上海理工大学

**《工程学导论（2组）---电子信息类》**

**课程报告**

**2020 /2021 学年第 1 学期**

**微服务架构体系研究综述**

|  |  |
| --- | --- |
|  |  |
|  |  |
| **姓 名** | **宁辉岳** |
| **学 号** | **1722010225** |
| **日 期** | **2020年12月** |

# 摘 要

微服务为我们提供了一种更好地持续交付业务影响的方式。相较于单体应用，使用微服务构建出来的应用是由一系列松耦合的、自治的服务组成的。通过开发这些只做一件事的服务，开发者可以避免大型应用中所存在的缺乏活力和混乱的状态。即便是对于已有的应用系统，开发者也可以一步步地将其中的功能抽取为独立的服务，这样可以使整个系统的可维护性更强。本文以综述的形式对微服务架构以及部分技术栈进行一步步地剖析，对入门级的开发人员快速了解企业微服务或者自己构建微服务应用提供帮助

**关键词：**  微服务 k8s docker 持续部署 持续集成 单体应用 SOA

# ABSTRACT

Microservices provide us with a better way to consistently deliver business impact. Incontrast to monolithic applications, applications built with microservices are composed of a series of loosely coupled, autonomous services. By developing these services that do only one thing, developers can avoid the lack of dynamism and chaos that exists in large applications. Even for existing applications, the developer can step-by-step extract the functionality into separate services, which makes the system more maintainable. This article provides a step-by-step analysis of the microservices architecture and part of the technology stack in the form of an overview, which can help entry-level developers to quickly understand enterprise microservices or to build their own microservices applications.

**KEYWORDS：**  Microservice Kubernetes Docker CI CD Monolith SOA

# 目 录

**摘要**

**ABSTRACT**

[摘 要 i](#_Toc58073327)

[ABSTRACT ii](#_Toc58073328)

[目 录 iii](#_Toc58073329)

[第1章 绪论 5](#_Toc58073330)

[1.1 概述 5](#_Toc58073331)

[1.1.1 SOA架构 5](#_Toc58073332)

[1.1.2 单体应用架构 5](#_Toc58073333)

[1.2 微服务的历史及其技术演变 5](#_Toc58073334)

[1.3 微服务技术研究意义 6](#_Toc58073335)

[第2章 技术综述 7](#_Toc58073336)

[2.1 目前主流的微服务技术框架 7](#_Toc58073337)

[2.1.1 Spring Cloud 7](#_Toc58073338)

[2.1.2 Dubbo 7](#_Toc58073339)

[2.1.3Istio 7](#_Toc58073340)

[2.2 Docker 8](#_Toc58073341)

[2.3Kubernets 9](#_Toc58073342)

[2.31 使用k8s的原因 9](#_Toc58073343)

[2.3.2 k8s原理简介 10](#_Toc58073344)

[2.3.3 k8s与docker的关系 11](#_Toc58073345)

[第3章 总结和展望 11](#_Toc58073346)

[参考文献 13](#_Toc58073347)

# 第1章 绪论

## 1.1 概述

简单来说，所谓微服务（MicroService）是一种有效的拆分应用，实现敏捷开发和部署的架构模式。微服务架构可以被认为是对SOA的特殊化，也是一种可以克服单体架构缺陷的替代模式。

### 1.1.1 SOA架构

SOA是一种软件设计模式，主要应用于不同应用组件之间通过网络协议来互交操作。从应用层面：SOA是一种应用框架，它着眼于日常的业务应用，并将它们划分为单独的业务功能和流程，即所谓的服务。从软件层面：SOA是一个组件模型，它将应用程序的不用功能单元（服务）通过这些服务之间定义良好的接口和契约联系起来。在2000年初，我们目睹了SOA的崛起，这是一种非常流行的软件架构设计范式。简而言之，SOA 是一种软件架构模式，用于构建大型的企业应用程序，这些应用程序通常要求集成多种服务，而每种服务使用不同的平台和编程语言来构建，并通过通用的通信机制进行交互。

### 1.1.2 单体应用架构

让我们以网店为例，我们知道，很多电商网站都可以通过多种设备访问，所以这些网站通常都为笔记本电脑和移动设备提供了不同的用户界面，我们也知道，多个操作或服务彼此依赖，以确保应用程序的正常运行，其中一些服务负责创建账号、显示产品目录、建立和验证购物车、生成账单、确认订单、完成支付等。

## 1.2 微服务的历史及其技术演变

在单体应用程序中，所有这些服务都在同一个应用程序层上运行，很显然，随着服务数量的增加，应用程序的规模将不断增长。这可能会让构建和维护应用程序代码库的开发人员不堪重负。

之后救星微服务出现了（以下为微服务概念产生的时间线）：

* 2005年Dr. PeterRodgers在Web ServicesEdge大会上提出了“Micro-Web-Services”的概念；
* 2011年一个软件架构工作组使用了“microservice”一词来描述一种架构模式；
* 2012年同样是这个架构工作组，正式确定用“microservice”来代表这种架构；
* 2012年ThoughtWorks的James Lewis针对微服务概念在QCon San Francisco2012发表了演讲；
* 2014年James Lewis和Martin Flower[[1]](#footnote-1)合写了关于微服务的一篇学术性的文章，详细阐述了微服务。

所以微服务概念是2012年出现的，作为加快Web和移动应用程序开发进程的一种方法，2014年开始受到各方的关注，而2015年，可以说是微服务的元年；越来越多的论坛、社区、blog以及互联网行业巨头开始对微服务进行讨论、实践，可以说这样更近一步推动了微服务的发展和创新。

## 1.3 微服务技术研究意义

团队使用微服务框架有以下优势：

* 团队独立：每个服务都是一个独立的开发团队，这个小团队可以是 2 到 5 人的开发人员组成；
* 技术独立：采用去中心化思想，服务之间采用 RESTful 等轻量协议通信，使用什么技术什么语言开发，别人无需干涉；
* 前后端分离：采用前后端分离开发，提供统一 Rest 接口，后端不用再为 PC、移动端开发不同接口；
* 数据库分离：每个微服务都有自己的存储能力，可以有自己的数据库。也可以有统一数据库；
* 服务拆分粒度更细，有利于资源重复利用，提高开发效率；
* 一个团队的新成员能够更快投入生产；
* 微服务易于被一个开发人员理解，修改和维护，这样小团队能够更关注自己的工作成果。
* 无需通过合作才能体现价值；
* 可以更加精准的制定每个服务的优化方案（比如扩展），提高系统可维护性；
* 适用于互联网时代，产品迭代周期更短。

# 第2章 技术综述

## 2.1 目前主流的微服务技术框架

目前比较火的主流微服务框架有Spring Cloud , Dubbo和Istio

### 2.1.1 Spring Cloud

Spring Cloud来自Spring，具有Spring 社区的强大支撑，还有Netflix强大的后盾与技术输出。Netflix作为一家成功实践微服务架构的互联网公司在几年前就把几乎整个微服务框架栈开源贡献给了社区，这些框架开源的整套服务架构套件是Spring Cloud的核心。

- Eureka:服务注册发现框架；

- Zuul:服务网关；

- Karyon：服务端框架；

- Ribbon：客户端框架；

- Hystrix：服务容错组件；

- Archaius：服务配置组件；

- Servo：Metrics组件；

- Blitz4j：日志组件；

### 2.1.2 Dubbo

Dobbo是一个分布式服务框架，是阿里开放的微服务化治理框架，致力于提高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。其核心部分[[2]](#footnote-2):

- 远程通讯: 提供对多种基于长连接的NIO框架抽象封装，包括多种线程模型，序列化，以及“请求-响应”模式的信息交换方式；

- 集群容错: 提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持；

- 自动发现: 基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。

Dubbo 也是采用全 Spring 配置方式，透明化接入应用，对应用没有任何 API 侵入，只需用 Spring 加载 Dubbo的配置即可，Dubbo 基于 Spring 的 Schema 扩展进行加载。当然也支持官方不推荐的 API 调用方式。

### 2.1.3Istio

lstio 作为用于微服务聚合层管理的新锐项目，是Google、IBM、Lyft（海外共享出行公司、Uber劲敌），首个共同联合开源的项目，提供了统一的连接，安全，管理和监控微服务的方案。

目前首个测试版是针对Kubernetes环境的，社区宣称在未来几个月内会为虚拟机和Cloud Foundry 等其他环境增加支持。lstio将流量管理添加到微服务中，并为增值功能（如安全性、监控、路由、连接管理和策略）创造了基础。

- HTTP、gRPC 和 TCP 网络流量自动负载均衡；

- 提供了丰富的路由规则，实现细颗粒度的网络流量行为控制；

- 流量加密、服务件认证，以及强身份声明；

- 全范围（Fleet-wide）的策略执行；

- 深度遥测和报告。

## 2.2 Docker

Docker 是一个开源应用容器（当然目前也分为CE和EE版本，不完全开源化，也存在收费版本），让开发者可以打包他们的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，也可以实现虚拟化。容器是完全使用沙箱机制，相互之间不会有任何接口。

Docker 作为容器工具可以把：业务逻辑容器、数据库容器、储存容器、队列容器使得软件可以拆分成若干个标准化容器，然后像搭积木一样组合起来，让彼此通信，从而形成微服务。

因此微服务很适合用 Docker 容器实现，每个容器承载一个服务。一台计算机同时运行多个容器，从而就能很轻松地模拟出复杂的微服务架构。

Docker容器技术主要涉及以下几个重要概念：

1）Docker引擎：是一个运行在服务器上的后台进程，可通过Docker命令与之通信。

2）镜像：主要用来打包应用程序，将其载入Docker引擎中便可运行镜像中的程序。

3）容器：镜像中可运行的实例。镜像和容器的关系与面向对象思想中类和对象的关

系相似。

4）镜像库：存放并管理所有的基础镜像和最终交付的镜像。[[3]](#footnote-3)

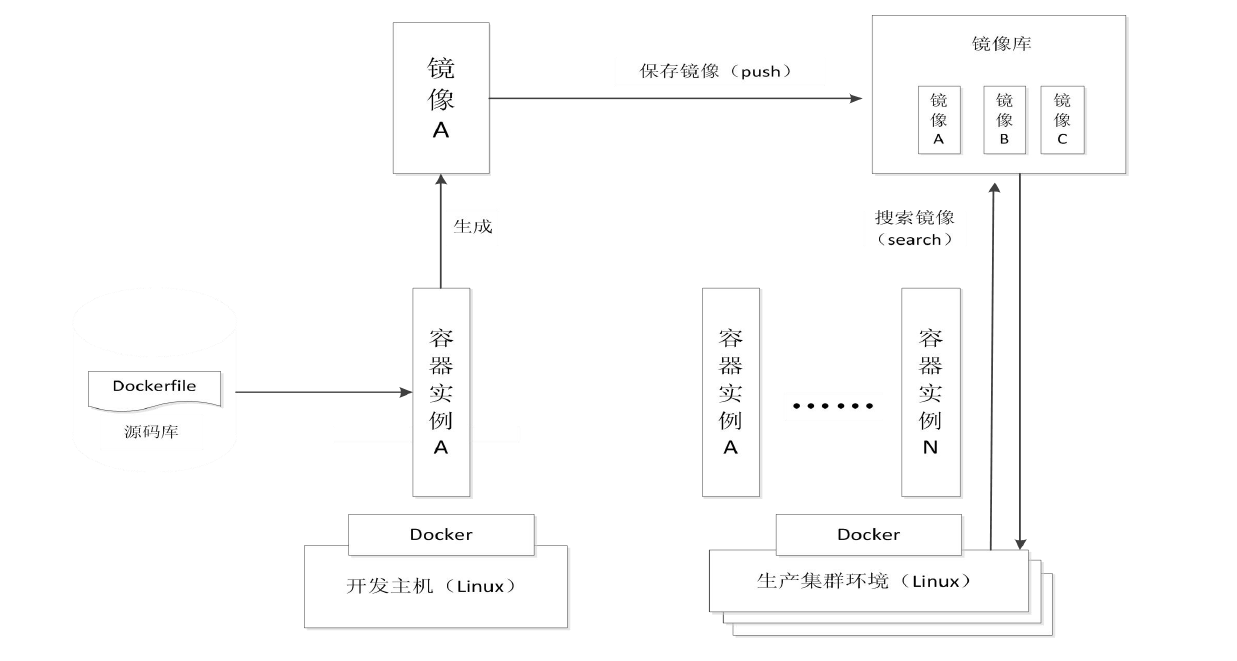


图2.2 Docker 架构图

如图2.2 所示，将应用程序和DockerFile 构建的镜像生成一个新的镜像保存在镜像仓库中，如果要在别的机器上运行微服务实例，只需从镜像仓库中拉取镜像并运行即可。通过各公司的大量实践证明，Docker 和微服务相结合，在独立部署、运维管理和快速交付上面会具有较大的优势。

## 2.3Kubernets

Kubernetes，简称 k8s或者 “kube”，是一个开源的 Linux 容器自动化运维平台，它消除了容器化应用程序在部署、伸缩时涉及到的许多手动操作。换句话说，你可以将多台主机组合成集群来运行 Linux 容器，而 Kubernetes 可以帮助你简单高效地管理那些集群。构成这些集群的主机还可以跨越公有云、私有云以及混合云。

Kubernetes 最开始是由 Google 的工程师设计开发的。Google 作为 Linux 容器技术的早期贡献者之一，曾公开演讲介绍 Google 如何将一切都运行于容器之中（这是 Google 的云服务背后的技术）。Google 一周内的容器部署超过 20 亿次，全部的工作都由内部平台 Borg 支撑。Borg 是 Kubernetes 的前身，几年来开发 Borg 的经验教训也成了影响 Kubernetes 中许多技术的主要因素。

### 2.31 使用k8s的原因

或者说使用k8s的优势在哪里

真实的生产环境应用会包含多个容器，而这些容器还很可能会跨越多个服务器主机部署。Kubernetes 提供了为那些工作负载大规模部署容器的编排与管理能力。Kubernetes 编排让你能够构建多容器的应用服务，在集群上调度或伸缩这些容器，以及管理它们随时间变化的健康状态。

Kubernetes 也需要与网络、存储、安全、监控等其它服务集成才能提供综合性的容器基础设施。当然，这取决于你如何在你的环境中使用容器。一个初步的 Linux 容器应用程序把容器视作高效、快速的虚拟机。一旦把它部署到生产环境或者扩展为多个应用，很显然你需要许多组托管在相同位置的容器合作提供某个单一的服务。随着这些容器的累积，你的运行环境中容器的数量会急剧增加，复杂度也随之增长。

Kubernetes 通过将容器分类组成“pod”来解决了容器增殖带来的许多常见问题。pod 为容器分组提供了一层抽象，以此协助你调度工作负载以及为这些容器提供类似网络与存储这类必要的服务。Kubernetes 的其它组件帮助你对 pod 进行负载均衡，以保证有合适数量的容器支撑你的工作负载。正确实施的 Kubernetes，结合类似 Atomic Registry、Open vSwitch、heapster、OAuth 和 SELinux的开源项目，让你可以管理你自己的整个容器基础设施。

### 2.3.2 k8s原理简介

Kubernetes集群包含有节点代理kubelet和Master组件(APIs, scheduler, etc)，一切都基于分布式的存储系统。下面是Kubernetes的架构图。

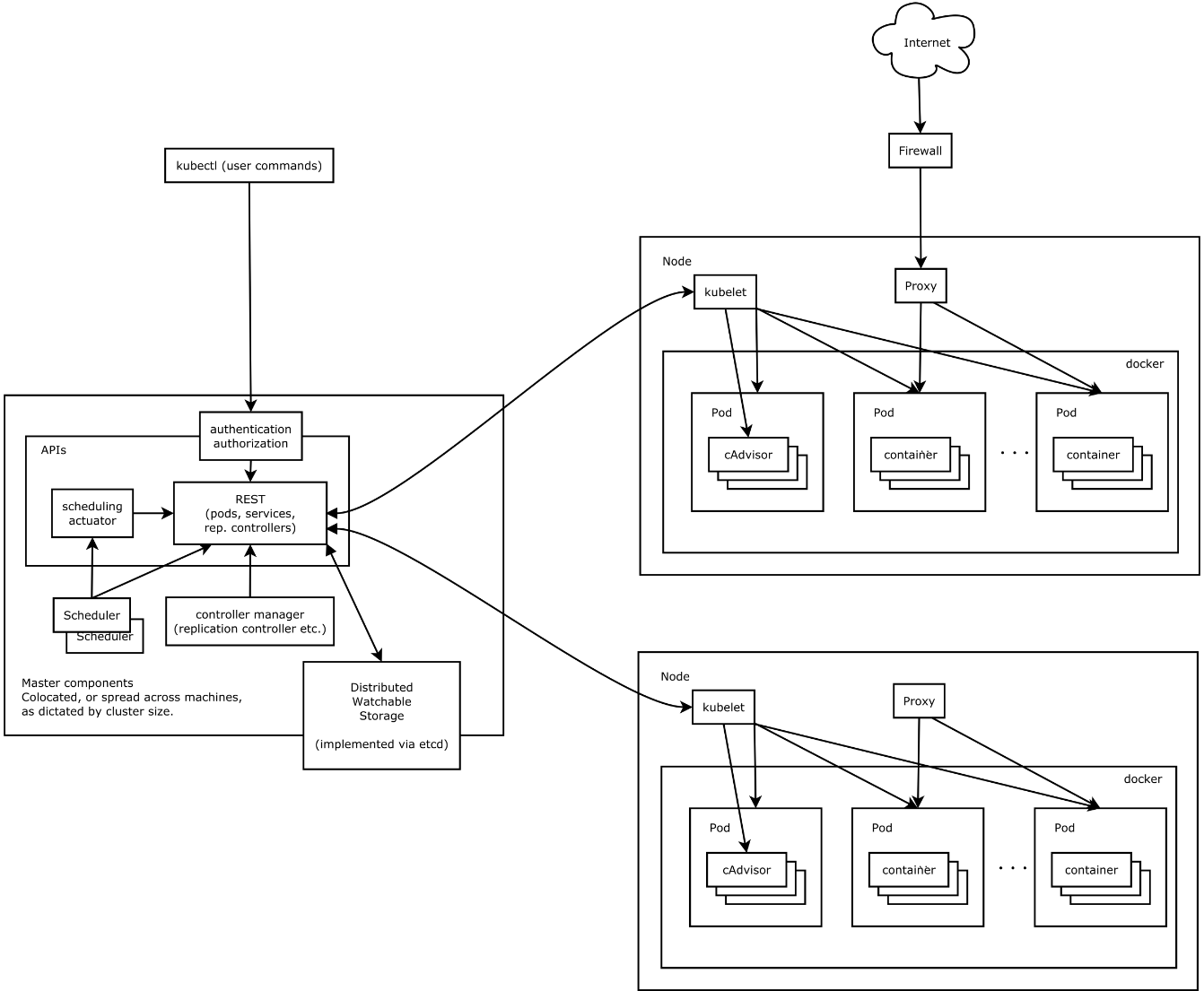


图2.2.2 kubernets架构图

和其它技术一样，大量的专有名词有可能成为入门的障碍。下面解释一些通用的术语，希望帮助你理解 Kubernetes。

-Master（主节点）： 控制 Kubernetes 节点的机器，也是创建作业任务的地方。

-Node（节点）： 这些机器在 Kubernetes 主节点的控制下执行被分配的任务。

-Pod： 由一个或多个容器构成的集合，作为一个整体被部署到一个单一节点。同一个 pod 中的容器共享 IP 地址、进程间通讯（IPC）、主机名以及其它资源。Pod 将底层容器的网络和存储抽象出来，使得集群内的容器迁移更为便捷。

-Replication controller（复制控制器）： 控制一个 pod 在集群上运行的实例数量。

-Service（服务）： 将服务内容与具体的 pod 分离。Kubernetes 服务代理负责自动将服务请求分发到正确的 pod 处，不管 pod 移动到集群中的什么位置，甚至可以被替换掉。

-Kubelet： 这个守护进程运行在各个工作节点上，负责获取容器列表，保证被声明的容器已经启动并且正常运行。

-kubectl： 这是 Kubernetes 的命令行配置工具。

### 2.3.3 k8s与docker的关系

  Docker技术依然执行它原本的任务。当 kubernetes 把 pod 调度到节点上，节点上的 kubelet 会指示 docker 启动特定的容器。接着，kubelet 会通过 docker 持续地收集容器的信息，然后提交到主节点上。Docker 如往常一样拉取容器镜像、启动或停止容器。不同点仅仅在于这是由自动化系统控制而非管理员在每个节点上手动操作的。

# 第3章 总结和展望

自2012 年微服务架构的概念被正式提出以来，微服务技术凭借其独立部署、可扩展性、快速迭代、容错性、易于开发等优点正在快速发展并逐渐成为业界最流行的软件架构。本文首先介绍了微服务架构的概念和主要特点，然后详细描述了国内外学术界对于微服务架构的学术论文与研究工作，最后介绍了微服务架构的优点和缺点以及与传统的单体式架构特点的对比。

微服务架构是互联网时代的特点与需求、分布式架构与敏捷开发的流行、持续集成持续部署、容器技术的慢慢成熟等条件结合的产物，是互联网时代技术浪潮中的一朵小小的浪花，其核心仍然是康威定律中探讨的组织结构与其设计的系统结构之间的关系。微服务技术对于应用服务开发团队来说，与其说是一种技术，不如说是一种针对软件开发的指导思想和设计理念。微服务架构本身不是目的，也不是能够解决所有

问题的灵药，更重要的还是开发团队结合自身实际需求与具体问题选择更合适的应用架构。

# 参考文献

1. Merkel D. Docker: lightweight Linux containers for consistent development and deployment[J]. 2014,2014(239).
2. 张晶, 黄小锋. 一种基于微服务的应用框架[J]. 计算机系统应用, 2016, 25(9):265-270.
3. 范迪, 朱志祥. 一种Dubbo 框架的授权认证方案[J]. 计算机技术与发展, 2017, 27(11):115-118.
4. Dobbo官网：<http://dubbo.apache.org/>
5. 阿里云开发者社区：<https://developer.aliyun.com/indexFeed/?spm=a2c6h.12873639.1364563.13.6e52423fPAZwKg>
6. Cosmina I, Harrop R, Schaefer C, et al. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools[M]. 2017.
7. 陈建娟,刘行行.基于Kubernetes 的分布式ELK 日志分析系统[J].电子技术与软件工程，2016(15):211-212

1. Martin Fowler是国际著名的OO专家，敏捷开发方法的创始人之一，现为ThoughtWorks公司的首

   席科学家。 [↑](#footnote-ref-1)
2. 来源Dobbo官网：http://dubbo.apache.org/ [↑](#footnote-ref-2)
3. Merkel D. Docker: lightweight Linux containers for consistent development and deployment[J]. 2014,

   2014(239). [↑](#footnote-ref-3)