面向高性能计算环境的微服务运维平台设计与实现*

张鼎超[1,2], 王小宁[1], 肖海力[1], 卢莎莎[1], 和 荣[1], 迟学斌[1,2]
(1. 中国科学院计算机网络信息中心, 北京 100190; 2. 中国科学院大学, 北京 100049)

摘 要:国家高性能计算环境为提高应用服务的持续交付能力逐步引进微服务架构。针对国家高性能计算环境由传统单体架构向微服务架构转变引入的新的运维问题,设计并实现了面向高性能计算环境的微服务运维平台,拟面向开发运维人员,降低开发难度,提升运维效率。重点研究并实现了微服务运维平台中的服务部署及管理、服务运行监控和服务弹性伸缩特色功能,通过应用化封装技术对服务部署及管理过程进行封装,同时设计用户权限管理机制,利用 EFK 和 Prometheus 分别完善高性能计算环境的日志收集功能和监控告警功能,通过 Horizontal Pod Autoscaler 资源对象实现基于 CPU、内存等核心指标以及 QPS 等自定义指标的服务规模弹性伸缩技术。测试结果表明,微服务运维平台可以实现高性能计算环境中以项目为划分依据的一键式服务部署、更新、删除等操作,提供交互性更好的可视化运行监控方案,应对流量高峰场景,增强应用服务可靠性。

关键词:高性能计算环境; 微服务; 运维平台; 容器编排; 弹性伸缩

0 引言

国家高性能计算环境,即中国国家网格(China National Grid, CNGrid),起源于国家"863"计划,在国家科技计划持续支持下其资源聚合能力得到了快速发展。目前,国家高性能计算环境的聚合计算资源超过 260 PFLOPS,总存储资源超过 200 PB。国家高性能计算环境基于科学计算中间件(scientific computing environment, SCE)提供计算服务,主要包括作业服务、资源服务以及数据服务,有效地支撑了生物医药应用社区、工业产品创新设计社区、新药创制社区、教育平台的建设,实现服务多样化和专业化,降低高性能计算应用成本,提升高性能计算应用的服务水平,方便用户进行科学计算与研究。随着计算资源的持续接入聚合、用户数量的不断上升、作业量的不断加大,应用服务的持续交付需求也逐渐增强,高性能计算环境的各类服务以及多种应用社区都开始从传统应用架构向结构更加灵活的微服务架构转变。

微服务是一些小而自治服务的统称。相较于传统的单体架构和面向服务架构,应用微服务化具有技术异构性、易于扩展、简化部署、与组织结构相匹配以及对可替代性优化等明显的优势[1]。微服务架构的蓬勃发展离不开底层容器技术的支持,容器是一种轻量级、自包含的软件打包技术,利用容器技术可以实现应用程序的简化部署。

微服务和容器技术的盛行推动着高性能计算环境中的系统服务和社区服务从单体架构向微服务架构形式转变。单体应用按业务领域被拆分为众多细粒度的服务,应用程序的部署迁移变得更加便捷,与此同时也因为容器数量的骤增,服务的管理控制、运行维护也越来越困难。Kubernetes[2,3] 作为 Google 公司开源的一款容器编排引擎,是一个完备的分布式系统支撑平台,其针对容器服务提供了自动化的部署回滚机制,具有透明的服务注册和服务发现能力、灵活的服务扩缩容特性、可配置的负载均衡机制以及强大的故障诊断和修复机制。Kubernetes 和微服务架构相辅相成,促进了高性能计算环境的微服务架构实践落地。

Kubernetes 主要通过命令行客户端或者开源社区提供的 Kubernetes dashboard 提供容器编排管理的服务,使用者需要熟悉容器领域的相关专业知识、Kubernetes 的应用架构,以及该生态环境中庞大复杂的各类工具与插件,不仅需要很高的学习成本、复杂的操作技巧,而且难以满足用户对于微服务架构高效便捷的期望。

为解决上述问题,本文构建了一个面向高性能计算环境的微服务运维平台。该微服务运维平台在业务层面上对服务部署和服务管理进行了进一步封装,屏蔽了 Kubernetes 和容器领域的相关概念,促使开发运维人员更专注于微服务应用自身,通过可视化的交互界面可以实现面向项目的应用服务一键部署和管理,同时集成了日志检索和监控告警技术,并为微服务配置了全面的自动扩缩容功能,使得微服务可以根据 CPU、内存以及用户自定义指标进行自动规模调整。该微服务运维平台可以同时运维管控高性能计算环境的系统服务和社区服务,达到降低技术门槛、提高运维效率、增强用户体验、提升应用服务可靠性的目的。

1 相关工作

目前学术界和工业界都在微服务架构的实践落地方面作出了重要的探索和贡献,尤其是众多企业各自给出了功能完善的微服务架构的落地方案。

微服务是一种从面向服务的体系结构中脱颖而出的体系结构方法,其提倡自我管理和轻量级用于提高软件的敏捷性、可伸缩性和自治性。Jamshidi 等人[4] 从技术和体系结构的角度研究了微服务的发展历程,并总结了微服务架构未来在服务模块化和重构、服务粒度、前端整合、资源监控管理以及服务故障恢复等方面将要遭遇的挑战。

DevOps 是一种旨在减少系统更改和将更改转移到生产环境过程中时间的实践。任何实现这些目标的技术都被视为 DevOps 实践。持续交付(continuous delivery, CD)是 DevOps 的一种做法,通过自动化机制将软件按需部署到任何环境。随着可部署服务数量的增加,CD 成为微服务架构的重要组成部分。Balalaie 等人[5] 通过商业移动后端的体系结构重构和服务迁移解释了 DevOps 在消除开发团队与运营团队之间协调关系障碍、平稳进行微服务架构迁移过程中的重要作用。

Marie-Magdelaine 等人[6] 提出了一个可视化微服务编排框架,该框架提供了一种了解微服务在不同层、生命周期和抽象级别的内部行为的方法。Mayer 等人[7] 提供了一个用于微服务监控和管理的仪表盘,支持集成服务的运行时信息和其他信息源,以提供有关微服务和微服务开发的静态信息。

企业级分布式应用服务(enterprise distributed application service, EDAS)[8] 是阿里云开发的一款应用托管、容器托管和微服务管理的 PaaS 平台,其提供了应用程序开发、部署、监控、运维一系列全栈式解决方案,简化了微服务向云上迁移的过程。EDAS 是一个多样的应用托管平台,用户可以根据具体的需求选择使用 ECS 集群、基于容器服务的 Kubernetes 集群或者是 EDAS Serverless 来对应用进行部署管控,不必去关心底层的基础设施。同时 EDAS 支持丰富的微服务框架,开发人员可以针对原生的 Dubbo、HSF 或是 Spring Cloud 框架对应用进行开发运维,并交于 EDAS 管理。

微服务引擎(cloud service dngine, CSE)[9] 是华为云开发的一款用于企业应用微服务化的解决方案,提供高性能微服务框架和一站式服务注册、服务治理、动态配置和分布式事务管理控制台,帮助用户实现微服务应用的快速开发和高可用运维。CSE 提供了 Java、Go、.NET、Node. js、PHP 等多语言微服务解决方案,支持开源核心框架 ServiceComb,同时基于开源框架 Spring Cloud 和 Service Mesh 开发的应用可以零业务代码修改,直接对接 CSE 运行环境。作为华为核心业务 CloudNative 转型基础底座,CSE 经过了华为终端业务亿级用户考验,因此十分稳定可靠。

京东云微服务平台(JDCloud distributed service framework, JDSF)[10] 是一种托管应用的服务治理框架,其围绕微服务实践落地流程提供了服务部署、注册、调用、日志和监控等生命周期管理功能,同时支持丰富的调用堆栈分析,在宏观上可以为用户提供全

收稿日期: 2020-01-10; **修回日期**: 2020-04-02 **基金项目**: 国家重点研发计划资助项目(2018YFB0204001); 中科院信息化专项课题资助项目(XXH13503-04)

作者简介:张鼎超(1994-),男,山东济南人,硕士研究生,主要研究方向为高性能计算、可视化与网格技术(zhangdingchao@ cnic. cn);王小宁(1981-),女,四川资阳人,副研究员,博士,主要研究方向为网格技术、云服务与分布式系统、高性能计算环境软件与技术;肖海力(1978-),男,湖北天门人,研究员,硕士,主要研究方向为网格技术、分布式系统;卢莎莎(1985-),女,河北饶阳人,工程师,硕士,主要研究方向为网格计算、持续交付;和荣(1988-),女,山东新泰人,工程师,硕士,主要研究方向为网格计算;迟学斌(1963-),男,吉林梅河口人,研究员,博士,主要研究方向为高性能计算、并行计算.

面的服务关系图谱,微观上给出了微服务间的调用链关系。JDSF 目前支持 Spring Cloud、Dubbo 等应用类型,同时兼容 Go、DotNet、Python 等语言的各种开发框架。

2　面向高性能计算环境的微服务运维平台架构

本文提出的微服务运维平台主要面向高性能计算环境中的运维开发人员,由以下三部分技术构成:

a)服务部署及管理技术,用于微服务部署、检索和治理等操作,简化运维人员操作复杂度。

b)服务运行监控技术,用于服务的日志检索和监控告警功能,便于用户追溯定位异常告警。

c)服务弹性伸缩技术,用于微服务根据其核心指标或者自定义指标自动扩缩容,提高微服务应用的可靠性,增强资源利用率。

技术之间交互过程如下:开发人员访问可视化 dashboard,通过服务部署及管理模块将服务以项目为单位部署于微服务运维平台,服务弹性伸缩模块由部署及管理模块获取服务静态信息,同时借助服务运行监控模块采集服务相关指标动态控制服务规模。整个运维平台以 docker 容器的形式运行在 Kubernetes 集群中提供服务,由 Kubernetes 负责运维平台的服务发现、滚动升级和故障恢复等管理控制。微服务运维平台的详细架构如图 1 所示。

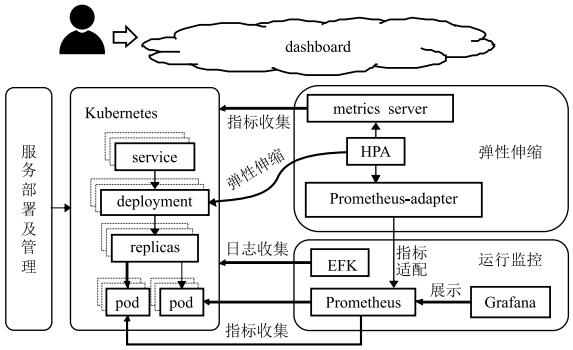

图1　微服务运维平台架构

2.1　服务部署及管理技术

服务部署及管理技术主要提供以下功能:服务部署、服务检索、配置更新、应用更新和服务删除。

原生 Kubernetes 作为一个功能完备的容器编排引擎,可以支持丰富的应用类型及灵活的功能配置,被广泛应用于各种应用的架构转型实践中,与此同时,数量庞大的专业概念、复杂多变的配置设置也大大增加了用户的使用难度。面向高性能计算环境的微服务运维平台旨在降低技术门槛和学习成本,简化开发人员的设计流程,提高运维人员的运维效率。针对高性能计算环境中系统、社区等应用服务的特点,服务部署及管理技术在保持应用功能完备的基础上定制化封装了 Kubernetes API,屏蔽了 service、deployment、configMap 和 horizontal pod autoscaler(HPA)等专业术语,取而代之的则是项目服务、配置文件和服务数目贴近应用服务的概念,很大程度上降低了开发运维人员的学习成本。服务概念对应关系如图 2 所示。

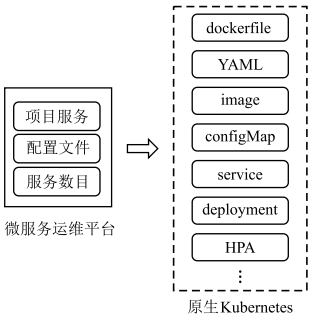

图2　服务概念对应关系

图 3 展示了服务部署的流程,通过定制封装原生 Kubernetes API,用户只需关注应用程序、参数以及配置文件即可实现高性能计算环境中微服务的一键部署及管理。主要定制化实现如下:

a)身份认证和参数检查。目前高性能计算环境中的应用主要为系统服务和社区服务,对于不同的服务项目有相应的开发运维团队进行运行维护,该技术基于 Kubernetes 的 namespace 构建了身份认证功能,设置用户对于不同项目的管理使用权限,增加不同项目应用之间的隔离性;补充了参数检查环节,对用户输入的应用参数进行合法性检查,提高微服务部署成功率。

b)docker image 定制优化。针对高性能计算环境中应用持续交付和配置更新的需求特点,本文搭建了具有漏洞安全扫描功能的本地镜像仓库,极大地提高了容器镜像的安全性以及镜像的传输速度;在构建应用容器镜像过程中,配置应用热更新设置,达到配置文件同步更新的目的;同时以动态可扩展形式构建镜像封装方案,在目前支持 Tomcat、MySQL 和 Spring Boot 应用的基础上,用户可以灵活集成其他应用类型。

c)configMap 定制优化。原生 Kubernetes 提供了 configMap 资源对象管理配置数据,既可以用来保存单个属性,也可以用来保存配置文件,用户通过环境变量或者挂载卷的形式使用,简化了配置文件的更新操作;configMap 同时存在一些固化的弊端,其以挂载卷的形式管理配置文件时,配置文件在容器内是以只读文件系统的形式存在,对于高性能计算环境中的应用服务,并没有加载配置文件的对应权限,导致应用部署失败。该技术针对于此弊端对 configMap 挂载机制进行优化,通过设置软链接避免文件权限的问题,完善配置文件的热更新。

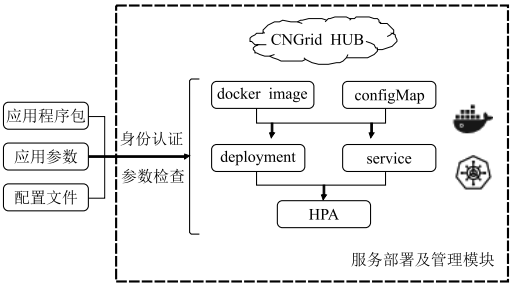

图3　服务部署流程

d)通过官方提供的客户端库,封装定制原生 Kubernetes 检索功能,丰富微服务检索信息;通过上传配置文件,自动更新 configMap,同步更新微服务中配置文件;通过上传应用程序包,自动更新容器镜像,同步更新微服务应用;一键式删除微服务应用对应的 deployment、service 以及配置文件管理工具 configMap,避免复杂重复操作。

2.2　服务运行监控技术

服务运行监控技术主要提供以下功能:

a)灵活的日志收集检索功能。微服务运维平台采用开源日志收集解决方案 Elasticsearch、Fluentd 和 Kibana(EFK)对高性能计算环境中服务日志进行收集、检索和展示。Elasticsearch 是一个实时的、分布式的可扩展搜索和分析引擎,在 Apache Lucene 基础上构建而成,因此在全文搜索方面表现十分出色,同时数据分布在不同的分片中,允许复制进行冗余备份;Fluentd 是一款用于统一日志层的开源数据采集器,允许用户在将日志数据索引到 Elasticsearch 之前,对日志数据进行过滤和转换,添加服务元信息等标签,提高数据检索的便捷性;Kibana 是一款功能强大的数据可视化和管理工具,允许用户通过 Web 界面检索浏览 Elasticsearch 日志数据,同时可以提供实时的直方图、折线图和饼状图。本文提供的基于 EFK 的日志收集架构如图 4 所示。

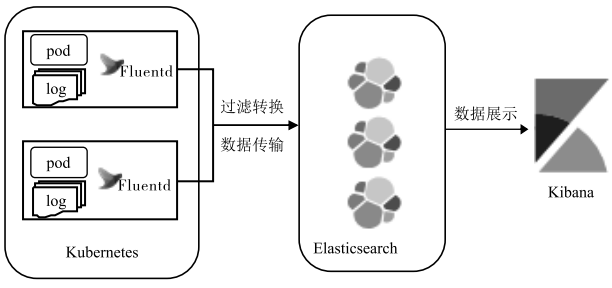

图4　日志收集架构

具体技术实现如下:在 Kubernetes 集群中通过 DaemonSet 资源对象部署 Fluentd 应用,收集每个服务器节点内部存储的容器日志,对日志数据添加项目名称、服务名称等标签,通过制定传输规则将日志存储在全文搜索引擎 Elasticsearch 中,并配有分布式持久化存储冗余备份,同时将可视化工具 Kibana 集成于微服务运维平台可视化界面中用于日志检索。

b)完善的运行监控告警功能。该运维平台集成了开源监控系统 Prometheus,其最初是在 SoundCloud 上构建的开源系统监控和告警工具包,于 2016 年加入 Cloud Native Computing Foundation 成为继 Kubernetes 之后的第二个托管项目。Prometheus 监控方案适用于监控收集时间序列数据,通过 exporter 插件和 Kube-state-metrics 工具采集资源对象的状态指标,在对多维数据收集和查询的方面具有独特的优势。基于Prometheus 的监控告警架构如图 5 所示。

具体技术实现如下:利用 Prometheus 监控 Kubernetes 集群节点以及部署服务的 CPU、内存、网络等核心指标,针对高性能计算环境

中微服务特点,设计监控指标和规则采集用户自定义指标;在 Prometheus server 中制定报警规则,并借助 alertManager 管理告警信息,灵活地选择诸如电子邮件、钉钉等工具进行消息提示;将可视化工具 Grafana 集成到微服务运维平台中用于提升用户的交互体验。

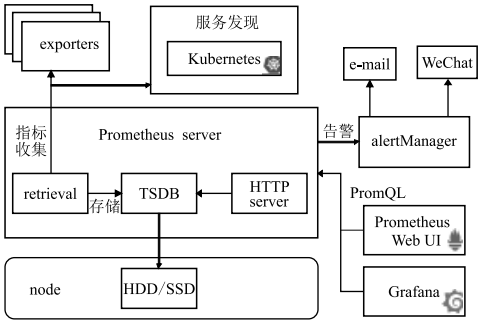

图5 监控告警架构

2.3 服务弹性伸缩技术

高性能计算环境中服务以系统服务和社区服务为主,形式主要为网站和 API 服务等在线任务类型,其对 CPU、内存、网络 I/O 等常规资源消耗较大。服务弹性伸缩技术主要针对上述情况用于高性能计算环境中微服务的自动规模控制。微服务可以依据自身的 CPU、内存等核心指标或者 QPS 等自定义指标进行规模的弹性伸缩,可以有效缓解流量突发带来的访问压力,应对业务高峰场景。

该技术基于 Kubernetes 的 HPA 资源对象实现,HPA 通过周期性的查询机制监控其指定的服务核心资源指标和用户自定义指标负载,根据当前指标和期望指标采用式(1)计算微服务的缩放比例。

$$dR = \text{ceil}[cR \times ((cMV/dMV))] \tag{1}$$

其中:dR、cR、cMV、dMR 分别表示期望服务数、当前服务数、当前指标和期望指标;ceil 表示向上取整函数。HPA 的工作模式如图 6 所示。

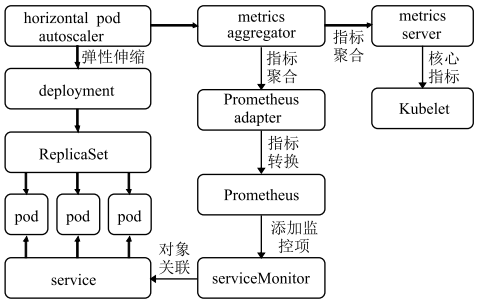

图6 HPA工作模式

服务弹性伸缩技术的实现流程如下:HPA 通过部署的 metrics server 监控微服务的 CPU、内存等核心指标,同时针对高性能计算环境中微服务多为在线任务的特点构建 Prometheus-adapter 并制定指标转换和计算规则,将 Prometheus 采集的用户自定义指标转换为其可以识别的指标,通过多指标监控可以实现灵活的微服务弹性伸缩效果。

3 性能测试

3.1 测试环境

为了验证面向高性能计算环境的微服务运维平台在封装服务部署和管理流程,屏蔽容器和 Kubernetes 相关领域专业概念带来的简便性和易用性效果,同时对构建的微服务运维平台进行弹性伸缩测试,本文搭建了一个单 master 节点的 Kubernetes 集群,将微服务运维平台以 docker 容器的形式部署到 Kubernetes 集群中,用户可以通过可视化的前端界面与微服务运维平台交互。集群配置如表 1 所示。

表 1 Kubernetes 集群服务器配置

节点类型	CPU 数	内核数	内存/GB
master	2	4	8
node1	1	2	4
node2	1	2	4
node3	1	2	4

本文采用开发人员实现的国家高性能计算环境中 portal 接口服务对微服务运维平台中有关服务部署及管理、服务运行监控以及服务弹性伸缩技术相关功能进行性能测试,同时利用 Kubernetes 原生命令行客户端 kubelet 进行对应功能实现以对比效果。

访问微服务运维平台可视化界面,通过客户端上传 portal 应用 war 包和配置文件,同时指定部署服务的项目名称、服务名称、部署数目等基本信息,一键生成可动态更新配置文件的微服务应用,测试得从上传文件到服务部署完成过程中消耗时间平均为 67 s,其中主要为上传文件包消耗的时间,大大降低了高性能计算环境中服务部署的时间成本。

服务部署及管理技术可视化效果如图 7 所示。

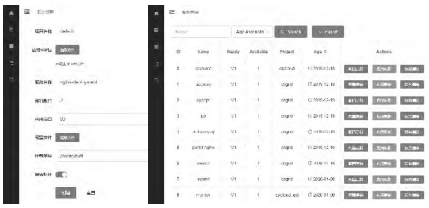

图7 服务部署及管理技术实现效果

访问微服务运维平台可视化界面,通过服务运行监控技术可以利用服务名称、项目名称、时间等关键字检索高性能计算环境中微服务的日志信息,同时查看部署微服务的 CPU、内存、网络等监控信息。

本文使用 Apache 组织开发的压力测试工具 Apache Benchmark 对部署的 portal 接口服务进行压力测试,验证微服务运维平台中服务弹性伸缩技术功能性能。测试过程采用并发数为 100,压力测试总次数为 100 000 次的访问请求,接口服务伸缩指标设置为 CPU,限额为资源请求的 80%,通过微服务运维平台的服务运行监控技术采集服务的负载信息和伸缩信息,测试结果如图 8 所示。从图中可以看出,微服务可以根据弹性伸缩算法应对压力测试,合理调整微服务规模。

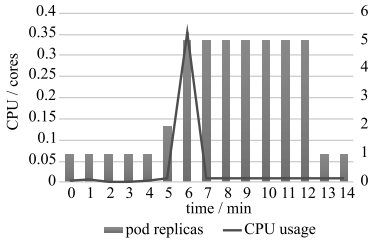

图8 服务弹性伸缩技术功能测试

4 结束语

本文针对高性能计算环境中应用服务的架构转型,为了满足提高开发运维效率、增强持续交付能力、增加应用服务稳定性以及降低学习成本的需求,基于 Kubernetes 构建了面向高性能计算环境的微服务运维平台,根据系统服务和社区服务类型特点定制了服务部署及管理技术、服务运行监控技术和服务弹性伸缩技术。经测试,该微服务运维平台功能与高性能计算环境应用服务需求相契合,降低了操作复杂度,同时保证了应用服务的持续稳定性。但弹性伸缩技术是在保证集群节点资源充足的情况下实现服务的横向扩展,在后续过程中将针对集群资源的纵向扩展加以设计实现,以更加契合高性能计算环境的服务需求。

参考文献:

[1] Newman S. 微服务设计[M]. 崔力强,张骏,译. 北京:人民邮电出版社,2016:3-7.

[2] Burns B, Grant B, Oppenheimer D, *et al*. Borg, Omega, and Kubernetes[J]. Queue,2016,14(1):70-93.

[3] Bernstein D. Containers and cloud: from LXC to Docker to Kubernetes [J]. IEEE Cloud Computing,2014,1(3):81-84.

[4] Jamshidi P, Pahl C, Mendonca N C, *et al*. Microservices: the journey so far and challenges ahead[J]. IEEE Software,2018,35(3):24-35.

[5] Balalaie A, Heydarnoori A, Jamshidi P. Microservices architecture enables DevOps: migration to a cloud-native architecture[J]. IEEE Software,2016,33(3):42-52.

[6] Marie-Magdelaine N, Ahmed T, Astruc-Amato G. Demonstration of an observability framework for cloud native microservices[C]//Proc of IFIP/IEEE Symposium on Integrated Network and Service Management. Piscataway,NJ:IEEE Press,2019:722-724.

[7] Mayer B, Weinreich R. A dashboard for microservice monitoring and management[C]//Proc of IEEE International Conference on Software Architecture Workshops. Piscataway,NJ:IEEE Press,2017:66-69.

[8] Alibaba Cloud. EDAS[EB/OL]. [2019-11-12]. https://www.aliyun.com/product/edas?spm=5176.224200.100.191.7d736ed6sRsQUm.

[9] Huawei Cloud. CSE[EB/OL]. [2019-11-12]. https://www.huaweicloud.com/product/cse.html.

[10] Jingdong Cloud. JDSF[EB/OL]. [2019-11-12]. https://www.jdcloud.com/cn/products/jd-distributed-service-framework.