DOI：10.13955/j.yzyj.2021.03.03.06

Java 微服务架构在邮政移动互联网应用研发设计中的思考与实践

查嵩炜

（中国邮政集团有限公司上海市信息技术局，上海　201100）

摘　要：通过对比分析单体式应用存在的缺陷以及在相同场景下微服务架构应用的优势，介绍了在 Java 体系中以 Spring Boot、Spring Cloud 为代表的微服务框架，并以上海邮政自主研发的个性化邮票面向移动互联网的业务系统中微服务架构实践为例，探讨了在处理微服务架构时软件系统服务拆分的思考，采用领域驱动的设计思路以及微服务基础设施的使用及部署。

关键词：微服务架构；单体式架构；服务拆分；领域驱动

中图分类号：F61　　**文献标识码：**A

随着上海邮政市场化程度日益提高以及互联网尤其是移动互联网的快速发展，越来越多的业务应用被要求在客户的手机和微信端运行。而采用传统的业务系统架构和思想进行移动端应用的开发越来越困难。同时，微服务架构作为一项新技术得到了国内外众多软件开发商的重视，已经成为互联网应用开发设计的潮流。上海邮政也对微服务架构的技术和思想进行了探索研究和应用，从而为邮政企业的业务发展提供了更好的技术支持和选择。

1　传统应用架构

单体式应用是传统的应用架构模式，通常会把所有的业务需求实现集中在一个应用程序中，并且运行在操作系统的一个进程中。在该架构中，代码和所依赖及生成的数据甚至界面元素也是集中且杂糅在一起的，最终形成所谓烟囱式结构。

采用单体式架构开发的面向互联网的业务系统，虽然项目开始建设时相对容易，但随着开发和生产活动的进行，会逐渐出现以下问题：一是应用中各个组件、服务模块之间高度耦合，维护难度随时间迅速增加；二是随着项目的进展，软件架构随着修改迅速退化；三是无法进行持续部署，交付时间长；四是生产时无法根据实际需要对部分功能进行横向扩展；五是容错性差，程序错误或者内存的过度消耗会导致整个应用系统瘫痪；六是整个应用系统会被长期限制在某种技术上，转型困难。在建设面向互联网应用需要快速迭代的系统时，这些问题会变得不可接受，因此急需一种新的架构和思想来打破这个困局。

2　微服务架构

微服务是采用一些小的独立的应用，运行在独立的进程中，采用轻量级的通讯方式，为独立业务开发，分布式管理和部署，可以采用不同语言和数据存储。微服务体现了一种架构思想和开发风格，其本质特征可以概括为“小而专注、独立而自治”，具体体现在：应用系统由多个服务组件组成，每个组件是可以被单独替换和升级的应用程序，专注

作者简介：查嵩炜（1980 ～），男，江苏苏州人，主要从事邮政信息技术应用研究。

收稿日期：2020—12—21

于解决一组小的紧密关联的业务内容，组件间需要定义清晰的边界，从而实现松耦合和高内聚，通过RESTful等轻量级的通讯协议进行通信。组件可以由不同的语言编写并获得最佳的实现方案。整个应用系统在需要时可以方便地采用去中心化的数据组织形式和不同的存储技术。通过微服务架构，应用系统的每个组件可以独立完成开发测试、部署和扩展，从而实现谁开发、谁运营，做全生命周期的产品，不断应对业务的更迭需求。

2.1　微服务架构的优势与挑战

微服务架构的出现就是为了解决传统单体式架构带来的痛点，因此相较于单体式应用来说，微服务架构存在以下优势：一是通过服务分拆，理清服务间的边界，使每个服务小型化，代码内聚而易理解，提高了开发效率，降低了维护难度，软件架构不易退化；二是服务之间可以独立部署，使敏捷开发、持续部署成为可能；三是可以针对每个服务组建开发团队，利用好不同团队特色，易于扩大和管理开发团队；四是每个服务可以各自进行x扩展和z扩展；五是服务可以根据自己的需要部署到合适的硬件服务器上；六是可提高容错性，一个服务出现问题不会导致整个系统瘫痪；七是系统的异构可以使其不会被长期限制在某个技术上。

虽然微服务有众多优点，但使用微服务也将面临以下挑战：一是在设计过程中，要实现应用系统的y轴扩展，就需要软件设计师进行大量工作；二是由于微服务架构是分布式系统，因此需要考虑CAP定理带来的固有复杂性；三是服务间接口依赖，当修改服务时，单体式应用中只需修改好相应的模块完成整合即可发布部署，但在微服务中必须考虑相关改变对不同服务产生的影响；四是由于被分解到多个主机和多个运行程序上，增加了对运维的要求和运行成本。因此，在具体实施过程中不能为了微服务而微服务，而应使微服务成为有效理清应用、实现更高效的敏捷开发和持续部署的手段。

2.2　Java中微服务的解决方案

随着微服务架构的提出，Java中也涌现出许多微服务实现框架，而Spring Boot加Spring Cloud的方案是目前Java微服务开发中的主流形式。

Spring Boot是Spring提供的快速配置脚手架，通过简化传统J2EE开发过程中的配置过程，解决了系统部署繁琐等问题。Spring Cloud则是一系列微服务基础设施框架的集合，通过Spring Boot再封装屏蔽掉复杂的配置和实现原理，最终给开发者留出一套简单易懂、易部署和易维护的系统开发工具包。因此，采用Spring Boot和Spring Cloud组合成为实现快速构建、快速部署和启动基于微服务架构的应用程序的重要手段。

3　微服务架构在上海邮政的应用实践

微服务架构已经在上海邮政开发的多个面向互联网、移动手机的应用中进行了实践，个性化邮票定制售卖系统是上海邮政使用微信小程序为用户提供定制化邮票产品的渠道。用户可以选择不同的个性化邮票产品模板，购买印有自己上传照片的邮票产品。

在传统的设计思路中，项目设计人员一般会建立一一对应的应用系统。每个应用系统都会按照对应的业务需求实现相应的功能逻辑，但是个性化邮票定制售卖系统如果采用单体式应用系统架构，除了上文提到的问题外，还面临以下问题：一是用户的互联网业务和管理端的业务功能是混合在一起的，存在潜在的安全风险；二是部分功能在多个系统中重复出现，比如微信用户信息管理、支付管理，由于本身体量较小，对单体式应用设计而言似乎还不足以单独列出来作为独立系统进行开发，但放置在各个应用系统中又形成了对应功能多头管理，不符合软件系统的单一职责的原则。上海邮政将个性化邮票定制售卖系统中面向用户的移动互联网实现部分单独拉出，采用微服务理念进行构建。

3.1　前后端彻底分离

上海邮政实现了前端交互页面的静态化和后端服务接口的彻底分离，即开发分离和部署分离。通过分离，前端界面可以由专门的前端开发人员进行设计和开发，实现开发工作专业化、项目管理工程化。前端页面的服务独立部署，减轻了一切由后端服务提供的系统压力，同时也方便在部署时采用高效的静态资源服务器和内容分发网络（CDN），为应用系统提供更好的浏览效率。

3.2　建立和使用微服务的基础设施

上海邮政使用Spring Cloud提供的微服务框架和基础设施。

3.2.1　部署服务注册和发现中心

服务注册和发现中心是微服务提供者和消费者之间的桥梁，用来解决微服务分布式系统中服务提供者 IP 地址不固定、服务消费者无法通过固定模式获取提供者位置的问题。

Spring Cloud 提供了 Netflix Eureka、consul、zookeeper、etcd 等作为服务注册和发现中心的功能，上海邮政最终选用了 Netflix Eureka 方案。它是一款轻量级的基于 RESTful 的服务注册发现的框架（见图 1），无需数据库支持，并且支持集群部署，被广泛应用于亚马逊的 AWS 云中。

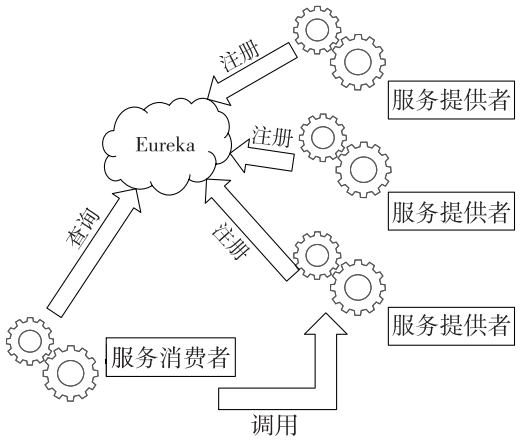

图 1　服务注册与发现中心 Eureka

服务提供者和服务消费者都连接到 Eureka；服务提供者将自身信息注册到 Eureka 中；服务消费者调用某个服务时到 Eureka 中查询相应服务的位置（IP 域名和端口）；服务消费者获得服务的列表（服务可能是多个）；服务消费者通过自己的负载均衡实现对服务提供者进行调用。

在实践中，考虑到性能和灾难恢复情况，上海邮政在生产系统中部署了多个 Eureka 实例，分别运行在各自的主机上。

3.2.2　使用服务网关

服务网关位于用户接入和服务之间，一端直接面向用户接入，可以实现多渠道的接入手段，另一端连接着微服务，作为微服务的消费者，实现负载均衡和流量控制，遇到灾难时进行熔断和服务降级。服务网关自身可以完成除业务逻辑外的用户身份认证和请求合法性校验等工作（见图 2）。

上海邮政结合使用 Spring Cloud 提供的 Ribbon 和 Feign 自行编号了服务网关，其中 Ribbon 提供了负载均衡，防止调用已经停止响应的服务，而 Feign 在开发上也帮助上海邮政无需在意调用远程服务的细节，可以像调用一个本地方法的形式，调用一个服务提供者提供的功能。

3.2.3　配置服务熔断与降级

作为互联网应用，实践中随时可能面临超出预期的业务流量导致某个服务无法响应，或者某个服务发生故障，使得依赖该服务的上一级服务超时等待而发生故障，如此传导引发“雪崩”效应，导致整个业务系统瘫痪。上海邮政在服务消费者上使用了熔断器的设计，通过熔断技术防止发生这类“雪崩”现象。Hystrix 是 Spring Cloud 提供的熔断器，可以实现服务的限流，灾难时熔断、优雅的服务降级以及自动恢复。在熔断后可以人为设置，为用户提供友好的界面提示信息，而不是让用户看到界面卡顿或者长时间等待却获得不知所云的错误信息。

3.2.4　使用配置中心

每个微服务实例都有自己的配置文件保存着配置信息，为解决信息散落在各处、管理更新困难的问题，Spring Cloud Config 提供了“配置中心”的解决方案。通过建立配置中心，实现每个微服务

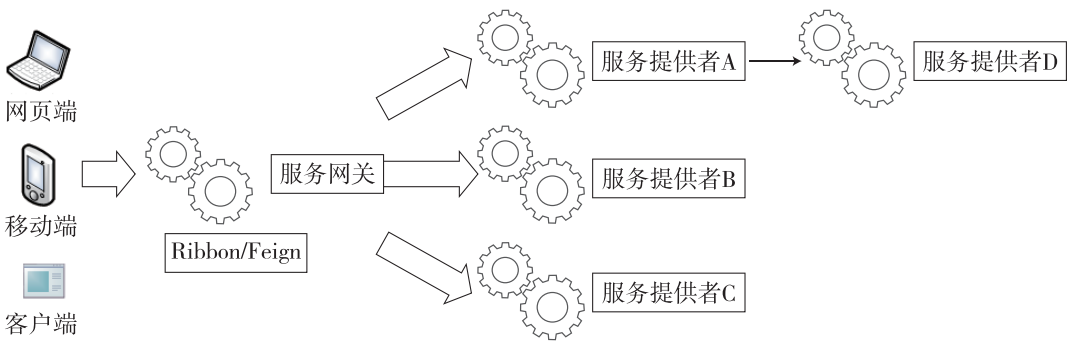

图 2　微服务服务网关

统一的配置管理和更新的实时同步。

3.2.5　安全与认证

应用系统安全性是每个系统实现的重中之重，主要包括用户的鉴权和微服务之间的认证。Spring Cloud Security 提供了一套完整的系统安全解决方案。结合微信 OAuth2 完成用户的登录，结合网关服务实现用户调用受保护 API 时的鉴权。

3.3　微服务的服务拆分思考

3.3.1　坚持应用系统视图层和服务层的分离设计

避免在视图层中直接实现业务逻辑，采用视图层和服务层分离的设计是最基本的解耦方法。通过这层隔离将业务逻辑独立出来，才能对业务进行分析、归纳、组合，这是实现各个应用系统模块间的可重用、松耦合、易维护的基础（见图 3）。

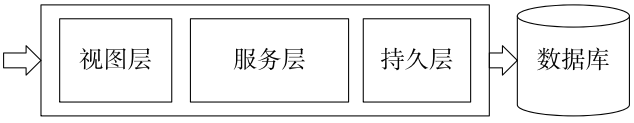

图 3　视图层与服务层分离的设计

3.3.2　使用领域驱动的设计思路

领域驱动设计是微服务进行服务拆分的理论基础，设计师通过从业务领域（逻辑）出发，建立应用系统和现实世界的关系。

在传统的数据库表驱动设计中，一切以数据库表为中心，通过需求文档和概要设计指导数据库表的设计，再通过数据库表完成程序设计，完成数据的增删改查。在领域驱动的设计中，一切以对象为中心，结合真实世界设计领域模型：即现实世界中有什么事物，就有什么对象；现实世界中有什么行为，就有什么方法；现实世界事物有什么关系，对象就有什么关联，而数据是对象保存自身状态的手段。因此，领域驱动设计的本质就是面向对象的设计，是面向对象的落地方式之一。

以个性化邮票定制售卖系统为例，领域驱动设计按下列步骤进行：首先对需求文档完成用例分析，再用用例分析设计领域模型（见图 4）；以领域模型为中心指导数据库设计和程序设计（见图 5）；需求变更时，先回到领域模型做修改，再将领域模型的变更映射为数据库表的修改和程序变更。

通过领域驱动设计，实现软件系统的高内聚和低耦合，最终达到服务逻辑边界的分离。而这一步的达成，为物理边界的分离，进而建立起微服务做好基础准备（见图 6）。

3.3.3　通过服务拆分形成微服务架构

当业务领域分析完成后，即可对整个业务的模块进行拆分，形成微服务。要形成低耦合、高内聚的微服务目标，在进行服务拆分时应该结合业务流程行进的顺序和领域分析的结果，将描述出来的每个服务都作为待选的微服务，并保证每个微服务

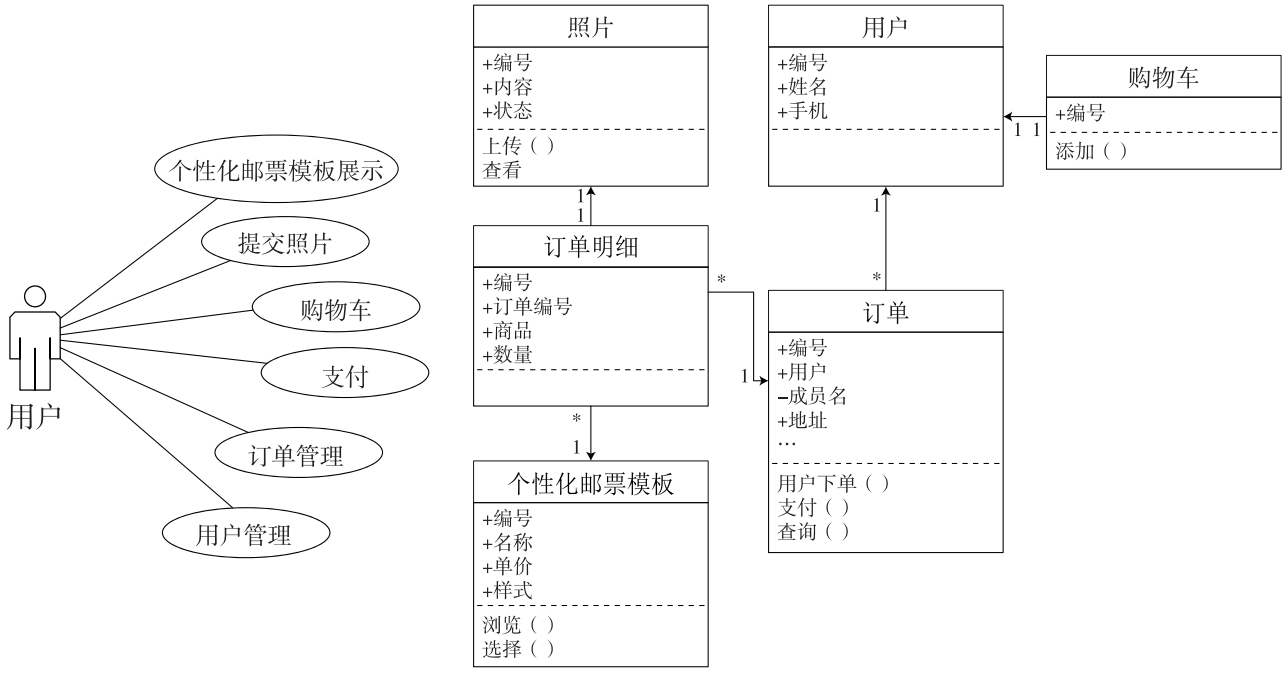

图 4　用例分析和领域模型

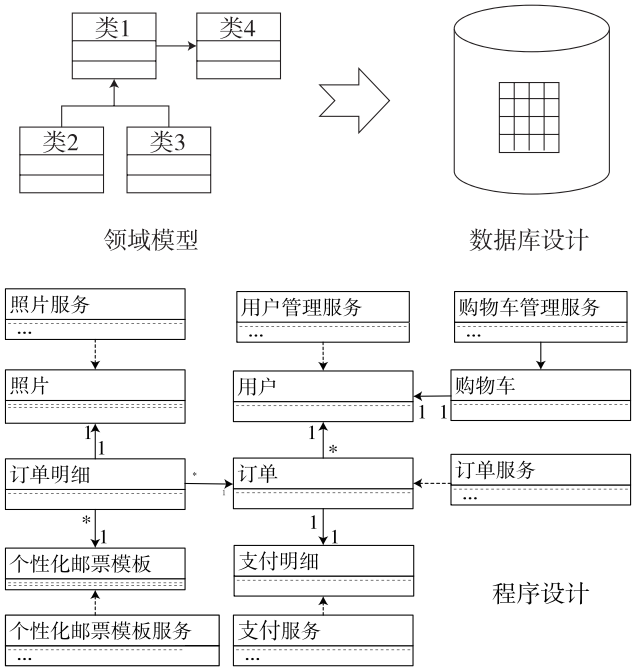

图 5　数据库和程序设计

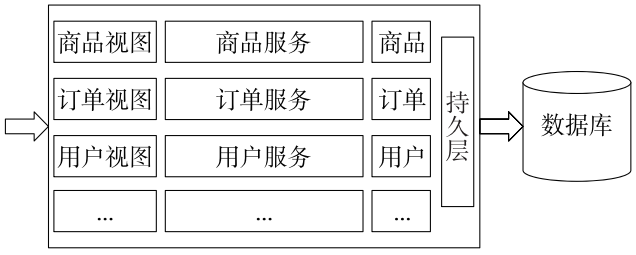

图 6　通过领域驱动划分出服务的逻辑边界

的独立性和完整性。这是一种理想状态，在实际实施过程中还必须考虑到：服务拆分后是否有利于开发团队的开发和维护以及团队间的沟通和合作，拆分粒度是否符合业务复杂度的需要；服务的拆分要考虑到服务的复用，避免重复开发；服务的拆分要保证事务完整性，尽可能避免出现分布式数据库事务、跨服务的事务；服务的拆分要尽可能少地出现服务嵌套调用，并控制服务链式调用数量，避免过度拆分。这是因为微服务之间都是经过网络通信的，过多的服务相互调用反而降低了服务的处理能力。因此，待选的微服务还需进行合并和提炼，微服务拆分工作不是一蹴而就的，而是一个不断设计和反馈的动态过程。

例如，通过对个性化邮票项目需求的领域分析会得到：微信用户登录信息认证、个性化邮票模板商品浏览选择、照片上传和编辑、购物车管理、订单管理、对订单进行微信支付、用户、会员信息维护等待选模块，但不会把每个模块都作为一个微服务来进行设计。实践中以用户订单生成前、支付中、生成后为边界切分微服务，如图 7 所示。

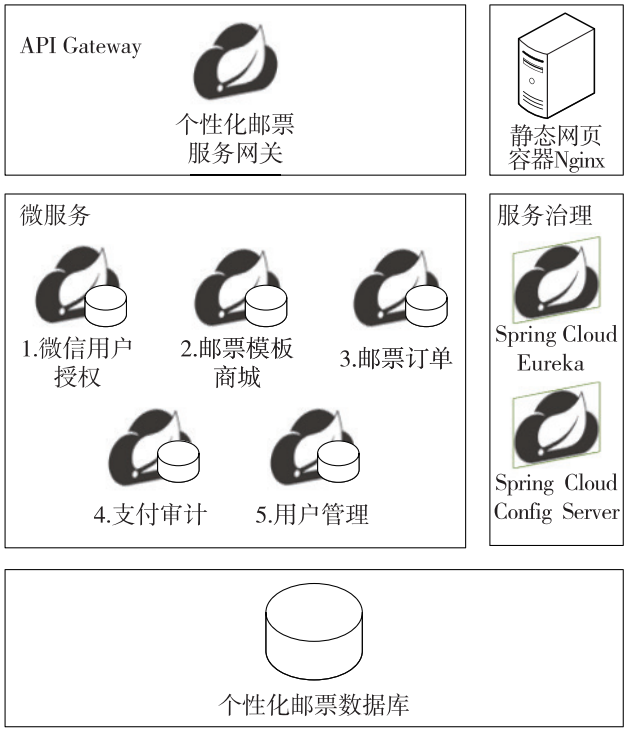

图 7　个性化邮票的微服务拆分

将个性化邮票模板商品浏览选择、照片上传和编辑、购物车管理等服务合并为邮票模板商城服务以处理用户下单支付前的业务，将订单管理、微信支付中的部分功能等服务合并为订单服务，以处理用户下单支付后的业务，如此提升拆分的粒度。同时，将微信用户登录信息认证、微信支付审计单独提取出来作为独立的微服务，用以提供公共服务。这样的拆分结果便于项目组开发，也可提高系统的响应速度，同时提供了系统的可复用性，为其他业务提供支持。

4　应用成果

目前，除了个性化邮票定制售卖系统外，微服务架构还被使用在以下项目或业务中：上海邮政公众号报刊订阅系统的微服务改造、邮电医院体检预约小程序建设、上海邮政企业号营销员俱乐部建设等，通过这些项目的运营，建立起一套自有的面向互联网应用的微服务平台，如图 8 所示。

通过微信为代表的移动互联网，结合微信公众号、小程序、企业号以及微信支付、卡券和微信

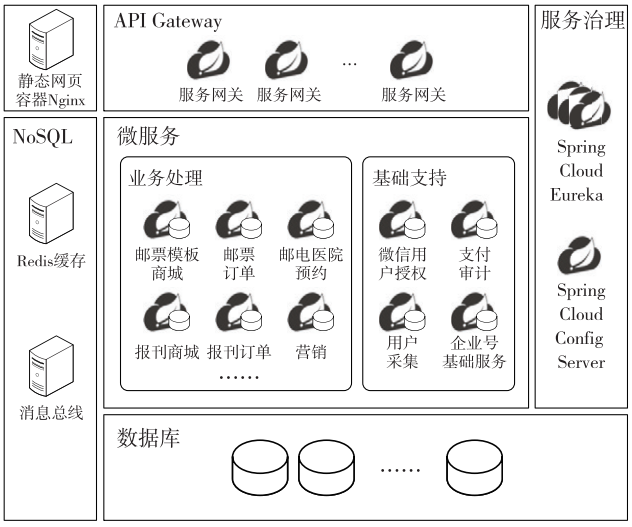

图8　微服务平台

红包等，不断满足上海邮政面向移动互联网的业务发展要求。

其中，个性化邮票定制售卖系统提供了比线下更方便的邮票定制流程，获得了用户的青睐，为黄楼支局带来了很好的业务收益。而建立在微信“上海邮政”公众号中的报刊订阅系统在2019年度完成了近100万笔订单，大收订期间系统同时处理数量超过500OPS。这些业务的开展取得了很好的经济效益和社会效益。

5　结语

本文探讨了微服务架构如何解决传统的单体式应用在开发运行面向互联网应用时的痛点，并介绍了Java环境中建设微服务的方案，以及如何向微服务架构转型，包括软件设计师在内的项目组成员应该如何践行微服务架构所要求的设计目标。微服务架构既是一种开发技术、运行部署的方式，也是设计思想上的转变。微服务架构更适合与互联网结合的有敏捷、快速迭代、快速响应等要求的系统开发和设计。

随着众多国内外厂商的追捧，越来越多的微服务构件加入微服务架构中。微服务架构天生是集群，因此和“云”有着天然的亲密性。以Docker + Kubernetes为代表的容器技术和容器编排的发展，共同构建起的“云原生”生态为微服务架构下服务治理、服务监控、服务伸缩、访问控制、日志采集等提供了平台化的解决方案，自动化运维、DevOps的概念在众多互联网厂商中获得应用，这些也是邮政进一步的研究方向。

阿联酋开通到以色列的邮政和国际优先服务

继阿联酋和以色列达成正式协议并建立全面外交关系后，阿联酋邮政集团已将以色列纳入其全球业务网络，并与以色列邮政紧密合作，为进入以色列全国各地的城市和目的地提供一条可靠的通信渠道。

阿联酋邮政集团公司首席执行官表示：“《亚伯拉罕协议》的签署是一个历史性时刻，相信新的通信渠道将进一步促进两国之间的合作。公司的服务将深化两国之间前景良好的业务和贸易关系，并促进两国之间文化的包容、沟通和交流。此外，与以色列邮政的合作伙伴关系也将带来思想的交流，推动创新，促进邮政行业的发展。”

随着新服务的推出，阿联酋客户可以使用寄达以色列的邮政服务和国际优先服务。阿联酋邮政服务目前遍及多个国家，包括通达190多个目的地的国际优先服务。由于新冠肺炎疫情仍然影响着国际业务运营，阿联酋邮政将继续与合作伙伴合作，将邮政服务恢复到尽可能多的目的地。

（杨永阁　译）

参　考　文　献

[1] MATIN FOWLER．Microsevices[EB/OL]．https://martinfowler.com/articles/microservices.html，2014—03—25

[2] BROOKS，FRED P．No silver buller—essence and accidents of software engineering[J]．IEEE Computer，1987（4）

[3] 沃恩·弗农．实现领域驱动设计[M]．北京：电子工业出版社，2014

[4] 克里斯·理查森．微服务架构设计模式[M]．北京：机械工业出版社，2019