doi:10.3969/j.issn.1000-1247.2017.06.004

电信网厅业务的微服务化研究

宋 科 王 峰 杨明川
中国电信股份有限公司北京研究院

摘要 首先介绍传统电信网厅的系统架构，并分析其在用户量大、迭代频繁等情况下暴露的弊端。接着对微服务概念和实现技术进行剖析，提出电信网厅系统的微服务化改造。最后对微服务系统的高可用性、可伸缩性进行验证。

关键词 微服务 Docker 电信网厅 OpenShift 高可用

1 引言

电信网厅作为各运营商的官方线上服务渠道，为广大用户通过互联网提供业务办理、费用查询、充值缴费、故障申请等一系列的服务，使用户更加方便地办理电信业务，但是随着电信用户数量的增长，网厅业务的更新越来越频繁、系统的负载越来越重，如何保证电信网厅系统的快速更新以及如何确保系统的高可用性成为广大运营商关注的话题。

传统的系统架构把应用程序分解为表示逻辑、业务逻辑和数据访问逻辑三层，每层负责不同的功能，各层的任务具体、细致。但是这个分层是逻辑上的分层，并不是物理上的分层，也就是说这三层的程序会运行在同一个机器的同一个进程中，这就导致在程序更新频繁时的交付周期较长，同时在用户请求数增多的情况下无法保证相应业务的灵活扩展。

微服务是一种近几年非常流行的架构模式，通过对应用进行纵向划分，解除业务与业务之间物理上的耦合，提高系统的灵活性、持续交付能力、高可用性等。

2 当前网厅业务系统架构分析

当前网厅业务基于面向对象的软件开发原则、企业架构模式等理念及方法，以有效组织软件结构和为用户提供不同功能为出发点，把应用程序划分为表示层、业务逻辑层和数据访问层。表示层是直接面对用户的交互部分，是用户可以看见、听见并输入相关内容的部分，用户通过表示层和软件进行交互，获取系统提供的服务。目前电信网厅的表示层是以浏览器、APP的形式提供给用户。业务逻辑层是根据用户的输入信息进行业务处理，进而转化成底层数据访问接口的格式。数据访问层实现对业务数据的持久化工作，同时把保存的数据提供给其他系统。

当前网厅业务系统是一个明显的单块架构应用，该应用功能集中、代码和数据中心化，所有模块被视作一个整体，运行时运行在同一个进程中。单块架构应用在业务量增大、版本更新频繁时短板会暴露出来。

维护成本增加：随着业务系统的功能增多，当出现缺陷时，引起该缺陷的原因就难以定位，修复系统需要花费较长的时间。

持续交付周期长：在代码越来越复杂的情况下，由于单块架构所有应用都封装成一整块，每次更新时需要涉及所有业务，项目构建和部署的时间也会相应加长，单位时间内项目构建效率变低。

可伸缩性差：在所有应用程序的代码都运行在同一个服务器上，服务器的网络命名空间、进程间通信命名空间等资源只能被一个应用使用，扩展应用的方法只有增加服务器数量。

这种扩展方式的成本非常高，而且不同功能资源使用情况不同，使用水平扩展的整体花费非常高。在业务量增多和版本更新频繁的情况下，单块架构已经不适合于电信的业务网厅系统，需要一种能弥补单块架构不足的全新架构来解决这些问题。

3 微服务及相关技术

3.1 微服务的概念及特征

微服务通过把小的服务开发成单一应用的形式，每个应用运行在单一的进程中，各个应用通过轻量级API进行通信，通过多个单一应用组成完整的应用软件架构风格。

微服务风格的系统具有以下特征。

通过服务实现组件化：由于服务能够独立部署，所以对

单个服务的改动不需要重新部署整个应用，同时服务间使用远程调用机制可以明确组件之间的接口，避免组件过度耦合。

围绕业务组织能力：微服务围绕业务能力拆分并组合，各个服务使用与业务范围相符的开源技术，有效融合多种语言、多种框架。

轻量级通信机制：微服务架构在尽可能解耦合的同时保持关联性，服务间使用REST协议进行通信，常用的方式为HTTP和轻量级消息队列。

高可用：把服务用作组件在设计时就要考虑到故障恢复技术，微服务架构使用轻量级的进程实现，对于出现异常的进程可以进行重新启动以达到系统故障的及时排除。

微服务架构模式有很多好处。首先，把巨大单体式应用分解为多个服务，解决复杂性问题。应用被分解为多个可管理的服务，每个服务都有一个用RPC或者消息驱动API定义清楚的边界。单体式编码方式很难实现的功能采用微服务架构模式使该功能模块化，由此，单个服务很容易开发、维护和理解。第二，微服务架构使得每个团队专门开发一个服务。开发者选择合适的开发技术来提供API服务，充分利用不同技术的优势。第三，微服务架构模式是每个微服务独立的部署，开发者不再需要协调其他服务部署对本服务的影响。这种改变可以加快部署速度，使得持续化部署成为可能。最后，微服务架构模式使得每个服务独立扩展，可以根据每个服务的规模来部署满足需求的规模，甚至于可以使用更适合于服务资源需求的硬件。

3.2 单个微服务的实现——Docker

Docker 是一个开源的应用容器引擎，让开发者可以打包其应用以及依赖包到一个可移植的容器中，然后发布到任何流行的 Linux 机器上。Docker的设想是：交付运行环境如同海运，OS如同一个货轮，每一个在OS基础上的软件都如同一个集装箱，用户可以通过标准化手段自由组装运行环境，同时集装箱的内容可以由用户自定义，也可以由专业人员设置。这样交付一个软件，就是一系列标准化组件集合的交付。

Docker核心解决的问题是使用lxc来实现虚拟机的功能，在不虚拟操作系统的情况下提高资源利用率。Docker使用namespace进行容器间网络、文件系统等资源隔离，使用cgroup实现资源的配额和度量。Docker把程序的代码和相关依赖打包到镜像中，该镜像在运行Docker Daemon的机器上都可以运行，达到一次部署多次执行的目的。

Docker也有缺点，在网络和安全方面尤为突出，这也成为多数开源产品改进的地方。

3.3 集群上的微服务——OpenShift

由于原生Docker容器的每个IP地址是网桥网络中随机的一个，而各网桥的IP地址是Docker随机分配的，这就导致Docker容器在集群部署时的服务发现成为一个大难题。同时，由于Docker直接使用宿主机的操作系统，在某一个容器崩溃的情况下可能会引起整个宿主机的崩溃，安全性无法保障。

基于原生Docker的缺点，RedHat研发了基于集群部署Docker的开源系统OpenShift。Openshift使用Open vSwitch建立网桥，使用VxLAN连接各宿主机的网桥，集群中使用ETCD作为中心数据库同步网络配置，解决网络连接问题。发布服务时，OpenShift为该服务提供一个虚拟IP地址，提供该服务的所有容器可以通过这个IP地址被访问到。安全方面，OpenShift提供了独有的授权/鉴权机制，来控制容器对宿主机的操作，同时限制容器中的用户，从而达到对宿主机的保护。

除此之外，OpenShift还提供一键部署的模板，输入相应参数就可以部署一整套集群；提供持久化存储抽象的持久化卷请求，维护集群中容器特定数量，从而实现高可用的副本控制器、基于资源及实际情况进行调度的调度控制器来进行更好的集群管理。

文中对电信网厅业务的容器化改造在OpenShift上进行，后续将演示改造后的微服务化系统的相关功能。

4 电信网厅业务微服务化的改进及相关功能验证

4.1 电信网厅微服务架构

基于电信网厅的常用业务，改造后的系统由固网服务、移动服务、固话服务和用户信息服务4个服务构成，各服务副本数暂定为2，可根据实际情况进行调整，每个服务的副本共享持久化数据库。更新系统时，只需要更新相应的服务，其他服务则可以继续运行。图1展示了电信网厅微服务化后的架构。

4.2 电信网厅微服务化的高可靠性验证

OpenShift提供的副本控制器保证集群中某个镜像的副本数量维持在某个特定的数量，这意味着某个容器发生故障而死亡，副本控制器会重新启动一个容器，如图2所示。同时，如果副本控制器对应的镜像改变，集群中所有的容器都会变成基于改变后镜像的容器。这种方案为项目更新带来极大的便利，更新的旧镜像的容器副本逐个消除，新镜像的容器副本逐个生成，整个过程中服务处于可用状态，如图3所示。

4.3 电信网厅微服务化的可伸缩性验证

由于每个容器有自己的网络协议栈、进程ID空间等资源，微服务化系统的可伸缩性不再受到服务器数量的限制，只取决于可用的计算、存储资源是否充足。在OpenShift平台

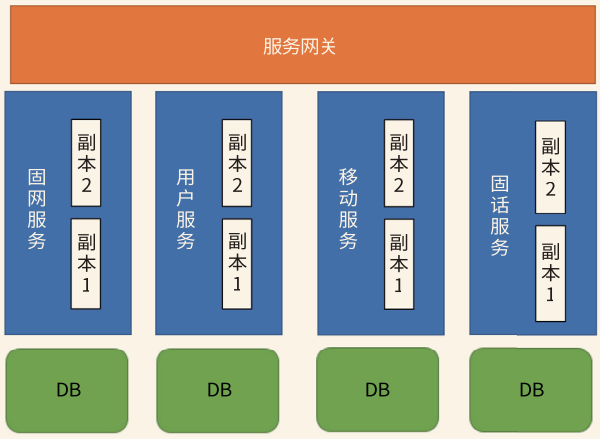

图1　网厅微服务化架构

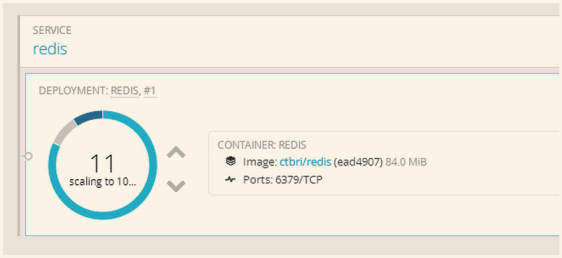

图2　单个容器发生故障

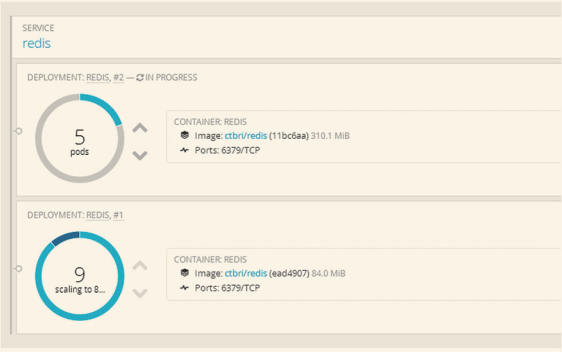

图3　微服务版本更新

中使用命令ocscale-relicas=num_of_pod rc rc_name就可以把rc_name对应的容器数量设置成num_of_pod，轻松实现服务的伸缩，具体示例如图4、图5所示。

5 结束语

综上分析单块架构在电信网厅需求不断更新情况下的不足及微服务架构的优势，为电信网厅分解相关业务，设计出基于微服务架构的电信网厅系统，并在基于Docker的OpenShift平台上模拟系统的微服务化，验证微服务化系统的高可用性和可伸缩性，为电信网厅业务的微服务化提供参考。

图4　服务扩展前

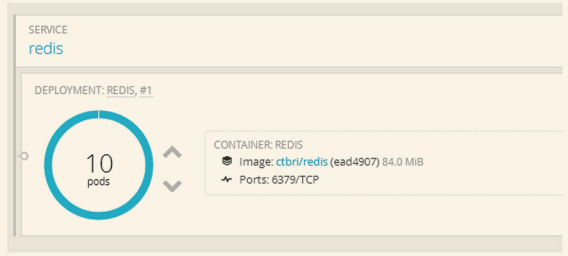

图5　服务拓展后

参考文献

[1] 王磊.微服务架构与实践[M].北京:电子工业出版社,2016

[2] 汪恺,张功萱,周秀敏.基于容器虚拟化技术研究[J].计算机技术与发展,2015(08)

[3] Stefan Walraven,Eddy Truyen,Wouter Joosen.Comparing PaaS offerings in light of SaaS development[J].Computing,2014(8)

[4] BOETTIGER C.An introduction to Docker for reproducible research.ACM SIGOPS Operating Systems Review,2015

[5] RichardsonC.Introductionto Microservices.https://www.nginx.com/blog/introduction—to—microservices/,2015

如对本文内容有任何观点或评论，请发E-mail至ttm@bjxintong.com.cn。

作者简介

宋科

现就职于中国电信股份有限公司北京研究院，助理工程师，主要研究方向为容器化的云计算、应用微服务化和研发运维一体化。

王峰

现任中国电信股份有限公司北京研究院云计算技术平台研发中心主任，教授级高级工程师，长期从事云计算和软件定义领域的关键技术和创新产品研发。

杨明川

现任中国电信股份有限公司北京研究院副总工程师，云计算与大数据产品线总监，参与多项国家重大专项研究，获得多项省部级科技奖，入选国家“863”计划专家库。