Application of Microservice Architecture in Commodity ERP Financial System

Wan Yan and Fu Shuai

***Abstract*—At present, bulk commodity enterprises mostly use single frame structure to develop ERP system, in which the function of financial system is relatively simple and the division boundary is fuzzy. With the deepening of business complexity and the expansion of data volume, the single architecture financial system has exposed some disadvantages, such as high concurrency, weak carrying capacity, insufficient security, low fault tolerance, poor scalability. In order to solve these problems, we use micro service architecture to design and develop commodity ERP financial system. Based on the design concept of micro service architecture, this paper splits the complex system business, selects the spring cloud framework to reconstruct the original single system architecture, reduces the coupling of the system, and realizes the technologies of service management, routing gateway, load balancing, fault-tolerant processing, security protection and so on. The system not only meets the complex business requirements, but also has high availability, high security and high scalability. Through the practical application in a multinational trading company, the practicability and efficiency of the system are proved.**

***Index Terms*—Bulk commodities, microservice architecture, financial system, spring cloud.**

I. Introduction

Bulk commodities refer to basic raw material commodities that can be bought and sold in large quantities in the circulation field. It is characterized by large quantity, large transaction amount, long transportation process and many stakeholders involved, including production, transaction, transportation, testing, reception and other links [1]. At present, informatization has become the development trend of globalization. ERP system integrates the enterprise's finance, procurement, production, sales, inventory and other business functions into an information management platform to improve the enterprise's information management level.

However, ERP has not been popularized in the field of bulk commodities in China. The main reason is that ERP has no universal fixed model, and some ERP systems can not meet the actual needs of commodity enterprises. For example, most of the purchases and sales of bulk commodity enterprises are carried out with non cash, resulting in most of the financial data being dynamic. At the same time, the trading chain of bulk commodities from purchase to sale is too long, involving multiple interests and multiple links [2], and the system construction is more complex. In addition, with the expansion of production and the complexity of business, the business needs of bulk commodity enterprises are constantly adjusting. There are a wide variety of businesses and a large amount of data. Each business information is independent of each other, but all affect the financial changes. The traditional single architecture financial management system has been difficult to meet the business needs of enterprises. It can not meet the new requirements in concurrency, fault tolerance, security and other aspects, and the scalability is weak. Although SOA architecture can alleviate the disadvantages of single architecture to a certain extent, it still has some disadvantages, such as high service communication cost and dependence on enterprise service bus [3].

At present, scholars at home and abroad attach great importance to the research of micro service architecture and its implementation mechanism. With the increasing complexity of software system functions, microservice architecture began to rise day by day. From the generation of microservices to the gradual application of software system platform, microservices technology is developing and improving [4]. In industrial applications, a complete micro service architecture is composed of multiple elements, which has an independent ecosystem. At present, the micro service framework has been successfully practiced and developed by more and more large and medium-sized enterprises, typically Amazon, Uber, Netflix and other companies, but it is not widely used in the field of bulk commodities.

Aiming at the problems of single architecture and SOA architecture financial system, this paper designs and develops a commodity ERP financial system based on micro service architecture. Through the concept of decoupling and splitting, the complex financial business is decomposed into different micro services. Each micro service is deployed and worked independently to facilitate management and maintenance [5]. The functions of service management, gateway routing, concurrency control and security protection are realized through related technologies. Compared with the traditional single architecture, the financial system has the following advantages: high development and deployment efficiency, high reliability, high security and strong maintenance and expansion ability.

II. System Requirements Analysis

A. Business Demand Analysis

Financial management system is in the core position in any commodity enterprise. It not only plays a role in its own department, but also has data interfaces in other departments, which directly affects the management efficiency of other departments [6]. The system analyzes the business needs of a large multinational company engaged in log board trade. It

Manuscript received April 27, 2022; revised June 21, 2022.

Wan Yan and Fu Shuai are with the College of Computer Science and Technology, Donghua University, Shanghai, CO 201620 China (e-mail: winniewan@dhu.edu.cn, 837893285@qq.com).

DOI: 10.7763/IJCTE.2022.V14.1324

has a large business responsibility, large business volume, wide coverage area and large number of users. The overall business requirements architecture is shown in Fig. 1, which mainly introduces the key business requirements:

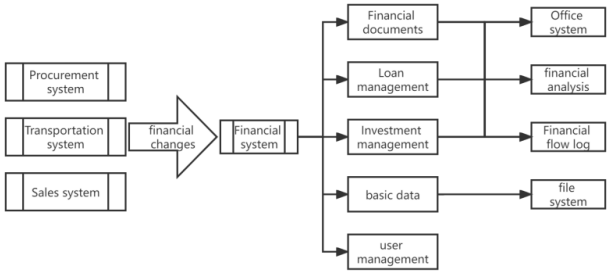

Fig. 1. Business requirements architecture.

1) Financial documents

This module involves all expenses in commodity ERP companies, which is mainly divided into two parts: expenditure document and income document. In view of the huge business volume of financial documents and the inefficiency and error prone of the traditional human work order method, all financial documents in the system will be automatically generated when there are changes in expenses, income and expenditure. In addition, due to the characteristics of dynamic settlement of bulk commodity transactions, the newly generated financial documents only generate payment receivable plan, and the corresponding payment receivable plans are generated after approval. The status before finance will be returned after anti approval.

2) Financial analysis

A mature financial system is inseparable from financial analysis. This module is mainly responsible for analyzing the profit and loss of the company. The financial analysis module of the single architecture is intricate and has a high degree of code coupling. Therefore, it is necessary to divide this module into independent units through the decoupling and splitting idea of microservices, and integrate and analyze the financial information of other modules. And provide visual data chart page to clearly reflect the financial situation of the company, and give corresponding purchase and sales opinions and future investment opinions of the company.

3) User management

As the core part of enterprise business, system user management is also very important. The user module includes user management, department management, role management and permission management. This module needs to set up a strict role permission system to ensure the privacy and security of the company's data.

B. Business Demand Analysis

1) Security requirements analysis

As a large multinational company with many employees and businesses all over the country, it has extremely strict requirements for the security and data privacy of the financial system. Traditional security authentication methods can not meet the strict security requirements, so it is necessary to build an independent security service to provide strict authentication, authorization and access control functions to avoid malicious damage to the system. At the same time, strictly control the authority of data and interface to ensure the privacy of data and user information.

2) Performance requirement analysis

Due to the huge business volume of the company, high concurrency will occur frequently. It is necessary for the system to shorten the request response time while meeting the needs of users, and when a module fails, it should not affect the normal use of other modules.

3) Other requirements analysis

The commodity sector will constantly adjust its business needs according to relevant policies. Therefore, the system needs to have high scalability and scalability to improve the efficiency of maintenance and update.

III. Microservice Architecture Application

The traditional enterprise financial system simply transplants the accounting information recorded manually to the computer and uses the software system to replace the original manual labor [7], ignoring the importance of system performance, safety and maintenance. Bulk commodity finance has the characteristics of large transaction amount, long transaction chain, many businesses and transaction units, large business volume, dynamic data changes, and data security is very important. In addition, with the expansion of production and the deepening of complexity, the traditional financial management system has been difficult to meet the business needs of enterprises. At present, most enterprises still use the ERP management system with a single architecture, in which the function of the financial system is relatively simple and rigid, and the maintenance and expansion ability is weak, which can not meet the more complex business needs and performance requirements. The microservice architecture can solve the problems existing in the ERP financial system with single architecture, and has higher applicability and scalability [8].

A. Architecture Technology Selection

Before the emergence of microservices, the website service architecture was usually one-stop, and a service application contained multiple modules and interfaces. The traditional development mode unifies the design of all functional interfaces and develops and realizes them layer by layer [9]. If a module appears anywhere, it will affect the use of the whole system.

SOA is a service-oriented architecture idea, which decomposes complex application systems into multiple independent business units, and realizes service integration by defining good interfaces between services, so as to achieve the overall function of the system [10]. On this basis, the microservice architecture is improved, integrated with the idea of componentization, and the business is completely serviced and componentized in the way of low coupling separation. Each microservice is deployed independently and responsible for the corresponding tasks of its own module [11]. The design of microservice application architecture is shown in Fig. 2. Compared with the traditional single architecture mode, the advantages of microservice architecture are obvious.

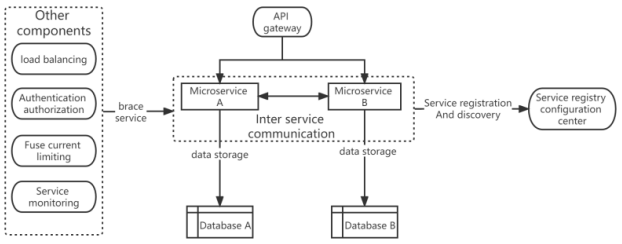

Fig. 2. Microservice architecture.

1) High availability

Microservices split the system through low coupling, and split the huge system into different microservices according to functions. Each microservice is responsible for independent business functions. Each microservice can be deployed and run independently, which brings high efficiency to the development, testing, deployment and maintenance stages [12]. And through some components, it can deal with concurrency and security problems, and has higher availability and security.

2) High fault tolerance

A small error in a module of a stand-alone architecture may lead to the collapse of the whole system. Microservices are independent of each other. The error of one microservice will not affect the use of other microservices, and the fault tolerance is higher. The data are also stored in a distributed way. Some database problems will not affect the data of other micro services to ensure data security.

3) High maintainability

The code coupling of microservice architecture is low, and the system can be flexibly adjusted according to the changes of business requirements. Deployment also does not require the redeployment of the whole system, which can save the cost of building and deploying projects [13]. Moreover, the micro service architecture is easier to update and upgrade, which is conducive to continuous integration.

4) High availability

The development efficiency of traditional monomer architecture is low. The code is written in the same project, and the conflict between developers when submitting code is serious. And the coupling degree between modules is high. When updating and upgrading, large-scale change projects are often required [14]. Because the microservice architecture is only related to API, the development efficiency is higher. At the same time, different microservices can be developed in different languages and development frameworks, which greatly increases the degree of freedom of development.

B. Architecture Technology Determination

According to the characteristics and specific needs of commodity enterprises, the overall architecture of the system is built based on spring cloud. Spring cloud is a micro service framework based on spring boot. It is an orderly collection of a series of frameworks and components. It has fully functional and lightweight micro service components, such as service governance, service gateway, intelligent routing, load balancing, circuit breaker mechanism, and other proven components [15].

C. Microservice Splitting

According to the basic idea of micro service architecture and combined with specific business requirements, the ERP financial system is finely divided into system services and business services. The system service is responsible for the configuration and communication of micro services to maintain the efficient operation of the system. The business service is responsible for the specific business logic functions of the financial system. Each microservice is only responsible for independent business and corresponding task processing, exposes only independent service interfaces, and returns the results to the outside or inside in the form of API. The specific division of micro services is shown in Table I.

TABLE I: MICROSERVICE SPLITTING

Service category	Service name	service function
system service	Service Management Center	Service registration and discovery
	Gateway routing center	Authentication, routing forwarding and flow control
	Authorized authentication service	Authorized service provision
	Security microservice	Safety tools
	Distributed transaction service	Handling distributed transactions
Business services	Document micro service	Automatic document generation
	Loan micro service	Deal with loan business
	Investment micro service	Deal with investment business
	Financial analysis micro service	Data statistical analysis
	Data microservice	Master data settings
	Office microservices	User office business
	Log microservice	Financial log record
	File microservice	Storing files and printing
	User micro service	User rights management

IV. DESIGN OF FINANCIAL SYSTEM BASED ON MICRO SERVICE ARCHITECTURE

A. Overall Architecture

In view of the complexity of the business of commodity companies, the complex business is decoupled and divided based on the architecture design concept of microservice. Spring cloud distributed framework is used to build a distributed system background with high concurrency, high availability, high performance and high security. Nacos configuration center is used to uniformly manage platform service configuration and realize service registration and discovery. Spring cloud gateway acts as a gateway component to uniformly expose APIs. The spring cloud loadbalancer controls load balancing to achieve high availability and high concurrency. Sentinel is responsible for monitoring the calls between services, and fuse protection for continuous failures. The overall architecture is shown in Fig. 3, which is mainly divided into the following four modules.

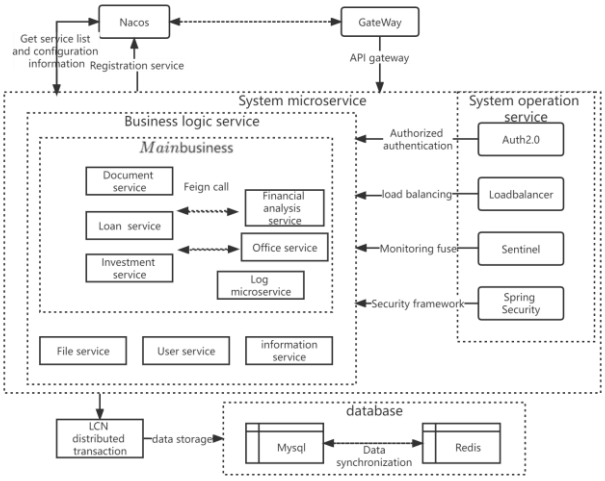

Fig. 3. Overall system architecture.

1) Service management

Nacos is selected as the service registration and discovery center of the whole micro service to discover, configure and manage micro services. Compared with Eureka, Nacos has its own configuration center, provides more technical support, and has a service management interface, which is more convenient for service management. The registration center is divided into server and client. The client is used to register its own services, obtain service list and configuration information, and maintain heartbeat. Server is used to manage registration services, provide service list, service governance and other functions [16]. The configuration center manages the configuration information of all applications or services in a centralized and dynamic way through dynamic configuration services, without redeploying services, which greatly increases the operation and maintenance capacity of the system.

2) Gateway routing

After a single application is split into multiple services, a unified entrance is needed externally to decouple the client and internal services. Using spring cloud gateway as the gateway to replace Netflix and zuul, the performance is higher than zuul, and has some advanced functions of the gateway. It uses reverse proxy to provide a unified routing method, completes authentication, flow control and other functions by defining filters and assertions, and realizes dynamic routing forwarding, authentication, protocol conversion and other functions by calling filters. At the same time, it also integrates the circuit breaker function of hystrix to fuse, and supports path rewriting [17].

3) Business logic module

Decompose the complex financial business into multiple independent and expandable micro services. The financial document service provides an interface for automatically generating financial documents. Financial analysis micro service integrates and analyzes all financial data. The log service records all changes in finance. The user management microservice is responsible for user related information processing and fine-grained permission control.High maintainability. The code coupling of microservice architecture is low, and the system can be flexibly adjusted according to the changes of business requirements. Deployment also does not require the redeployment of the whole system, which can save the cost of building and deploying projects. Moreover, the micro service architecture is easier to update and upgrade, which is conducive to continuous integration.

4) Database module

Data storage adopts the combination of MySQL and redis, and the data of different micro services are stored in different databases. Redis caches the data in memory and periodically stores the data to disk through the data persistence mechanism [18]. Redis is used as intermediate storage to improve the storage speed and query speed, and greatly improve the system efficiency. The system uses Druid as a high-performance database connection pool, and realizes the functions of data source monitoring, web application monitoring, URI monitoring, slow query and so on through the built-in monitoring tools.

B. High Availability Implementation

The bulk commodity financial system has large business flow, complex business relationship, long transaction chain and a large number of high concurrency scenarios. The system realizes high availability through concurrency control, gateway current limiting, fuse degradation and other means.

1) Concurrency control

The system abandons its dependence on Netflix and adopts spring cloud loadbalancer as the load balancer. By adding corresponding interceptors to the client tool class, load balancing is completed in the interceptors. First, set the interceptor and inject it into the client accessing the rest service through automatic configuration class. The interceptor is responsible for the logical processing of load balancing. It uses polling as the load balancing algorithm, selects a service address from the service list to call, and realizes the load balancing call of the client.

2) Gateway current limiting

In order to prevent server overload and service paralysis caused by excessive requests, flow control needs to be done in the system. Write a filter in the spring cloud gateway to implement current limiting. In order to deal with a large number of sudden requests, a more flexible token bucket algorithm is used to limit the flow of some service interfaces with overload risk by configuring parameters such as the total capacity and filling rate of the token bucket.

3) Fuse degradation

Due to network or other reasons, micro service calls may be blocked and then cause avalanche effect. To solve this problem, the system integrates sentinel as a microservice link protection mechanism to improve the stability of the service.The thread pool isolation commonly used by hystrix will cause a lot of losses, and the semaphore isolation used at the same time cannot automatically degrade slow calls [19]. Sentinel performs semaphore isolation through the flow control of the number of concurrent threads. In addition, sentinel supports more fuse degradation dimensions, and multiple indicators can be fused by configuring personalized degradation logic. In case of call timeout or unstable increase in the proportion of exceptions, limit the call to this service to avoid affecting other services and causing cascading errors.

4) High availability of service tier and storage tier

The service layer is stateless, and the nodes in the cluster can completely replace each other. The downtime of any node will not cause the system to stop service. It greatly improves the high availability of the service layer. Multiple nodes in the storage layer share data storage, and the downtime of any node will not affect the whole database. Synchronize data between multiple copies to ensure strong consistency of data. When a single copy fails, it will not affect the reliability of data. Perform master-slave replication, and switch to the standby after the host goes down.

C. Extensibility Implementation

The demand of commodity business is extremely complex and will be adjusted and updated frequently. If there is no reasonable structure to write code, the later maintenance and update efficiency of the system will be very low. The system manages module dependency through maven, and manages the code in modules according to the system architecture design. Clear boundaries are set between services, which is easy to expand and deploy on demand. Specifically, the maintainability and scalability of the code are enhanced by defining stable interfaces, using design patterns reasonably, and modularizing general modules.

V. Security Framework Implementation

As the core business of enterprises, financial security is very important. At present, many security modules and rights management have problems such as too high coupling with the core business of the project and poor reusability, which can not be well applied to the microservice Architecture [20]. In order to effectively deal with malicious network attacks and avoid illegal intrusion and unauthorized operation, the system adopts spring Security + auth2 0, as a security solution, takes the functions of authentication, authorized access and attack protection as an independent microservice to realize security protection.

A. Spring Security

Spring security includes three core components: security aspect, authentication manager and access decision Manager [21]. Its core is a set of filter chains. The filter contained in the security aspect intercepts the URL request, extracts the authentication information from it and gives it to the subsequent authentication manager and access decision manager for authorization processing. Spring security supports a variety of mechanisms to verify the identity of the login. The generated token is stored in redis. The front-end returns the front-end menu that the user can access and dynamically loads the page, so as to display different page effects for different roles.

B. OAuth2.0

OAuth2.0, as an open standard third-party authorization agreement, divides microservices into authorization services and resource services [22]. The authorization service is responsible for the functions of user authority management and identity authentication. Resource services are responsible for specific business functions. Multiple resource services share an authorization service. The message is symmetrically encrypted during login, and the filter of the gateway module verifies the requested verification code and decrypts the password, and then routes and forwards [23]. OAuth2. 0 checks the user information and generates a token by querying the user information in the database. When the user carries a token to request resources from the resource server, oauth2 0 intercepts tokens for checking, processing and querying user information, and converts stateless tokens into user information. And support high-performance mode through expansion to solve oauth2 0 the performance bottleneck of the whole process to provide business support for high-performance scenarios.

C. Permission Setting

Large enterprises have a large number of employees, and the leakage of data privacy will have a serious impact on enterprises. Role-based access control method is an effective method to solve the resource access control of large enterprises. Using this method to set strict permission control for data and users can reduce the complexity of authorization management, reduce management overhead, and have good scalability.

Permission design is divided into function permission and Data permission. The function permission is injected with the corresponding permission ID in the life cycle by assigning the permission flag to the menu. Then, by obtaining the user menu list and comparing with the requested address and request method, judge whether there is permission to decide whether to render the corresponding elements. Data permission granularity is divided by department and role. According to the principle of mybatis interceptor, add {corresponding objects in the query parameters, intercept the mapper method containing the processing parameters, query the data permission configuration of the role of the current user, automatically match the filtering rules according to the current user role and splice them into a new SQL to complete the data permission filtering function.

VI. Conclusion

By analyzing the defects and actual needs of ERP financial system with traditional single architecture, this paper expounds the advantages of micro service architecture, decouples and splits the complex bulk commodity financial business based on the idea of micro service architecture, and builds a financial system with high concurrency, high availability and high security. System services are responsible for service registration and discovery, routing gateway, load balancing, security protection and fault-tolerant processing mechanism, and business services are responsible for specific financial requirements and functions. The coupling between micro services is low, which has the advantages of flexibility, scalability and decentralization [24]. The system solves the problems existing in the traditional single architecture system, and provides a reference for the financial information management of enterprises and the development of ERP financial system in different fields.

With the continuous innovation of micro service related technologies and the increasing complexity of business needs in the field of bulk commodities, how to accurately

communicate between micro services, how to better monitor and manage services [25], how to deal with a large number of business requests faster, and how to better combine micro services with ERP are worthy of our exploration and improvement in the future.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Fu shuai had carried out architecture design, project development and wrote the paper.Wan Yan conducted demand research, project development and reviewed the paper.

Acknowledgment

Wan Yan and Fu Shuai would like to thank the laboratory of the school of computer science of Donghua University for the experimental conditions and machinery and equipment, the business documents provided by Shanghai bulk commodity enterprises, each teacher of the school for his help and puzzle solving, the students in the laboratory for their support and help, and everyone who helped us.

References

[1] C. Huang, "Research on international logistics cost management of bulk commodities of Honghai group," M.A. dissertation, Guangxi Univ., Guangxi, China, 2017.
[2] B. Lu, "Why is it difficult for China's bulk commodity enterprises to go on ERP," *Fortune Today*, vol. 3, pp. 27, 2017.
[3] X. Ling, "SOA overview," *Computer Applications and Software*, vol. 10, pp. 122-199, 2007.
[4] Y. Xin, J. Niu, Z. Xie, K. Zang, and X. Mao, "Overview of microservice architecture implementation framework," *Computer Engineering and Application*, vol. 54, pp. 10-17, 1993.
[5] A. Sill, "The design and architecture of microservices," *IEEE Cloud Computing*, vol. 3, pp. 76-80, 2016.
[6] L. Gui, "Research on the application of ERP system in the financial management of G company," M.A. dissertation, Anhui Univ., Anhui, China, 2016.
[7] Y. Zhang, "Discussion on the application of ERP system to help enterprise financial internal control management," *Accounting Study*, vol. 14, pp. 185-186, 2021.
[8] J. Soldani, D. A. Tamburri, and W. V. D. Heyvel, "The pains and gains of microservices: A systematic grey literature review," *The Journal of Systems & Software*, 2018.
[9] Z. Li, "Development and impact analysis of microservice Architecture," *Computer System Application*, vol. 26, pp. 82-86, 2017.
[10] J. Zhang, Y. Wang, and X. Huang, "Implementation of a microservice framework," *Information System Engineering*, vol. 1, pp. 154-155, 2017.
[11] C. Li, "Overview of microservice architecture research," *Software Guide*, vol. 18, pp. 1-7, 2019.
[12] C. Chen, "Analysis of container based microservice architecture," *Information System Engineering*, vol. 3, pp. 95-96, 2016.
[13] J. Deng and C. Cao, "Research on some key issues of microservice," *Journal of Wuyi University*, vol. 30, pp. 49-54, 2016.
[14] W. Wei, "System design and development based on micro service architecture," *Computer Products and Circulation*, vol. 4, p. 169, 2015.
[15] J. Thones, "Microservices," *Software IEEE*, vol. 32, p. 116, 2018.
[16] F. Wang, "Design and implementation of microservicing of business system based on spring cloud," *Electronic Technology and Software Engineering*, vol. 8, pp. 60-61, 2018.
[17] G. Li, "A load balancing strategy under microservice architecture," *Information and Communication*, vol. 8, pp. 84-85, 2019.
[18] J. Ren, "Design and implementation of Redis based unified monitoring platform," M.A. dissertation, University of Chinese Academy of Sciences., Beijing, China, 2016.
[19] Z. Xiao, "Design and implementation of microservice communication framework," M.A. dissertation, Beijing Jiaotong Univ., Beijing, China, 2017.
[20] W. Tang, "Design and implementation of SOA architecture oriented microservice security system," M.A. dissertation, Nanjing Univ., Nanjing, China, 2016.
[21] S. Xiao, Y. He, J. Tian, and D. Hou, "Dynamic rights management system based on JWT + spring security," *Information and Computer*, vol. 33, pp. 131-134, 2021.
[22] D. Recordon, D. Hardt, and E. Hammerlahav, "TheOAuth2.0 authorization protocol," *International Journal of Pharmaceutics*, vol. 271, pp. 1-2, 2012.
[23] Y Liu, "Based on spring and oauth2.0's third-party authorization framework," *Computer Technology and Development*, vol. 27, pp. 167-170, 2017.
[24] Z. Feng, Y. Xu, X. Xue, and S. Chen, "Current situation and prospect of microservice technology development," *Computer Research and Development*, vol. 57, pp. 1103-1122, 2020.
[25] P. Jamshidi, C. Pahl, N. C. Mendonca, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software*, vol. 3, 2018.

Copyright © 2022 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).

Wan Yan was born in China, 1970. Her final graduate school is Shanghai Jiaotong University. She conducted postdoctoral research at the University of Texas at Austin from September 2007 to September 2008. Now she is a professor at Donghua University. Her research fields are image processing, automatic fiber recognition and 3D human body scanning.

From January 2004 to December 2008, she was responsible for the research on the automatic recognition system of profiled fibers with the special fund for the authors of National Excellent Doctoral Dissertations in Colleges and universities of the Ministry of Education (Research on computer image automatic analysis and detection technology of fibers and textiles). From May 2006 to March 2007, she was the project leader of "software development and operation maintenance of ticket checking machine of Shanghai Rail Transit Line". She published many papers, such as Wan, Y., Yao, L., Xu, B., Zeng, P., Shaped Fiber Identification Using A Distance-Based Skeletonization Algorithm, is accepted by Textile Research Journal.

Fu Shuai was born in China, 1997. He received a bachelor of engineering degree from Tianjin University of technology in 2009. At present, he is a master's student in Donghua University. His research field is software development.

He has achieved excellent results in undergraduate studies, won school level scholarships, and participated in some computer competitions and achieved good results. During postgraduate study, he participated in project development and won school level scholarships.