Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-5 (2020) 507–511

Design and Implement of Microservice System for Edge Computing

Peng Sha[1,2] Shichao Chen[1,3] Liling Zheng[4] Xudong Liu[5] Haotian Tang[5] Yeqian Li[5]

[1] *Institute of Automation, Chinese Academy of Sciences, Beijing 100190 China (e-mail: shichao.chen@ia.ac.cn)*

[2] *School of Automation, Southeast University, Nanjing 210096 China (e-mail: shapeng1998@foxmail.com)*

[3] *Faculty of Information Technology, Macau University of Science and Technology, Macau 999078 China (e-mail: shichao.chen@ia.ac.cn)*

[4] *The Cloud Computing Center, Chinese Academy of Sciences, Dongguan 523808 China (e-mail: zhengliling@casc.ac.cn)*

[5] *Faculty of Information and Technology, Beijing University of Technology, Beijing 100124 China (e-mail: hicx@bjut.edu.cn, 1455442383@qq.com, 2944662887@qq.com)*

Abstract: In the Internet of Things (IoT) scenario, the hardware resources in the edge computing paradigm are limited, so the applications in the cloud computing paradigm are not suitable for edge computing. New system architecture and key technologies are needed for application implement and deployment in edge computing. Based on the hardware performance constraints and scene functional requirements under edge computing, this paper designs and develops a microservice system for edge computing. So it introduces the design of microservice system architecture for edge computing. And it analyzes the key technologies of the system. It selects the PostgreSQL database to achieve data storage, the Flask framework to build back-end services, and the Vue.js framework to build front-end services. Meanwhile, docker containers are used for system deployment and testing. The test results prove that the system has perfect functions and stable operation, which is very suitable for edge devices with limited resources. According to the start-up time and resource consumption of the project, the distributed deployment of applications has better performance than the unified deployment.

Keywords: Edge computing, Microservice, Flask, PostgreSQL, Distributed deployment

1. INTRODUCTION

With the rapid development of the Internet of Things (IoT), smart devices can access the network very conveniently. It makes network devices rapidly increase and increases network data inevitably. Under this background, the cloud computing will cause network congestion, data leakage, and the heavy load of cloud storage resources due to data transmission and processing. It is difficult to meet the actual requirement for delay-sensitive applications such as autonomous driving, telemedicine, and the industrial Internet. The edge computing came into being in the context of IoT and forms a good paradigm complement to cloud computing (Shi et al., 2016a, b; Satyanarayanan, M. 2017). Edge computing can realize the real-time data processing on the edge side and reduce the transmission of redundant data to cloud computing centers, thereby further reducing the network bandwidth occupation and the computing and storage pressure of cloud computing center (Weisong et al., 2017).

Both edge computing and cloud computing can implement data processing, system monitoring and control. It can be said that edge computing is an extension of cloud computing on the edge network (Nasir et al., 2018). To achieve cloud computing functions on the edge, it needs to adapt the technologies and architectures that are suitable for edge resource characteristics. Cloud server clusters have powerful computing, storage, and other resources. There is no need to consider the limitations of hardware resources during system deployment and operation (Han et al., 2015; Villamizar, M. et al., 2015). Compared with cloud computing, the computing and storage resources of hardware on the edge are relatively limited (Varghese et al., 2016), so the applications and technical systems under the cloud computing paradigm are not suitable for edge computing. Therefore, for edge applications, they need to compatible system architectures and key technologies. The key technologies include a lightweight database, a lightweight web backend and web frontend, and a microservice system architecture that is easier for expanding and developing.

*This work was supported in part by the National Key Research and Development Program of China (No.2019YFB1704100), National Natural Science Foundation of China under Grants U1909204, 61773381, U1811463, 61773382 & 61872365.

10.1016/j.ifacol.2021.04.137

Program development based on the microservice architecture (Balalaie et al., 2016; Fernandez et al., 2019; Alshuqayran, N. et al., 2016; Hasselbring, W. et al., 2017) has been widely used. Compared with the macro-architecture, it has the advantages of function expansion, service decoupling, and high resource utilization (Liang et al.,2019). Meanwhile, the microservices can be allocated computing resources according to the demands of applications. Therefore, an application based on the microservice architecture is suitable for edge computing platforms.

System monitoring and management, configuration, and application under the traditional edge computing platform require professionals to operate the edge server or edge node on the scene. It greatly reduces the system's interactivity and rapid deployment for ordinary users. So we aim to build a lightweight microservice system for edge computing in an IoT scenario with limited resources. This system is used to realize the visual management of the resources on the edge and the convenient deployment of application services. It also accelerates the function expansion and business management of the edge computing system.

For achieving convenient development of edge system applications and intelligent management of resources under the edge computing paradigm, we firstly study the business architecture and functions of the microservice system on the edge computing. Secondly, according to the characteristics of the limited hardware resources on the edge, it analyses the key technologies in the process of application development. Finally, this microservice system for the edge computing platform is deployed and tested on the Raspberry Pi 3B+. The test results show that the designed system is reasonable and provides a feasible solution for the visual dynamic management and control for edge computing devices.

2. SYSTEM ARCHITECTURE AND FUNCTIONS

2.1 Business Scenario Architecture

The architecture of the edge computing system mainly includes three layers, field layer, edge computing layer, and cloud platform layer. The microservice system designed and implemented is mainly applied on the edge side, so the business scenarios are also mainly for edge devices. In this scenario, it mainly includes edge servers and edge nodes. An edge server may be composed of one or more servers, and an edge server connects to multiple edge nodes. At the same time, each edge node can connect to multiple terminals which can be sensors or actuators. Fig. 1 is a business scenario architecture.

In this architecture, an edge server and edge node have computing and storage resources, but the hardware resources and performance of an edge server are stronger than those of an edge node. The microservice system for the edge computing platform has a series of function modules, i.e., real-time monitoring, computing task management, algorithm model management, device management, and user management for all devices on the edge. It can be deployed on an edge server or an edge node to implement edge system management and form a closed-loop system to improve the robustness of an entire cloud-edge system.

2.2 System Functions

According to the analysis of business scenarios, we design a microservice system, which mainly includes some major function modules to achieve the management, control, and monitoring of an edge system. It is shown in Fig. 2.

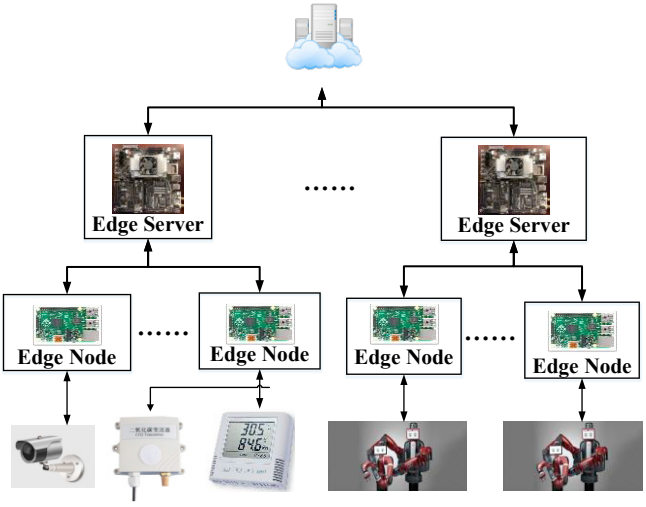


Fig. 1. Business scenario architecture.

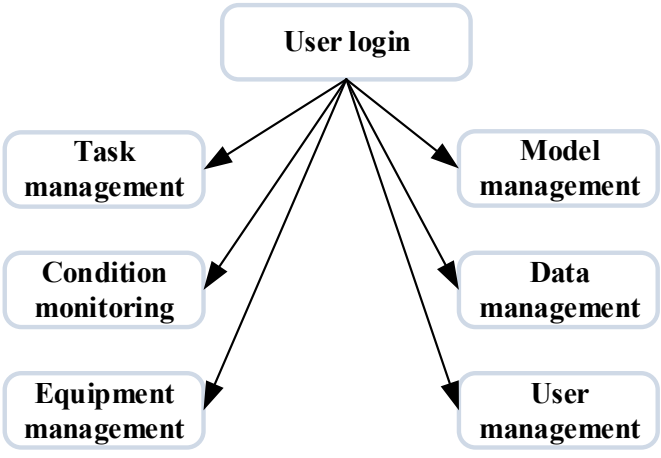


Fig. 2. Microservice system functions.

- Home page

Home page is an overview interface for the entire microservice system. Users can have an intuitive understanding for the running status of the entire microservice system through it. Many alarm information can be discovered in real-time. It is divided into three areas, including equipment statistical analysis, task execution statistical analysis, and equipment monitoring analysis area. They are displayed with a graphical interface to facilitate more intuitive human-machine interaction.

- Condition monitoring

Condition monitoring module mainly realizes real-time monitoring about the working condition of edge hardware, including CPU and memory usage of edge servers and edge nodes, and alarms. Meanwhile, it also realizes real-time monitoring and early warning of the sensing data (temperature, pressure, etc.) in the monitoring area.

- Device management

Device management module mainly implements the management of edge servers, edge nodes, and terminals. Its functions include the addition, deletion, search, and modification of devices.

- Model management

Model management module mainly realizes the effective management of intelligent models (task scheduling model, target detection model, etc.) on the edge. Its functions include the addition, deletion, and modification of models.

- Task management

Task management module mainly realizes the management of computing tasks on the edge, and its functions include the addition, deletion, search of the computing tasks. In addition, it can modify the working status of computing tasks. The system can manually add corresponding computing tasks to the appropriate edge devices for calculation.

- Data management

Data management module mainly manages the monitoring data on the edge. Its functions include the statistics, storage, and download of monitoring data.

- User management

User management mainly implements the hierarchical management of users. It includes a super administrator and some ordinary administrators. Super administrators can manage ordinary administrators, including the addition, deletion, and modification of ordinary administrators, and set the management rights of ordinary administrators.

3. KEY TECHNOLOGIES

In an IoT scenario, according to the actual function requirements, both the overall implement and operating efficiency of the system must be considered in the system design and implement (Truong et al., 2020). Meanwhile, the performance limitations of hardware resources must be taken into account. A lightweight database, web backend, and frontend framework are needed to achieve this system. Therefore, we choose the lightweight database PostgreSQL to realize the data storage, a Flask framework to build the edge business logic for backend service, and a Vue.js framework to build the frontend page for realizing data visualization. Besides, we also use the RESTful API to be responsible for front-end and back-end communication and docker containers to achieve distributed deployment of applications, which greatly improve development and deployment efficiency. The key technologies stack of the system is shown in Fig. 3.

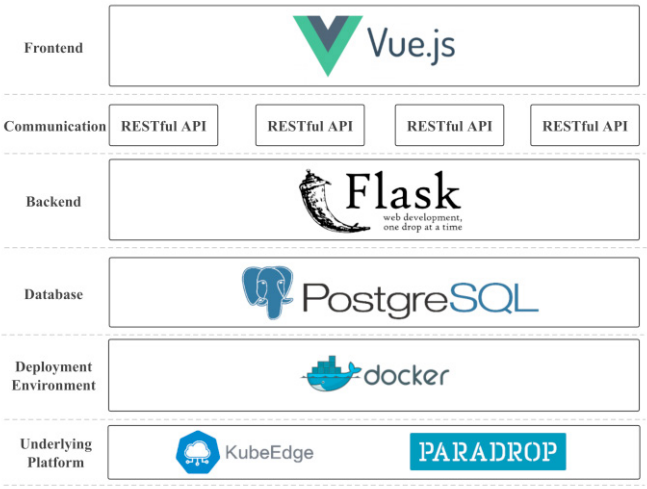


Fig. 3. Key technologies stack of system

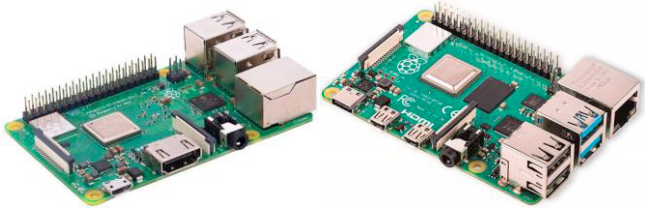
4. SYSTEM DEPLOYMENT AND TESTING

4.1 System Deployment

We use a docker container to deploy the system. This system is developed on a PC with an X86 architecture, but it is deployed on a Raspberry Pi with ARM architecture. So Docker Buildx is required to build docker images supported by a multi-hardware platform architecture. It is a Docker CLI plug-in, and its principle to build a multi-system architecture image is using a different builder from Docker to achieve the construction of a multi-system architecture image. We make docker images of the database, Flask backend, and Vue.js frontend respectively. In the commands to build an image, "--platform" is used to specify the CPU architecture supported by the built image. We specify the three architectures of Linux/ARM, Linux/ARM64, and Linux/AMD64 to adapt to the diversity of heterogeneous platforms in edge computing platforms.

This system is deployed in Raspberry Pi series hardware to evaluate the performance of common hardware in edge computing. There are two deployment methods for this system: unified deployment and distributed deployment. The unified deployment of the system refers to the unified deployment of all engineering applications, including the system database, web backend, and frontend to the same hardware platform. We use a Raspberry Pi 3B+, which runs the Raspbian Buster Lite operating system. The distributed deployment of the system refers to the separate deployment of the database and web front-end engineering applications. We use two Raspberry Pi devices, and one is Raspberry Pi 3B+ and another is Raspberry Pi 4B (running Ubuntu 20.04 LTS operating system). The database and web backend that read data more frequently are deployed to the more powerful Raspberry Pi 4B, and we deploy the web front to a Raspberry Pi 3B+. The difference between distributed deployment and unified deployment is that the front-end and back-end are

deployed on different machines. We adopt the development method of separating frontend and backend, so it only needs to specify the back-end API address on the front-end during deployment without worrying about the communication between the frontend and backend. Raspberry Pi 3B+ and 4B are shown in Fig. 4.



(a) 3B+ (b) 4B

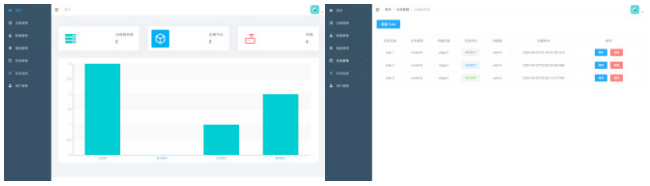
Fig. 4. Raspberry Pi

4.2 System Test

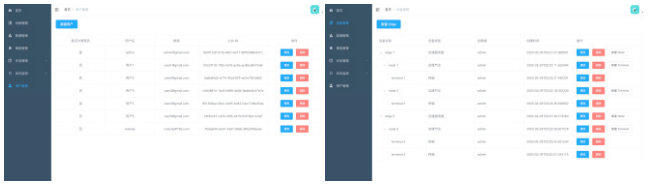
The system test mainly includes two parts: a system function test and a system performance test. The two parts will be described in detail below.

- System function test

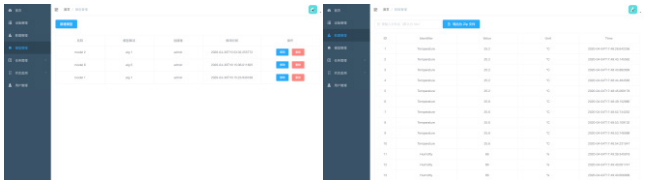
The system mainly includes homepage, condition monitoring, device management, model management, task management, data management, user management modules. They are shown in Fig. 5.



(a) Home page (b) Task management



(c) User management (d) Device management



(e) Model management (f) Data management

Fig. 5. Microservice system interface.

- System performance test

The performance test of the microservice system is applied to a Raspberry Pi 3B+ and 4B, and the relevant parameters of the hardware are shown in Table 1 and Table 2. We do many performance evaluation, i.e., start-up time and system resource consumption, in the two deployment methods.

After several tests, the comparison results of unified deployment and distributed deployment are shown in Fig. 6. As can be seen from Fig. 6 that during unified deployment, Raspberry Pi 3B+ takes 22.84s to start the entire project. During distributed deployment, Raspberry Pi 4B takes over the database and backend deployment, so Raspberry Pi 3B+ takes only 6.478s to complete the start-up of the project. Therefore, the distributed deployment has brought a huge increase in project start-up speed for the Raspberry Pi 3B+.

Table 1. Raspberry Pi 3B+

SOC	Broadcom BCM2837B0
CPU	64-bit 1.4GHz quad-core
Wired network	330Mbps Ethernet
Operating system	Raspbian Buster Lite
Memory	1G
Storage space	32G

Table 2. Raspberry Pi 4B

SOC	Broadcom BCM2711B0
CPU	64-bit 1.5GHz quad-core
Wired network	Gigabit Ethernet
Operating system	Ubuntu 20.04 LTS
Memory	4G
Storage space	32G

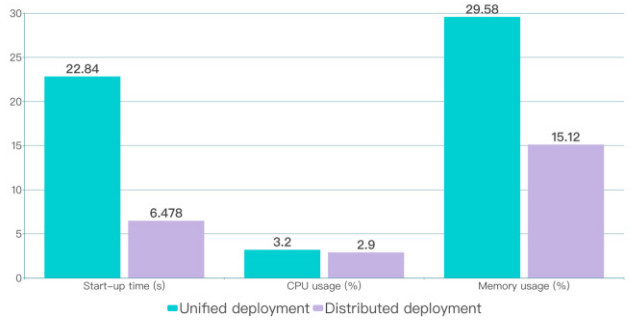


Fig. 6. Performance comparison of different deployment methods.

Meanwhile, the resources occupied by unified deployment and distributed deployment in Raspberry Pi 3B+ are not much different in terms of CPU consumption. While there is a gap of about 140M in memory usage between the distribution deployment and unified deployment, and the impact of this 100M gap on edge computing hardware with limited resources is also great. It can be seen that the distributed deployment of the system for Raspberry Pi 3B+ can reduce resource consumption during system operation.

Therefore, it can be seen from the performance test that distributed deployment using Docker is more conducive to taking advantage of multi-platform comparing to unified deployment. Distributed deployment brings a certain degree of improvement to edge devices. And it relieves the working pressure of edge devices. Of course, the use of distributed deployment will also increase the energy consumption of the entire system, so the distributed and unified deployment should be determined according to the actual situation.

5. CONCLUSIONS

In an IoT scenario, we firstly study an architecture of microservice system for edge computing, including business scenario architecture and system functions. Secondly, according to the resource constraints of edge hardware and system function requirements, key technologies such as a lightweight database, web backend, and web frontend are selected to develop the system in combination with the microservice. Finally, we use docker containers to deploy application images on Raspberry Pi, and perform function and performance tests of the system. The test results prove that the system application has perfect functions, stable and reliable operation. It is suitable for deployment on edge devices. Besides, the performance test results show that compared with the unified deployment, the distributed deployment has better performance in terms of application start-up time and resource consumption. And it can give full play to the advantages of the system.

REFERENCES

Alshuqayran, N., Ali, N., & Evans, R. (2016). A Systematic Mapping Study in Microservice Architecture. service-oriented computing and applications.

Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. IEEE Software, 33(3), 42-52.

Fernandez, J. M., Vidal, I., & Valera, F. (2019). Enabling the Orchestration of IoT Slices through Edge and Cloud Microservice Platforms. Sensors, 19(13).

Han Z., Miao P., Xinxin L., Xiaolin L. Yuguang F.(2015). Exploring Fine-Grained Resource Rental Planning in Cloud Computing. *IEEE Transactions on Cloud Computing,*3(3),304-317.

Hasselbring, W., & Steinacker, G. (2017). Microservice Architectures for Scalability, Agility and Reliability in E-Commerce. ieee international conference on software architecture workshops.

Liang B., Chase W., Xiaoxuan B., Nana R., Mengqing S. (2019). Performance Modeling and Workflow Scheduling of Microservice-Based Application in Clouds. *IEEE Transactions on Parallel and Distributed Systems,* 30(9), 2101-2116.

Nasir A., Yan Z, Amir T., Tor S (2018). Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal,*5(1),450-465.

Satyanarayanan, M. (2017). The Emergence of Edge Computing. IEEE Computer, 50(1), 30-39.

Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016a). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal,* 3(5), 637-646.

Shi, W., Dustdar, S. (2016b). The Promise of Edge Computing. *IEEE Computer,* 49(5), 78-81.

Truong, H., and Klein, P. (2020). DevOps Contract for Assuring Execution of IoT Microservices in the Edge. the internet of things.

Varghese B., Wang, N., Barbhuiya S., Kilpatrick P., Nikolopoulos D.S.(2016). Challenges and Opportunities in Edge Computing. *IEEE International Conference on Smart Cloud,* Volume 1,20-26. IEEE.

Weisong, S., Hui, S., Jie, C., Quan, Z., and Wei, L. (2017). Edge Computing—An Emerging Computing Model for the Internet of Everything Era. *Journal of Computer Research and Development,* 54(5), 907-924.

Xu, R., Jin, W., and Kim, D. (2019). Microservice Security Agent Based On API Gateway in Edge Computing. Sensors, 19(22).

Villamizar, M., Garces, O., Castro, H., Verano, M., Salamanca, L., Casallas, R., & Gil, S. (2015). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. computing colombian conference.