Technical Research and Application Analysis of Microservice Architecture

Leyi Gou, Qing Chen, Jing Liang and Xuehua Liao

ABSTRACT

The emergence of micro-service architecture solves the problems of high complexity, difficult maintenance, difficult project expansion and technical innovation of monolithic architecture in traditional application systems. This paper studies and analyzes the key problems and solutions in the construction of distributed micro-service architecture, summarizes and compares the current mainstream micro-service architecture solutions, and finally adopts the optimal solution to build the microservice architecture model of intelligent logistics distribution platform.

1. INTRODUCTION

Monomer architecture refers to all the functional modules of the application as a whole package and deploy. Applications with a monolithic architecture are easy to develop, test, and deploy, and many projects adopt this pattern early in development. As the overall application becomes more complex, the time to build and deploy the project increases, and when different functional modules of the application have conflicting resource requirements, it becomes more difficult to scale individual applications. In order to better conduct distributed application development and achieve the goals of high concurrency, high availability and high performance, Martin Fowler and James Lewis put forward the idea of micro-service. The emergence and
successful application of microservices at home and abroad make it a new choice of system architecture. Many projects of large companies have migrated from traditional monolithic architecture to microservice architecture [5].

2. MICROSERVICE ARCHITECTURE KEY ISSUES

Microservice architecture is an architectural idea, a collection of small services. As a distributed system solution, Microservice Architecture can build complex distributed microservice applications. In distributed system development, the following four problems are usually encountered. How do clients access these?

3. The Service Gateway

Micro service architecture splits a system into several independent services, each of which is basically deployed independently on a virtual machine or cloud. The foreground needs to remember and manage how many services there are in the background. When a service goes offline or changes, the foreground needs to be redeployed. The microservices architecture is usually stateless inside the system. After logging in to a service, users need to log in again if they want to use other services. Therefore, it is necessary to record the user's session state. An example is shown in Figure 1, when there are multiple services and the IP of the A service has changed, if the user is using a desktop client, the desktop client will automatically update, but if the user is using a mobile client, then the user needs to update the app, so it is very troublesome to update if there is a change, as shown in Figure 1.

Figure 1. Service calls from different clients. Figure 2. Using API gateway.
Therefore, there needs to be a place for unified maintenance, management of permissions, etc. Generally, an authorization server, called API gateway, needs to be set between multiple services in the background and the front page, as shown in Figure 2. On the PC or Android/iOS side, just remember the IP of the API gateway, and the API gateway remembers the IP address of each service. In this way, even if the service is upgraded or updated offline, it only needs to be adjusted in the API gateway, and the PC or Android/iOS side will not need to update or perform other operations.

At present, the mainstream API gateways are Spring Cloud Zuul and Spring Cloud Gateway. The API gateway uses the Ribbon to implement the client-side load by default. Through the service discovery and registration center, the address and port of all back-end services can be obtained, and the load balancing algorithm is used to balance the forwarding request to the background microservice.

3.1 The Service Discovery and Registration Center

In microservice architecture, in order to achieve load balancing, there are often multiple services for one function. According to the usage requirements of the application, new service nodes may be deleted or added at any time. Then, how do the services perceive each other, how should the additional and reduced services be managed, and how can the API gateway obtain each service? At this time, the Service Discovery and Registration Center will be used.

As shown in Figure 3, in order to achieve load balancing, two A services and two B services are deployed. When each service goes online, it needs to register with the service discovery and registration center first, and register the IP of the service instance to the service discovery and registry. The Service Discovery and Registration Center manages all IP addresses and maintains a heartbeat connection with the service node. If a service is detected, it will update its IP list. When the API gateway needs any service, configure the corresponding service name and other information in the configuration, and then send the request to the service discovery and registration center to tell the center what it needs, such as "I want a service A, please assign a service. Give me A's IP and port", the service registration and discovery center will feedback the corresponding information to the API gateway. If a service A goes offline, the service discovery and registration center will delete the corresponding service in the list. When the gateway requests it, the center will return the existing service A IP information in the updated list.
There are three main types of service discovery and registration centers: Zookeeper, Eureka of Spring Cloud Netflix, and Nacos of Spring Cloud Alibaba.

### 3.2 Service Communication

There are many mature solutions for communication between services. Currently, the most common ones are synchronous calls and asynchronous message calls. Spring cloud has two service invocation methods, one is LoadBalanceClient + RestTemplate and the other is Feign. Spring Cloud Alibaba also has two service invocation methods, one is Ribbon + RestTemplate and the other is Feign. In actual development, we usually use Feign, which is a declarative pseudo-Http client that communicates in a synchronous blocking manner. With Feign, you only need to create an interface and annotate it. Feign integrates Ribbon by default, and Nacos is also very compatible with Feign, which implements load balancing by default.

### 3.3 Service Fault Tolerance

In order to ensure that the entire system fails without the failure of individual services during the entire microservice chain call, the service fault tolerance scheme is provided in the microservice solution. In the project, if the CPU memory is blocked due to an abnormality of a certain service, the request for the abnormal state can be directly returned by the fuse, thereby reducing resource waste.

Fuse mode is currently implemented with Sentinel components, Hystrix components, and more.
4. SOLUTION FOR MICROSERVICE ARCHITECTURE

There are currently three mainstream microservice architecture solutions: Solution 1: Spring Boot + Spring Cloud Netflix; Solution 2: Spring Boot + Dubbo+ Zookeeper; Solution 3: Spring Boot + Spring Cloud Alibaba. Spring Cloud is an ecosystem that integrates a range of solutions for distributed microservices development. Spring Cloud must be based on Spring Boot. In Figure 4, the key technologies used in solving these three sets of solutions are summarized and compared.

<table>
<thead>
<tr>
<th>Problems</th>
<th>Solution 1:</th>
<th>Solution 2:</th>
<th>Solution 3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client access method</td>
<td>Use the API Gateway. Netflix provides Zool components.</td>
<td>Use the API Gateway. However, this solution does not provide its own gateway component, so it needs to use third-party components or implement it yourself.</td>
<td>Use the API gateway. Spring Cloud Gateway can replace Zool.</td>
</tr>
<tr>
<td>Communication between services</td>
<td>Netflix provides a Feign client that uses HTTP communication.</td>
<td>Use Dubbo, a high-performance JavaRPC communication framework.</td>
<td>Use the Feign client.</td>
</tr>
<tr>
<td>Service governance</td>
<td>Use the Service Registration and Discovery Center. Netflix offers Eureka.</td>
<td>Use Zookeeper to resolve service registration and discovery issues.</td>
<td>Spring Cloud Alibaba provides Nacos components instead of Eureka.</td>
</tr>
<tr>
<td>Solution to problems with the service</td>
<td>Use the fuse mechanism. Netflix offers Hystrix.</td>
<td>Hystrix with Netflix.</td>
<td>Alibaba has opened up the Sentinel component and implemented the fuse mode.</td>
</tr>
<tr>
<td>Advantage</td>
<td>Netflix provides a stack of help for building a microservices architecture.</td>
<td>After the first set of programs stopped maintenance, it was a supplement to it.</td>
<td>Provides a one-stop solution for microservices architecture development.</td>
</tr>
<tr>
<td>Disadvantage</td>
<td>In 2018, the Spring Cloud Netflix family of technology stacks entered maintenance mode, eliminating the</td>
<td>This program is not perfect and needs to borrow many third-party components.</td>
<td>Continue to observe.</td>
</tr>
</tbody>
</table>

Each of the above schemes has its own advantages and disadvantages. We should select according to the actual situation of the application. This paper adopts the third scheme. Spring Cloud Alibaba is a micro-service architecture solution proposed by Alibaba. The Spring Boot + Spring Cloud Alibaba solution is a one-stop distributed system development solution after Spring Cloud Netflix, which simplifies the development process and improves development efficiency. Developers can load applications into Alibaba's ecosystem with simple annotations and configuration. With the support of Alibaba, the project can achieve high concurrency, high performance and high availability.
5. MODEL ANALYSIS OF INTELLIGENT LOGISTICS DISTRIBUTION PLATFORM BASED ON MICROSERVICE ARCHITECTURE

The rapid development of the logistics industry has made the use of logistics distribution platforms more and more extensive. A large number of logistics transactions require logistics distribution platforms to have high availability and high performance. Therefore, this paper adopts the idea of microservice architecture and uses the third scheme from the above analysis. At the end of the entire logistics supply and distribution chain, the terminal micro-service intelligent practical development is carried out, that is, the regional logistics intelligent distribution platform model is built. In the microservice architecture, functions that change for the same reason are grouped together to form a micro-service, and functions that change for different reasons are placed in different micro-services. From the functional division, the simple intelligent distribution platform is divided into four modules: courier recruitment, user retrieval, message notification, and payment.

![Figure 5. Microservice Framework for Intelligent Distribution ·Platform.](image)

Through the functional division of the intelligent distribution platform, according to the characteristics of the micro-service architecture, the microservice framework diagram of the intelligent distribution platform shown in Figure 5 is constructed. First, service providers and service consumers need to register with the Service Registration and Discovery Center. This article uses Nacos as a service discovery and registration center. After the registration is successful, the consumer can route to the corresponding service through the API gateway. In this process, the load balancing server will be allocated according to the status of the service provider. The API gateway adopts the Spring Cloud Gateway and implements the fuse mode using the Sentinel component.

As shown in Figure 5, the operation process of the platform is as follows:

1. Users fill in the item information on the desktop client, mobile client or express counter. After the filling is completed, the platform will generate the
delivery code, indicating successful delivery. At the same time, the platform will call the message to inform the micro service, which will inform the Courier to pick up the item.

(2) After the delivery is confirmed to be correct, the Courier micro service will inform the payment service and ask the user to pay.

When the Courier delivers the goods, the RFID tag of the goods is scanned at the end of the express cabinet. At this time, the Courier micro service will call the message to inform the micro service, which will inform the user to pick up the goods.

(3) The platform built through micro-service architecture can achieve the goal of high concurrency, high availability and high performance. However, when the functional requirements increase, we should reasonably divide the services, otherwise it will bring certain difficulties to the development.

6. CONCLUSIONS

Micro service architecture solves the problems of difficult expansion, technical innovation, low start-up efficiency and continuous deployment, and provides a new set of distributed solutions for application system development. This paper analyzes and introduces the main technical links in the micro-service architecture construction process, summarizes and compares the current mainstream micro-service architecture solutions, and finally uses the Spring Boot + Spring Cloud Alibaba solution to build the intelligent logistics distribution platform model, which has certain reference value for the construction of application systems.

ACKNOWLEDGEMENTS

Fund Projects Participated in the Project: Science and Technology Key Research and Development Project of Sichuan Science and Technology Department (NO.2018GZ0270); National Natural Science Foundation of China (NO.11575121)
REFERENCES


