基于容器化的云边高效应用管控方法及应用

张建文[1],毛正雄[1],姜莹[2],黄世锋[3],张劲松[3],杨本富[3],杨仕军[2]

1. 云南电网有限责任公司信息中心,昆明 650011
2. 云南电网有限责任公司昆明供电局,昆明 650011
3. 云南云电同方科技有限公司,昆明 650011

摘要　随着边缘计算需求激增,云边协同成为未来重要演进方向,云边协同发展的难点之一是对边缘端及运行于其上的应用的部署、管理和更新升级,而容器技术为解决以上问题提供了思路。基于容器技术轻量化、部署快、易移植的特点,研究了基于容器化的云边高效应用管控方法,包括在云边协同的总体架构中增加容器设计、基于容器进行应用协同设计;研究了云端部署更新管理机制;对容器的应用部署及更新、容器及应用下发等设计内容也进行了研究和实现。通过实验验证,该方法在系统能耗、效率、计算时延和云端CPU(中央处理器)使用效率方面得到了显著提高,进而使系统更新速率、服务器可用边缘节点数量以及处理的数据容量都有所提升,能够大幅度对云边协同系统进行优化,同时也大幅度提高了整体系统的计算能力。通过该技术的应用可以改善智慧用电平台边缘侧计算资源紧张、边缘设备异源异构、服务管理需求复杂多样等问题,实现了智慧用电云边协同平台的高效应用管控。

关键词　云边协同;容器设计;Docker;基于容器的应用协同

随着物联网和边缘计算应用场景的日益丰富,边缘计算需求快速增加,云边协同成为未来重要演进方向。在云边协同典型应用场景之一的电力行业,物联网和边缘计算等技术在数字电网[1]建设,特别是智能发电、输变电、配电、用电各环节中都得到了应用落地[2]。在大量终端连接到网络后,实际业务需求面临集中突发增加,此时需要边缘端同步运行多个应用程序。对边缘端及其运行的应用程序而言,实现快速方便地部署、管理、更新、升级成为难点;此外电力行业的不同边缘应用有不同的部署要求,有些要求运行于宿主机,而有些应用要求更好的独立性和隔离性。容器技术因其轻量化、部署简单、规范统一、多环境兼容性、启动迅速、易扩及易迁移等特点为上述难点问题提供了解决思路[3]。

收稿日期:2023-09-13;修回日期:2023-12-19

作者简介:张建文,教授级高级工程师,研究方向为电力信息化、调度自动化,电子信箱: 1254949330@qq.com

引用格式:张建文, 毛正雄, 姜莹, 等. 基于容器化的云边高效应用管控方法及应用[J]. 科技导报, 2024, 42(9): 51-59;
doi: 10.3981/j.issn.1000-7857.2022.07.01102

1 背景

1.1 容器技术

虚拟化交付给用户的资源不是物理实体,而是满足用户资源配置需求的逻辑资源[4]。虚拟化技术经历了多年发展,目前处于容器级虚拟化阶段,Docker是最典型的容器技术之一[5],具有轻量化、部署快、易移植、可弹性伸缩等特点。Docker作为开源的应用容器引擎,开发者可打包其应用到可移植的镜像中,并可发布到任何流行的Linux或Windows机器上,容器使用沙箱机制,相互之间没有任何接口[6-7]。镜像通过文件系统层构造并可共享一些公共文件。在一台机器上运行的多个Docker容器可共享该机器的操作系统内核,容器之间相对隔离,互不影响,可保证各部分应用和服务的独立性,从而确保开发者在对一单体应用程序进行开发或变更时不会影响其他组件部分[8]。容器服务提供了统一的容器运行环境,可为快速积木式应用服务搭建提供平台支撑,并可解决各类应用独立运行和资源高效共享问题[9]。

1.2 云边协同

随着边缘计算技术越来越多应用的落地,云边协同成为备受关注的协同计算形式[10]。云边协同既包含底层算力设施的资源协同,也包含上层数据协同、业务协同、应用管理协同。其中,业务协同主要面向微服务应用场景实现应用在边缘节点的快速部署,在边缘节点创造应用实例,在云节点实现应用分发、编排,以提升边缘应用的部署效率;应用管理协同是指边缘节点提供应用部署环境,并实现对本地应用的生命周期管理。云节点负责应用开发和测试,同时对边缘节点应用的生命周期进行管理。

在工业、能源、交通等典型行业的云边协同应用场景下,往往面临海量设备接入、边缘资源受限[11]、资源严重异构、网络通信质量不稳定、统一运维管理复杂、安全风险控制难度高等挑战,阻碍了云边协同生态和产业的发展。为解决这一问题,本研究将容器技术从云端下沉到边缘侧,以容器化方式实现边缘侧应用及算法的部署和执行[12]。

2 基于容器化的云边高效应用管控

2.1 云边协同总体架构中的容器设计

云边协同主要包括2部分:云端管理平台和边端边缘计算模块。云边协同总体架构中容器的设计内容和功能如表1所示。

表1 云边协同架构中的容器设计内容及功能

部署位置	设计内容	功能简述
云端	云端管理平台	应用镜像构建
	Registry容器镜像仓库	负责容器镜像的存取
	容器镜像仓库	容器镜像管理,与Registry仓库进行集成,实现基于应用包的容器镜像界面化构建、查询等
	组件仓库	将容器镜像注册为组件并添加详细属性及运行的默认配置
	节点组件管理	从组件仓库中选择组件,并设置组件的部署方式(宿主机部署/容器部署/附加至容器部署)及运行参数
	远程边缘组件监控	远程查看容器及APP信息,包括容器及APP列表、版本信息、运行状态
边端	边缘代理程序	Edge Daemon模块:负责接收应用/容器的部署、更新请求,系统升级请求等,来部署、更新应用容器,以及执行系统升级操作
	容器的本地运维及监控	支持配置和修改容器资源,支持容器本地启动、停止、安装和卸载,容器CPU、内存、磁盘使用情况查询,以及资源使用超过设定阈值时告警等
	容器内应用本地运维及监控	支持查询应用软件信息、支持边缘应用本地升级,升级过程中支持断点续传、支持应用软件本地启动、停止、安装、更新、卸载等功能、应用异常退出时自动重启等

2.2　基于容器的应用协同

基于容器的应用协同如图1所示,在应用协同中,将开发好的业务应用上架到应用超市;应用超市负责对应用镜像的更新、发布等进行管理;镜像仓库用于存储具体业务应用镜像,供边缘设备进行拉取;应用协同负责根据实际业务需要将应用超市的应用及其运行、认证信息清单派发至指定边端设备;边缘设备根据云端派发的清单拉取对应的业务应用镜像并对其进行实例化。

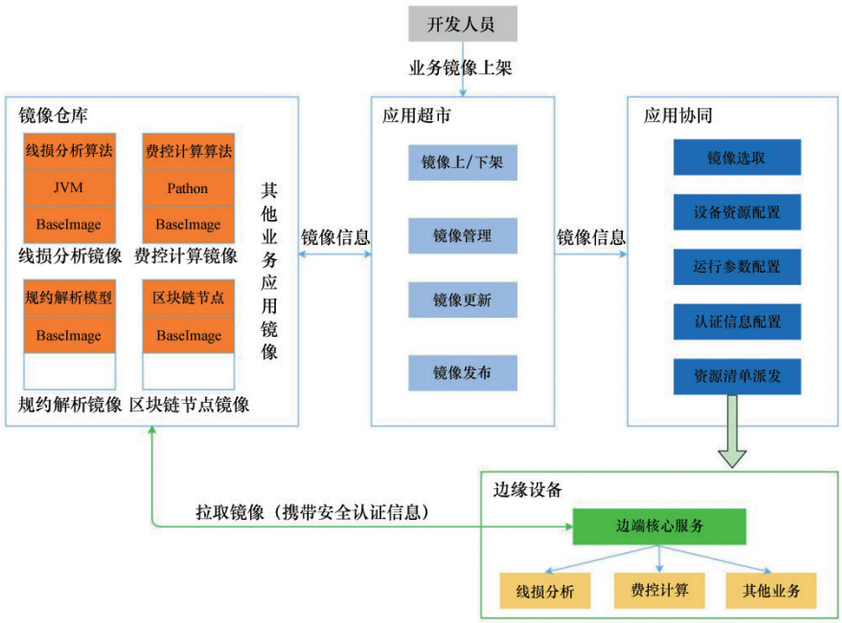


图1　基于容器的应用协同

1）镜像仓库。本方案采用开源Registry构建私有镜像仓库为整个项目提供镜像服务[13-14]。在Registry基础功能的基础上,为了确保私有仓库的安全及可靠性,着重增加如下功能:支持多个身份验证系统、基于职能角色的访问权限控制管理、漏洞扫描功能、日志和审核、可扩展的灵活应用程序接口、直观的用户界面、自动持续镜像垃圾收集、有效利用活动对象资源、无需下载或只读模式。

2）应用超市。应用超市是对镜像仓库的可视化管理,开发人员根据应用超市规则开发相应的应用APP镜像,并将镜像发布到应用超市相关栏目下;管理人员对该镜像进行审核及上架;应用协同人员根据已上架的应用进行对应业务的应用编排管理,应用超市借鉴目前成熟的移动云网平台应用超市模块来实现。

3）应用协同。应用协同是将云端应用镜像下发到边缘设备上实现云端应用的边缘协同[15],应用协同基于开源KubeEdge框架实现[16-17],并增加设备资源配置功能、运行参数配置功能、认证信息配置功能、资源清单派发功能等,使该框架更加满足在电网中的需求。

通过云端应用协同指令的下发,在边缘侧核心服务中主动拉取云端镜像库中对应的手机软件(APP),实现应用的同步协同及更新、删除等操作[18]。

2.3　云端部署及更新管理机制

云端部署及更新的流程包括6个环节(图2):(1)将写好的应用包上传至系统;(2)将应用包构建成镜像置入容器镜像仓库;(3)将应用包直接注册成边缘组件或将构建好的容器镜像注册为边缘应用,供边缘设备使用;(4)创建数字化的边缘设备,并配置需要运行于其中的边缘组件;(5)将配置好的边缘应用清单下发至边缘核心服务,由边缘核心服务依据配置进行部署和更新;(6)边缘管理

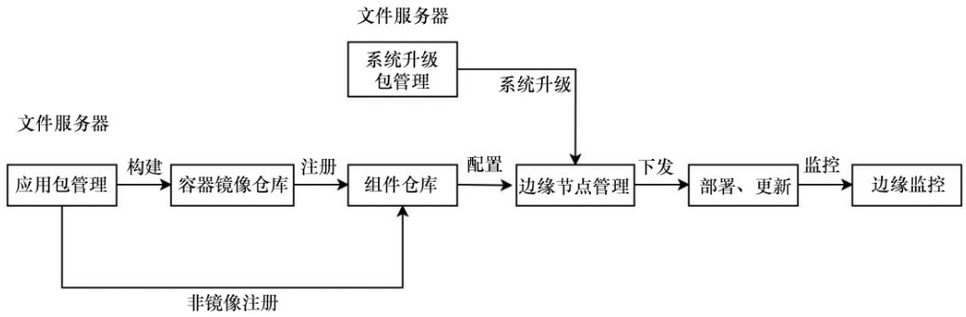


图2 云端部署及更新的流程

平台既可对边缘设备及其中的运行应用程序进行监控,又可支持对系统升级包的管理。边缘管理平台通过配置边缘设备的升级包,下发升级请求至边缘基础组件,从而进行升级操作。

基于云端应用部署及更新机制[19],边缘设备在安装完成后可进行初始化设置,无需本地部署操作,即实现界面化远程在线配置部署应用,可显著降低操作难度和对专业知识的要求。

2.4 容器、应用部署更新设计

应用部署支持3种方式,如图3所示:宿主机部署(非容器部署)、容器部署(单容器单应用)、应用附加至容器部署(单容器多应用)。在页面上可指定应用的部署方式,一次部署命令可包含多个应用的部署,每个应用可以指定不同的部署方式,边缘端按配置的部署方式来部署应用。

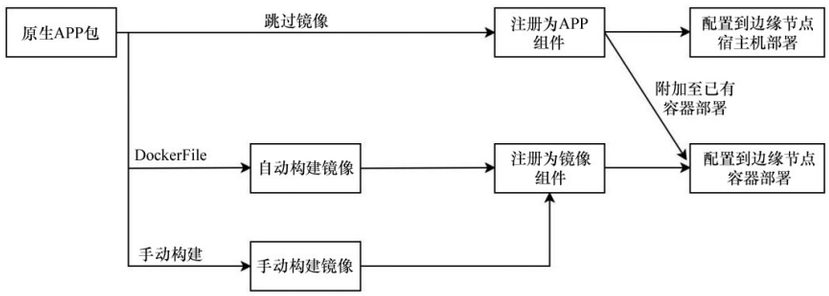


图3 容器应用部署更新流程

容器运行、附加至容器运行、宿主机运行3种部署方式运行参数配置的数据结构保持一致,3种运行方式对各配置项的支持情况如表2所示。

表2 不同运行方式对各配置项的支持情况

配置项/部署方式	容器运行	附加至容器运行	宿主机运行
环境变量	支持	支持	支持
端口暴露	支持	重建容器支持	不涉及
重启策略	支持	支持	支持
文件挂载	支持	重建容器支持	支持
资源限制	支持	超限后重启	支持(Cgroup)
Hostname指定	支持	重建容器支持	不支持(出于安全考虑)
特权模式	支持	重建容器支持	不支持
串口映射	支持	重建容器支持	不涉及
启动参数设置	支持	支持	支持
CMD参数	支持	支持	支持

应用附加至容器内运行的更新流程如图4所示,在更新流程中,附加时如果要重建容器,需保证重建后容器内原始应用不受影响,但应用会重启(图4中蓝色字体部分)。其中,应用文件下载支持断点续传。

应用宿主机更新升级流程如图5所示,在应用宿主机部署流程中,应用文件下载支持断点续传。

2.5　容器及应用下发

如图6所示,红色路径为容器/应用下发处理过程,具体流程如下。(1) 用户在Web页面上发起部署更新流程,预先配置好边缘节点中需要运行的组件清单、部署方式(容器/宿主机/附加至容器内)及运行参数等。(2) 管理平台后台将部署请求发布到MQTT Broker(请求中不包含容器/应用包文件流,仅包含文件下载路径),下载路径中的授权码在1 h后自动过期。(3) 边缘端Edge Hub转发模块订阅到请求后转发给边缘代理模块。(4) 边缘代理模块根据应用/容器的下载路径,从文件服务(镜像仓

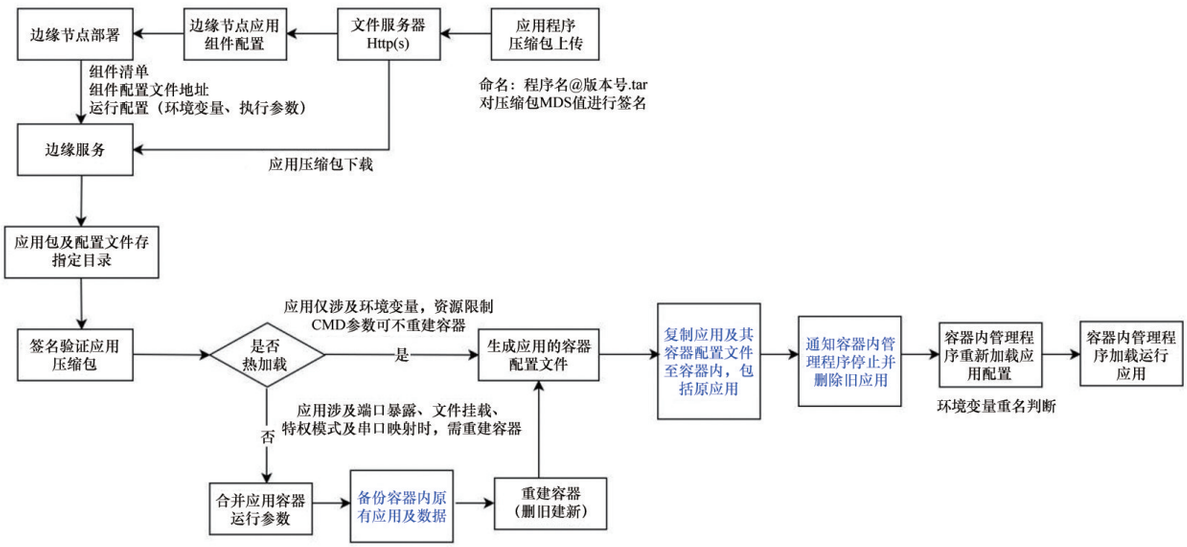


图4　应用附加至容器内运行的更新流程

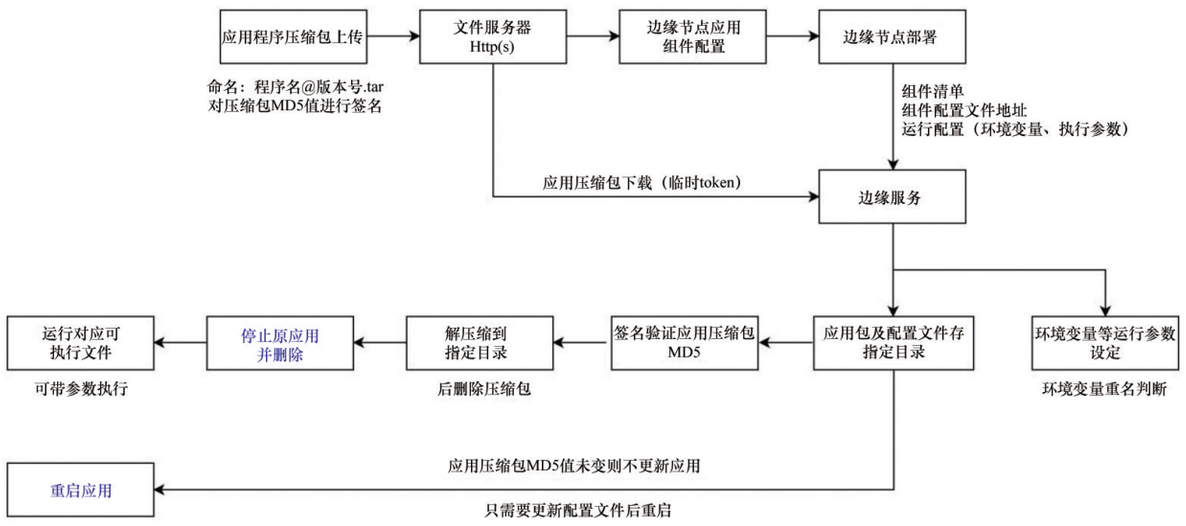


图5　应用宿主机更新升级流程

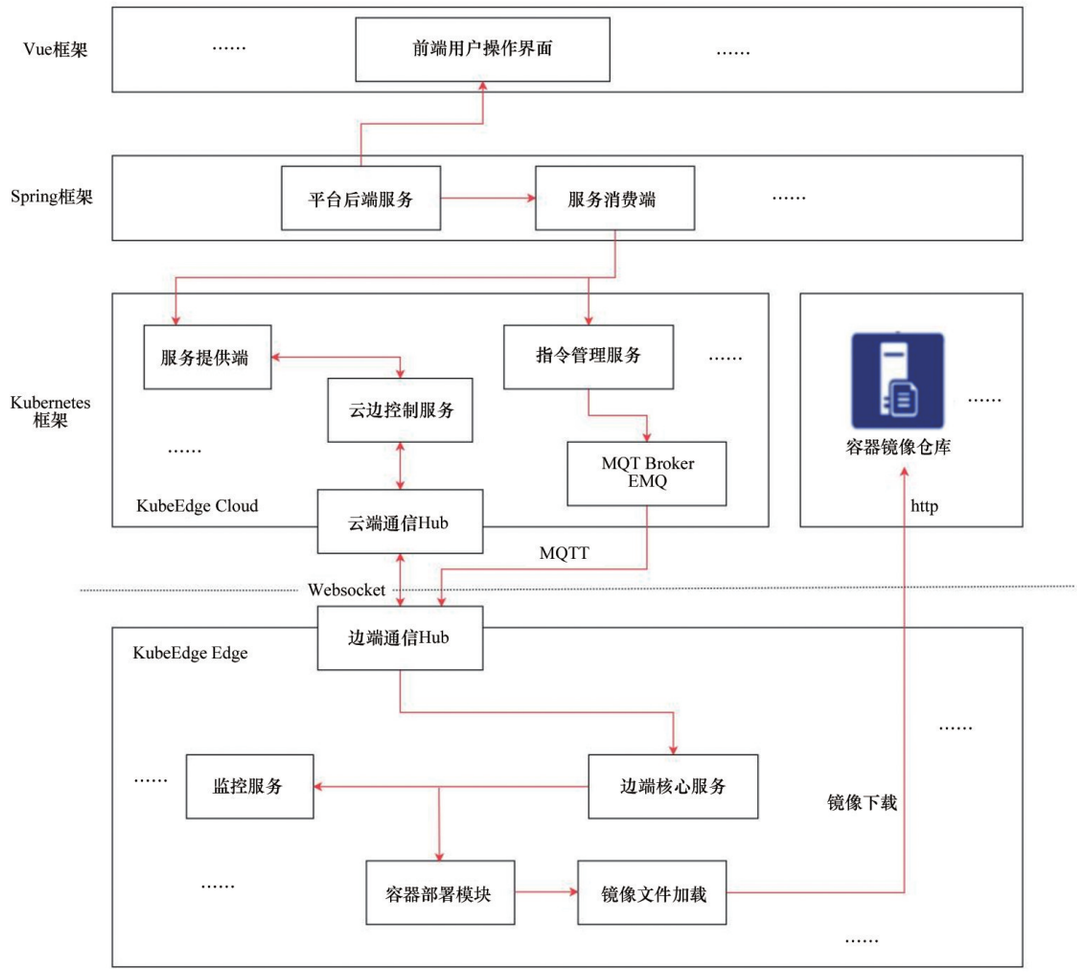


图6 容器及应用下发流程

库)中采用Http(s)协议将文件(包括应用包及挂载文件)下载到本地,并对应用文件进行签名验证,验证通过,则继续更新流程,否则返回错误。(5) Edge Daemon依据请求中的应用清单、容器运行参数来对应用进行部署更新;容器及宿主机应用由Edge Daemon执行部署更新动作,附加至容器中运行的应用由容器内APP管理程序执行部署更新动作。(6) 过程中边缘核心服务会异步上报部署/更新进度及结果信息,返回到页面上展示给用户。

2.6 应用运行参数的传入

整个容器镜像和应用包发布版本后则不允许修改,当镜像或应用包修改后会形成新版本;部署至不同终端要求运行参数不同时,可通过3种方式传送至应用:一是文件挂载方式,即将配置文件挂载至应用内;二是环境变量传入,即应用在运行中通过读取环境变量获取其运行参数;三是应用命令行参传入,即通过在执行应用时传入运行参数。

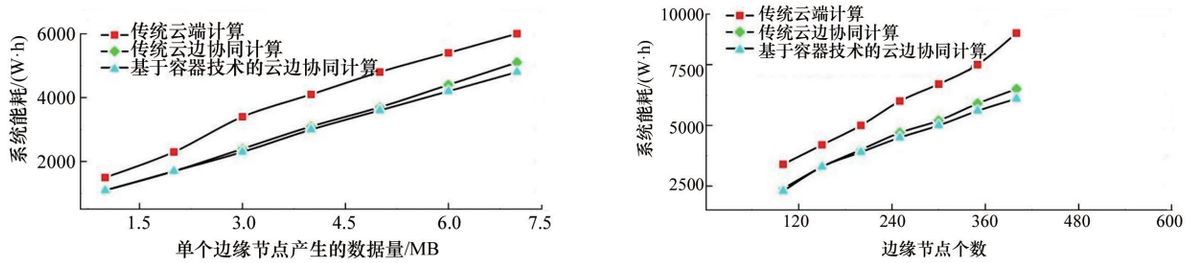
在电力采集系统中,宽带电力线载波(HPLC)台区下的集中器需要每天采集96点实时用电数据用于供电质量相关指标分析,高频数据一般包括电压、电流、功率因数等[20]。集中器根据现场电能表的情况,定期采集电能表中的数据,将采集成功的数据加上时标存储至集中器中,等待主站召测或主动上报。不同台区数据的采集策略各不相同,只能针对不同台区的特性注入不同的数据采集策略,实现运行参数的高效动态注入。

3　测试结果

将提出的技术方案应用于智能电网实验平台进行验证,该实验平台可以模拟1000个边缘节点以内实际电网运行状态并反馈电网参数,其中包括调度中心数据、厂站端数据等。本次模拟所采用的模拟平台搭建于服务器,采用GRAM和GROM。

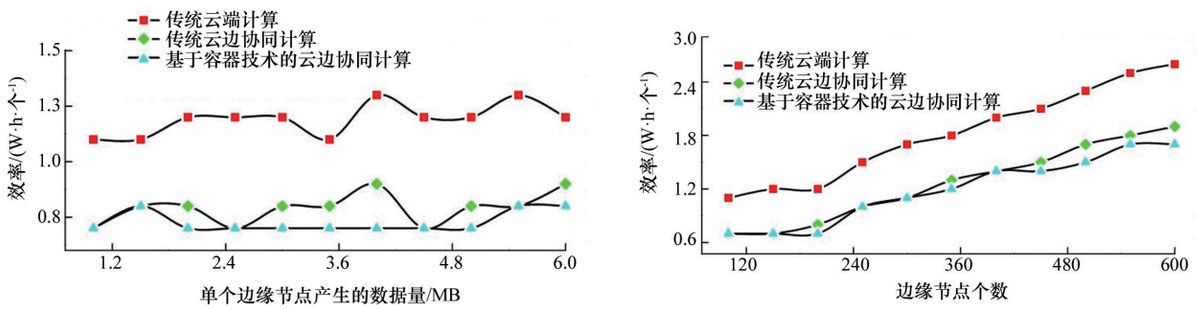
图7、图8是相较传统云端计算,未采用容器技术的云边协同计算和基于容器化的云边协同计算各自在能耗方面的表现。其中系统能耗指计算系统的功耗,效率指平均每条任务所产生的功耗。可以看到,当边缘节点产生不同数据量时,云边协同计算相比传统云端计算的系统能耗大幅下降,效率则有所提升;而基于容器化的云边协同计算相比传统云边协同计算也有小幅提升,这是因为基于容器化的云边协同计算在系统优化上更加出色。

对传统云端计算、传统云边协同计算和基于容器技术的云边协同计算这3种计算方式的计算表现进行对比,结果如图9所示。在不同的边缘节点数以及边缘节点平均数据量情况下,基于容器化的云边协同计算在计算时延上都优于传统云边协同计算,并大幅领先云计算。图10则展示了在不同边缘节点平均数据量情况下,3种计算方式的云端CPU使用率。随着数据量增多,基于容器化的云边



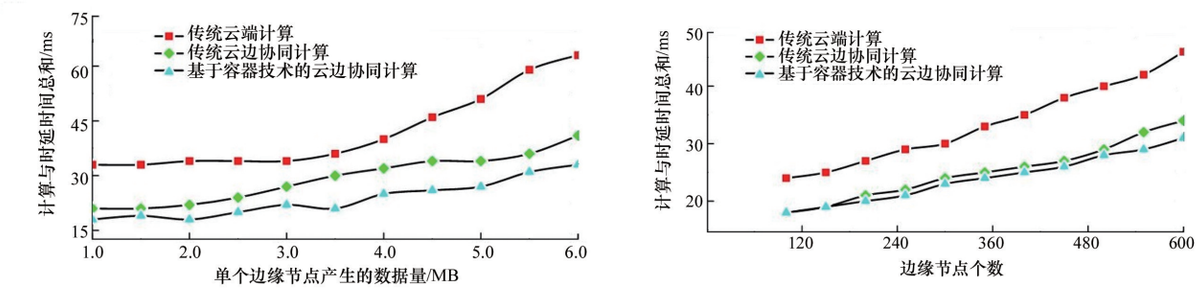
(a) 边缘节点数据量对能耗的影响　(b) 边缘节点数对系统能耗的影响

图7　系统能耗对比



(a) 边缘节点数据量对效率的影响　(b) 边缘节点数对效率的影响

图8　系统效率对比



(a) 边缘节点数据量对时延的影响　(b) 边缘节点数对时延的影响

图9　时延对比

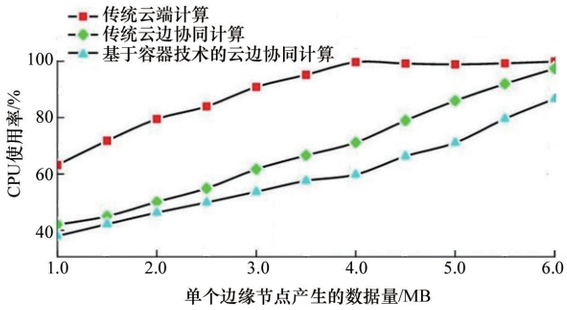


图10 云端CPU使用率对比

应用协同在云端CPU使用率上的优势更加明显,表明该技术能够提升云边协同计算所容纳的数据量。

表3为传统云边协同计算与基于云边容器技术的云边协同计算在平均系统数据更新速率、平均服务器可用边缘节点数量以及平均数据容量3个指标的对比,对比结果显示,本研究方法可使系统更新速率、服务器可用边缘节点数量及处理数据容量都有所提升,可大幅度优化云边协同系统,同时可提高整体系统计算能力。

表3 系统关键数据指标对比

	平均系统更新速率/h	平均服务器可用边缘节点数量/个	平均数据容量/MB
传统云边协同计算	76	212	65
基于容器技术的云边协同计算	19	318	94

4 结论

在智能电网用电场景下将容器技术从云端下沉到边缘侧,主要采用基于开源项目KubeEdge、Registry镜像仓库作为核心基础框架,并结合移动云网应用超市技术实现通用性的协同服务,即以容器化方式实现基于应用包的容器镜像功能,并实现边缘侧应用及算法的部署和执行。不同于传统的应用部署方式,此架构支持多种部署方式,包括宿主机部署(非容器部署)、容器部署(单容器单应用)、应用附加至容器部署(单容器多应用)的混合部署模式。该方法既兼容传统的宿主机部署方式,也支持新型的容器化部署方式。最后通过实验验证表明,基于容器化的云边应用协同系统在系统能耗、效率、计算时延和云端CPU使用效率等方面的性能得到显著提升,从而使系统更新速率、服务器可用边缘节点数量及处理数据容量都得以提升,一定程度上也实现了对云边协同系统的优化和系统整体计算能力的提升。

通过该系统的应用,可有效改善智能电网智慧用电平台边缘侧计算资源紧张、边缘设备异源异构、服务管理需求复杂多样等问题,实现了电网智慧用电云边协同平台高效应用管控。

参考文献(References)

[1] 白浩, 周长城, 袁智勇, 等. 基于数字孪生的数字电网展望和思考[J]. 南方电网技术, 2020, 14(8): 18-24, 40.
[2] 杜羽, 张兆云, 赵洋. 边缘计算在智能电网中的应用综述[J]. 湖北电力, 2021, 45(3): 72-81.
[3] 王骏翔, 郭磊. 基于Kubernetes和Docker技术的企业级容器云平台解决方案[J]. 上海船舶运输科学研究所学报, 2018, 41(3): 51-57.
[4] 耿贞伟, 权鹏宇, 李少华. 基于容器技术的电力系统业务应用资源池系统设计研究[J]. 数字技术与应用, 2017(1): 45-47.
[5] 陈清金, 陈存香, 张岩. Docker技术实现分析[J]. 信息通信技术, 2015, 9(2): 37-40.
[6] 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, 25(8): 138-141.
[7] 李迎. 基于Docker的新一代PaaS平台关键技术研究[D]. 西安: 西安理工大学系统工程学院, 2019.
[8] 刘思尧, 李强, 李斌. 基于Docker技术的容器隔离性研究[J]. 软件, 2015, 36(4): 110-113.
[9] 杨清波, 陈振宇, 刘东, 等. 基于容器的调控云PaaS平台的设计与实现[J]. 电网技术, 2020, 44(6): 2030-2037.
[10] 朱明伟. 云边资源协同技术研究[C]//5G网络创新研讨会论文集. 北京: 移动通信杂志社, 2021: 125-129.
[11] 李彬, 贾滨诚, 曹望璋, 等. 边缘计算在电力需求响应业务中的应用展望[J]. 电网技术, 2018, 42(1): 79-87.
[12] 中国信息通讯研究院. 云边协同关键技术态势研究报

告[R]. 北京: 云计算开源产业联盟, 2021.
[13] 宋胜攀, 刘振慧, 庄东燃. 开源容器技术安全分析[J]. 保密科学技术, 2021(1): 29-35.
[14] 刘亚. 关于Docker镜像仓库技术的研究[J]. 科学技术创新, 2021(29): 76-78.
[15] 许孟豪, 孙美, 林静, 等. 基于云边架构的容器应用协同技术研究[J]. 铁路计算机应用, 2023, 32(4): 7-10.
[16] 黄倩, 黄蓉, 王友祥, 等. 开源边缘计算平台研究分析[J]. 邮电设计技术, 2021(10): 88-92.
[17] 陈卫, 郑炜, 汤毅. 基于KubeEdge的云边协同技术架构的探索[J]. 微型电脑应用, 2020, 36(7): 155-157.
[18] 赵航, 刘胜, 罗坤, 等. 面向KubeEdge边缘计算系统应用研究[J]. 智能科学与技术学报, 2022, 4(1): 118-128.
[19] 郭献彬, 林志达, 曹小明, 等. Docker容器技术的应用部署平台设计与研究[J]. 单片机与嵌入式系统应用, 2021, 21(6): 11-14, 19.
[20] 刘丽娜, 李方硕, 王韬, 等. HPLC应用场景下的台区曲线采集效率提升方法研究[J]. 重庆理工大学学报(自然科学), 2022, 36(8): 263-269.

Efficient application management and control method for container-based cloud edge

ZHANG Jianwen[1], MAO Zhengxiong[1], JIANG Ying[2], HUANG Shifeng[3], ZHANG Jinsong[3], YANG Benfu[3], YANG Shijun[2]

1. Information Center of Yunnan Power Grid Co., Ltd., Kunming 650011, China
2. Kunming Power Supply Bureau of Yunnan Power Grid Co., Ltd., Kunming 650011, China
3. Yunnan Yundian Tongfang Technology Co., Ltd., Kunming 650011, China

Abstract With the surge of demand for edge computing, cloud edge collaboration has become an important evolution direction. One of the difficulties of cloud edge collaboration development lies in the deployment, management, update and upgrade of edge end and applications running on it. Container technology provides ideas for solving the above problem. Based on the lightweight, fast deployment, and easy portability characteristics of container technology, an efficient cloud edge application control method based on containerization was studied, including adding container design to the overall architecture of cloud edge collaboration and using container based application collaboration design; studying the cloud deployment update management mechanism; conducting research and implementation on the design content of container application deployment and updates, as well as container and application distribution. Experimental verification showed that this method could significantly improve system energy consumption, efficiency, computational latency, and cloud CPU utilization efficiency, thereby improving system update rate and increasing the number of available edge nodes on the server and data processing capacity. It could greatly optimize the cloud edge collaborative system and also greatly improve the overall system's computing power. The application of this technology can improve the problems of tight computing resources, heterogeneous edge devices, and complex and diverse service management requirements on the edge of the Power Grid smart electricity platform, achieving efficient application control of the Power Grid smart electricity cloud edge collaborative platform.

Keywords cloud-edge collaboration; container design; Docker; application collaboration based on container

（责任编辑　傅雪）