面向 KubeEdge 边缘计算系统应用研究

赵航[1,2]，刘胜[1]，罗坤[1]，陈世超[1]，孔令辉[3]，贾凡[1,4]

（1. 中国科学院自动化研究所复杂系统管理与控制国家重点实验室，北京 100190；
2. 中国科学院空天信息创新研究院，北京 100094；
3. 中国科学院自动化研究所模式识别国家重点实验室，北京 100190；
4. 墨尔本大学工程学院，澳大利亚 墨尔本 3051）

摘　要：随着物联网的快速发展，传统的以云计算为中心的数据处理方式显现出诸多问题，如带宽占用大、高时延等。而边缘计算以低延时、高可靠的数据处理方式成为云计算的有利补充。主要对面向云边协同的 KubeEdge 边缘计算系统及其应用展开全面的讨论与分析。首先对 KubeEdge 边缘计算系统的架构、功能、关键技术进行了详细的介绍；其次将 KubeEdge 边缘计算系统应用于双臂协作机器人零部件装配的场景，对面向机器人装配场景的 KubeEdge 云边协同系统的功能和性能进行了全面、系统的分析与测试。测试结果显示，基于云边协同的 KubeEdge 边缘计算系统可以较好地满足机器人零部件装配场景的功能和应用需求，为 KubeEdge 边缘计算系统的实际应用提供了基础参考。

关键词：边缘计算；KubeEdge；云边协同；机器人

中图分类号：TP391

文献标识码：A

doi: 10.11959/j.issn.2096−6652.202100

Research on application of edge computing system based on KubeEdge

ZHAO Hang[1,2], LIU Sheng[1], LUO Kun[1], CHEN Shichao[1], KONG Linghui[3], JIA Fan[1,4]

1. The State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
2. Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China
3. National Laboratory of Pattern Recognition Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
4. School of Engineering, University of Melbourne, Melbourne 3051, Australia

Abstract: With the increasing development of Internet of things, traditional data processing methods based on cloud computing have shown many problems, such as high bandwidth occupation and time delay. Edge computing can be a supplement for cloud computing with the characteristics of low latency and high reliability to process data. The KubeEdge edge computing system and its application were mainly discussed and analyzed. Firstly, the system architecture, functions, and key technologies of KubeEdge were introduced. Secondly, the KubeEdge was applied to the parts assembly scenario with dual-arm cooperative robot and the functions and performances of a cloud-edge collaboration system based on KubeEdge for the robot assembly application were analyzed and tested. The experimental results show that the system can satisfy the functional and application requirements of the scenario, which also provides basic reference and guidance for practical applications of KubeEdge.

Key words: edge computing, KubeEdge, cloud-edge collaboration, robot

收稿日期：2020−10−09；**修回日期**：2021−04−01

通信作者：陈世超，shichao.chen@ia.ac.cn

基金项目：国家重点研发计划基金资助项目（No.2018YFB1700602）；国家自然科学基金资助项目（No.61773381，No.61773382，No.61872365）；北京市自然科学基金资助项目（No.4182065）

Foundation Items：The National Key Research and Development Program of China (No.2018YFB1700602), The National Natural Science Foundation of China (No.61773381, No.61773382, No. 61872365), Beijing Natural Science Foundation (No.4182065)

1　引言

云计算[1]的广泛应用深深变革了人们的生活方式，它为交通、制造、农业、城市等提供了智能化的管理方法。同时，基于云计算的计算模式解决了基础计算硬件设施高投入的问题，并且可以为硬件设备提供计算、存储和数据分析的私有/公有云服务。云计算可以充分发挥计算和存储等优势，为资源有限的硬件设备提供快速的计算服务[2]，从而提供更好的用户体验。随着物联网和人工智能技术的快速发展，网络设备数量和智能性需求迅速增加，因此设备产生的数据呈指数级增长，数据的上传和云端数据处理给网络带宽和服务器造成的压力越来越大[3]。在这种情形下，以云计算为核心的集中式数据处理模式将无法高效处理边缘设备产生的数据；在万物互联背景下，传统云计算逐渐显现出不足，具体表现在以下方面。

• 实时性不够。工业互联网、虚拟现实、自动驾驶等场景需要毫秒级的反应时间，而传统云计算模型中，数据传输时延为 25~50 ms[4]，有的甚至更高，这种传输时延很难满足应用需求。

• 带宽不足。边缘设备实时产生大量数据，将数据全部传输到云端会给网络带宽造成巨大压力。

• 能耗较大。云计算中心会消耗极大的能源，工业和信息化部在 2019 年 2 月公布的数据显示，截至 2017 年年底，各类在用数据中心达到 28.5 万个，全年耗电量超过 1 200 亿千瓦时，约占我国全社会用电量的 2%。随着接入网络的设备增多，处理的数据量越来越大，能耗将会成为限制云计算中心发展的瓶颈。

• 对数据安全不利。数据在传输过程中可能会在云端存储时出现泄露，尤其是视频数据等。

为了解决以上问题，面向边缘设备的数据处理范式——边缘计算应运而生。边缘计算是在网络边缘侧执行数据计算与分析的一种新型计算模型[5]。它是在更接近数据产生源头的网络边缘侧融合计算、网络、存储以及应用的分布式开放平台，通过就近提供智能服务满足行业数字化的敏捷联接、实时业务、数据优化、应用智能、安全与隐私保护等关键需求[6]。边缘计算可以实现数据在边缘侧的预处理或完全处理，从而减少了数据传输的带宽压力，满足实时应用对时延的要求。但是边缘设备的计算、存储等能力有限，尤其对于人工智能仍然需要云计算的强大计算能力、存储能力作为支撑。所以边缘计算与云计算互为补充，边缘计算需要云计算中心强大的计算能力和海量存储的支持，而云计算中心也需要边缘计算对边缘设备的海量数据进行处理[7]。因此，边缘计算与云计算需要通过紧密协同才能更好地满足各种需求场景，从而放大边缘计算与云计算的应用价值[8]。

边缘计算要发挥其作用需要相匹配的软件系统作为支持，而国内外涌现了一批侧重于不同功能、适用于不同应用场景的边缘计算平台，如针对物联网应用场景的 ParaDrop、EdgeX Foundry、Apache Edgent、FocusStack、AirBox，针对时延敏感移动应用的 CloudLet、PCloud，针对网络运营商的 CORD、AKraino Edge Stack，针对智能家居的 Vigilia、HomePad 以及针对虚拟现实的 MUVR 等[9]。以上边缘计算平台都在各自领域发挥了巨大作用，本文主要针对基于云边协同的 KubeEdge 边缘计算系统的架构、功能、关键技术及其应用进行详细的讨论与分析。

2　KubeEdge 边缘计算系统

KubeEdge 是一个开源的边缘计算平台，其基于 Kubernetes（K8s）原生的容器编排和调度能力实现了云边协同、计算下沉、海量边缘设备管理、边缘自治等功能，具有完全开放、可扩展、易开发、易维护、支持离线模式和跨平台等特点，KubeEdge 的具体特点如下。

（1）支持复杂的边云网络环境

KubeEdge 采用多路复用的边云消息通道，支持与位于私有网络中的边缘节点进行通信；具有应用层可靠增量同步机制，支持在高时延、低质量网络环境下工作。

（2）应用/数据边缘自治

KubeEdge 支持边缘离线自治，保证边缘离线时的业务运行和故障恢复能力，其主要体现在边缘元数据的持久化与边缘域名系统（domain name system，DNS）解析两方面；KubeEdge 支持边缘数据流式处理，定义了边缘的数据清洗、数据分析等处理工作。

（3）边云一体资源调度和流量协同

KubeEdge 基于统一的 K8s 框架进行扩展，支持边缘节点和云节点混合管理，提供边云数据通信和边–边数据通信能力。

（4）支持海量边缘设备管理

KubeEdge 针对资源受限场景进行自身组件轻量化；KubeEdge 采用可插拔的边缘设备管理框架，支持用户自定义扩展，解耦底层通信协议。

（5）开放生态

KubeEdge 可以 100%兼容 K8s 原生能力，支持用户使用 K8s 原生 API 统一管理边缘应用；KubeEdge 支持多种边缘设备通信协议，支持自定义插件扩展边缘设备协议。

2.1　系统架构

KubeEdge 边缘计算系统主要由 CloudCore 和 EdgeCore 两个可执行程序分别实现云端和边缘端的功能，从而达到云边协同的目的。KubeEdge 边缘计算系统整体架构如图 1 所示[10]。

（1）KubeEdge 云端组件

KubeEdge 云端组件包含在 CloudCore 程序中，主要包括 CloudHub 、 EdgeController 、 DeviceController、Admission Webhook 以及 CSI Driver。CloudCore 作为 K8s 的一种插件，可以通过 List-Watch 机制与 K8s 进行通信，实现无侵入地将原先 K8s 的云端节点管理功能拓展到边缘，KubeEdge 云端组件如图 2 所示。

EdgeController 通过控制 Kubernetes API Server 实现云边节点的应用和配置状态同步。

DeviceController 用于管理边缘设备，确保 Device CRD 信息的云边同步。

CSI Driver 用于同步存储元数据或 Volume 的相关信息到边缘，并代替边缘端作为存储的后端接口，供云上组件 External-Provisioner 和 External-Attacher 调用。

Admission Webhook 是对象校验组件，主要负责对 Device、DeviceModel 等对象以及扩展 API 进行合法性校验，采用容器化形式部署，用户只需以容器的形式运行新版本的镜像。

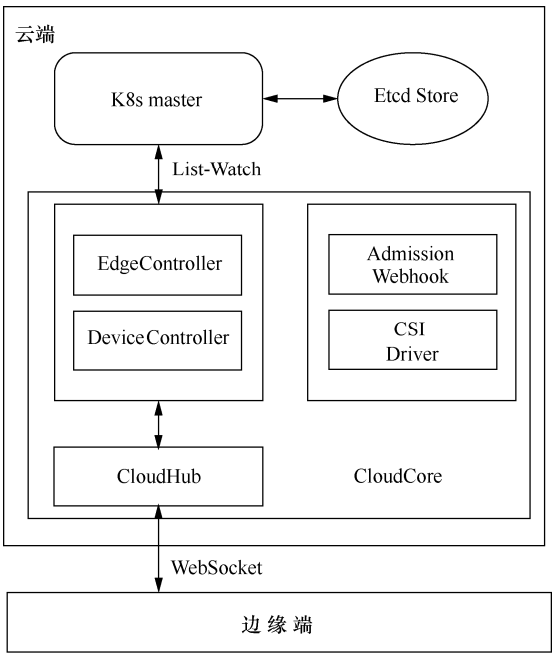


图 2　KubeEdge 云端组件

CloudHub 作为一个 WebSocket 服务端，其主要作用在于监听、缓存和发送云端的数据到

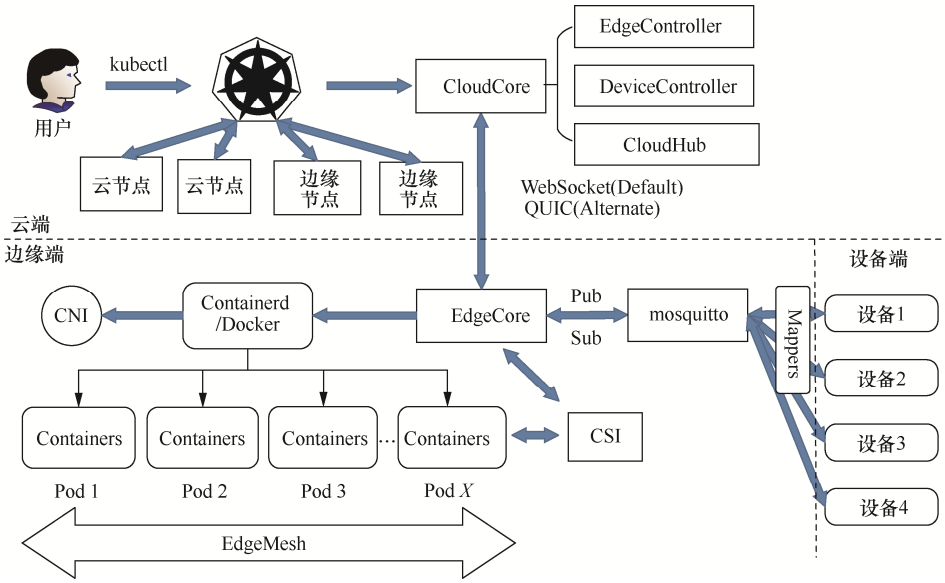


图 1　KubeEdge 边缘计算系统整体架构

EdgeHub。

（2）KubeEdge 边缘组件

KubeEdge 边缘组件包含在 EdgeCore 程序中，主要包括 Edged、MetaManager、EdgeHub、EventBus、EdgeMesh 等，用于应用程序容器化管理，KubeEdge 边缘组件如图 3 所示。

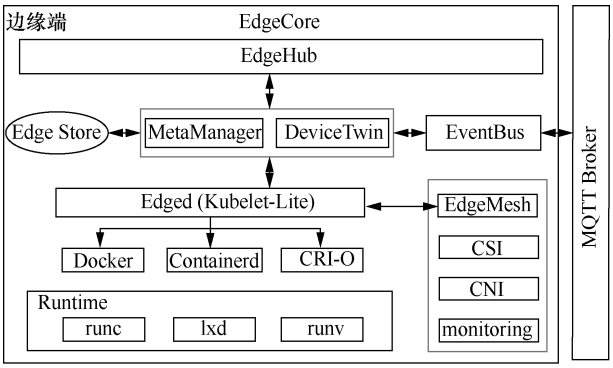


图 3　KubeEdge 边缘组件

Edged 是运行在边缘端的轻量化 Kubelet 组件，用于代替 Kubelet 在 KubeEdge 中实现对 K8s 资源对象的生命周期管理，支持 Docker、Containerd 以及 CRI-O 这 3 种高级容器运行时。

MetaManager 负责本地元数据的持久化，其后端与一个 SQLite 的数据库相连。KubeEdge 边缘端与云端通信的所有数据都会在本地数据库中保存一份。当本地存在数据时直接在本地调用，避免频繁的网络交互，同时在离线情况下本地数据也依旧可以发挥作用。该模块是 KubeEdge 边缘节点自治能力的核心。

EdgeHub 是一个 WebSocket 客户端，负责边缘端与云端的双向通信，在下发云端消息的同时对边缘端各模块的消息进行收集和上传，保证 KubeEdge 云边消息的同步。

EventBus 是一个消息队列遥测传输（message queuing telemetry transport，MQTT）客户端，保证其他组件可以从 MQTT 服务器（如 mosquitto、emqx）进行消息订阅和发布。

EdgeMesh 主要用于服务发现，并可用于边缘端流量的转发。EdgeMesh 内置域名解析能力，从而使边缘节点的域名解析不依赖于中心 DNS。支持容器存储接口（container storage interface，CSI）、容器网络接口（container network interface，CNI）以及监控（monitoring）。

DeviceTwin 用于抽象物理设备并在云端生成一个设备状态的映射，存储并同步设备状态到云。

Runtime 是管理与运行容器的关键组件，支持 runc、lxd 以及 runv 等低级容器运行时。

KubeEdge 的云端是基于 K8s 进行的功能扩展，因此可以无缝兼容 K8s。而 KubeEdge 在边缘端重新开发了节点 Agent，大幅度优化了边缘组件的资源占用，即重新开发轻量化的 Kubelet，实现对容器、Volume、node 等 K8s 资源对象的生命周期管理。云边之间通过 WebSocket 实现信息交互与同步，而通过 MetaManager 和 DeviceTwin 模块，边缘端和设备端的基础数据可以本地化，实现了边缘节点自治。

总之，KubeEdge 通过优良的架构和技术能实现边缘智能、数据边缘计算分析等功能，有效解决了云−边服务的数据协同、任务协同、管理协同、安全协同的问题；同时，通过数据本地化处理、边缘节点离线自治解决了云边之间的网络可靠性和带宽限制问题。另外，KubeEdge 通过优化边缘组件的资源占用有效解决了边缘资源的约束问题，且利用云边之间构建的双向多路复用网络通道有效解决了云端难以管理高度分布的海量节点和设备的问题。同时 KubeEdge 边缘端支持对接物联网主流的通信协议（MQTT、蓝牙（bluetooth）、ZigBee 等），有效解决了异构硬件接入难的问题。

2.2　系统功能

（1）任务编排与调度

KubeEdge 边缘计算系统本身是基于 K8s 扩展的，依托 K8s 的容器编排和调度能力，可将本机容器化应用编排和管理扩展到边缘端设备。

（2）云边协同

KubeEdge 在云端实现对边缘设备和节点的统一管理，一方面可以直接从云端向边缘端下发更新数据与配置信息；另一方面可以在云端对边缘端的各项数据进行实时监控，降低系统的运维难度。KubeEdge 使用的是基于 WebSocket 协议的通信机制，该通信机制比 K8s 的 List-Watch 机制扩展性更好，能够允许海量边缘节点和设备的接入，更适用于解决边缘端网络环境不稳定导致的频繁重连问题。

（3）设备管理

KubeEdge 采用映射器（Mapper）组件来实现边缘设备的接入，Mapper 实际上是用于转换不同设备通信协议从而使之能接入 KubeEdge 的应用程

序，开发者可以根据需要开发设备所支持的协议的 Mapper。目前官方已支持的协议有 MQTT、bluetooth、OPCUA、Modbus、Wi-Fi、ZigBee 等。KubeEdge 通过 DeviceTwin 实现设备状态的更新和同步，并允许用户以 K8s 的扩展 API 对设备进行管理。

（4）边缘节点离线自治

KubeEdge 对边缘端收到的应用、设备元数据进行本地持久化，因此边缘节点能在与云端完全断开连接的模式下自主工作，在网络连接恢复时再与云端进行数据同步。相比 K8s 的 Kubelet 在内存中缓存对象的方式，这样可以有效保证边缘节点在离线、故障恢复时的业务自治和快速自愈。KubeEdge 目前的应用管理操作需要通过 K8s master 进行，在边缘节点离线的场景下，本地持久化的数据仅用于管理节点上的应用和设备，云边通信恢复后节点将根据来自 CloudCore 的最新消息更新本地元数据，从而在满足边缘自治的同时保证边缘应用与云端数据的最终一致。

2.3　关键技术

（1）容器运行时技术

在 KubeEdge 边缘端的 Edged 与 Docker、CRI-O 的通信设计中，对所有相关组件和容器运行时做解耦，通过标准的容器运行时接口（container runtime interface，CRI）可以选择合适的容器运行时。KubeEdge 支持 Docker、Containerd、CRI-O 和 Kata Container 等主流容器运行时，且所有容器运行时的交互都是通过 CRI 完成的。容器可以将应用软件打包成标准化单元，这些标准化单元具有运行软件所需的所有功能，包括库、系统工具、代码和运行时。容器可以对应用程序进行全生命周期管理，并提供标准的应用程序接口便于二次开发与集成应用。使用容器技术可以将应用程序快速部署和扩展到任何环境中，实现跨平台的支持。另外，容器技术与底层共享操作系统，没有管理程序的额外开销，性能更加优良，系统负载更低。

（2）容器编排与调度技术

KubeEdge 构建在 K8s 之上，为网络和应用程序提供核心基础架构支持，并在云端和边缘端部署应用，同步元数据。而 KubeEdge 上众多应用的部署与运行都以容器化的方式进行。K8s 提供了应用部署、规划、更新与维护的机制，实现了容器的编排与调度，并提供了容器网络接口和容器存储接口，通过第三方实现的插件进行使用。K8s 的核心是将调度编排应用容器分布在不同主机，通过 CNI 构建一个能通信的网络，从而实现服务的分布式计算。

（3）边缘端容器存储技术

不同于 K8s 在云上有数据中心，边缘端的存储方式是每个节点都能很容易地访问到存储的后端。为了减少边缘节点与云端 K8s master 的交互，减少交互开销，KubeEdge 利用云端组件 CSI Driver 将自身伪装成存储的后端，实际上存储的后端分布在边缘供 External-Provisioner 和 External-Attacher 调用，调用的请求通过消息发布到边缘从而处理 Volume，这是 KubeEdge 系统实现对容器存储的支持的核心设计。

（4）云边协同技术

KubeEdge 采取云端控制面管理边缘集群的模式，并不局限于 KubeEdge 本身的 EdgeSite 集群，所有标准的 K8s 集群都可以通过 KubeEdge 中 EdgeSite 边缘协同的方式进行管理。在这种方式下，业务逻辑很大程度上会下沉到边缘，云上更多的是一个统一的入口以及与其他云厂商提供服务的联动。因此，EdgeSite 可以将每一个 K8s 本身的 API 作为一个 Deployment 直接部署到某个边缘的集群中。

3　系统应用

在智能制造领域，双臂协作机器人结合视觉传感器可以实现对工业零件的自主识别、定位和装配，从而在特定场合代替人力操作。在传统云计算模式下，视觉检测模型通常部署在远端的云计算中心，工业现场机器人对零部件的识别需要先将视频图像传到云计算中心，然后再将识别结果返回工业现场，这样极大增加了识别时间，严重影响了机器人的作业效率及安全性。边缘计算的低时延和高可靠的数据处理特点可以在机器人装配场景中发挥优势，解决零件识别效率低和时延高的问题。然而边缘计算只能解决边缘设备的局部管理问题，无法获得全局的决策信息，并且由于资源受限其不能为算法模型训练提供有力支持。因此，基于云边协同的应用架构能更好地结合云计算和边缘计算的特点，利用算法模型云端训练、边缘执行的模式，可以充分发挥两者的优势[11]。本文将 KubeEdge 边缘计算系统与双臂协作机器人（以下简称双臂机器人）Baxter 结合，以构建一种云边协同的智能制造服务系统[12]。

3.1　系统架构

针对智能制造中的机器人零部件装配场景，构建基于 KubeEdge 和 Baxter 的云边协同智能制造服务系统，基于 KubeEdge 的工业双臂机器人云边协同应用架构如图 4 所示。

该架构由中心云—边缘节点—边缘设备 3 个层次构成。中心云获取全局信息，利用丰富的计算资源进行算法模型的训练与更新，并定向、定量地将模型统一下发到边缘节点，完成边缘端业务算法模型的差异化部署或更新；同时将边缘节点未能识别的数据信息实时上传至云端，云端利用这些未识别的数据进行算法模型的再训练与迭代。边缘节点一方面对双臂机器人进行直接控制，基于云端下发的算法模型在边缘端进行实时计算，从而完成零件识别与装配的业务操作流程；另一方面可以将本地数据上传到中心云，为云上算法训练提供样本。另外，在边缘端，边缘节点利用本地局域网可以实现区域内的业务协同，提供更灵活的制造业务编排方案。

3.2　系统测试

使用图 5 所示的架构测试基于 KubeEdge 和 Baxter 的云边协同智能制造服务系统的功能和性能，具体测试如下。

在本次系统测试中，中心云由公有云服务器支撑，边缘节点和双臂机器人接入本地局域网，节点与双臂机器人之间通过网关连接，在网络上是互通

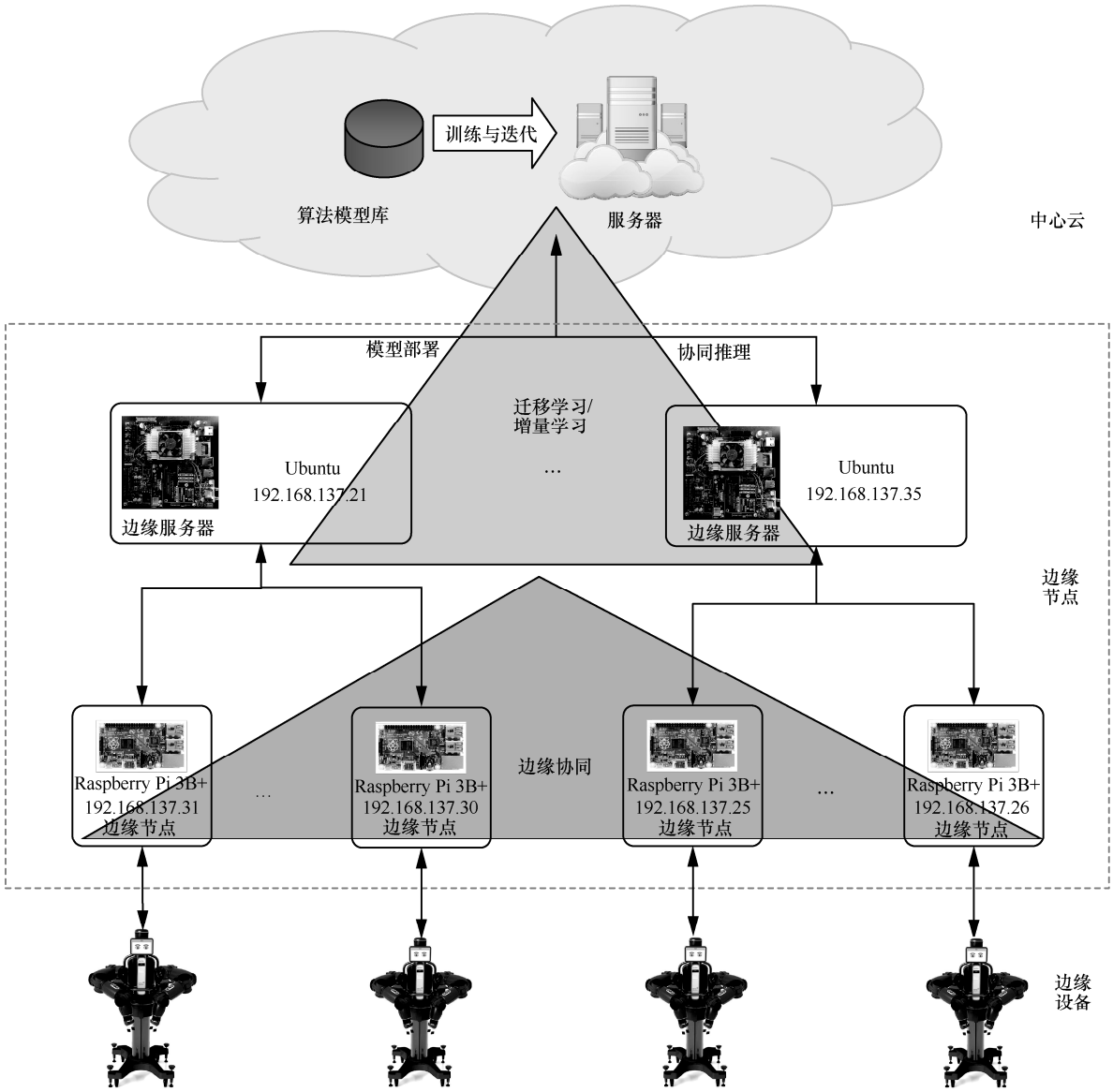


图 4　基于 KubeEdge 的工业双臂机器人云边协同应用架构

的，这可以为边缘业务迁移提供网络条件（后文将详细介绍）。系统测试平台软硬件环境见表 1。

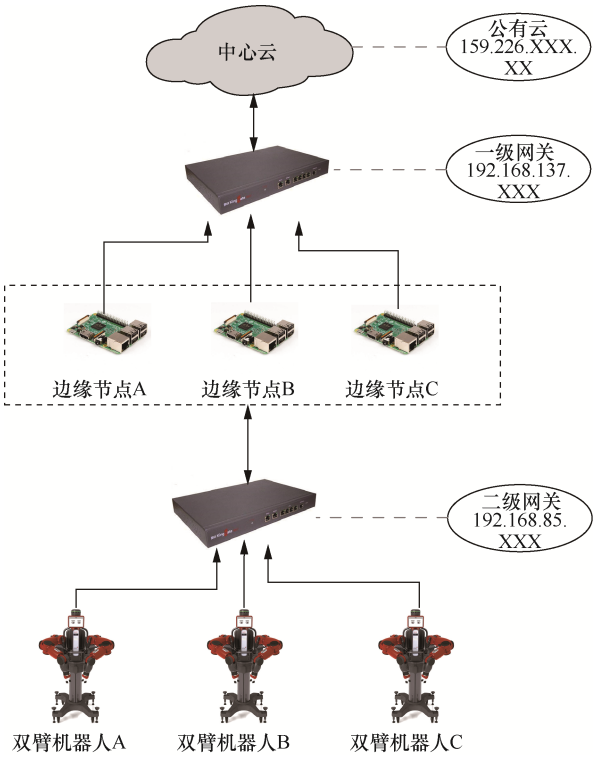


图 5　系统测试平台架构

表 1　　系统测试平台软硬件环境

节点类型	硬件环境	软件环境
中心云	CSTCloud, 16 GB RAM，4 Intel CPU core, 3.50 GHz	CentOS 7 KubernetesV1.18 KubeEdgeV1.5
边缘节点	Raspberry Pi 3B+, 1.4 GHz 64bit, 4 ARM Cortex-A53 CPU core	Ubuntu Mate KubernetesV1.18 KubeEdgeV1.5

3.2.1　功能测试

本节主要介绍对上述应用架构在实际生产过程中可能存在的功能需求进行的定性或半定量测试，以说明该架构在功能上的可行性与有效性。

（1）云边协同作业

基于 KubeEdge 的云边协同架构如图 6 所示，主要业务流程是云端进行算法模型的训练与迭代并下发业务算法模型到边缘端，边缘节点运行部署算法模型的容器使能双臂机器人，最后完成零件的自主识别和装配。

在云−边通信环节，一方面中心云通过云端（CloudCore）下发 AI 模型到边缘节点，边缘节点通过边缘端（EdgeCore）拉取中心云的 Docker 容器镜像来完成模型部署；另一方面边缘节点对数据进行脱敏处理并上传至中心云用于模型训练。在边−端通信环节，双臂机器人将传感器采集的零件数据上传到边缘节点用于 AI 识别，边缘节点根据识别结果下发操作指令，且边−端通信要保证足够低的时延。本项测试任务即测试上述流程在连续作业中的可行性。经测试，从云端模型下发到机器人业务开始运行，所需时间不到 1 min，其中边缘节点拉取 Docker 容器镜像所需时间约 50 s，可见，排除容器镜像下载所用时间后，整个流程的部署可以在很短时间内实现。在业务连续运行的一段时间内进行测试，测试结果显示，双臂机器人每次从零件识别到装配所需时间不到 1 s，KubeEdge 的低时延满足实际的生产业务需求。

（2）边缘离线自治

KubeEdge 边缘离线自治一方面体现在边缘节

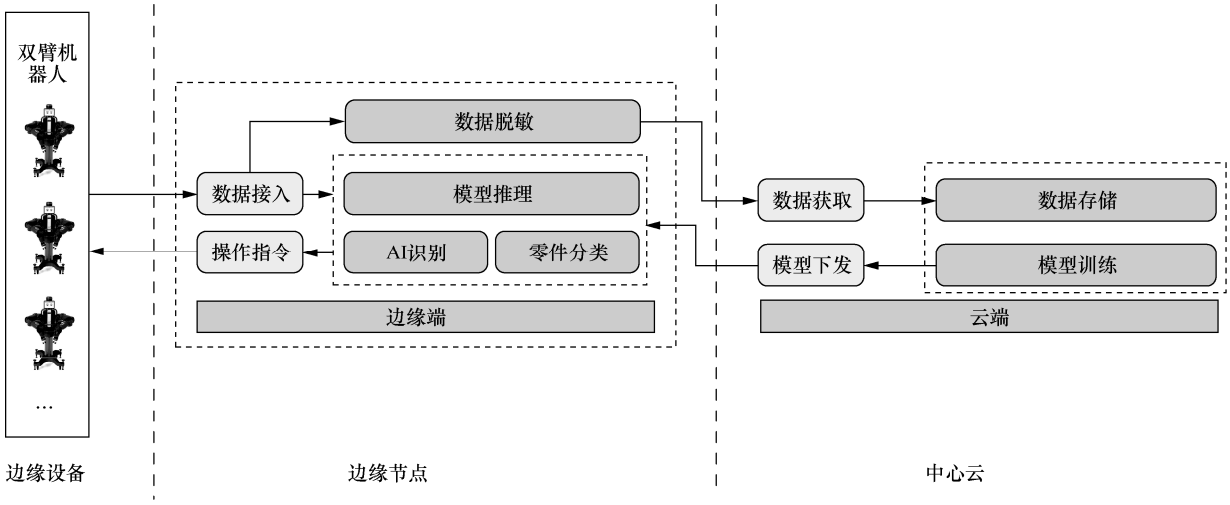


图 6　基于 KubeEdge 的云边协同架构

点在失去与中心云的连接后，可以保持原先业务正常运行不受影响，同时将元数据进行本地持久化存储，在恢复连接后可以利用本地元数据及时同步云边状态；另一方面体现在边缘节点因故障重启后，利用本地元数据可以自主恢复原先运行的业务。针对网络通信断开时的边缘自治能力测试，笔者人为地断开边缘节点与中心云的连接，然后在云端向边缘节点下发新的算法模型，以此观察网络恢复前后的业务运行状态。测试结果显示，断开网络连接后业务正常运行，而在恢复网络连接后，双臂机器人开始执行最新下发的算法模型，测试结果满足预期功能。针对断电重启的情况，笔者人为对边缘节点进行下电上电操作，以此观察机器人的业务执行状态，测试结果显示，重新上电后双臂机器人可正常恢复原先的业务，同样满足预期功能。

（3）边缘业务迁移

中心云在一定时间内无法接收到某一边缘节点的心跳响应（heartbeat response），则会认为该边缘节点为未就绪（NotReady）状态。出现这种情况通常有两方面的原因：一是边缘节点自身宕机；二是边缘节点运行正常，但是无法与云端进行通信。针对上述情况，本文基于 KubeEdge 的任务编排调度功能设计了一种边缘业务迁移策略，具体如下：

- 若当前边缘节点状态为 NotReady，中心云会首先通过其他边缘节点的反馈信息来判断当前节点对应的机器人业务是否还在正常运行（边–边协同）；
- 若机器人业务运行正常，则表明是网络通信问题，此时不影响生产作业，即不进行业务迁移；
- 若机器人业务运行不正常，则判定该边缘节点发生故障，此时 KubeEdge 云端执行业务迁移策略，利用 K8s 的容器驱逐功能（Taint&Toleration），将当前故障节点上的容器部署到同一生产线上的其他边缘节点上，从而驱动与该故障节点对应的双臂机器人，实现业务的连续进行。

边缘业务迁移过程如图 7 所示。通过人为对边缘节点进行断电或断网处理来测试上述功能逻辑。测试结果显示，该设计是可行且有效的，这表明基于 KubeEdge 的云边协同架构可以实现对边缘节点的统一调度管理，对单节点故障有一定的缓冲作用，从而在一定程度上保证了边缘业务的连续性。

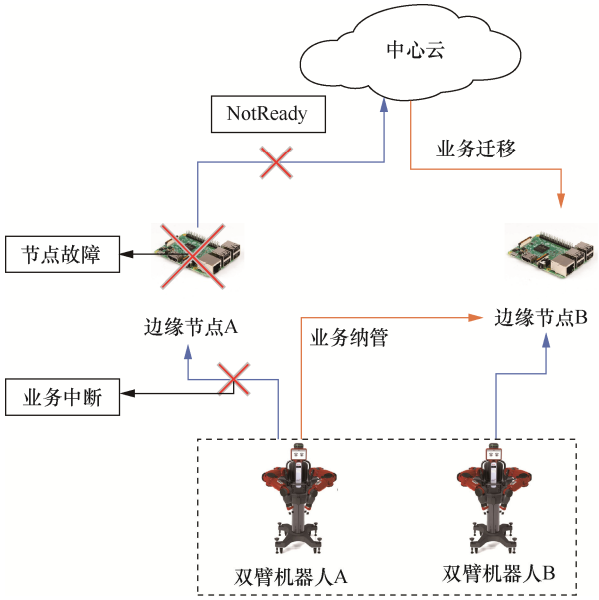


图 7　边缘业务迁移过程

3.2.2　性能测试

结合边缘计算的特性要求，KubeEdge 的系统性能评估可以从通信时延、资源开销、边缘离线自治以及云边协同这几个方面展开。对于 KubeEdge 通信时延评估，参考文献[3]给出了 KubeEdge 在不同网络位置节点之间的通信时延数据，结果表明，KubeEdge 的运行对网络通信时延的影响较小，满足边缘计算的低时延要求，这一点在此不再赘述。本文结合笔者团队的工业双臂机器人的实际应用场景和平台，主要对上述其他 3 个方面，即 KubeEdge 边缘节点资源开销、边缘离线自治能力以及云边协同能力进行定量评估测试。对于边缘节点资源开销评估，本文测试了机器人业务执行过程中边缘节点的主程序 EdgeCore 内存及 CPU 占用率来反映其资源占用情况；对于 KubeEdge 的边缘离线自治能力评估，本文通过测试从边缘节点上电到业务运行所需时间和从边缘节点离线到恢复同步所需时间这两个指标进行说明；而云边协同能力则通过测试边缘节点进行一次业务迁移所需时间进行说明。

（1）EdgeCore 内存和 CPU 占用率测试

通过标准 Linux 命令“ps”来观察 EdgeCore 进程的内存（这里指常驻内存集 Resident Set Size）及 CPU 占用率情况，获得测试结果。3 台边缘节点在无业务运行时 EdgeCore 的内存和 CPU 占用率情况见表 2，部署不同容器个数时 EdgeCore 的内存和 CPU 占用率情况见表 3。

表 2　3 台边缘节点无业务运行时的 EdgeCore 的内存和 CPU 占用情况

边缘节点	内存占用/KB	CPU 占用率
A	60 412	1.5%
B	60 328	1.4%
C	61 345	1.5%

表 3　部署不同容器个数时 EdgeCore 的内存和 CPU 占用情况

容器个数/个	内存占用/KB	CPU 占用率
5	65 132	1.9%
10	69 384	2.0%
15	70 380	2.1%
20	72 076	2.3%
25	73 240	2.4%
30	74 456	2.5%
35	75 900	2.6%
40	76 828	2.8%

进一步测试可知，边缘节点单个容器时平均占用的内存约为 20 MB，但由表 3 测试结果可知，KubeEdge 边缘组件 EdgeCore 的内存占用与容器的资源占用无关，仅与容器规模有关，并且 EdgeCore 的内存占用会随着容器个数的增加而增加，原因是 EdgeCore 需要占用更多内存来实现对不同容器的管理。但其总体内存占用较少，可以在有限内存的边缘计算场景下工作。

（2）从边缘节点上电到业务运行所需时间

该项测试主要针对 KubeEdge 边缘节点上电后边缘端各组件以及业务应用容器的启动时间，目的在于验证 KubeEdge 在边缘节点故障重启后快速恢复业务的能力。通过记录从边缘节点重启到业务恢复所需的时间来获取测试结果。笔者一方面测试了多个边缘节点重启后恢复全部业务所需时间，测试结果见表 4；另一方面测试了同时运行多个容器的单个边缘节点重启后恢复全部业务所需时间，测试结果见表 5。

表 4　多个边缘节点重启后恢复全部业务所需时间

重启的节点个数/个	业务恢复所需时间/s
1	27
2	30
3	32

表 5　同时运行多个容器的单个边缘节点重启后恢复全部业务所需时间

容器个数/个	业务恢复所需时间/s
5	32
10	34
15	36
20	37
25	37
30	38
35	38
40	42

测试结果显示，KubeEdge 边缘端所连接的边缘节点个数和以及边缘节点所运行的容器个数均会影响业务的启动时间。考虑 Raspberry Pi 系统的加载时间，单个边缘节点从故障重启到全部业务恢复所需时间在 30 s 左右，对于同时运行多个容器（承担多个机器人业务）的边缘节点，从重启到全部业务恢复会花费几十秒甚至更长时间。通过进一步测试得知，本实验环境中 Raspberry Pi 系统的平均启动时长为 25 s，可以看到，绝大部分时间花费在 Raspberry Pi 系统的加载上，因此还有进一步优化的空间。

上述实验结果表明，KubeEdge 在边缘节点遇到故障重启时，边缘数据不会丢失，并且原先的业务可以在较短时间内恢复运行，具有较好的离线自治能力。

（3）从边缘节点离线到恢复同步所需时间

本项测试的主要目标是验证边缘节点因网络波动等原因与云端断开连接后，在网络恢复之后能够及时与云端取得通信并同步数据的能力。具体的测试方法为：在断开边缘节点所在网络的网关与云端的连接的条件下，重新恢复连接，然后观察云端数据界面的更新时间戳，并与恢复网络连接的时间点进行对比，从而得到测试结果。

多次测试的结果显示，在网络时延约为 50 ms 的条件下，从边缘节点恢复网络通信到中心云接收到新的数据所需时间小于 2 s（此处边缘节点的数据上报周期为 1 s），并且在断开云端的连接后，双臂机器人可正常运行不受影响。由此可以看到，KubeEdge 不仅可以在离线情况下保证边缘业务的正常运行，网络恢复后还可以在较短时间内与云端取得通信并上报数据，具有较好的离线自治能力。

（4）边缘节点进行一次业务迁移所需时间

对 3 个边缘节点 A、B、C 进行断电操作，中断相应机器人的业务，然后测试经过业务迁移后该双臂机器人恢复运行所需要的时间。

本文共进行了 3 次实验，每次实验分别测试一个边缘节点故障和两个边缘节点同时故障这两种情况下进行业务迁移所需的时间，结果见表 6。

表 6　故障节点数不同的情况下业务迁移所需时间

实验次数	故障节点	迁移节点	业务恢复时间/s
1	A	B	45
	A&B	C	50
2	B	A	43
	A&C	B	52
3	C	A	45
	B&C	A	53

经过进一步测试，从边缘节点故障到云端判定该节点为 NotReady 状态需要 40~42 s，并且可以通过配置特定参数来减少这一时长。

从测试结果可以看出，不考虑节点状态检测周期后，单个边缘节点故障进行业务迁移所需时间在 5 s 以内，而当两个边缘节点同时需要进行业务迁移时，恢复全部业务的时间在 10 s 以上。其原因在于，实验环境只配置了 3 个边缘节点，两个故障节点的机器人业务会同时迁移到第 3 个边缘节点上，而一台机器人的作业需要多个容器（8~10 个）同时运行来完成，故此时第 3 个边缘节点同时运行着 3 个业务所需要的容器。因此，单个边缘节点同时运行多个业务时容器的资源开销较大，容器同步时间便会较长，增加了业务恢复的时间。进一步测试可以看到，负载 3 个业务的边缘节点在运行上有明显卡顿，故实际生产过程中一般会根据边缘节点的资源状况来分配业务，资源有限的边缘节点只会最多额外负载一个容器，从而保证业务的正常运行。

4　结束语

本文对 KubeEdge 边缘计算系统进行了全面的、系统性的分析，从系统的云端和边缘端两个方面对系统架构、核心功能等进行详细分析，并以此为基础对支撑系统核心功能的容器技术、容器编排与调度技术、云边协同技术等进行了详细的阐述。结合 KubeEdge 的云边协同的特性，将 KubeEdge 边缘计算系统与双臂协作机器人 Baxter 结合应用于智能制造中的零部件装配场景，从功能和性能两方面对该智能制造系统进行详细测试与分析。实验结果表明，具有云边协同的智能制造系统可以较好地满足装配场景的功能需求和应用需求。目前测试平台的节点规模还较小，未来的进一步工作是在更大规模节点的环境下进行详细测试。

参考文献：

[1] ARMBRUST M, FOX A, GRIFFITH R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.

[2] KHAN A U R, OTHMAN M, MADANI S A, et al. A survey of mobile cloud computing application models[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1): 393-413.

[3] XIONG Y, SUN Y L, XING L, et al. Extend cloud to edge with KubeEdge[C]//Proceedings of 2018 IEEE/ACM Symposium on Edge Computing. Piscataway: IEEE Press, 2018: 373-377.

[4] VARGHESE B, WANG N, BARBHUIYA S, et al. Challenges and opportunities in edgecomputing[C]//Proceedings of 2016 IEEE International Conference on Smart Cloud. Piscataway: IEEE Press, 2016: 20-26.

[5] SHI W S, CAO J, ZHANG Q, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[6] 李林哲, 周佩雷, 程鹏, 等. 边缘计算的架构、挑战与应用[J]. 大数据, 2019, 5(2): 3-16.
LI L Z, ZHOU P L, CHENG P, et al. Architecture, challenges and applications of edge computing[J]. Big Data Research, 2019, 5(2): 3-16.

[7] CHEN S C, LI Q J, ZHOU M C, et al. Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm[J]. Sensors, 2021, 21(3): 779.

[8] 边缘计算产业联盟, 工业互联网产业联盟. 边缘计算与云计算协同白皮书[Z]. 2018.
Edge Computing Consortium, Alliance of Industrial Internet. White paper on collaboration between edge computing and cloud computing[Z]. 2018.

[9] LIU F, TANG G M, LI Y, et al. A survey on edge computing systems and tools[J]. Proceedings of the IEEE, 2019, 107(8): 1537-1562.

[10] WANG S A, HU Y X, WU J. KubeEdge.AI: AI platform for edge devices[J]. arXiv preprint, 2020, arXiv:2007.09227.

[11] 李肯立, 刘楚波. 边缘智能: 现状和展望[J]. 大数据, 2019, 5(3): 69-75.
LI K L, LIU C B. Edge intelligence:state-of-the-art and expectations[J]. Big Data Research, 2019, 5(3): 69-75.

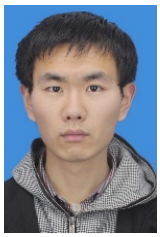
[12] 王大伟, 王卓, 王鹏, 等. 基于边缘计算的云原生机器人系统[J]. 智能科学与技术学报, 2020, 2(3): 275-283.
WANG D W, WANG Z, WANG P, et al. Cloud native robot system based on edge computing[J]. Chinese Journal of Intelligent Science and Technology, 2020, 2(3): 275-283.

[作者简介]

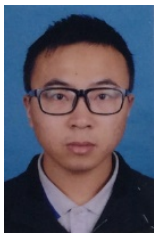
赵航（1999– ），男，中国科学院空天信息创新研究院硕士生，主要研究方向为边缘计算。

陈世超（1987– ），男，中国科学院自动化研究所复杂系统管理与控制国家重点实验室助理研究员，主要研究方向为边缘计算、工业物联网、机器学习等。

刘胜（1978– ），男，中国科学院自动化研究所复杂系统管理与控制国家重点实验室副研究员，主要研究方向为组合优化、建模仿真、切割和布局问题等。



孔令辉（1999– ），男，中国科学院自动化研究所模式识别国家重点实验室硕士生，主要研究方向为类脑计算与脑机接口。



罗坤（1992– ），男，中国科学院自动化研究所复杂系统管理与控制国家重点实验室硕士生，中国科学院大学人工智能学院硕士生，主要研究方向为边缘计算、机器人。

贾凡（1996– ），男，墨尔本大学工程学院硕士生，主要研究方向为边缘计算、物联网、分布式系统等。