IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 15, 2019, accepted July 5, 2019, date of publication July 10, 2019, date of current version December 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927743

A Novel Cloud Platform for Service Robots

JIN LIU, FENGYU ZHOU, LEI YIN, AND YUGANG WANG
School of Control Science and Engineering, Shandong University, Jinan, China

Corresponding author: Fengyu Zhou (zhoufengyu@sdu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1302400, in part by the National Natural Science Foundation of China under Grant 61773242, in part by the Key Program of Scientific and Technological Innovation of Shandong Province under Grant 2017CXGC0926, and in part by the Key Research and Development Program of Shandong Province under Grant 2017GGX30133.

ABSTRACT With the development of computer technology and artificial intelligence (AI), service robots are widely used in our daily life. While the manufacturing cost of the robots is too expensive for most small technology companies. The biggest technical limitations are the design of the robot service and the resources sharing of the robot groups. As far as we know, there is no complete and open-source service robot cloud service platform. To solve the above problems, in this paper, a novel robot cloud platform called cloud robotics intelligent cloud platform (CRICP) is designed, which consists of gateway layer, an interface layer, service pool, and algorithm layer. Gateway layer mainly solves the problem of robot access control and service invocation requests scheduling. In addition, a standardized access method is proposed to overcome the problem that heterogeneous service robots cannot access the cloud platform. Interface layer is responsible for protocol injection, including motoring protocol, a management protocol, and other algorithms invocation protocol or service invocation interfaces protocol. Service Pool consists of different kinds of robot services which could scale based on the historical data analysis. In Algorithm layer, we can implement machine learning (ML) algorithm, deep learning (DL) algorithm, distributed algorithm, and so on. Finally, voice recognition service invocation experiment and cloud service dynamic scaling test are taken as an example to verify the availability and accuracy of our platform. Moreover, the compared results with a local framework and SOA also verifies the superiority of our platform.

INDEX TERMS Cloud platform, robotics, service robots, service pool, multi-layer.

I. INTRODUCTION

Nowadays, the aging phenomenon of population becomes more and more serious. The cost of labor continues to rise and the role of service robots becomes increasingly important. Traditional service robots are designed as a single service unit which is mainly composed of driven system, control system and sensor system. However, service robots are hard to be promoted. There are two reasons to explain the above questions. On one hand, due to the limitation of its own hardwares and resources, a huge technical bottleneck still exists in environmental perception and understanding, emotion interaction, intellectual development and other aspects. On the other hand, as the service robot software technology keeps updating, more superior hardwares are needed to adapt, and the cost of robots will become higher as well.

The associate editor coordinating the review of this manuscript and approving it for publication was Yingxiang Liu.

With the development of cloud computing [1], a great number of cloud technologies are emerging which have changed the way of software application and resources sharing. Driven by these technologies, different kinds of cloud platforms have been designed for different purposes [2], [3]. Many services paradigms evolved from that since then, such as Infrastructure as a Service (IaaS) [7], Platform as a Service (PaaS) [8] and Software as a Service (SaaS) [9]. Service developers could use the huge resources and powerful computing power everywhere by the above ways [5]. The cloud could offer various advantages and provide a good idea for solving the existing problems of service robots.

The continuous developments of these above technologies have spawned the concept of cloud service robots. The new paradigm can take the advantages of parallel computing and data sharing. It also makes the robot lighter and smarter. A networked robotic system model was introduced in [10], which is composed of robotic devices and connected via a wired or wireless network. The model was similar to a

stand-alone robot which improves the robot's processing power and intelligence level tremendously as well. However, the model was also limited by resources and information sharing. In 2010, James J. Kuffner proposed the concept ''cloud robotics'' [4], which was different from the traditional design way of robot. He regarded the service robots as an agent. Robots could offload complex data processing, planning, decision-making and other services to the cloud. The knowledge capability of robots was extended and beyond the limit of physical ontology. Some key problems in service robots have also been solved. Thereafter, researchers tried to make robots lighter, cheaper and smarter. Thus, a new framework which can provide sufficient needs for service robots is imminent.

Cloud service platform was a new paradigm which is different from tradition cloud computing platform. The earliest service robot cloud platform was based on a typical distributed computing module [11], such as M2M/M2C (Machine to Machine/Machine to Cloud), ROS (Robot Operating System) [12] and UNR-PF (Ubiquitous Network Robot Platform) [13]. Most of them were used in a small scale. The distributed monomer was a mainstream development framework. Moreover, the monomer could make efficient resource-sharing and programming, while developers were forced to use a specified programming language and style, and the usage was limited in small environments. Meanwhile, some researchers tried to apply high performance computing (HPC) to cloud robots [14], and combined Hadoop framework with simultaneous localization and mapping (SLAM) [6]. There are problems in all the above ways of setting up a cloud platform. Most of them can accomplish in a small network-based environment, while they can't get out of the laboratory.

Therefore, based on the previous work of the predecessors, a new service robot cloud platform is presented. The platform could make heterogeneous service robots access the platform more easily and get the services they need. We also provide more easier way for developers to design light, stateless and efficient service module. Finally, voice recognition service invocation by four kinds of robots is adopted to verify the new platform.

The paper is structured as follows. Related work discusses previous work about robot cloud platform in Section II. In Section III, the architecture of our platform is designed in detail. Service encapsulation and access standard are proposed in IV. In Section V, four experiments with voice recognition service are taken as an example to verify our proposed platform. Section VI provides conclusions of the paper and future works.

II. RELATED WORK

Service robot cloud platform was first developed in multi-sensor cloud platform [15], [16]. Researchers visualized the physical sensors into cloud sensors. The platform could separate system management and sensor data in this way, and eliminate the heterogeneity between service robots and the cloud platform. It could also achieve fast response in small scenes and small scales. In [17], an improved sensor network was proposed, which was composed of sensor networks, gateways, and a testbed server. Users could get more accurate experimental data and simulation data from the platform. However, the functionality of the above platform is too simple, which focuses mainly on the data processing of sensors.

In 2011, the European scientists started a groundbreaking service robot cloud project called ''RoboEarth'' [18]. The project aimed to break through the original single technology limit, and build a huge resources and computing power repository. Robots could share their own information and knowledge. Thus they could evolve quickly by learning from each other. Developers used the resources on the platform to improve the accuracy of service robots in context awareness and behavior planning. Thereafter, Google and Microsoft developed a new project called ''RobotBrain'' [19], which tried to make the platform as a brain of all the robots and decision for the robots quickly and accurately. In 2012, Narita *et al.* [20] proposed a robot platform and designed a communication protocol called robot service network protocol (RSNP) that it could assign the tasks according to the user's requirements. After that, they proposed Jeeves based on RSNP [21]. Multi-machine collaboration of cloud robots was realized through the interaction of information, the dynamic search and distribution of cloud resources.

Rayuta [22] was regarded as a famous cloud engine and a service platform framework. It used cloud resources and parallel computing algorithms to achieve robotic services, and provided solutions for robots on monitoring data, cloud data sharing and services schedule. A bridge of cloud computing and robotics was proposed in [23]. The author attempted to create a large centralized robot cloud platform and build a robotic bridge between the physical world and the cyber world. The author proved that building a large robot center is better than multiple independent centers. In [24], a Robot Cloud Center (RCC) based on Service-oriented architecture (SOA) and cloud computing was proposed. The part of each function of the robot was packaged into a service in the SOA framework. RCC was reusable, visualized, scalable and easy to upgrade. A family service robot platform architecture was proposed in [25], which could be seen as a RaaS unit module and developed by using Lua programming language.

In order to make it more flexible and easier to access service robot cloud service platform, our laboratory has done a lot of work in recent years. In 2005, Chen *et al.* [26] designed a cloud computing platform interface model based on SOA architecture. By using service description and reconstruction, the differences among systems were eliminated. In 2017, based on the previous work, Zhou and Yin [27] proposed service quality assessment and scheduling algorithm of robot cloud platform based on SOA interface layer. The platform could provide efficient and computational intensive services for distributed heterogeneous robots. However, the above works didn't have a systematic management of

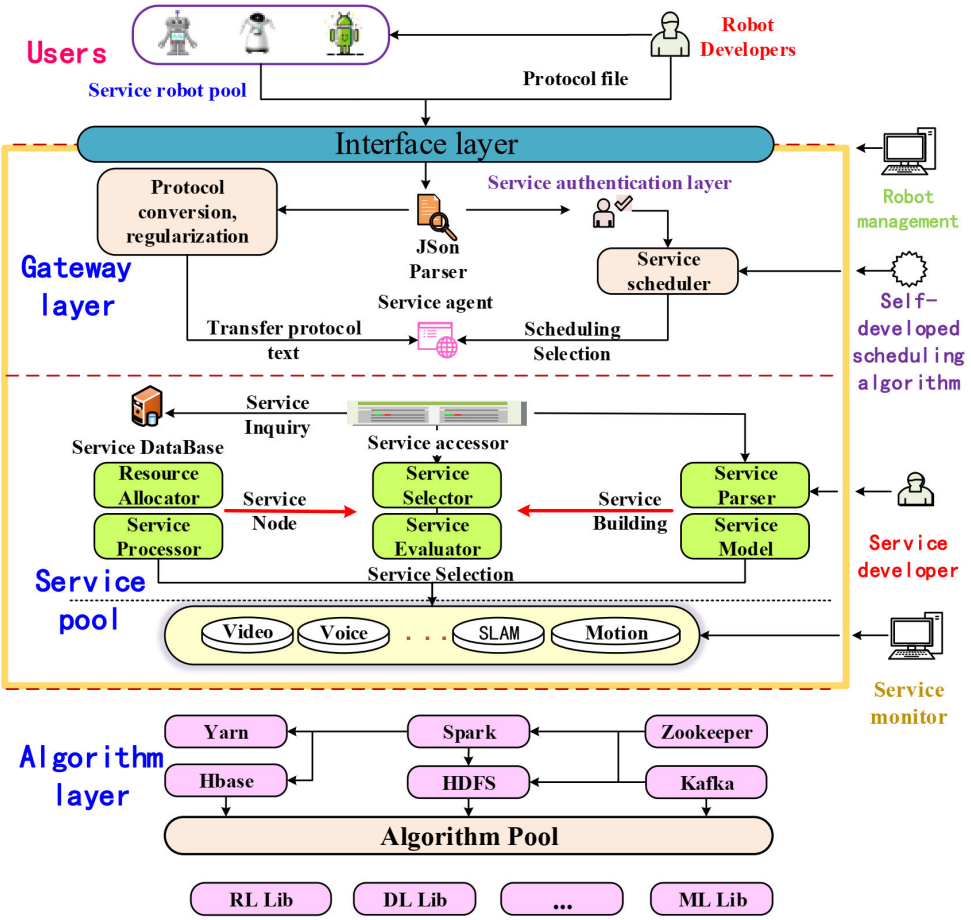


FIGURE 1. The overall architecture of the service robot cloud platform.

service robots. Therefore, it's essential to design a new service robot cloud platform.

III. ARCHITECTURE OF THE SERVICE ROBOT CLOUD PLATFORM

The whole architecture is shown in Fig.1. It is based on the distributed computing architecture Spark [28] engine and divided into a Gateway layer, a Service Pool and an Algorithm layer. The periphery of the entire architecture is the Interface layer which could connect all layers.

The Gateway Layer is mainly responsible for protocol conversion, robot authentication and service scheduling. The robots or developers send the service requests according to specified JSON file format. The JSON parser parses the data, protocol text, and service requirements. The protocol converter will implement standardized conversion of heterogeneous robot protocols. Service authentication layer uses three-access authentication control to implement the legality check of the robot. After the verification, the service scheduler invokes the service by applying the scheduling algorithm.

The Service Pool is primarily responsible for service parsing and configuration, service evaluation and selection, and cloud service storage. Service developers submit the robot cloud service codes. The basic services and models will be encapsulated and injected into the Service Pool through the parsing layer and model layer. After the service request is parsed by the service accessor, the service selection layer performs service evaluation and selection, and the result is assigned to the service node. The service node layer applies for resources according to the service request and deals with the service request.

The Algorithm Layer is mainly responsible for algorithm invocation. It is built on an elastic physical cluster to provide sufficient computing resources for complex or a large-scale operations. We adopt Spark as the core computing engine. The file sharing system is based on the Hadoop Distributed File System (HDFS). The resources management utilizes the Yet another resource negotiator (Yarn) framework. The distributed data storage uses the column family database named HBase. The computing log monitoring is composed of kafka system. The consistency and accuracy of data are fully guaranteed by the distributed application coordination service Zookeeper. This layer also provides machine learning (ML) library, deep learning (DL) library, etc. And it can also be implemented developer's algorithm through the Interface Layer.

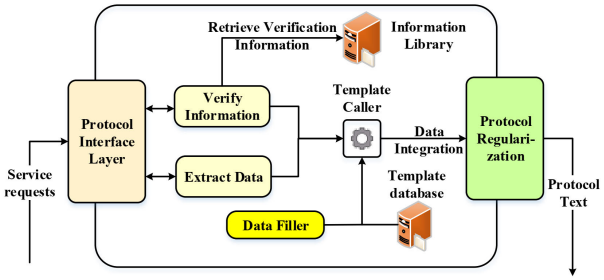


FIGURE 2. Structure of the protocol conversion layer.

The Interface Layer is responsible for providing the appropriate connection interface among the above layers. It increases the flexibility in development, usage and management indeed. Robot managers could have a detailed management of the robots through the Interface Layer. Algorithm developers can implement their own efficient scheduling algorithm. Service developers can have an easier way to submit robot services designed or improved by themselves, and they can also have an overview about the service utilization with service monitor.

A. GATEWAY LAYER

As a middleware for connecting external service requests and internal service invocations, the Gateway Layer consists of a protocol conversion layer, a service authentication layer and a scheduling layer. The layer parses protocol file, authenticates the service robot and implements service scheduling algorithm. The service robot could invoke the cloud service based on the defined interface through Interface Layer.

1) PROTOCOL CONVERSION LAYER

The Protocol conversion layer mainly checks and converts the parsed protocol files, which is shown in Fig.2. When the service robots send a service request, the layer retrieves the data and other involved information from JSON parser firstly, and compares its verification information with the data from the Information Library. The results of the verification will be sent to Gateway Layer as the processing information. Then, the template caller extracts the data template from the template database. The data filler backs up the data template and initializes the variables therein. Finally, the regularized protocol file is verified and formatted after data integration. The final file will be output to the service agent through Interface Layer.

2) SERVICE AUTHENTICATION LAYER

The Service authentication layer performs security authentication on the service robots. The simple security authentication model is shown in Fig.3.

The security certification process is as follows.

Step one, the Authentication layer retrieves the ontology information in Service Information Database according to the authentication request, and checks ID and secret key.

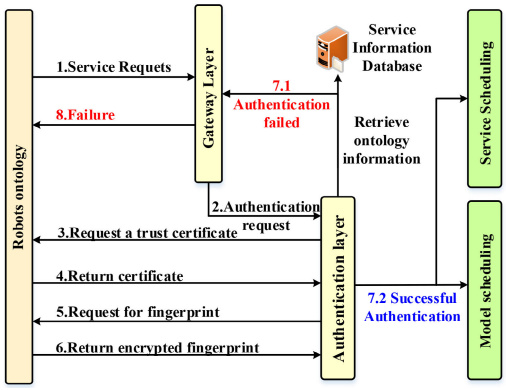


FIGURE 3. Security authentication model.

Step two, the platform asks for a trust certificate obtained by the service robot developer on the platform from the service robot. After the certificate is received, the platform will check it again.

Step three, the platform requests to compare the fingerprint which is mainly composed of Media Access Control Address (MAC). The service robot receives the information and returns the encrypted fingerprint. After that, the Authentication Layer will check it again.

Step four, the Model scheduler selects the existing and appropriate service model in the pool according to the requirements of the service robot. The service scheduler completes the scheduling of the cloud services.

The Service authentication layer utilizes a three-level access control model to ensure the security and robustness of the cloud platform fully, and the credibility of the service robot cloud platform is enhanced as well.

3) SCHEDULING LAYER

The scheduling layer uses the service scheduler which can be applied developers' algorithm through the Interface Layer. It ensures that the cloud platform meets the real-time invocations requirement when a large-scale service robot accesses. The Scheduling layer model is shown in Fig.4.

The master scheduler schedules the cloud service in Service Pool by applying the algorithm designed by the researchers through the Interface Layer. The scheduling monitor obtains the running status of master scheduler and writes the results to the log collection repository. When the requests monitor detects a large-scale service invocations, the scheduling algorithm optimizer updates the current service invocation strategy by using corresponding optimization algorithms. The auxiliary scheduler will help to alleviate the load balancing pressure to ensure the stability when a large-scale cloud service requests access the platform.

B. SERVICE POOL

The Service Pool is mainly responsible for cloud service storage, construction and processing, which is composed of service node, a service controller, a service builder and a basic service storage pool. Service Pool could also implement the

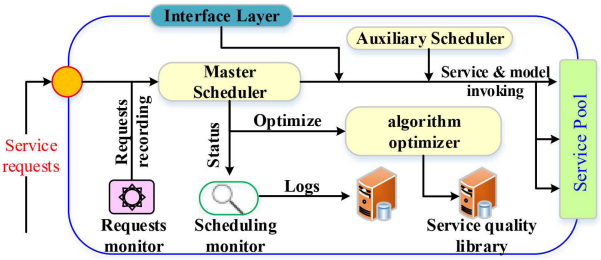


FIGURE 4. Service scheduler model.

query and invocation of the requested service, the construction and injection of the service, thus the cloud knowledge could be built and the intellectual evolution of robots could be realized. The storage capacity is theoretically considered to be infinite, and the dynamic expansion can achieve a hundredfold increase in cloud services. The simultaneous access and service real-time invocation of a great number of service robots could be satisfied.

1) SERVICE POOL HIERARCHY

The Service Node is deployed on a cloud cluster based on the distributed architecture which configures the resources required by the service properly. The Service Processor will invoke the cloud service and process the data transmitted by the robot.

The Service Selection layer selects, evaluates the services and writes the optimal services into the service quality database. The evaluation algorithm could also be implemented through the Interface Layer. A simple way to evaluate is implemented, including the number of service invocations (CNU), single robot execution accuracy (REA), single cloud resource occupancy rate (CRU) and single user evaluation (UCS). The formula is as follows.

$$S = \omega_h \frac{\sum REA + UCS}{CNU} + \omega_l \frac{\sum CRU}{CNU} \tag{1}$$

In (1), ω_* is the evaluation weight value. The platform will recommend the best service according to the formula.

The Service Building layer parses the service script submitted by service developers and detaches a monitor model of service so that managers could have a clear understanding of service execution. The way of building a service will be introduced in next section.

As the container of service robot cloud service increases, the basic service storage pool will adopt distributed parallel expansion storage, the capability of service virtualization awareness, and multi-node hot standby.

2) ELASTIC SCALING STRATEGY

When a large-scale service requests are received, the quality of service on the platform decreases. Aiming at this problem, based on the historical data and threshold method [29], this paper proposes a cloud service elastic scaling strategy based on request state. It realizes the dynamic configuration of cloud resources and improves the utilization of cloud resources.

By analyzing the basic characteristics of the service robot cloud service, without loss of generality, the cloud service characteristics of the service robots are described in the following three aspects.

Aspect one, occupied bandwidth (OB) represents the bandwidth that a service needs to consume during operation.

Aspect two, occupied memory (OM) represents the memory that a service needs to consume during operation.

Aspect three, service cost (SC) represents the cost of the service from being called to being released.

To describe the elastic scaling strategy, the following four definitions are given.

Definition one, CPU utilization indicates the CPU usage at the service invocation time t.

$$CU = \frac{\sum_{i=1}^{m} C_i^{used}}{CT} \tag{2}$$

where CU is the utilization of CPU. C_i^{used} is the amount of CPU used by the i-th server currently. CT is the total amount of CPU in the cluster and m is the total number of servers in the cluster.

Definition two, memory utilization indicates the memory usage at the service invocation time t.

$$MU = \frac{\sum_{i=1}^{m} M_i^{used}}{MT} \tag{3}$$

where MU is the utilization of memory. M_i^{used} is the amount of memory used by the i-th server currently. MT is the total amount of memory in the cluster and m is also the total number of servers in the cluster.

Definition three, the service request collection period T, the collection interval T_0, the number of acquisitions k. When the number of detected services exceeds the CU and MU thresholds. The system samples the number of service requests every T_0 and a total of total k times.

Definition four, the service request rate SA indicates the ratio of the service request in the service request sequence and time within a short time Δt.

The elastic scaling strategy is shown in Fig.5.

The specific workflow is as follows.

1) The cloud service platform initializes MU, CU, T, T_0, k and SA. After that, the platform extracts the state parameters of the platform, and waits for the service requests of service robots.
2) When the cloud platform status parameter MU is greater than 85% and CU is greater than 90%, the platform will sample the request data parameter of the service robot according to the value of T, T_0 and k. When MU is lower than 10% and CU is lower than 8%, the state parameters of the cloud service deployed on the cloud service platform will be extracted. In other cases, the platform will be monitored again in real time.

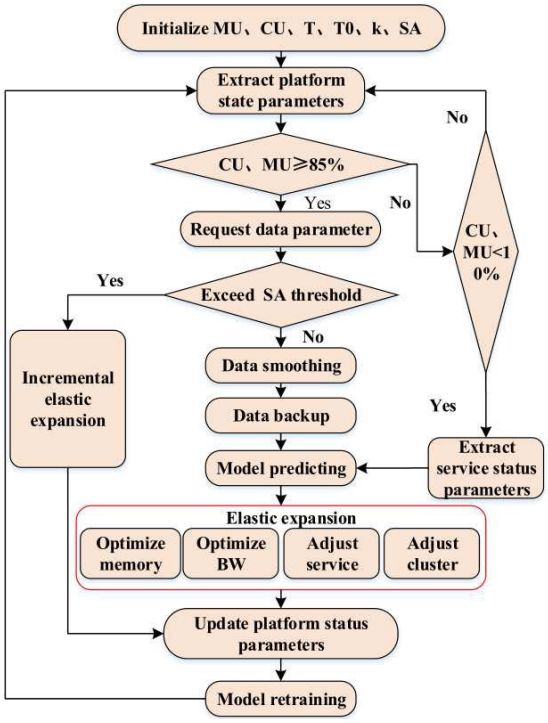


FIGURE 5. Scaling strategy flow diagram.

3) When the service request rate SA exceeds the set threshold, the cloud service platform will perform incremental elastic scaling according to formula 4. The platform configures the resources such as memory and bandwidth of the cloud service to satisfy the requests. The current platform state parameter is updated according to the changing of service parameter and the historical state parameter database at the time of service invocation.

$$\Delta = \lambda(SA' - SA) \tag{4}$$

where Δ is the increment. λ is the coefficient of the increment. SA' is the current rate of the requests, while SA is the set threshold.

When the service request rate is less than the SA threshold, the platform will sample the request data and the service parameters for k times. Perform data smoothing processing and save the processing results to HDFS.

4) The cloud platform invokes the autoregressive prediction model [30] from the Algorithm Layer and obtains the service extension prediction value. After that, the scheduler combines the specific attributes of the service OB, OM and SC with the needed resources, and optimizes the deployment of the platform.

IV. SERVICE ENCAPSULATION AND ACCESS STANDARD

A. SERVICE ENCAPSULATION

The service designed by developers will be encapsulated by a specified way through the Interface Layer. The structure of the service building layer is shown in Fig.6.

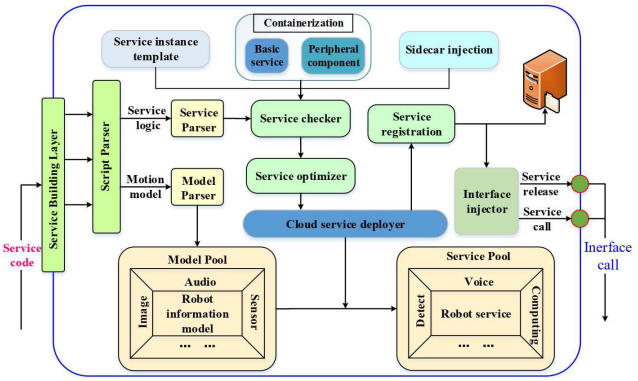


FIGURE 6. Structure of the service building layer.

The construction process is as follows.

Firstly, the developer's service code is submitted to the Service Building Layer. The script parser decomposes the code into service scripts and model scripts. When the service associates movement, sensors and etc. The model is usually needed to provide the service model which makes the managers have an overview about the service. The service scripts contain the main service logic.

Secondly, the Model Parser obtains the included model and injects it into Model Pool. Meanwhile, the Service Parser will parser the service logic script and transmit it to Service checker.

Thirdly, after the verification, the basic component are separated including package, deep-learning model, and the peripheral architecture including network, authentication. It is made as a container which could be used on any platforms. The service instance template is applied for the container, and sidecar will be injected, thus the service traffic could be monitored.

Fourthly, the service optimizer optimizes the resources including bandwidth, memory and other resources required by the service. After that, the service is injected into the Service Pool and registered into the service registration database through the Registrar.

Finally, the service is injected the invocation interface through the Interface injector. The service robots could invoke the service in specified format file through the released interfaces.

B. ACCESS STANDARD

The access standard makes a simple way to invoke the service through the Interface Layer. It mainly contains the interaction protocol between the service robot and cloud service platform. The design requirement is to ensure that a large-scale robots can interact with the cloud platform efficiently. Therefore, this paper designs an interface protocol format based on JSON language and it is shown in Fig.7.

This simple interface format defines the necessary parameters for the service robot legality verification, invocation parameters and entity service parameters. Heterogeneous robots can get the same processing and results according

```
"Robot_Id":10001, //Robot registration number
"token":8acf29cbacdb4e068f5a2af4d7c87fe4, //corresponding secret
"Id_Key":"001", //corresponding key
"ApplyService": "facedetect", //the kind of service invoked
"data":"files", //the data needs to be transmitted
"Test_Infor":{  // verification information
    "date":"2019-04-17",
    "password": "lemon",
    "company": "SDU"
},
"Append_Infor":{ // extended information
    "Time":{
        "Delay":30,
        "WaitOut":15
    },
    "RobotKind": "Bigv1",
    "RobotSensor_Url": url
}
... //designed fields according to the service developer
```

FIGURE 7. Structure of the service building layer.

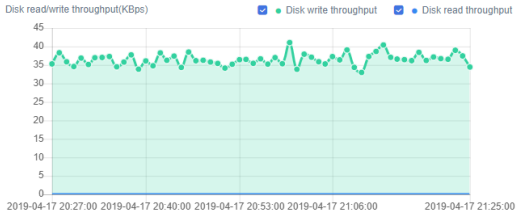


FIGURE 8. Disk usage.

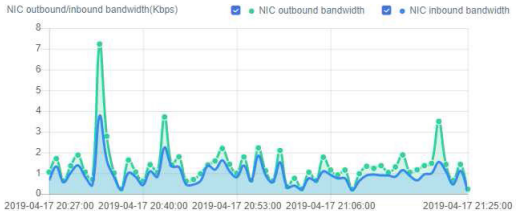


FIGURE 9. NIC usage.

to the specific standard access. The developers could have a flexible way to customize the file content according to the characteristics of the service robots, which solves the access problem of the heterogeneous service robots in a simple and better way. The developers could also interact with the platform conveniently, and the robustness of the service invocation can be ensured.

In order to make it clear, here we implement the cluster monitor, robot company management interface and service management interface that is shown in Fig.8, Fig.9, Fig.10 and Fig.11.

As can be seen from Fig.8 and Fig.9, we could know the usage of disks and NIC when the robots invoke services. Fig.10 shows an easier way to manage the companies and robots which want to register into our cloud platform. As can be seen from Fig.11, the state of the running service could be monitored. The contents in the read rectangle which is in the upper middle part of the picture shows the service name. A complete understanding and status of the specific service can be seen from the little chart with two red lines. In the lower left corner of the picture, a services list is recommended for the those robots registered on the platform.

V. EXPERIMENT AND ANALYSIS

In this section, we take a voice recognition service as an example. Several experiments are designed to verify the ability of our proposed platform when different scale and different kinds of service robots access.

A. EXPERIMENT SETUP

The service robots used in the experiments are shown in Fig.12.

These are intelligent escort robot, humanoid robot, desktop robot, and quadruped robot developed independently by our laboratory or jointly developed by the cooperation company.

The escort robot is equipped with ARM architecture Raspberry Pi II and centos/7 operating system. The desktop robot is equipped with Raspberry Pi III and Ubuntu16.04 operation system. The humanoid robot designed by UBTECH Company is equipped with more sophisticated motion control and voice system, and operating system is android. The quadruped robot is equipped with flexible quadruped motion system and with ubuntu operating system. All the service robots are equipped with microphone arrays and wireless modules from the project partner AISPEECH Company.

The service robot cloud platform adopts 16 cloud servers provided by the project cooperation unit Inspur Cloud Platform Development Department. The single server configuration is shown as Table.1. The number of CPU cores per server is 16. The memory is 32G, and the data disk is 1TB. To make the platform easier to develop, ubuntu operating system is adopted. In order to develop the service, we use the integrated development tools named IntelliJ IDEA (IDEA) and the programming language is python and Java.

B. SERVICE INVOKING

The above four kinds of robots are used in the experiment to call the speech recognition service deployed on the cloud service platform.

All the robots are deployed in the same room under the same network. Here, an escort robot called DaZhi (DZ in short) is taken as an example. The process of interaction is shown in Fig.13.

TABLE 1. The information about the development.

Items	Configuration
Cores	16
Memory	32G
Data disk	1TB
Operating system	ubuntu16
BGP	10M
Software	IDEA
Programming language	Python, Java

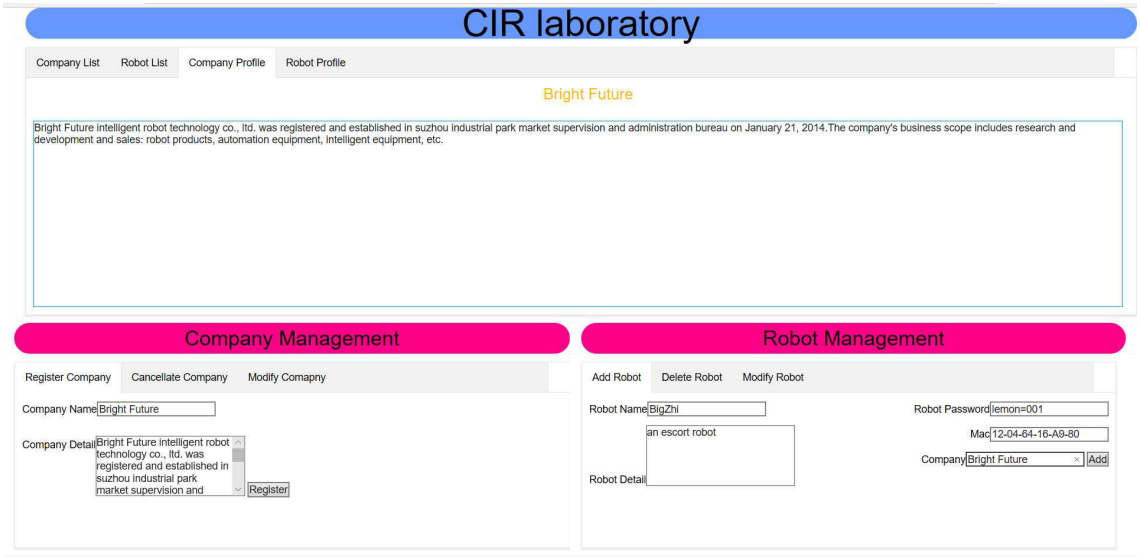


FIGURE 10. Company management.

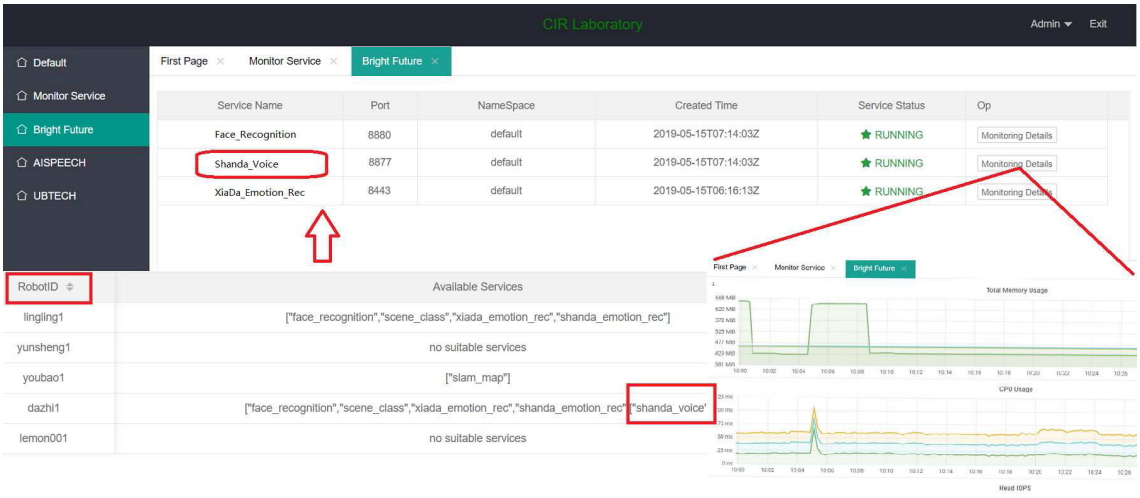


FIGURE 11. Service management.

Firstly, the Gateway Layer will make validation when DZ tries to invoke services. Access token will be returned while DZ registers on the platform or tries to register again.

Secondly, registered DZ can apply for desirable services. Once DZ has signed its sensors, the cloud service platform will recommend a list of available services. Otherwise, DZ can only get one recommend service.

Thirdly, the service invocation requests will be accepted by the platform. The scheduler gets the service from Service Pool, and the service will invoke the corresponding speech recognition algorithm and other relevant algorithms from Algorithm Pool. After the process is finished, the recognition results will be returned to DZ.

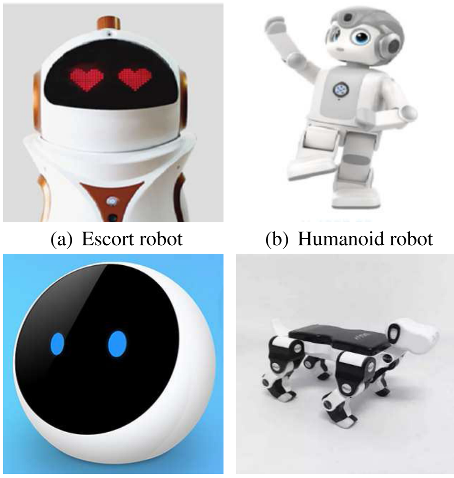
In the above process, the invocation protocol file is shown in Fig.14.

In order to verify the performance of cloud platform, we use four types of robots. The number of each type is 10. Robots are randomly combined to invoke the service and given a verification on average delay and accuracy when the number of the accesses is 10, 20, 30 and 40.

1) THE AVERAGE DELAY TEST

The average delay shows the response of the cloud service platform when the service is invoked. The lower the average delay is, the faster the responses of the platform will be.

The relationship between the number of service invocations and the average delay is shown in Fig.15. As can be seen from the figure, when the number of accesses is 10,20,30,40, respectively, the average delay time are 10.3ms, 10.7ms, 11ms and 11.8ms. As the number of service invocations increases, the number of the fluctuation spikes and the peak value of spikes also increases. This is due to the frequent service invocations from service robots and fluctuations in network conditions.



(a) Escort robot (b) Humanoid robot

(c) Desktop robot (d) Quadruped robot

FIGURE 12. Service robots.

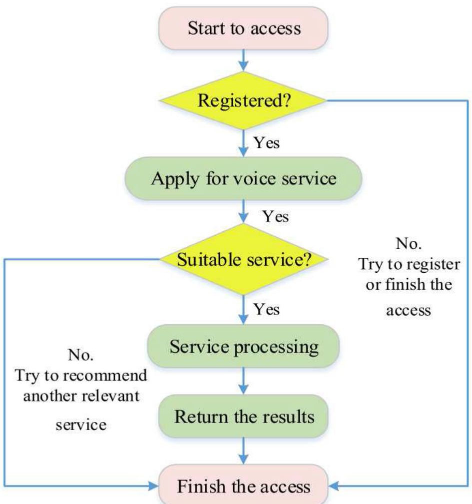
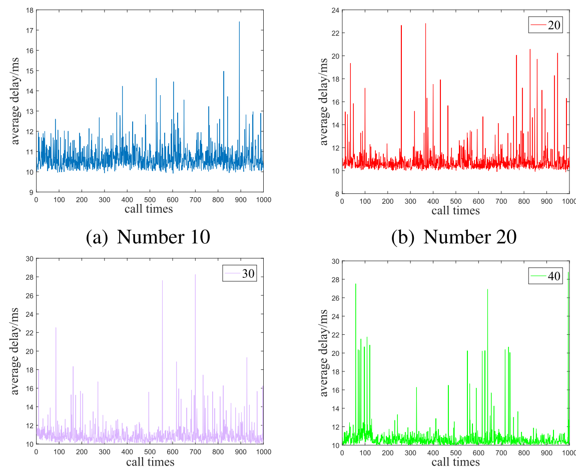


FIGURE 13. The process of interaction.


```
{
    "Robot_Id":11021,
    "token":"8acf29cbacdb4e068f5a2af4d7c87fe4",
    "ApplyService": "voice_rec",
    "data":"/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAIBAQEBAQIBAQECAgICAgQDAgICAgUEBAMEBgUGBgYFBgYGBwkIBgcJBwYGCAs
    ICQoKCgoKB....wDi54p8N+XG8scOhreorvGVyWcdQGXdt3KSOWAPUV9IV8hWX/BRfxfr2mhR8PbOwuo3CP5N+0qSEg92QFRkdME4718w/
    E2L4reJf7R8V+GvifPpAvdRaW7s1tYpowXlaRgvmIeMknB4yScZJJ9o/Zz/AGuvFHwV+F9lpFzZvrNtYQvJdpeS/vLh2bcziT7yMWJJzuHtnmvVtD
    /4Kh/CS/cLq3w98R2+Ytx+zmCXDcccyJxz1/Sv/9k=",
    "Test_Infor":{
        "date":"2019-04-17",
        "password": "lemon",
        "company": "SDU"
    },
    "Append_Infor":{
        "Time": {
            "Delay":30,
            "WaitOut":15
        },
        "RobotKind": "Bigv1",
    }
```


FIGURE 14. Demand file format for service invocation.



(a) Number 10 (b) Number 20

(c) Number 30 (d) Number 40

FIGURE 15. The result of average delay test.

2) THE ACCURACY TEST

Accuracy η is the percentage of the total number when robots invoke services that recognizes the voice successfully which can be illustrated as follows.

$$\eta = \frac{ST}{ST + FT}\% \tag{5}$$

where ST is the number of the service that recognizes the voice successfully. FT is the total number of service that occurs errors or fails to recognize the voice.

The accuracy test results are shown in Fig.16. As can be seen from the figure, when the number of invocations is less than 1000, the accuracy of the voice recognition is above 96%. As the number of accesses increases, the accuracy rate of the robot service invocation remains basically unchanged. When the number of service invocations increases, the accuracy rate decreases. The reason is that multiple service invocations cause the bit error rate to rise temporarily, while the actual service demand can still be met.

3) THE CLOUD SERVICE DYNAMIC SCALING TEST

The elastic scale factor that we propose here represents the elastic scaling ability of the service robot cloud service platform when robots invoke the services. The larger the scale of service invocations becomes, the stronger the scheduling ability and elasticity of the platform will be. The calculation formula of the factor is as follows.

$$\zeta = 1 - \mu_1\left(\frac{\sum_{i=1}^{m}(CU' - CU_0)}{m-1}\right) - \mu_2\left(\frac{\sum_{i=1}^{m}(MU' - MU_0)}{m-1}\right) \tag{6}$$

where ζ is the elastic scale factor. m is the number of service invocations, μ_* represents the weight coefficient. CU' and MU' are the current CPU utilization and the current memory utilization respectively. CU_0 and MU_0 are the threshold.

We set the sampling $T = 100$ms, $SA = 30$ times/ms, $k = 3$, $T_0 = 20$ms. The number of services is 2000, 3000,

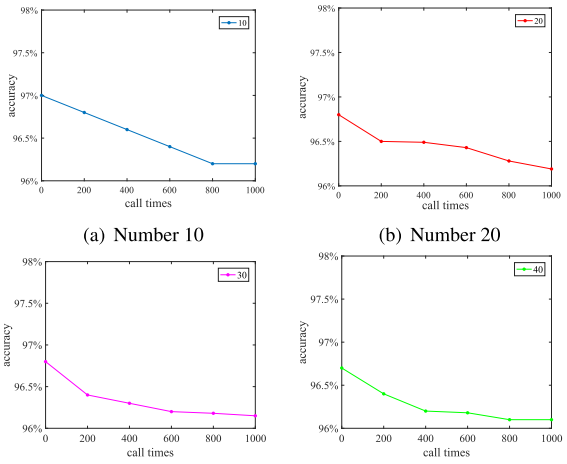


FIGURE 16. The result of accuracy test.

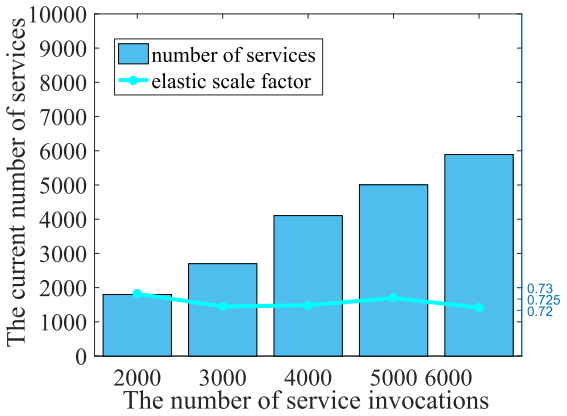


FIGURE 17. The Result of cloud services scale flexibly.

4000, 5000 and 6000. The result is shown in Fig.17. As can be seen from the figure, the current number of services is close to the ideal number of requirements. As the number of service invocations increases, the elastic scale factor has a small fluctuation, while the average of the factor is still higher than 0.72. It can be seen that our proposed platform could achieve the elastic of services, improve service invocation efficiency and reduce the waste of resources.

4) THE COMPARATION WITH OTHER FRAMEWORK

More service invocations on SOA and local framework (e.g. local laptop) to verify the time-savings of CIRCP. The result is shown in Fig.18.

As can be seen from the figure, although the total time consumption increases with the number of service invocations increasing. CIRCP takes the least time to complete the service invocation, and the total time consumption gap becomes larger as well. The total time cost gap is near 5000ms between local framework and CIRCP when the number of service is 6000. The gap will be larger as could be seen from the trend. When run the services on the local framework, it may be met the fundamental demand when fewer service invocations are accessed. while the number of service calls becomes larger,

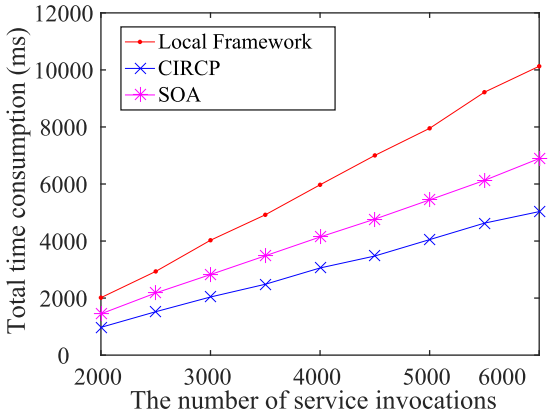


FIGURE 18. The result of total time consumption.

local framework has the worst performance in the three ways. The result also shows that CIRCP has a better performance in time-savings than the above two frameworks.

VI. CONCLUSION

The continuous development of service robot technology and cloud computing have promoted the research of cloud service platforms. While the cost of robot service design and resources sharing remains high, and the technical limitations are still difficult to overcome. This paper proposes a cloud service platform for service robots based on the work of the predecessors. The Gateway Layer and Service Pool are designed in detail to solve the problem of robot access control and service invocation requests scheduling. An access standard is designed for heterogeneous service robots, and an elastic scaling strategy is proposed to improve the stability of the platform. In experiment section, cloud platform is verified through the service invocations. The results show that the platform could be used for the service robots, and could have a better performance than SOA and local framework in time-savings.

In the future work, we will do more research on scheduling to achieve efficient scheduling of the service robot cloud platform. We do also hope that more and more research institutes join us and make a better service robot cloud platform.

ACKNOWLEDGMENT

In addition, the authors thank the anonymous reviewers for providing valuable comments to improve this paper

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] P. Soille, A. Burger, D. De Marchi, P. Kempeneers, D. Rodriguez, V. Syrris, and V. Vasilev, "A versatile data-intensive computing platform for information retrieval from big geospatial data," *Future Gener. Comput. Syst.*, vol. 81, pp. 30–40, Apr. 2018.

[3] S. J. E. Taylor, T. Kiss, A. Anagnostou, G. Terstyanszky, P. Kacsuk, J. Costes, and N. Fantini, "The CloudSME simulation platform and its applications: A generic multi-cloud platform for developing and executing commercial cloud-based simulations," *Future Gener. Comput. Syst.*, vol. 88, pp. 524–539, Nov. 2018.

[4] J. J. Kuffner, "Cloud-enabled robots," in *Proc. EEE-RAS Int. Conf. Humanoid Robot.*, Nashville, TN, USA, 2010.
[5] G. Fylaktopoulos, M. Skolarikis, I. Papadopoulos, G. Goumas, A. Sotiropoulos, and I. Maglogiannis, "A distributed modular platform for the development of cloud based applications," *Future Gener. Comput. Syst.*, vol. 78, pp. 127–141, Jan. 2017.
[6] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "DAvinCi: A cloud computing framework for service robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, May 2010, pp. 3084–3089.
[7] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. ACM Symp. Operating Syst. Princ.*, 2003, pp. 29–43.
[8] Google. *Google Apps Engine*. [Online]. Available: http://www.google.com/apps
[9] M. H. Weier and L. Smith, "Businesses get serious about software as a service," *Information Week*, pp. 46–48, 2007.
[10] *IEEE Society of Robotics and Automation's Technical Committee on: Networked Robots*. [Online]. Available: http://www-users.cs.umn.edu/~isler/tc/
[11] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: Architecture, challenges and applications," *IEEE Netw.*, vol. 26, no. 3, pp. 21–28, May/Jun. 2012.
[12] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009, vol. 3, no. 3, p. 5.
[13] M. Tenorth, K. Kamei, S. Satake, T. Miyashita, and N. Hagita, "Building knowledge-enabled cloud robotics applications using the ubiquitous network robot platform," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2013, pp. 5716–5721.
[14] L. Camargo-Forero, P. Royo, and X. Prats, "Towards high performance robotic computing," *Robot. Auton. Syst.*, vol. 107, pp. 167–181, Sep. 2018.
[15] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure—Physical sensor management with virtualized sensors on cloud computing," in *Proc. NBiS*, vol. 10, Sep. 2010, pp. 1–8.
[16] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, "A survey on sensor-cloud: Architecture, applications, and approaches," *Int. J. Distrib. Sensor Netw.*, vol. 9, no. 2, pp. 917–923, 2013.
[17] A. Kanzaki, T. Hara, Y. Ishi, T. Yoshihisa, Y. Teranishi, and S. Shimojo, "A sensor network testbed integrating multiple networks," in *Proc. 10th Int. Conf. Mobile Data Manage., Syst., Services Middleware*, Taipei, Taiwan, May 2009, pp. 361–362.
[18] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, and B. Schiessle, "Roboearth-a world wide Web for robots," *IEEE Robot. Autom. Mag. WWW Robots*, vol. 18, no. 2, pp. 69–82, 2011.
[19] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Gálvez-López, K. Häussermann, R. Janssen, J. M. M. Montiel, and A. Perzylo, "RoboEarth," *IEEE Robot. Autom. Mag.*, vol. 18, no. 2, pp. 69–82, Jun. 2011.
[20] M. Narita, M. Shimamura, R. Hiura, and T. Yamaguchi, "A common platform for robot services through robot services initiative (RSi) activities," in *Proc. 4th Int. Conf. Ubiquitous Robots Ambient Intell. (URAIS)*, South Korea, 2007, pp. 1029–1034.
[21] S. Nakagawa, N. Igarashi, Y. Tsuchiya, M. Narita, and Y. Kato "An implementation of a distributed service framework for cloud-based robot services," in *Proc. 38th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Montreal, QC, Canada, Oct. 2012, pp. 4148–4153.
[22] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 481–493, Apr. 2014.
[23] Z. Du, L. He, Y. Chen, Y. Xiao, P. Gao, and T. Wang, "Robot cloud: Bridging the power of robotics and cloud computing," *Future Gener. Comput. Syst.*, vol. 74, pp. 337–348, Sep. 2017.
[24] Z. Du, W. Yang, Y. Chen, X. Sun, X. Wang, and C. Xu, "Design of a robot cloud center," in *Proc. 10th Int. Symp. Auton. Decentralized Syst.*, Tokyo, Japan, Mar. 2011, pp. 269–275.
[25] G. Li, H. Wang, X. Ying, and J. Liu, "A proxy-based cloud infrastructure for home service robots," in *Proc. 27th Chin. Control Decis. Conf. (CCDC)*, Qingdao, China, May 2015, pp. 5718–5723.
[26] H. Chen, F. Y. Zhou, and T. Tian, "Design of SOA interface model in service robot cloud computing platform," *J. Shandong Univ.*, vol. 45, no. 4, pp. 31–39, 2015.
[27] F. Zhou, L. Yin, R. Song, T. Tian, and H. Chen, "A novel building and scheduling method of cloud platform services for robot," *Robot*, vol. 39, no. 1, pp. 89–98, 2017.
[28] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Netw. Syst. Design Implement.*, Canada, 2012, p. 2.
[29] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi, "Integrated and autonomic cloud resource scaling," in *Proc. IEEE Netw. Oper. Manage. Symp.*, Maui, HI, USA, Apr. 2012, pp. 1327–1334.
[30] S. M. T. Nezhad, M. Nazari, E. A. Gharavol, "A novel DoS and DDoS attacks detection algorithm using ARIMA time series model and chaotic system in computer networks," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 700–703, Apr. 2016.
[31] T. Goldschmidt, S. Hauck-Stattelmann, S. Malakuti, and S. Grüner, "Container-based architecture for flexible industrial control applications," *J. Syst. Archit.*, vol. 84, pp. 28–36, Mar. 2018.
[32] M. Tenorth, D. Nyga, and M. Beetz, "Understanding and executing instructions for everyday manipulation tasks from the World Wide Web," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, May 2010, pp. 1486–1491.
[33] G. Toffetti, S. Brunner, M. Blöchlinger, J. Spillner, and T. M. Bohnert, "Self-managing cloud-native applications: Design, implementation, and experience," *Future Gener. Comput. Syst.*, vol. 72, pp. 165–179, Jul. 2017.
[34] X. V. Wang, L. Wang, A. Mohammed, and M. Givehchi, "Ubiquitous manufacturing system based on Cloud: A robotics application," *Robot. Comput.-Integr. Manuf.*, vol. 45, pp. 116–125, Jun. 2017.
[35] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Appl. Comput. Informat.*, vol. 14, no. 1, pp. 1–16, 2018.
[36] C. Mouradian, S. Yangui, and R. H. Glitho, "Robots as-a-service in cloud computing: Search and rescue in large-scale disasters case study," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Las Vegas, NV, USA, vol. 1, Jan. 2018, pp. 1–7.

JIN LIU received the B.S. degree in measurement and control technology and instrument from the School of Mechanical Engineering, Qingdao University, Qingdao, China, in 2018. He is currently pursuing the M.S. degree with the School of Control Science and Engineering, Shandong University, Jinan, China. His research interests include cloud robot, cloud computing, and service scheduling algorithm.

FENGYU ZHOU was born in Shandong, China, in 1969. He received the Ph.D. degree in electrical engineering from Tianjin University, Tianjin, China, in 2008. He is currently a Professor with the School of Control Science and Engineering, Shandong University, Jinan, China.

LEI YIN received the M.S. degree in control engineering from the School of Control Science and Engineering, Shandong University, Jinan, China, in 2010, where he is currently pursuing the Ph.D. degree. His research interests include cloud robot, cloud computing, and control theory.

YUGANG WANG was born in Shandong, China, in 1988. He received the B.S. degree in mathematics and applied mathematics from the College of Sciences, China University of Mining and Technology, Xuzhou, China, in 2014. He is currently pursuing the Ph.D. degree with the School of Control Science and Engineering, Shandong University, Jinan, China. His research interests include iterative learning control and service robotics.