

Kubernetes 服务部署实践

袁璞 崔方迪 侯志鑫

中国电子科技集团有限公司 第十五研究所 试验训练专业部

DOI:10.12238/acair.v2i2.7419

[摘要] 本文研究了Kubernetes平台的基本概念、基于Kubernetes的服务部署方法以及动态扩容和负载均衡的实验验证。通过实验展示了这些方法和优化策略的应用效果。研究结果表明,Kubernetes作为容器编排平台具有高度可扩展性和自动化运维特性。

[关键词] Kubernetes平台; 扩容; 负载均衡

中图分类号: U225.7v **文献标识码:** A

Service Deployment Practice of Kubernetes

Pu Yuan Fangdi Cui Zhixin Hou

Experimental Training Department of the 15th Research Institute of China Electronics Technology Group Co., Ltd

[Abstract] This article studies the basic concepts of the Kubernetes platform, service deployment methods based on Kubernetes, and experimental verification of dynamic scaling and load balancing. The application effects of these methods and optimization strategies were demonstrated through experiments. The research results indicate that, Kubernetes, as a container orchestration platform, has high scalability and automated operation and maintenance features.

[Key words] Kubernetes platform; scalability; load balancing

1 研究背景与意义

随着互联网技术的飞速发展,传统的服务部署模式已经无法满足当今复杂、动态和大规模的业务需求。例如,如何有效地部署和管理大量的服务,如何处理服务之间的通信和协作,如何实现服务的高可用和弹性扩展,如何进行服务的监控和故障排查等^[1]。为了解决这些问题,需要一种强大、灵活且易于使用的平台。Kubernetes作为一个开源的容器编排和管理平台,提供了一系列强大的功能和工具,可以有效地支持微服务架构的实现和管理。

Kubernetes能够自动化地部署、扩展和管理容器化的应用,支持服务发现和负载均衡,提供了存储系统、网络和安全的解决方案,还能够进行资源的调度和监控,以及日志的管理等。因此,Kubernetes已经成为了服务部署的理想平台。

然而,尽管Kubernetes提供了丰富的功能和工具,但如何有效地使用Kubernetes实现和优化服务架构,仍然是一个值得研究的问题。例如,如何设计和部署服务,如何实现服务的动态扩展和弹性,如何提高服务的可用性。这些问题的解决需要深入理解微服务架构和Kubernetes平台,需要掌握一系列设计原则和实践策略。

1.1 论文结构

本论文的结构如下:第二部分介绍Kubernetes平台的基本

概念和特点;第三部分阐述基于Kubernetes的服务部署方法;第四部分通过实验测试Kubernetes的动态扩容和负载均衡,展示这些实现方法和优化策略的应用和效果;最后,对全文进行总结。

2 Kubernetes 概述

Kubernetes是一个开源的容器编排和管理平台,由Google发起并由Cloud Native Computing Foundation(CNCF)维护。Kubernetes提供了一系列强大的功能和工具,可以有效地支持服务架构的实现和管理。

Kubernetes的主要功能包括:自动化的部署、扩展和管理容器化的应用;服务发现和负载均衡;存储系统、网络和安全的解决方案;资源的调度和监控;日志的管理等。这些功能使得Kubernetes成为了微服务架构的理想平台^[2]。

Kubernetes的架构主要包括Master节点和Worker节点两部分。Master节点负责整个集群的管理和控制,包括调度、API服务、集群状态存储等;Worker节点负责运行应用容器,包括容器运行时、服务代理、资源监控等。Kubernetes使用声明式的配置语言描述应用的部署状态,然后通过控制循环来确保实际状态与期望状态一致。

Kubernetes的优势主要体现在以下几个方面:首先,Kubernetes提供了丰富的功能和工具,可以满足微服务架构的各种需求;其

次, Kubernetes支持多种云平台和操作系统, 具有很高的兼容性和可移植性; 最后, Kubernetes有着活跃的社区和丰富的生态系统, 提供了大量的插件和工具, 可以方便地扩展和定制。

3 基于Kubernetes的服务部署

3.1 部署流程

Kubernetes 的部署主要流程如下:

(1) 部署镜像到Pod。a. 创建容器镜像: 首先, 开发者需要创建一个容器镜像, 并将其推送到容器注册中心(如Docker Hub或私有的容器仓库)。b. 编写配置文件: 接着, 开发者编写一个配置文件(通常是YAML格式), 定义了一个或多个Pod的期望状态, 其中包括应当在Pod中运行的容器镜像。c. 创建Pod: 通过 `kubectl apply` 命令, Kubernetes的控制平面接收到这个配置文件, 并基于文件中的定义创建Pod。d. 调度Pod: Kubernetes的调度器(scheduler)选择一个合适的节点来运行这个Pod, 并将Pod与该节点绑定。e. 启动容器: 所选节点上的 kubelet 组件负责启动Pod, 它通过容器运行时(如Docker)拉取配置文件中指定的容器镜像, 并启动容器。f. 健康检查: Pod中的容器一旦启动, kubelet会定期进行健康检查, 以确保容器正常运行。如果检查失败, kubelet可以重启容器, 确保服务的持续性。

(2) 服务发现和负载均衡。a. 定义服务(Service): 在Kubernetes中, 服务(Service)是定义一组逻辑上相关联的Pod访问策略的抽象, 它使得外部访问这组Pod变得容易。b. 服务IP和端点: 当Service被创建时, 它会被分配一个稳定的IP地址(ClusterIP)。Kubernetes内部的DNS服务会将这个Service的名称解析到这个IP地址。同时, Service会监视匹配其选择器的Pod, 并将它们作为端点。c. 负载均衡: 当流量发送到Service的IP地址时, kube-proxy组件负责将流量分发到后端的Pod上。kube-proxy可以采用不同的模式来实现负载均衡, 例如轮询方式。d. 外部访问: 除了ClusterIP之外, Service还可以以NodePort、LoadBalancer或Ingress的形式暴露给集群外部, 从而允许外部流量访问集群中的服务。e. 自动化的服务发现: 由于Pod的IP地址可能会频繁变更, Service提供了一个稳定的访问接口, 而不需要客户端知道后端Pod的具体IP地址。

通过这些机制, Kubernetes不仅能够有效地将容器化应用部署到Pod中, 还能够保证这些应用的高可用性和可伸缩性, 同时简化服务发现和负载均衡的复杂性。

3.2 注意事项

在基于Kubernetes的服务部署中, 每个服务通常会被打包成一个或多个Docker容器^[3], 然后通过Kubernetes进行管理和调度。每个微服务的所有容器实例被封装在一个Kubernetes的Pod中, Pod是Kubernetes的最小部署单元, 它可以保证同一个Pod中的所有容器共享相同的网络和存储空间。服务的部署需要考虑以下几个方面:

(1) 容器化: 服务应该被打包成Docker容器, 这样可以保证服务的可移植性和一致性^[4]。

(2) 声明式配置: 服务的部署状态应该通过Kubernetes的声

明式配置进行描述, 然后由Kubernetes自动地调整系统的实际状态, 使其与期望状态一致。

(3) 自动化部署: 服务的部署应该通过持续集成/持续部署(CI/CD)的方式进行, 这样可以保证服务的快速迭代和高质量。

在基于Kubernetes的微服务架构中, 服务的扩展主要通过增加或减少Pod的数量进行。Kubernetes提供了弹性扩展的机制, 可以根据服务的负载或其他自定义的指标, 自动地增加或减少Pod的数量。服务的扩展需要考虑以下几个方面:

(1) 扩展策略: 服务的本论文详细探讨了基于Kubernetes的微服务架构的实现和优化策略。首先, 论文介绍了微服务架构和Kubernetes平台的基本概念和特点, 然后详细阐述了基于Kubernetes的微服务架构的实现方法, 包括微服务的设计与部署、服务的扩展与弹性、服务发现与负载均衡、容错与恢复、监控与日志管理等方面。接着, 论文提出了一系列优化策略, 包括微服务的细粒度管理、自动化部署与持续集成/持续部署、服务的动态扩展与弹性、高效的服务发现与负载均衡、容错与灾难恢复策略、实时监控与日志管理等。最后, 通过一个实际案例, 论文展示了这些实现方法和优化策略的应用和效果。本论文的研究结果为开发者和系统管理员提供了实用的参考, 有助于他们更好地实现和优化基于Kubernetes的微服务架构。性和资源限制进行设置, 例如CPU的阈值、内存的阈值、Pod的最大数量和最小数量等。

(2) 扩展速度: 服务的扩展速度应该根据服务的变化速度和稳定性需求进行控制, 例如每次扩展的Pod数量、扩展的冷却时间等。

在基于Kubernetes的微服务架构中, 容错主要通过Kubernetes的健康检查和重启策略进行。Kubernetes可以定期地对Pod进行健康检查, 如果检查失败, 就会自动地重启Pod, 从而恢复服务的正常运行^[5]。容错需要考虑以下几个方面:

(1) 健康检查的策略: 健康检查的策略应该根据服务的业务逻辑和性能需求进行选择, 例如HTTP检查、TCP检查、执行命令等。

(2) 重启策略的选择: 重启策略的选择应该根据服务的稳定性需求进行选择, 例如始终重启、失败时重启、永不重启等。

(3) 数据的备份和恢复: 数据的备份应该定期进行, 数据的恢复应该通过Kubernetes的API进行, 例如创建和恢复PersistentVolume和PersistentVolumeClaim。

(4) 服务的备份和恢复: 服务的备份应该通过Kubernetes的API进行, 例如导出和导入Deployment和Service。服务的恢复应该通过Kubernetes的API进行, 例如滚动更新和回滚。

4 实验测试

4.1 pod扩容测试

首先将微服务打包成镜像并上传到docker仓库, 然后通过yaml文件创建一个pod, 该pod用于启动微服务, 通过指令并查看pod状态。创建pod如图1所示:

```
root@k8s-master01 work1# kubectl apply -f springcloud-t.yaml
```

图1 创建pod

查看pod的运行状态如图2所示,查看pod的服务信息如图3所示:

```
root@k8s-master01 work1# kubectl get pods | grep springcloud
springcloud-t-93kef9cek1-1fnva 1/1 Running 0 120s
```

图2 pod的运行状态

```
root@k8s-master01 work1# kubectl get svc | grep springcloud
springcloud-t-93kef9cek1-1fnva NodePort 10.254.22.10 <none> 8001:36666/TCP
```

图3 pod的服务信息

通过HPA实现k8s集群POD的水平自动伸缩,它是基于CPU和内存利用率对Deployment和Replicaset控制器中的pod数量进行自动扩缩容。自动扩容之后查看service这个pod的cpu和内存使用情况,可以看到pod的cpu使用了1m,内存使用了3m。查看pod的资源利用信息,如图4所示:

NAME	CPU(cores)	MEMORY(bytes)
springcloud-t-93kef9cek1-1fnva	1m	3Mi

图4 pod的资源利用信息

通过压力测试工具对k8s进行压测,测试之前查看以下pod的副本数,可以看到当前pod只有一个副本。输入压测指令进行压力测试,如图5所示:

```
root@k8s-master01 work1# ab -c 1000 -n 1000000000 http://172.16.60.120:36666/
```

图5 压力测试

再次查看pod的副本数,可以看到当前pod启动了三个副本,如图6所示:

springcloud-t-93kef9cek1-1fnva	1/1	Running	0	10m2s
springcloud-t-93kef9cek1-e23nw	1/1	Running	0	5m11s
springcloud-t-93kef9cek1-9jlk2	1/1	Running	0	61s

图6 HPA扩容后pod数量

等压力测试结束之后过一段时间,再次查看pod副本数,发现副本数量又恢复到了一个。

4.2 负载均衡测试

当服务出现问题或者服务进行更新的时候,Kubernetes会根据自身负载均衡算法对资源进行调度,从而保证了系统的可用性。

首先关闭一个node1中的pod服务,这时访问页面发现页面报404的错误,如图7所示:



图7 服务中断

这时Kubernetes集群检测到了node1中的pod发生了故障,

于是立即调用负载均衡算法在node2中创建一个pod并运行。当再次访问页面时,发现服务已经可以正常访问。



图8 服务恢复

5 结语

在本文中,我们深入研究了Kubernetes平台的基本概念和特点,探讨了基于Kubernetes的服务部署的实现方法,并通过实验验证了Kubernetes的动态扩容和负载均衡的效果。通过这些研究,我们得出了以下结论和总结:

首先,我们对Kubernetes平台进行了介绍,包括其核心概念和主要特点。我们了解到Kubernetes作为一个开源的容器编排平台,具有高度可扩展性、自动化运维、服务发现和负载均衡等强大功能,为容器化应用的部署和管理提供了便利。

其次,我们详细阐述了基于Kubernetes的服务部署的实现方法和注意事项。

在接下来的实验中,我们测试了Kubernetes的动态扩容和负载均衡功能。通过实验结果的分析,我们验证了Kubernetes在应对不同负载情况下的自动扩容和负载均衡能力,展示了其在提高系统性能和稳定性方面的优越表现。

我们认为在未来的研究中,可以进一步探讨Kubernetes在混合云、多集群管理、安全性和监控等方面的应用和优化策略,以进一步完善容器化应用的部署和管理体系,推动云原生技术的发展和应用。

[参考文献]

- [1]李林峰.分布式服务框架原理与实践[M].北京:电子工业出版社,2016:1-23.
- [2]马希琳.基于Kubernetes容器集群资源调度策略研究[D].西安:西安科技大学,2019.
- [3]王磊.微服务架构与实践[M].北京:电子工业出版社,2015:236.
- [4]张建谢天钧.基于Docker的平台即服务架构研究[J].信息技术与信息化,2014(10):131-133.
- [5]杨鹏飞.基于Kubernetes的资源动态调度的研究与实现[D].杭州:浙江大学,2017.

作者简介:

袁璞(1990—),男,汉族,河南商丘人,从事IT方向的工作。