Front. Comput. Sci., 2025, 19(4): 194203
https://doi.org/10.1007/s11704-024-3449-x

RESEARCH ARTICLE

ACbot: an IIoT platform for industrial robots

Rui WANG[1], Xudong MOU[1,2], Tianyu WO (✉)[2,3], Mingyang ZHANG[4], Yuxin LIU[1], Tiejun WANG[1,5], Pin LIU[6], Jihong YAN[4], Xudong LIU[1,2,5]

1 School of Computer Science and Engineering, Beihang University, Beijing 100191, China
2 Zhongguancun Laboratory, Beijing 100194, China
3 School of Software, Beihang University, Beijing 100191, China
4 School of Mechatronics Engineering, Harbin Institute of Technology, Harbin 150001, China
5 Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China
6 School of Information Engineering, China University of Geosciences, Beijing 100083, China

© Higher Education Press 2025

Abstract As the application of Industrial Robots (IRs) scales and related participants increase, the demands for intelligent Operation and Maintenance (O&M) and multi-tenant collaboration rise. Traditional methods could no longer cover the requirements, while the Industrial Internet of Things (IIoT) has been considered a promising solution. However, there's a lack of IIoT platforms dedicated to IR O&M, including IR maintenance, process optimization, and knowledge sharing. In this context, this paper puts forward the multi-tenant-oriented ACbot platform, which attempts to provide the first holistic IIoT-based solution for O&M of IRs. Based on an information model designed for the IR field, ACbot has implemented an application architecture with resource and microservice management across the cloud and multiple edges. On this basis, we develop four vital applications including real-time monitoring, health management, process optimization, and knowledge graph. We have deployed the ACbot platform in real-world scenarios that contain various participants, types of IRs, and processes. To date, ACbot has been accessed by 10 organizations and managed 60 industrial robots, demonstrating that the platform fulfills our expectations. Furthermore, the application results also showcase its robustness, versatility, and adaptability for developing and hosting intelligent robot applications.

Keywords IIoT platform, industrial robots, cloud-edge collaboration, intelligent applications

1 Introduction

In recent years, with the advent of high technologies including cloud computing, 5G, Artificial Intelligence (AI), industrial robots are also flourishing. They are widely deployed in factories to do repetitive, monotonous, or dangerous operations such as assembly, welding, spraying, and polishing [1]. Meanwhile, the willingness to use IR in various industries has further increased because of sluggish economic growth, the aging population, and COVID-19. According to statistics from the International Federation of Robotics (IFR), IR installations hit a new record level of 553,052 units in 2022, adding another 5% to installations in 2021. In manufacturing enterprises, a robot is part of a production line and completes a fixed and repetitive process throughout almost its entire life cycle, and the process parameters may change as it ages. Therefore, the most concerning issue in the industry is their O&M, including intelligent maintenance, process parameter optimization, and sharing of knowledge. Making IRs more intelligent, efficient, reliable, and easier to maintain with the help of next-generation Information and Communication Technologies (ICTs) is a key issue.

Specifically, complex, large-scale, and multi-party IR intelligent applications pose the following challenges. On the one hand, traditional O&M methods have to be broken through to reduce downtime, improve process quality, and lower labor costs. For example, preventive maintenance may be unnecessary and costly, process optimization based on the Six Sigma strategy may be inaccurate, and knowledge sharing through documentation and expert experience is inefficient. Some researchers have proposed predictive maintenance methods [2,3], machine learning-based process optimization methods [4,5], and knowledge graph construction methods [6,7]. All of these methods utilize large amounts of historical data to build intelligence applications, but less attention is paid to data collection and storage, application deployment, and continuous improvement of models. On the other hand, parties related to IRs including manufacturers, users, and research institutions all hope to establish a multiple-tenant mechanism to strengthen collaboration while ensuring data isolation, thus achieving economies of scale. Since China is by far the largest IR market, this demand from related Chinese organizations is particularly strong. Therefore, there is a need for a novel robotics paradigm that can meet the requirements for intelligent IR O&M service development and deployment in multi-tenant scenarios.

Received May 31, 2023; accepted January 8, 2024

E-mail: woty@buaa.edu.cn

In recent years, some scholars have successively proposed Networked Robotics (NR) [8,9], Cloud Robotics (CR) [10−12], Fog Robotics (FR) [13−15], and Edge Robotics (ER) [16,17] paradigms. Among them, RoboEarth [18] and Rapyuta [19], as representatives of CR, invoke cloud technologies to access powerful computing, storage, and communication resources. They can process and share information from various robots or agents (other smart objects, machines, humans, etc.). Unfortunately, CR is a centralized computing paradigm, and it is hard to meet the high bandwidth, ultra-low latency, and privacy-sensitive requirements of some industrial scenarios like equipment monitoring and real-time manufacturing. Therefore, FR and ER come into being, which deploy decentralized fog or edge servers closer to robots based on cloud computing. However, for Small and Medium-sized Enterprises (SMEs) related to IR, building a cloud platform is costly and lacks relevant technical staff. Therefore, the three-layer architecture of robot-fog/edge-cloud still has a gap to large-scale applications. Besides, FR and ER ignore the collaboration of different types of IR organizations and miss the knowledge sharing and economies of scale brought by a robotics ecosystem consisting of multiple tenants.

To address the aforementioned challenges, IIoT introduces the Internet of Things (IoT) technology into manufacturing. It connects asset management with business processes through information systems, achieving integration of ICTs and Operational Technology (OT), and has been considered a promising solution. Until 2022, there are about 78 well-known general-purpose IIoT platforms, such as Google Cloud IoT, AWS IoT, Alibaba Cloud IoT Platform, and Huawei Cloud [20]. However, these general-purpose IIoT platforms are not suitable for direct application to IR because of IR's programmability, high flexibility and degrees of freedom, and intelligence. Technologically, the consumer Internet/IoT and the IIoT are more different than they are similar, such as reliability, adaptability, system value, and the number of devices [21]. The production characteristics of each industry in the industrial field (e.g., automotive, electronics, metallurgy, and chemical industry) are very different. It is unrealistic to solve the digital transformation of each industry through one big platform that is similar to the consumer Internet/IoT. For IIoT, we should advance separately through vertical applications for important industries. Currently, there are some IIoT platforms for vertical application areas, such as energy [22], mining [23], building [24], and metallurgy [25]. However, there is no relatively mature IIoT platform for IRs that are key devices in industrial manufacturing.

To this end, this paper presents a holistic IIoT platform for IR: ACbot[1)]. Note that this article covers and extends our preliminary work [26] in which we present the original architecture of the IIoT Platform for IR. The key strength of ACbot lies in that it is, to our knowledge, the first holistic solution for large-scale IR management based on IIoT. ACbot addresses the problems of multi-tenancy, device heterogeneity, intelligent service hosting, etc., achieving various IR-related organization accession. It manages IR knowledge, enables AI-based manufacturing process optimization, accelerates the transition from preventive maintenance to predictive maintenance, and ultimately drives the establishment of a robotics ecosystem. Specifically, we follow the Unified Modeling Language (UML) class notation to define the information model for multi-tenant-oriented data within ACbot. Next, the cloud-edge-device three-layer functional architecture was designed and implemented with containerization technologies which mediate the resources across the cloud and multiple edges consistently. In the functional architecture, the deployment & operation service engine schedules or scales our microservices seamlessly on the virtual computing environments with asymmetric resources between edge and cloud. Furthermore, we propose an implementation architecture including data flow and control flow for intelligent applications of IR. We develop four vital applications based on it: real-time monitoring, health management, process optimization, and knowledge graph. Finally, the ACbot platform is validated in real-world application scenarios. It's designed to be the stepping stone for developing similar intelligent applications. We summarize our contributions as follows:

- ACbot, the first IIoT platform for IR with a multi-tenancy-oriented information model and a three-tier cloud-edge-device architecture.
- A cloud-edge collaboration mechanism implemented by a containerized solution, microservices service, and orchestration engine.
- Above the architecture, we have developed real-time monitoring, health management, production process optimization, and a knowledge graph as four intelligent applications for IR.
- We apply the ACbot platform in real-world scenarios for validation. It accesses 10 companies and academic institutions, manages 60 IRs, collects about 100 billion time series points from devices, and provides the above-mentioned services for them.

The remainder of the paper is organized as follows. Section 2 reviews the state of the art in this research area. Section 3 looks into the IIoT platform for IR, including its information model, functional as well as implementation architecture, and four typical real-world IR-oriented intelligent applications. After illustrating the validation of the ACbot in Section 4, we conclude this paper with some final remarks and conclusions in Section 5.

2 Related work

The present work is based on two different lines of research, IR intelligence and the implementation of IIoT platforms. Consequently, an overview of both lines is put forward in this section.

2.1 Industrial robot intelligence

Traditional IRs require preventive maintenance [2,3] to keep

[1)] The name comes from the Institute of **A**dvanced **C**omputing Technology's industrial Ro**bot**s IIoT platform (ACbot), at Beihang University, China.

their normal condition, process optimization through traditional statistical learning methods such as the Six Sigma strategy [4,5], and knowledge sharing through documentation and expert experience [6,7]. These methods are feeble and cumbersome, making it difficult to cope with today's large-scale robot applications. During the past decade, research on integrating various emerging ICTs in IR has been booming to improve IR intelligence. In the context of machine learning and Industry 4.0, [27,28] systematically summarizes the transition from preventive to predictive maintenance approaches for industrial equipment. [2] proposes to build a regression model between current data and externally measured accuracy errors of the robot. Then predictive maintenance is performed by predicting the change in robot accuracy error. [3] creates industrial robot simulation models using digital twins and enriching the degradation curves with historical data. For process optimization, [4,5,29] review machine learning and big data applications in optimizing products and production processes to break away from Six Sigma strategies in the manufacturing industry. For knowledge sharing, [6] proposes a robotic knowledge graph to assist those who lack the relevant knowledge in disassembly tasks. [7] proposes a fault diagnosis knowledge graph using the historical maintenance logs of the robot transmission system. However, the above methods are only proven to work in a few isolated cases, without an environment for deploying applications or continuously improving the performance of the models as well. As a result, it is risky to apply them widely in real-world industrial production.

To address the above challenges, many novel robotics paradigms have been proposed, such as Networked Robotics [8,9], Cloud Robotics [10−12], and Fog/Edge Robotics [13−17]. The concept of NR was first introduced by [8], which views robots as a collection of resources on a network to have robots and environment sensors communicate and cooperate through a distributed architecture. Even though these efforts have expanded the ability of standalone robots, they remain insufficient for fully supporting increasingly complicated robotic functions. To this end, CR [10] suggests that robots could employ cloud services such as databases and elastic computing services to store and process data from robots and sensors. Moreover, the cloud-based knowledge sharing between different robots makes it possible to build lightweight, low-cost, and smart Robot applications [11,12]. RoboEarth [18] provides a networked information repository to speed up the robot's learning and adaptation, and robots connected to it can perform meaningful tasks that were not explicitly planned in design time. Based on the RoboEarth engine, [19] designed an open-source cloud robotics platform Rapyuta. Rapyuta allocates each robot with a secure computing environment called the rectangular box, enabling the robot to migrate heavy computing tasks to the cloud. Google Cloud Robotics Platform combines AI, robotics, and cloud technologies to achieve an open cloud robotics ecosystem. While CR satisfies the requirements of compute-intensive tasks, it ignores the time-sensitive and privacy-sensitive tasks that are common in industrial scenarios. In addition, these robotics cloud platforms are mainly oriented to service robots and rarely involve industrial robots.

In order to balance the requirements of compute-intensive and time-sensitive tasks, [13] follows the trail of fog computing to improve the next generation of CR. FR consists mainly of the cloud and adaptable fog server, which acts as a companion to the cloud to ensure advanced performance with the lowest possible latency by shoving the data near the robot [14,15]. ER is similar to FR, except that it adds edge computing to handle time-sensitive tasks [16,17]. However, there is still a lack of a platform that would enable multiple vendors and fabricators for robots to develop and host intelligent robotic services, sharing resources and reducing costs across a large pool of users.

2.2 IIoT platforms

According to all relevant literature, current manufacturing industry trends point to building complex systems that integrate traditional production, automation, and computational intelligence. As a result, researchers have proposed three concepts that have considerable overlap: Industrial Internet [30], Industry 4.0 [31], and Industrial Internet of Things. As a subset of IoT, IIoT refers to a system comprising networked smart objects, associated information technologies, and optional cloud/edge/fog computing platforms, which enable intelligent services within the industrial environment, to optimize overall production value [32]. Because IIoT encompasses a wider range of content than the other two concepts [33] and is more relevant to IR's application scenarios, this paper adopts the IIoT paradigm. Especially, platform architecture is the focus of IIoT research.

So far, due to the technical fragmentation, inherent heterogeneity, and diversity of application areas, there are many architectures for the IoT ecosystem [34]. Among them, those related to industrial scenarios are Reference Architectural Model Industrie 4.0 (RAMI 4.0), Industrial Internet Reference Architecture (IIRA), and Industrial Internet Architecture (IIA). With a three-dimensional layer model, RAMI 4.0 focuses on next-generation industrial manufacturing systems to integrate different user perspectives. IIRA is similar to IIA in that both describe cloud-edge-device three-layer architectures from the business, usage, functional, and implementation perspectives, respectively. The functional viewpoint describes the basic platform functions, such as resource scheduling and operation and maintenance management. In terms of this, we pay great attention to the related work for cloud-edge collaboration. Kubernetes (K8s) is commonly accepted as an open-source framework for managing distributed containerized applications at the cloud layer. As for the edge layer, many lightweight resource management frameworks are proposed, including K3s, KubeEdge, EdgeX Foundry, Eclipse IoT, and ETSI MEC, with KubeEdge and K3s being popular because of their high compatibility with K8s. K3s is a lightweight version of K8s that lessens external package dependencies and simplifies deployment, but it lacks the ability to schedule the cloud-edge resource collaboratively and thus could not integrate with K8s well [35]. KubeEdge extends K8s' cloud-native microservice management capabilities to the edge and can orchestrate edge

devices effectively [36,37]. KubeEdge is in development, and its compatibility with the industrial sector still needs to be improved, such as supporting Open Platform Communications Unified Architecture (OPC UA) protocol and IR access.

The implementation viewpoint focuses on the technologies and components required to develop intelligent applications. In particular, the implementation viewpoint presents the widely accepted three-tier pattern from a business perspective, comprising edge, platform, and enterprise tiers. The edge tier uses the proximity network to collect and pre-process data from devices. The platform tier leverages the service network to receive, process, and forward control commands to the edge tier. The enterprise tier implements domain-specific applications and decision-support services. Unlike consumer Internet and IoT, the above architecture is only a generic design and needs to be modified according to the industry characteristics when applied to specific industries [22−25]. Similarly, this paper designs and implements an IIoT architecture in ACbot from both the functional and implementation viewpoints based on the generic architecture of IIRA and IIA.

3 ACbot platform

This section explains the proposed ACbot solution in detail. Since managing robot-related data is the foundation of the whole platform, we first describe the information model used to define the main data within it. Next, we propose a three-layer functional architecture to manage platform resources, orchestrate microservices, and automatically scale microservice instances. Last, we put forward the specific implementation architecture of the ACbot platform to process robot-related data and deploy intelligent applications.

3.1 Information model

ACbot is an IIoT platform for the IR industry, involving various users, platform services, devices, and data. For example, its users include research institutions, IR manufacturers, and industrial manufacturing companies; its platform services include data collection, data upload/receiving, data processing, and intelligent applications; its devices include various servers, edge devices, sensors, and IRs; and the data involves static tabular data that represent device attributes and dynamic time-series data that represent device states. Without an abstract and unified model, collecting multi-modal data from different devices and providing various services for users with distinct requirements can be a headache. Therefore, to realize efficient information interaction among various users and collaborative development and integration of services within the platform, the first step is to define a common information model for the whole ACbot platform. Based on the reference information model for IoT [38,39], our information model has two key design rationales. First, it separates the data of different tenants and users in a logical and secure manner, while also providing access controls and permissions to ensure data privacy and security. Second, it models the static and dynamic data separately.

Specifically, Fig. 1 depicts the information model of the main data in the ACbot platform using UML class diagrams. The model design follows an entity-attribute approach where entities have their custom attributes and inherited ones from the superclass. Overall, our information model contains three groups of entities. The first group is entities of the company and device to cover multiple participants and ensure their data isolation. The former is modeled as CompanyInfo, which contains basic information on the organizations connected to ACbot, including manufacturers, users, and IR research institutions. The latter is EquipmentInfo, containing basic attributes (IP, access time, location, etc.) of devices like sensors, edge devices, robots, and robotic end-effectors, and it has a bi-directional association with companyInfo.

The second group of entities focuses on robot-related devices including edge devices, robots, and robotic end effectors. End effectors are installed on the wrist of a robotic arm to interact with the environment, whose type depends on the application scenario, such as polishing, welding, spraying, handling, and assembly. In order to cope with the static and dynamic data of robots and end-effectors, their data are modeled as static (RobotAttribute or EndEffector) and dynamic (RobotDynamicInfo, WeldEffectorDynamicInfo or

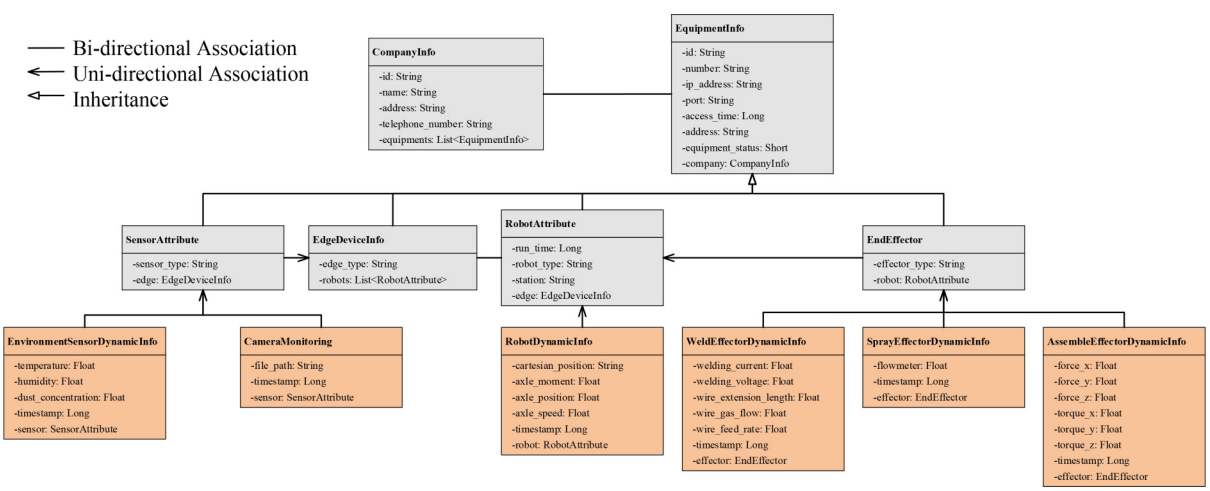


Fig. 1 ACbot information model. This model design follows the UML class notation with association and inheritance relationships. The black box represents the class for tabular data and the orange box represents the class for time-series data

SprayEffectorDynamicInfo) models separately. There is a bi-directional association between EdgeDeviceInfo and RobotAttribute, and dynamic models are associated with static models. The last group of entities refers to external sensors deployed at the edge to collect environmental data and video cameras to monitor the site. Specifically, EnvironmentSensor-DynamicInfo and CameraMonitoring represent the time-series data of environment and monitoring, both of which have uni-directional associations with the SensorAttribute.

3.2 Platform functional architecture

As shown in Fig. 2, the function of ACbot is structured in cloud-edge-device three layers and two core Platform-as-a-Service (PaaS) functional components, i.e., cloud-edge resource Collaboration Engine (CoEngine) and Deployment & Operation service Engine (DeOpEngine). The *device layer* includes sensors, cameras, robots, and actuators to collect environmental and video data from industrial sites and execute specific production tasks in the factory area. The *edge layer* includes various heterogeneous computing devices that connect devices via wired/wireless proximity networks to collect data or distribute control information. The *cloud layer* is a cloud computing center that consolidates processes and analyzes data flows from the edge layer, and then receives, processes, and forwards control commands from users to the edge layer. Finally, according to the characteristics and requirements of IRs, four intelligent applications are designed and implemented based on this architecture. In the upcoming sections, a detailed description of the two core functional components is given.

3.2.1 Cloud-edge resource collaboration engine

ACbot adopts containerization technology Docker and deploys it both in the cloud and on the edges. Since distributed resources are not conducive to service quality, it is critical to design a cloud-edge resource collaboration mechanism that extends the cloud to edges and enables unified scheduling of network, computing, and storage. In ACbot, the CoEngine collaborates the resources between the cloud and edges from the following aspects: 1) The cloud and edges are loosely coupled and edge nodes sense each other to form a decentralized semiautonomous system, which reduces response delays, and ensures reliability. When the resources in edges are about to be exhausted, some low-priority tasks could be offloaded to cloud nodes elastically to meet edges QoS constraints. 2) The CoEngine keeps tracking every resource node by monitoring their available network, computing, and storage resources and reports to the DeOpEngine. 3) The CoEngine and the DeOpEngine share the Scheduler, which runs our cloud-edge collaborative scheduling mechanism and deploys applications. This paper constructs the ACbot CoEngine with K8s on the cloud and KubeEdge at the edge, as depicted in Fig. 2. We'll also introduce the scheduling mechanism at the end of this part.

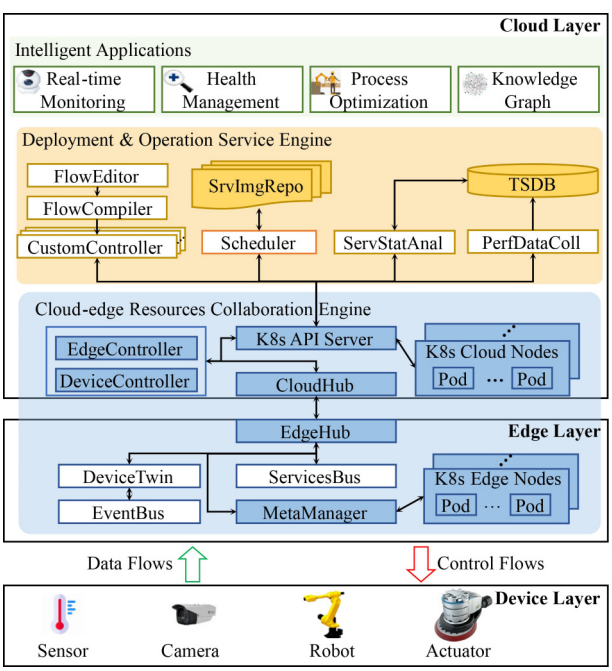


Fig. 2 Functional architecture of ACbot. The colorless modules represent our contribution. The Scheduler serves both in the CoEngine and DeOpEngine

The CoEngine@Cloud is mainly composed of K8s API Server, K8s cloud nodes, CloudHub, EdgeController, and DeviceController. The K8s API Server provides HTTP REST interfaces to create, retrieve, update, and delete (CRUD) various resource objects (pods, replication controllers, and services) located on the cloud and the edge. It deploys different pods to the nodes according to the scheduling instructions from the Scheduler. K8s Nodes are virtual or physical machines that could deploy one or more pods, which are divided into cloud nodes and edge nodes according to their physical location. A pod includes one or more containers and is the basic resource unit that can be created and managed in ACbot, its life cycle is relatively short and can be destroyed at any time. EdgeController and DeviceController are extended K8s controllers that manage edge and device metadata so that their metadata and status data are synchronized between edge and cloud. The CloudHub is a WebSocket server caching and sending scheduled messages to EdgeHub.

The CoEngine@Edge mainly includes EdgeHub, K8s edge nodes, ServiceBus, MetaManager, DeviceTwin, and EventBus. EdgeHub is a WebSocket client that receives resource scheduling instructions from the cloud and reports the status changes of edge hosts and devices in reverse. K8s Edge nodes play a similar role to those in the cloud described above. ServiceBus is an HTTP client that accepts requests from microservices on the cloud. MetaManager caches and synchronizes metadata with the cloud through a local lightweight database (SQLite). DeviceTwin stores and syncs device status to the cloud, and provides standardized query interfaces for other applications. EventBus is an MQTT client, which handles publishing and subscription between devices and the edge. Our primary contribution to CoEngine@Edge is expanding its access ability to the device layer and service interaction with the cloud layer. The original KubeEdge supports only Bluetooth and Modbus protocols for device access, which are insufficient for managing and collecting data from various IRs, executors, sensors, and other devices in IR application scenarios. To this end, we implement customized

protocol driver interfaces in EventBus to include TCP, MQTT, OPC UA, and Real Time Streaming Protocol (RTSP) using the KubeEdge Mapper-SDK, and we define new industrial devices in DeviceTwin K8s CustomResourceDefinition (CRD), including the device models and their instances. On the other hand, we developed Websocket interfaces in the ServiceBus for some of our applications to interact between the cloud and edge, which the original KubeEdge does not allow.

Our scheduling mechanism [40] is proposed to decide where to deploy the services according to the cloud-edge resource usage. To be specific, our mechanism aims at placing the microservices facing the heterogeneity of cloud-edge: limited resources of edge devices, and low bandwidth of Wide Area Network (WAN). We design a two-stage scheduling method: 1) a cloud-edge partitioning algorithm for congestion optimization, and 2) a topology-based edge resource allocation. In the first stage, we utilize the idea of maximum flow minimum cut to generate different division schemes that are diverse in the impacts for cloud-edge and finally the cloud-edge division scheduling scheme with the best global performance by Dynamical Programming (DP), which could avoid task congestion caused by cloud-edge heterogeneity and thus improve bandwidth utilization. In the second stage, we design the deployment of Operators (Ops) in each domain using the tree topology to reduce the hops that the data passes by and to improve the total efficiency of the services, reducing the overall latency.

3.2.2 Deployment & operation service engine

ACbot adopts microservices [41] for developing intelligent applications by decomposing them into multiple microservices with small businesses, strong independence, and loose coupling. While microservices are agile and scalable to improve developing productivity, there is the requirement for quick response to critical business. In the cloud-edge industrial application scenario with asymmetry resources and complex topology, flexible automating deployment and operations mechanisms are even vital. Therefore, in ACbot, the DeOpEngine is built up to realize an agile development of services through manual service orchestration, and dynamically scale the number of microservice instances to adapt to workload changes and guarantee the quality of service as well as cluster resource utilization. As shown in Fig. 2, DeOpEngine consists of FlowEditor, Performance Data Collection (PerfDataColl), Service State Analysis (ServStatAnal), Flow Compiler, and Scheduler to achieve two functions of manual service orchestration and automatic scaling.

The service orchestration function provides the agile Service Fabric (SF) interface for users to construct their business-oriented applications, which may be composed of various micro-services. It takes three steps to launch an application from the raw programs of the micro-services: create Ops, build a workflow, and deploy an instance. First, the programs and running context of the micro-services can be packaged as Docker images according to their Dockerfiles, and uploaded to the service & image repository (SrvImgRepo). When the Docker images are pulled to a running environment and execute the programs, they are instantiated as Ops, which are the basic computational units that perform a custom task on data streams. Second, the users can build their workflow by stringing a series of Ops in our FlowEditor. The FlowEditor based on Scalable Vector Graphics (SVG) allows users to string the workflow through drag-and-drop Ops over a graphical display. Users can modify the information of Ops in the FlowEditor, including inputs, outputs, key parameters, image addresses, and attributes. Then the FlowCompiler compiles the orchestrated flow defined in the FlowEditor into a workflow object written in a K8s CRD, including the description of the Ops and the whole data flow. Finally, for each workflow, we develop a CustomeController that monitors all newly created instances to manage the resource. Upon receiving a deployment request, it invokes the K8s API server to assign the pods and services. Next, the Scheduler running our schedule mechanism decides where to put the pods to ensure that the workflow runs smoothly and efficiently. Then the API server notices the K8s nodes to pull the Docker images and run them in the pods. Once the pods are launched, the CustomController will be informed to update the state of the workflow. The creation of a workflow instance is complete, and the expansion and contraction of running services will be discussed in the next paragraph.

The automatic scaling function is responsible for the dynamic on-demand scaling of deployed microservice instances, mainly implemented by PerfDataColl, ServStatAnal, and Scheduler components. By deploying the Istio service grid and Prometheus monitoring software, The PerfDataColl collects performance metadata including CPU utilization, logging, service response time, request traffic, etc., and stores those data in a time-series database (TSDB) to be aware of the running condition of microservices. By mining the historical data collected by the PerfDataColl, the ServStatAnal gets the invocation topology between microservices, detects anomalous microservices according to the rule base, and generates scaling plans. In the configurable rule base, anomalous microservice detection rules and edge microservice scaling rules are worth being detailed. According to the utilization of allocated resources, the anomalies include microservice overload and resource surplus. First, considering $P95$ denotes a value at the 95% position of the response-time order sequence, we define microservice overload rules as:

$$P95_{now} \geqslant \mu_{P95} + 3 * \sigma_{P95}, \tag{1}$$

where $P95_{now}$ is the real-time response time, and μ_{P95} and σ_{P95} are the mean and standard deviation of $P95$ for the past period. If the $P95_{now}$ of a microservice violates the 3σ rule, it can be detected as an overloaded microservice. The microservice resource surplus rule is similarly defined as:

$$P95_{now} < \mu_{P95} - 3 * \sigma_{P95}. \tag{2}$$

On the other hand, compared to the cloud layer, the edge has limited resources and typically runs microservices with high real-time requirements in industrial scenarios, so their scaling rules are very different from those in the cloud. For instance, a scaling rule should always ensure that some low-latency critical microservices run at the edge. When anomalous

microservices at some edge need to be scaled, some will be offloaded to the cloud side according to the time-delay constraints priorities. The Scheduler will not be described here, see above for details.

3.3 Industrial robotic intelligent applications

Figure 3 shows the implementation architecture of ACbot, which concerns basic components and microservices needed to implement and their coordinated mechanism for supporting industrial robotic intelligent applications. Specifically, the implementation architecture of ACbot includes intelligent applications, stream/batch computing, components, microservices, and data/control flows. The edge deploys lightweight basic components and real-time microservices to implement data cleaning, online model inference, and control flow delivery. The cloud runs data-intensive and computation-intensive components and microservices, such as databases, message queue clusters, model training, knowledge graph construction, and model training. These microservices and components coordinate to form the intelligent applications in Fig. 3, of which health condition monitoring and prediction belong to the health management intelligent application. It's worth noting that the control flow is implemented through the EventBus mentioned above, which will pass the control commands to the robot via TCP or OPC UA protocol. When the health monitoring application generates an alarm or there are new process parameters from the process optimization application, a shutdown command or a set of optimized parameters will be distributed to the robot controller. However, it should be verified by on-site engineers on the teach pendant for safety reasons.

Before detailing the intelligent applications, it is necessary to explain the cloud-edge-device communication schemes of the ACbot. Similar to [20,42], this paper will describe the communication protocol of ACbot in two dimensions, physical and logical data transportation in Table 1. Since in most industrial scenarios, an Automated Guided Vehicle (AGV) interacts with other IRs and is a vital member of the production line, we also consider it a kind of IR. In the physical data transportation dimension, most IRs connect to the edge with a wired connection, while in contrast, AGVs use wireless protocols such as WiFi or WIA-FA [43] to manage it. At the logical level, they communicate with the edges through TCP, MQTT, OPC UA, and RTSP protocols. Edge-cloud interaction is realized through MQTT, HTTP, and WebSocket protocols over wired connections or 4G/5G wireless networks.

3.3.1 Real-time monitoring

Real-time monitoring in ACbot involves the performance and behavior of IRs in real time to prevent robot downtime, consisting of data collection, processing, transmission, storage, and visualization. As shown in Fig. 3, the real-time monitoring application observes the factory area, aggregates data from devices to the cloud, and provides data for other intelligent applications. On the edge, the data collection microservice connects with cameras, sensors, IRs, and IR effectors to collect high-frequency dynamic data into the message queue component. The message queue component implemented by a standalone Kafka caches high-frequency data to prevent data loss. The data cleaning microservice

Table 1 Communication protocols in ACbot

Data transportation	Device-edge	Edge-cloud
Physical	wiredwireless: WiFi, WIA-FA	wiredwireless: 4G, 5G
Logical	TCP, MQTT, OPC UA, RTSP	MQTT, HTTP, WebSocket

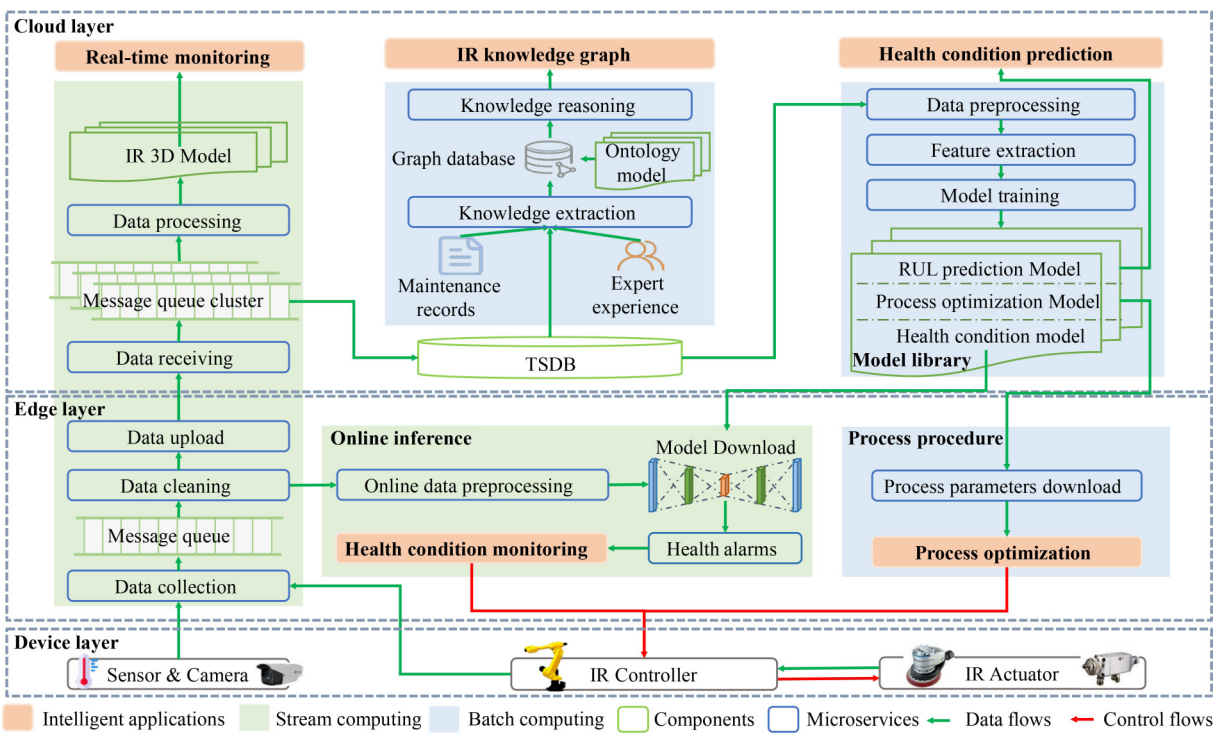


Fig. 3 Implementation architecture of ACbot

based on at least one stream processing semantics verifies the integrity, correctness, and chronological alignment of time series to build the foundation for later data analysis. In the cloud, the data-receiving microservice passes the data transmitted from various edges into the message queue cluster (Kafka cluster) and then it is stored in TSDB (InfluxDB). The data processing microservice selects the last 5 minutes of data from the Kafka cluster, loads IR 3D models file, i.e., Unified Robot Description Format (URDF), and implements real-time monitoring of IR in the factory area through Web pages.

It is worth noting that since the data traverses over the WAN, the so-called hard real-time is difficult to guarantee. In addition, the functional positioning of the IIoT platform is distinct from traditional industrial automation applications like Manufacturing Execution Systems (MES). ACbot aims at mining and leveraging the knowledge of maintenance and processes by collecting massive amounts of data. Therefore, real-time monitoring in ACbot is a soft real-time application, that is, a few data missing the deadline can be tolerated without causing severe harm. To achieve soft real-time, ACbot discards data with a delay of more than 10 seconds and then linearly interpolates the missing data, since we collect the data in a high frequency, and a small portion of loss and corresponding interpolation will not deviate significantly from the actual states.

3.3.2 Health management

The health management application, including health condition monitoring and prediction, is an IR O&M approach utilizing stream/batch computing and artificial intelligence inference to detect, estimate, and track degraded conditions of IRs, thus reducing unexpected downtime and alleviating maintenance costs.

Health condition monitoring is a process of monitoring the operating parameter to detect a change that is indicative of a developing breakdown, and the parameter is the so-called Health Indicator (HI). The key factors in this process are constructing the health condition model and generating appropriate HIs. Novelty, this paper considers the problem as a time-series Anomaly Detection (AD) task performed between cloud and edge and chooses the anomaly score as HI. Currently, because of the increase in data volume and dimensionality, few anomalies, and costly anomaly labeling, many unsupervised AD methods have emerged, such as tree-based [44], autoencoder-based [45,46], and one-class classification-based [47] methods. Accordingly, we propose a reconstruction-based health condition monitoring model for IRs, and the model architecture is shown in Fig. 4. In practice, it is realized in two stages, i.e., the offline model training on the cloud and online model inference of edge, as shown in Fig. 3. The former processes the data stored in TSDB, extracts features, and trains the health condition model on the cloud. Then the edge downloads the trained model, outputs the real-time health indicator by online data preprocessing, and provides health alarms for O&M staff.

Take a 6-axis industrial robot for example. In the training phase on the cloud, the input time series $\mathbf{X}_i \in \mathbb{R}^{18\times T}$ collected from the IR is passed to a multi-layer temporal convolution

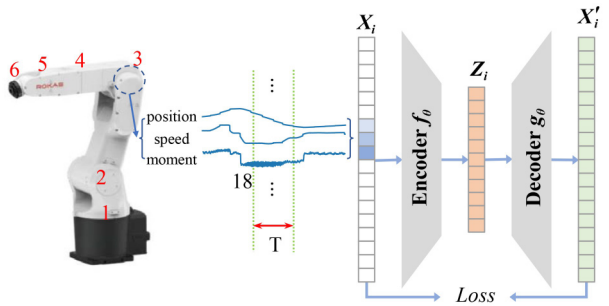


Fig. 4 Architecture of the health condition monitoring model. The left is the model of the 6-axis IR. The blue curves are the third axis's position, speed, and moment data. T is the time window

feature encoder $f_\theta : \mathcal{X} \mapsto \mathcal{Z}$ which outputs latent representations $\mathbf{Z}_i$. The encoder network has a 5-block temporal convolutional architecture, each block comprising a Conv1D layer, a BatchNormalization (BN) layer, and a ReLU activation function. Furthermore, latent representation $\mathbf{Z}_i$ is fed to a multi-layer temporal deconvolutional feature decoder $g_\theta : \mathcal{Z} \mapsto \mathcal{X}$ produces reconstruction time series $\mathbf{X}'_i$ to learn temporal dependencies. The decoder network has a 5-block temporal deconvolutional architecture, each block comprising a ConvTranspose1d layer, a BN layer, and a ReLU activation function. Finally, the overall loss function of the reconstruction-based health condition monitoring model between $\mathbf{X}_i$ and $\mathbf{X}'_i$ is defined as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{X}_i - \mathbf{X}'_i\|^2. \tag{3}$$

The HI $S_i = \|\mathbf{X}_i - \mathbf{X}'_i\|^2$ will be generated for the time series $\mathbf{X}_i$ in the online model inference of edge. Then, the following formula is applied to determine whether a health alarm is generated:

$$y_i = \begin{cases} \text{anomaly}, & S_i > \tau, \\ \text{normal}, & S_i \leqslant \tau, \end{cases} \tag{4}$$

where τ is a predefined threshold.

Health condition prediction tries to foresee the Remaining Useful Life (RUL) by assessing the extent of deviation or degradation of an IR from its expected normal operating conditions. As shown in Fig. 3, health condition prediction is a non-real-time application relative to monitoring, so it is trained and predicted directly on the cloud. The RUL prediction model is generally composed of HI construction and degradation curve fitting [48]. For HI construction, a single physical HI is too simple and weak to represent the health condition of IRs, which own complex structures and may face diverse failures. Hence, we use the virtual HI obtained from the above health condition monitoring, i.e., reconstruction-based HI by fusing multiple multi-sensor signals. Reconstruction-based HI loses physics meanings and just presents a virtual description of the degradation trends of IRs. On the other hand, IR damages are usually cumulative, so we use an exponential degradation model to fit the degradation curve. To better describe the degradation processes of IRs, this paper uses a single exponential function and a sum of two exponential functions as the basic exponential degradation

model. Their mathematical expressions are given as follows:

$$\begin{cases} \mathrm{HI}(t) = a * \exp(bt) + c, \\ \mathrm{HI}(t) = a * \exp(bt) + c * \exp(dt), \end{cases} \tag{5}$$

where HI is the reconstruction error S_i, t is the sampling time, and a, b, c, and d are four unknown parameters that need to be fitted. Then, this paper employs the Fréchet distance [49] to find the optimal fitting curve, i.e., the RUL prediction model. Finally, the RUL is obtained by calculating the time when the fitting degradation curve reaches the pre-defined HI threshold.

3.3.3 Process optimization

IRs have been widely used in welding, polishing, spraying, and other process scenarios for their reliability and controllability. In industrial decision-making, process optimization refers to the optimization of some process parameters to reach the optimal goal without violating some constraints. The most common optimization goals are minimizing cost and maximizing throughput or efficiency. For IRs, the process parameters are actuator data (welding current and voltage, gas flow, and so on) and robot end data (speed, angle), which are contained in our information model (see Fig. 1). The optimization goals vary depending on the process scenario, such as less energy consumption in assembly and higher quality in welding. As shown in Fig. 3, process data is accumulated from the edge layer into the TSDB. Then a process optimization model is trained in the cloud using batch processing to obtain the optimal process decision, which is delivered to IRs for execution. In this procedure, the vital models include process evaluation and parameter optimization.

Process evaluation aims at depicting relationships between process parameters and optimization objectives in various applications where operation efficiency, product quality, and energy cost of IR are focused on. Due to the inhomogeneity and complexity of industrial processes, the scale of data collection varies. It is critical to use appropriate and powerful models to characterize the features of diverse application scenes. Take the robot in the assembly scenario as an example. Assembly robots usually execute tasks of adding parts to semi-finished products where the cycle time and energy cost are normally taken into account, especially in heavy-load environments. Based on the rich robot and actuator data from ACbot's TSDB, the neural network is used to fit the correlation between the process parameters and the optimization objectives. As in our previous work [50], an Epson C4 IR was selected to validate the proposed method where a multi-layer perception model was utilized for energy consumption prediction.

However, it's much harder to accumulate data in the welding scenarios than in assembly ones due to the high costs of time and materials, as well as the huge influence on practical production processes. Therefore, based on the existing welding data in ACbot, a new welding robot process evaluation is carried out using a transfer-learning approach. In particular, dynamic strategies on model parameter transfer are designed concerning their similarities in input layer structures and evaluation objectives. According to our paper [51], it is seen that both the model accuracy and algorithm stability are dramatically enhanced when transferring model parameters of Epson C4 IR to those of SIASUN SR10C IR.

Parameters optimization finds the optimal combination of process settings to meet the requirements of IR application. Based on prediction models built through process evaluation, it is possible to define these models as objective functions and seek optimal inputs, known as process parameters. Specifically, we can use heuristic algorithms like genetic algorithms for single objectives, evolutionary algorithms based on decomposition, mon-dominated sorting genetic algorithms for multiple objectives, and so on. Figure 5 illustrates a welding scenario for optimizing the time, quality of welding bead, and energy cost. Red marks represent the experiment data in the objective space and the blue marks mean the Pareto optimality gained by the MOEA/D algorithm in this situation with no better aspects between these solutions. Then, the optimal parameters are generated and dispatched to guide the practical production through the process procedure in the edge layer of the ACbot shown in Fig. 3.

3.3.4 IR knowledge graph

The IR knowledge graph is a knowledge base that converts multi-source heterogeneous data into a semantic knowledge network, which lays the foundation for the intelligent maintenance of IRs. As illustrated in Fig. 3, the process of building the IR knowledge graph on the cloud is composed of four technical processes, i.e., ontology modeling, knowledge extraction, reasoning, and storage. At first, an ontology is a description of relevant concepts in the domain, including classes, instances, relationships, attributes, and restrictions. In this paper, the IR ontology model mainly contains the IR mechanical structure ontology, the IR maintenance case ontology, and the IR health condition monitoring algorithm ontology. This model is constructed utilizing protégé modeling tools through a seven-step methodology and a top-down way. After that, the modeling results are described using Web Ontology Language (OWL) and imported into the graph database to store.

Second, based on the ontology model described above, Named Entity Recognition (NER) and Relation Extraction (RE) techniques are applied to extract knowledge from unstructured data, such as maintenance records, expert experience documents, academic books and papers, and product manuals. The NER locates and classifies entities into

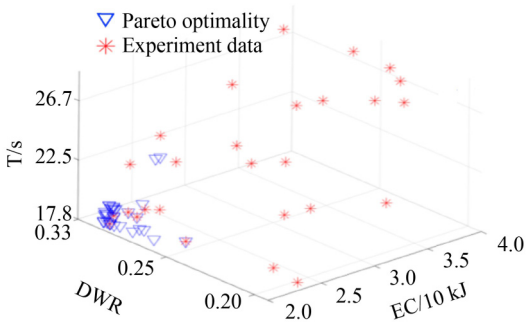


Fig. 5 Pareto front toward welding efficiency, quality, and energy consumption of IR. The quality of the weld is expressed by the Depth-to-Width Ratio (DWR)

pre-defined categories related to IRs, such as a person, location, equipment name and type, failure phenomenon and causes, repair method, and algorithm name. Specifically, the NER model in this paper contains BERT [52], Bi-directional Long Short-Term Memory (BiLSTM), and Conditional Random Field (CRF). BERT is pre-trained to get word embeddings, then BiLSTM gets the context vectors of the sequences, and finally, CRF outputs the predicted label sequences. RE detects and classifies semantic relationships in entities into the triple <subject, relation, object>. This paper adopts rule-based RE including word segmentation, syntactic dependency parser, and RE rules. In detail, we use the word-segmentation component Jieba to cut sentences into phrases and then analyze the grammatical relationships between words into a dependency tree by the HanLP component. Combining extracted entities and dependency trees, this paper defines the RE rule to obtain triples.

Third, knowledge reasoning identifies errors and infers new knowledge from existing data to improve the completeness of the knowledge graph. This paper adopts rule-based knowledge reasoning, by building first-order predicate logic rules. For example, one of the first-order predicate logic reasoning rules is as follows:

$$\begin{aligned}&(ReducerAbrasion, Cause, RobotArmFailure)\\ \wedge&(RobotArmFailure, Cause, RobotVibration)\\ \Rightarrow&(ReducerAbrasion, Cause, RobotVibration).\end{aligned} \tag{6}$$

Finally, this paper stores knowledge in a graph database Neo4j.

4 Validation of the ACbot

To validate the feasibility of the proposed platform, ACbot has been online for over 1200 days in the real world. The following content provides details of the evaluation scenarios, application and experiment results, and a comparison with related platforms.

4.1 Application scenarios description

ACbot was established by Beihang University (BUAA), uniting Alibaba Cloud, Harbin Institute of Technology (HIT), Shenyang Institute of Automation Chinese Academy of Sciences, SIASUN, ROKAE, etc. Over the past three years, these organizations have worked together to carry out an ambitious plan to monitor and control different types of IRs, which are performing diverse processes in different industries, plant areas, and even locations, through developing and promoting the application of the ACbot platform. In this context, this subsection describes the platform configuration and the details of application scenarios.

Platform configuration Table 2 sums up the configuration of the ACbot platform, divided into cloud-edge-device three layers. The cloud computing center was built by Beihang University in Hangzhou, China, and consists of CPU, GPU, storage servers, and 1000 Mbps network bandwidth. The CPU servers are 8 DELL R740, with CoEngine@Cloud, DeOpEngine, basic components, and microservices deployed. The GPU servers consist of 8 NVIDIA Tesla V100 for training intelligent models. The storage servers are 5 Sugon ParaStor300S with petabytes of storage capacity to store huge amounts of IR time series. Overall, the cloud layer contains diverse and rich resources that can support the development and deployment of current intelligent services for IRs.

The edges are located in each factory area running lightweight base components and real-time microservices, such as CoEngine@Edge, stand-alone Kafka, data collection microservice, and online inference microservice. To adapt to different manufacturing environments, ACbot includes edges that vary in computing capabilities, appearance, and structure. Raspberry Pi, the smaller edge equipment, is placed in the control cabinets of IRs and is mainly suitable for single IR accessing. Industrial PCs such as Vensin Genius P20 and Tardetech TD-MPC-1407 are mainly suitable for workstations located in harsh environments like sordid workshops. Besides,

Table 2 Summary of the ACbot platform configuration

Layer	Equipment	Number	Location	Description
Cloud	DELL R740 Server	8	Hangzhou	CPU Core 32, RAM 256 GB, SDD 960 GB, HDD 40 TB
	Sugon ParaStor300S storage server	5	Hangzhou	CPU Core 16, RAM 64 GB, SDD 480 GB, HDD 350 TB
	NVIDIA Tesla V100 GPU	8	Hangzhou	Graphics Memory 32 GB
	Total	21	CPU Core 336, GPU 8, RAM 256 GB, SDD 9.8 TB, HDD 2.0 PB, Network Bandwidth 1000 Mbps	
Edge	Raspberry Pi 4B	2	Beijing, Binzhou	CPU Core 4, RAM 8 GB, SD Card 64 GB, LAN 1
	Vensin Genius P20 Industrial PC	8	Beijing, Tianjin, Shenyang, etc.	CPU Core 2, RAM 8 GB, SDD 128 GB, LAN 4
	Tardetech TD-MPC-1407 Industrial PC	3	Shenyang, Xiamen, etc.	CPU Core 2, RAM 8 GB, SDD 128 GB, LAN 2
	DELL PowerEdge T150	2	Hangzhou, Harbin	CPU Core 4, RAM 16 GB, SDD 960 GB, HDD 4TB, LAN 2
Device	Polishing Robot	4	Jining	ROKAE XB-4s, SIASUN SRQ5C, OnRobot Sander
	Weld Robot	2	Tianjin	SIASUN SR10C, Yaskawa Welding Actuator
	Spray Robot	1	Shenyang	SIASUN SR10C, Schutze Spray Actuator
	Assembly Robot	2	Harbin, Jining	Epson C4, ROKAE XB-7, OnRobot MG10 Gripper
	AGV	2	Tianjin	SIASUN AGV
	Other Robot	49	Beijing, Shenyang, Suzhou, etc.	SIASUN IRs, ROKAE IRs
	Sensor	3	Hangzhou, Jining	OnRobot 6-axis Force Sensor, FristSensor Environment Sensor
	Camera	3	Beijing, Tianjin, Hangzhou	Hikvision HD Cameras
	Total	66	IR 60, Actuator 15, Sensor 3, Camera 3	

edge servers with relatively abundant resources are placed around the production line to access all IRs in the line.

The device layer mainly includes various types of IRs and actuators, sensors, and cameras, coming from various process scenarios, such as polishing, welding, spraying, handling, and assembly. The IRs are mainly from three robot manufacturers, SIASUN, ROKAE, and Epson, the actuators and sensors are mainly from OnRobot, Schutze, and Yaskawa, and the cameras are from Hikvision.

Application scenarios As shown in Table 2, ACbot manages 60 IRs from 10 organizations that fall into three categories according to the industries they engaged in, 3 academic institutions, 2 IR manufacturers, and 5 industrial manufacturing companies. The academic institutions include BUAA, HangZhou Innovation Institute of Beihang University (HZII), and HIT. BUAA holds a vision inspection platform consisting of 4 IRs for propeller assembly. HZII possesses a crafts assembly line of 8 IRs, including polishing, assembling IRs, vision inspection cameras, and an AGV. HIT performs its studies on a workstation with 2 IRs for gear assembly and welding. The IR manufacturers are SIASUN and ROKAE, both of which use self-developed robots to build IR accelerated life testing platforms and invested in 9 and 4 robots. The industrial manufacturing companies include Zhejiang IVKE Machinery, Binzhou Bohai Piston, Xiamen Liju, Qingzhou Dewei Power, and Wuhan Erechi Technology, which are customers of SIASUN and ROKAE in the automotive and electronics industries. Meanwhile, these five companies are involved in assembly, polishing, spraying, welding, and handling process scenarios.

4.2 Application results

Based on the proposed information model, CoEngine and DeOpEngine, we have deployed real-time monitoring, health management, process optimization, and knowledge graph intelligence applications in real scenarios, and the detailed web pages are shown in Fig. 6. Figure 6(a) shows the FlowEditor of the service orchestration in the DeOpEngine, where each Op is deployed in a separate Docker container. To support the applications, ACbot processes and saves the data collected from the device layer in the TSDB, where data engineers can visually view historical data on ACbot, as shown in Fig. 6(b). Until now, about 100 billion time-series points have been saved in our TSDB, occupying 2 TB of storage.

The real-time monitoring application displays the data collected from devices in real-time on the page, and experienced engineers can identify potential problems from changes in the data. In order to provide more intuitive monitoring, ACbot simulates the IRs' motion attitude by injecting position, moment, speed, and actuator time-series data (Fig. 6(c)) in a URDF model, and the platform also shows the real-time motion route and wheel moment time-series data of AGVs (Fig. 6(d)). In addition, ACbot supports real-time

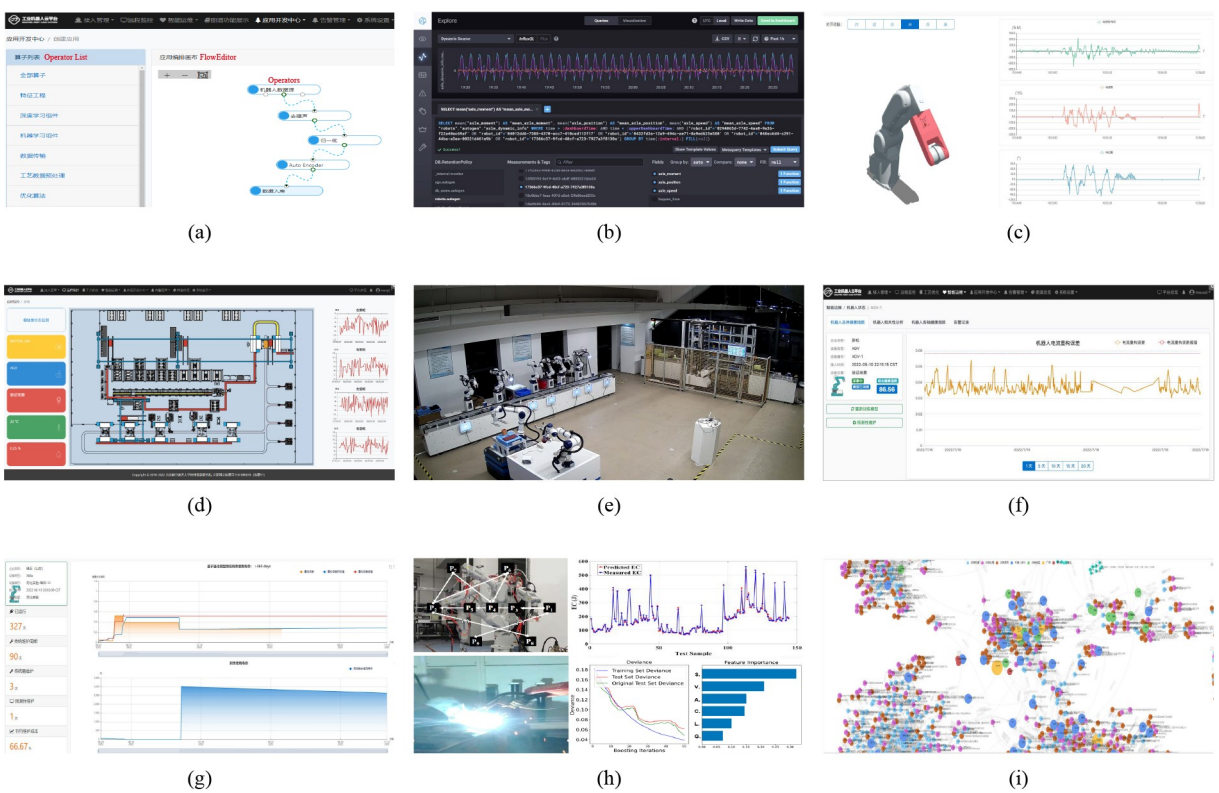


Fig. 6 ACbot application results. (a) FlowEditor of service orchestration; (b) IR time series; (c) IR monitoring; (d) AGV monitoring; (e) camera monitoring; (f) IR health condition monitoring; (g) IR health condition prediction; (h) Process optimization; (i) IR knowledge graph. (c), (d), and (e) are from the real-time monitoring intelligent application. (f) and (g) belong to the health management intelligent application. In (h), the upper part fits the correlation between the assembly parameters and energy consumption. In the bottom half of (h), the line graph is the correlation fitting deviation between welding parameters and objectives, and the histogram is the importance ranking of the process parameters including Speed (S), Voltage (V), Angle (A), Current (C), Length (L), and Gas flow (G)

monitoring of industrial sites recorded by cameras (Fig. 6(e)). For real-time monitoring, the average delay $T_{delay} < 1$s of the cloud real-time model fully meets the monitoring needs of industrial manufacturing companies.

Figure 6(f) shows the health condition monitoring, which alarms and notifies the manager of the IR when the reconstruction-based HI value S_i exceeds the threshold. The proposed health monitoring method is applied to SIASUN and ROKAE IRs, with a total of 26 abnormal alarms and the accuracy is 84.6%. In addition, ACbot predicts the RUL by calculating when the virtual HI proposed in the process reaches a preset threshold, as shown in Fig. 6(g). We evaluated our proposed health condition prediction method on IR accelerated life testing platforms of SIASUN and ROKAE, and the closer the training data was to the ground-truth useful life, the higher the prediction accuracy was. In summary, the average prediction accuracy of our model is 80.5%. The upper part of Fig. 6(h) shows the results of fitting the correlation between assembly parameters and energy consumption for HIT's Epson C4 robot with an average error of 4.6%. The optimal assembly parameters were then found by the genetic algorithm, resulting in an average energy consumption reduction of 12.3% at the same cycle time. Similarly, the lower part of Fig. 6(h) fits the correlation between welding parameters and objectives (quality, efficiency, and energy consumption) with an average error of 6.8% and reveals that welding speed is the most important process parameter. The welding optimization results are then obtained using the MOEA/D algorithm, with a quality improvement of 9.1%, an efficiency increase of 5.0%, and an energy consumption decrease of 8.9%. As shown in Fig. 6(i), the IR knowledge graph contains 2087 entity nodes and 4795 relationships. To facilitate the use of IR O&M engineers, our knowledge graph provides a knowledge Question-Answering (Q&A) function. Out of 1000 selected questions, this Q&A function gets 826 answers matching the expert experience.

Overall, based on user feedback, ACbot has performed well against a variety of application requirements, such as fast response, high availability, and easy operation, demonstrating the effectiveness and robustness of each component in our platform. Also, the four intelligent applications deployed on ACbot have decreased downtime, improved process quality, and reduced the workload of O&M staff, indicating the effectiveness and necessity of each application. With this, we are trying to show the function of intelligent application agile development for IRs provided by ACbot.

4.3 Evaluation of cloud-edge resource collaboration

We achieve the collaboration of cloud-edge resources through operator orchestration and their resource allocation since all computing tasks are split into multiple Ops with the data flow between them. We abstract the orchestration of multiple Ops into a computational graph problem, where the operator is a node and the data flow is an edge in the graph. Thus, to verify the effectiveness, we conducted experiments on three methods over the average latency and back-pressure rate as the computational graph scale increased. We chose two methods Baseline and [53] (called SBON in this paper) to compare with our method. The Baseline is a full cloud scheduling mechanism that divides all Ops other than the data collection Op into cloud domains. SBON is a classic QoS-aware cloud-fog/edge resource scheduling algorithm expanding the data stream processing platform Storm. Therefore, we take it as one of the comparisons. There are no experimental data available for SBON in large-scale jobs because of its time-consuming nature, which makes it unsuitable for scheduling 200 or more computational graphs simultaneously. As Fig. 7 shows, the proposed method outperforms both Baseline and SBON in terms of latency metrics, with latency reductions of 24.33% and 15.66% compared to them. In terms of back-pressure rate metrics, our method also outperforms other mechanisms, with an 11.18% reduction compared to Baseline and a 3.94% reduction to SBON.

4.4 Comparison with related platforms

In addition to RoboEarth [18] and Rapyuta [19], a lot of research has been done on CR, FR [13−15], and ER [16,17], but the majority of them are just proposing paradigms and practicing in a few cases, without larger-scale applications. Therefore, as shown in Table 3, we compare key features of ACbot, RoboEarth, and Rapyuta. First, RoboEarth and Rapyuta focus on service and logistics robots, which are single-tenant and individually provide robotic intelligence solutions for each business. Their architecture lacks the edge layer, which results in the need for a larger bandwidth. For intelligent services, RoboEarth and Rapyuta are mainly concerned with Simultaneous Localization And Mapping (SLAM) tasks by collecting camera data.

4.5 Lessons learnt

From the application of ACbot for more than 3 years, we can draw up some remarks. Firstly, because of the variety of IR manufacturers, types, actuators, and processes, the information model in ACbot has been modified several times during practice. Therefore, the information models established for vertical application areas of IIoT should ensure its extensibility and versatility. Secondly, industrial manufacturing companies are particularly concerned about the protection of IR and process data, so our team has put a lot of effort into promoting the ACbot platform. In the future, the prosperity of robot IIoT may mainly rely on the alliance of robot manufacturers and large manufacturing companies, especially car manufacturers that use a large amount of IR. Finally, the data mining algorithms in ACbot are developed by engineers in IR-related fields such as HIT, SIASUN, and ROKAE. To facilitate algorithm development, ACbot provides some interfaces to support algorithm development

Table 3 Summary of key features supported by ACbot vs. other platforms

	ACbot	RoboEarth	Rapyuta
Domain	industrial robot	service robot	logistics robot
Tenant	multi	single	single
Architecture	cloud-edge-device	cloud-device	cloud-device
Real-time monitoring	✓	✓	✓
Health management	✓	–	–
Process optimization	✓	–	–
Knowledge graph	✓	–	–
SLAM	–	✓	✓

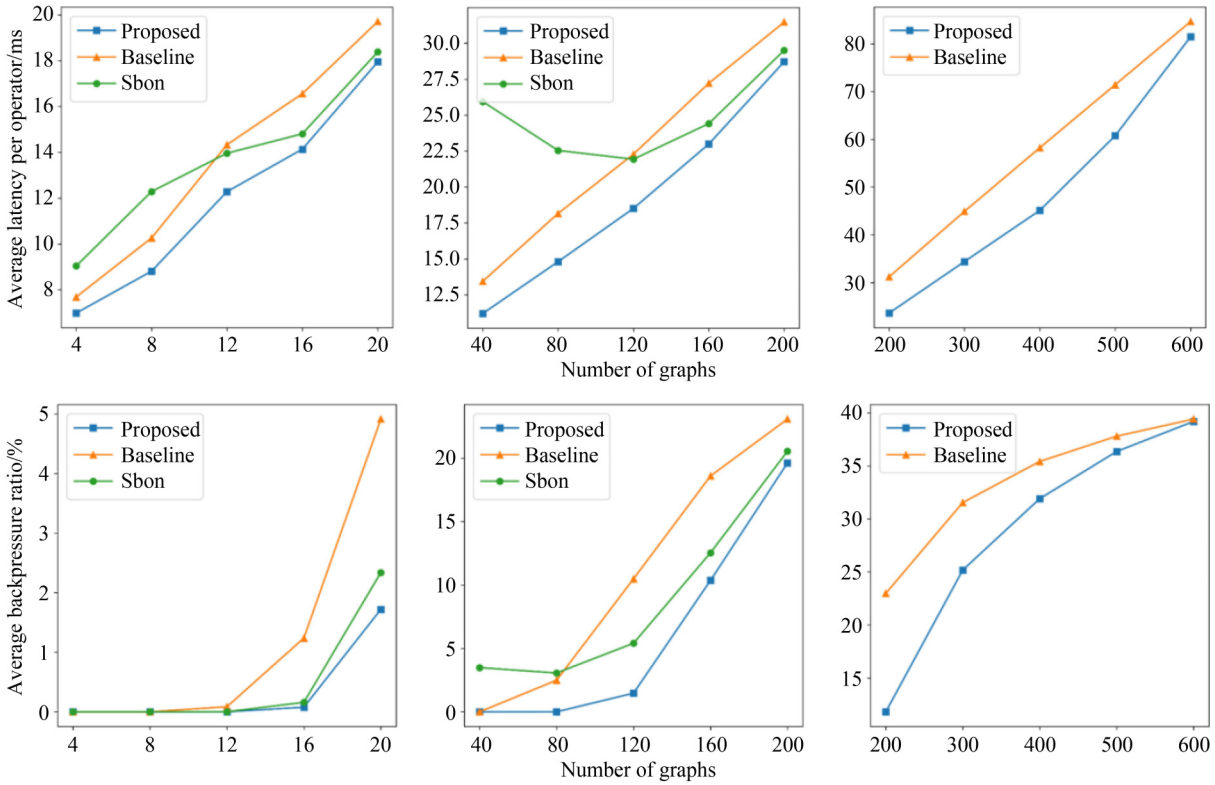


Fig. 7 Average latency and back-pressure rate of cloud-edge resource scheduling mechanisms

through the DeOpEngine and TSDB. It is worthwhile to learn that IIoT platforms should provide basic data mining functions (e.g., data pre-processing or multi-scale aggregation) so that domain experts can focus on developing business-related algorithms. All in all, these applications have helped us to confirm that the integration of traditional manufacturing with IIoT is currently a key requirement. This makes it possible to develop more sophisticated smart manufacturing services in an intelligent society which seems to be the next natural step toward more efficient manufacturing.

5 Conclusion

With the booming of Industry 4.0, Industrial Internet, and IIoT, it is a challenge to make IRs more intelligent, efficient, reliable, and easier to maintain. In this context, a large-scale IIoT-based IR platform (ACbot) is first proposed in this paper to solve the problems of multitenancy, device heterogeneity, intelligent service development, and hosting, and establish an IR ecosystem to promote cooperation between robot manufacturers, users, and researchers. In detail, a multi-tenant-oriented IR information model is built within ACbot, and in practice this model can be adapted to various types of IRs, actuators, and sensors, verifying its versatility and extensibility. After that, the ACbot functional architecture including CoEngine and DeOpEngine is proposed to manage resources and microservices across the cloud and multiple edges and simplifies the development of intelligent applications. Then, an implementation architecture for ACbot's intelligent applications is presented, and implementation details for real-time monitoring, health management, process optimization, and the knowledge graph are described. In over 3 years, the platform has been instantiated in real application scenarios having 10 organizations, 21 servers, 15 edges, 60 IRs, and 5 manufacturing processes, which demonstrates the effectiveness and robustness of ACbot. Meanwhile, our application results show that ACbot improves process quality, productivity, and maintainability of IRs.

Despite the remarkable benefits obtained by ACbot, there are areas for further improvement. Currently, the largest IR customers are automotive manufacturers with more than 1000 IRs in a single plant, but they have doubts about accessing the ACbot platform because of issues such as vendor lock-in and data security concerns. Consequently, in the future, new cooperation mechanisms and standardized architectures need to be explored with manufacturing companies and vendors to facilitate IIoT platforms to access a large number of industrial devices and deploy more intelligent applications. On the other hand, although an IIoT ecosystem of IR has been initially established through ACbot, it is still necessary to expand its influence so as to open the door to crowdsourced development.

Acknowledgements This research was supported by the Zhejiang Province Key R&D Program of China (2023C01070).

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

References

1. Hägele M, Nilsson K, Pires J N, Bischoff R. Industrial robotics. In: Siciliano B, Khatib O, eds. Springer Handbook of Robotics. 2nd ed. Cham: Springer, 2016, 1385−1422
2. Borgi T, Hidri A, Neef B, Naceur M S. Data analytics for predictive maintenance of industrial robots. In: Proceedings of the 2017

International Conference on Advanced Systems and Electric Technologies. 2017, 412−417
3. Aivaliotis P, Arkouli Z, Georgoulias K, Makris S. Degradation curves integration in physics-based models: towards the predictive maintenance of industrial robots. Robotics and Computer-Integrated Manufacturing, 2021, 71: 102177
4. Köksal G, Batmaz İ, Testik M C. A review of data mining applications for quality improvement in manufacturing industry. Expert Systems with Applications, 2011, 38(10): 13448–13467
5. Weichert D, Link P, Stoll A, Rüping S, Ihlenfeldt S, Wrobel S. A review of machine learning for the optimization of production processes. The International Journal of Advanced Manufacturing Technology, 2019, 104(5−8): 1889−1902
6. Ding Y, Xu W, Liu Z, Zhou Z, Pham D T. Robotic task oriented knowledge graph for human-robot collaboration in disassembly. Procedia CIRP, 2019, 83: 105–110
7. Deng J, Wang T, Wang Z, Zhou J, Cheng L. Research on event logic knowledge graph construction method of robot transmission system fault diagnosis. IEEE Access, 2022, 10: 17656–17673
8. McKee G T, Schenker P S. Networked robotics. In: Proceedings of SPIE 4196, Sensor Fusion and Decentralized Control in Robotic Systems III. 2000, 197−209
9. Schwager M, Rus D, Slotine J J. Decentralized, adaptive coverage control for networked robots. The International Journal of Robotics Research, 2009, 28(3): 357–375
10. Kuffner J. Cloud-enabled humanoid robots. In: Proceedings of the 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids). 2010
11. Hu G, Tay W P, Wen Y. Cloud robotics: architecture, challenges and applications. IEEE Network, 2012, 26(3): 21–28
12. Kehoe B, Patil S, Abbeel P, Goldberg K. A survey of research on cloud robotics and automation. IEEE Transactions on Automation Science and Engineering, 2015, 12(2): 398–409
13. Gudi S L K C, Ojha S, Clark J, Johnston B, Williams M A. Fog robotics: an introduction. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. 2017
14. Gudi S L K C, Ojha S, Johnston B, Clark J, Williams M A. Fog robotics for efficient, fluent and robust human-robot interaction. In: Proceedings of the 17th IEEE International Symposium on Network Computing and Applications. 2018, 1−5
15. Pujol V C, Dustdar S. Fog robotics-understanding the research challenges. IEEE Internet Computing, 2021, 25(5): 10–17
16. Groshev M, Baldoni G, Cominardi L, de la Oliva A, Gazda R. Edge robotics: are we ready? An experimental evaluation of current vision and future directions Digital Communications and Networks, 2023, 9(1): 166–174
17. Huang P, Zeng L, Chen X, Luo K, Zhou Z, Yu S. Edge robotics: edge-computing-accelerated multirobot simultaneous localization and mapping. IEEE Internet of Things Journal, 2022, 9(15): 14087–14102
18. Waibel M, Beetz M, Civera J, D'Andrea R, Elfring J, Gálvez-López D, Häussermann K, Janssen R, Montiel J M M, Perzylo A, Schießle B, Tenorth M, Zweigle O, Van De Molengraft R. RoboEarth. IEEE Robotics & Automation Magazine, 2011, 18(2): 69–82
19. Mohanarajah G, Hunziker D, D'Andrea R, Waibel M. Rapyuta: a cloud robotics platform. IEEE Transactions on Automation Science and Engineering, 2015, 12(2): 481–493
20. Arnold L, Jöhnk J, Vogt F, Urbach N. IIoT platforms' architectural features–a taxonomy and five prevalent archetypes. Electronic Markets, 2022, 32(2): 927–944
21. Schneider S. The industrial internet of things (IIoT): applications and taxonomy. In: Geng H, ed. Internet of Things and Data Analytics Handbook. Hoboken: John Wiley & Sons, Inc., 2017, 41−81
22. Li H, Li X, Cheng Q. A fine-grained privacy protection data aggregation scheme for outsourcing smart grid. Frontiers of Computer Science, 2023, 17(3): 173806
23. Zhou C, Damiano N, Whisner B, Reyes M. Industrial internet of things (IIoT) applications in underground coal mines. Mining Engineering, 2017, 69(12): 50–56
24. Maatoug A, Belalem G, Mahmoudi S. A location-based fog computing optimization of energy management in smart buildings: DEVS modeling and design of connected objects. Frontiers of Computer Science, 2023, 17(2): 172501
25. Han Y, Zhang C J, Wang L, Zhang Y C. Industrial IoT for intelligent steelmaking with converter mouth flame spectrum information processed by deep learning. IEEE Transactions on Industrial Informatics, 2020, 16(4): 2640–2650
26. Wang R, Mou X, Sun J, Liu P, Guo X, Wo T, Liu X. Cloud-edge collaborative industrial robotic intelligent service platform. In: Proceedings of 2020 IEEE International Conference on Joint Cloud Computing. 2020, 71−77
27. Carvalho T P, Soares F A A M N, Vita R, Francisco R D P, Basto J P, Alcalá S G S. A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, 2019, 137: 106024
28. Silvestri L, Forcina A, Introna V, Santolamazza A, Cesarotti V. Maintenance transformation through industry 4. 0 technologies: a systematic literature review. Computers in Industry, 2020, 123: 103335
29. Belhadi A, Zkik K, Cherrafi A, Yusof S M, El fezazi S. Understanding big data analytics for manufacturing processes: insights from literature review and multiple case studies. Computers & Industrial Engineering, 2019, 137: 106099
30. Qin W, Chen S, Peng M. Recent advances in industrial internet: insights and challenges. Digital Communications and Networks, 2020, 6(1): 1–13
31. Hermann M, Pentek T, Otto B. Design principles for industrie 4.0 scenarios: a literature review. Dortmund: Technische Universität Dortmund, 2015, 45
32. Boyes H, Hallaq B, Cunningham J, Watson T. The industrial internet of things (IIoT): an analysis framework. Computers in Industry, 2018, 101: 1–12
33. Sisinni E, Saifullah A, Han S, Jennehag U, Gidlund M. Industrial internet of things: challenges, opportunities, and directions. IEEE Transactions on Industrial Informatics, 2018, 14(11): 4724–4734
34. Weyrich M, Ebert C. Reference architectures for the internet of things. IEEE Software, 2016, 33(1): 112–116
35. Fathoni H, Yang C T, Chang C H, Huang C Y. Performance comparison of lightweight kubernetes in edge devices. In: Proceedings of the 16th International Symposium on Pervasive Systems, Algorithms and Networks. 2019, 304−309
36. Tao Z, Xia Q, Hao Z, Li C, Ma L, Yi S, Li Q. A survey of virtual machine management in edge computing. Proceedings of the IEEE, 2019, 107(8): 1482–1499
37. Fogli M, Kudla T, Musters B, Pingen G, Van den Broek C, Bastiaansen H, Suri N, Webb S. Performance evaluation of kubernetes distributions (K8s, K3s, KubeEdge) in an adaptive and federated cloud infrastructure for disadvantaged tactical networks. In: Proceedings of 2021 International Conference on Military Communication and Information Systems (ICMCIS). 2021, 1−7
38. Bauer M, Bui N, De Loof J, Magerkurth C, Nettsträter A, Stefa J, Walewski J W. IoT reference model. In: Bassi A, Bauer M, Fiedler M, Kramp T, Kranenburg R, Lange S, Meissner S, eds. Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model. Berlin: Springer, 2013, 113−162
39. Yu S, Huang Y, Du T, Teng Y. The proposal of a modeling methodology for an industrial internet information model. PeerJ Computer Science, 2022, 8: e1150
40. Wang T, Mou X, Hu J, Wang R, Wo T. Two-stage scheduling of stream computing for industrial cloud-edge collaboration. In: Proceedings of 2022 IEEE International Conference on Joint Cloud Computing (JCC). 2022, 57−64
41. Khoshnevis S. A search-based identification of variable microservices for enterprise SaaS. Frontiers of Computer Science, 2023, 17(3): 173208
42. Dias J P, Restivo A, Ferreira H S. Designing and constructing internet-of-things systems: an overview of the ecosystem. Internet of Things, 2022, 19: 100529
43. Liang W, Zheng M, Zhang J, Shi H, Yu H, Yang Y, Liu S, Yang W, Zhao X. WIA-FA and its applications to digital factory: a wireless network solution for factory automation. Proceedings of the IEEE, 2019, 107(6): 1053–1073

44. Li Q, Yu Z, Xu H, Guo B. Human-machine interactive streaming anomaly detection by online self-adaptive forest. Frontiers of Computer Science, 2023, 17(2): 172317
45. Kieu T, Yang B, Guo C, Jensen C S. Outlier detection for time series with recurrent autoencoder ensembles. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. 2019, 2725−2732
46. Mou X, Wang R, Wang T, Sun J, Li B, Wo T, Liu X. Deep autoencoding one-class time series anomaly detection. In: Proceedings of 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2023, 1−5
47. Wang R, Liu C, Mou X, Gao K, Guo X, Liu P, Wo T, Liu X. Deep contrastive one-class time series anomaly detection. In: Proceedings of the International Conference on Data Mining. 2023, 694−702
48. Lei Y, Li N, Guo L, Li N, Yan T, Lin J. Machinery health prognostics: a systematic review from data acquisition to RUL prediction. Mechanical Systems and Signal Processing, 2018, 104: 799–834
49. Har-Peled S, Raichel B. The fréchet distance revisited and extended. ACM Transactions on Algorithms, 2014, 10(1): 3
50. Zhang M, Yan J. A data-driven method for optimizing the energy consumption of industrial robots. Journal of Cleaner Production, 2021, 285: 124862
51. Yan J, Zhang M. A transfer-learning based energy consumption modeling method for industrial robots. Journal of Cleaner Production, 2021, 325: 129299
52. Qiang J, Zhang F, Li Y, Yuan Y, Zhu Y, Wu X. Unsupervised statistical text simplification using pre-trained language modeling for initialization. Frontiers of Computer Science, 2023, 17(1): 171303
53. Cardellini V, Grassi V, Lo Presti F, Nardelli M. On QoS-aware scheduling of data stream applications over fog computing infrastructures. In: Proceedings of 2015 IEEE Symposium on Computers and Communication (ISCC). 2015, 271−276

Rui Wang received his MS degree from the School of Mechanical Engineering, Beihang University, China in 2017. He is currently a PhD candidate at the School of Computer Science and Engineering, Beihang University, China. His research interests mainly include cloud computing, IIoT, and time series analysis.

Xudong Mou received her BS and MS degrees in the School of Computer Science and Engineering, Beihang University, China in 2017 and 2021, respectively. She is working towards a PhD degree at the School of Computer Science and Engineering, Beihang University, China. Her research interest is time series anomaly detection.

Tianyu Wo (Member, IEEE) received his BS and PhD degrees in computer science from Beihang University, China in 2001 and 2008, respectively. He is a professor at the College of Software, Beihang University, China. His current research interests include distributed systems and IoT.

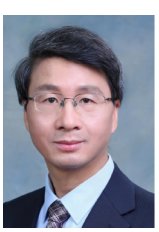
Mingyang Zhang received his BE degree from the School of Mechatronics Engineering, Harbin Institute of Technology, China in 2017. Now he is a PhD candidate in the School of Mechatronics Engineering, Harbin Institute of Technology, China. His research interests include IIoT and cloud-based process optimization of industrial robots.

Yuxin Liu received her BS degree from the School of Electronic Information Engineering, Central South University, China in 2019. She received her MS degree at the School of Computer Science, Beihang University, China in 2022. Her research interest is the knowledge graph.

Tiejun Wang received her BS degree from the School of Information Science and Engineering, Shandong Agricultural University, China in 2020. She is currently working towards a PhD degree at the School of Computer Science and Engineering, Beihang University, China. Her research interest is time series analysis.

Pin Liu received his PhD degree from the School of Computer Science and Engineering, Beihang University, China in 2022. Currently, he is a lecturer at the School of Information Engineering, China University of Geosciences (Beijing), China. His research interest is time series data augmentation.

Jihong Yan is a professor and the Deputy Dean of the School of Mechatronics Engineering, Harbin Institute of Technology, China. She has led several National Key R&D Programs and NSFC projects. She has published over 150 papers with more than 2000 citations. Her research interests include intelligent manufacturing, IIoT, and integrated optimal operation of manufacturing systems.



Xudong Liu is a professor at the School of Computer Science and Engineering, Beihang University, China. He has led several China 863 Programs and government projects. He has published over 100 articles. He holds more than 20 patents. His research interests include software middleware technology, software development methods and tools, large-scale information technology projects, and the application of research and teaching.