为其提供高可靠的连接、低时延的服务。从目前看，CDN 的定位一直是通信网的辅助网络，随着通信量的提升，将有机会成为信息通信的基础设施，因此，边缘计算作为 CDN 发力的抓手，随着技术的不断发展，性能带来的优越性，可通过大规模部署边缘节点、边缘网络、边缘存储、边缘缓存等资源，在资源的供应侧提供有力的资源储备和性能优化条件，以增强用户的体验。

CDN 技术的基本原理是广泛采用各种缓存服务器，将这些缓存服务器分布到用户访问相对集中的地区或网络中，在用户访问网站时，利用全局负载技术将用户的访问指向距离最近的、工作正常的缓存服务器上，由缓存服务器直接响应用户请求。

CDN 有哪些好处？采用 CDN 技术最大的好处就是加速了网站的访问，使用户与内容之间的物理距离缩短，用户的等待时间也得以缩短，而且，分发至不同线路的缓存服务器也让跨运营商之间的访问得以加速。例如中国移动手机用户访问中国电信网络的内容源，可以通过在中国移动架设的 CDN 服务器进行加速，加速效果非常明显。此外，CDN 还有安全方面的好处。内容进行分发后，源服务器的 IP 被隐藏，其受到攻击的概率会大幅下降，而且，当某个服务器发生故障时，系统

会调用邻近的正常的服务器继续提供服务，避免对用户造成影响。正因为 CDN 的好处很多，所以，目前所有主流的互联网服务提供商都采用了 CDN 技术，所有的云服务提供商也都提供了 CDN 服务（价格也不算贵，按流量计费）。

CDN 强调内容的备份和缓存，而边缘计算的基本思想则是功能缓存（Function Cache，FC），这实际上借鉴了 CDN 的基本思想，所以 CDN 是边缘计算最初的原型。

3.1.4　任务单元和微数据中心

Cloudlet 是 2013 年 Carnegie Mellon University（卡内基梅隆大学）提出的概念，Cloudlet 是一个连接到互联网的可信且资源丰富的主机或机群，它部署在网络边缘与互联网连接并可以被周围的移动设备所访问，为设备提供服务。Cloudlet 将原先移动计算的两层架构“移动设备-云”变为三层架构“移动设备-Cloudlet-云”，Cloudlet 也可以像云一样为用户提供服务，所以它又被称为“小云”（Data Center in a Box，DCB）。

Cloudlet 主要用来支持移动云计算中计算密集型与延迟敏感型的应用（如人脸识别），当附近有 Cloudlet 可以使用时，计算资源匮乏的移动设备可以将计算密集型任务卸载到 Cloudlet，移动设备既不需要处理计算密集型任务，又能保证任务的快速完成，因此，Cloudlet 的主要功能就是配置并运行部分来自移动设备或云端的计算任务。Cloudlet 的软件栈分为三层：第一层由操作系统和 Cache 组成，其中 Cache 主要对云中的数据进行缓存；第二层是虚拟化层，将资源虚拟化，并通过统一的平台 OpenStack++ 对资源进行管理；第三层是虚拟机实例，移动设备卸载的应用都在虚拟机中运行，这样可以弥补移动设备与 Cloudlet 应用运行环境（操作系统、函数库等）的差异。一个在移动设备与 Cloudlet 之间的计算迁移过程如图 3.3 所示。

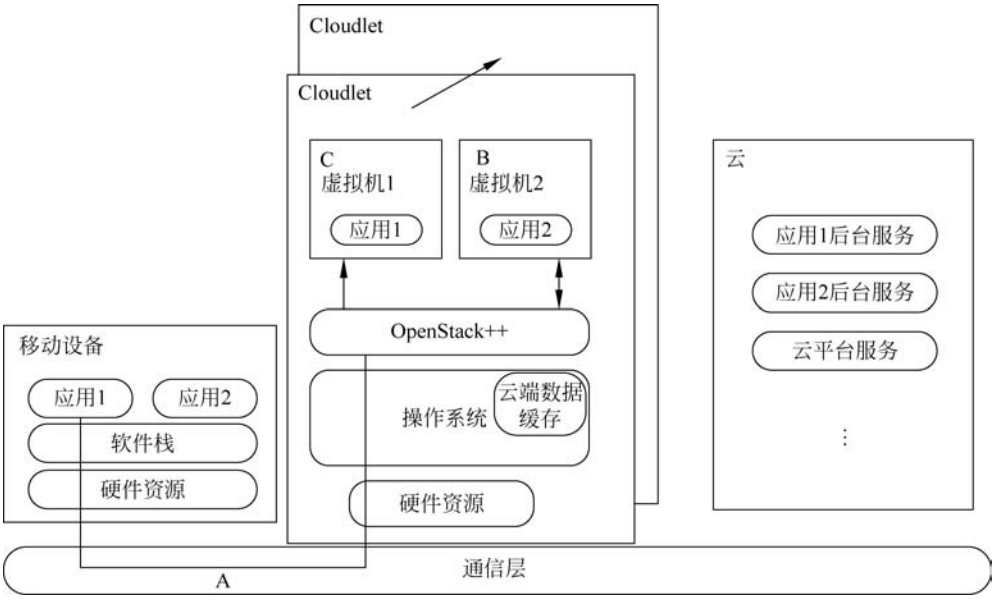

图 3.3　移动设备与 Cloudlet 之间的计算迁移过程

首先，从图 3.3 中可以看出，Cloudlet 使用一个单独虚拟机来为每个移动设备卸载的任务提供运行环境，这样做虽然会在一定程度上造成计算资源的浪费，却有很多优点，主要包括将 Cloudlet 的软件环境与运行应用的环境隔离，二者之间不会相互影响，增强稳定性；因其动态资源分配，随用随取，对应用软件的开发几乎没有任何约束，可弥补多种多样的移动设备与 Cloudlet 应用运行环境（如操作系统、函数库等）的差异。这些优势有助于实现 Cloudlet 的自我管理，大大简化云端对 Cloudlet 的管理复杂度。

与云不同，Cloudlet 部署在网络边缘，主要面向的是移动设备，并且只服务附近的用户，所以在进行计算迁移时需要支持移动性，移动的设备可以自动选择使用最近的 Cloudlet，如图 3.2 所示，Cloudlet 对应用移动性的支持主要依赖以下三个关键步骤。

（1）Cloudlet 资源发现（Cloudlet discovery）：Cloudlet 在地理位置上广泛分布，同一时刻，周围可能有多个可用的 Cloudlet 服务器，所以资源发现机制可以快

速发现周围可用的 Cloudlet，并选择最合适的作为卸载任务的载体。

（2）虚拟机快速配置（VM rapid provisioning）：移动性导致用户不可能提前在 Cloudlet 上准备好配置相关的资源（如虚拟机镜像），通过网络上传这些资源需要漫长的等待，严重影响可用性，所以需要一种虚拟机快速配置技术在选定的 Cloudlet 上启动运行应用的虚拟机，并配置运行环境。

（3）资源切换（VM handoff）：Cloudlet 的特点是贴近用户，但在很多情况下，Cloudlet 无法跟随用户一起移动。虽然保持网络的连接就可以一直维持服务，但是移动设备与 Cloudlet 的网络距离可能会增加，网络的带宽、时延等因素也可能会随之恶化，因此一个更好的解决方案是将任务从原来的 Cloudlet 无缝地切换到附近的 Cloudlet 上执行，资源切换机制就实现这个功能。

动态虚拟机合成（Dynamic VM Synthesis）是 Cloudlet 支持移动性的关键技术，可以将虚拟机镜像拆分为基底（Base）与覆盖层（Overlay），基底与覆盖层也可以重新组合为虚拟机镜像。基底包含虚拟机的操作系统、函数库等基础软件，这一部分在虚拟机镜像之间都是重复的，并且占用空间大，而覆盖层是一个很小的二进制增量文件，只包含用户在原始虚拟机上的一些定制信息，占用空间小。在虚拟机配置和资源切换时，使用动态虚拟机合成技术可以只传输轻量的覆盖层，减少了数据传输量，加快了虚拟机配置和资源切换的速度，保证了应用在 Cloudlet 中可以得到及时的资源供给。

此外，微软研究院提出了微型数据中心的概念，作为当今超大规模云数据中心的延伸。与 Cloudlet 类似，微型数据中心也设计用于满足需要较低延迟或在电池寿命或计算方面面临限制的应用程序的需求。微型数据中心装在一个机箱中，是一个独立的、安全的计算环境，包括运行客户应用程序所需的所有计算、存储和网络设备。微型数据中心的功率范围为 1～100kW，以满足考虑 IT 负载的可扩展性和延迟要求，并且如果将来需要更多容量，也可以扩展。

微数据中心（MDC）是硬件、软件和电缆的多功能组合，用作端到端的网络集

线器，类似于电信室或网络室，但比典型的企业数据中心规模小得多。它的目的是提供公司和工业网络联系的紧密性。设计良好的 MDC 可以保护控制和信息数据的完整性、可用性和机密性。它促进了从工厂到企业的连接，提供了对制造过程的更大可视性，以识别问题、优化过程和规划未来。

MDC 的定义特征是它在单个空间中容纳完整的数据中心基础设施——电子设备、贴片场、电缆管理、接地/粘接、电源和铜/光纤电缆——但其大小要满足制造环境的需求。MDC 是一个新概念，代表业务管理的下一个阶段，是分支机构的核心解决方案，设计理念为了简化、开放、“熄灯管理（Lights Out Management，LOM）”，集设备集成、计算与存储、路由与交换、安全与 VPN、广域网与 WLAN 接入、VoIP 等企业常用应用、打印与文件服务器等功能于一体。所有这些都是预先集成的，因此交付和部署变得非常快速和方便。它有烟雾传感器、温度和湿度传感器、漏水传感器、智能 PDU 和圆顶摄像头等环境传感器。MDC 还可以采取没有服务器的网络集线器形式，其存在主要是为了将电缆和交换机连接在一起。对于大型制造联合体或远程位置，MDC 可以充当数据收集节点，将制造数据向上传递给企业（存储和转发）。最后，MDC 还可以为虚拟机（VM）系统提供高可靠和高效率的服务器。

微数据中心通过“熄灯管理”进行管理，它允许用户或系统管理员远程管理服务器。该程序还允许不仅便于监测，还便于执行的操作，如重启、关机、故障排除、报警设置、风扇速度控制和操作系统重新安装。

3.2 边缘计算系统

3.2.1 Apache Edgent

Apache Edgent 是一个开源的编程模型，它可以被嵌入边缘设备，用于提供对

连续数据流的本地实时分析。Apache Edgent 解决的问题，是如何对来自边缘设备的数据进行高效的分析处理。为加速边缘计算应用在数据分析处理上的开发过程，Apache Edgent 提供了一个开发模型和一套 API 用于实现数据的整个分析处理流程。基于 Java 或安卓的开发环境，Apache Edgent 应用的开发模型如图 3.4 所示。

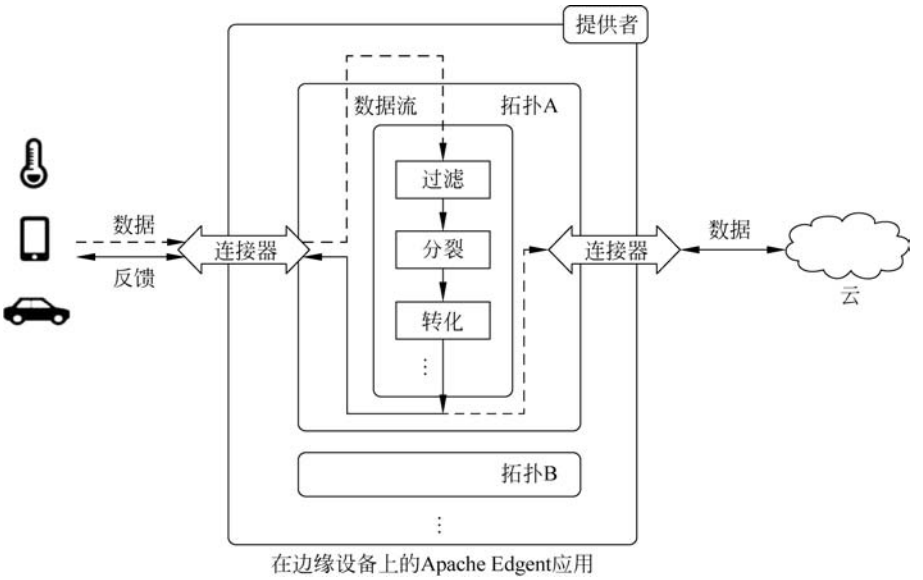

图 3.4　Apache Edgent 应用的开发模型

该模型由提供者、拓扑、数据流、数据流的分析处理、后端系统 5 个组件组成。

(1) 提供者：一个提供者对象包含了有关 Apache Edgent 应用程序的运行方式和位置信息，并具有创建和执行拓扑的功能。

(2) 拓扑：一个容器，描述了数据流的来源和如何更改数据流的数据。数据的输入、处理和导出至云的过程都记录在拓扑中。

(3) 数据流：Apache Edgent 提供了多种连接器以不同方式接入数据源，例如支持消息队列遥测传输(MQTT)、超文本传输协议(HTTP)和串口协议等，用户还可以添加自定义代码以控制传感器或设备的数据输入。此外，Apache Edgent 的数

据不局限于来自真实传感器或者设备的数据，还支持文本文件和系统日志等。

(4) 数据流的分析处理：Apache Edgent 提供一系列功能性的 API 以实现对数据流的过滤、分裂、变换和聚合等操作。

(5) 后端系统：由于边缘设备的计算资源稀缺，Edgent 应用程序无法支撑复杂的分析任务。用户可以使用连接器，通过 MQTT 和 Apache Kafka 方式连接至后端系统，或者连接至 IBM Watson IoT 平台进一步对数据做处理。

Apache Edgent 应用可部署于运行 Java 虚拟机的边缘设备中，实时分析来自传感器和设备的数据，减少了上传至后端系统(如云数据中心)的数据量，并降低了传输成本。Apache Edgent 的主要系统特点是提供了一套丰富的数据处理 API，切合物联网应用中数据处理的实际需求，降低应用的开发难度并加速开发过程。Apache Edgent 的主要应用领域是物联网，此外，它还可以被用于分析日志、文本等类型的数据，例如嵌入服务器软件中用以实时分析错误的日志。

3.2.2 OpenStack

OpenStack 是一个由美国宇航局 NASA 与 Rackspace 公司共同开发的云计算平台项目，并且通过 Apache 许可证授权开放源码。它可以帮助服务商和企业实现云基础架构服务。下面是 OpenStack 官方给出的定义：OpenStack 是一个可以管理整个数据中心里大量资源池的云操作系统，包括计算、存储及网络资源。管理员可以通过管理台管理整个系统，并可以通过 Web 接口为用户划定资源。

OpenStack 是一个云操作系统，它控制整个数据中心的大型计算、存储和网络资源池，所有这些都是通过具有通用身份验证机制的 API 和 Dashboard 管理的。通过 Dashboard，让管理员可以控制、赋予他们的用户去提供资源的权限(能够通过 Dashboard 控制整个 OpenStack 云计算平台的运作)。除了标准的基础设施(服务功能)外，其他组件还提供编制、故障管理和其他服务之间的服务管理，以确保用

户应用程序的高可用性。

OpenStack覆盖了网络、虚拟化、操作系统、服务器等各方面。它是一个正在开发中的云计算平台项目，根据成熟及重要程度的不同，被分解成核心项目、孵化项目，以及支持项目和相关项目。每个项目都有自己的委员会和项目技术主管，而且每个项目都不是一成不变的，孵化项目可以根据发展的成熟度和重要性，转变为核心项目。下面列出了10个核心项目(OpenStack服务)。

(1) 计算(Compute)：Nova。一套控制器，用于为单个用户或使用群组管理虚拟机实例的整个生命周期，根据用户需求来提供虚拟服务。负责虚拟机创建、开机、关机、挂起、暂停、调整、迁移、重启、销毁等操作，配置CPU、内存等信息规格。自Austin版本集成到项目中。

(2) 对象存储(Object Storage，OS)：Swift。一套用于在大规模可扩展系统中通过内置冗余及高容错机制实现对象存储的系统，允许进行存储或者检索文件。可为Glance提供镜像存储，为Cinder提供卷备份服务。自Austin版本集成到项目中。

(3) 镜像服务(Image Service，IS)：Glance。一套虚拟机镜像查找及检索系统，支持多种虚拟机镜像格式(AKI、AMI、ARI、ISO、QCOW2、Raw、VDI、VHD、VMDK)，有创建上传镜像、删除镜像、编辑镜像基本信息的功能。自Bexar版本集成到项目中。

(4) 身份服务(Identity Service，IS)：Keystone。为OpenStack其他服务提供身份验证、服务规则和服务令牌的功能，管理Domains、Projects、Users、Groups、Roles。自Essex版本集成到项目中。

(5) 网络 & 地址管理(Network)：Neutron。提供云计算的网络虚拟化技术，为OpenStack其他服务提供网络连接服务。

为用户提供接口，可以定义Network、Subnet、Router，配置DHCP、DNS、负载均衡、L3服务，网络支持GRE、VLAN。插件架构支持许多主流的网络厂家和技

术，如 OpenvSwitch。自 Folsom 版本集成到项目中。

（6）块存储（Block Storage，BS）：Cinder。为运行实例提供稳定的数据块存储服务，它的插件驱动架构有利于块设备的创建和管理，如创建卷、删除卷，在实例上挂载和卸载卷。自 Folsom 版本集成到项目中。

（7）UI 界面（Dashboard）：Horizon。OpenStack 中各种服务的 Web 管理门户，用于简化用户对服务的操作，例如启动实例、分配 IP 地址、配置访问控制等。自 Essex 版本集成到项目中。

（8）测量（Metering）：Ceilometer。像漏斗一样，能把 OpenStack 内部发生的绝大多数事件收集起来，然后为计费和监控及其他服务提供数据支撑。自 Havana 版本集成到项目中。

（9）部署编排（Orchestration）：Heat。提供了一种通过模板定义的协同部署方式，实现云基础设施软件运行环境（计算、存储和网络资源）的自动化部署。自 Havana 版本集成到项目中。

（10）数据库服务（Database Service，DS）：Trove。为用户在 OpenStack 的环境提供可扩展、可靠的关系和非关系数据库引擎服务。自 Icehouse 版本集成到项目中。

3.2.3 EdgeX Foundry

EdgeX Foundry 是一个面向工业物联网边缘计算开发的标准化互操作性框架，部署于路由器和交换机等边缘设备上，为各种传感器、设备或其他物联网元器件提供即插即用功能并管理它们，进而收集和分析它们的数据，或者导出至边缘计算应用或云计算中心做进一步处理。它针对的问题是物联网元器件的互操作性问题。目前，具有大量设备的物联网产生大量数据，迫切需要结合边缘计算的应用，但物联网的软硬件和接入方式的多样性给数据接入功能带来困难，影响了边缘计

算应用的部署。EdgeX Foundry 的主旨是简化和标准化工业物联网边缘计算的架构，创建一个围绕互操作性组件的生态系统。

EdgeX Foundry 主要具有减少网络延迟、网络流量，提高安全性等优势，对于不同 IoT 节点软件，更容易与云端的通用架构连接并且互操作性强。

EdgeX Foundry 在物联网方面有很广泛的应用：终端客户可以快速、轻松地部署 IoT Edge 解决方案，灵活地适应不断变化的业务需求；硬件制造商可以通过互操作性实现更强大的安全系统，加快扩张速度；软件供应商可与第三方应用程序实现互操作，无须重新创建连接；系统集成商可通过即插即用的方式加快产业速度。越来越多的物联网应用企业，重视并强调前端装置的端点运算和分析能力，希望在前端集中处理更多的运算分析工作，加快数据分析和处理速度，以解决不同供应商端点运算装置、应用和服务件的互操作性。

EdgeX Foundry 是一系列松耦合、开源的微服务集合。这些微服务分为 4 个层次：设备服务层、核心服务层、支持服务层和导出服务层。以核心服务层为界，整个服务架构可以分为“北侧”和“南侧”。“北侧”包含云计算中心和与云计算中心通信的网络，还包含支持服务层与导出服务层。“南侧”包含物理领域中的全部物联网对象及与它们直接通信的网络边缘。EdgeX Foundry 还包含了两个贯穿整个框架且为各层提供服务的基础服务层——安全和系统管理。安全服务中的元器件为 EdgeX Foundry 中的各类设备提供保护，支持认证、授权和计费（Authentication、Authorization、Accounting，AAA）访问控制、高级加密标准（Advanced Encryption Standard，AES）数据加密、证书认证、超文本传输安全协议（HTTPS）等保护方法。系统管理工具提供了监控 EdgeX Foundry 运行情况的能力，在未来可能会提供服务配置、为管理平台提供信息等能力。

EdgeX Foundry 并不仅是一个标准，还是一个极具可操作性的开源平台。图 3.5 展示了 EdgeX Foundry 的架构。架构的设计遵循了以下原则：架构应与平台无关，能够与多类别操作系统进行对接；架构需具有高灵活性，其中的任意部分应

该都可以进行升级、替换或扩充；架构需具有存储和转发的功能，支持离线运行，并保证计算能力能够靠近边缘。图 3.5 的最下方是“南侧”，指的是所有物联网元器件，以及与这些设备、传感器或其他物联网元器件直接通信的边缘网络。图 3.5 的最上方是“北侧”，指的是云计算中心或企业系统，以及与云中心通信的网络部分。南侧是数据产生源，而北侧收集来自南侧的数据，并对数据进行存储、聚合和分析，如图 3.5 所示，EdgeX Foundry 位于南侧和北侧两者之间，由一系列微服务组成，而这些微服务可以被分成 4 个服务层和两个底层增强系统服务。微服务之间通过一套通用的 RESTful 应用程序编程接口（API）进行通信。

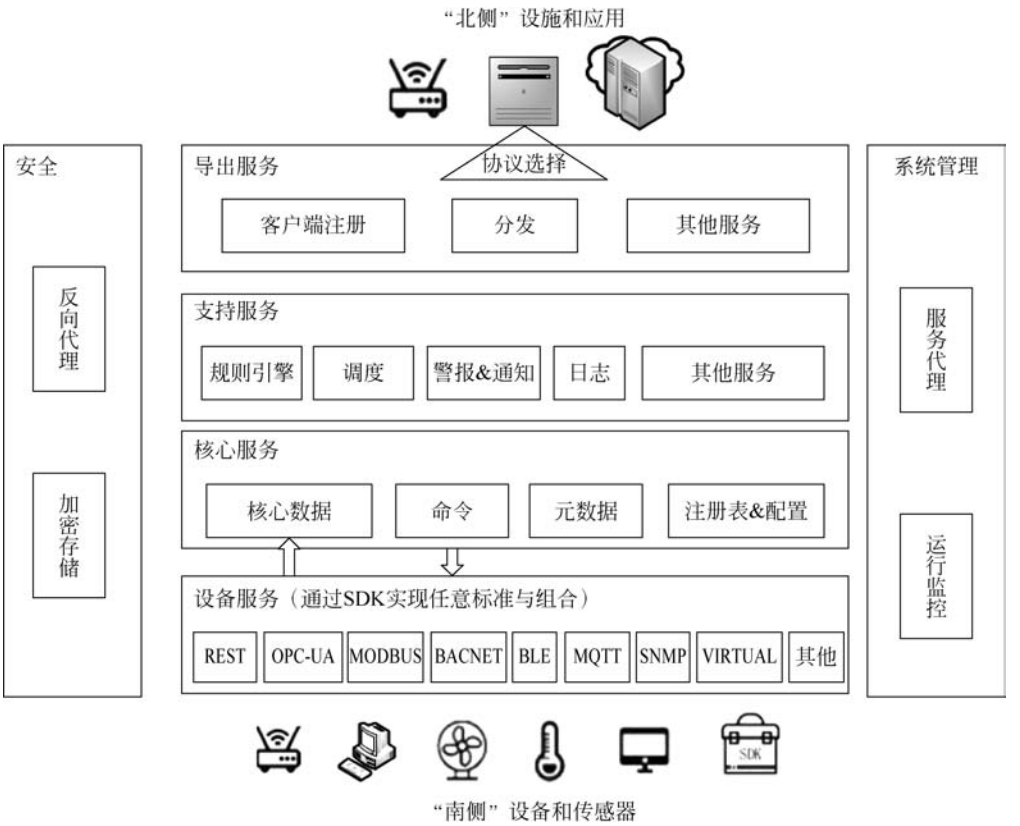

图 3.5　EdgeX Foundry 的架构

1. 设备服务层

设备服务层(Device Service,DS)是与南侧设备或物联网对象交互的边缘连接器,设备服务可以同时服务于一个或多个设备(传感器、制动器等)。DS管理的“设备”不是简单的单一物理设备,它可能是EdgeX Foundry的另一个网关(及该网关的所有设备)、设备管理器或设备聚合器/设备集合。设备服务层的微服务通过每个物联网对象本身的协议与设备、传感器、执行器和其他物联网对象进行通信。DS层将由IoT对象生成和传递的数据转换为常见的EdgeX Foundry数据结构,并将转换后的数据发送到Core Service Layer及EdgeX Foundry其他层的其他微服务。

2. 核心服务层

核心服务层介于北侧与南侧之间,这里的北侧即上文所述的信息域,南侧即上文所述的物理域。核心服务层非常简单,却是EdgeX Foundry框架内非常重要的一环。核心服务层主要由以下组件组成。

(1) 核心数据:提供持久性存储库和从南侧对象收集数据的相关管理服务,存储和管理来自南侧设备的数据。

(2) 命令:负责将云计算中心的需求驱动至设备端,并提供命令的缓存和管理服务。

(3) 元数据:提供配置新设备并将它们与其拥有的设备服务配对功能。

(4) 相关配置:存储设备服务的相关信息,为其他EdgeX Foundry微服务提供关于EdgeX Foundry内相关服务的信息,包括微服务配置属性。

3. 支持服务层

支持服务层包含各种微服务,该层微服务主要提供边缘分析服务和智能分析

服务，并可以为框架本身提供日志记录、调度、规则引擎和数据清理等支持功能。以规则引擎微服务为例，允许用户设定一些规则，当检测到数据满足规则要求时，将触发一个特定的操作。例如规则引擎可监测控制温度传感器，当检测到温度低于25℃时，触发对空调的关闭操作。

4. 导出服务层

导出服务层用于将数据传输至云计算中心，也负责提供网关客户端注册等功能，并对与云计算中心传递的数据格式和规则进行实现，由客户端注册和分发等微服务组件组成。前者记录已注册的后端系统的相关信息，后者将对应数据从核心服务层导出至指定客户端。它保证了 EdgeX Foundry 的独立运行，在其不与云计算中心连接时，仍可以对边缘设备的数据进行收集。

5. 系统管理和安全服务

系统管理服务提供安装、升级、启动、停止和监测控制 EdgeX Foundry 微服务的功能。安全服务用以保障来自设备的数据和对设备的操作安全。

最新版本的 EdgeX Foundry 没有为用户自定义应用提供计算框架，用户可以将应用部署在网络边缘，将该应用注册为导出客户端，进而将来自设备的数据导出至应用来处理。EdgeX Foundry 的设计满足硬件和操作系统无关性，并采用微服务架构。EdgeX Foundry 中的所有微服务能够以容器的形式运行于各种操作系统，并且支持动态增加或减少功能，具有可扩展性。EdgeX Foundry 的主要系统特点是为每个接入的设备提供通用的 RESTful API 以操控该设备，便于大规模地监测控制物联网设备，满足物联网应用的需求。EdgeX Foundry 的应用领域主要在工业物联网，如智能工厂、智能交通等场景，以及其他需要接入多种传感器和设备的场景。

3.2.4　Azure IoT Edge

云计算服务提供商是边缘计算的重要推动者之一，基于云边融合的理念，致力于将云服务能力拓展至网络边缘。微软公司推出了 Azure IoT Edge，并在 2018 年宣布将 Azure IoT Edge 开源。

Azure IoT Edge 是一种混合云和边缘的边缘计算框架，旨在将云功能拓展至具备计算能力的边缘设备（如路由器和交换机等）上，以获得更低的处理时延和实时反馈。Azure IoT Edge 运行于边缘设备上，但使用与云上的 Azure IoT 服务相同的编程模型，因此用户在开发应用的过程中除对计算能力的考量外，无须考虑边缘设备上部署环境的差异，还可以将在云上原有的应用迁移至边缘设备上运行。

如图 3.6 所示，Azure IoT Edge 由 IoT Edge 模块、IoT Edge 运行时和 IoT Edge 云界面组成，前两者运行在边缘设备上，后者则是一个在 Azure 云上提供服务的管理界面。

（1）IoT Edge 模块：对应于用户的边缘计算应用程序。一个模块镜像即一个 Docker 镜像，模块里包含用户的应用代码，而一个模块实例就是一个运行着对应的模块镜像的 Docker 容器。基于容器技术，IoT Edge 具备可扩展性，用户可动态添加或删除边缘计算应用。由于相同的编程模型，Azure 机器学习和 Azure 数据流分析等 Azure 云服务也可以部署到 IoT Edge 模块，此特性便于在网络边缘部署复杂的人工智能应用，加快了开发过程。

（2）IoT Edge 运行时：由 IoT Edge 中心和 IoT Edge 代理两个组件构成，前者负责通信功能，后者负责部署和管理 IoT Edge 模块，并监测控制模块的运行。IoT 中心是在 Azure 云上的消息管理中心，IoT Edge 中心与 IoT 中心连接并充当其代理。IoT Edge 中心通过 MQTT、高级消息队列协议（Advanced Message Queuing Protocol，AMQP）和 HTTPS 获取来自传感器和设备的数据，实现设备接入的功

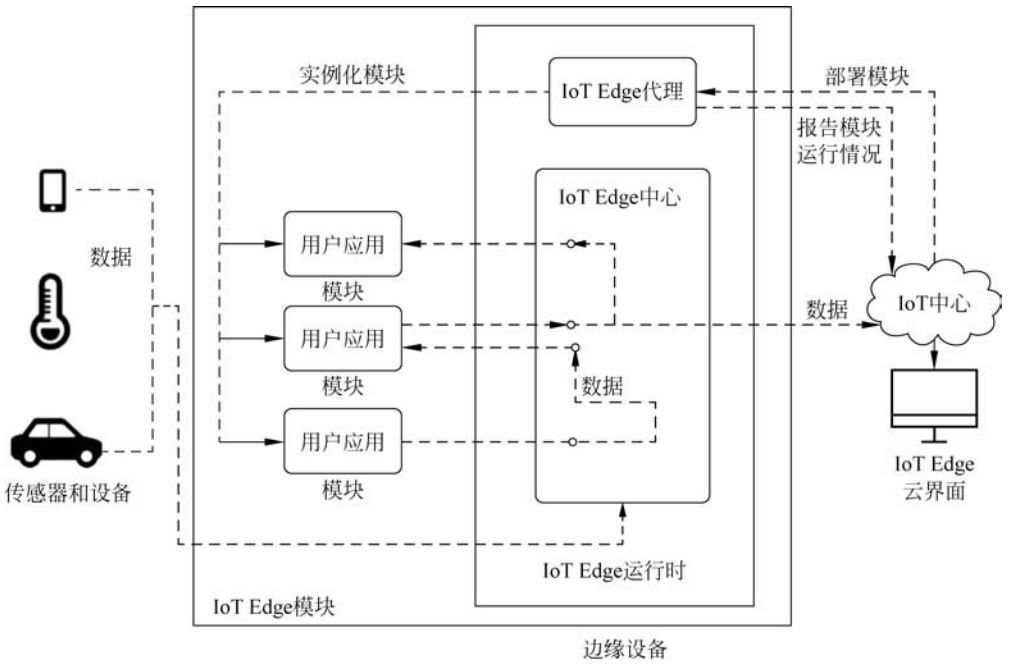

图 3.6　Azure IoT Edge 计算框架

能。此外，IoT Edge 中心作为消息中转站，连接 IoT Edge 模块之间的消息通信。IoT Edge 代理从 IoT Hub 接收 IoT Edge 模块的部署信息，实例化该模块，并保证该模块的正常运行，如对故障模块进行重启，并将各模块的运行状态报告至 IoT 中心。

（3）IoT 云界面：提供设备管理的功能。用户通过云界面进行添加设备、部署应用和监测控制设备等操作，为用户大规模部署边缘计算应用提供了方便。

Azure IoT Edge 的主要系统特点是强大的 Azure 云服务的支持，尤其是人工智能和数据分析服务的支持。Azure IoT Edge 具有广阔的应用领域，除了物联网场景，原有在云上运行的应用也可以根据需求迁移至网络边缘上运行。目前 Azure IoT Edge 已有智能工厂、智能灌溉系统和无人机管理系统等使用案例。

3.2.5　AWS Greengrass

AWS IoT Greengrass是将云功能扩展到本地设备的软件。这使设备可以更靠近信息源来收集和分析数据，自主响应本地事件，同时在本地网络上彼此安全地通信。本地设备还可以与AWS IoT Core安全通信并将IoT数据导出到AWS云。AWS IoT Greengrass开发人员可以使用AWS Lambda函数和预构建的连接器来创建无服务器应用程序，这些应用程序将部署到设备上以进行本地执行。

作为全球最大的云服务提供商，亚马逊公司专门针对边缘计算和物联网推出了AWS Greengrass软件，旨在减少设备和数据处理层之间的时延、减少将大量数据传输到云端的带宽成本，并在本地保留敏感数据以实现合规和安全性。AWS Greengrass支持互联设备的本地计算、消息收发、数据缓存和同步功能。借助AWS Greengrass，开发人员可以使用亚马逊的AWS Lambda服务和AWS IoT平台，跨AWS云和本地设备无缝运行物联网应用程序接口，确保物联网设备快速响应本地事件，运行时采用间歇性连接，并最大程度地降低将物联网数据传输到云的成本。

AWS Greengrass主要由以下3种要素构成。

(1) 软件分发AWS Greengrass核心软件、AWS Greengrass核心软件开发工具包。

(2) 云服务AWS Greengrass API。

(3) 当功能Lambda运行时：设备状态(Thing Shadows Implementation，TSI)、消息管理器、组管理、发现服务。

更多关于AWS Greengrass的功能、如何安装和配置AWS Greengrass、如何启动和运行AWS Greengrass、AWS Greengrass安全性等相关问题，读者可查阅亚马逊AWS Greengrass官方网站。

3.3 本章小结

边缘计算的核心思想就是将云计算能力延伸到网络边缘，本章主要介绍了边缘计算的关键技术，并对新兴的边缘计算系统做了概要介绍。