Postprint: Optimization Exploration of a Microservices-Based News Production System

Authors: Han Xiao, Li Jieyuan

Date: 2023-10-08T00:00:00+00:00

Abstract

As Internet technology continues to evolve, the digital transformation across various industries is accelerating. After analyzing the advantages of microservices architecture and the shortcomings of news production systems developed using traditional monolithic architecture, this study optimized and implemented a news production system based on microservices architecture, thereby enhancing system scalability, flexibility, and development efficiency while accumulating valuable experience.

Full Text

Exploring Optimization of News Production Systems Based on Microservices Architecture

Han Xiao, Li Jieyuan (Xinhua News Agency Technology Bureau, Beijing 100803)

Abstract: With the advancement of Internet technology, the Internetization process across industries continues to accelerate. After analyzing the advantages of microservices architecture and the shortcomings of news production systems developed using traditional monolithic architecture, we optimized and implemented a news production system using microservices architecture. This improved system scalability, flexibility, and development efficiency while accumulating valuable experience.

Keywords: Internet technology; Internetization process; microservices architecture; news production system; software development

Chinese Library Classification: TP393
Document Code: A
Article ID: 1671-0134(2021)02-062-02
DOI: 10.19483/j.cnki.11-4653/n.2021.02.016

The news production systems previously developed by our organization predominantly employed monolithic architecture, resulting in large, complex applications that made subsequent agile development and business expansion difficult. Because some modules were not effectively decoupled, they required significant effort for iteration, deployment, and management after system launch, with certain modules even taking up to an hour to compile and deploy. The advantages of microservices architecture over traditional monolithic architecture better align with our organization' s needs for media convergence and intelligent development. Through microservitization, when business functions change, the system can rapidly adjust corresponding services to complete development and testing tasks. Leveraging cloud architecture and platform-based deployment capabilities enables automated deployment and quick adaptation to business changes.

This paper first introduces microservices, then analyzes the current state of traditional software development models in large, complex systems, subsequently elaborates on microservices partitioning, and finally describes the optimization and implementation of the news production system using the Spring Cloud microservices framework.

3.1 Microservices Development Framework Migration

Microservices architecture is a software development framework oriented toward Internet application services, primarily applied to server-side software development for Internet applications, and evolved from Service-Oriented Architecture (SOA). It advocates dividing monolithic applications into a set of small services that coordinate and cooperate with each other [2]. The advantages of microservices architecture include: services are divided into minimal granularities, each application is small with clear boundaries and single responsibilities; each microservice can be developed independently with lower technology stack requirements for development teams, making individual services easier to develop, understand, and maintain; each microservice can be deployed independently, thereby accelerating deployment speed; each microservice can independently choose its technology stack and be updated and extended flexibly. However, microservices applications are distributed systems that increase development and deployment complexity, presenting challenges such as data consistency, service dependencies, and service monitoring.

This paper attempts to transform our organization' s news production system from traditional service architecture to Spring Cloud microservices architecture. The infrastructure migration primarily involves the following changes:

Service Discovery Method Change:
Services are registered, discovered, and consumed through Consul Client communicating with Consul server.

[Figure 1: see original paper]

Interface Definition Method Change:
The web layer's method of calling lower-layer services uses REST-based interface calls. Spring Cloud OpenFeign is integrated to automatically configure and bind remote REST calls, with circuit breakers (Hystrix) preventing cascade failures. This enables transformation by only modifying service interface classes, ensuring minimal changes and impact.

The API interface defined for the new SampleXX service is as follows:

```
@FeignClient(value= "SampleXX ", fallback=ServiceClientHystrix.class)
public interface SampleXX {
    @RequestMapping(value = "/xinhua/getUserInfoByUserName", method = RequestMethod.GET)
    String getUserInfoByUserName(@ResquestBody String jsonStr);
}
```

Service Startup Method Change:
The original web layer services were all deployed in Tomcat containers and started by Tomcat. This has been changed to Spring Boot independent application startup.

3.2 Service Deployment Architecture Design

To facilitate future distributed system deployment and management, a container layer was introduced on top of the virtualization layer. In the deployment architecture design, microservices-dependent components require changes and additions, including: new application gateway service Spring Cloud Gateway, application configuration service Config Server, service registration and discovery Consul Server/Client, circuit breaker monitoring dashboard Hystrix Dashboard, and distributed link monitoring Zipkin Server. The logical deployment architecture after migration is shown below.

[Figure 2: see original paper]

3.3 Application Layer Microservices Design

Application layer microservices are divided into four hierarchical layers: data service layer, data management layer, basic functional component layer, and business service layer. Each layer provides API support for upper-layer services and integrates through the business scenario layer to uniformly serve B/S clients, C/S clients, and mobile terminals.

[Figure 3: see original paper]

Data Service Layer: Integrates resource data from various data sources.
Data Management Layer: Provides unified file management, image management, and primary VMS and other basic management services.
Basic Service Layer: Provides basic services such as authentication, AI services, message engine, streaming media, local authentication and authorization

services, and management configuration services.
Business Service Layer: Provides business services including resource management, resource display, content processing, content workflow and rule control, and content publishing.

These services will no longer be released as a single integrated unit but will be disassembled individually, with each microservice application having its own deployment, resource, scaling, and monitoring requirements.

3.4 Miniaturized Deployment

Based on detailed microservices decomposition, we implemented modular packaging, modular continuous integration, and modular deployment. Using Docker technology enables cross-platform rapid deployment and startup of basic applications and services. Kubernetes is used for unified management of Docker containers to achieve high load, high availability, and elastic scaling.

The deployment process is divided into preparation and deployment phases. The preparation phase enables module selection functionality. The deployment phase provides module deployment replica quantity and other basic configuration functions, as well as multi-domain access configuration functionality, with one-click deployment provided after configuration. This deployment process simplifies miniaturized deployment operations and achieves maintainability.

[Figure 4: see original paper]

System architecture design is crucial during the construction process, requiring balance among system availability, scalability, maintainability, reliability, and high performance, while also considering business functional requirements. Therefore, architecture design must be clarified during the initial project construction phase. To support continuous business expansion and functional additions, we attempted to optimize the business system' s architecture design based on microservices. The design scheme elaborated in this paper has been implemented in actual work. Subsequent agile development work on this architecture can quickly respond to business adjustments and requirement changes, improving system applicability and scalability. Our team will continue to summarize experience, strengthen reflection, and explore further to gain relevant insights.

References: [1] Wu Wenjun, Yu Xin, Pu Yanjun, Wang Qunbo, Yu Xiaoming. Complex Software Development in the Microservices Era [J]. Computer Science, 2020(12): 11-17.
[2] Zhao Ran, Zhu Xiaoyong. Review of Microservices Architecture [J]. Network New Media Technology, 2019(1): 58-59.

Author Information:
Han Xiao (1985-), female, Beijing, Senior Engineer at Xinhua News Agency Communication Technology Bureau, research direction: news production system construction and development.

Li Jieyuan (1983-), male, Shanxi, Senior Engineer at Xinhua News Agency Communication Technology Bureau, research direction: project management.

(Responsible Editor: Zhang Xiaojing)

Note: Figure translations are in progress. See original paper for figures.