

基于 Kubernetes 的校园容器云的设计研究^①

谢超群

(福建中医药大学现代教育技术中心 福州 350000)

摘 要 随着高校信息化的不断深入, 高校数据中心在传统虚拟化模式下, 存在服务器资源利用率低, 应用服务部署和迁移困难等问题。本文结合高校数据中心的实际情况, 利用 Docker 容器技术和 Kubernetes 容器集群技术设计了一种高校数据中心容器云。该平台实现了应用服务容器的调度和管理, 快速部署和迁移, 提高了高校数据中心的资源利用率和运维效率。

关键词 Docker Kubernetes; 数据中心; 虚拟化

Design and Research of Campus Container Cloud based on Kubernetes

Xie Chaoqun

(Modern Educational Technology Center, Fujian University of Traditional Chinese Medicine, FuZhou 350000, China)

Abstract With the deepening of information technology in Colleges and universities, the traditional virtualization mode of data center in Colleges and universities has some problems, such as low utilization rate of server resources, difficult deployment and migration of application services. Facing the actual situation of university data center, this paper designs a container cloud of university data center by using Docker container technology and Kubernetes container cluster technology. The platform realizes the scheduling and management of application service containers, rapid deployment and migration, and improves the resource utilization and operation and maintenance efficiency of university data center.

Keywords Docker Kubernetes; Data Center; Virtualization

^①本文系 2017 年度福建省中青年教师科研项目基金项目“基于 Docker 容器技术的校园应用云计算平台”(项目编号: JAT170299)的研究成果之一。

1 引言

随着高校信息化的不断深入,学校的各种业务应用服务的数量和种类不断增多,高校数据中心在传统运维模式下面临着以下的问题:

(1) 应用服务软件环境需重复部署,软件环境准备时间较长

在高校数据中心传统运维模式下,软件环境的准备需经过操作系统的安装、中间件的安装、WEB服务的安装等多个环节,无法实现软件环境的快速部署发布。

(2) 数据中心资源利用率较低

采用传统方式部署应用服务,需重新准备服务器安装操作系统及相应的软件环境,浪费大量的服务器计算和存储资源,无法充分利用数据中心资源。

(3) 应用服务无法根据负载灵活调度

在高校数据中心的某些服务场景下,应用负载会临时增加,传统模式下无法快速部署应用服务来自适应业务负载的临时增长。

(4) 应用服务迁移困难

当应用服务需要从一台服务器向其他服务器迁移时,需重新部署操作系统和软件环境,无法实现应用服务和软件环境的打包快速迁移。

针对高校数据中心所面临的现状,采用Docker容器技术将高校应用服务和相应的软件环境制作容器镜像,达到快速部署和迁移的目的,同时结合Google开源的Kubernetes容器编排技术来实现容器资源的灵活调度和管理。

2 Docker 容器技术简介

Docker容器技术是一种基于Go语言开发的开源容器技术,它基于命名空间将服务器的

资源划分成独立的资源容器,该资源容器包含了进程、网络、IPC管道等资源形成一个类似操作系统的运行容器。Docker容器可以将操作系统的资源划分到相互隔离的资源容器,并可以高效调度不同资源容器所使用的资源,满足容器应用的高效运行的需求。^[1]与传统的虚拟化技术相比,Docker容器的指令执行可以直接运行在本地CPU,不需要通过中间的虚拟化层进行指令模拟或者编译,降低了程序运行中CPU运行的开销,提高了程序指令的运行效率。Docker具有将运行的应用容器打包成镜像的功能,在实际使用过程中,可以将应用容器的操作系统、软件运行环境和应用程序整体打包为一个镜像,推送到Docker的镜像仓库中;当需要重新部署类似的应用时可以直接从镜像仓库拉取所需要的镜像,再利用镜像直接生成应用容器,不需要像传统模式下重新安装操作系统和软件环境,提高了应用服务的部署速度,解决了应用服务迁移困难的问题。传统虚拟化技术由于操作系统和硬件虚拟化的开销,比Docker应用容器要消耗更多的CPU,内存和IO资源,因此可以在同样配置的服务器下,可以部署更多的应用服务,资源利用率更高。Docker采用Union FS分层文件系统,当Docker的镜像进行修改时会在原有的基础镜像层上生成一个新的层次,所有的修改都在新生成的镜像层进行添加,原有的镜像层并不会发生改变。Docker的这个特性使部署新的应用更加简单,不需要从头开始构建整个Docker应用镜像,只需要从Docker仓库拉取一个相似度比较高的Docker镜像,在这个镜像上进行修改扩展就可以部署好自定义的应用服务。^[2]

3 Kubernetes 技术简介

Kubernetes是Google公司用Go语言开发的一个开源容器集群管理平台, 它可以实现Docker容器的自动部署维护、自动调度、服务发现、自适应负载伸缩等功能, 能高效简单

地管理Docker容器应用。Kubernetes集群采用 Master/Node的基本架构, Master主节点负责整个集群的运行和管理, Node从节点主要为容器提供计算资源, 整个Kubernetes的基础架构如图1所示。

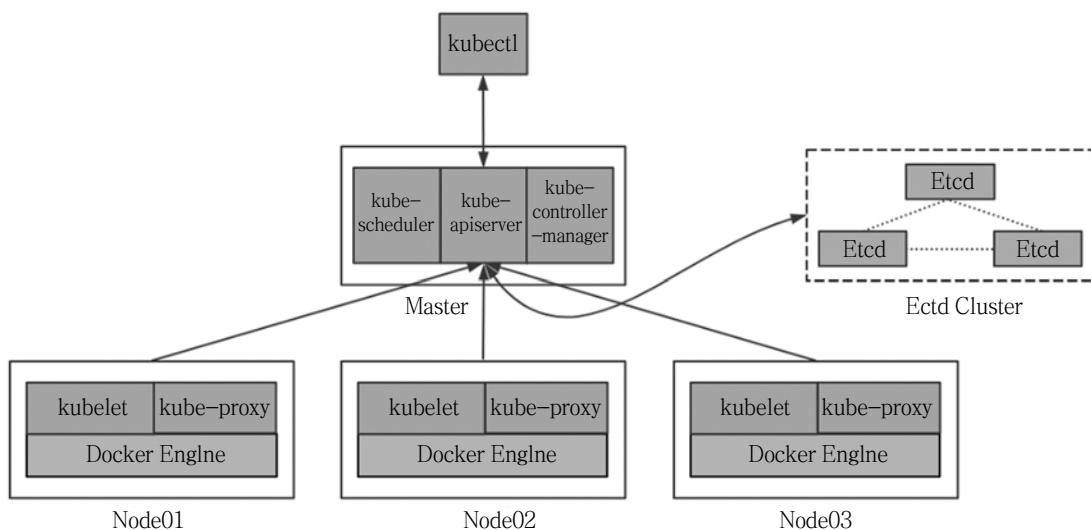


图1 Kubernetes 的基础架构图

Kubernetes的Master节点主要由kube-apiserver、kube-controller-manager、kube-scheduler三大组件构成, 并需要Etcd组件配合来存储整个集群的相关配置信息。kube-apiserver是Kubernetes重要的核心组件, 协调各个组件之间的通信, 它以HTTP API接口服务的方式提供集群资源操作的唯一入口, 其他组件对象资源的增删改查和监听操作需通过kube-apiserver处理后再提交给Etcd存储。^[3] kube-apiserver还提供认证、授权、访问控制、API注册和发现等功能。kube-controller-manager负责监控整个Kubernetes集群的工作状态, 保证集群的各种对象资源处在预期的工作状态, 它可以实现集群资源的故障检测、自动扩展、滚动更新等功能。kube-

scheduler负责调度集群的资源调度, 它接受来自kube-controller-manager的操作请求, 利用集群调度算法将资源对象调度到满足条件的节点上。etcd是一种高效的分布式键值对存储系统, 在Kubernetes中主要用于存储kube-apiserver发送过来的集群持久化配置信息, 如Pod、Service等对象的配置信息。^[4]

Kubernetes的Node节点主要由kubelet、kube-proxy、Docker Engine等组件构成。kubelet负责容器的全生命周期管理, 每个Node节点上都会运行一个kubelet服务进程, 它接受来自Master节点的指令控制容器的启动、停止和回收, 每个kubelet进程会在API Server上注册Node节点自身信息, 定期向Master节点汇报节点的资源使用情况, 并通

过cAdvisor监控节点和容器的资源。kubelet同时也提供集群数据卷和网络的管理的功能。kube-proxy运行在每个Node节点上,监听API Server 中 service和endpoint的变化情况,通过配置节点的iptables来实现Node节点上的容器Pod的网络代理和负载均衡等工作。Docker Engine是安装在每台Node节点上的容器运行引擎,Kubernetes通过Docker Engine在每台Node节点的运行容器镜像来生成应用容器。

4 校园容器云的设计研究

4.1 研究背景

随着高校信息化的不断深入发展,高校数据中心的各种应用服务的数量和种类不断增加,传统的服务器虚拟化的模式下,部署一个应用服务需要安装操作系统、软件环境、应用程序才能完成一个应用服务的部署,部署周期长,运维效率低下。部署完成后如需要

迁移到其他的服务器可能需要重新完成上述的过程,无法实现应用服务和软件环境的打包快速迁移。虚拟化模式下,应用服务的虚拟机个数无法随着负载的变化而实现动态自适应变化,如大学选课系统等应用服务。由于虚拟化模式下每台部署的虚拟机操作系统需占用资源,同时指令运行需要转换和翻译,导致资源利用率低、指令执行效率低下。针对以上传统虚拟化模式下高校数据中心面临的问题,采用Docker容器承载传统的某些虚拟化应用,并结合Kubernetes容器编排技术来实现容器的自适应调度和管理,建设校园数据中心容器云能解决上述问题,提高数据中心的资源利用率和运行效率。

4.2 校园容器云的设计

面对传统虚拟化模式下所存在的问题,可以利用Docker容器技术和Kubernetes容器编排技术搭建学校数据中心的容器云平台。整个容器云平台的架构图如图2所示:

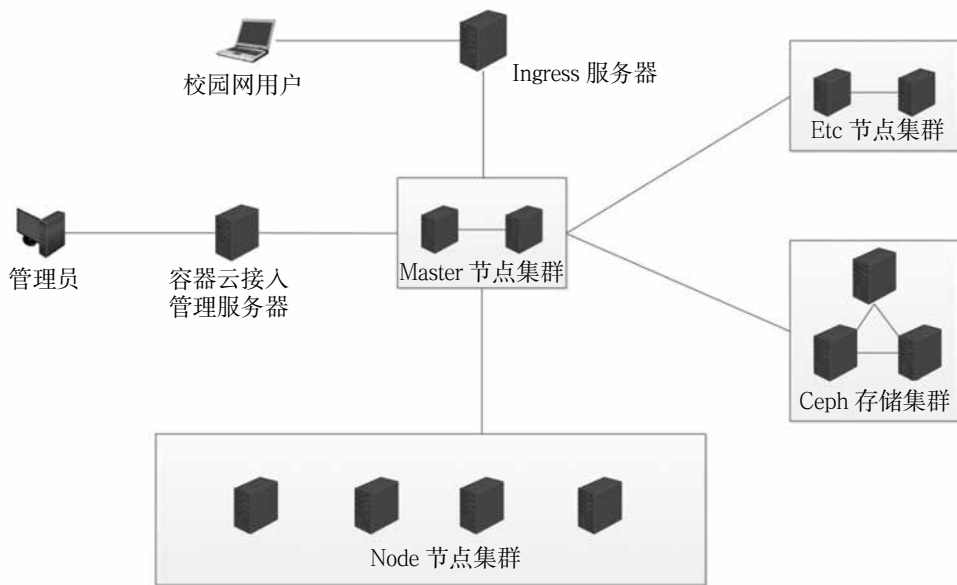


图2 校园容器云体系架构图

整个校园容器云平台,采用Kubernetes容器编排技术的Master/Node的架构来搭建,底层容器化技术采用Docker容器技术。Master节点采用二台服务器部署kube-apiserver、kube-controller-manager、kube-scheduler、Docker Registry核心组件作为Master负载均衡节点集群,保证整个Kubernetes集群的高可用性,同时为满足管理员接入容器终端的要求,将VNCServer远程终端服务端也部署在该服务器集群上。Kubernetes集群的Master节点需要Etcd键值对系统来存储整个集群的配置信息,采用二台服务器配置Etcd键值对系统集群满足高可靠性的要求。Kubernetes集群的后端存储采用三台高速互联的服务器安装ceph块存储服务,对接Master节点的Docker Registry组件为整个容器云平台提供高效、可靠的容器镜像存储服务。Node节点采用四台服务器部署kubelet、kube-proxy、Docker Engine组件,为保证不同Node节点之间的容器网络互通,在每台Node节点上也部署flannel,在每台Node节点上生成一个容器子网,子网之间通过UDP/VxLAN对报文进行封装,从而实现跨主机的容器之间的通信;flannel主要依靠etcd键值对系统来记录容器子网和节点的对应关系,因此还需要将flannel和etcd集群进行相应的对接。^[5] 为便于管理整个校园容器云平台,前端接入管理服务器采用一台服务器作为Kubernetes集群的Node节点,对原生的Kubernetes dashboard镜像进行改写,整合进容器终端组件noVNC,采用改写后的镜像Kubernetes dashboard容器部署在该Node节点上;同时采用NodePort 的方式将容器访问暴露在集群外,集群管理员通过Node ip和暴露的端口就可以访问Kubernetes dashboard集群管理面板,并可以通过noVNC

远程终端访问每个容器的管理终端。校园容器云平台的应用容器对外接入服务,采用一台服务器作为Kubernetes集群的Node节点,部署公共镜像库中的Ingress Controller。Ingress Controller可以将Kubernetes集群的内部容器服务,通过反向代理的方式让集群外部的网络用户可以访问到,从而达到对外暴露内部应用容器服务的目的。Ingress Controller可以监控Kubernetes集群内部的服务的增加或者删除,自动修改内置的反向代理的配置,可以实现服务自动发现和对外暴露。通过Ingress Controller校园网用户可以通过域名的方式方便地访问校园容器云的服务。校园容器云的日志收集采用Deployment的方式在Kubernetes集群部署Elasticsearch日志索引容器服务和Kibana日志分析容器服务,Elasticsearch可以高效索引海量的日志,Kibana具备各种可视化日志分析组件,满足高效便捷的分析集群日志的需求。日志收集的采集端以DaemonSet在每台Node节点上部署Filebeat容器服务作为集群日志的采集端,Filebeat是一种轻量级的日志采集服务,资源占有率低,可以高效快速的采集日志。^[6]

5 结束语

在传统虚拟化模式下,高校数据中心面临着服务器资源利用率低,应用服务部署效率不高等问题。利用Kubernetes搭建的Docker容器云平台能解决上述问题,该平台利用Docker容器技术替代了传统的虚拟化技术,可以提高资源的利用率 and 应用服务的部署密度;同时通过搭建高可靠性镜像仓库,达到了应用快速部署和迁移的目的;最后通过搭建Kubernetes容器管理集群实现了容器的自适应调度和管理,满足了高校实际应用需求,提高了高校数据中心的信息化水平。

参考文献

- [1] 杨保华. Docker技术入门与实战[M]. 北京: 机械工业出版社, 2018: 3—6.
- [2] 华为Docker实践小组. Docker进阶与实战[M]. 北京: 机械工业出版社, 2016: 3—10.
- [3] 龚正. Kubernetes权威指南[M]. 北京: 电子工业出版社, 2017: 12—18.
- [4] Hideto Saito, Kubernetes Cookbook [M]. Packt Publishing, 2018.
- [5] 陆平, 左奇. 基于Kubernetes的容器云平台实战 [M]. 北京: 机械工业出版社, 2018: 73—77.
- [6] 翟雅荣, 于金刚. 基于Filebeat自动收集Kubernetes日志的分析系统[J]. 计算机系统应用, 27 (09): 81—86.