Design of Reading Platform Based on Microservice Architecture

Yang LI, Yan-song CUI and Dan-zhi WANG

School of Electronic Engineering, Beijing University of Posts and Telecommunications, China

Keywords: Distributed, Docker, Microservice architecture.

Abstract. Under the general trend of informatization and intelligence, the traditional paper book reading cannot meet the rapid development of intelligent Internet at present, either in terms of information content or convenience. In order to build a reading platform system that is easy to deploy and maintain, an Internet reading platform system based on container technology (such as Docker) is proposed. This paper analyzes the structure and characteristics of the microservice architecture, and gives a specific planning design around the platform framework such as Spring Cloud, encapsulation and deployment of the microservice architecture.

Introduction

Internet today has been into people's hearts and fully integrated into the society every level. All walks of life, such as government departments, organizations and individual citizens are fully accept and rely on the Internet, using the Internet to provide a variety of functions and convenience. With agile development, DevOps, the development of IT construction concept and the maturity of Docker container technology, Microservice Architecture becomes possible to be carried out in recent years. By contrast, most web projects package all the service side together and run in a single web container, and many enterprise Java applications are packaged as war packages. In this trend, the platform design idea based on microservice architecture has also provided a new idea for reading platform planning and design. Figure 1 shows the differences between a microservice architecture and an overall architecture.

![Microservice Architecture](image)

**Figure 1. Difference.**

Microservice Architecture

The Design Concept of Microservice Architecture

The traditional Internet application platform is to package the application into a complete application and deploy all the applications to run in a process. This kind of traditional architecture model is easy to be developed, tested and deployed. If all the functions of a system are developed with a single engineering configuration. The technology is not flexible enough and the scalability is not high enough. Figure 2 shows a traditional overall architecture deployment.
Microservice architecture is an architectural pattern gradually established in practice by Amazon, eBay, Netflix and other companies to decompose large systems for development and maintenance. Its core idea is that functions in the complicated application system decompose to each discrete services in the form of independent business unit. There is a clear boundary of function in each service center around specific business, which generally uses lightweight communication mechanism like JSON to conduct information interaction [1]. Building an application system based on a microservice architecture is a new option.

The concept of microservice architecture is a product of rapid development of technology concepts such as agile development, continuous delivery, virtualization, Dev Ops [2]. From the platform, the different micro small project of form a complete service, through a separate deployment, maintenance alone, separate management, through such as lightweight communication mechanism between the microservice to interact with the micro way of building service in the deployment of upgrade only change need to upgrade the microservice deployment[3].

### Analysis of Characteristics of Microservice Architecture

Compared with the traditional single application architecture, the microservice architecture decouples the application system by decomposing functions into discrete services, which has obvious advantages:

- **Atomic service:** The main idea of the microservice framework is to decompose a large overall system into a series of services with a single function without changing the overall function. The more functional each service is, the less dependent it is on other services.

- **Independent deployment:** When making a requirement change for a service that applies a microservice architecture, you can build and redeploy independently without relying on other microservices. Virtual machine containers can be deployed in the cloud, which can improve resource utilization and reduce deployment cost [4].

### Interaction Mechanism of Microservice Based on Virtual Container

The use of container technology can greatly alleviate the problems caused by the microservice architecture. Linux container technology allows different containers to share the same kernel, with complete isolation between containers. Although the microservice architecture brings many advantages, it is not easy to build, deploy, and maintain distributed microservice systems [5]. The lightweight, application-oriented virtualization environment provided by the container provides an ideal carrier for microservices. Similarly, cloud services based on container technology will greatly simplify the entire process of creation, integration, deployment, operation and maintenance of container microservices, thus it can promote the large-scale practice of microservices in the cloud.

Container technology is an ideal solution for encapsulation and cloud deployment of microservices. Running space of the container provides application isolation, each container contains a complete operating environment, and exclusive resources, changes in the container will not affect the operation of the other container, this service encapsulation and operation of microservice architecture provides the carrier, and promote the tiny segment cloud mass practice [6].
Reading Platform Demand Analysis

Starting from the practical application, this platform has realized the functions of e-book online reading, purchasing and community communication. The system is developed by using Spring Cloud framework to improve system scalability and maintenance convenience. Compared with the traditional system architecture, the service architecture is more of an emphasis on autonomous service. Pay attention to the independent development, deployment, and operation and service, effectively reduce the enterprise bus the difficulty coefficient of development work, has the characteristics of flexibility [7].

Demand Positioning

Microservice architecture is designed to address the shortcomings of traditional architecture, such as code redundancy, maintenance difficulties, and new function development difficulties. You also need to meet the needs of the reading platform when building the platform.

Business Functions

According to the design method of the microservice architecture, the functional modules of the reading platform are divided into four microservices:

- **user information management**
  Input the relevant information according to the user registration, including the user's basic personal information, login password, etc.

- **book resource management**
  The publishing house of paper books, book author, introduction in Chinese and English, price of paper books and price of e-books, etc.

- **order management**
  The platform provides online payment and purchase. Users can add books to the shopping cart and make purchases. The system will record the user's order information and provide query function.

- **public information management**
  Users can create topic group, and then through the system recommended show interest in the user page, create a topic to discuss environment, record the user the topic content, users can transfer images.

Reading Platform Framework Design

Reading platform design a total of four modules, each module has a separate database and engineering, and to deploy on a different server or by container technology deployed on the same server, then through the API interface between micro total engineering services in the way of the call. As a microservice platform, the reading platform mainly includes basic service layer, microservice middleware layer and monitoring layer, as shown in figure 5.
In the implementation of the microservice framework, the mature open source framework in the industry was selected to realize the development and operation of microservices through framework integration. Spring Boot provides basic functions such as dependency injection, persistence, logging and caching. Integrated early assembly to achieve service registration discovery; The Ribbon component realizes client load balancing. Service latency and fault tolerance for Hystrix components; Zuul components provide dynamic routing, monitoring, flexibility, security and other edge services. As shows in Figure 6.

**Performance Analysis**

The use of containers to encapsulate microservices can make up for the deficiency of microservices. Below is a performance test of my traditional architecture and microservice architecture.

In test 1, the application system is divided into 4 microservices, and mysql is used for database. Microservices are encapsulated using docker container. The traditional service is deployed to a separate physical machine, and the microservice deploys parts of four microservices to four virtual machines, each configured with 4Core/16GB. Traditional services are deployed to a single server of 16Core/64Gb. As shown in table 1.

<table>
<thead>
<tr>
<th>Service architecture types</th>
<th>Microservice architecture</th>
<th>Traditional SOA architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>database</td>
<td>mysql/100000 data</td>
<td>mysql/100000 data</td>
</tr>
<tr>
<td>Number of microservices</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>The total code</td>
<td>1500000</td>
<td>1500000</td>
</tr>
<tr>
<td>Server configuration</td>
<td>16Core/64GB</td>
<td>16Core/64GB</td>
</tr>
</tbody>
</table>

Test content and result analysis. The resources (not including the database) of traditional architecture and microservice architecture and performance testing are shown in table 2 and table 3 respectively. After being deployed as a container, the microservice package has flexible elastic scaling function and can save a lot of hardware resources. When the system complexity increases, the performance is better. For non-business functions such as log monitoring and performance analysis, public microservices can be developed. Application values are focused on business logic and realize real-time development. The use of microservice architecture and container technology makes agile development and continuous integration possible.
Table 3. Performance tests for traditional and microservice architectures.

<table>
<thead>
<tr>
<th>Traditional architecture</th>
<th>Microservice architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>The maximum amount of concurrency supported is 750</td>
<td>The maximum amount of concurrency supported is 2300, a nearly three-fold increase</td>
</tr>
<tr>
<td>The TPS supported is 185</td>
<td>The TPS supported is 177, which is 95% of the original TPS</td>
</tr>
<tr>
<td>The average virtual machine CPU utilization (4*4Core) is around 35%</td>
<td>The average virtual machine CPU utilization (2*2Core) is around 55%</td>
</tr>
</tbody>
</table>

Test 2 tests the relationship between production efficiency and complexity. Traditional engineering is deployed to a physical machine, and the number of microservices gradually increases from 2 to 4, 6, 8. As the system becomes more and more complex, the production efficiency changes accordingly. Concurrency is limited to 500. For less complex systems, the additional burden of managing microservices reduces productivity. As complexity increases, productivity begins to decline rapidly, and the reduced coupling of microservices reduces productivity. This is shown below.

As the number of system code increasing, only one traditional architectural engineering, system response time increases, if the project into microservice architecture form, shorten response time, performance will be improved, this is the service architecture and traditional architecture in comparison, the most prominent advantages.

Conclusion

The microservice architecture solves the problems of huge projects, complicated code, and difficult update and iteration. It realizes the low coupling, easy expansion and scalability of various services in the platform, and provides convenience for the upgrade and maintenance of the platform. In addition, in addition to reading and communicating information, its distributed design and easy access to cross-service communication via API provide great convenience for extending functions. Microservice architecture for building a fully functional, deployment, simple, easy to maintain the Internet information communication platform provides a new way of thinking, also represents a development trend of engineering for service in the future.

References


