The design and implementation of a campus web information system based on micro-service architecture

Yifei Gong¹, Kengbin Chen², Feng Gu³ and Fei Wang⁴

¹Graduate School and University Center, City University of New York, New York, United States
²Collage of Information Technology, Beijing Normal University Zhuhai, Zhuhai, China
³College of Staten Island, City University of New York, New York, United States
⁴Collage of Information Technology, Beijing Normal University Zhuhai, Zhuhai, China

Corresponding author and e-mail: Fei Wang, wangfeizhuhai@aliyun.com

Abstract. In the development of Web information system, the traditional monolithic application architecture is simple and effective for small-scale projects. Because all business logics are combined together in those projects, secondary development and maintenance become very difficult. In recent years, there are methods to solve this problem, which are popular Web distributed architecture solution. The first step of the solution is to separate front-end and back-end of Web system and the second step is to decompose system’s services into micro-services. Micro-service architecture splits application into smaller scale services and each service completes only performs special function. Through service registration centre, micro-service architecture provides unified mechanism of service discovery and registration, it provides unified interfaces for calls from front-end through gateway technology; it protects stability and fault tolerance via back-off mechanism and circuit-breaker mechanism. We have developed a campus Web information system ,which adopts two techniques, the separation of front-end and back-end and micro-service architecture By using Spring Cloud framework as back-end developing architecture, the solution of splitting application to micro-services is obtained, and then problems in monolithic application architecture are solved. The high reusability of back-end services is realized and the advantages of micro-service architecture are tested in our system.

1. Introduction
Along with high developing of Internet and cloud computing, people are more and more dependent on data service. The former monolithic application architecture and service-oriented application architecture can’t meet business needs gradually. The rise of distributed micro service architecture is an inevitable trend of software application architecture which towards flexible, low coupling, high extensible, and dynamic scaling in the future. In monolithic application architecture, all business logics are written in a single application, resulting in excessive coupling between services, which is not conducive to secondary development and maintenance of the system. For each system’s level
In recent years, micro-service technology has rapidly become a hot topic in the field of software architecture. The theory of micro-service architecture is constantly improved, and the implementation technology is continually emerging. The proposal and application of micro-service architecture solution and the separation of front-end and back-end technology [1-2], which are led by technology giant enterprises, are all new solutions for the back-end development and system architecture design.

The separation of front-end and back-end becomes possible from theory and enables decoupling of business logic. By a unified, stateless, and encrypted RESTful API, the interactions of front-end and back-end complete data requests. Decoupling of different services reduces maintenance costs, improves development efficiency, and greatly improves the user experience.

A campus Web information system described in this article separates the front-end and back-end of it by defining RESTful style interfaces. Spring Cloud framework is used as micro-service architecture for back-end which solves the problem of monolithic application architecture and realizes the high reuse of back-end services.

2. The technologies of Micro-Service and separation of Front-End and Back-End

2.1. The Significance of the Separation of Front-End and Back-End

In the pattern of un-separation of front-end and back-end, front-end pages are controlled, rendered, or redirected by back-end. Because back-end controls front-end display, the coupling degree of two ends is very high [3-4]. In the pattern of separation of front-end and back-end, the back-end only returns the data required by front-end no longer rendering HTML pages and controlling the effect of front end. Back-end provides data via asynchronous interface (AJAX).

2.2. The Technology of Micro-Service

For a project developed in Java, a monolithic application means a single Jar or War package, which contains all the content of the entire project, including front-end pages and back-end business logics. Because it is a single package deployment, it is called as monolithic application architecture. There are several problems with monolithic architecture system:

- Low development efficiency and difficult code maintenance: because of high coupling between project modules, when a function needs modified, it can relate to or affect other functions.
- Low deployment efficiency: monolithic application needs deployed as a whole, and to modify small function of the system will result in redeployment.
- Low reliability: because all modules are running in the same process, the failure of any module can cause the entire system to crash.
- Low expansion capacity: because of the high degree of coupling between modules, massively changing project are required when adding new functions to system.

For the problem of monolithic application architecture, a solution of micro-service architecture is proposed. The highly coupled functions of the system are decomposed into discrete micro-service to realize the decoupling of the corresponding system. Each service communicates through a RESTful API, as shown in Figure 1.

![Figure 1. The Architecture of Micro-Service.](image-url)
Each micro-service is developed according different business logic and meanwhile there are
dependent service relationships among micro-services which can independently provide service
outward and inward so that there are associated dependencies among them. However, under the
management of the service registry and discovery center, a unified architectural pattern is still
presented externally. The advantages of micro-service are shown as bellow:

- Because micro-service technique decomposes the huge monolithic application into several
  small services, each service has clear function, small amount of code, and relatively simple
development and maintenance.
- Each service can be deployed independently. If requirement of a service is changed, this single
  service needs to be re-coded and deployed without affecting the use of other services.
- Each service extends independently, so that a project can implement fine-grained extensions
  based on new requirement.

2.3. Decomposing the Monolithic Design into Micro-Service Design
In the micro-service architecture, the services in monolithic architecture are decomposed into small
size modules, which are independent and low coupling and can independently provide service outward
and inward so that there are associated dependencies among them [5-7].

In our campus information system [8], there is a teaching affair administration module and the
following content illustrate how to design micro-service in this module whose functions are as follow
and shown in Figure 2:

- Data collection: get data periodically, including new students information.
- Data updating: store new student’s information to database, such as user information
  modification and semester grades.
- Data query: according user query return corresponding data, such as grades or personal
  information.
- Data presentation: display data obtained from last step on the page

Functions as mentioned above are coupling, easy and effective in monolithic application. But if
“data updating” service collapse and many users are using “data query” at the same time, following
problems will appear:

- Data query service is disabled or the collapse of data updating leads to system snow slide.
- Dirty reading of data in the process of data reading

In order to ensure the correctness and instantaneous of data, high availability of services, and
stability of system, the separation of front-end and back-end is needed and services are designed and
implemented by micro-service architecture, as shown in Figure 3.

Figure 2. The monolithic Architecture.  
Figure 3. Micro-service Architecture in Web projects.

Under the design principles of micro-service and the separation of front-end and back-end, services
and pages display are separated so that pages are real time visibility without invisibility owing to
system collapse. Even if in the case of system collapse, prompt message can be returned to users,
which embody system fault-tolerant ability. As a result of the separation of front-end and back-end,
the effectiveness of data and services can be guaranteed. Every function runs independently as a single
module, i.e. micro-service, so it has some advantages as bellow:
If the function of data collection is collapse, the updating for the data obtained last time will not be influenced. In the process of data updating, data collection and data query service for users can be executed at the same time. When data query service is collapse, the back-end still maintains data validity and instantaneity.

By using roll back technique, current layer notices its upper and lower layer that its service is disabled, which can’t influence the ability of upper and lower layer for data collection, data updating, and data display. In this micro-service architecture, each service is single application and runs in a independent process. The center of service registration and discovery provides unified management for registration and discovery. Gateway technique provides unified interface outward. Rollback and circuit breaker protect stability and fault tolerance of system.

3. Key technologies of spring cloud Micro-Service architecture

3.1. Service Registration Center Based on Eureka
Eureka component provided by the Spring Cloud micro-service solution is adopted as the service registration center of the entire micro-service system and each service registers its own service to Eureka including its host address, port number, service name, and communication protocol, etc. Eureka manages these services and uses the heartbeat mechanism to detect if the service is available, so that to achieve high availability and troubleshooting in system.

3.2. Token management based on OAuth2
In order to solve the complexity of maintaining user’s status in session-cookie, our project adopts token instead of traditional technique to maintain user’s status. A token is a string generated by server as the unique identity of a client request and also is unique identification in back-end. By using a token, the user's status are stored on the server side and implemented by caching mechanism, which effectively solve the problem of traditional user status recognition. Because token can be stored in memory or persistent databases, the availability of recognition is guaranteed.

Under OAuth2 standard, resource owner, resource server, authorization server, and role of client are defined. Through the request of different roles, a client authorization authentication be completed and the user's status information is kept in memory, as shown in Figure 4. The system records and authenticates users by issuing tokens to them and which are stored in memory or in Redis database.

3.3. Circuit breaker based on Netflix Hystrix
On highly concurrent requests, a service is not guaranteed to be 100% high available. If a service has a problem or is blocked, it may cause blocking between services, fault propagation, and even if the failure of the whole system that is avalanche effect. To solve this problem, we use Netflix Hystrix component provided by Spring Cloud micro-service solution.

Hystrix sets a series of thresholds by configuration and count failure times, timeout times, within the prescribed limits for a particular service. If a service exceeds the threshold, it will be automatically fused by Hystrix, which isolates dependencies with other services and protects the availability of the entire system. Meanwhile this service automatically calls the downgrade handler or adopts the fallback mechanism and after a period of time, tries to recover again automatically.

3.4. Routing gateway technique
The purpose of routing gateway is to provide only one external interface for all micro-services and routing gateway proxies requests to different services so that front-end only needs to access one gateway address. If there is no routing gateway, multiple services provide multiple call addresses to the front-end, which increases the complexity of client. Zuul component of Spring Cloud manages micro-service requests and dynamically loads, compiles, and run requests for micro-service
The advantage of gateway module is that for all micro-service under the micro-service architecture, the interface, port number, and routing information are exposed uniformly to the outside, so as to effectively guarantee the security of micro-service. Gateway can be unified to achieve load balancing, circuit breaker mechanism, rate adjustment and other common configuration settings, as shown in Figure 5.

![Gateway Module in Information System](image)

**Figure 5.** Gateway Module in Information System.

### 4. Design and implementation of our campus information system

The front-end implementation of our campus information system is described in detail in reference [8]. The back-end of our campus information system is realized by micro-service architecture. Each service on the server side is separated and decoupled so that services are not affected by each other, and hot plugging can be supported. At the initial stage of system startup, all micro-services are registered in the registration center to realize mutual service discovery, including the technical common module and the business module. Each module belongs to an independent micro-service. Eureka component maintains the availability of each service through heartbeat mechanism.

Through the authorization management of the OAuth2 component, the user obtains the authenticated unique identification token. A user uses a token to send a request to the system gateway to retrieve some data which is provided by a micro-service. The gateway of the system retrieves service information corresponding to the micro-service from the registration center and the request is forwarded to the micro-service through the port number and the unique identification of the micro-service. When a micro-service gets data, it will be stored in cache layer, sent to control layer, and finally displayed on page, as shown in Figure 6.

![The Back-End Architecture of Our Campus Information System](image)

**Figure 6.** The Back-End Architecture of Our Campus Information System.
4.1. The implementation of service registration center

Micro-service architecture model needs to provide a uniform registry and discovery center among services as well as between services and the outside services, which allows services to discover, registry, and call to each other internally and externally. We built registration center using Eureka component provided by Spring Cloud framework. Eureka integrates Spring Boot’s automatic configuration loading mechanism and effectively provides customizable and highly available configuration. Ultimately, a registration and discovery center is provided to the producers and consumers of a service. A service declares its information and service it provides to the Eureka and informs the connection type. Eureka will detect the status of the service according to the heartbeat mechanism through the information it provides. The configuration information of Eureka in application. The yml file is as follows:

```
spring:
  application:
    name: register-service

eureka:
  instance:
    hostname: localhost
    prefer-ip-address: true
  lease-expiration-duration-in-seconds: 30
  lease-renewal-interval-in-seconds: 30

server:
  port: 8760
```

In the configuration information above, the key descriptions are as follows:
- `eureka.client.service-url.defaultZone`: the URL address at which the service is registered.
- `spring.application.name`: unique identity for each micro-service, used for identification between services, primarily for registry and gateway configuration.
- `eureka.instance.prefer-ip-address`: supporting for distributed deployment, in different ip domains, the prefix ip address recognized by Eureka.
- `eureka.instance.lease*`: controlling the registration center to detect the heartbeat mechanism. According to the needs of the service architecture, the specific heartbeat detection time is adjusted to avoid that it is too long to cause the service to fail or too short to cause network to be instability, which results the service was incorrectly removed by the registration center.

4.2. The implementation of the gateway

The gateway module is implemented by Netflix Zuul component of Spring Cloud. Zuul filters requests using the filter mode, identifies, and transmits service according to its name in “spring.application.name”. Several parts of configuration information of gateway in application.yml file are as follow:

```
spring:
  application:
    name: campus-gateway-service

zuul:
  routes:
    uaa:
      path: /uaa/**
      sensitiveHeaders:
      serviceId: campus-auth-service
      exsapi:
```

......
4.3. The implementation of circuit breaker

Netflix Hystrix component provides a layer of defense mechanism for the system. During the actual run of the system, it is inevitable that there will be abnormal fluctuations caused by the call failure, timeout, exception, services temporarily unavailable, services maintenance, and unstable network. In these cases, there is still a need to protect the high availability of the system from a single failure or an exception which lead to a cascading crash of overall system services, a common avalanche effect at the system level. Hystrix effectively solve the above problems through mechanisms such as service isolation, degradation callback, and timeout retry.

With the assistance of the Hystrix, the system can quickly implement the circuit breaker by means of simple annotations and configuration file. Hystrix opens the corresponding annotation in the BnuzAppZuulGatewayApplication.java file. Performance optimization is performed by combining configuration information provided by Hystrix with the actual server scenario, meanwhile fallback processing is performed for failed requests.

5. Conclusions

Using Spring Cloud framework as back-end developing architecture, the solution of splitting application to micro-services is realized so that problems in monolithic application architecture are solved. The high reusability of back-end services is realized and the advantages of micro-service architecture are tested in our system. Through developing process for our system, we accumulate more experience. With the extension and development of the project requirement, more iteration will be done in future.

References

[1] Wang Jian and Luo Zheng 2020 China. Design and Implementation of WEB Project Front and Back Separation. Software Engineer, vol 23(4), p 22.
[2] Du Yanmei and Huang Xiaofang 2018 China. Frontend-Backend Separation Development Model and Practice for Enterprise Level Web Applications. Journal of Southwest University of Science and Technology, vol 33(2), p 83.
[3] Meng Xiangshuang 2019 China. Application Development Research of Front-end and Back-end Separated WEB. Electronic Component and Information Technology, vol 3(6), p 40.
[4] Zhang Zhipeng and Huang sujuan 2019 China. Design and Implementation of Single Page APP Based on React Technique. Microcomputer Applications, vol 35(10), p 71.
[5] Wu Xia, Ai Fangjiu, 2020 China. Crowdfunding system based on microservice architecture. Journal of Hubei University(Natural Science Edition) , vol 42(1), p 103.
[6] Luo Qinkai and Ni Chengzhang 2019 China. Application of Workflow Technology Based on Micro-service in Cloud Management Platform. Computer Technology and Development, vol 29(9), p 122.
[7] Chen Shiding and Liu Xiang 2019 China. Design method of early warning assistant decision system for micro-service architecture”. Journal of Nanjing University of Information Science & Technology, vol 11(5), p 609.
[8] Yifei Gong, Kengbin Chen, and Feng Gu, 2020 China. The Architecture of Micro-services and the Separation of Frond-end and Back-end Applied in a Campus Information System, 3rd International Conference on Computational Intelligence in Data Science, (page unknow).