Hindawi
Mobile Information Systems
Volume 2022, Article ID 1344720, 11 pages
https://doi.org/10.1155/2022/1344720

Research Article

Infrastructure Smart Service System Based on Microservice Architecture from the Perspective of Informatization

Yanyan Lv[1] **and Wenken Tan**[2]

[1]*School of Digital Construction, Shanghai Urban Construction Vocational College, Shanghai 201499, China*
[2]*College of Architecture and Urban Planning, Tongji University, Shanghai 200092, China*

Correspondence should be addressed to Wenken Tan; twk@tongji.edu.cn

Received 28 April 2022; Revised 1 June 2022; Accepted 20 June 2022; Published 4 July 2022

Academic Editor: Ashish Bagwari

With the rapid development of engineering information technologies such as building Information Model (BIM), geographic information system (GIS), Internet of Things, big data, and cloud computing, and the intelligent management of infrastructure will become an inevitable development trend. Microservice architecture, because of the service component and other related characteristics, using this architecture can build a high availability and low coupling application system, which effectively improves the service quality of the system, and therefore the microservice architecture has gradually become the flow software development architecture. Based on the research of infrastructure digitization and integration of construction and maintenance, this paper proposes the concept of infrastructure intelligent service system (is3) from the perspective of information flow, so as to realize the intelligent management of infrastructure. This paper is based on the system requirements analysis. The intelligent service system of infrastructure with microservice architecture is designed, and its performance is tested through experiment. The test results show that in the brain high development test, the designed service reached no error, and the average response day was stable below 27 ms; in the continuous high concurrent test, the average response time of the designed service side remained within 100 ms when the concurrency is 12000, and no request occurred during the test.

1. Introduction

In the process of the continuous and in-depth development of the national smart city construction, the use of information technology can manage the entire life cycle of the project construction from beginning to end, running through all stages of the project construction process, which is conducive to promoting the construction of smart cities. With the advancement of Internet technology, human society has entered the cloud information era in which the Internet of Things technology, big data, and cloud computing technology are integrated and developed. In underground engineering activities, the development and progress of these information technologies is conducive to the information and intelligent management of underground engineering. Digital management methods have been widely used in infrastructure. With the explosive growth of data and the rapid development of information technology, "digitization" is gradually developing towards "intelligence."

This paper presents an understanding of Infrastructure Smart Services (iS3). iS3 mainly includes the functions of data collection, data processing, data visualization, and a series of analysis of infrastructure data for the whole life cycle of underground space engineering buildings. The whole life cycle of underground engineering refers to the survey stage, design stage, construction stage, and monitoring stage of the project, as well as the maintenance stage of the engineering structure during the operation period after the project is successfully completed. Due to the abundant underground engineering data, the iS3 system has many business functions. In order to realize the service of easy maintenance and modification of the system, as well as the analysis and decision-making of underground engineering data, this paper develops an infrastructure intelligent service system based on the microservice architecture. In recent years, the microservice architecture development model has been widely used in the Internet. Unlike the monolithic architecture system, the microservice is a system composed of a group of tiny services. Compared with the traditional monolithic application, the business modules in the microservice system that need to expand their functions or need to remodify their functions can be offline alone, and the entire system does not need to be offline, which makes the system easier to redeploy, and the system operation and maintenance management will be more convenient. The infrastructure intelligent service system is developed based on the idea of microservice architecture, and the data exchange is also completed between the microservices through the communication between the interfaces, which can promote the rapid transmission of 3D model files in the network and between services.

2. Related Work

In terms of theoretical research, experts and scholars from various countries have discussed the application and development prospects of intelligent infrastructure construction, and have done relevant research in the technical and application aspects of digital management. To achieve more accurate structural health monitoring, Spencer et al. designed a wireless smart sensor platform for civil infrastructure. The platform uses a 24-bit high-precision analog-to-digital converter with 8 analog input differential channels and programmable antialiasing filters to meet critical structural health monitoring needs, enabling tightly synchronized sensing. It addressed the data loss problem and efficiently implements the demanding numerical algorithms required for system identification and damage detection on resource-limited sensor nodes [1]. Al-Humairi and Kamal proposed a real-time Covid-19 system to track and identify suspected cases using an IoT platform to capture user symptoms and notify relevant agencies. Taking into account the effect of scanning distance compared to contact wearable sensors, she conducted a monitoring experiment that tested different age groups. The results showed that the system achieves 99.9% accuracy in the range of (500 ± 5) cm [2]. Imoize et al. researched sustainable social intelligence infrastructure supporting 6G, introduced the evolution background of different wireless communication standards and emerging 6G applications such as multisensory extended reality and digital replication. In addition, he discussed the technology-driven challenges facing the implementation of 6G and proposes possible solutions to these challenges [3]. To enable secure communication between smart meters and infrastructure, Khalid et al. proposed an anonymous key agreement protocol for smart grid infrastructure that enables smart meters to connect anonymously to public infrastructure. He verified the validity of the protocol through random Oracle models and automated ProVerif tools [4]. Kaluarachchi explored the potential advantages of smart and green infrastructure in cities to help cities achieve considerable environmental and socioeconomic benefits; he introduced the concepts of grey, green, and smart infrastructure and discussed the advantages of using nature-based integrated smart, green solutions [5]. Selim and Elgohary discussed the public-private partnership (PPP) in smart infrastructure projects from the perspective of stakeholders, and studied and analyze the role of stakeholders in smart infrastructure projects through the concept of PPP. The study aimed to establish a successful PPP model for the smart infrastructure project phase, clearly demonstrating the roles of stakeholders. Especially in the case of high project cost, it can help improve work efficiency and work quality while reducing costs [6]. Smith developed a real-time adaptive traffic signal control system to divert urban road traffic flow. The system combined principles of automatic planning and scheduling, multiagent systems, and traffic theory to treat traffic signal control as a decentralized online planning process. In operation, signal timing plans are repeatedly generated and executed at each intersection to optimize the number of vehicles currently sensed passing through the intersection [7].

3. Infrastructure Service System and Microservice Development Framework

3.1. Infrastructure Service System. The infrastructure service system is based on the same data standard to collect, process, visualize, and analyze the data of underground engineering from the preliminary geological survey data, topographic data, environmental data, design data, construction data, monitoring data, until the later operation and maintenance of data collection, processing, visualization and analysis, and decision-making functions, so as to serve the infrastructure management system of the entire life cycle of underground engineering [8], which is shown in Figure 1. The system also solves the problems of data information loss and poor interaction in the whole life cycle of the project. Data visualization refers to the visual expression in the form of combining engineering BIM three-dimensional model data and GIS two-dimensional graphics data with engineering data. From the perspective of information flow, the infrastructure smart service system can be understood as an integrated decision-making service system that collects, processes, expresses, and analyzes the life-cycle data of the infrastructure. It mainly serves infrastructure objects such as roads, bridges, tunnels, integrated pipe corridors, and foundation pits, covering the whole life cycle of different information flow nodes in various stages from planning, survey, design, and construction to operation and maintenance.

3.1.1. Data Collection. Data collection refers to monitoring and sensing infrastructure status and acquiring infrastructure data through various types of sensors, and converting the acquired infrastructure data according to certain rules to facilitate data transmission, processing, storage, display, recording, and control [9].

3.1.2. Data Processing. Because the environment where the sensor is placed may collect some information irrelevant to

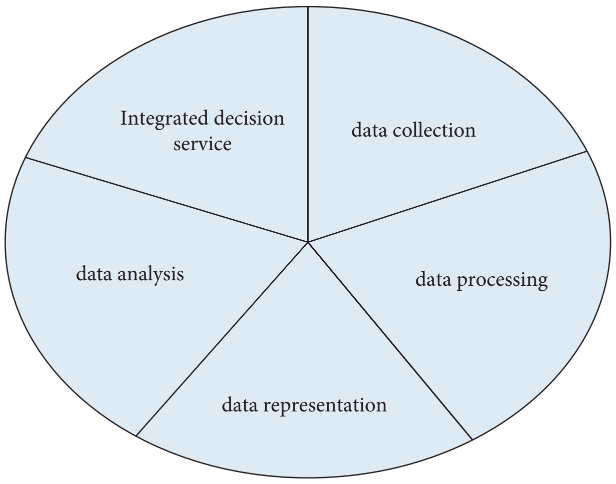


Figure 1: Conceptual diagram of infrastructure service system.

the state of the infrastructure, it is necessary to use some techniques to process the collected data [10]. The data at different stages of the infrastructure is denoised, classified, correlated, and fused by technical means. At the same time, infrastructure data should be standardized, including the standardization of data encoding and exchange.

3.1.3. Data Expression. Data representation is to display a large amount of abstract data generated by infrastructure in the process of engineering construction in a visual way [11]. Infrastructure data is included in planning, survey, design, construction, operation and maintenance. It detects and controls projects through 3D models and 2D GIS graphics.

3.1.4. Data Analysis. Data analysis is to use physical and mathematical methods to carry out qualitative and quantitative analysis of engineering construction in different aspects of the project and the whole life cycle [12], such as statistical analysis, artificial intelligence analysis, cost analysis, and big data analysis.

3.2. Microservice Development Framework. Microservices are services composed of multiple individual applications that can be centrally managed after completion through different programming languages and databases [13]. Microservices are divided by business functions, and these independent microservices are combined through the same protocol to form a final system or application, and data exchange between microservices is completed through network communication. If you need to expand a specific business function, you only need to expand the service of the business function, and you do not need to expand the entire system or application, so that you can improve the business with more confidence and improve the development efficiency of the team. The iS3 system in this article is developed using SpringCloud, a popular microservice framework. It also uses the development component Eureka provided by the SpringCloud family to complete the service registration discovery function, GateWay to complete the microservice gateway function, and Feign to implement the communication between microservices. The following content introduces the technical principles of these three parts in detail.

3.2.1. Service Registration and Discovery Eureka. The Eureka component implements the registration and discovery functions of microservices through the Eureka server and the Eureka client. Figure 2 shows the relationship between the two parts.

Eureka server and Eureka client are essentially a service, but they are configured differently in the service configuration file [14]. After each service is started, it registers its own IP, port number, service name, and other information with the Eureka server and periodically sends a signal to the Eureka server to check whether the service is alive. If the services need to call each other, the service consumer first uses the service name to initiate an application to the registry, obtains information such as the ip and port number of the service provider, and then calls the service. And the detailed information of these providers is cached locally, so that it can be used directly when there is a need for invocation [15].

3.2.2. Service Gateway. Every service in the actual development process needs to be packaged and published, so each service needs to use a different ip, address, and port number. The user's one click function on the client may need to request multiple services on the client. Microservice gateway is often developed into a system, which can avoid directly connecting the client and the server by isolating the client. The user's operation in the client only needs to send the request to the gateway system and then forward it to the back end for multiple services through the gateway. In this way, the real business services can be hidden in the Intranet, reduce the complexity of the client code, improve the security of the system, and also verify the client requests in the gateway system, monitor, and analyze the data. In this way, the backend microservices only need to care about the realization of business logic, and the gateway system to do their own duties [16]. Figure 3 briefly shows the overall architecture diagram of the gateway in the microservice architecture.

The simple routing management mode designed by SpringCloud's microservice component Gateway can not only forward all requests from clients uniformly and efficiently, but also perform security, monitoring, and current limiting on the microservice system. Figure 4 provides the basic flow of Spring Cloud's gateway component Gateway when processing client requests.

In the microservice gateway, the request sent by the client first finds the route corresponding to the request in the Gateway Handler Mapping, and sends the result to the Filter Web processor to find the corresponding filter chain. After the request is processed, the actual backend service is called, and the result is finally returned. It is possible to pass through the filter chain before requesting the backend

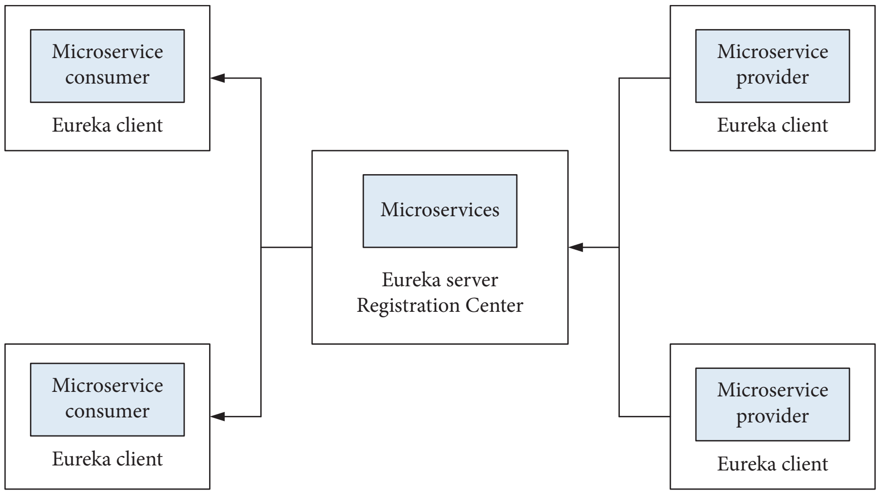


Figure 2: Eureka server and Eureka client.

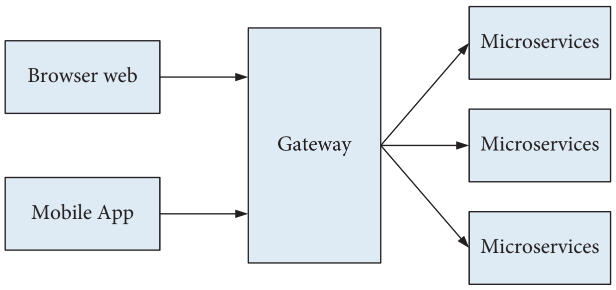


Figure 3: Schematic diagram of microservice gateway architecture.

business logic and after returning the result, so a dotted line should be added between the filter chains to separate [17].

3.2.3. Service Calls to Feign. In a system of microservice architecture, the services can obtain the required data through mutual visits, avoid the development of the same interface requirements for multiple services, and reduce the code repetition rate [18]. The communication between the service caller and the service provider in the Spring Cloud is HttpClient-based. Spring Cloud encapsulated httpClient, providing Rest Template and Feign communication, with the same Http call to the service as HttpClient. The Rest Template mode requires writing the service provider *s* url on the business code to call. However, if it is a more complex development situation, for system security considerations, you generally do not want to provide the details of the API. In this case, Feign components can be used for interservice calls. Feign encapsulates the Http calling process and is a declarative component. Noting the interface of the service provider in the actual development, the service name of the service provider is configured. Developers only need to call the interface of the paired service provider [19, 20]. This simplifies the writing of service requests and is more suitable for programming habits oriented to interface development. The Feign usage method is shown in Tables 1 and 2, Table 1 is the Feign process of the service provider, and Table 2 is the Feign process of the service caller.

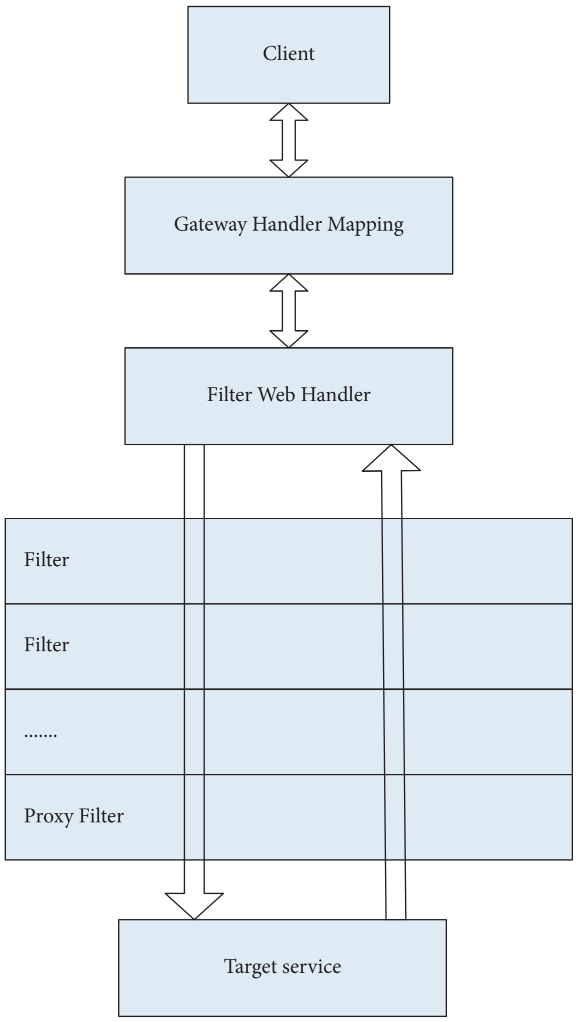


Figure 4: Flowchart of request processing by spring cloud gateway.

3.3. CHWBL Algorithm. The CHWBL algorithm is improved on the basis of the hash algorithm. The hash algorithm is currently a widely used load balancing algorithm in the field of load balancing. The main purpose is to ensure

Table 1: Feign process for service providers.

Begin	Define interface
1	@ FeignClient (name = “Product”)
2	Public interface ProductClient
3	@GetMapping (“/getProduct”)
4	Public string getProduct (String productId)
End	

Table 2: Feign process for service callers.

Begin	Inject at the place of the call
1	@ Autowired
2	Private ProductClient client;
3	@GetMapping(value = “getProduct”)
4	Public string getProduct (string productId)
5	Return client.getProduct (productId)
End	

that the same request is sent to the same service node each time to achieve cache hits and cross-domain problems [21].

The first hash algorithm is to send the user request to a specific server by performing a hash operation on the user’s IP and then modulo the number of servers, which can be expressed as formula (1) as follows:

$$h = \text{hash}(IP)\%n. \tag{1}$$

h is the corresponding server number obtained after the hash operation; hash is the hash algorithm used; IP is the IP address requested by the user; and n is the size of the service list.

Using this simple hashing algorithm to handle load balancing, when servers increase or decrease, all user requests need to relocate servers. Consistent hashing algorithm is optimized for this problem on the basis of simple hashing algorithm, abstracting the entire hash value space into a hash ring composed of nodes with a size of 232. The characteristic strings of the system nodes are hashed by the consistent hashing algorithm. The characteristic string is usually the IP address, name or port number of the service node, and a concatenated string, etc. The obtained hash calculation result is mapped to the hash ring, and the hash calculation formula (2) is as follows:

$$h = \text{hash}(IP)\%2^{32}, \tag{2}$$

where S is the characteristic string and 2^{32} is the size of the hash ring.

However, there is a problem with consistency hash algorithm. Conconsistency hash algorithm does not have a specific strategy of load balancing, and its load balancing effect mainly depends on whether hash algorithm is enough to randomly map requests and nodes to hash rings uniformly, if the randomness is insufficient. Uneven distribution can easily lead to a load tilt.

In view of the above problems, the significant improvement effect is CHWBL algorithm. In order to solve the problem of load tilt, CHWBL algorithm sets the upper limit of the load for each node. When the load of the node reaches the upper limit, the node will no longer accept the allocation of the request. This method can not only inherit the advantages of consistency hash algorithm, but also optimize the load tilt problem of consistency hash algorithm to prevent some nodes from overloading others but being relatively idle.

For example, when the system receives a request at time s, the load balancer of the system will calculate the total number of requests at the current time of the system, which can be expressed as formula (3) as follows:

$$L_{\text{sum}} = \sum_{i=1}^{n} L_{i(s)} + L_{\text{new}}. \tag{3}$$

Among them, L_{sum} is the total request load of the current system; $L_{i(s)}$ is the number of load requests being processed by the i-th node at time s; and L_{new} is a new request. At this time, the limit of node load is set as the average value of each node load, which can be expressed as formula (4) as follows:

$$\overline{L} = \frac{L_{\text{sum}}}{n}. \tag{4}$$

As shown above, $\overline{L}$ is expressed as the average load of all nodes; n is expressed as the total number of nodes. On this basis, the concept of a balance constant ε is introduced to dynamically control the upper limit of the load of the node. It can be set according to actual needs, and can be set separately or uniformly for different service nodes.

Through this method, the upper limit of the node load can be expressed as formula (5) as follows:

$$M_i = \overline{L} * (1 + \varepsilon). \tag{5}$$

As shown above, M_i is the upper limit of the load of the node, which is calculated by the average load of the system and the balance constant ε.

If the current system has m requests and n service nodes, the load number M of the nodes should satisfy the following formula:

$$M < \left\lceil (1 + \varepsilon) * \frac{m}{n} \right\rceil. \tag{6}$$

By limiting the upper limit of the load of service nodes in the consistent hashing algorithm, the CHWBL algorithm significantly improves the load balancing effect compared with the traditional consistent hashing algorithm, and retains the original excellent characteristics of the consistent hashing algorithm.

In a heterogeneous cluster with a distributed microservice architecture, there are differences in performance among the servers in the cluster. It cannot calculate the upper limit of the load of each node only based on the average load of the system like a homogeneous cluster. In order to adapt CHWBL to a heterogeneous cluster of microservices, it is necessary to set a weight for the nodes in the microservice architecture. The weight of each node can be expressed as formula (7) as follows:

$$W_i = \frac{P_{\text{cpu}} * W_{\text{cpu}}}{AVG_{\text{cpu}}} + \frac{P_{\text{ram}} * W_{\text{ram}}}{AVG_{\text{ram}}} + \frac{P_{\text{net}} * W_{\text{net}}}{AVG_{\text{net}}}. \tag{7}$$

Among them, W_i is the weight of the node; P_{cpu}, P_{ram}, and P_{net} represent the performance indicators of the current node's CPU MIPS, memory ram size, and network net bandwidth; W_{cpu}, W_{ram}, and W_{net} represent the weight of each indicator; and AVG_{cpu}, AVG_{ram}, and AVG_{net} represent the average performance indicator of all nodes in the system.

Combining the entropy weight method to calculate, the current three indicators can be expressed as X_1, X_2, and X_3. Among them, $X_j = \{x_1, x_2, \ldots x_n\}$ represents the parameter of each node of this indicator. Standardize each index to get Y_{ij} of each index, which can be expressed as formula (8) as follows:

$$Y_{ij} = \frac{X_{ij} - \min(X_j)}{\max(X_j) - \min(X_j)}. \tag{8}$$

Among them, Y_{ij} is the normalized value of the j-th indicator of the i-th node; X_{ij} is the specific value before standardization; $\min(X_j)$ represents the smallest value in a group of indicators; $\max(X_j)$ represents the largest value in the indicator. Calculate the proportion of the jth index of the ith node, which can be expressed as formula (9) as follows:

$$p_{ij} = \frac{Y_{ij}}{\sum_{i=1}^{n} Y_{ij}}. \tag{9}$$

p_{ij} is the proportion of the indicator. The information entropy of each index can be obtained by the proportion of the index, which can be expressed as formula (10) as follows:

$$E_j = -ln(n) \sum_{i=1}^{n} p_{ij} ln(p_{ij}), \tag{10}$$

where E_j represents the information entropy of the jth indicator; The weight of the indicator can be calculated through the information entropy of the indicator, which can be expressed as formula (11) as follows:

$$W_j = \frac{1 - E_j}{\sum_j 1 - E_j}. \tag{11}$$

W_j represents the weight of each indicator of W_{cpu}, W_{ram}, and W_{net}. These three weights need to be added together to be 1, which can be expressed as formula (12) as follows:

$$W_{\text{cpu}} + W_{\text{ram}} + W_{\text{net}} = 1. \tag{12}$$

At this time, the sum of the service node weights is equal to the number of nodes, which can be expressed as formula (13) as follows:

$$\sum_{i=1}^{n} W_i = n. \tag{13}$$

The above weight division method defines the weight according to the different performance of the nodes, so as to ensure that the nodes with good performance can be divided into higher weights, so as to allocate more load; on the contrary, nodes with poor performance have smaller weight division and less request load allocated. Then, according to the calculation formula of the CHWBL algorithm, the number of nodes is replaced by the sum of the weights of each node, T, because $T = n$, the average load of the node can be expressed as formula (14) as follows:

$$\overline{L} = \frac{L_{\text{sum}}}{T}. \tag{14}$$

After introducing the balance constant ε, the upper limit of the load of each node can be expressed as formula (15) as follows:

$$M_i = W_i * \overline{L} * (1 + \varepsilon). \tag{15}$$

At this time, the number of requests of each node in the system should not exceed the upper limit of its own load M_i.

4. Microservice Architecture Infrastructure Smart Service System Design

4.1. System Requirements. Infrastructure intelligent service system is based on the infrastructure data management of the whole life cycle of underground engineering with the same data standard, which mainly includes data collection, data processing, data visual expression of each link of the project, qualitative analysis of infrastructure data, and the corresponding decision-making function of the project. The data collected by various sensors and the data from different stages of underground engineering need to be imported into the corresponding engineering database of the system. Users can view the two-dimensional plan and 3D model map of the underground space infrastructure. Based on the various engineering data of the infrastructure collected in the database, a qualitative and quantitative analysis of the problems arising in the operation and maintenance process in the underground engineering can be conducted.

Because the environment where the sensor is placed may collect some information unrelated to the state of the infrastructure, some technologies are needed to process the collected data. Then through the same data standard processing, it was saved to the system engineering database. Users can also realize the management functions of data display, query, delete, and modify the engineering data information stored in the database. Develop strategies and qualitatively analyze underground engineering based on the data collected from the current infrastructure. In the design stage, underground space infrastructure will have a variety of two-dimensional plans, horizontal section, vertical profile, and two-dimensional and 3 D view models. In iS3 system, abstract data such as engineering data and model data in infrastructure can be used for visual expression and interactive sharing, so as to facilitate users' visual cognition of abstract data such as engineering and engineering data. In iS3 system, 2D graphics data and 3 D model data, including engineering data, can be uploaded and managed through files. The system also needs to design the function of user registration and login, grant the corresponding function permission to legal users, and the corresponding module function in the visit system. Table 3 shows the systematic divided functional categories and detailed functional descriptions.

Table 3: Statistics on functional requirements of iS3.

Function	Description
BIM model display	Build and render 3D models on the web 2D CAD graphics add component elements
2D graphics display	2D graphics as ArcGIS layers Browser 2D graphics with component elements, GIS data as basemap
Engineering data	Query, deletion, and modification of facility data The original engineering data provided by the cooperative unit
User management	User registration, login, permission verification, and other functions

4.2. System Design Ideas. The design of the infrastructure smart service system mainly includes two parts: the microservice technology stack and the business service. Among them, the SpringCloud technology is used to build the microservice architecture of the system. The business service refers to the service that truly realizes the business logic and provides results for the client's request. The infrastructure smart service system selects some components in SpringCloud when building the microservice architecture, and these components are indispensable in system development. It mainly includes the service registration and discovery component Eureka, the gateway component Gateway, and the mutual invocation component Feign between microservices.

The system is separated from frontend and backend: the frontend development framework is Vue.js, and the infrastructure data is stored in the MySQL data server; backend service development uses SpringBoot technology. SpringBoot improves and optimizes the traditional Spring development framework, reducing the lengthy and insignificant configuration file writing work. Using SpringBoot technology to develop will greatly improve the development efficiency of the application and greatly shorten the development cycle of the project. The development advantages of SpringBoot technology can be summarized as: ① Compared with the traditional framework structure, SpringBoot can quickly construct projects and the developed projects can run independently. ② SpringBoot does not need to rely on other servers externally due to the embedded Tomcat and other servers. ③ For the integrated use of many popular frameworks, you only need to install the dependency packages of the corresponding frameworks or configure them.

When part of the 2D graphics data designed in the system is loaded into the browser, the current geographic location needs to be displayed as the base map, and the 2D graphics can be imported into the ArcGIS server as a layer. The client loads the map resources and corresponding two-dimensional graphics to the frontend page for display by requesting the interface of the ArcGIS server. Table 4 shows the environment resource configuration requirements for system development to meet the needs of iS3 system development.

Table 4: iS3 system development environment.

Software type	Name and version
Operating system	Windows7 and above
Database	MySQL5.8
Backend development platform	IDEA2019.03
Frontend development platform	VSCode2017
Browser	Chrome
Java platform	Jdk1.8

4.3. Overall System Architecture. The infrastructure intelligent service system designed in this paper uses the microservice architecture built by the Spring Cloud framework. The system as a whole includes a three-layer architecture of data layer, business service layer and display layer. Figure 5 shows the overall architecture of the infrastructure smart service system.

4.3.1. Display Layer. The essence of the infrastructure smart service system is a web application, using the Vue framework and Three.js technology. The Vue framework is mainly used to design Web pages, and the Three.js technology is used to load JSON 3D model files, build, and render 3D models. When a user clicks a function on a web page, the browser sends a request to the backend. These requests first need to be forwarded through the microservice gateway system, after which OAuth2.0 authenticates and authorizes the current user, and then the backend-related microservice application responds. The microservice gateway undertakes services such as service aggregation and protocol conversion. By providing a coarse-grained API to the outside world, it reduces the http requests that need to be initiated when a page needs to access multiple microservices and decouples the front and back ends of the iS3 system.

4.3.2. Business Service Layer. The background business of the infrastructure smart service system mainly includes basic services and application services. The basic services include the Eureka microservice registry, the log service that monitors the running status of the system, and the cache service that uses Redis technology. The main application services of iS3 include data third-party interface service, engineering data management service, 2D data display service, file management service, BIM model display service, and user service.

4.3.3. Data Layer. The data layer includes infrastructure data, 3D model data, and 2D graphics data for different aspects of the project.

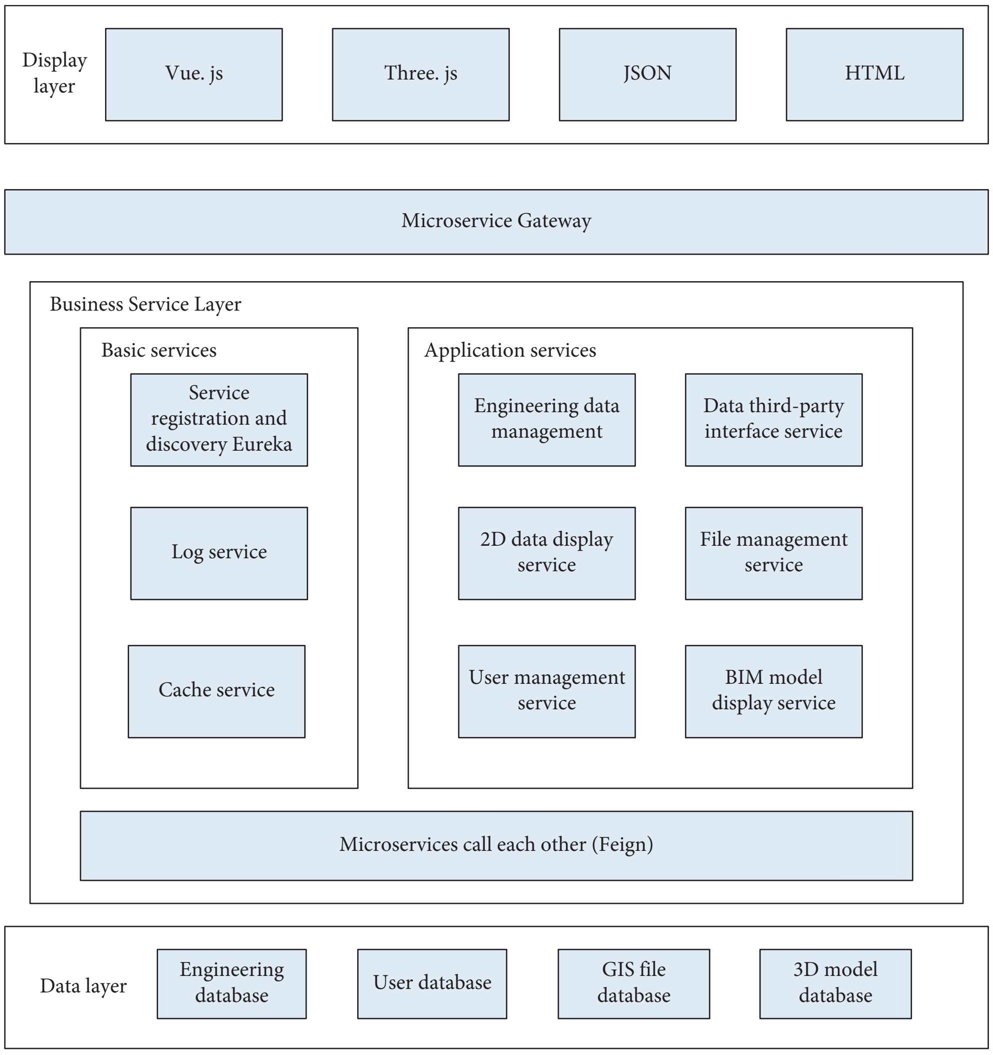


Figure 5: Overall architecture of iS3 system.

According to the requirements analysis and overall architecture design of the infrastructure intelligent service system, this section will divide the detailed functional modules of the business service layer of the system, and the system business service layer is mainly divided into the basic service module and the application service module. The basic service module mainly includes three parts: service registration and discovery module, log management, and cache management. The application service module is the main function module in the iS3 system, which needs to interact with the database and exchange data with the frontend page, which is related to the overall function of the system. The application service layer mainly includes six parts: data third-party interface service, engineering data management service, file management service, BIM model display service, 2-dimensional data display service, and user service to realize the mapping of requirements and functions. After dividing several modules of the application service layer, the specific business functions of each microservice need to be designed in more detail. Figure 6 shows the main functions of the design of the infrastructure intelligent service system.

The data third-party interface is responsible for automatically screening the collected infrastructure data according to the specified requirements, transforming it according to the unified standards, and finally storing the work in the engineering database. Engineering data management service realizes the management functions of infrastructure data query, increase, book division, and modification by operating the infrastructure data in the engineering database. The file management service can realize the unified upload and management of uploaded files (including engineering data files, 2-D graphics line data files, 3-D model data files,). The 2D model display service is responsible for loading the 2D graphics combined with ArcGIS into the frontend page for display and realizing the linkage of 2D data and infrastructure data. User service mainly designs the unified registration, login, modify personal information, and exit system functions, at the same time, the user service also provide permission setting and verification information function. The BIM Model Display service is responsible for uploading and displaying the BIM model, and can also view the

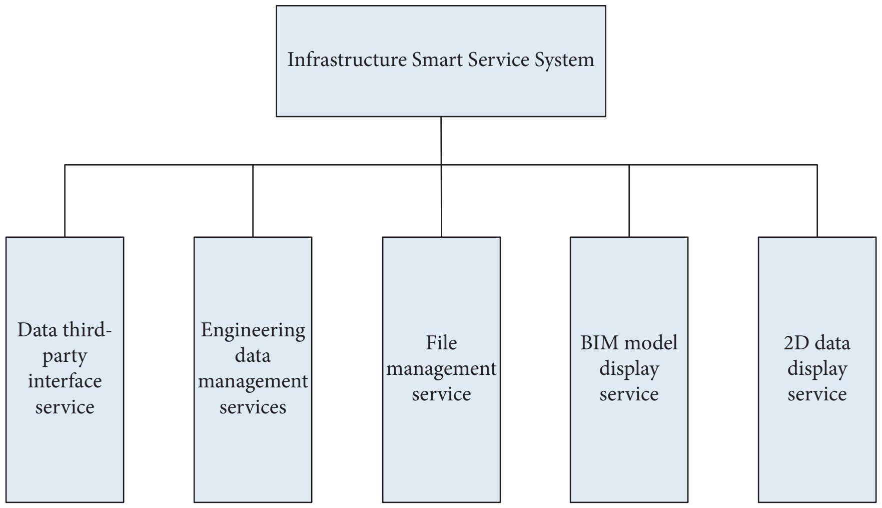

Infrastructure Smart Service System
Data third-party interface service
Engineering data management services
File management service
BIM model display service
2D data display service

Figure 6: System function module design diagram.

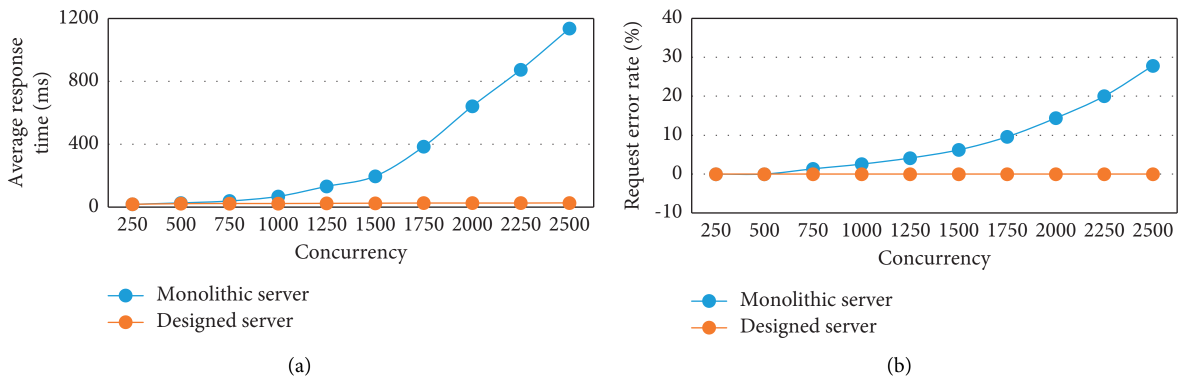


Figure 7: Instantaneous high-concurrency test results: (a) the average response time of the front and rear service end for instantaneous high concurrency and (b) the request error rate of the front and rear service under instantaneous high concurrency.

corresponding infrastructure data by clicking on the components in the model.

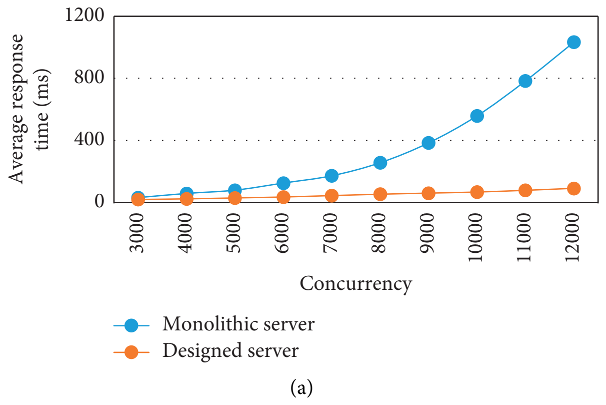
5. Microservice Architecture Infrastructure Smart Service System Functional Test

This section will stress test the server side of the infrastructure smart service system before and after the design, and compare it with the server side under the previous single architecture. To this end, Apache JMeter was selected as a stress testing tool to test the average response time and error rate of the server under high concurrency, simulate the concurrent access of a large number of users to the server through parameter settings, so as to verify the performance of the server.

5.1. Instantaneous High-Concurrency Test. In the transient high concurrency test, set the Ramp-UpPeriod parameter of the JMeter thread group to 1s. That is, start all threads within 1s, and then gradually increase the number of threads to increase the concurrency. The test results are shown in Figure 7:

As can be seen from Figure 7, under the instantaneous high-concurrency test, the server performance before and after the design is close when the concurrency number is within 500. As the concurrency number increases, the average response time between the two also increases, and when the concurrency number reaches 1500. The server error rate before the design was 6.21%. When the concurrent number reaches 2500, the average response time of the server before the design exceeds 1s and the error rate is 27.85%, while the designed server has no error and the average response time is relatively stable. The analysis of the test results can show that the designed server can significantly reduce the average response time and error rate under the instantaneous high-concurrency situation, and then improve the overall performance.

5.2. Continuous High-Concurrency Testing. In continuous high-concurrency testing, the Ramp-UpPeriod parameter of the JMeter thread group is set to 10s, which is changed to start all threads within 10s, and increase the number of threads successively directly from the concurrency number 2000. The results obtained from the test are shown in Figure 8:



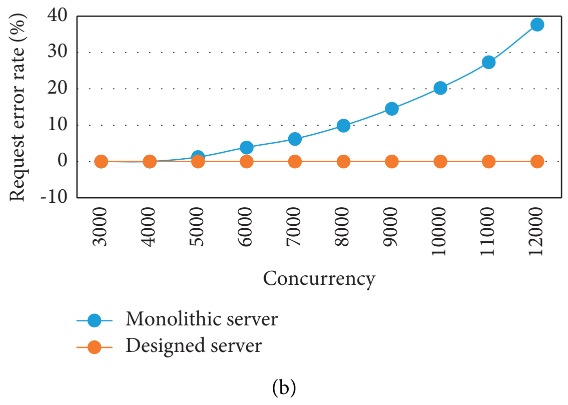


FIGURE 8: Continuous high-concurrency test results: (a) the average response time of the before and rear service for continuous high concurrency and (b) the request error rate of the front and rear service for continuous high concurrency.

As can be seen from Figure 8, under the continuous high-concurrency situation, the average response time and error rate growth trend of the before and after the design server are basically consistent with the test situation under the instantaneous high development. With the increase of the concurrency, the average response time and error rate also increase significantly, the average response time exceeds 1s at 12000 and the error rate reaches 37.73%, while the average response time is stable, maintained within 100 ms when the concurrency is 12000, and no request error occurs during the test.

Based on the above test results, it can be clearly concluded that the average response time and error rate of the service side are much lower than that of the design in the case of instantaneous high development or continuous high concurrency. Therefore, it can be concluded that the iS3 system server based on the microservice architecture can effectively improve its concurrent processing capacity and have excellent performance.

6. Conclusion

This paper makes a detailed analysis of the specific requirements and design of infrastructure intelligent service system. The innovative content is to combine microservice architecture to design iS3 system, study, and realize the data display of iS3 system and the display and interaction of 3D model in the Web in business service link. However, with the development of the Internet, big data, cloud computing, the Internet of Things, and the popularization of mobile technology, we need to pay attention to future research directions. The future research needs to be carried out from the following aspects: (1) The current iS3 system is running on a single server. Once the server collapses for some reason, the whole system will be offline and users cannot visit it. Considering this case, we can extend single points to clusters. (2) The current iS3 is only applicable to dry Web pages. For users who want to view the project progress and infrastructure status anytime and anywhere, they can design a simple version of the mobile iS3 system to facilitate the visual expression of iS3.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest with any financial organizations regarding the material reported in this manuscript.

References

[1] B. F. Spencer, J. W. Park, and K. A. Mechitov, "Next generation wireless smart sensors toward sustainable civil infrastructure," *Procedia Engineering*, vol. 171, pp. 5–13, 2017.

[2] S. Al-Humairi and A. Kamal, "Design a smart infrastructure monitoring system: a response in the age of COVID-19 pandemic," *Innovative Infrastructure Solutions*, vol. 6, no. 3, pp. 1–10, 2021.

[3] A. L. Imoize, O. Adedeji, and N. Tandiya, "6G enabled smart infrastructure for sustainable society: opportunities, challenges, and research roadmap," *Sensors*, vol. 21, no. 1709, pp. 1–57, 2021.

[4] M. Khalid, X. Li, and C. S. Ashraf, "Pairing based anonymous and secure key agreement protocol for smart grid edge computing infrastructure," *Future Generation Computer Systems*, vol. 88, pp. 491–500, 2018.

[5] Y. Kaluarachchi, "Potential advantages in combining smart and green infrastructure over silo approaches for future cities," *Frontiers of Engineering Management*, vol. 8, no. 1, pp. 98–108, 2021.

[6] A. M. Selim and A. S. Elgohary, "Public–private partnerships (PPPs) in smart infrastructure projects: the role of stakeholders," *HBRC Journal*, vol. 16, no. 1, pp. 317–333, 2020.

[7] S. Smith, "Smart infrastructure for future urban mobility," *AI Magazine*, vol. 41, no. 1, pp. 5–18, 2020.

[8] D. G. Broo and J. Schooling, "Towards data-centric decision making for smart infrastructure: data and its challenges," *IFAC-PapersOnLine*, vol. 53, no. 3, pp. 90–94, 2020.

[9] I. C. Konstantakopoulos, A. R. Barkan, and S. He, "A deep learning and gamification approach to improving human-building interaction and energy efficiency in smart infrastructure," *Applied Energy*, vol. 237, pp. 810–821, 2019.

[10] M. Podpora, A. Gardecki, and A. Kawala-Sterniuk, "Humanoid receptionist connected to IoT subsystems and smart infrastructure is smarter than expected," *IFAC-PapersOnLine*, vol. 52, no. 27, pp. 347–352, 2019.

[11] B. Cully and J. Jaco, "Make data-driven decisions using intelligent dashboards and smart infrastructure," *Journal - American Water Works Association*, vol. 111, no. 3, pp. 72–74, 2019.

[12] T. S. Ustun, S. Hussain, and H. Kirchhoff, "Data standardization for smart infrastructure in first-access electricity systems," *Proceedings of the IEEE*, vol. 107, no. 9, pp. 1790–1802, 2019.

[13] L. Brown, "Improving municipal management with smart infrastructure," *World Water and Environmental Engineering*, vol. 42, no. 1, pp. 24-25, 2019.

[14] A. Wyllie, "Improving lives in our global society by being at the forefront of the smart infrastructure revolution," *Civil Engineering*, vol. 172, no. 1, pp. 1–6, 2018.

[15] H. Dobashi, Y. Nagata, and M. Takano, "New smart infrastructure management system of metropolitan expressway," *Concrete Journal*, vol. 56, no. 1, pp. 25–29, 2018.

[16] C. J. Bouch, C. D. Rogers, and M. J. Powell, "Developing alternative business models for smart infrastructure," *Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction*, vol. 171, no. 2, pp. 1–38, 2018.

[17] N. Cuong and T. C. Duy, "Information technology infrastructure for smart tourism in da nang city," *International Journal of Hyperconnectivity and the Internet of Things*, vol. 5, no. 1, pp. 98–108, 2021.

[18] A. Penn and K. A. Sayed, "Spatial information models as the backbone of smart infrastructure," *Environment and Planning B*, vol. 44, no. 2, pp. 197–203, 2017.

[19] S. T. E. W. A. R. T. Kantor, "Finding a way in the fog: the race to outpace hackers vs. Smart infrastructure," *POWERGRID International*, vol. 22, no. 7, pp. 20–24, 2017.

[20] A. Cho, "Smart infrastructure shapes future growth," *Engineering news-record*, vol. 279, no. 17, pp. 56-57, 2017.

[21] P. R. C. Araujo, R. Holanda Filho, and J. Rodrigues, "Middleware for integration of legacy electrical equipment into smart grid infrastructure using wireless sensor networks," *International Journal of Communication Systems*, vol. 31, no. 1, Article ID e3380.1, 2017.